

# The Journal

Volume 8/2, December 2016

---

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

## Contents

Editorial . . . . .	4
---------------------	---

### Contributed Research Articles

<b>multipleNCC</b> : Inverse Probability Weighting of Nested Case-Control Data . . . . .	5
<b>QPot</b> : An R Package for Stochastic Differential Equation Quasi-Potential Analysis . .	19
Design of the <b>TRONCO</b> BioConductor Package for TRanslational ONCOlogy . . . . .	39
<b>diverse</b> : an R Package to Analyze Diversity in Complex Systems . . . . .	60
Simulating Correlated Binary and Multinomial Responses under Marginal Model Specification: The <b>SimCorMultRes</b> Package . . . . .	79
<b>eiCompare</b> : Comparing Ecological Inference Estimates across EI and EI:R×C . . . . .	92
<b>rnrfa</b> : An R package to Retrieve, Filter and Visualize Data from the UK National River Flow Archive . . . . .	102
<b>Qtools</b> : A Collection of Models and Tools for Quantile Inference . . . . .	117
Two-Tier Latent Class IRT Models in R . . . . .	139
Variants of Simple Correspondence Analysis. . . . .	167
<b>hdm</b> : High-Dimensional Metrics . . . . .	185
Normal Tolerance Interval Procedures in the <b>tolerance</b> Package . . . . .	200
<b>easyROC</b> : An Interactive Web-tool for ROC Curve Analysis Using R Language Environment . . . . .	213
<b>tigris</b> : An R Package to Access and Work with Geographic Data from the US Census Bureau . . . . .	231
Escape from Boxland . . . . .	243
<b>Ake</b> : An R Package for Discrete and Continuous Associated Kernel Estimations . .	258
An Introduction to Principal Surrogate Evaluation with the <b>pseval</b> Package . . . . .	277
Calculating Biological Module Enrichment or Depletion and Visualizing Data on Large-scale Molecular Maps with <b>ACSNMineR</b> and <b>RNaviCell</b> Packages . . . . .	293
Subgroup Discovery with Evolutionary Fuzzy Systems in R: the <b>SDEFSR</b> Package .	307
<b>dCovTS</b> : Distance Covariance/Correlation for Time Series . . . . .	324
<b>comf</b> : An R Package for Thermal Comfort Studies . . . . .	341
<b>water</b> : Tools and Functions to Estimate Actual Evapotranspiration Using Land Surface Energy Balance Models in R . . . . .	352

---

<b>quantreg.nonpar:</b> An R Package for Performing Nonparametric Series Quantile Regression. . . . .	371
<b>nmfgpu4R:</b> GPU-Accelerated Computation of the Non-Negative Matrix Factorization (NMF) Using CUDA Capable Hardware . . . . .	383
Computing Pareto Frontiers and Database Preferences with the <b>rPref</b> Package . . . . .	394
<b>micompr:</b> An R Package for Multivariate Independent Comparison of Observations .	406
<b>mixtox:</b> An R Package for Mixture Toxicity Assessment . . . . .	422
Weighted Distance Based Discriminant Analysis: The R Package <b>WeDiBaDis</b> . . . . .	435
Distance Measures for Time Series in R: the <b>TSdist</b> Package . . . . .	455
<b>condSURV:</b> An R Package for the Estimation of the Conditional Survival Function for Ordered Multivariate Failure Time Data. . . . .	464
<b>ggfortify:</b> Unified Interface to Visualize Statistical Results of Popular R Packages. .	478
Measurement units in R . . . . .	490
<b>mctest:</b> An R Package for Detection of Collinearity among Regressors. . . . .	499

## News and Notes

R Foundation News . . . . .	510
Changes on CRAN . . . . .	511
News from the Bioconductor Project . . . . .	514
Changes in R . . . . .	515

The R Journal is a peer-reviewed publication of the R Foundation for Statistical Computing. Communications regarding this publication should be addressed to the editors. All articles are licensed under the Creative Commons Attribution 3.0 Unported license (CC BY 3.0, <http://creativecommons.org/licenses/by/3.0/>).

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

**Editor-in-Chief:**  
Michael Lawrence

**Editorial Board:**  
Bettina Grün, Roger Bivand and John Verzani

**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 Michael Lawrence

On behalf of the editorial board, I am pleased to publish Volume 8, Issue 2 of the R Journal. This issue contains 33 contributed research articles. Each of them either presents an R package, a specific extension of an R package or applications using R packages available from the Comprehensive R Archive Network (CRAN, <http://CRAN.R-project.org>). This issue highlights the breadth and depth of the R package ecosystem, covering advances in statistical computing and visualization, as well as novel applications of R in specific domains. The authors have described a small but representative sample of the now more than 11000 packages distributed through CRAN and Bioconductor.

As usual the bulk of this issue presents advancements in the field of applied statistics, including **multipleNCC** for inverse probability weighting of nested case-control data, **SimCorMultRes** for simulating correlated categorical responses, **Qtools** for quantile inference, and **MLCIRTwithin** for discovering latent traits in questionnaire responses. The **CAVariants** package implements multiple methods for correspondence analysis, and **hdm** provides tools for computing uncertainty in high-dimensional, sparse models. There are articles describing how to analyze normal tolerance intervals with the **tolerance** package, perform associated kernel estimation using **ake**, evaluate principal surrogates with **pseval**, find subgroups using evolutionary fuzzy methods implemented in **SDEFSR**, and use the distance covariance function to analyze time series data with **dCovTS**. Further articles describe **quantreg.nonpar** for quantile regression with non-parametric series, **micompr** for multivariate independent comparison of observations, **WeDiBaDis** for weighted discriminant analysis, **TSDist** for computing distances for time series, **condSURV** for estimating conditional survival functions, and **mctest** for testing collinearity between regressors.

We are fortunate to present a number of data visualization packages including: **rnrfa** for viewing data from the UK National River Flow Archive, **easyROC**, a GUI for analyzing ROC curves, **geozoo** for generating libraries of high-dimensional shapes, and **ggfortify** for getting data into shape for plotting.

Researchers continue to find new ways to apply R to scientific pursuits, including **QPot** for understanding how stochasticity affects systems of differential equations, **nmfgpu4R** for large scale non-negative matrix factorization (NMF) using GPUs, and the **units** package for computing on scientific units. Applications to biology include **TRONCO** for modeling tumor progression and **ACSNMiner** for detecting module enrichment and depletion. Other applications include **diverse** for analyzing diversity in complex systems, **comf** for analyzing thermal comfort data, **water** for estimating evapotranspiration from satellite images, **eiCompare** for comparing ecological inference estimates, particularly in the context of analyzing voting patterns, **mixtox** for assessing the toxicity of chemical mixtures, **tigris** for accessing geographic data from the US Census, and **rPref** for computing Pareto frontiers, useful for implementing preference-based database queries.

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

I hope you enjoy the issue.

*Michael Lawrence*

[Michael.Lawrence@r-project.org](mailto:Michael.Lawrence@r-project.org)

# multipleNCC: Inverse Probability Weighting of Nested Case-Control Data

by Nathalie C. Støer and Sven Ove Samuelsen

**Abstract** Reuse of controls from nested case-control designs can increase efficiency in many situations, for instance with competing risks or in other multiple endpoints situations. The matching between cases and controls must be broken when controls are to be used for other endpoints. A weighted analysis can then be performed to take care of the biased sampling from the cohort. We present the R package **multipleNCC** for reuse of controls in nested case-control studies by inverse probability weighting of the partial likelihood. The package handles right-censored, left-truncated and additionally matched data, and varying numbers of sampled controls and the whole analysis is carried out using one simple command. Four weight estimators are presented and variance estimation is explained. The package is illustrated by analyzing health survey data from three counties in Norway for two causes of death: cardiovascular disease and death from alcohol abuse, liver disease, and accidents and violence. The data set is included in the package.

## Introduction

The nested case-control (NCC) design (Thomas, 1977) (incidence density sampling/risk set sampling) is popular within epidemiology due to its cost-effectiveness and efficiency. At each event time,  $m$  controls are sampled from the subjects at risk. The controls must be event-free at the time their case experienced the event. They might also be matched to the cases on additional factors. Covariates are then obtained for cases and sampled controls.

The analysis of nested case-control data has traditionally been carried out using stratified Cox-regression, where the stratification is with respect to matched case-control sets. That method, however, does not allow for breaking the matching, i.e., analyze the data without directly considering the matched sets. Hence, the sampled controls cannot be used for other cases than they originally were sampled for. In many situations one may want to reuse the controls, for instance when analyzing a subset of the original cases or when there exist more than one endpoint of interest. A few examples of such studies are Hultman et al. (1999); Parsonnet et al. (1991); Floderus et al. (1993); Øyen et al. (1997); Tynes and Haldorsen (1997); Hankinson et al. (1998); Grimsrud et al. (2002); Levine et al. (2004); Clendenen et al. (2011); Meyer et al. (2013).

We assume a competing risks situation in this paper for ease of presentation. Thus there are different types of endpoints, and the subjects can experience at most one of them. A special instance of this is a setting with only one type of endpoint, the single event situation is therefore covered by the competing risks situation. The R package **multipleNCC** (Støer and Samuelsen, 2016) is built with competing risks in mind, however it is not limited to such situations. Even though more complex event history settings would often call for more advanced multi-state modelling, reuse of controls may be handled by using **multipleNCC** together with `coxph()` from package **survival** (Therneau, 2016; Therneau and Grambsch, 2000), see Section [The R package multipleNCC](#).

A method for breaking the matching in NCC designs was introduced by Samuelsen (1997) and further studied by Chen (2001); Samuelsen et al. (2007); Saarela et al. (2008); Salim et al. (2009); Cai and Zheng (2012); Salim et al. (2012); Støer and Samuelsen (2012, 2013); Støer et al. (2014). This method bases the estimation on a weighted partial likelihood, thus weighted Cox-regressions are carried out. The weights are inverse sampling probabilities, which must be estimated from the data, and different estimators have been suggested.

Even though it is fairly easy to estimate the weights, it is an extra step in the analysis. Having a more automatic estimation procedure, i.e., a one-line call in R with similar syntax as `coxph()` will make this way of analyzing NCC data more generally available.

We present the R package **multipleNCC** in this paper. The function `wpl()` estimates weights and carries out weighted Cox-regressions. The users can choose between four options for weight estimation. The function handles both right-censored and left-truncated data and has some possibilities for variance estimation, apart from robust variances. Additional matching is incorporated for three of the four weight estimators, and varying number of controls for all four. This is the first statistical software that performs inverse probability weighting (IPW) aimed at NCC data.

The outline of the paper is as follows; we introduce the general framework of inverse probability weighting for nested case-control data in Section [Inverse probability weighting in NCC](#). Then Sections [Weight estimation](#) and [Variance estimation](#) follow. The package is described in Section [The R package multipleNCC](#) and illustrated in Section [Analysis of example data set](#). A comparison between

the traditional estimator and the IPW estimators is given in Section [Comparison between stratified coxph and wpl](#), followed by Section [Discussion](#).

## Inverse probability weighting in NCC

We have a cohort consisting of  $n$  subjects, where the  $i$ -th individual is followed from the left-truncation time  $l_i$ , which may be zero, to time of event  $\tilde{t}_i$  or time of censoring  $c_i$ . Thus the cohort members are followed from  $l_i$  to  $t_i = \min(\tilde{t}_i, c_i)$ . There are  $K$  competing endpoints, and each subject can at most experience one of them.

At each event time,  $m$  controls are sampled for the case experiencing the event and  $m = m(t)$  may depend on time. Finally, let us introduce what we call sampling-status indicator  $S_i$ , which is required input to `wpl()`. This indicator takes values in  $\{0, 1, \dots, K, K+1\}$ , zero indicates a non-sampled subject in the cohort, 1 indicates sampled controls (and not cases) for any of the endpoints in question, while “ $2, \dots, K+1$ ” indicate cases of one of  $K$  types.

The NCC design is tightly connected to the Cox proportional hazards model and our model for endpoint  $k$  can be written as

$$h_{ki}(t|x_i, z_i) = h_{k0}(t) \exp(\beta_k' x_i + \gamma_k' z_i). \quad (1)$$

Here  $h_{k0}(t)$  is the baseline hazard for endpoint  $k$ ,  $x_i$  are covariates and confounders while  $z_i$  are additional matching variables (additional to matching on time), if additional matching has been carried out, otherwise  $z_i$  will be zero. The  $\beta_k$  and  $\gamma_k$  are the log-hazard ratios connected to  $x$  and  $z$  for the  $k$ -th endpoint.

Since the matching is broken with IPW, it will generally be important to adjust for the matching variables ([Støer and Samuelsen, 2013](#)). They are included as linear functions in Equation 1 for simplicity, however more general models with other types of functions are possible.

The controls in a NCC design are matched to the cases on at risk status and possibly additional factors. Due to this matching, it is not straightforward to reuse the controls for other endpoints/cases since the matching must be broken. [Samuelsen \(1997\)](#) suggested a weighted partial likelihood which resembles the standard Cox-likelihood

$$L_k(\beta, \gamma) = \prod_j \frac{\exp(\beta_k' x_j + \gamma_k' z_j)}{\sum_{i \in \mathcal{R}_j} \exp(\beta_k' x_i + \gamma_k' z_i) w_i}. \quad (2)$$

This likelihood enables the controls to be used for other endpoints and [Saarela et al. \(2008\)](#) and [Salim et al. \(2009\)](#) were the first to discuss this likelihood in connection to competing risks. The product is over all cases of type  $k$ , while the sum is over a set  $\mathcal{R}_j$ , defined as all cases (of all types) and all controls at risk at  $t_j$ . The weight,  $w_i = 1/p_i$ , is the inverse probability that individual  $i$  is ever being sampled. This probability will be 1 for cases since all of them are sampled by design, and it must be estimated from the data for the controls. We assume time invariant covariates, although time-dependent covariates are in theory possible as long as they are known at all event times at which the subject is at risk. This has, however, not yet been implemented in package `multipleNCC`.

The fundamental idea behind inverse probability weighting is to adjust for the biased sample from the cohort. The sample is biased, first and foremost, with respect to the proportion of cases and controls, but with additional matching it can also be biased with respect to matching variables. The idea is then to let each control represent a number of subjects in the cohort by giving them weights larger than 1. The less probable it was for a given subject to be sampled, the more subjects in the cohort it should represent since that “type” of controls likely are under-represented in the NCC sample. By using inverse sampling probabilities as weights, this is accomplished. The analysis is then carried out “as if the data were from a cohort study”, thus by a weighted Cox-regression. This idea was first proposed by [Hansen and Hurwitz \(1943\)](#) with a survey sampling perspective for sampling with replacement, and later generalised to sampling without replacement by [Horvitz and Thompson \(1952\)](#). Inverse probability weighting is also commonly used in the context of missing data ([Robins et al., 1994](#)).

## Additional matching

To increase efficiency and adjust for confounding, the controls in a nested case-control design are often matched on additional factors than at risk status. This can for instance be year of birth, sex or years since first employment. We divide such matching into two groups: category matching and caliper matching ([Cochran and Rubin, 1973](#)).

With category matching, the controls are required to have the same value on the matching variable as the case, and the matching variable is often a categorical covariate. As an example, the controls can be matched to the cases on sex and a male case is then required to have a male control. With caliper matching, the matching variable is typically continuous and the control's value of the matching variable must lie within a specified interval around the case's value. For instance the controls could be matched on year of birth plus/minus 2 years, i.e., the birth year of a control must be within two years of the case's birth year.

## Weight estimation

The weights in Equation 2 must be estimated from the data at hand, and three types of estimators have been considered: Kaplan-Meier (KM) type of weights (Samuelson, 1997; Salim et al., 2009; Cai and Zheng, 2012), more model based logistic regression type (Mark and Katki, 2006; Samuelson et al., 2007; Saarela et al., 2008; Stør and Samuelson, 2013) and local averaging (Chen, 2001), referred to as Chen-weights.

### KM type of weights

The Kaplan-Meier type of estimator without additional matching can be formulated as

$$p_i = 1 - \prod_{l_i < t_j < t_i} \left\{ 1 - \frac{m}{n(t_j)} \right\}. \quad (3)$$

Here  $n(t_j)$  is the number at risk in the cohort at time  $t_j$  and  $m$  the number of sampled controls per case. The estimator in Equation 3 resembles the Kaplan-Meier estimator. The KM weights can be generalized to situations with additional matching by taking the product only over subjects that meet the matching criteria, and letting the denominator only consist of subjects at risk that meet the matching criteria. Let  $n_j(t_j)$  count the subjects at risk at time  $t_j$  who meet the matching criteria of case  $j$ . Then the formula with additional matching can be expressed as

$$p_i = 1 - \prod_j \left\{ 1 - \frac{m}{n_j(t_j)} I(\text{Control } i \text{ could be sampled for case } j) \right\}. \quad (4)$$

By replacing  $m$  with  $m(t_j)$ , where  $m(t_j)$  is the number of sampled controls for the case at  $t_j$ , the situation with varying number of controls is covered.

### Logistic regression weights

A more model based approach is to use logistic regression models, either traditional logistic regression or the more flexible generalized additive model (GAM; Hastie and Tibshirani, 2009, Chap. 9) with logit-link. The sampling indicator,  $O_i$ , is used as outcome and the left-truncation time and censoring time as covariates with  $\xi$  being an intercept term:

$$p_i = \mathbb{E}[O_i | t_i, l_i] = \frac{\exp(\xi + f(t_i, l_i))}{1 + \exp(\xi + f(t_i, l_i))}. \quad (5)$$

It is important to note that this regression is carried out on the cohort excluding cases of all types. The reason for this is that all cases are sampled with a known probability of 1, thus including them in the regression would interfere with the estimation of the sampling probabilities for the controls.

With  $f(t_i, l_i) = f_1(t_i) + f_2(l_i)$  where  $f(\cdot)$  are linear functions, the estimator in Equation 5 is the traditional logistic regression model, and the inverse of those probabilities are referred to as GLM-weights. When  $f(\cdot)$  are smooth functions, the result is a GAM-model. In situations with additional matching, the matching variables should also be included in the regression model. Category matched variables are included as categorical factors while caliper matched variables are included as continuous covariates with GLM-weights and as smoothed functions with GAM-weights.

The number of controls for each case is not an explicit part of the estimation procedure for logistic regression weights and therefore no extra care must be taken in situations with time- and endpoint-dependent number of controls.

### Local averaging weights

Before we can introduce the Chen-weights method (Chen, 2001), which is also known as local averaging, we need some additional notation. Let  $0 = l^0 < l^1 < \dots < l^A$  be a partition of the range of left-truncation times and  $0 = t^0 < t^1 < \dots < t^B$  a partition of the range of follow-up times where  $t^A$  and  $t^B$  are the upper limit of the left-truncation times and censoring times respectively. We also define  $\mathcal{J}_a = (l^{a-1}, l^a]$  and  $\mathcal{I}_b = (t^{b-1}, t^b]$ . The intervals in each direction are taken to be of the same length in `wpl()`, however the interval length can in principle vary. The local averaging weights can then be expressed as

$$w_{ab} = \frac{\sum_{i=1}^n I(l_i \in \mathcal{J}_a, t_i \in \mathcal{I}_b, i \text{ is not a case})}{\sum_{i=1}^n I(l_i \in \mathcal{J}_a, t_i \in \mathcal{I}_b, i \text{ is a sampled control and not a case})}.$$

All controls included in the study in  $\mathcal{J}_a$  with a censoring time in  $\mathcal{I}_b$  are given weight  $w_{ab}$ . Hence, all subjects sampled within the same combination of intervals will be given the same weight. Samuelsen et al. (2007) noted that this amounts to post-stratifying on grouped left-truncation and censoring times.

Generalizing these weights to handle additional matching would require partitioning of the matching variables in addition to the range of left-truncation and right-censoring times, and this will introduce a large number of combination of intervals. Due to this, the weights will likely be unstable since a small number of subjects would belong to each group. The Chen-weights are therefore only implemented for situations without additional matching.

As with logistic regression weights, the number of controls per case is not an explicit part of the estimation procedure for local averaging. Situations with time- and endpoint-dependent number of controls are therefore handled with no modification of the estimator.

### Variance estimation

Since the subjects enter the weighted partial likelihood (Equation 2) whenever they are at risk, the likelihood contributions are not independent and the variance estimation cannot be based on the inverse of the information matrix only. A simple, yet sometimes conservative solution (Cai and Zheng, 2012), is to use robust variances (Lin and Wei, 1989; Barlow, 1994). This is the default option in `wpl()`. A variance estimator for the KM-weights can be found in Samuelsen (1997) when there is no additional matching. A variance estimator for Chen-weights is given in Chen (2001), but since Samuelsen et al. (2007) argues that this estimator can be considered as a post-stratified case-cohort estimator, one may apply the variance estimator of Borgan et al. (2000). We have implemented the variance estimator of Samuelsen (1997), which exactly accounts for the procedure used to estimate the weights, and extended it to allow for additional matching, see details below and web-appendix to Cai and Zheng (2012). For the Chen-weights, we have implemented the variance estimator based on post-stratification, but only without additional matching since, as discussed in Section Local averaging weights, the approach will be difficult to extend to additional matching.

Samuelsen (1997) showed that the covariance matrix with KM-weights (without additional matching) can be estimated by

$$\hat{\Gamma} = \hat{I}^{-1} + \hat{I}^{-1} \hat{\Delta} \hat{I}^{-1}. \quad (6)$$

Here  $\hat{I}^{-1}$  is the inverse of the information matrix returned from the weighted Cox-regression and

$$\hat{\Delta} = \sum_i U_i U_i^\top \frac{1 - p_i}{p_i^2} + \sum_{i \neq j} U_i U_j^\top \frac{\widehat{\text{cov}}_{ij}}{p_{ij} p_i p_j}. \quad (7)$$

$U_i$  is the score contribution for individual  $i$  and  $U_i^\top$  its transpose,  $p_{ij}$  the estimated probability that both individual  $i$  and  $j$  were sampled and  $\widehat{\text{cov}}_{ij}$  the estimated covariance between sampling indicators for these two individuals. Note that the score contributions  $U_i$  and the  $W_i$ 's considered by Samuelsen (1997) are asymptotically equivalent. The indices  $i$  and  $j$  in the sums run over all non-cases in the study. Let also  $U$  be a matrix consisting of all  $U_i$  for non-cases. It was argued by Samuelsen (1997) that the term  $p_{ij}$  can be replaced by  $p_i p_j$ . A formula for  $\widehat{\text{cov}}_{ij}$  is given in Samuelsen (1997). We have implemented this variance formula both with and without additional matching, although in the latter case the  $\widehat{\text{cov}}_{ij}$  needs modification. If two individuals  $i$  and  $j$  cannot both be sampled for any same case, then these two individuals are sampled independently and the covariance of their sampling indicators equals zero. If they can both be sampled for one or more of the same cases then  $\widehat{\text{cov}}_{ij}$  is obtained using Equation (3.1) in Samuelsen (1997), replacing the number at risk at time  $t_j$  by the number at risk  $n_j(t_j)$  at  $t_j$  satisfying the matching criteria, and taking the product only over the cases where both  $i$

and  $j$  can be sampled as controls.

The covariance matrix estimator in Equation 6 has been considered numerically hard to calculate. However, it can be simplified by noting that Equation 7 can be expressed as a matrix product. Let  $R$  be a matrix with the terms  $(1 - p_i) / p_i^2$  for all non-cases along the diagonal and terms  $\widehat{\text{cov}}_{ij} / (p_i p_j)^2$  otherwise. Let furthermore  $U_{(l)}$  be a vector of the  $l$ -th component of  $U$ . Then the component  $\widehat{\Delta}_{kl}$  of  $\widehat{\Delta}$  can be expressed as  $U_{(l)}^\top R U_{(k)}$  and taken together this means that with  $U$  being the matrix consisting of all  $U_i$  for the non-cases we get  $\widehat{\Delta} = U^\top R U$ . Thus the calculation of the second sum in Equation 7 as a double for-loop is avoided. However, with additional matching and many sampled non-cases  $\widehat{\text{cov}}_{ij}$  and the matrix  $R$  may still be computationally demanding.

## The R package multipleNCC

The **multipleNCC** package provides one main function: `wpl()` which carries out the weighted Cox-regressions. It calls one of four internal functions, `pKM()`, `pGAM()`, `pGLM()` and `pChen()`, for weight estimation and may call two additional functions, `ModelbasedVar()` and `PoststratVar()`, for variance estimation. The functions that estimate the sampling probabilities can be accessed through the wrapper functions `KMprob()`, `GAMprob()`, `GLMprob()` and `Chenprob()`. The function `wpl()` is used for estimation of hazard ratios. It has a number of mandatory arguments and some which are set to default values if not specified by the user, see Table 1. An important part of the `wpl()` function is the weighted Cox-regression which is carried out by `coxph()` from the **survival** package.

It is important to note that most arguments should have cohort dimension (Table 1). By that we mean that the arguments should have the same length as the number of subjects  $n$  in the cohort. Thus, partially known covariate information must be imputed. These imputed values are not included in the estimation, hence any values can be chosen, however NA should be avoided.

The GLM-weights are estimated with the `glm()` function in R with 'family = binomial'. For GAM-weights, the `gam()` function in the **mgcv** package (Wood, 2006, 2015) is used, also with 'family = binomial'. The KM- and Chen-weights are estimated as explained in Sections **KM type of weights** and **Local averaging weights**.

For GLM-weights, the matching variables are included in the logistic regression as continuous covariates with caliper matching and as categorical covariates for category matching. For GAM-weights the matching variables are included as smooth functions with caliper matching and as categorical covariates for category matching. If a matching variable has many levels, a large number of parameters must be estimated and some sort of grouping of the levels of the category matching variable(s) could be sensible. Such grouping must be applied to the matching variable before it enters `wpl()`.

If the method of variance estimation is not specified, the robust option is chosen by default. The "Modelbased" option can be chosen for KM-weights. Using the "Modelbased" option for other weights will result in an error message. Similarly if the option "Poststrat" is chosen together with other weights than "Chen", an error message will be displayed.

The code for fitting a `wpl` model is

```
R> wpl(formula, data, samplestat)
```

The formula is a *formula* object and has the same syntax as the formula in the `coxph()` function. The minimal code for fitting a `wpl` model is therefore

```
R> wpl(Surv(survival_time, status) ~ X, data, samplestat)
```

This is a model with one control per case, no additional matching or left-truncation and KM-weights. With left-truncation and GLM-weights it would be

```
R> wpl(Surv(left_time, survival_time, status) ~ X, data, samplestat,
+      weight.method = "glm")
```

and with additional matching

```
R> wpl(Surv(left_time, survival_time, status) ~ X + M, data, samplestat,
+      weight.method = "glm", match.var = M, match.int = c(-2, 2))
```

In this model the controls are matched to the cases on only one additional factor *M*, which for instance could be year of birth, plus/minus two years, represented by 'match.int = c(-2,2)'. Generally, it would be important to adjust for the additional matching variables, since the matching has been broken, thus *M* is also included as an ordinary covariate. If there are more than one matching

Argument	Description	Default value
Surv object	A survival object.	Mandatory
formula	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object. The status variable going into Surv should have one for cases and zero for controls and non-sampled subjects. Cohort dimension.	Mandatory
data	A data.frame used for interpreting the variables named in the formula. Cohort dimension.	Mandatory
samplestat	A vector containing sampling and status information: 0 represents non-sampled subjects in the cohort; 1 represents sampled controls; 2,3,...indicate different events. Cohort dimension.	Mandatory
m	Number of sampled controls. A scalar if equal number of controls for all cases. If unequal number of controls per case: a vector of length equal to the number of cases. The vector must be in the same order as the cases in the samplestat vector.	1
weight.method	One of four weights: "KM", "gam", "glm", "Chen".	"KM"
no.intervals	Number of intervals for censoring times for Chen-weights with only right-censoring.	10
variance	"robust", or "Modelbased" for KM-weights and "Poststrat" for Chen-weights.	"robust"
left.time	Entry time if the survival times are left-truncated. Cohort dimension.	Zero
no.intervals.left	Number of intervals for Chen-weights with left-truncation. A vector of the form [number of intervals for left-truncated time, number of intervals for survival time].	[3, 4]
match.var	If the controls are matched to the cases (on other variables than time), match.var is the vector or matrix of matching variables. Cohort dimension.	Zero
match.int	A vector of length $2 \times$ number of matching variables. For caliper matching (matched on value plus/minus epsilon) match.int should consist of c(-epsilon, epsilon). For exact matching match.int should consist of c(0, 0).	Zero

**Table 1:** Arguments to wpl().

variable, for instance  $M_1$  and  $M_2$ , both of them should be included as covariates and the match.var should be a matrix consisting of the  $M_1$  and  $M_2$ . The match.int argument is still a vector with the intervals in the same order as in match.var. Thus the syntax for a model with one caliper matching criterion and one category criterion could be

```
R> wpl(Surv(left_time, survival_time, status) ~ X + M1 + M2, data, samplestat,
+      weight.method = "gam", match.var = cbind(M1, M2), match.int = c(-2, 2, 0, 0))
```

Note that the interval should be 'c(0,0)' for category matching. As an example, if the controls were matched on year of birth plus/minus 2 years, county of residence and years since first employment plus/minus 6 months, with year of birth and year since first employment measured in years, 'match.int = c(-2,2,0,0,-0.5,0.5)'.

In a traditional analysis of NCC data, subjects appear in the data set as many times as they are sampled. For instance if a subject is first sampled as a control and later itself becomes a case, it appears in the data set first as a control and then a second time as a case. With IPW, the subjects should only appear in the data set once, and it is important to let all subjects who at some point became a case, be a case in the data set.

The event indicator included in the Surv() function could have a one for cases and a zero for

non-cases. However, this event indicator is not actually used by the program. With only right-censored data an event indicator is not required by ‘Surv’ and it can thus be omitted. When the data is left-truncated the event indicator must be included and we therefore suggest to always include it even though it is not used by the program. All information regarding events, possibly of different types, controls and non-sampled subjects in the cohort is included through the sampling-status indicator ‘samplestat’. Non-sampled subjects should be given value zero, sampled controls (who are not cases of any type) should have 1, while cases of the first type should be given 2, cases of the second type 3, etc. By default all subjects with non-zero values (except for the cases in question) are used as controls. If this is not desirable, the sampling-status indicator can be modified, see Section [Sub-endpoints and complex multiple endpoint situations](#).

The estimated weights can be extracted directly from the *wpl* object by \$weights, it is, however important to note that the data is sorted by inclusion time inside *wpl* and the ordering of the weights is therefore not the same as the ordering of the original data. The sampling probabilities can also be estimated directly with KMprob, GAMprob, GLMprob and Chenprob which also return them in the same order as the input data. These four functions have similar syntax, although with some differences in required arguments.

```
R> KMprob(survtime, samplestat, m, left.time = 0, match.var = 0, match.int = 0)
R> GLMprob(survtime, samplestat, left.time = 0, match.var = 0, match.int = 0)
R> GAMprob(survtime, samplestat, left.time = 0, match.var = 0, match.int = 0)
R> Chenprob(survtime, samplestat, no.intervals = 10, left.time = 0,
+           no.intervals.left = c(3, 4))
```

Some arguments to these functions are mandatory, while some arguments should only be supplied in given situations. Those arguments are ‘left.time’ and ‘no.intervals.left’ which should only be included with left-truncated data, and ‘match.var’ and ‘match.int’ only with additionally matched data. All arguments to the functions above can be found in Table 1, except for ‘survtime’ which is the follow-up time. It is important to note that ‘survtime’, ‘samplestat’, ‘left.time’ and ‘match.var’ should have length or number of rows equal to the cohort size  $n$ .

When the controls are matched on additional factors it may happen that there are none, or too few, eligible controls for some cases. Normal practice is then to widen the matching criteria somewhat for those particular cases. But by doing this there will be fewer subjects at risk that meet the original matching criterium than there are sampled controls. *wpl()* will therefore print out a warning telling the user for which case this happens and that the controls for that particular case are given weight = 1. This is reasonable since those controls are sampled with probability close or equal to 1.

**multipleNCC** does not carry out a traditional analysis using a stratified Cox-regression. The main reason for this is that each subject should only appear once in the data set provided to **multipleNCC**. Without explicit information about the original case-control sets it is impossible to reconstruct the original nested case-control data where each subject appears as many times as they are sampled. Additionally, it is so simple to carry out a traditional nested case-control analysis with a stratified Cox-regression that there is no reason to include it in **multipleNCC**.

### Sub-endpoints and complex multiple endpoint situations

A main endpoint can often be divided into sub-endpoints. For instance a cancer endpoint can be divided into metastatic and non-metastatic cancer. Analysis of sub-endpoints using all sampled controls can be carried out with *wpl()* by making a new sampling-status indicator with unique values for each sub-type. So instead of having a *samplestat* indicator from 0–2 (0 = non-sampled subjects, 1 = sampled controls, 2 = cancer cases), it will have values 0–3, where 2 corresponds to metastatic cancer and 3 to non-metastatic cancer.

Many multiple endpoint applications do not fit into a competing risks framework. However, reuse of controls can still be of interest. The solution can then be to estimate the sampling probabilities with KMprob, GAMprob, GLMprob or Chenprob and use the inverse of those estimated probabilities as weights in the ordinary coxph() function with robust variances. An example of a multiple endpoint situation which does not fit into the competing risks framework is the case of subsequent events (Støer et al., 2014). This is a situation with two types of events and the second type may only occur after the first, e.g. incidence of prostate cancer and death from prostate cancer. Using all sampled controls when analyzing the subsequent endpoint may increase the efficiency substantially.

### Other models and estimators

Inverse probability weighting is a standard approach for handling missing data and case-control data can be considered as data missing by design. Thus the functions for estimating inclusion probabilities

described above can be used more generally for addressing other models than proportional hazards. For instance [Samuelson \(1997\)](#) discussed the possibility of fitting parametric survival models, [Suisa et al. \(1998\)](#) considered estimation of standardized mortality ratios and [Cai and Zheng \(2012\)](#) discussed estimation from estimating equations in general with IPW from NCC studies.

Often software allows for weighting and then it is straightforward to obtain the weighted estimators. Examples are the Nelson-Aalen and Breslow estimators for cumulative hazards or additive hazards models. With respect to variance estimation of weighted estimators we believe that robust variances often will give results that closely match the model based, although theoretically they are conservative ([Samuelson, 1997](#); [Cai and Zheng, 2012](#)). In general, however, theory has not been carefully developed for such estimators and implementation of robust procedures has not generally been validated, thus care should be taken when interpreting the variance estimates.

Furthermore, we believe that the IPW weights for NCC are in particular useful when considering time to event data with the original cohort follow-up scheme. We are less convinced that the IPW approach is the best choice when for instance estimating the population means of the exposures obtained in the NCC or considering regression models for how such exposures depend on other cohort information.

## Other packages

There exists a number of packages for weighted analysis for different purposes. Some are aimed at causal inferences such as [ipw](#) ([van der Wal and Geskus, 2011](#)) and [MatchIt](#) ([Ho et al., 2011](#)). Others have a missing data or survey sampling perspective. Two such packages are [NestedCohort](#) ([Mark and Katki, 2006](#); [Katki and Mark, 2008](#)) and [survey](#) ([Lumley, 2004, 2014](#)). The [survey](#) package was developed for analyses of survey data, but include functions for two-phase designs. A nested case-control design can be seen as a two-phase design where the cohort is collected at Phase 1 and the Phase 2 data is the cases and sampled controls with additional covariate information. The Phase 2 sampling scheme is of course somewhat more complex than what is usually seen in survey sampling. However, when Chen-weights are used for the nested case-control design, a stratified Phase 2 sampling is implicitly assumed, which is a well-known survey sampling design. Hence, a weighted analysis using Chen-weights can be carried out by specifying a stratified two-phase design with the function `twophase` and carry out the Cox-regression using this design with `svycoxph`.

The [NestedCohort](#) package is a general package for cohort analyses with a missing data/two-phase design perspective. The theory behind it is based to some extent on [Robins et al. \(1994\)](#) and the variance estimators also build on this paper. Weighted Cox-regressions are carried out with `nested.coxph` where a working model is assumed for the sampling probabilities in Phase 2. This working model is a logistic regression model for the sampling indicator with all variables contributing to the missingness as covariates. This is identical to specifying `weight.method = "glm"` in `wpl`. However, the variance estimator in `nested.coxph` may be somewhat better in special situations where the robust variances are conservative. The [NestedCohort](#) package can however only be used for `glm`-weights so that the more commonly used KM-weights are not applicable with this package. The authors of the package also state that fine matching is problematic with [NestedCohort](#). In the data example below `nested.coxph` gave identical estimates and standard errors as using `glm`-weights with `wpl`.

## Analysis of example data set

### The data set CVD\_Accidents

We use a collection of cardiovascular health screenings as our cohort, also used in [Aalen et al. \(2008\)](#). All men and women aged 35–49 from three Norwegian counties: Oppland, Sogn og Fjordane and Finnmark were invited to participate in health screenings from 1974–1978. The screening consisted of a health examination including measurement of height and weight. The participants were also asked to respond to a questionnaire which among other things contained questions regarding smoking status.

More than 90% of the invited subjects chose to participate which resulted in a cohort of about 50 000 subjects. The cohort was linked to the Causes of Death Registry kept by Statistics Norway and followed up for deaths until the end of year 2000. Since there were few subjects at risk younger than 40 years, the survival times were left-truncated at age 40 or age at health screening. The left-truncation time is named `agestart` in the data. The survival time is named `agestop` and is either the age at death or censoring.

The data set included in [multipleNCC](#) and used as illustration here, is a random sample of 3933 subjects from the cohort. It is a cohort study, but we have carried out a synthetic nested case-

control study within this smaller cohort. Having full cohort information enables comparison between `wpl()` and corresponding analyses carried out on the full cohort. The synthetic data generation was performed in two steps. In the first step a nested case-control sampling was carried out by sampling one control per case matched sex and  $BMI \pm 2$ . This was done in a for-loop over event times sampling one subject at random from those still alive at that time in addition to fitting the matching criterium. When the controls are not additionally matched or matched only on a categorical variable the `ccwc`-function from the `Epi` package (Carstensen et al., 2016) offers a simple way to create a nested case-control design from a cohort. However, if some of the matching variables are continuous it still has to be done as described above. The second step is necessary only for the weighted Cox-regression and it involves removing duplicate subjects, due to that some controls are sampled multiple times and some controls later become cases, from the sampled data. For those controls who later become cases it is important to keep the “case-entry”, while for those controls sampled multiple times, the entries will be identical and one of them is kept at random.

Four types of deaths are recorded in the data: (1) cancer, (2) cardiovascular disease, (3) alcohol abuse, liver disease, accidents and violence, and (4) other medical causes. For simplicity we only use cardiovascular disease ( $n = 236$ ) and alcohol abuse, liver disease, accidents and violence, henceforth referred to as ALAV ( $n = 60$ ). The data set is included in `multipleNCC` and can be loaded by `data("CVD_Accidents")`.

When we analyze this data set with IPW, we will use all sampled controls for both endpoints, hence supplement the controls for ALAV deaths with the cardiovascular disease controls and cases. And vice versa, supplement controls for cardiovascular disease cases with cases and controls for ALAV cases. For ALAV deaths, the number of controls increases substantially when including the controls for cardiovascular deaths.

The matching variables are BMI ( $M_1 = \text{bmi}$ ) and sex ( $M_2 = \text{sex}$ ). The matching variables are adjusted for in the model to remove confounding due to breaking the matching. We included BMI as continuous variable and sex is a categorical variable. Smoking ( $X = \text{smoking3gr}$ ) is our explanatory variable and is categorized into never smoked, former smoker and current smoker.

## Fitting models

We consider the two models

$$\begin{aligned} h_{1i}(t_i|X, M_1, M_2) &= h_{10}(t_i) \exp(\beta'_1 X_i + \gamma_{11} M_{1i} + \gamma_{12} M_{2i}) \\ h_{2i}(t_i|X, M_1, M_2) &= h_{20}(t_i) \exp(\beta'_2 X_i + \gamma_{21} M_{1i} + \gamma_{22} M_{2i}). \end{aligned}$$

Here  $h_1(\cdot)$  and  $h_2(\cdot)$  are the hazard for death from CVD and ALAV, respectively. To fit the weighted partial likelihoods corresponding to those models, the command below can be used

```
R> fit <- wpl(Surv(agestart, agestop, dead24) ~ factor(X) + bmi + sex,
+   data = CVD_Accidents, m = 1, samplestat, weight.method = "glm",
+   match.var = cbind(CVD_Accidents$bmi, CVD_Accidents$sex),
+   match.int = c(-2, 2, 0, 0))
```

The ‘status’ argument to the ‘Surv’ function (in this example ‘dead24’) can in practice contain anything since the information regarding who are cases and controls, and also which type of endpoint the cases experienced are provided through `samplestat`. The ‘match.int’ argument will in this situation be a vector of two elements, ‘`match.int = c(-2, 2, 0, 0)`’, since the controls are matched to the cases on  $BMI \pm 2$ , and sex.

The ‘summary’ and ‘print’ commands can be used to display the results of the analysis. The output resembles output from traditional Cox-regression (see next page), and the results for the different endpoints are printed below each other. Both the naive and robust/estimated standard errors are printed, although only the robust/estimated should be reported.

The \$-operator is usually used when specific parts of an object (i.e., ‘`fit$coefficients`’) are to be extracted. When there is more than one endpoint this will only give you the corresponding element for the first endpoint, i.e., ‘`fit$coefficients`’ will give you (0.47107, 1.34245, 0.08051, -1.22307), thus only the estimates for the cardiovascular disease endpoint. To extract any element from the object, `[[[]]]` can be used, i.e., ‘`fit[[23]]`’ will give you the estimates from the second analysis of death from ALAV. For a full list of elements in the `wpl` object, the `names()` function can be used. Each element will occur the same number of times as there are different endpoints, and the first occurrence corresponds to the first endpoint, the second occurrence to the second endpoint, etc.

```
R> summary(fit)
```

```
Endpoint 1 :
```

```

Call:
wpl.formula(formula = Surv(agestart, agestop, dead24) ~ factor(smoking3gr) +
  bmi + factor(sex), data = CVD_Accidents,
  samplestat = CVD_Accidents$samplestat,
  match.var = cbind(CVD_Accidents$bmi, CVD_Accidents$sex),
  match.int = c(-2, 2, 0, 0), weight.method = "glm")

n= 566, number of events= 236

              coef exp(coef) se(coef) robust se      z Pr(>|z|)
factor(smoking3gr)2  0.47107   1.60171  0.22099   0.26057  1.808  0.07062 .
factor(smoking3gr)3  1.34245   3.82842  0.19400   0.23424  5.731 9.98e-09 ***
bmi                  0.08051   1.08384  0.01799   0.02562  3.143  0.00167 **
factor(sex)2        -1.22307   0.29433  0.15980   0.22475 -5.442 5.27e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

              exp(coef) exp(-coef) lower .95 upper .95
factor(smoking3gr)2    1.6017    0.6243   0.9611   2.6692
factor(smoking3gr)3    3.8284    0.2612   2.4190   6.0591
bmi                   1.0838    0.9226   1.0308   1.1396
factor(sex)2           0.2943    3.3976   0.1895   0.4572

Endpoint 2 :
Call:
coxph(formula = Surv(left.time.ncc, survtime.ncc, status.ncc ==
  i) ~ x + cluster(ind.no.ncc), weights = 1/p)

n= 566, number of events= 60

              coef exp(coef) se(coef) robust se      z Pr(>|z|)
factor(smoking3gr)2 -0.61629   0.53994  0.45484   0.48347 -1.275 0.202413
factor(smoking3gr)3  0.92343   2.51792  0.32202   0.34402  2.684 0.007270 **
bmi                  0.08383   1.08744  0.03813   0.04587  1.828 0.067619 .
factor(sex)2        -1.42549   0.24039  0.32702   0.36888 -3.864 0.000111 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

              exp(coef) exp(-coef) lower .95 upper .95
factor(smoking3gr)2    0.5399    1.8520   0.2093   1.3928
factor(smoking3gr)3    2.5179    0.3972   1.2829   4.9417
bmi                   1.0874    0.9196   0.9939   1.1897
factor(sex)2           0.2404    4.1599   0.1167   0.4954

```

The sampling probabilities can be directly estimated using one of the functions: KMprob, GAMprob, GLMprob and Chenprob, for instance

```

R> glmp <- GLMprob(CVD_Accidents$agestop, CVD_Accidents$samplestat,
+   left.time = CVD_Accidents$agestart, match.var = cbind(CVD_Accidents$sex,
+   CVD_Accidents$bmi), match.int = c(0, 0, -2, 2))
R> kmp <- KMprob(CVD_Accidents$agestop, CVD_Accidents$samplestat, 1,
+   left.time = CVD_Accidents$agestart, match.var = cbind(CVD_Accidents$sex,
+   CVD_Accidents$bmi), match.int = c(0, 0, -2, 2))

```

It is of interest to examine the weights for the sampled controls only, as the cases have weight 1 and the non-sampled subjects will not affect estimates. We can for instance inspect summary statistics

```
R> summary(1/glmp[CVD_Accidents$samplestat == 1])
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
	4.575	6.545	9.201	13.240	15.080	73.180

```
R> summary(1/kmp[CVD_Accidents$samplestat == 1])
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
--	------	---------	--------	------	---------	------

	CVD					
	Former smoker			Current smoker		
	HR (95 % CI)	SE( $\beta$ )	Eff.	HR (95 % CI)	SE( $\beta$ )	Eff.
Cohort	1.72 (1.11–2.66)	0.22	1.00	3.25 (2.21–4.79)	0.20	1.00
Trad.	1.66 (0.94–2.93)	0.29	0.59	3.52 (2.09–5.94)	0.27	0.55
IPW-GLM	1.60 (0.96–2.67)	0.26	0.73	3.83 (2.42–6.06)	0.23	0.72
IPW-KM	1.65 (0.98–2.78)	0.26	0.66	3.97 (2.48–6.35)	0.24	0.69

	ALAV					
	Former smoker			Current smoker		
	HR (95 % CI)	SE( $\beta$ )	Eff.	HR (95 % CI)	SE( $\beta$ )	Eff.
Cohort	0.56 (0.23–1.38)	0.46	1.00	2.05 (1.07–3.90)	0.33	1.00
Trad.	0.48 (0.13–1.69)	0.65	0.50	2.90 (1.11–7.61)	0.49	0.45
IPW-GLM	0.54 (0.21–1.39)	0.48	0.90	2.52 (1.28–4.94)	0.34	0.92
IPW-KM	0.55 (0.21–1.40)	0.49	0.90	2.56 (1.28–5.02)	0.35	0.89

**Table 2:** Results from analyses on the entire cohort, traditional nested case-control analyses and IPW analyses with GLM-weights and robust variances, and KM-weights and estimated variances. Never smoked is used as reference. Eff. – variance for cohort estimator divided by variance for corresponding estimator. SE( $\beta$ ) – standard error (robust SE for IPW estimators with estimated SE in parentheses for KM-weights).

2.481 6.839 8.772 13.150 15.020 95.810

The two types of weights correspond well, although KM-weights have a somewhat heavier tail. It is worth noting that the subjects with the largest weights are those with the shortest follow-up time and therefore those subjects do not have a too large impact on the analysis even though their weight is large, since they are included in few risk sets. GAM-weights were similar to GLM-weights, but with a somewhat heavier tail (not shown) and Chen-weights are not applicable here since they are not implemented for additional matching.

### Comparison between stratified coxph and wpl

Table 2 displays a comparison between the full cohort analysis, the traditional estimator for nested case-control data and the IPW-estimator using `wpl()` with GLM- and KM-weights. Being a smoker significantly increases the risk of death from cardiovascular disease. Being a former smoker also increases the risk of death from cardiovascular disease, although only significantly in the cohort analysis. Being a former smoker has a non-significant protective effect on death from ALAV, while being a smoker significantly increases the risk of dying from ALAV.

The hazard ratios estimated with the traditional estimator and with `wpl()` are fairly similar to the cohort estimates taking into account the size of the standard errors. The most pronounced difference is between the cohort hazard rate and the hazard rate for the traditional estimator for being a smoker on death from ALAV, 2.05 vs. 2.90. However, considering the size of the standard errors this is not a large difference.

For the cardiovascular disease endpoint, the standard errors of the IPW analyses are somewhat smaller than the standard errors of the traditional estimator, resulting in a little bit higher efficiency for the IPW-estimators. For the ALAV endpoint, the standard errors are substantial lower and a large efficiency gain is obtained with the IPW-estimators. The reason for this is that the IPW-estimators make use of a number of extra controls as all cases and controls from the cardiovascular disease endpoint are used as additional controls. On average the number of controls per case increase from 1 to more than 8. We have chosen to include cardiovascular disease cases as additional controls for the cases who died from ALAV, and also the cases who died from ALAV as additional controls for the cases who died from cardiovascular disease. However, the cases who are used as additional controls will contribute little to the analysis since they are non-cases (for the particular endpoint) with weight equal to 1. We have reported robust standard errors for KM-weights in Table 2, however the estimated standard errors are very similar, for CVD endpoint 0.27 and 0.24 and for ALAV 0.48 and 0.35 for former and current smoker respectively.

## Discussion

We have demonstrated the **multipleNCC** package in R which allows for breaking the matching and reusing controls in NCC designs. The main function `wpl()` estimates sampling probabilities and perform weighted Cox-regressions. It handles right-censored, left-truncated and additionally matched data, and varying number of sampled controls. We have also explained how the variance can be estimated without additional matching (KM- and Chen-weights) and with additional matching (KM-weights).

The package is particularly useful in situations with multiple outcomes. It has a competing risks perspective, in the sense that with more than one type of endpoint, each endpoint is estimated separately and all controls and cases of other types are used as additional controls. In many situations the competing risks framework may not be suitable, although reusing controls can still be of interest. The solution can then be to estimate the sampling probabilities for all cohort members, using one of the four functions `KMprob`, `GAMprob`, `GLMprob` or `Chenprob`, and carry out weighted analysis that fit the situation at hand.

The `gam()` function from the **mgcv** package is used for estimation of the GAM-weights. We could alternatively have used the `gam()` function from the **gam** package (Hastie, 2015) or even a different form of smoothing. It has however become evident that the exact value of the weights are not too important as long as they are fairly reasonable, thus the choice of smoothing does probably not affect final hazard ratios and standard errors. For the same reason there are usually only minor differences with regards to final hazard ratios and standard errors between the four weight estimators discussed in Section [Weight estimation](#) (Støer and Samuelsen, 2012, 2013).

Sometimes the cases and controls are matched closer together than is strictly necessary. Very close matching can lead to small KM-weights since the probability of being sampled will be large for the controls that were sampled (and very small for most of the non-sampled subjects) and this could lead to biased estimates (Støer and Samuelsen, 2013). A solution could be to increase the length of '`match.int`'. For example in the data example above, we could replace '`match.int = c(-2, 2)`' with '`match.int = c(-4, 4)`'. Widening the matching interval could introduce bias, thus it is important to carry this out with caution. The equivalence of this for category matching is to reduce the number of levels of the matching variable by some sort of grouping.

## Bibliography

- O. O. Aalen, Ø. Borgan, and H. K. Gjessing. *Survival and Event History Analysis*. Statistics for Biology and Health. Springer-Verlag, 1st edition, 2008. doi: 10.1007/978-0-387-68560-1. [p12]
- W. E. Barlow. Robust variance estimation for the case-cohort design. *Biometrics*, 50(4):1064–1072, 1994. doi: 10.2307/2533444. [p8]
- Ø. Borgan, B. Langholz, S. O. Samuelsen, L. Goldstein, and J. Pogoda. Exposure stratified case-cohort designs. *Lifetime Data Analysis*, 6(1):39–58, 2000. doi: 10.1023/a:1009661900674. [p8]
- T. Cai and Y. Zheng. Evaluating prognostic accuracy of biomarkers in nested case-control studies. *Biostatistics*, 13(1):89–100, 2012. doi: 10.1093/biostatistics/kxr021. [p5, 7, 8, 12]
- B. Carstensen, M. Plummer, E. Laara, and M. Hills. *Epi: A Package for Statistical Analysis in Epidemiology*, 2016. URL <https://CRAN.R-project.org/package=Epi>. R package version 2.0. [p13]
- K. N. Chen. Generalized case-cohort sampling. *Journal of the Royal Statistical Society B*, 63(4):791–809, 2001. doi: 10.1111/1467-9868.00313. [p5, 7, 8]
- T. V. Clendenen, E. Lundin, A. Zeleniuch-Jacquotte, K. L. Koenig, F. Berrino, A. Lukanova, A. E. Lokshin, A. Idahl, N. Ohlson, G. Hallmans, V. Krogh, S. Sieri, P. Muti, A. Marrangoni, B. M. Nolen, M. L. Liu, R. E. Shore, and A. A. Arslan. Circulation inflammation markers and risk of epithelial ovarian cancer. *Cancer Epidemiology, Biomarkers and Prevention*, 20(5):799–810, 2011. doi: 10.1158/1055-9965.epi-10-1180. [p5]
- W. G. Cochran and D. B. Rubin. Controlling bias in observational studies: A review. *Sankhyā: The Indian Journal of Statistics A*, 35(4):417–446, 1973. doi: 10.1017/cbo9780511810725.005. [p6]
- B. Floderus, T. Persson, C. Stenlund, A. Wennberg, Å. Öst, and B. Knave. Occupational exposure to electromagnetic fields in relation to leukemia and brain tumors: A case-control study in Sweden. *Cancer Causes and Control*, 4(5):465–476, 1993. doi: 10.1007/bf00050866. [p5]

- T. K. Grimsrud, S. R. Berge, T. Haldorsen, and A. Andersen. Exposure to different forms of nickel and risk of lung cancer. *American Journal of Epidemiology*, 156(12):1123–1132, 2002. doi: 10.1093/aje/kwf165. [p5]
- S. E. Hankinson, W. C. Willett, G. A. Colditz, D. J. Hunter, D. S. Michaud, B. Deroo, B. Rosner, F. E. Speizer, and M. Pollak. Circulating concentrations of insulin-like growth factor-I and risk of breast cancer. *The Lancet*, 351(9113):1393–1396, 1998. doi: 10.1016/s0140-6736(97)10384-1. [p5]
- M. H. Hansen and W. N. Hurwitz. On the theory of sampling from finite populations. *Annals of Mathematical Statistics*, 14:333–362, 1943. doi: 10.1214/aoms/1177731356. [p6]
- T. Hastie. *gam: Generalized Additive Models*, 2015. URL <https://CRAN.R-project.org/package=gam>. R package version 1.12. [p16]
- T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models*. Chapman & Hall, London, 2009. [p7]
- D. E. Ho, K. Imai, G. King, and E. A. Stuart. MatchIt: Nonparametric preprocessing for parametric causal inference. *Journal of Statistical Software*, 42(8):1–28, 2011. doi: 10.18637/jss.v042.i08. [p12]
- D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260):663–685, 1952. doi: 10.1080/01621459.1952.10483446. [p6]
- C. M. Hultman, P. Sparén, N. Takei, R. M. Murray, and S. Cnattingius. Prenatal and perinatal risk factors for schizophrenia and affective psychosis, and reactive psychosis of early onset: Case-control study. *British Medical Journal*, 318(7181):421–426, 1999. doi: 10.1136/bmj.318.7181.421. [p5]
- H. A. Katki and S. D. Mark. Survival analysis for cohorts with missing covariate information. *R News*, 8(1):14–19, 2008. URL [https://CRAN.R-project.org/doc/Rnews/Rnews\\_2008-1.pdf](https://CRAN.R-project.org/doc/Rnews/Rnews_2008-1.pdf). [p12]
- R. J. Levine, S. E. Maynard, C. Qian, K. H. Lim, L. J. England, K. F. Yu, E. F. Schisterman, R. Thadhani, B. P. Sachs, F. H. Epstein, B. M. Sibai, V. P. Sukhatme, and S. A. Karumanchi. Circulating angiogenic factors and the risk of preeclampsia. *New England Journal of Medicine*, 350(7):672–683, 2004. doi: 10.1056/nejmoa031884. [p5]
- D. Y. Lin and L. J. Wei. The robust inference for the Cox proportional hazards model. *Journal of the American Statistical Association*, 84(408):1074–1078, 1989. doi: 10.1080/01621459.1989.10478874. [p8]
- T. Lumley. Analysis of complex survey samples. *Journal of Statistical Software*, 9(1):1–19, 2004. doi: 10.18637/jss.v009.i08. [p12]
- T. Lumley. *survey: Analysis of Complex Survey Samples*, 2014. URL <https://CRAN.R-project.org/package=survey>. R package version 3.30. [p12]
- S. D. Mark and H. A. Katki. Specifying and implementing nonparametric and semiparametric survival estimators in two-stage (sampled) cohort studies with missing case data. *Journal of the American Statistical Association*, 101(474):460–471, 2006. doi: 10.1198/016214505000000952. [p7, 12]
- H. E. Meyer, T. E. Robsahm, T. Bjørge, M. Brunstad, and R. Blomhoff. Vitamin D, season and the risk of prostate cancer: A nested case-control study within Norwegian health studies. *American Journal of Clinical Nutrition*, 97(1):147–154, 2013. doi: 10.3945/ajcn.112.039222. [p5]
- N. Øyen, T. Markestad, R. Skjærven, L. M. Irgens, K. Helweg-Larsen, B. Alm, G. Norvenius, and G. Wennergren. Combined effects of sleeping position and prenatal risk factors in sudden infant death syndrome: The Nordic epidemiological SIDS study. *Pediatrics*, 100(4):613–621, 1997. doi: 10.1542/peds.100.4.613. [p5]
- J. Parsonnet, G. D. Friedman, D. P. Vansdersteen, Y. Chang, J. H. Vogelman, N. Orentreich, and R. K. Sibley. Helicobacter pylori infection and the risk of gastric carcinoma. *New England Journal of Medicine*, 325(16):1127–1131, 1991. doi: 10.1056/nejm199110173251603. [p5]
- J. Robins, A. Rotnitzky, and L. Zhao. Estimation of regression coefficients when some regressors are not always observed. *Journal of the American Statistical Association*, 89(427):846–866, 1994. doi: 10.1080/01621459.1994.10476818. [p6, 12]
- O. Saarela, S. Kulathinal, E. Arjas, and E. Läärä. Nested case-control data utilized for multiple outcomes: A likelihood approach and alternatives. *Statistics in Medicine*, 27(28):5991–6008, 2008. doi: 10.1002/sim.3416. [p5, 6, 7]

- A. Salim, C. Hultman, P. Sparén, and M. Reilly. Combining data from 2 nested case-control studies of overlapping cohorts to improve efficiency. *Biostatistics*, 10(1):70–79, 2009. doi: 10.1093/biostatistics/kxn016. [p5, 6, 7]
- A. Salim, Q. Yang, and M. Reilly. The value of reusing prior nested case-control data in new studies with different outcome. *Statistics in Medicine*, 31(11–12):1291–1302, 2012. doi: 10.1002/sim.4494. [p5]
- S. O. Samuelsen. A pseudolikelihood approach to analysis of nested case-control studies. *Biometrika*, 84(2):379–394, 1997. doi: 10.1093/biomet/84.2.379. [p5, 6, 7, 8, 12]
- S. O. Samuelsen, H. Ånestad, and A. Skrondal. Stratified case-cohort analysis of general cohort sampling designs. *Scandinavian Journal of Statistics*, 34(1):103–119, 2007. doi: 10.1111/j.1467-9469.2006.00552.x. [p5, 7, 8]
- N. C. Støer and S. O. Samuelsen. Comparison of estimators in nested case-control studies with multiple outcomes. *Lifetime Data Analysis*, 18(3):261–283, 2012. doi: 10.1007/s10985-012-9214-8. [p5, 16]
- N. C. Støer and S. O. Samuelsen. Inverse probability weighting in nested case-control studies with additional matching - a simulation study. *Statistics in Medicine*, 32(30):5328–5339, 2013. doi: 10.1002/sim.6019. [p5, 6, 7, 16]
- N. C. Støer and S. O. Samuelsen. *multipleNCC: Weighted Cox-Regression for Nested Case-Control Data*, 2016. URL <https://CRAN.R-project.org/package=multipleNCC>. R package version 1.2-1. [p5]
- N. C. Støer, H. E. Meyer, and S. O. Samuelsen. Reuse of controls in nested case-control studies. *Epidemiology*, 25(2):315–317, 2014. doi: 10.1097/ede.0000000000000057. [p5, 11]
- S. Suissa, M. Edwardes, and J. Boivin. External comparisons from nested case-control designs. *Epidemiology*, 9(1):72–78, 1998. doi: 10.1097/00001648-199801000-00015. [p12]
- T. M. Therneau. *survival: A Package for Survival Analysis in S*, 2016. URL <https://CRAN.R-project.org/package=survival>. R package version 2.40-1. [p5]
- T. M. Therneau and P. M. Grambsch. *Modeling Survival Data: Extending the Cox Model*. Springer-Verlag, 2000. doi: 10.1007/978-1-4757-3294-8. [p5]
- D. C. Thomas. Addendum to: “Methods of cohort analysis: Appraisal by application to asbestos mining” by Liddell FDK, McDonald JC and Thomas DC. *Journal of the Royal Statistical Society A*, 140(4):469–491, 1977. doi: 10.2307/2345280. [p5]
- T. Tynes and T. Haldorsen. Electromagnetic fields and cancer in children residing near Norwegian high-voltage power lines. *American Journal of Epidemiology*, 145(3):219–226, 1997. doi: 10.1093/oxfordjournals.aje.a009094. [p5]
- W. M. van der Wal and R. B. Geskus. ipw: An R package for inverse probability weighting. *Journal of Statistical Software*, 43(13):1–23, 2011. doi: 10.18637/jss.v043.i13. [p12]
- S. Wood. *mgcv: Mixed GAM Computation Vehicle with GCV/AIC/REML Smoothness Estimation*, 2015. URL <https://CRAN.R-project.org/package=mgcv>. R package version 1.8-6. [p9]
- S. N. Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, 2006. [p9]

*Nathalie C. Støer*  
Department of Mathematics  
University of Oslo  
Norway  
and  
Medical Epidemiology and Biostatistics  
Karolinska Institutet  
Sweden  
[nathalie.stoer@ki.se](mailto:nathalie.stoer@ki.se)

*Sven Ove Samuelsen*  
Department of Mathematics  
University of Oslo  
Norway  
[osamuels@math.uio.no](mailto:osamuels@math.uio.no)

# QPot: An R Package for Stochastic Differential Equation Quasi-Potential Analysis

by Christopher M. Moore, Christopher R. Stieha, Ben C. Nolting, Maria K. Cameron, and Karen C. Abbott

**Abstract** **QPot** (pronounced *kīōō + pāt*) is an R package for analyzing two-dimensional systems of stochastic differential equations. It provides users with a wide range of tools to simulate, analyze, and visualize the dynamics of these systems. One of **QPot**'s key features is the computation of the quasi-potential, an important tool for studying stochastic systems. Quasi-potentials are particularly useful for comparing the relative stabilities of equilibria in systems with alternative stable states. This paper describes **QPot**'s primary functions, and explains how quasi-potentials can yield insights about the dynamics of stochastic systems. Three worked examples guide users through the application of **QPot**'s functions.

## Introduction

Differential equations are an important modeling tool in virtually every scientific discipline. Most differential equation models are deterministic, meaning that they provide a set of rules for how variables change over time, and no randomness comes into play. Reality, of course, is filled with random events (i.e., noise or stochasticity). Unfortunately, many of the analytic techniques developed for deterministic ordinary differential equations are insufficient to study stochastic systems, where phenomena like noise-induced transitions between alternative stable states and metastability can occur. For systems subject to stochasticity, the quasi-potential is a tool that yields information about properties such as the expected time to escape a basin of attraction, the expected frequency of transitions between basins, and the stationary probability distribution. **QPot** (abbreviation of Quasi-Potential; Moore et al., 2016) is an R package that allows users to calculate quasi-potentials, and this paper is a tutorial of its application. This package is intended for use by any researchers who are interested in understanding how stochasticity impacts differential equation models. **QPot** makes quasi-potential analysis accessible to a broad range of modelers, including those who have not previously encountered the topic. The key functions in package **QPot** are listed in Table 1.

## Adding stochasticity to deterministic models

Consider a differential equation model of the form

$$\begin{aligned} \frac{dx}{dt} &= f_1(x(t), y(t)) \\ \frac{dy}{dt} &= f_2(x(t), y(t)). \end{aligned} \tag{1}$$

In many cases, state variables are subject to continuous random perturbations, which are commonly modeled as white noise processes. To incorporate these random influences, the original system of deterministic differential equations can be transformed into a system of stochastic differential equations:

$$\begin{aligned} dX(t) &= f_1(X(t), Y(t)) dt + \sigma dW_1(t) \\ dY(t) &= f_2(X(t), Y(t)) dt + \sigma dW_2(t). \end{aligned} \tag{2}$$

X and Y are now stochastic processes (a change emphasized through the use of capitalization); this means that, at every time  $t$ ,  $X(t)$  and  $Y(t)$  are random variables, as opposed to real numbers.  $\sigma \geq 0$  is a parameter specifying the noise intensity, and  $W_1(t)$  and  $W_2(t)$  are Wiener processes. A Wiener process is a special type of continuous-time stochastic process whose changes over non-overlapping time intervals,  $\Delta t_1$  and  $\Delta t_2$ , are independent Gaussian random variables with means zero and standard deviations  $\sqrt{\Delta t_1}$  and  $\sqrt{\Delta t_2}$ , respectively. The differential notation in equations (2) is a formal way of representing a set of stochastic integral equations, which must be used because realizations of Wiener processes are not differentiable (to be precise, with probability one, a realization of a Wiener process will be almost nowhere differentiable). The functions  $f_1$  and  $f_2$  are called the deterministic

Function	Main arguments	Description
TSTraj()	Deterministic skeleton, $\sigma$ , $T, \Delta t$	Creates a realization (time series) of the stochastic differential equations.
TSplot()	TSTraj() output	Plots a realization of the stochastic differential equations, with an optional histogram side-plot. Plots can additionally be two-dimensional, which show realizations in $(X, Y)$ -space.
TSDensity()	TSTraj() output	Creates a density plot of a trajectory in $(X, Y)$ -space in one or two dimensions.
QPotential()	Deterministic skeleton, stable equilibria, bounds, mesh (number of divisions along each axis)	Creates a matrix corresponding to a discretized version of the local quasi-potential function for each equilibrium.
QPGlobal()	Local quasi-potential matrices, unstable equilibria	Creates a global quasi-potential surface.
QPIinterp()	Global quasi-potential, $(x, y)$ -coordinates	Evaluates the global quasi-potential at $(x, y)$ .
QPContour()	Global quasi-potential	Creates a contour plot of the quasi-potential.
VecDecomAll()	Global quasi-potential, deterministic skeleton, bounds	Creates three vector fields: the deterministic skeleton, the negative gradient of the quasi-potential, and the remainder vector field. To find each field individually, the functions VecDecomVec(), VecDecomGrad(), or VecDecomRem() can be used.
VecDecomPlot()	Deterministic skeleton, gradient, or remainder field	Creates a vector field plot for the vector, gradient, or remainder field.

**Table 1:** Key functions in package **QPot**.

skeleton. The deterministic skeleton can be viewed as a vector field that determines the dynamics of trajectories in the absence of stochastic effects. We will forgo a complete overview of stochastic differential equations here; interested readers are encouraged to seek out texts like [Allen \(2007\)](#) and [Iacus \(2009\)](#). We note that throughout this paper we use the Itô formulation of stochastic differential equations.

## The quasi-potential

Consider System (2), with deterministic skeleton (1). If there exists a function  $V(x, y)$  such that  $f_1(x, y) = -\frac{\partial V}{\partial x}$  and  $f_2(x, y) = -\frac{\partial V}{\partial y}$ , then System (1) is called a *gradient system* and  $V(x, y)$  is called the system's *potential function*. The dynamics of a gradient system can be visualized by considering the  $(x, y)$ -coordinates of a ball rolling on a surface specified by  $z = V(x, y)$ . Gravity causes the ball to roll downhill, and stable equilibria correspond to the bottoms of the surface's valleys.  $V(x, y)$  is a Lyapunov function for the system, which means that if  $(x(t), y(t))$  is a solution to System (1), then  $\frac{d}{dt}(V(x(t), y(t))) \leq 0$ , and the only places that  $\frac{d}{dt}(V(x(t), y(t))) = 0$  are at equilibria. This means that the ball's elevation will monotonically decrease, and will only be constant if the ball is at an equilibrium. The basin of attraction of a stable equilibrium  $e^*$  of System (1) is the set of points that lie on solutions that asymptotically approach  $e^*$ .

The potential function is useful for understanding the stochastic System (2). As in the deterministic case, the dynamics of the stochastic system can be represented by a ball rolling on the surface  $z = V(x, y)$ ; in the stochastic system, however, the ball experiences random perturbations due to noise terms in the System (2). In systems with multiple stable equilibria, these random perturbations can cause a trajectory to move between different basins of attraction. The depth of the potential (that is, the difference in  $V$  at the equilibrium and the lowest point on the boundary of its basin of attraction), is a useful measure of the stability of the equilibrium (see [Nolting and Abbott, 2016](#)). The deeper the

potential, the less likely it will be for stochastic perturbations to cause an escape from the basin of attraction. This relationship between the potential and the expected time to escape from a basin of attraction can be made precise (see formulae in the appendices of Nolting and Abbott, 2016). Similarly, the potential function is directly related to the expected frequency of transitions between different basins, and to the stationary probability distribution of System (2).

Unfortunately, gradient systems are very special, and a generic system of the form (1) will almost certainly not be a gradient system. That is, there will be no function  $V(x, y)$  that satisfies  $f_1(x, y) = -\frac{\partial V}{\partial x}$  and  $f_2(x, y) = -\frac{\partial V}{\partial y}$ .

Fortunately, quasi-potential functions generalize the concept of a potential function for use in non-gradient systems. A non-gradient system's quasi-potential,  $\Phi(x, y)$ , possesses many of the properties of a gradient system's potential function; in particular, a non-gradient system's quasi-potential is related to its stationary probability distribution in the same way that a gradient system's potential function is related to its stationary probability distribution. Furthermore,  $\Phi(x, y)$  is a Lyapunov function for the deterministic skeleton of a non-gradient system, just as a potential function is for a gradient system. Therefore, the surface  $z = \Phi(x, y)$  is a highly useful stability metric. The quasi-potential also provides information about the expected frequency of transitions between basins of attraction and the expected time required to escape each basin.

The mathematical definition of the quasi-potential is rather involved. We refer readers to Cameron (2012), Nolting and Abbott (2016), and the references therein for the technical construction.

In both this paper and in package **QPot**, the function that we refer to as the quasi-potential is  $\frac{1}{2}$  times the quasi-potential as defined by Freidlin and Wentzell (2012). This choice is made so that the quasi-potential will agree with the potential in gradient systems.

**QPot** is an R package that contains tools for calculating and analyzing quasi-potentials (which, for the special case of gradient systems, are simply potentials). The following three examples show how to use the tools in this package. The first example is a simple consumer-resource model from ecology. This example is explained in detail, starting with the analysis of the deterministic skeleton, proceeding with simulation of the stochastic system, and finally demonstrating the calculation, analysis, and interpretation of the quasi-potential. The second and third examples are covered in less detail, but illustrate some special system behaviors. Systems with limit cycles, like Example 2, require a slightly different procedure than systems that only have point attractors. Extra care must be taken constructing global quasi-potentials for exotic systems, like Example 3.

## Example 1: A consumer-resource model with alternative stable states

Consider the stochastic version (*sensu* (2)) of a standard consumer-resource model of plankton ( $X$ ) and their consumers ( $Y$ ) (Collie and Spencer, 1994; Steele and Henderson, 1981):

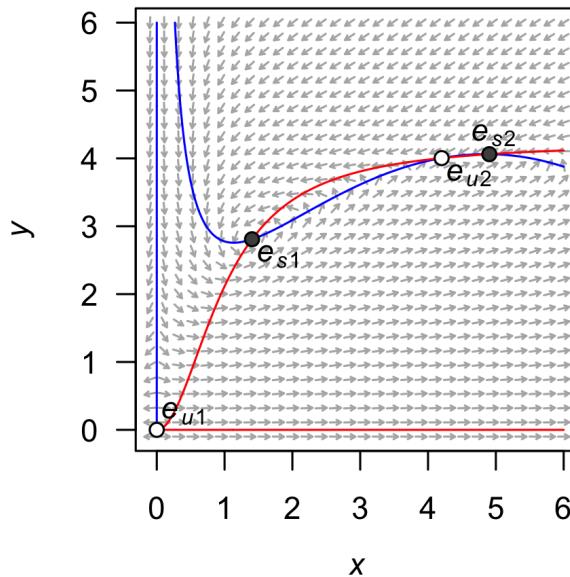
$$\begin{aligned} dX(t) &= \left( \alpha X(t) \left( 1 - \frac{X(t)}{\beta} \right) - \frac{\delta X(t)^2 Y(t)}{\kappa + X(t)^2} \right) dt + \sigma dW_1(t) \\ dY(t) &= \left( \frac{\gamma X(t)^2 Y(t)}{\kappa + X(t)^2} - \mu Y(t)^2 \right) dt + \sigma dW_2(t). \end{aligned} \tag{3}$$

The model is formulated with a Type III functional response; the relationship between the plankton density and the per-capita consumption rate of plankton is sigmoid.  $\alpha$  is the plankton's maximum population growth rate,  $\beta$  is the plankton carrying capacity,  $\delta$  is the maximal feeding rate of the consumers,  $\gamma$  is the maximum conversion rate of plankton to consumer (which takes into account maximum feeding rate),  $\kappa$  controls how quickly the consumption rate saturates, and  $\mu$  is the consumer mortality rate. We will analyze this example with a set of parameter values that yield two stable states:  $\alpha = 1.54$ ,  $\beta = 10.14$ ,  $\gamma = 0.476$ ,  $\delta = 1$ ,  $\kappa = 1$ , and  $\mu = 0.112509$ .

### Step 1: Analyzing the deterministic skeleton

There are preexisting tools in R for analyzing the deterministic skeleton of System (3), which will be described briefly in this subsection. Many of these tools can be found in the CRAN Task View for Differential Equations (<https://CRAN.R-project.org/view=DifferentialEquations>), but we use a select few in our analysis. The first step is to find the equilibria for the system and determine their stability with linear stability analysis. Equilibria can be found using the package **rootSolve** (Soetaert and Herman, 2008). **rootSolve** provides routines that allow users to find roots of nonlinear functions, and perform equilibria and steady-state analysis of ordinary differential equations (ODEs). In Example 1, the equilibria are  $\mathbf{e}_{u1} = (0, 0)$ ,  $\mathbf{e}_{s1} = (1.4049, 2.8081)$ ,  $\mathbf{e}_{u2} = (4.2008, 4.0039)$ ,  $\mathbf{e}_{s2} = (4.9040, 4.0619)$ , and  $\mathbf{e}_{u3} = (10.14, 0)$ . Eigenvalues of the linearized system at an equilibrium can be found by using

`eigen()` in package **base** over the Jacobian matrix (`jacobian.full()` in package **rootSolve**), which determines the asymptotic stability of the system.  $\mathbf{e}_{u1}$  is an unstable source and  $\mathbf{e}_{u2}$  and  $\mathbf{e}_{u3}$  are saddles. The eigenvalues corresponding to  $\mathbf{e}_{s1}$  are  $-0.047 \pm 0.458i$  and the eigenvalues corresponding to  $\mathbf{e}_{s2}$  are  $-0.377$  and  $-0.093$ . Hence  $\mathbf{e}_{s1}$  is a stable spiral point and  $\mathbf{e}_{s2}$  is a stable node. To ease transition from packages such as **deSolve** (Soetaert et al., 2010) and **rootSolve** to our package **QPot**, we include the wrapper function `Model2String()`, which takes a function containing equations and a list of parameters and their values, and returns the equations in a string that is usable by **QPot** (see the help page for an example).



**Figure 1:** A stream plot of the deterministic skeleton of System (3). The blue line is an  $x$ -nullcline (where  $\frac{dx}{dt} = 0$ ) and the red line is a  $y$ -nullcline (where  $\frac{dy}{dt} = 0$ ). Open circles are unstable equilibria and filled circles are stable equilibria. Made using the package **phaseR**.

The package **phaseR** (Grayling, 2014a,b) is an R package for the qualitative analysis of one- and two-dimensional autonomous ODE systems using phase plane methods (including the linear stability analysis described in the preceding paragraph). We use **phaseR** to generate a stream plot of the deterministic skeleton of System (3) (Figure 1). Note that a *stream plot* is a phase plane plot that displays solutions of a system of differential equations; these solutions are also called streamlines. The **deSolve** package can be used to find solutions corresponding to particular initial conditions of the deterministic skeleton of System (3). During the analysis of the deterministic skeleton of a system, it is important to note several things. The first is the range of  $x$  and  $y$  values over which relevant dynamics occur. In Example 1, transitions between the stable equilibria are a primary point of interest, so one might wish to focus on a region like the one displayed in Figure 1, even though this region excludes  $\mathbf{e}_{u3}$ . The regions of phase space that the user finds interesting will determine the window sizes and ranges used later in the quasi-potential calculations. Second, it is important to note if there are any limit cycles. If there are, it will be necessary to identify a point on the limit cycle. This can be accomplished by calculating a long-time solution of the system of ODEs to obtain a trajectory that settles down on the limit cycle (see Example 2). Finally, it is important to note regions of phase space that correspond to unbounded solutions. As explained in subsequent sections, it is worth examining system behavior in negative phase space, even in cases where negative quantities lack physical meaning.

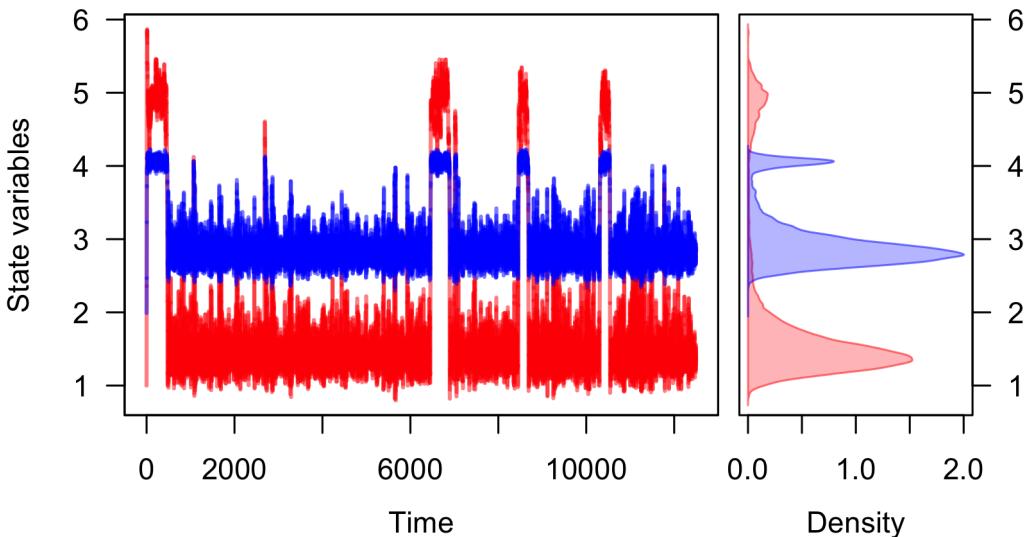
## Step 2: Stochastic simulation

**QPot** contains several tools for generating and visualizing realizations of systems of the form (2). Examining these realizations can help users understand qualitative features of the system before computing and analyzing the quasi-potential. **Sim.DiffProc** (Guidoum and Boukhetala, 2016) and **yuima** (Brouste et al., 2014) are two packages that offer a full suite of stochastic differential equation simulation options, and many of the tools that they contain are more efficient than those in **QPot**. Users interested in very large-scale simulation are encouraged to seek out those packages. For general exploration of a model's behavior prior to quasipotential analysis, however, `TSTraj()` is extremely helpful and does not require the use of a separate package.

Here, we show how to use **QPot** to obtain realizations of System (3) for a specified level of noise intensity,  $\sigma$ . To do this, `TSTraj()` in **QPot** implements the Euler-Maruyama method. All other

code/function references hereafter are found in **QPot**, unless specified otherwise. To generate a realization, the following arguments are required: the right-hand side of the deterministic skeleton for both equations, the initial conditions  $(x_0, y_0)$ , the parameter values, the step-size  $\Delta t$ , and the total time length  $T$ . The function `TSTraj()` accepts strings of equations with the parameter values already included (supplied by the user or made with `Model2String()`) or can combine the equations with the parameter values supplied as `parms`. We supply the function `Model2String()` to replace parameters with their values in an equation, but the user can also input the values themselves and may need to do so with complicated equations (see the `Model2String()` help page). Using `Model2String()` will allow a user to catch problems before they cause complications in the C code within function `QPotential()`.

```
var.eqn.x <- "(alpha * x) * (1 - (x / beta)) - ((delta * (x^2) * y) / (kappa + (x^2)))"
var.eqn.y <- "((gamma * (x^2) * y) / (kappa + (x^2))) - mu * (y^2)"
model.parms <- c(alpha = 1.54, beta = 10.14, delta = 1, gamma = 0.476,
  kappa = 1, mu = 0.112509)
parms.eqn.x <- Model2String(var.eqn.x, parms = model.parms)
## Do not print to screen.
parms.eqn.y <- Model2String(var.eqn.y, parms = model.parms, suppress.print = TRUE)
model.state <- c(x = 1, y = 2)
model.sigma <- 0.05
model.time <- 1000      # we used 12500 in the figures
model.deltat <- 0.025
ts.ex1 <- TSTraj(y0 = model.state, time = model.time, deltat = model.deltat,
  x.rhs = parms.eqn.x, y.rhs = parms.eqn.y, sigma = model.sigma)
## Could also use TSTraj to combine equation strings and parameter values.
## ts.ex1 <- TSTraj(y0 = model.state, time = model.time, deltat = model.deltat,
##   x.rhs = var.eqn.x, y.rhs = var.eqn.y, parms = model.parms, sigma = model.sigma)
```



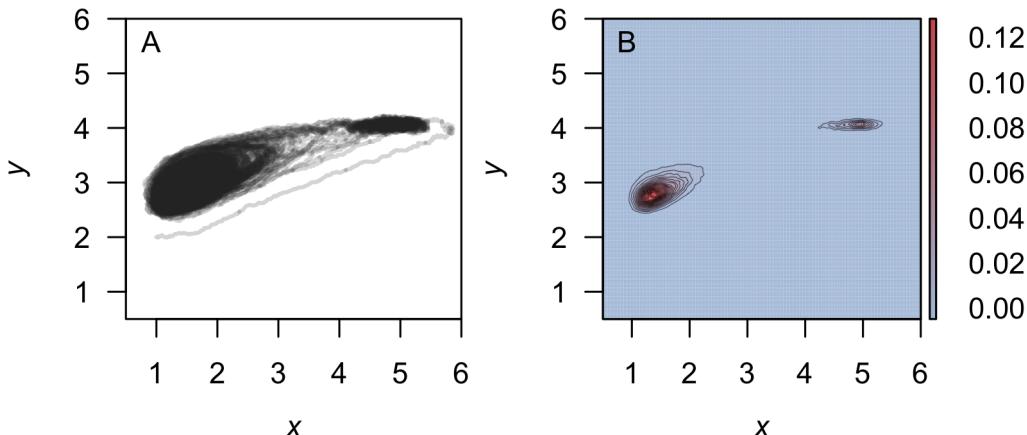
**Figure 2:** A realization of System (3) created using `TSPlot()`, with  $x$  in blue and  $y$  in red. The left panel shows the time series. The right panel, which is enabled with the default `dens = TRUE`, shows a histogram of the  $x$  and  $y$  values over the entire realization.

Figure 2 shows a realization for  $\sigma = 0.05$ ,  $\Delta t = 0.025$ ,  $T = 1.25 \times 10^4$ , and initial condition  $(x_0, y_0) = (1, 2)$ . The argument `dim = 1` produces a time series plot with optional histogram side-plot. The `dim = 2` produces a plot of a realization in  $(x, y)$ -space. If the system is ergodic, a very long realization will approximate the steady-state probability distribution. Motivated by this, a probability density function can be approximated from a long realization using the `TSDensity()` function (e.g., Figure 3b).

<code>TSPlot(ts.ex1, deltat = model.deltat)</code>	# Figure 2
<code>TSPlot(ts.ex1, deltat = model.deltat, dim = 2)</code>	# Figure 3a
<code>TSDensity(ts.ex1, dim = 1)</code>	# like Figure 2 histogram
<code>TSDensity(ts.ex1, dim = 2)</code>	# Figure 3b

Bounds can be placed on the state variables in all of the functions described in this subsection. For example, it might be desirable to set 0 as the minimum size of a biological population, because

negative population densities are not physically meaningful. A lower bound can be imposed on the functions described in this subsection with the argument `lower.bound` in the function `TSTraj()`. Similarly, it might be desirable to set an upper bound for realizations, and hence prevent runaway trajectories (unbounded population densities are also not physically meaningful). An upper bound can be imposed on the functions described in this subsection with the argument `upper.bound`.



**Figure 3:** (A) The realization of System (3) created using `TSPlot()` plotted in  $(x, y)$ -space with `dim = 2`. (B) A density plot obtained from a realization of System (3) using the function `TSDensity()` with `dim = 2`. Red corresponds to high density, and blue to low density.

### Step 3: Local quasi-potential calculation

The next step is to compute a local quasi-potential for each attractor. Because **QPot** deals with two-dimensional systems, *attractor* will be used synonymously with *stable equilibrium* or *stable limit cycle*. A limit cycle will be considered in example 2. For now, suppose that the only attractors are stable equilibrium points,  $\mathbf{e}_{si}, i = 1, \dots, n$ . In the example above,  $n = 2$ . For each stable equilibrium  $\mathbf{e}_{si}$ , we will compute a local quasi-potential  $\Phi_i(x, y)$ .

In order to understand the local quasi-potential, it is useful consider the analogy of a particle traveling according to System (2). In the context of Example 1, the coordinates of the particle correspond to population densities, and the particle's path corresponds to how those population densities change over time. The deterministic skeleton of (2) can be visualized as a force field influencing the particle's trajectory. Suppose that the particle moves along a path from a stable equilibrium  $\mathbf{e}_{si}$  to a point  $(x, y)$ . If this path does not coincide with a solution of the deterministic skeleton, then the stochastic terms must be doing some work to move the particle along the path. The more work is required, the less likely it is for the path to be a realization of System (2).  $\Phi_i(x, y)$  is the amount of work required to traverse the easiest path from  $\mathbf{e}_{si}$  to  $(x, y)$ . Note that  $\Phi_i(x, y)$  is non-negative, and it is zero at  $\mathbf{e}_{si}$ .

In the basin of attraction for  $\mathbf{e}_{si}$ ,  $\Phi_i(x, y)$  has many properties analogous to the potential function for gradient systems. Key among these properties is that the quasi-potential is non-increasing along deterministic trajectories. This means that the quasi-potential can be interpreted as a type of energy surface, and the rolling ball metaphor is still valid. The difference is that, in non-gradient systems, there is an additional component to the vector field that causes trajectories to circulate around level sets of the energy surface. This is discussed in more detail in Step 6, below.

**QPot** calculates quasi-potentials using an adjustment developed by Cameron (2012) to the ordered upwind algorithm (Sethian and Vladimirsky, 2001, 2003). The idea behind the algorithm is to calculate  $\Phi_i(x, y)$  in ascending order, starting with the known point  $\mathbf{e}_{si}$ . The result is an expanding area where the solution is known.

Calculating  $\Phi_i(x, y)$  with the function `QPotential()` requires a text string of the equations and parameter values, the stable equilibrium points, the computation domain, and the mesh size. If the equations do not contain the parameter values, the function `Model2String()` can be used to insert the values into the equations, as presented above. For (3), this first means inputting the equations:

$$\begin{aligned} f_1(x, y) &= 1.54x \left(1 - \frac{x}{10.14}\right) - \frac{x^2 y}{1 + x^2} \\ f_2(x, y) &= \frac{0.476 x^2 y}{1 + x^2} - 0.112509 y^2. \end{aligned}$$

In R:

```
## If not done in a previous step.
parms.eqn.x <- Model2String(var.eqn.x, parms = model.parms)
## Do not print to screen.
parms.eqn.y <- Model2String(var.eqn.y, parms = model.parms, suppress.print = TRUE)
## Could also input the values by hand and use this version.
## parms.eqn.x <- "1.54 * x * (1.0 - (x / 10.14)) - (y * (x^2)) / (1.0 + (x^2))"
## parms.eqn.y <- "((0.476 * (x^2) * y) / (1 + (x^2))) - 0.112509 * (y^2)"
```

The coordinates of the points  $\mathbf{e}_{si}$ , which were determined in Step 1, are  $\mathbf{e}_{s1} = (1.4049, 2.8081)$  and  $\mathbf{e}_{s2} = (4.9040, 4.0619)$ .

```
eq1.x <- 1.40491
eq1.y <- 2.80808
eq2.x <- 4.9040
eq2.y <- 4.06187
```

Next, the boundaries of the computational domain need to be entered. This domain will be denoted by  $[Lx_1, Lx_2] \times [Ly_1, Ly_2]$ . The ordered-upwind method terminates when the solved area encounters a boundary of this domain. Thus, it is important to choose boundaries carefully. For example, if  $\mathbf{e}_{si}$  lies on one of the coordinate axes, one should not use that axis as a boundary because the algorithm will immediately terminate. Instead, one should add padding space. This is important even if the padding space corresponds to physically unrealistic values (e.g., negative population densities). For this example, a good choice of boundaries is:  $Lx_1 = Ly_1 = -0.5$ , and  $Lx_2 = Ly_2 = 20$ . This choice of domain was obtained by examining stream plots of the deterministic skeleton and density plots of stochastic realizations (Figures 1–3). The domain contains all of the deterministic skeleton equilibria, and it encompasses a large area around the regions of phase space visited by stochastic trajectories (Figures 1–3). Note that a small padding space was added to the left and bottom sides of the domain, so that the coordinate axes are not the domain boundaries.

```
bounds.x <- c(-0.5, 20.0)
bounds.y <- c(-0.5, 20.0)
```

In some cases, it may be desirable to treat boundaries differently in the upwind algorithm. This is addressed below in Section 6.7.

Finally, the mesh size for the discretization of the domain needs to be specified. Let  $N_x$  be the number of grid points in the  $x$ -direction and  $N_y$  be the number of grid points in the  $y$ -direction. Note that the horizontal distance between mesh points is  $h_x = \frac{Lx_2 - Lx_1}{N_x}$ , and the vertical distance between mesh points is  $h_y = \frac{Ly_2 - Ly_1}{N_y}$ . Mesh points are considered adjacent if their Euclidean distance is less than or equal to  $h = \sqrt{h_x^2 + h_y^2}$ . This means that diagonal mesh points are considered adjacent. In this example, a good choice is  $N_x = N_y = 4100$ . This means that  $h_x = h_y = 0.005$ , and  $h \approx 0.00707$ . In general, the best choice of mesh size will be a compromise between resolution and computational time. The mesh size must be fine enough to precisely track how information moves outward along characteristics from the initial point. Too fine of a mesh size can lead to very long computational times, though. The way that computation time scales with grid size depends on the system under consideration (see below for computation time for this example), because the algorithm ends when it reaches a boundary, which could occur before the algorithm has exhaustively searched the entire mesh area.

```
step.number.x <- 1000
step.number.y <- 1000 # we used 4100 in the figures
```

The update radii factors,  $K_x$  and  $K_y$ , are two other adjustable parameters for the algorithm. These are `k.x` and `k.y` in `QPotential()`. These two parameters determine the neighborhood of points that can be used to update a given point.  $K_x$  and  $K_y$  are the distances (measured in mesh units) in the  $x$  and  $y$  direction that bound this neighborhood for any given point. The selection of the best values for these parameters involves several nuanced considerations. For a discussion of these issues, please see Cameron (2012). For users who wish to avoid these details, we suggest using the defaults  $K_x = 20$  and  $K_y = 20$ .

The R interface implements the `QPotential()` algorithm using C code. By default `QPotential()` outputs a matrix that contains the quasi-potentials to the R session. The time required to compute the quasi-potential will depend on the size of the region and the fineness of the mesh. This example with  $K_x = K_y = 20$  and  $N_x = N_y = 4100$  has approximately  $1.7 \times 10^7$  grid points, which leads to run times of approximately 2.25 min (2.5 GHz Intel Core i5 processor and 8 GB 1600 MHz DDR3 memory). When one reaches around  $5 \times 10^8$  grid points, computational time can be several hours.

Setting the argument `save.to.R` to TRUE (default) outputs the matrix into the R session, and setting the argument `save.to.HD` to TRUE saves the matrix to the hard drive as a tab-delimited text file `filename` in the current working directory. For  $N_x = N_y = 4100$ , the saved file occupies 185 MB.

```
eq1.local <- QPotential(x.rhs = parms.eqn.x, x.start = eq1.x, x.bound = bounds.x,
x.num.steps = step.number.x, y.rhs = parms.eqn.y, y.start = eq1.y,
y.bound = bounds.y, y.num.steps = step.number.y)
```

Step 3 should be repeated until local quasi-potentials  $\Phi_i(x, y)$  have been obtained for each  $\mathbf{e}_{si}$ . In Example 1, this means calculating  $\Phi_1(x, y)$  corresponding to  $\mathbf{e}_{s1}$  and  $\Phi_2(x, y)$  corresponding to  $\mathbf{e}_{s2}$ .

```
eq2.local <- QPotential(x.rhs = parms.eqn.x, x.start = eq2.x, x.bound = bounds.x,
x.num.steps = step.number.x, y.rhs = parms.eqn.y, y.start = eq2.y,
y.bound = bounds.y, y.num.steps = step.number.y)
```

Each local quasi-potential  $\Phi_i(x, y)$  is stored in R as a large matrix. The entries in this matrix are the values of  $\Phi_i$  at each mesh point. To define the function on the entire domain (i.e., to allow it to be evaluated at arbitrary points in the domain, not just the discrete mesh points), bilinear interpolation is used. The values of  $\Phi(x, y)$  can be extracted using the function `QPIInterp()`. Inputs to `QPIInterp()` include the  $(x, y)$  coordinates of interest, the  $(x, y)$  domain boundaries, and the `QPotential()` output (i.e., the matrix with rows corresponding to  $x$ -values and columns corresponding to  $y$ -values). `QPIInterp()` can be used for any of the local quasi-potential or the global quasi-potential surfaces (see the next subsection).

#### Step 4: Global quasi-potential calculation

Recall that  $\Phi_i(x, y)$  is the amount of work required to travel from  $\mathbf{e}_{si}$  to  $(x, y)$ . This information is useful for considering dynamics in the basin of attraction of  $\mathbf{e}_{si}$ . In many cases, however, it is desirable to define a global quasi-potential that describes the system's dynamics over multiple basins of attraction. If a gradient system has multiple stable states, the potential function provides an energy surface description that is globally valid. We seek an analogous global function for non-gradient systems. Achieving this requires *pasting* local quasi-potentials into a single global quasi-potential. If the system has only two attractors, one can define a global quasi-potential, though it might be nontrivial (see Example 3 ahead). In systems with three or more attractors such a task might not be possible (Freidlin and Wentzell, 2012). For a wide variety of systems, however, a relatively simple algorithm can accomplish the pasting (Graham and Tél, 1986; Roy and Nauman, 1995). In most cases, the algorithm amounts to translating the local quasi-potentials up or down so that they agree at the saddle points that separate the basins of attraction. In Example 1,  $\mathbf{e}_{u1}$  lies on the boundary of the basins of attraction for  $\mathbf{e}_{s1}$  and  $\mathbf{e}_{s2}$ . Creating a global quasi-potential requires matching  $\Phi_1$  and  $\Phi_2$  at  $\mathbf{e}_{u2}$ .  $\Phi_1(\mathbf{e}_{u2}) = 0.007056$  and  $\Phi_2(\mathbf{e}_{u2}) = 0.00092975$ . If one defines

$$\Phi_2^*(x, y) = \Phi_2(x, y) + (0.007056 - 0.00092975) = \Phi_2(x, y) + 0.00612625,$$

then  $\Phi_1$  and  $\Phi_2^*$  match at  $\mathbf{e}_{u2}$ . Finally, define

$$\Phi(x, y) = \min(\Phi_1(x, y), \Phi_2^*(x, y)),$$

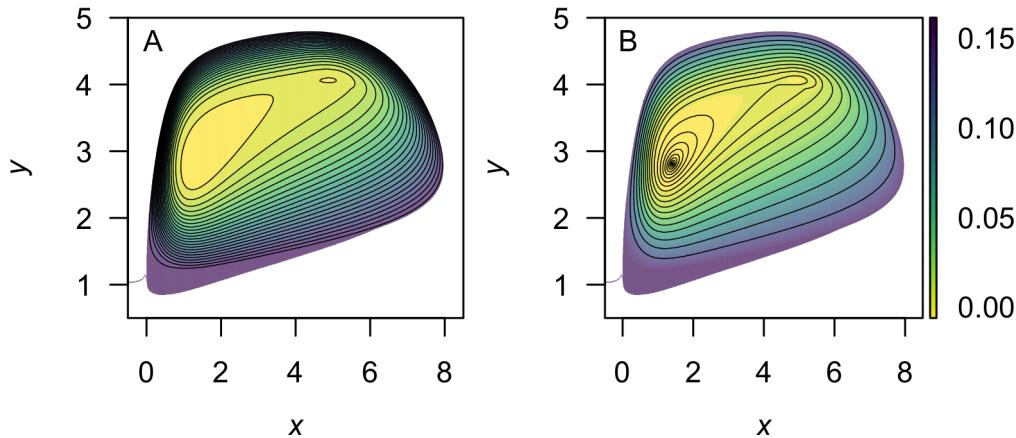
which is the global quasi-potential. For systems with more than two stable equilibria, this process is generalized to match local quasi-potentials at appropriate saddles. `QPot` automates this procedure. A fuller description of the underlying algorithm is explained in Example 3, which requires a more nuanced understanding of the pasting procedure.

```
ex1.global <- QPGlobal(local.surfaces = list(eq1.local, eq2.local),
unstable.eq.x = c(0, 4.2008), unstable.eq.y = c(0, 4.0039),
x.bound = bounds.x, y.bound = bounds.y)
```

This function `QPGlobal()` calculates the global quasi-potential by automatically pasting together the local quasi-potentials. This function requires the input of all the discretized local quasi-potentials, and the coordinates of all unstable equilibria. The output is a discretized version of the global quasi-potential. The length of time required for this computation will depend on the total number of mesh points; for the parameters used in Example 1, it takes a couple of minutes. As with the local quasi-potentials, the values of  $\Phi(x, y)$  can be extracted using the function `QPIInterp()`.

#### Step 5: Global quasi-potential visualization

To visualize the global quasi-potential, one can simply take the global quasi-potential matrix from `QPGlobal()` and use it to create a contour plot using `QPContour()` (Figure 4).



**Figure 4:** A contour plot of the quasi-potential of System (3). Yellow corresponds to low values of the quasi-potential, and purple to high values. The `c.parm` parameter in `QPContour()` can be used to generate non-equal contour spacing (e.g., for finer resolution near equilibria). The default creates evenly spaced contour lines ((A); `c.parm = 1`). In (B), contour lines are concentrated at the bottom of the basin (`c.parm = 5`). Default plot colors are generated from package `viridis` (Garnier, 2016), by setting `col.contour = viridis(n = 25, option = "D")` in `QPContour()`.

```
QPContour(surface = ex1.global, dens = c(1000, 1000), x.bound = bounds.x,
y.bound = bounds.y, c.parm = 5) # right side of Figure 4
```

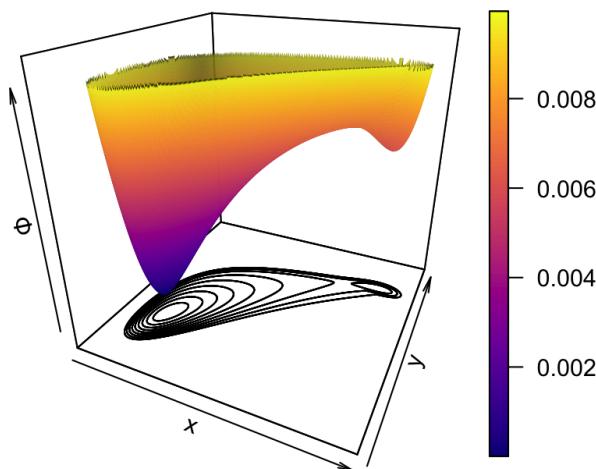
`QPContour()` is based on the `.filled.contour()` function from the base package **graphics**. In most cases, the mesh sizes used for the quasi-potential calculation will be much finer than what is required for useful visualization. The argument `dens` within `QPContour()` reduces the points used in the graphics generation. Although it might seem wasteful to perform the original calculations at a mesh size that is finer than the final visualization, this is not so. Choosing the mesh size in the original calculations to be very fine reduces the propagation of errors in the ordered upwind algorithm, and hence leads to a more accurate numerical solution.

An additional option allows users to specify contour levels. R's default for the `contour()` function creates contour lines that are equally spaced over the range of values specified by the user. In some cases, however, it is desirable to use a non-equidistant spacing for the contours. For example, equally-spaced contours will not capture the topography at the bottom of a basin if the changes in height are much smaller than in other regions in the plot. Simply increasing the number of equally-spaced contour lines does not solve this problem, because steep areas of the plot become completely saturated with lines. `QPContour()` has a function for non-equidistant contour spacing that condenses contour lines at the bottoms of basins. Specifically, for  $n$  contour lines, this function generates a list of contour levels,  $\{v_i\}_{i=1}^n$ , specified by:

$$v_i = \max(\Phi) \left( \frac{i-1}{n-1} \right)^c.$$

$c = 1$  yields evenly-spaced contours. As  $c$  increases, the contour lines become more concentrated near basin bottoms. Figure 4 shows equal contour lines (left panel) and contour lines that are concentrated at the bottom of the basin (right panel, `c.parm = 5`).

Finally, creating a 3D plot can be very useful for visualizing the features of more complex surfaces. This is especially helpful when considering the physical metaphor of a ball rolling on a surface specified by a quasi-potential (Nolting and Abbott, 2016). R has several packages for 3D plotting, including static plotting with the base function `persp()` and with the package `plot3D` (Soetaert, 2014). Interactive plotting is provided by `rgl` (Adler et al., 2015). To create an interactive 3D plot for Example 1 using `rgl`, use the code: `persp3d(x = ex1.global)`. Figure 5 shows a 3D plot of example 1 using `persp3d(z = ex1.global)` in `plot3D` that clearly illustrates the differences between the two local basins. Users can also export the matrix of quasi-potential values and create 3D plots in other programs.



**Figure 5:** A 3D plot of the quasi-potential of System (3) using `persp3D()` in package `plot3D`. 3D plotting can further help users visualize the quasi-potential surfaces. Plot colors are generated from package `viridis` by setting `col = viridis(n = 100, option = "A")` and `contour = TRUE`.

### Step 6: Vector field decomposition

Recall that the deterministic skeleton (1) can be visualized as a vector field, as shown in Figure 1. In gradient systems, this vector field is completely determined by the potential function,  $V(x, y)$ . The name *gradient system* refers to the fact that the vector field is the negative of the potential function's gradient,

$$\begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix} = -\nabla V(x, y) = -\begin{bmatrix} \frac{\partial V}{\partial x}(x, y) \\ \frac{\partial V}{\partial y}(x, y) \end{bmatrix}.$$

In non-gradient systems, the vector field can no longer be represented solely in terms of the gradient of  $\Phi(x, y)$ . Instead, there is a remainder component of the vector field,  $\mathbf{r}(x, y) = \begin{bmatrix} r_1(x, y) \\ r_2(x, y) \end{bmatrix}$ . The vector field can be decomposed into two terms:

$$\begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix} = -\nabla\Phi(x, y) + \mathbf{r}(x, y) = -\begin{bmatrix} \frac{\partial\Phi}{\partial x}(x, y) \\ \frac{\partial\Phi}{\partial y}(x, y) \end{bmatrix} + \begin{bmatrix} r_1(x, y) \\ r_2(x, y) \end{bmatrix}.$$

The remainder vector field is orthogonal to the gradient of the quasi-potential everywhere. That is, for every  $(x, y)$  in the domain,

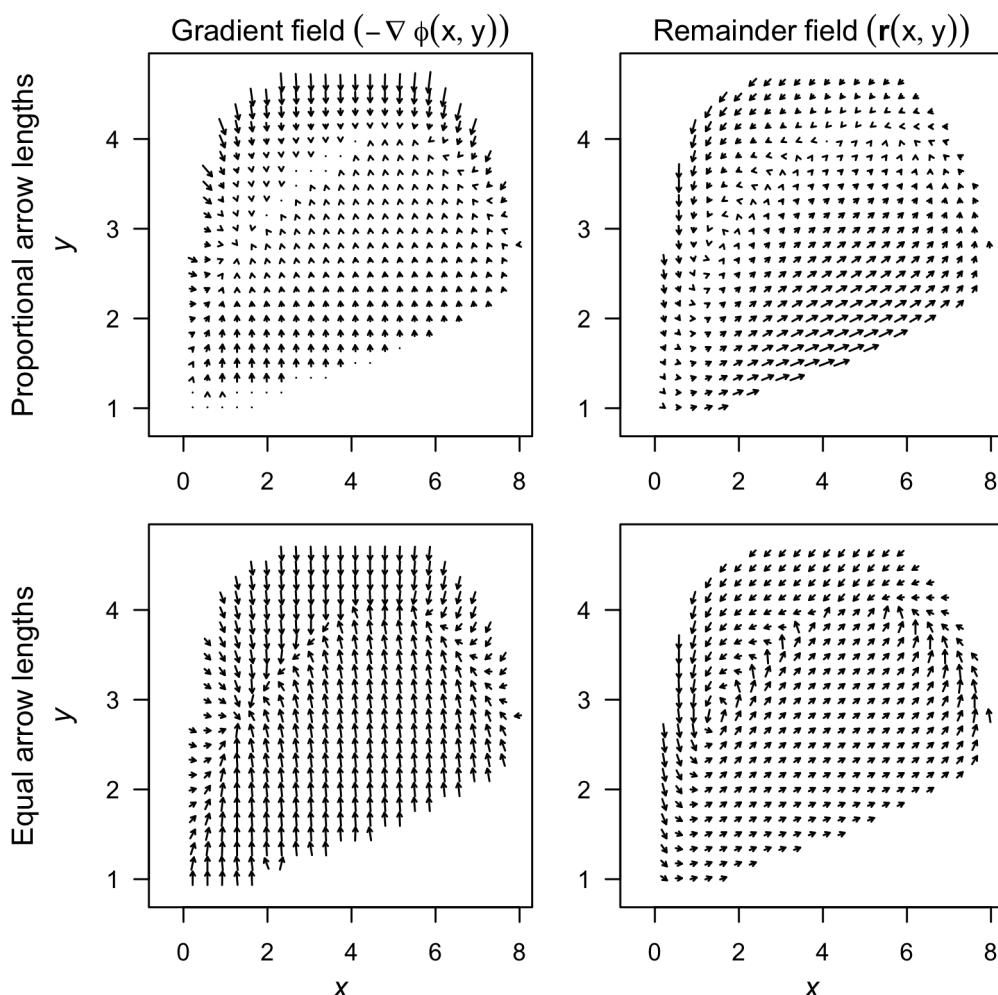
$$\nabla\Phi(x, y) \cdot \mathbf{r}(x, y) = 0.$$

An explanation of this property can be found in Nolting and Abbott (2016).

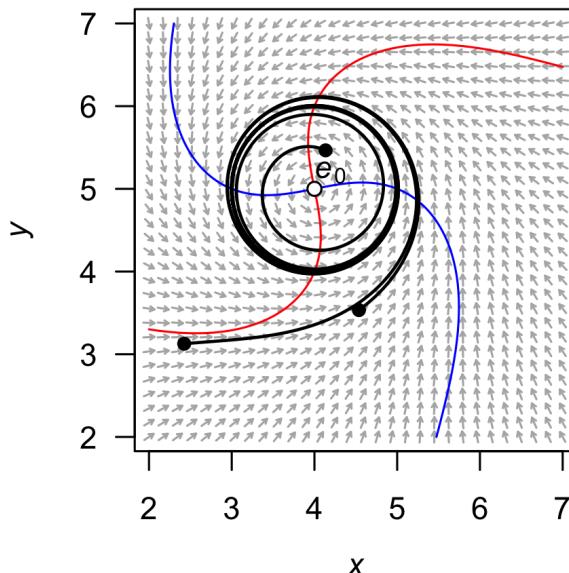
The remainder vector field can be interpreted as a force that causes trajectories to circulate around level sets of the quasi-potential. **QPot** enables users to perform this decomposition. The function `VecDecomAll()` calculates the vector field decomposition, and outputs three vector fields: the original deterministic skeleton,  $\mathbf{f}(x, y)$ ; the gradient vector field,  $-\nabla\Phi(x, y)$ ; and the remainder vector field,  $\mathbf{r}(x, y)$ . Each of these three vector fields can be output alone using `VecDecomVec()`, `VecDecomGrad()`, or `VecDecomRem()`. These vector fields can be visualized using the function `VecDecomPlot()`. Code to create the vector fields from `VecDecomAll()` is displayed below; code for generating individual vector fields can be found in the man pages accessible by `help()` for `VecDecomVec()`, `VecDecomGrad()`, or `VecDecomRem()`. The gradient and remainder vector fields are shown in the left and right columns of Figure 6, respectively, with proportional vectors (top row) and equal-length vectors (bottom row). Three arguments within `VecDecomPlot()` are important to creating comprehensible plots: `dens`, `tail.length`, and `head.length`. The `dens` parameter specifies the number of arrows in the plot window along the  $x$  and  $y$  axes. The argument `tail.length` scales the length of arrow tails. The argument `head.length` scales the length of arrow heads. The function `arrows()` makes up the base of `VecDecomPlot()`, and arguments can be passed to it, as well as to `plot()`. The code below produces all

three vector fields from the multi-dimensional array returned by `VecDecomAll()`:

```
## Calculate all three vector fields.
VDA11 <- VecDecomAll(surface = ex1.global, x.rhs = parms.eqn.x, y.rhs = parms.eqn.y,
  x.bound = bounds.x, y.bound = bounds.y)
## Plot the deterministic skeleton vector field.
VecDecomPlot(x.field = VDA11[, , 1], y.field = VDA11[, , 2], dens = c(25, 25),
  x.bound = bounds.x, y.bound = bounds.y, xlim = c(0, 11), ylim = c(0, 6),
  arrow.type = "equal", tail.length = 0.25, head.length = 0.025)
## Plot the gradient vector field.
VecDecomPlot(x.field = VDA11[, , 3], y.field = VDA11[, , 4], dens = c(25, 25),
  x.bound = bounds.x, y.bound = bounds.y, arrow.type = "proportional",
  tail.length = 0.25, head.length = 0.025)
## Plot the remainder vector field.
VecDecomPlot(x.field = VDA11[, , 5], y.field = VDA11[, , 6], dens = c(25, 25),
  x.bound = bounds.x, y.bound = bounds.y, arrow.type = "proportional",
  tail.length = 0.35, head.length = 0.025)
```



**Figure 6:** The gradient (left column) and remainder (right column) fields, plotted with `arrow.type = "proportional"` (top row) and `arrow.type = "equal"` (bottom row) arrow lengths using `VecDecomPlot()` for System (3).



**Figure 7:** A stream plot of the deterministic skeleton of System (4). The blue line is an  $x$ -nullcline (where  $\frac{dy}{dt} = 0$ ) and the red line is a  $y$ -nullcline (where  $\frac{dx}{dt} = 0$ ). The open circle is an unstable equilibrium. Particular solutions are shown as black lines, with filled circles as initial conditions. Made using the package **phaseR**.

## Example 2: A model with a limit cycle

Consider the following model:

$$\begin{aligned} dX(t) &= \left( -(Y(t) - \beta) + \mu (X(t) - \alpha) \left( 1 - (X(t) - \alpha)^2 - (Y(t) - \beta)^2 \right) \right) dt + \sigma dW_1(t) \\ dY(t) &= \left( (X(t) - \alpha) + \mu (Y(t) - \beta) \left( 1 - (X(t) - \alpha)^2 - (Y(t) - \beta)^2 \right) \right) dt + \sigma dW_2(t). \end{aligned} \quad (4)$$

This model will demonstrate **QPot**'s ability to handle limit cycles. We will analyze this example with  $\mu = 0.2$ ,  $\alpha = 4$ , and  $\beta = 5$ .

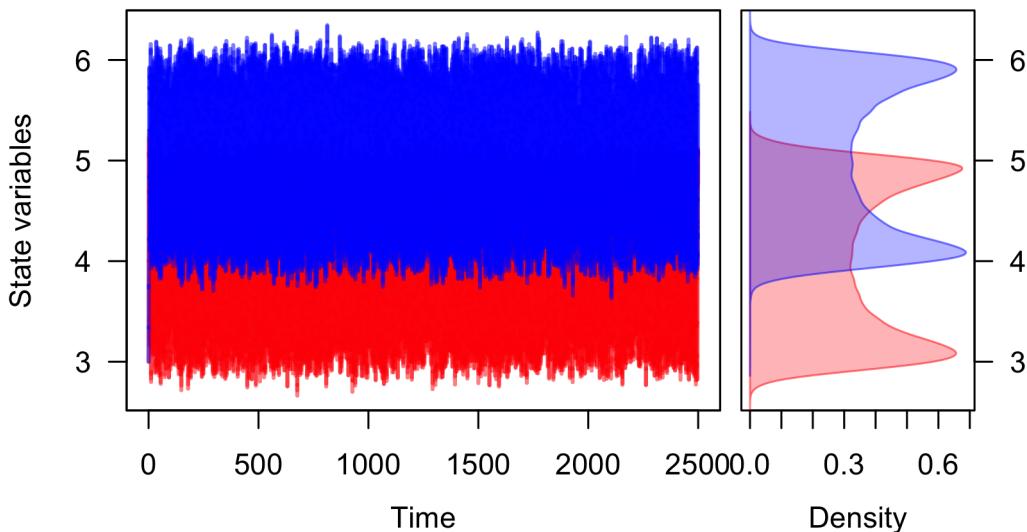
### Step 1: Analyzing the deterministic skeleton

The deterministic skeleton of this system has one equilibrium,  $e_0 = (4, 5)$ , which is an unstable spiral point. Figure 7 shows a stream plot of the deterministic skeleton of System (4). A particular solution of the deterministic skeleton of System (4) can be found using **rootSolve** and **deSolve**. The stream plot and a few particular solutions suggest that there is a stable limit cycle. To calculate the limit cycle, one can find a particular solution over a long time interval (e.g., Figure 7 has three trajectories run for  $T = 100$ ). The solution will eventually converge to the limit cycle. One can drop the early part of the trajectory until only the closed loop of the limit cycle remains. There are more elegant ways to numerically find a periodic orbit (even when those orbits are unstable). For more information on these methods, see Chua and Parker (1989). In this example, the limit cycle is shown by the thick black line in Figure 7. For calculation of the quasi-potential, it is sufficient to input a single point that lies on the limit cycle. For this example, one such point is  $z = (4.15611, 5.98774)$ .

### Step 2: Stochastic simulation

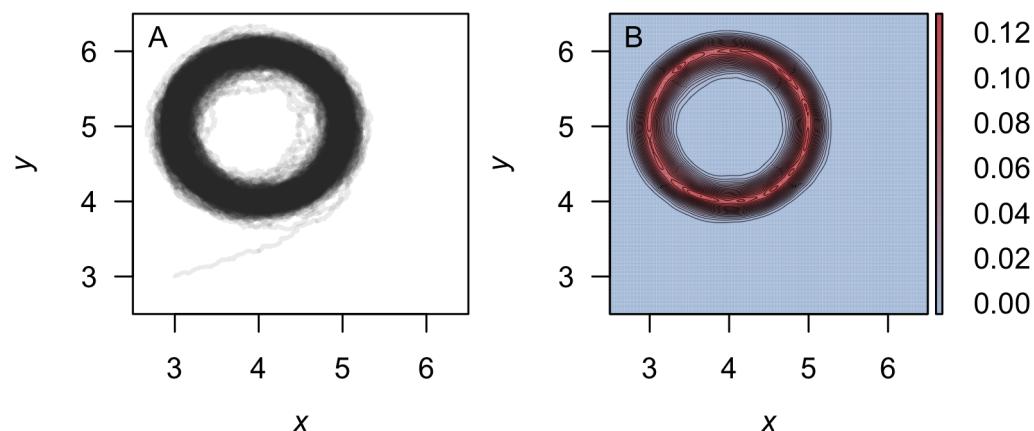
Figures 8 and 9a show a time series for a realization of (4) with  $\sigma = 0.1$ ,  $\Delta t = 5 \times 10^{-3}$ ,  $T = 2500$  and initial condition  $(x_0, y_0) = (3, 3)$ . Figure 9b shows a density plot of a realization with the same parameters, except  $T = 2.5 \times 10^3$ .

```
var.eqn.x <- "- (y - beta) + mu * (x - alpha) * (1 - (x - alpha)^2 - (y - beta)^2)"
var.eqn.y <- "(x - alpha) + mu * (y - beta) * (1 - (x - alpha)^2 - (y - beta)^2)"
model.state <- c(x = 3, y = 3)
model.parms <- c(alpha = 4, beta = 5, mu = 0.2)
model.sigma <- 0.1
model.time <- 1000 # we used 2500 in the figures
```



**Figure 8:** A realization of System (4) created using `TSPlot()`, with  $x$  in blue and  $y$  in red. The left side of (a) shows the time series. The right side of (a), which is enabled with the default `dens = TRUE`, shows a histogram of the  $x$  and  $y$  values over the entire realization.

```
model.deltat <- 0.005
ts.ex2 <- TSTraj(y0 = model.state, time = model.time, deltat = model.deltat,
  x.rhs = var.eqn.x, y.rhs = var.eqn.y, parms = model.parms, sigma = model.sigma)
TSPlot(ts.ex2, deltat = model.deltat) # Figure 8
TSPlot(ts.ex2, deltat = model.deltat, dim = 2, line.alpha = 25) # Figure 9a
TSDensity(ts.ex2, dim = 1) # Histogram
TSDensity(ts.ex2, dim = 2) # Figure 9b
```



**Figure 9:** (A) The realization of System (4) plotted in  $(x, y)$ -space (`dim = 2` in the function `TSPlot()`) (B) A density plot obtained from a realization of System (4) using `TSDensity()` with `dim = 2`. Red corresponds to high density, and blue to low density.

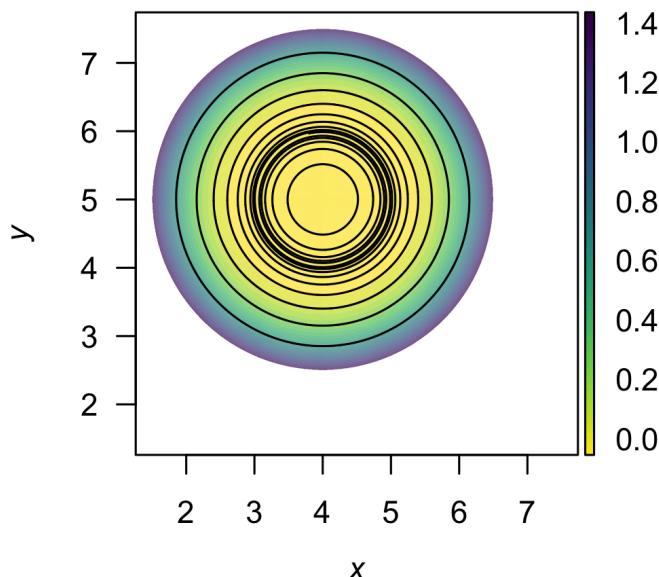
### Step 3: Local quasi-potential calculation

In this example, there are no stable equilibrium points. There is one stable limit cycle, and this can be used to obtain a local quasi-potential. Using  $\mathbf{z}$  as the initial point for the ordered-upwind algorithm and  $Lx_1 = -0.5$ ,  $Ly_1 = -0.5$ ,  $Lx_2 = 7.5$ ,  $Ly_2 = 7.5$ ,  $N_x = 4000$  and  $N_y = 4000$ , one obtains a local quasi-potential,  $\Phi_{\mathbf{z}}(x, y)$ . The following code generates the local quasi-potential  $\Phi_{\mathbf{z}}(x, y)$ :

```

eqn.x <- Model2String(var.eqn.x, parms = model.parms)
eqn.y <- Model2String(var.eqn.y, parms = model.parms)
eq1.qp <- QPotential(x.rhs = eqn.x, x.start = 4.15611, x.bound = c(-0.5, 7.5),
  x.num.steps = 4000, y.rhs = eqn.y, y.start = 5.98774, y.bound = c(-0.5, 7.5),
  y.num.steps = 4000)

```



**Figure 10:** A contour plot of the quasi-potential of System (4) using `QPContour()`. Yellow corresponds to low values of the quasi-potential, and purple to high values.

#### Step 4: Global quasi-potential calculation

There is only one local quasi-potential in this example, so it is the global quasi-potential,  $\Phi(x, y) = \Phi_z(x, y)$ .

#### Step 5: Global quasi-potential visualization

Figure 10 shows a contour plot of the global quasi-potential.

```
QPContour(eq1.qp, dens = c(1000, 1000), x.bound = c(-0.5, 7.5),
  y.bound = c(-0.5, 7.5), c.parm = 10)
```

#### Example 3: More complicated local quasi-potential pasting

In Example 1, the procedure for pasting local quasi-potentials together into a global quasi-potential was a simple, two-step process. First, one of the local quasi-potentials was translated so that the two surfaces agreed at the saddle point separating the two basins of attraction. Second, the global quasi-potential was obtained by taking the minimum of the two surfaces at each point. A general algorithm for pasting local quasi-potentials, as explained in [Graham and Tél \(1986\)](#) and [Roy and Nauman \(1995\)](#), is slightly more complicated. This process is automated in `QPGlobal()`, but it is worth understanding the process in order to correctly interpret the outputs.

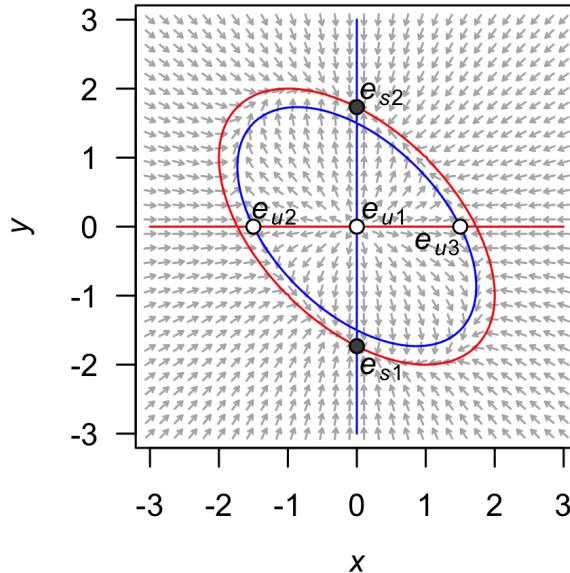
To understand the full algorithm, consider the following model:

$$\begin{aligned} dX(t) &= X(t) \left( (1 + \alpha_1) - X(t)^2 - X(t)Y(t) - Y(t)^2 \right) dt + \sigma dW_1(t) \\ dY(t) &= Y(t) \left( (1 + \alpha_2) - X(t)^2 - X(t)Y(t) - Y(t)^2 \right) dt + \sigma dW_2(t). \end{aligned} \tag{5}$$

For this analysis, let  $\alpha_1 = 1.25$  and  $\alpha_2 = 2$ . We have selected this model because it demonstrates how `QPot` can handle an exceptionally tricky global quasi-potential construction.

### Step 1: Analyzing the deterministic skeleton

The deterministic skeleton of this system has five equilibria. These are  $\mathbf{e}_{u1} = (0, 0)$ ,  $\mathbf{e}_{s1} = (0, -1.73205)$ ,  $\mathbf{e}_{s2} = (0, 1.73205)$ ,  $\mathbf{e}_{u2} = (-1.5, 0)$  and  $\mathbf{e}_{u3} = (1.5, 0)$ . The eigenvalue analysis shows that  $\mathbf{e}_{u1}$  is an unstable node,  $\mathbf{e}_{s1}$  and  $\mathbf{e}_{s2}$  are stable nodes,  $\mathbf{e}_{u2}$  and  $\mathbf{e}_{u3}$  are saddles. Figure 11 shows a stream plot of the deterministic skeleton of (5). The basin of attraction for  $\mathbf{e}_{s1}$  is the lower half-plane, and the basin of attraction for  $\mathbf{e}_{s2}$  is the upper half-plane.



**Figure 11:** A stream plot of the deterministic skeleton of System (5). The blue line is an  $x$ -nullcline (where  $\frac{dx}{dt} = 0$ ) and the red line is a  $y$ -nullcline (where  $\frac{dy}{dt} = 0$ ). Open circles are stable equilibria and filled circles are unstable equilibria. Made using the package **phaseR**.

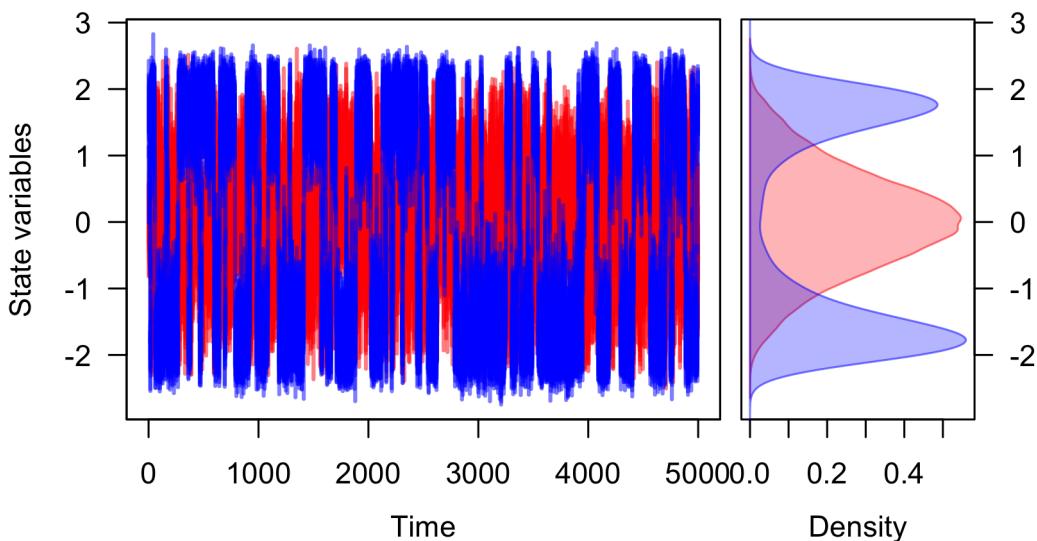
### Step 2: Stochastic simulation

Figures 12 and 13a show a time series for a realization of System (5) with  $\sigma = 0.8$ ,  $\Delta t = 0.01$ ,  $T = 5000$  and initial condition  $(x_0, y_0) = (0.5, 0.5)$ . Figure 13b shows a density plot of this realization.

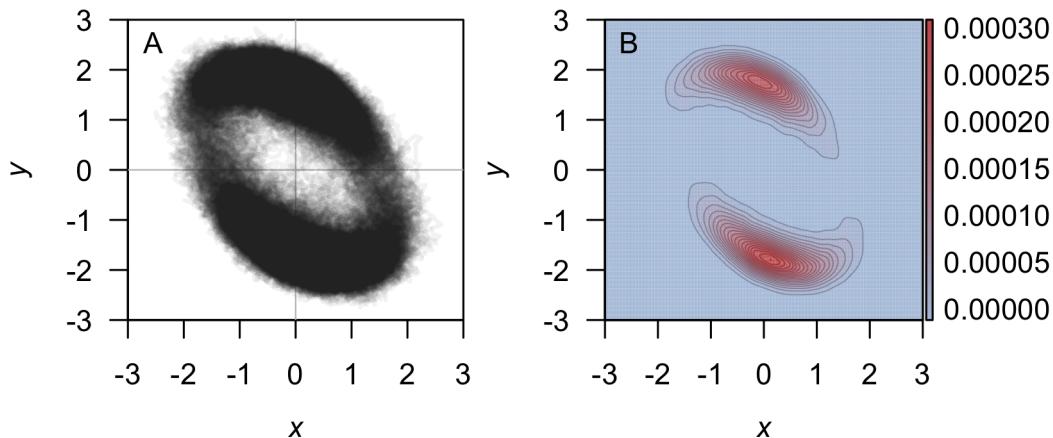
```
var.eqn.x <- "x * ((1 + alpha1) - (x^2) - x * y - (y^2))"
var.eqn.y <- "y * ((1 + alpha2) - (x^2) - x * y - (y^2))"
model.state <- c(x = 0.5, y = 0.5)
model.parms <- c(alpha1 = 1.25, alpha2 = 2)
model.sigma <- 0.8
model.time <- 5000
model.deltat <- 0.01
ts.ex3 <- TSTraj(y0 = model.state, time = model.time, deltat = model.deltat,
  x.rhs = var.eqn.x, y.rhs = var.eqn.y, parms = model.parms, sigma = model.sigma)
TSPlot(ts.ex3, deltat = model.deltat) # Figure 12
TSPlot(ts.ex3, deltat = model.deltat, dim = 2, line.alpha = 25) # Figure 13a
TSDensity(ts.ex3, dim = 1) # Histogram of time series
TSDensity(ts.ex3, dim = 2, contour.levels = 20, contour.lwd = 0.1) # Figure 13b
```

### Step 3: Local quasi-potential calculation

Two local quasi-potentials need to be calculated,  $\Phi_1(x, y)$  corresponding to  $\mathbf{e}_{s1}$ , and  $\Phi_2(x, y)$  corresponding to  $\mathbf{e}_{s2}$ . In both cases, sensible boundary and mesh choices are  $Lx_1 = -3$ ,  $Ly_1 = -3$ ,  $Lx_2 = 3$ ,  $Ly_2 = 3$ ,  $N_x = 6000$ , and  $N_y = 6000$ .



**Figure 12:** A realization of System (5) created using `TSPlot()`, with  $x$  in blue and  $y$  in red. The left panel shows the time series. The right panel, which is enabled by default with parameter `dens = TRUE` in the function `TSPlot()`, shows a histogram of the  $x$  and  $y$  values over the entire realization.



**Figure 13:** (A) The realization of System (5) plotted in  $(x, y)$ -space with `TSPlot()` with `dim = 2`. (B) A density plot obtained from the realization of System (5) by using the function `TSDensity()` with `dim = 2`, `contour.levels = 20`, and `contour.lwd = 0.1`. Red corresponds to high density, and blue to low density.

```
equation.x <- Model2String(var.eqn.x, parms = model.parms)
equation.y <- Model2String(var.eqn.y, parms = model.parms)
bounds.x <- c(-3, 3); bounds.y <- c(-3, 3)
step.number.x <- 6000; step.number.y <- 6000
eq1.x <- 0; eq1.y <- -1.73205
eq2.x <- 0; eq2.y <- 1.73205
eq1.local <- QPotential(x.rhs = equation.x, x.start = eq1.x, x.bound = bounds.x,
  x.num.steps = step.number.x, y.rhs = equation.y, y.start = eq1.y,
  y.bound = bounds.y, y.num.steps = step.number.y)
eq2.local <- QPotential(x.rhs = equation.x, x.start = eq2.x, x.bound = bounds.x,
  x.num.steps = step.number.x, y.rhs = equation.y, y.start = eq2.y,
  y.bound = bounds.y, y.num.steps = step.number.y)
```

#### Step 4: Global quasi-potential

If one were to naively try to match the local quasi-potentials at  $\mathbf{e}_{u2}$ , then they would not match at  $\mathbf{e}_{u3}$ , and vice versa. To overcome this problem, it is necessary to think more carefully about how trajectories transition between basins of attraction. This issue can be dealt with rigorously (Graham

and Tél, 1986; Roy and Nauman, 1995), but the general principles are outlined here. Let  $\Omega_1$  be the basin of attraction corresponding to  $e_{s1}$  and  $\Omega_2$  be the basin of attraction corresponding to  $e_{s2}$ . Let  $\partial\Omega$  be the separatrix between these two basins (i.e., the  $x$ -axis). The most probable way for a trajectory to transition from  $\Omega_1$  to  $\Omega_2$  involves passing through the lowest point on the surface specified by  $\Phi_1$  along  $\partial\Omega$ . Examination of  $\Phi_1$  indicates that this point is  $e_{u2}$ . In the small-noise limit, the transition rate from  $\Omega_1$  to  $\Omega_2$  will correspond to  $\Phi_1(e_{u2})$ . Similarly, the transition rate from  $\Omega_2$  to  $\Omega_1$  will correspond to  $\Phi_2(e_{u3})$ . The transition rate into  $\Omega_1$  must equal the transition rate out of  $\Omega_2$ . Therefore, the two local quasi-potentials should be translated so that the minimum heights along the separatrix are the same. In other words, one must define translated local quasi-potentials  $\Phi_1^*(x, y) = \Phi_1(x, y) + c_1$  and  $\Phi_2^*(x, y) = \Phi_2(x, y) + c_2$  so that

$$\min(\Phi_1^*(x, y)|(x, y) \in \partial\Omega) = \min(\Phi_2^*(x, y)|(x, y) \in \partial\Omega).$$

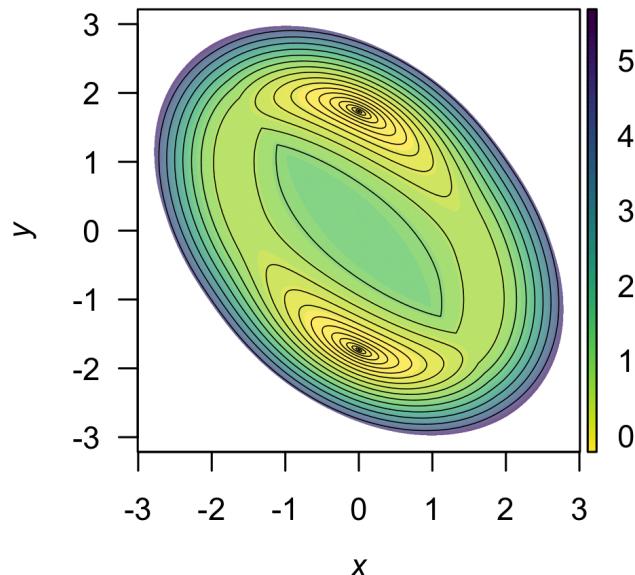
In Example 1, the minima of both local quasi-potentials occurred at the same point, so the algorithm amounted to matching at that point. In Example 3, the minimum saddle for  $\Phi_1$  is  $e_{u2}$  and the minimum saddle for  $\Phi_2$  is  $e_{u3}$ ; the heights of the surfaces at these respective points should be matched. Thus,  $c_1 = \Phi_2(e_{u3}) - \Phi_1(e_{u3})$  and  $c_2 = \Phi_1(e_{u2}) - \Phi_2(e_{u2})$ . Conveniently in Example 3, this is satisfied without requiring any translation (one can use  $c_1 = c_2 = 0$ ). Finally, the global quasi-potential is found by taking the minimum value of the matched local quasi-potentials at each point. This process is automated in **QPot**, but users can also manipulate the local quasi-potential matrices manually to verify the results. This is recommended when dealing with unusual or complicated separatrices. The code below applies the automated global quasi-potential calculation to Example 3.

```
ex3.global <- QPGlobal(local.surfaces = list(eq1.local, eq2.local),
  unstable.eq.x = c(0, -1.5, 1.5), unstable.eq.y = c(0, 0, 0), x.bound = bounds.x,
  y.bound = bounds.y)
```

### Step 5: Global quasi-potential visualization

Figure 14 shows a contour plot of the global quasi-potential. Note that the surface is continuous, but not smooth. The lack of smoothness is a generic feature of global quasi-potentials created from pasting local quasi-potentials. Cusps usually form when switching from the part of solution obtained from one local quasi-potential to the other.

```
QPContour(ex3.global, dens = c(1000, 1000), x.bound = bounds.x, y.bound = bounds.y,
  c.parm = 5)
```



**Figure 14:** A contour plot of the quasi-potential of System (5) using the function `QPContour()`. Yellow corresponds to low values of the quasi-potential, and purple to high values.

## Boundary behavior

It is important to consider the type of behavior that should be enforced at the boundaries and on coordinate axes ( $x = 0$  and  $y = 0$ ). By default, the ordered-upwind method computes the quasi-potential for the system defined by the user, without regard for the influence of the boundaries or the significance of these axes. In some cases, however, a model is only valid in a subregion of phase space. For example, in many population models, only the non-negative phase space is physically meaningful. In such cases, it is undesirable to allow the ordered-upwind method to consider trajectories that pass through negative phase space. In the default mode for `QPotential()`, if  $(x, y)$  lies in positive phase space,  $\Phi(x, y)$  can be impacted by the vector field in negative phase space, if the path corresponding to the minimum work passes through negative phase space. The argument `bounce = "d"` corresponds to this (d)eault behavior. A user can prevent the ordered upwind method from passing trajectories through negative phase space by using the option `bounce = "p"` for (p)ositive values only. This option can be interpreted as a reflecting boundary condition. It forces the front of solutions obtained by the ordered upwind method to stay in the defined boundaries, which is positive phase space in this case. A more generic option is `bounce = "b"` for (b)ounce, which allows users to supply reflecting boundaries other than the coordinate axes. These are set with `x.bound` and `y.bound`. Small numerical errors at a reflecting boundary can cause the algorithm to terminate prematurely. To avoid this, the option `bounce.edge` adds a small amount of padding between the reflecting boundary and the edge of the computational domain.

## Different noise terms

In the cases considered so far, the noise terms for the  $X$  and  $Y$  variables have had identical intensity. This was useful for purposes of illustration in the algorithm, but it will often be not true for real-world systems. Fortunately, **QPot** can accommodate other noise terms with coordinate transforms. Consider a system of the form:

$$\begin{aligned} dX(t) &= f_1(X(t), Y(t)) dt + \sigma g_1 dW_1(t) \\ dY(t) &= f_2(X(t), Y(t)) dt + \sigma g_2 dW_2(t). \end{aligned} \quad (6)$$

$\sigma$  is a scaling parameter that specifies the overall noise intensity. The parameters  $g_1$  and  $g_2$  specify the relative intensity of the two noise terms. To transform this system into a form that is usable for **QPot**, make the change of variable  $\tilde{X} = g_1^{-1}X$  and  $\tilde{Y} = g_2^{-1}Y$ . In the new coordinates, the drift terms (that is, the terms multiplied by  $dt$ ), are different. These are  $\tilde{f}_1(\tilde{X}, \tilde{Y}) = g_1^{-1}f_1(g_1\tilde{X}, g_2\tilde{Y})$  and  $\tilde{f}_2(\tilde{X}, \tilde{Y}) = g_2^{-1}f_2(g_1\tilde{X}, g_2\tilde{Y})$ . These new drift terms should be used as the deterministic skeleton that is input into **QPot**. After obtaining the global quasi-potential for these transformed coordinates, one can switch back to the original coordinates for plotting.

Many models contain multiplicative noise terms. These are of the form:

$$\begin{aligned} dX(t) &= f_1(X(t), Y(t)) dt + \sigma g_1 X(t) dW_1(t) \\ dY(t) &= f_2(X(t), Y(t)) dt + \sigma g_2 Y(t) dW_2(t). \end{aligned} \quad (7)$$

To transform this system into a form that is usable for **QPot**, make the change of variable  $\tilde{X} = g_1^{-1}\ln(X)$  and  $\tilde{Y} = g_2^{-1}\ln(Y)$ . This is called the Lamperti transform (Iacus, 2009). It is not always possible to transform a multidimensional stochastic differential equation with multiplicative noise into one with additive noise (Pavliotis, 2014), but in special cases like (7) it is. This coordinate change is non-linear, so Itô's lemma introduces extra terms into the drift of the transformed equations. If  $\sigma$  is small, though, these terms can be discounted, and the new drift terms will remain independent of  $\sigma$ . These new drift terms can be input into **QPot**. After obtaining the global quasi-potential for these transformed coordinates, one can switch back to the original coordinates.

## Conclusion

**QPot** is an R package that provides several important tools for analyzing two-dimensional systems of stochastic differential equations. Future efforts will work toward extending **QPot** to higher-dimensional systems, but this is a computationally challenging task. **QPot** includes functions for generating realizations of the stochastic differential equations, and for analyzing and visualizing the results. A central component of **QPot** is the calculation of quasi-potential functions, which are highly useful for studying stochastic dynamics. For example, quasi-potential functions can be used to compare the stability of different attractors in stochastic systems, a task that traditional linear stability

analysis is poorly suited for (Nolting and Abbott, 2016). By offering an intuitive way to quantify attractor stability, quasi-potentials are poised to become an important means of understanding phenomena like metastability and alternative stable states. **QPot** makes quasi-potentials accessible to R users interested in applying this new framework.

## Author contributions

K.C.A., C.M.M., B.C.N., and C.R.S. designed the project. M.K.C. wrote the C code for finding the quasi-potential; C.M.M. and C.R.S. wrote the R code and adapted the C code.

## Acknowledgments

This work was supported by a Complex Systems Scholar grant to K.C.A. from the James S. McDonnell Foundation. M.K.C. was partially supported by NSF grant 1217118. We also thank the anonymous reviewers for their constructive comments and suggestions which helped us to improve the quality of our paper.

## Bibliography

- D. Adler, D. Murdoch, O. Nenadic, S. Urbanek, M. Chen, A. Gebhardt, B. Bolker, G. Csardi, A. Strzelecki, and A. Senger. *rgl: 3D Visualization Device System for R using OpenGL*, 2015. URL <https://CRAN.R-project.org/package=rgl>. R package version 0.95.1247. [p27]
- E. J. Allen. *Modeling with Ito Stochastic Differential Equations*, volume 22 of *Mathematical Modelling: Theory and Applications*. Springer-Verlag, 2007. [p20]
- A. Brouste, M. Fukasawa, H. Hino, S. M. Iacus, K. Kamatani, Y. Koike, H. Masuda, R. Nomura, T. Ogihara, Y. Shimuzu, M. Uchida, and N. Yoshida. The YUIMA project: A computational framework for simulation and inference of stochastic differential equations. *Journal of Statistical Software*, 57(4):1–51, 2014. doi: 10.18637/jss.v057.i04. [p22]
- M. K. Cameron. Finding the quasipotential for nongradient SDEs. *Physica D*, 241(18):1532–1550, 2012. [p21, 24, 25]
- T. S. P. L. Chua and T. S. Parker. *Practical Numerical Algorithms for Chaotic Systems*, chapter 5. Springer-Verlag, 1989. [p30]
- J. S. Collie and P. D. Spencer. Modeling predator-prey dynamics in a fluctuating environment. *Canadian Journal of Fisheries and Aquatic Sciences*, 51(12):2665–2672, 1994. [p21]
- M. I. Freidlin and A. D. Wentzell. *Random Perturbations of Dynamical Systems*, volume 260. Springer-Verlag, 2012. [p21, 26]
- S. Garnier. *viridis: Default Color Maps from ‘matplotlib’*, 2016. URL <https://github.com/sjmgarnier/viridis>. R package version 0.3.4. [p27]
- R. Graham and T. Tél. Nonequilibrium potential for coexisting attractors. *Physical Review A*, 33(2):1322–1337, 1986. [p26, 32, 34]
- M. J. Grayling. *phaseR: Phase Plane Analysis of One and Two Dimensional Autonomous ODE Systems*, 2014a. URL <https://CRAN.R-project.org/package=phaseR>. R package version 1.3. [p22]
- M. J. Grayling. phaseR: An R package for phase plane analysis of autonomous ODE systems. *The R Journal*, 6(2):43–51, 2014b. [p22]
- A. Guidoum and K. Boukhetala. *Sim.DiffProc: Simulation of Diffusion Processes*, 2016. URL <https://CRAN.R-project.org/package=Sim.DiffProc>. R package version 3.2. [p22]
- S. M. Iacus. *Simulation and Inference for Stochastic Differential Equations: With R Examples*, volume 1. Springer Science & Business Media, 2009. [p20, 36]
- C. Moore, C. Stieha, B. Nolting, M. Cameron, and K. Abbott. *QPot: Quasi-Potential Analysis for Stochastic Differential Equations*, 2016. URL <https://CRAN.R-project.org/package=QPot>, <https://github.com/bmarksplash7/QPot>. R package version 1.2. [p19]

- B. C. Nolting and K. C. Abbott. Balls, cups, and quasi-potentials: Quantifying stability in stochastic systems. *Ecology*, 97(4):850–864, 2016. [p<sup>20, 21, 27, 28, 37</sup>]
- G. A. Pavliotis. *Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations*, volume 60 of *Texts in Applied Mathematics*. Springer-Verlag, New York, 2014. [p<sup>36</sup>]
- R. V. Roy and E. Nauman. Noise-induced effects on a non-linear oscillator. *Journal of Sound and Vibration*, 183(2):269–295, 1995. doi: 10.1006/jsvi.1995.0254. [p<sup>26, 32, 35</sup>]
- J. A. Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton-Jacobi equations. *Proceedings of the National Academy of Sciences*, 98(20):11069–11074, 2001. doi: 10.1073/pnas.201222998. [p<sup>24</sup>]
- J. A. Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton-Jacobi equations: Theory and algorithms. *SIAM Journal on Numerical Analysis*, 41(1):325–363, 2003. doi: 10.1137/S0036142901392742. [p<sup>24</sup>]
- K. Soetaert. *plot3D: Plotting Multi-Dimensional Data in R*, 2014. URL <https://CRAN.R-project.org/package=plot3D>. R package version 1.0-2. [p<sup>27</sup>]
- K. Soetaert and P. M. Herman. *A Practical Guide to Ecological Modelling: Using R as a Simulation Platform*. Springer Science & Business Media, 2008. [p<sup>21</sup>]
- K. Soetaert, T. Petzoldt, and R. W. Setzer. Solving differential equations in R: Package deSolve. *Journal of Statistical Software*, 33(9):1–25, 2010. doi: 10.18637/jss.v033.i09. [p<sup>22</sup>]
- J. H. Steele and E. W. Henderson. A simple plankton model. *American Naturalist*, 117(5):676–691, 1981. [p<sup>21</sup>]

Christopher M. Moore  
Department of Biology  
Case Western Reserve University  
United States  
[life.dispersing@gmail.com](mailto:life.dispersing@gmail.com)

Christopher R. Stieha  
Department of Biology  
Case Western Reserve University  
United States  
[stieha@hotmail.com](mailto:stieha@hotmail.com)

Ben C. Nolting  
Department of Biology  
Case Western Reserve University  
United States

Current:  
Department of Mathematics and Statistics  
California State University, Chico  
United States  
[bnolting@gmail.com](mailto:bnolting@gmail.com)

Maria K. Cameron  
Department of Mathematics  
University of Maryland  
United States  
[cameron@math.umd.edu](mailto:cameron@math.umd.edu)

Karen C. Abbott  
Department of Biology  
Case Western Reserve University  
United States  
[kcabbott@case.edu](mailto:kcababbott@case.edu)

# Design of the TRONCO BioConductor Package for TRanslational ONCology

by Marco Antoniotti, Giulio Caravagna, Luca De Sano, Alex Graudenzi, Giancarlo Mauri, Bud Mishra, and Daniele Ramazzotti

**Abstract** Models of *cancer progression* provide insights on the order of accumulation of genetic alterations during cancer development. Algorithms to infer such models from the currently available mutational profiles collected from different cancer patients (*cross-sectional data*) have been defined in the literature since late the 90s. These algorithms differ in the way they extract a *graphical model* of the events modelling the progression, e.g., somatic mutations or copy-number alterations.

**TRONCO** is an R package for TRanslational ONcology which provides a series of functions to assist the user in the analysis of cross-sectional genomic data and, in particular, it implements algorithms that aim to model cancer progression by means of the notion of selective advantage. These algorithms are proved to outperform the current state-of-the-art in the inference of cancer progression models. **TRONCO** also provides functionalities to load input cross-sectional data, set up the execution of the algorithms, assess the statistical confidence in the results, and visualize the models.

**Availability.** Freely available at <http://www.bioconductor.org/> under GPL license; project hosted at <http://bimib.disco.unimib.it/> and <https://github.com/BIMIB-DISCo/TRONCO>.

**Contact.** [tronco@disco.unimib.it](mailto:tronco@disco.unimib.it)

## Introduction

In the last two decades many specific genes and genetic mechanisms involved in different types of cancer have been identified. Yet our understanding of cancer and of its varied progressions is still largely elusive, as it still faces fundamental challenges.

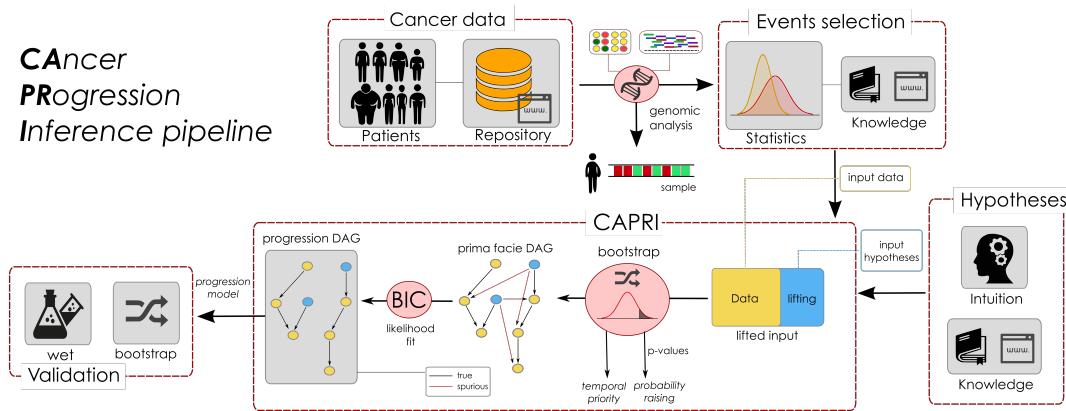
Meanwhile, a growing number of cancer-related genomic data sets have become available (e.g., see [NCI and the NHGRI \(2005\)](#)). There now exists an urgent need to leverage a number of sophisticated computational methods in biomedical research to analyse such fast-growing biological data sets. Motivated by this state of affairs, we focus on the problem of *reconstructing progression models* of cancer. In particular, we aim at inferring the plausible sequences of *genomic alterations* that, by a process of *accumulation*, selectively make a tumor fitter to survive, expand and diffuse (i.e., metastasize).

We developed a number of algorithms (see [Olde Loohuis et al. \(2014\)](#); [Ramazzotti et al. \(2015\)](#)) which are implemented in the *TRanslational ONCology (TRONCO)* package. Starting from cross-sectional genomic data, such algorithms aim at reconstructing a probabilistic progression model by inferring “selectivity relations,” where a mutation in a gene *A* “selects” for a later mutation in a gene *B*. These relations are depicted in a combinatorial graph and resemble the way a mutation exploits its “*selective advantage*” to allow its host cells to expand clonally. Among other things, a selectivity relation implies a putatively invariant temporal structure among the genomic alterations (i.e., *events*) in a specific cancer type. In addition, a selectivity relation between a pair of events here signifies that the presence of the earlier genomic alteration (i.e., the *upstream event*) is advantageous in a Darwinian competition scenario raising the probability with which a subsequent advantageous genomic alteration (i.e., the *downstream event*) “survives” in the clonal evolution of the tumor (see [Ramazzotti et al. \(2015\)](#)).

Notice that, in general, the inference of cancer progression models requires a complex data processing pipeline (see [Caravagna et al. \(2015\)](#)), as summarized in Figure 1. Initially, one collects *experimental data* (which could be accessible through publicly available repositories such as TCGA) and performs *genomic analyses* to derive profiles of, e.g., somatic mutations or copy-number variations for each patient. Then, statistical analysis and biological priors are used to select events relevant to the progression (e.g., *driver mutations*). This complex pipeline can also include further statistics and priors to determine cancer subtypes and to generate *patterns of selective advantage* (e.g., hypotheses of mutual exclusivity). Given these inputs, our algorithms (such as CAPRESE and CAPRI) can extract a progression model and assess *confidence measures* using various metrics based on non-parametric bootstrap and hypergeometric testing. *Experimental validation* concludes the pipeline. The **TRONCO** package provides support to all the steps of the pipeline.

## Inference algorithms

**TRONCO**, provides a series of functions to support the user in each step of the pipeline, i.e., from data import, through data visualization, and, finally, to the inference of cancer progression models.



**Figure 1:** Data processing pipeline for the cancer progression inference. **TRONCO** implements a pipeline consisting in a series of functions and algorithms to extract cancer progression models from cross-sectional input data. The first step of such a pipeline consists in collecting *experimental data* (which could be accessible through publicly available repositories such as TCGA) and performing *genomic analyses* to derive profiles of, e.g., somatic mutations or copy-number variations for each patient or single cells. Then, both statistical analysis and biological priors are adopted to select the significant alterations for the progression; e.g., *driver mutations*. This complex pipeline can also include further statistics and priors to determine cancer subtypes and to generate *patterns of selective advantage*; e.g., hypotheses of mutual exclusivity. Given these inputs, the implemented algorithms (i.e., CAPRESE and CAPRI) can extract a progression model and assess various *confidence* measures on its constituting relations such as non-parametric bootstrap and hypergeometric testing. *Experimental validation* concludes the pipeline, see [Ramazzotti et al. \(2015\)](#) and [Caravagna et al. \(2015\)](#).

Specifically, in the current version, **TRONCO** implements the CAPRESE and CAPRI algorithms for cancer progression inference, which we briefly describe in the following.

Central to these algorithms, is Suppes' notion of *probabilistic causation*, which can be stated in the following terms: a selectivity relation between two observables  $i$  and  $j$  is said to hold if (1)  $i$  occurs earlier than  $j$  – *temporal priority* (TP) – and (2) if the probability of observing  $i$  raises the probability of observing  $j$ , i.e.,  $P(j | i) > P(j | \bar{i})$  – *probability raising* (PR). For the detailed description of the methods, we refer the reader to [Olde Loohuis et al. \(2014\)](#); [Ramazzotti et al. \(2015\)](#).

### CAncer PRogression Extraction with Single Edge

The *CAncer PRogression Extraction with Single Edges* algorithm, i.e., CAPRESE, extracts tree-based models of cancer progression with (i) multiple independent starting points and (ii) branches. The former models the emergence of different progressions as a result of the natural heterogeneity of cancer (cf., [Olde Loohuis et al. \(2014\)](#)). The latter models the possibility of a clone to undergo positive selection by acquiring different mutations.

The inference of CAPRESE's models is driven by a *shrinkage* estimator of the confidence in the relation between pairs of genes, which augments robustness to noise in the input data.

As shown in [Olde Loohuis et al. \(2014\)](#), CAPRESE is currently the state-of-the-art algorithm to infer tree cancer progression models, although its expressivity is limited to this kind of selective advantage models (cf., [Ramazzotti et al. \(2015\)](#)). Since this limitation is rather unappealing in analyzing cancer data, an improved algorithm was sought in [Ramazzotti et al. \(2015\)](#).

### CAncer PRogression Inference

The *CAncer PRogression Inference* algorithm, i.e., CAPRI, extends tree models by allowing multiple predecessors of any common downstream event, thus allowing construction of directed acyclic graph (DAGs) progression models.

CAPRI performs maximum likelihood estimation for the progression model with constraints grounded in Suppes' *prima facie* causality (cf., [Ramazzotti et al. \(2015\)](#)). In particular, the search space of the possible valid solutions is limited to the selective advantage relations where both TP and PR are verified. On this reduced search space, the likelihood fit is performed. CAPRI was shown to be effective and polynomial in the size of the inputs.

### Algorithms' structures

The core of the two algorithms is a simple quadratic loop<sup>1</sup> that *prunes* arcs from an initially totally connected graph. Each pruning decision is based on the application of Suppes' *probabilistic causation* criteria.

The pseudocode of the two implemented algorithms along with the procedure to evaluate the confidence of the arcs by bootstrap is summarized in Algorithms 1, 2, 3 and 4, which depict the data preparation step, the CAPRESE and CAPRI algorithms, and finally the optional *bootstrap* step.

#### Algorithm 1: TRONCO Data Import and Preprocessing

**Input:** a data set containing MAF or GISTIC scores (e.g., as obtained from cBio portal) ([Cerami et al. \(2012\)](#); [Beroukhim et al. \(2007\)](#)).

**Result:** a data structure containing Boolean flags for “events,” relative frequencies and other metadata.

```

1 From the data set (depending on the data format) derive a Boolean matrix  $M$ , where
   each entry  $\langle i, j \rangle$  is true if event  $i$  is “present” in sample/patient  $j$ .
2 forall events  $e$  do
3   | Compute the frequency of the event  $e$  in the data set and save it in a map  $F$ .
4   | Compute the joint probability of co-occurrence of pair of events in the data set and
      | save it in a map  $C$ .
5 end
6 return A data structure comprising the Boolean matrix  $M$ , the maps  $F$  and  $C$  and other
   | metadata.

```

#### Algorithm 2: CAPRESE algorithm

**Input:** a data set of  $n$  events, i.e., genomic alterations, and  $m$  samples packed in a data structure obtained from Algorithm 1.

**Result:** a *tree model* representing all the relations of selective advantage.

*Pruning based on Suppes' criteria.*

```

1 Let  $G \leftarrow$  a complete directed graph over the vertices  $n$ .
2 forall arcs  $(a, b)$  in  $G$  do
3   | Compute a score  $S(\cdot)$  for the nodes  $a$  and  $b$  based on Suppes' criteria.
   | Verify Suppes' criteria, that is:
4   |   if  $S(a) \geq S(b)$  and  $S(a) > 0$  then
5   |     | Keep  $(a, b)$  as edge.                                I.e., select 'a' as "candidate parent".
6   |   else if  $S(b) > S(a)$  and  $S(b) > 0$  then
7   |     | Keep  $(b, a)$  as edge.                                I.e., select 'b' as "candidate parent".
8 end

```

*Fit of the prima facie directed acyclic graph to the best tree model.*

```

9 Let  $\mathcal{T} \leftarrow$  the best tree model obtained by Edmonds' algorithm (see Edmonds \(1967\)).
10 Remove from  $\mathcal{T}$  any connection where the candidate father does not have a minimum
    level of correlation with the child.
11 return The resulting tree model  $\mathcal{T}$ .

```

### Package design

In this section we will review the structure and implementation of the TRONCO package. For the sake of clarity, we will structure the description through the following functionalities that are implemented in the package.

<sup>1</sup>For CAPRI the  $n$  actually depends on the structural complexity of the input “patterns,” i.e., of the Boolean formulae employed in the “lifting operation;” more information of this in [Ramazzotti et al. \(2015\)](#).

**Algorithm 3:** CAPRI

**Input:** a data set of  $n$  variables, i.e., genomic alterations or patterns, and  $m$  samples.  
**Result:** a graphical model representing all the relations of “selective advantage.”

*Pruning based on the Suppes’ criteria*

- 1 Let  $G \leftarrow$  a directed graph over the vertices  $n$
- 2 **forall** arcs  $(a, b) \in G$  **do**
- 3   | Compute a score  $S(\cdot)$  for the nodes  $a$  and  $b$  in terms of Suppes’ criteria.
- 4   | Remove the arc  $(a, b)$  if Suppes’ criteria are not met.
- 5 **end**

*Likelihood fit on the prima facie directed acyclic graph*

- 6 Let  $\mathcal{M} \leftarrow$  the subset of the remaining arcs  $\in G$ , that maximize the log-likelihood of the model, computed as:  $LL(D | \mathcal{M}) - ((\log m)/2) \dim(\mathcal{M})$ , where  $D$  denotes the input data,  $m$  denotes the number of samples, and  $\dim(\mathcal{M})$  denotes the number of parameters in  $\mathcal{M}$  (see [Koller and Friedman \(2009\)](#)).
- 7 **return** The resulting graphical model  $\mathcal{M}$ .

**Algorithm 4:** Bootstrap Procedure

**Input:** a model  $\mathcal{T}$  obtained from CAPRESE or a model  $\mathcal{M}$  obtained from CAPRI, and the initial data set.  
**Result:** the confidence in the inferred arcs.

- 1 Let  $counter \leftarrow 0$
- 2 Let  $nboot \leftarrow$  the number of bootstrap sampling to be performed.
- 3 **while**  $counter < nboot$  **do**
- 4   | Create a new data set for the inference by random sampling of the input data.
- 5   | Perform the reconstruction on the sampled data set and save the results.
- 6   |  $counter = counter + 1$
- 7 **end**
- 8 Evaluate the confidence in the reconstruction by counting the number of times any arc is inferred in the sampled data sets.
- 9 **return** The inferred model  $\mathcal{T}$  or  $\mathcal{M}$  augmented with an estimated confidence for each arc.

- **Data import.** Functions for the importation of data both from flat files (e.g., MAF, GISTIC) and from Web querying (e.g., cBioPortal [Cerami et al. \(2012\)](#)).
- **Data export and correctness.** Functions for the export and visualization of the imported data.
- **Data editing.** Functions for the preprocessing of the data in order to tidy them.
- **External utilities.** Functions for the interaction with external tools for the analysis of cancer subtypes or groups of mutually exclusive genes.
- **Inference algorithms.** In the current version of **TRONCO**, the CAPRESE and CAPRI algorithms are provided in a polynomial implementation.
- **Confidence estimation.** Functions for the statistical estimation of the confidence of the reconstructed models.
- **Visualization.** Functions for the visualization of both the input data and the results of the inference and of the confidence estimation.

**Data import**

The starting point of **TRONCO** analysis pipeline is a data set of genomics alterations (i.e., somatic mutations and copy number variations) which need to be imported as a **TRONCO**-compliant data structure, i.e., a R list structure containing the required data both for the inference and the visualization. The data import functions take as input such genomic data and from them create a **TRONCO**-compliant data structure consisting in a list variable with the different parameters needed by the algorithms.

The core of data import from text files, is the function

```
import.genotypes(geno, event.type = "variant", color = "Darkgreen")
```

This function imports a matrix of 0/1 alterations as a **TRONCO** compliant data set. The input geno can be either a data frame or a file name. In any case the data frame or the table stored in the file must have a column for each altered gene and a row for each sample. Column names will be used to determine gene names; if data are loaded from a file, the first column will be assigned as row names.

**TRONCO** imports data from other file format such as MAF and GISTIC, by providing wrappers of the function `import.genotypes`. Specifically, the function

```
import.MAF(file, sep = "\t", is.TCGA = TRUE)
```

imports mutation profiles from a *Manual Annotation Format* (MAF) file. All mutations are aggregated as a unique event type labeled "Mutation" and are assigned a color according to the default of function `import.genotypes`. If the input is in the TCGA MAF file format, the function also checks for multiple samples per patient and a warning is raised if any are found. The function

```
import.GISTIC(x)
```

also transforms GISTIC scores for copy number alterations (CNAs) in a **TRONCO**-compliant object. The input can be a matrix with columns for each altered gene and rows for each sample; (in this case colnames/rownames must be provided). If the input is a string, an attempt to load a table from the indicated file is performed. In this case the input table format should be consistent with TCGA data for focal CNA; i.e., there should hence be: one column for each sample, one row for each gene, a column Hugo\_Symbol with every gene name and a column Entrez\_Gene\_Id with every gene's Entrez ID. A valid GISTIC score should be any value of: "Homozygous Loss" (-2), "Heterozygous Loss" (-1), "Low-level Gain" (+1), and "High-level Gain" (+2).

Finally, **TRONCO** also provides utilities for the query of genomic data from cBioPortal Cerami et al. (2012). This functionality is provided by the function

```
cbio.query(cbio.study = NA, cbio.dataset = NA, cbio.profile = NA, genes)
```

which is a wrapper for the CGDS package Jacobsen (2011). This can work either automatically, if one sets `cbio.study`, `cbio.dataset` and `cbio.profile`, or interactively. A list of genes to query with less than 900 entries should be provided. This function returns a list with two data frames: the required genetic profile along with clinical data for the `cbio.study`. The output is also saved to disk as an Rdata file. See also the cBioPortal webpage at <http://www.cbioportal.org>.

The function

```
show(x, view = 10)
```

prints (on the R console) a short report of a data set x, which should be a **TRONCO**-compliant data set.

All the functions described in the following sections will assume as input a **TRONCO**-compliant data structure.

## Data export and correctness

**TRONCO** provides a series of function to explore the imported data and the inferred models. All these functions are named with the 'as.' prefix.

Given a **TRONCO**-compliant imported data set, the function

```
as.genotypes(x)
```

returns the 0/1 genotypes matrix. This function can be used in combination with the function

```
keysToNames(x, matrix)
```

to translate column names to event names, given the input matrix with colnames/rownames which represent genotypes keys. Also, functions to get the list of genes, events (i.e., each column in the genotypes matrix, it differs from genes, as the same genes of different types are considered different events), alterations (i.e., genes of different types are merged as 1 unique event), samples (i.e., patients or also single cells), and alteration types. See the functions

```
as.genes(x, types = NA)
as.events(x, genes = NA, types = NA)
as.alterations(x, new.type = "Alteration", new.color = "khaki")
as.samples(x)
as.types(x, genes = NA)
```

Functions of this kind are also implemented to explore the results, most notably the models that have been inferred

```
as.models(x, models = names(x$model)))
```

the reconstructions

```
as.adj.matrix(x, events = as.events(x), models = names(x$model), type = "fit")
```

the patterns (i.e., the *formulae*)

```
as.patterns(x)
```

and the confidence

```
as.confidence(x, conf)
```

Similarly, the library defines a set of functions that extract the cardinality of the compliant **TRONCO** data structure

```
nevents(x, genes = NA, types = NA)
ngenes(x, types = NA)
npatterns(x)
nsamples(x)
ntypes(x)
```

Furthermore, functions to asses the correctness of the inputs are also provided. The function

```
is.compliant(x,
            err.fun = "[ERR]",
            stage = !(all(is.null(x$stages)) || all(is.na(x$stages))))
```

verifies that the parameter *x* is a compliant data structure. The function

```
consolidate.data(x, print = FALSE)
```

verifies if the input data are consolidated, i.e., if there are events with 0 or 1 probability or indistinguishable in terms of observations. Any indistinguishable event is returned by the function *duplicates(x)*.

Finally, **TRONCO** provides functions to access TCGA data.

```
TCGA.multiple.samples(x)
```

checks whether there are multiple sample in the input, while

```
TCGA.remove.multiple.samples(x)
```

removes them accordingly to TCGA barcodes naming rules.

## Data editing

**TRONCO** provides a wide range of editing functions. We will describe some of them in the following; for a technical description we refer to the manual.

### Removing and merging

A set of functions to remove items from the data is provided; such functions are characterized by the 'delete.' prefix. The main functions are

```
delete.gene(x, gene)
delete.samples(x, samples)
delete.type(x, type)
delete.pattern(x, type)
```

These respectively remove genes, samples (i.e., tumors profiles), types (i.e., type of alteration such as somatic mutation, copy number alteratio, etc.), and patterns from a **TRONCO** data structure *x*. Conversely it is possible to *merge* events and types:

```
merge.events(x, ..., new.event, new.type, event.color)
merge.types(x, ..., new.type = "new.type", new.color = "khaki")
```

## Binding

The purpose of the binding functions is to combine different data sets. The function

```
ebind(...)
```

combines events from one or more data sets, whose events need be defined over the same set of samples. The function

```
sbind(...)
```

combines samples from one or more data sets, whose samples need to be defined over the same set of events. Samples and events of two data set can also be intersected via the function

```
intersect.datasets(x, y, intersect.genomes = TRUE)
```

## Changing and renaming

The functions

```
rename.gene(x, old.name, new.name)
rename.type(x, old.name, new.name)
```

can be used respectively to rename genes or alterations types.

The function

```
change.color(x, type, new.color)
```

can be used to change the color associated to the specified alteration type in x.

## Selecting and splitting

Genomics data usually involve a large number of genes, most of which are not relevant for cancer development (e.g., they may be passenger mutations). For this reason, **TRONCO** implements the function

```
events.selection(x, filter.freq = NA, filter.in.names = NA, filter.out.names = NA)
```

which allows the user to select a subset of genes to be analyzed. The selection can be performed by frequency and gene symbols. The 0 probability events can are removed by the function `trim(x)`. Moreover, the functions

```
samples.selection(x, samples)
ssplit(x, clusters, idx = NA)
```

respectively filter a data set x based on the selected sample's id and then splits the data set into clusters (i.e., groups). The last function can be used to analyze specific subtypes within a tumor.

## External utilities

**TRONCO** permits the interaction with external tools to (*i*) reduce inter-tumor heterogeneity by cohort subtyping and (*ii*) detect fitness equivalent exclusive alterations. The first issue can be attacked by adopting clustering techniques to split the data set in order to analyze each cluster subtype separately. Currently, **TRONCO** can export and import data from [Hofree et al. \(2013\)](#) via the function

```
export.nbs.input(x, map_hugo_entrez, file = "tronco_to_nbs.mat")
```

and the previously described splitting functions.

In order to handle alterations with equivalent fitness, **TRONCO** interacts with the tool MUXEX proposed in [Babur et al. \(2014\)](#). The interaction is ensured by the functions

```
export.mutex(
  filename = "to_mutex",
  filepath = "./",
  label.mutation = "SNV",
  label.amplification = list("High-level Gain"),
  label.deletion = list("Homozygous Loss"))
import.mutex.groups(file, fdr = 0.2, display = TRUE)
```

Such exclusivity groups can then be further added as patterns (see the next section).

## Inference algorithms

The current version of TRONCO implements the *progression reconstruction algorithms* CAPRESE Olde Loohuis et al. (2014) and CAPRI Ramazzotti et al. (2015).

**CAPRESE.** The CAPRESE algorithm Olde Loohuis et al. (2014) can be executed by the function

```
tronco.caprese(data, lambda = 0.5, do.estimation = FALSE, silent = FALSE)
```

with data being a **TRONCO** data structure. The parameter lambda can be used to tune the shrinkage-like estimator adopted by CAPRESE, with the default being 0.5 as suggested in Olde Loohuis et al. (2014).

**CAPRI.** The CAPRI algorithm Ramazzotti et al. (2015) is executed by the function

```
tronco.capri(data,
              command = "hc",
              regularization = c("bic", "aic"),
              do.boot = TRUE,
              nboot = 100,
              pvalue = 0.05,
              min.boot = 3,
              min.stat = TRUE,
              boot.seed = NULL,
              do.estimation = FALSE,
              silent = FALSE)
```

with data being a **TRONCO** data structure. The parameters command and regularization allow respectively to choose the heuristic search to be performed to fit the network and the regularizer to be used in the likelihood fit (see Ramazzotti et al. (2015)). CAPRI can be also executed with or without the bootstrap preprocessing step depending on the value of the parameter do.boot; this is discouraged, but can speed up the execution with large input data sets.

As discussed in Ramazzotti et al. (2015), CAPRI constrains the search space using Suppes' *prima facie* conditions which lead to a subset of possible valid selective advantage relations. The members of this subset are then evaluated by the likelihood fit. Although uncommon, it may so happen (especially when patterns are given as input) that such a resulting *prima facie* graphical structure may still contain cycles. When this happens, the cycles are removed through the heuristic algorithm implemented in

```
remove.cycles(adj.matrix,
               weights.temporal.priority,
               weights.matrix,
               not.ordered,
               hypotheses = NA,
               silent)
```

The function takes as input a set of weights in terms of confidence for any valid selective advantage edge; ranks all the valid edges in increasing confidence levels; and, starting from the less confident, goes through each edge removing the ones that can break the cycles.

## Patterns

CAPRI allows for the input of patterns, i.e., group of events which express possible selective advantage relations. Such patterns are given as input using the function

```
hypothesis.add(data,
                pattern.label,
                lifted.pattern,
                pattern.effect = "*",
                pattern.cause = "*")
```

This function is wrapped within the functions

```
hypothesis.add.homologous(x,
                           pattern.cause = "*",
                           pattern.effect = "*",
                           genes = as.genes(x),
```

```

        FUN = OR)
hypothesis.add.group(x,
                      FUN,
                      group,
                      pattern.cause = "*",
                      pattern.effect = "*",
                      dim.min = 2,
                      dim.max = length(group),
                      min.prob = 0)

```

which, respectively, allow the addition of analogous patterns (i.e., patterns involving the same gene of different types) and patterns involving a specified group of genes. In the current version of **TRONCO**, the implemented patterns are Boolean, i.e., those expressible by the Boolean operators AND, OR and XOR (functions AND(...), OR(...), and XOR(...)).

### Confidence estimation

To assess the confidence of the selectivity relations found, **TRONCO** uses *non-parametric* and *statistical* bootstraps. For the non-parametric bootstrap, each event row is uniformly sampled with repetitions from the input genotype and then, on such an input, the inference algorithms are performed. The assessment concludes after  $K$  repetitions (e.g.,  $K = 100$ ). Similarly, for CAPRI, a statistical bootstrap is provided: in this case the input data set is kept fixed, but different seeds for the statistical procedures are sampled (see, e.g., Wu (1986) for an overview of these methods). The bootstrap is implemented in the function

```

tronco.bootstrap(reconstruction,
                  type = "non-parametric",
                  nboot = 100,
                  verbose = FALSE)

```

where `reconstruction` is a **TRONCO**-compliant object obtained by the inference by one of the implemented algorithms.

### Visualization and reporting

During the development of the **TRONCO** package, a lot of attention was paid to the visualization features which are crucial for the understanding of biological results. Listed below is a summary of the main features; for a detailed description of each function, please refer to the manual.

**ONCOPRINT.** ONCOPRINTs are compact means of visualizing distinct genomic alterations, including somatic mutations, copy number alterations, and mRNA expression changes across a set of cases. They are extremely useful for visualizing gene set and pathway alterations across a set of cases, and for visually identifying trends, such as trends in mutual exclusivity or co-occurrence between gene pairs within a gene set. Individual genes are represented as rows, and individual cases or patients are represented as columns. See <http://www.cbioportal.org/>. The function

```
oncoprint(x)
```

provides such visualizations with a **TRONCO**-compliant data structure as input. The function

```
oncoprint.cbio(x)
```

exports the input for the cBioPortal visualization, see <http://www.cbioportal.org/public-portal/oncoprinter.jsp>.

It is also possible to annotate a description and tumor stages to any oncoprint by means of the functions

```

annotate.description(x, label)
annotate.stages(x, stages, match.TCGA.patients = FALSE).

```

**Reconstruction.** The inferred models can be displayed by the function `tronco.plot`. The features included in the plots are multiple, such as the choice of the regularizer(s), editing font of nodes and edges, scaling nodes' size in terms of estimated marginal probabilities, annotating the pathway of each gene and displaying the estimated confidence of each edge. We refer to the manual for a detailed description.

**Reports.** Finally, **TRONCO** provides a number of reporting utilities. The function

```
genes.table.report(x,
                    name,
                    dir = getwd(),
                    maxrow = 33,
                    font = 10,
                    height = 11,
                    width = 8.5,
                    fill = "lightblue")
```

can be used to generate **LATEX** code to be used as report, while the function

```
genes.table.plot(x, name, dir = getwd())
```

generates histograms reports.

## TRONCO use cases

In this section, we will present a case study for the usage of the **TRONCO** package based on the work presented in [Ramazzotti et al. \(2015\)](#). Specifically, the example is from [Piazza et al. \(2013\)](#) where they used a high-throughput *exome sequencing technology* to identify somatically acquired mutations in 64 ACML patients, and found a previously unidentified recurring *missense point mutation* hitting the SETBP1 gene.

The example illustrates the typical steps that are necessary to perform a *progression reconstruction* with **TRONCO**. The steps are the following:

1. Selecting “Events”.
2. Adding “Hypotheses”.
3. Reconstructing the “Progression Model”.
4. Bootstrapping the Data.

**Selecting Events.** We will start by loading the **TRONCO** package in R along with an example *data set* that is part of the package distribution.

```
> library(TRONCO)
> data(aCML)
> hide.progress.bar <- TRUE
```

We then use the function `show` to get a short summary of the aCML data set that has just been loaded.

```
> show(aCML)
Description: CAPRI - Bionformatics aCML data.
Dataset: n=64, m=31, |G|=23.
Events (types): Ins/Del, Missense point, Nonsense Ins/Del, Nonsense point.
Colors (plot): darkgoldenrod1, forestgreen, cornflowerblue, coral.
Events (10 shown):
  gene 4 : Ins/Del TET2
  gene 5 : Ins/Del EZH2
  gene 6 : Ins/Del CBL
  gene 7 : Ins/Del ASXL1
  gene 29 : Missense point SETBP1
  gene 30 : Missense point NRAS
  gene 31 : Missense point KRAS
  gene 32 : Missense point TET2
  gene 33 : Missense point EZH2
  gene 34 : Missense point CBL
Genotypes (10 shown):
   gene 4 gene 5 gene 6 gene 7 gene 29 gene 30 gene 31 gene 32 gene 33 gene 34
patient 1    0     0     0     0     1     0     0     0     0     0     0
patient 2    0     0     0     0     1     0     0     0     0     0     1
patient 3    0     0     0     0     1     1     0     0     0     0     0
patient 4    0     0     0     0     1     0     0     0     0     0     1
patient 5    0     0     0     0     1     0     0     0     0     0     0
patient 6    0     0     0     0     1     0     0     0     0     0     0
```

Using the function `as.events`, we can have a look at the genes flagged as “mutated” in the data set (i.e., the *events* that **TRONCO** deals with).

```
> as.events(aCML)
      type          event
gene 4  "Ins/Del"    "TET2"
gene 5  "Ins/Del"    "EZH2"
gene 6  "Ins/Del"    "CBL"
gene 7  "Ins/Del"    "ASXL1"
gene 29 "Missense point" "SETBP1"
gene 30 "Missense point" "NRAS"
gene 31 "Missense point" "KRAS"
gene 32 "Missense point" "TET2"
gene 33 "Missense point" "EZH2"
...
gene 88 "Nonsense point" "TET2"
gene 89 "Nonsense point" "EZH2"
gene 91 "Nonsense point" "ASXL1"
gene 111 "Nonsense point" "CSF3R"
```

These events account for alterations in the following genes.

```
> as.genes(aCML)
[1] "TET2"    "EZH2"    "CBL"     "ASXL1"   "SETBP1"  "NRAS"   "KRAS"   "IDH2"    "SUZ12"
[10] "SF3B1"   "JARID2"  "EED"     "DNMT3A"  "CEBPA"   "EPHB3"  "ETNK1"  "GATA2"   "IRAK4"
[19] "MTA2"    "CSF3R"   "KIT"     "WT1"     "RUNX1"
```

Now we can take a look at the alterations of only the gene `SETBP1` across the samples.

```
> as.gene(aCML, genes = 'SETBP1')
      Missense point SETBP1
patient 1           1
patient 2           1
patient 3           1
...
patient 12          1
patient 13          1
patient 14          1
patient 15          0
patient 16          0
patient 17          0
...
patient 62          0
patient 63          0
patient 64          0
```

We consider a subset of all the genes in the data set to be involved in patterns based on the support we found in the literature. See [Ramazzotti et al. \(2015\)](#) as a reference.

```
> gene.hypotheses = c('KRAS', 'NRAS', 'IDH1', 'IDH2', 'TET2', 'SF3B1', 'ASXL1')
```

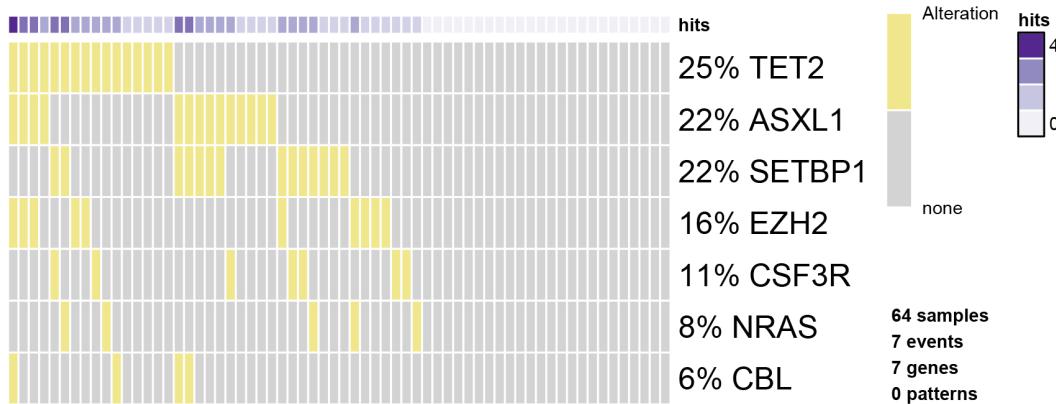
Regardless from which types of mutations we include, we select only the genes which appear altered in at least 5% of the patients. Thus, we first transform the data set into “*alterations*” (i.e., collapsing all the event types for the same gene) and then we consider only these events from the original data set.

```
> alterations = events.selection(as.alterations(aCML), filter.freq = .05)
*** Aggregating events of type(s) {Ins/Del, Missense point, Nonsense Ins/Del, Nonsense point}
     in a unique event with label "Alteration".
     Dropping event types Ins/Del, Missense point, Nonsense Ins/Del, Nonsense point for 23 genes.
*** Binding events for 2 datasets.
*** Events selection: #events=23, #types=1 Filters freq|in|out = \{TRUE, FALSE, FALSE\}
Minimum event frequency: 0.05 (3 alterations out of 64 samples).
Selected 7 events.
```

Selected 7 events, returning.

We now show a plot of the selected genes. Note that this plot has no title, as, by default, the function `events.selection` does not add any. The resulting figure is shown in Figure 2.

```
> oncrint(alterations, font.row = 12, cellheight = 20, cellwidth = 4)
*** Oncrint for ""
    with attributes: stage=FALSE, hits=TRUE
Sorting samples ordering to enhance exclusivity patterns.
```



**Figure 2:** The oncrint function in TRONCO. Result of the oncrint function in TRONCO on the aCML data set.

**Adding Hypotheses.** We now create the *data set* to be used for the inference of the progression model. We consider the original data set and from it we select all the genes whose mutations are occurring at least 5% of the times together with any gene involved in any hypothesis. To do so, we use the parameter `filter.in.names` as shown below.

```
> hypo = events.selection(aCML,
                           filter.in.names = c(as.genes(alterations),
                           gene.hypotheses))
*** Events selection: #events=31, #types=4 Filters freq|in|out = \{FALSE, TRUE, FALSE\}
[filter.in] Genes hold: TET2, EZH2, CBL, ASXL1, SETBP1 ... [10/14 found].
Selected 17 events, returning.
> hypo = annotate.description(hypo, 'CAPRI - Bionformatics aCML data (selected events)')
```

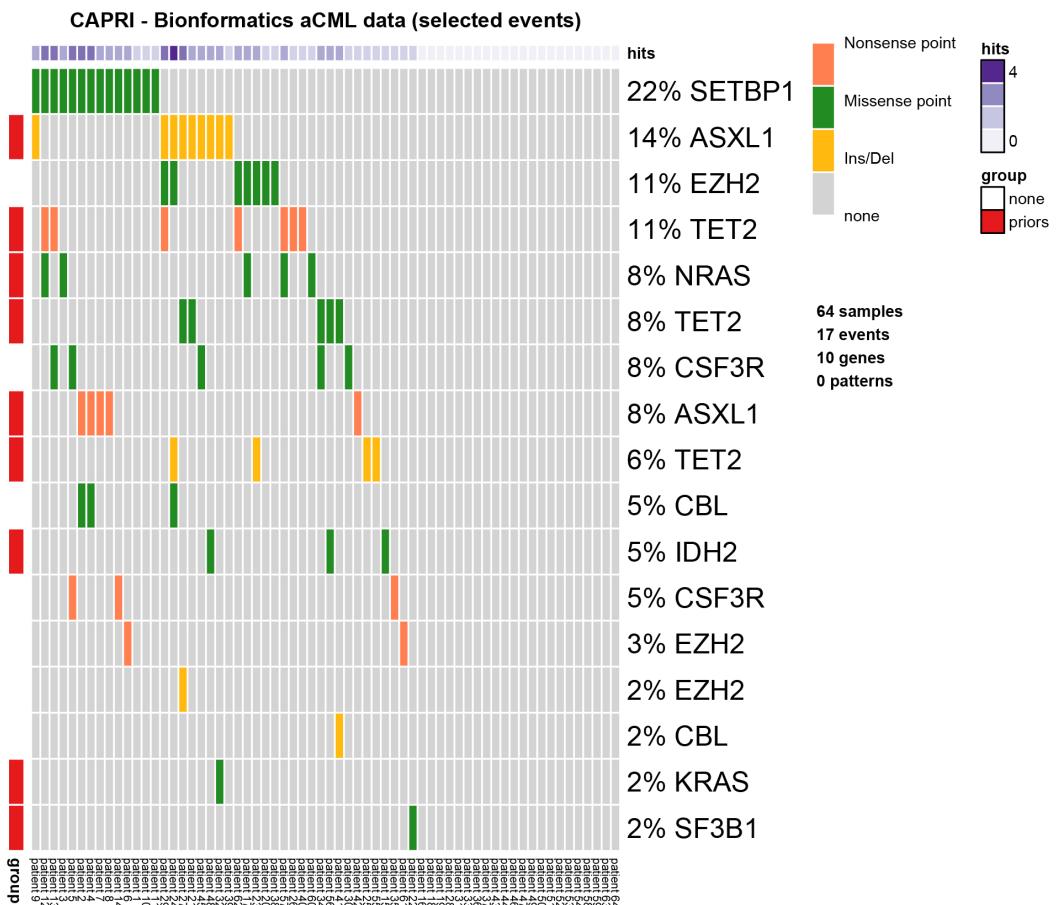
We now call `oncrint` of this latest data set where we annotate the genes in `gene.hypotheses` in order to identify them in Figure 3. The sample names are also shown.

```
> oncrint(hypo,
           gene.annot = list(priors = gene.hypotheses),
           sample.id = T,
           font.row = 12,
           font.column = 5,
           cellheight = 20,
           cellwidth = 4)
*** Oncrint for "CAPRI - Bionformatics aCML data (selected events)"
with attributes: stage=FALSE, hits=TRUE
Sorting samples ordering to enhance exclusivity patterns.
Annotating genes with RColorBrewer color palette Set1 .
```

We now also add the hypotheses that are described in CAPRI's manuscript. Hypothesis of hard exclusivity (XOR) for NRAS/KRAS events (Mutation). This hypothesis is tested against all the events in the data set.

```
> hypo = hypothesis.add(hypo, 'NRAS xor KRAS', XOR('NRAS', 'KRAS'))
```

We then try to include also a soft exclusivity (OR) pattern but, since its “signature” is the same of the hard one just included, it will not be included. The code below is expected to result in an error.



**Figure 3:** Annotated oncoprint. Result of the oncoprint function on the selected data set in TRONCO with annotations.

```
> hypo = hypothesis.add(hypo, 'NRAS or KRAS', OR('NRAS', 'KRAS'))
Error in hypothesis.add(hypo, "NRAS or KRAS", OR("NRAS", "KRAS")) :
[ERR] Pattern duplicates Pattern NRAS xor KRAS.
```

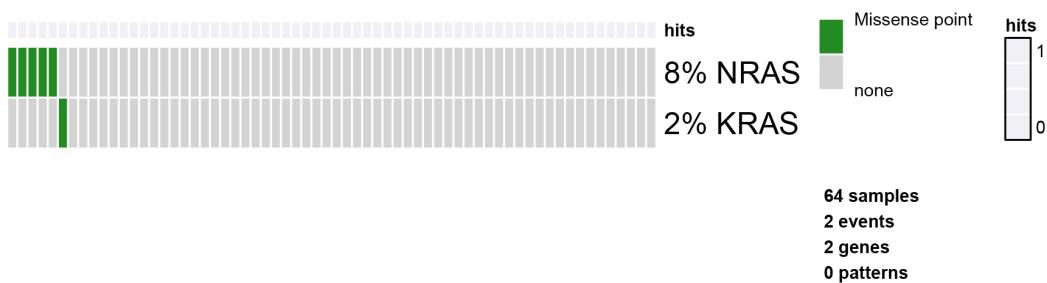
To better highlight the perfect (hard) exclusivity among NRAS/KRAS mutations, one can examine further their alterations. See Figure 4.

```
> oncoprint(events.selection(hypo,
+             filter.in.names = c('KRAS', 'NRAS')),
+             font.row = 12,
+             cellheight = 20,
+             cellwidth = 4)
*** Events selection: #events=18, #types=4 Filters freq|in|out = \{FALSE, TRUE, FALSE\}
[filter.in] Genes hold: KRAS, NRAS ... [2/2 found].
Selected 2 events, returning.
*** Oncoprint for ""
with attributes: stage=FALSE, hits=TRUE
Sorting samples ordering to enhance exclusivity patterns.
```

We repeated the same analysis as before for other hypotheses and for the same reasons, we will include only the hard exclusivity pattern.

```
> hypo = hypothesis.add(hypo, 'SF3B1 xor ASXL1', XOR('SF3B1', OR('ASXL1')), '*')
> hypo = hypothesis.add(hypo, 'SF3B1 or ASXL1', OR('SF3B1', OR('ASXL1')), '*')
Error in hypothesis.add(hypo, "SF3B1 or ASXL1", OR("SF3B1", OR("ASXL1"))):
[ERR] Pattern duplicates Pattern SF3B1 xor ASXL1.
```

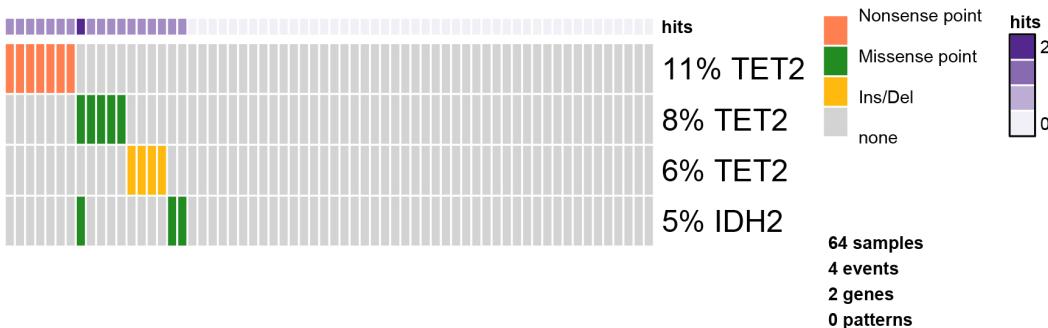
Finally, we now repeat the same for genes TET2 and IDH2. In this case three events for the gene TET2 are present: "Ins/Del", "Missense point" and "Nonsense point". For this reason, since we are not



**Figure 4:** RAS oncoprint. Result of the oncoprint function in TRONCO for only the RAS genes to better show their hard exclusivity pattern.

specifying any subset of such events to be considered, all TET2 alterations are used. Since the events present a perfect hard exclusivity, their patterns will be included as a XOR. See Figure 5.

```
> as.events(hypo, genes = 'TET2')
  type          event
gene 4 "Ins/Del"    "TET2"
gene 32 "Missense point" "TET2"
gene 88 "Nonsense point" "TET2"
> hypo = hypothesis.add(hypo, 'TET2 xor IDH2', XOR('TET2', 'IDH2'), '*')
> hypo = hypothesis.add(hypo, 'TET2 or IDH2', OR('TET2', 'IDH2'), '*')
> oncoprint(events.selection(hypo, filter.in.names = c('TET2', 'IDH2')), font.row = 12,
           cellheight = 20, cellwidth = 4)
*** Events selection: #events=21, #types=4 Filters freq|in/out = \{FALSE, TRUE, FALSE\}
[filter.in] Genes hold: TET2, IDH2 ... [2/2 found].
Selected 4 events, returning.
*** Oncoprint for ""
with attributes: stage=FALSE, hits=TRUE
Sorting samples ordering to enhance exclusivity patterns.
```



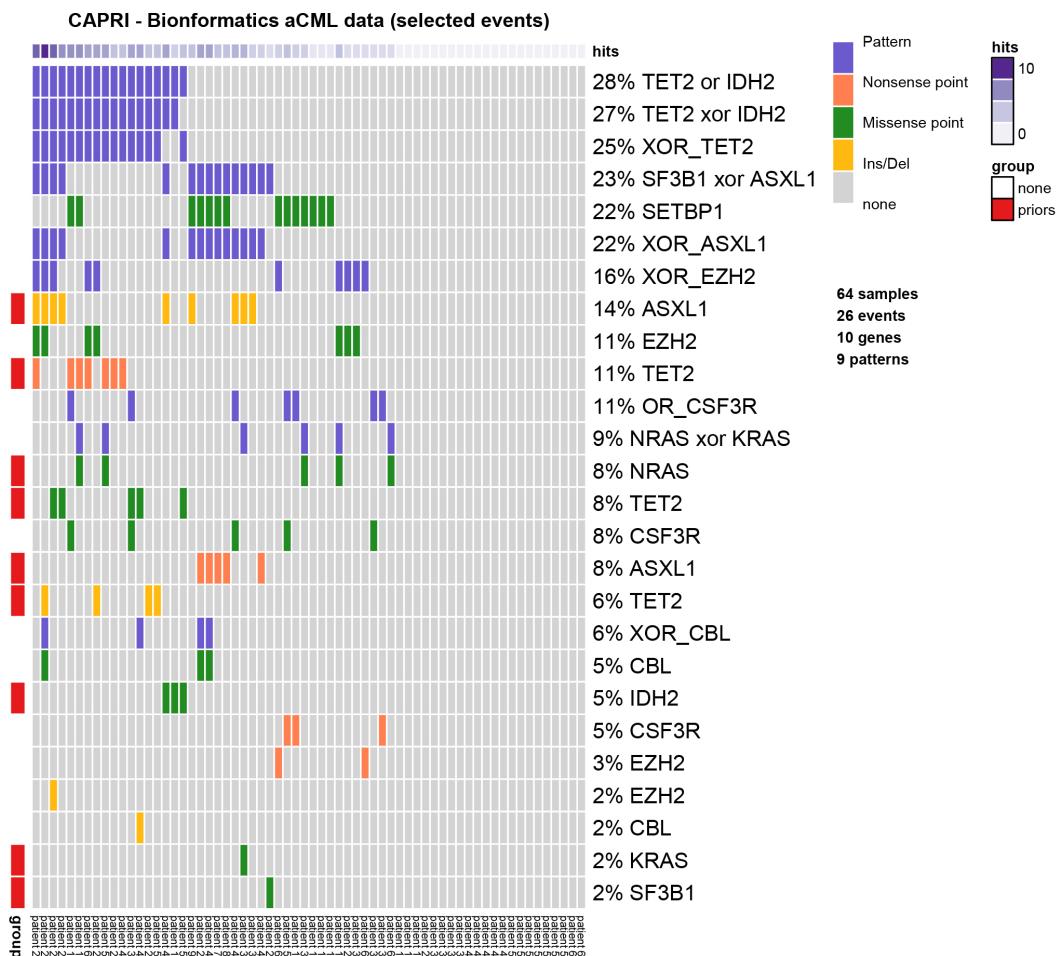
**Figure 5:** TET/IDH2 oncoprint. Result of the oncoprint function in TRONCO for only the TET/IDH2 genes.

We now finally add any possible group of homologous events. For any gene having more than one event associated to it, we also add a soft exclusivity pattern among them.

```
> hypo = hypothesis.add.homologous(hypo)
*** Adding hypotheses for Homologous Patterns
Genes: TET2, EZH2, CBL, ASXL1, CSF3R
Function: OR
Cause: *
Effect: *
Hypothesis created for all possible gene patterns.
```

The final data set that will be given as input to CAPRI is now finally shown. See Figure 6.

```
> oncoprint(hypo,
  gene.annot = list(priors = gene.hypotheses),
  sample.id = T,
  font.row = 10,
  font.column = 5,
  cellheight = 15,
  cellwidth = 4)
*** Oncoprint for "CAPRI - Bionformatics aCML data (selected events)"
with attributes: stage=FALSE, hits=TRUE
Sorting samples ordering to enhance exclusivity patterns.
Annotating genes with RColorBrewer color palette Set1 .
```



**Figure 6:** Final data set for CAPRI. Result of the oncoprint function in TRONCO on the data set used in [Ramazzotti et al. \(2015\)](#).

**Reconstructing Progression Models.** We next infer the model by running the CAPRI algorithm with its default parameters: we use both AIC and BIC as regularizers; Hill-climbing as heuristic search of the solutions; and exhaustive bootstrap (`nboot` replicates or more for Wilcoxon testing, i.e., more iterations can be performed if samples are rejected), p-value set at 0.05. We set the seed for the sake of reproducibility.

```
> model = tronco.capri(hypo, boot.seed = 12345, nboot = 10)
*** Checking input events.
*** Inferring a progression model with the following settings.
  Dataset size: n = 64, m = 26.
  Algorithm: CAPRI with "bic, aic" regularization and "hc" likelihood-fit strategy.
  Random seed: 12345.
```

```

Bootstrap iterations (Wilcoxon): 10.
  exhaustive bootstrap: TRUE.
  p-value: 0.05.
  minimum bootstrapped scores: 3.
*** Bootstrapping selective advantage scores (prima facie).
  Evaluating "temporal priority" (Wilcoxon, p-value 0.05)
  Evaluating "probability raising" (Wilcoxon, p-value 0.05)
*** Loop detection found loops to break.
  Removed 26 edges out of 68 (38%)
*** Performing likelihood-fit with regularization bic.
*** Performing likelihood-fit with regularization aic.
The reconstruction has been successfully completed in 00h:00m:02s

```

We then plot the model inferred by CAPRI with BIC as a regularizer and we set some parameters to get a good plot; the confidence of each edge is shown both in terms of temporal priority and probability raising (selective advantage scores), and hypergeometric testing (statistical relevance of the data set of input). See Figure 7.

```

> tronco.plot(model,
  fontsize = 13,
  scale.nodes = .6,
  regularization = "bic",
  confidence = c('tp', 'pr', 'hg'),
  height.logic = 0.25,
  legend.cex = .5,
  pathways = list(priors = gene.hypotheses),
  label.edge.size = 5)
*** Expanding hypotheses syntax as graph nodes:
*** Rendering graphics
  Nodes with no incoming/outgoing edges will not be displayed.
  Annotating nodes with pathway information.
  Annotating pathways with RColorBrewer color palette Set1 .
  Adding confidence information: tp, pr, hg
  RGraphviz object prepared.
  Plotting graph and adding legends.

```

**Bootstrapping the Data.** Finally, we perform non-parametric bootstrap as a further estimation of the confidence in the inferred results. See Figure 8.

```

> model.boot = tronco.bootstrap(model, nboot = 10)
Executing now the bootstrap procedure, this may take a long time...
Expected completion in approx. 00h:00m:03s
*** Using 7 cores via "parallel"

*** Reducing results

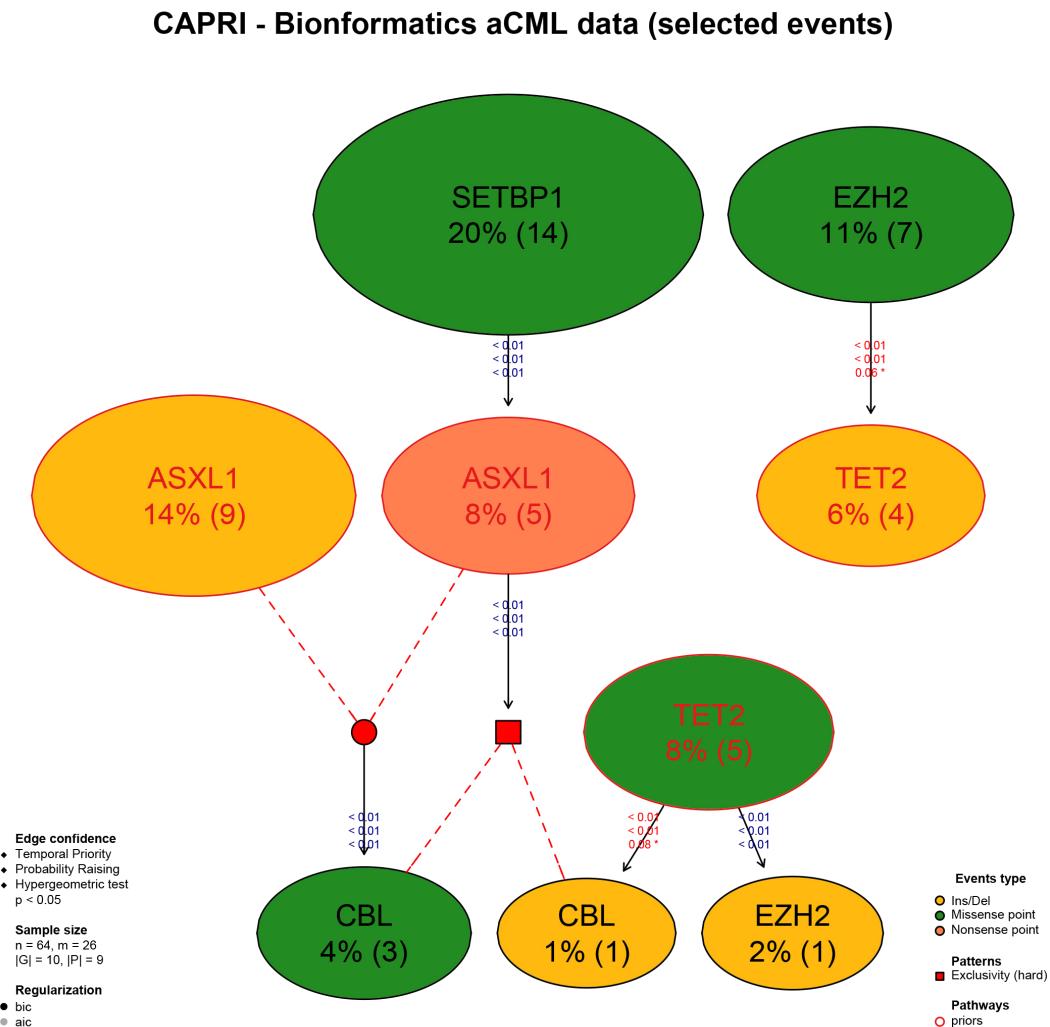
```

Performed non-parametric bootstrap with 10 resampling and 0.05 as pvalue for the statistical tests.

```

> tronco.plot(model.boot,
  fontsize = 13,
  scale.nodes = 0.6,
  regularization = "bic",
  confidence = c('npb'),
  height.logic = 0.25,
  legend.cex = 0.5,
  pathways = list(priors = gene.hypotheses),
  label.edge.size = 10)
*** Expanding hypotheses syntax as graph nodes:
*** Rendering graphics
  Nodes with no incoming/outgoing edges will not be displayed.
  Annotating nodes with pathway information.
  Annotating pathways with RColorBrewer color palette Set1 .
  Adding confidence information: npb

```



**Figure 7:** Reconstruction by CAPRI. Result of the reconstruction by CAPRI on the input data set.

RGraphviz object prepared.

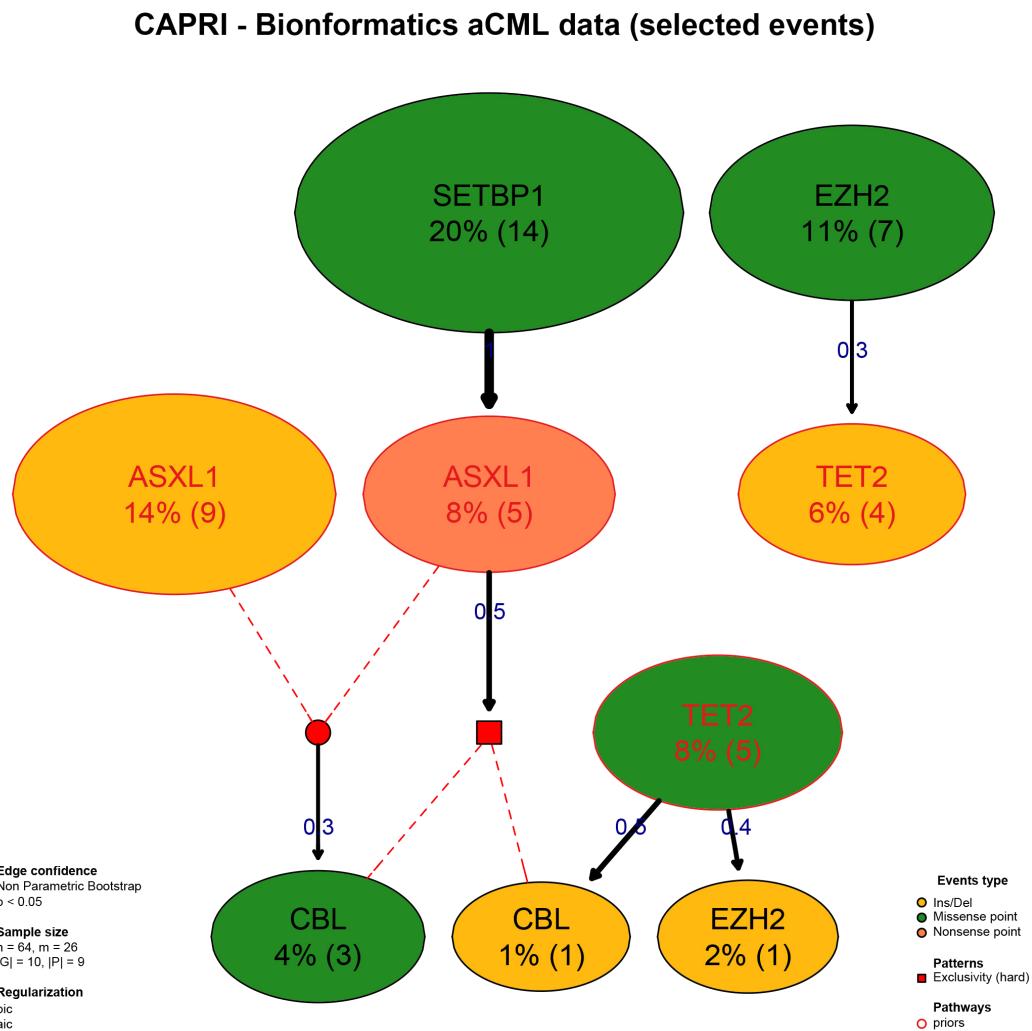
Plotting graph and adding legends.

We now conclude this analysis with an example of inference with the CAPRESE algorithm. As CAPRESE does not consider any pattern as input, we use the data set shown in Figure 3. These results are shown in Figure 9.

```
> model.boot.caprese = tronco.bootstrap(tronco.caprese(hypo))
*** Checking input events.
*** Inferring a progression model with the following settings.
    Dataset size: n = 64, m = 17.
    Algorithm: CAPRESE with shrinkage coefficient: 0.5.
The reconstruction has been successfully completed in 00h:00m:00s
Executing now the bootstrap procedure, this may take a long time...
Expected completion in approx. 00h:00m:00s
```

Performed non-parametric bootstrap with 100 resampling and 0.5 as shrinkage parameter.

```
> tronco.plot(model.boot.caprese,
              fontsize = 13,
              scale.nodes = 0.6,
              confidence = c('npb'),
              height.logic = 0.25,
```



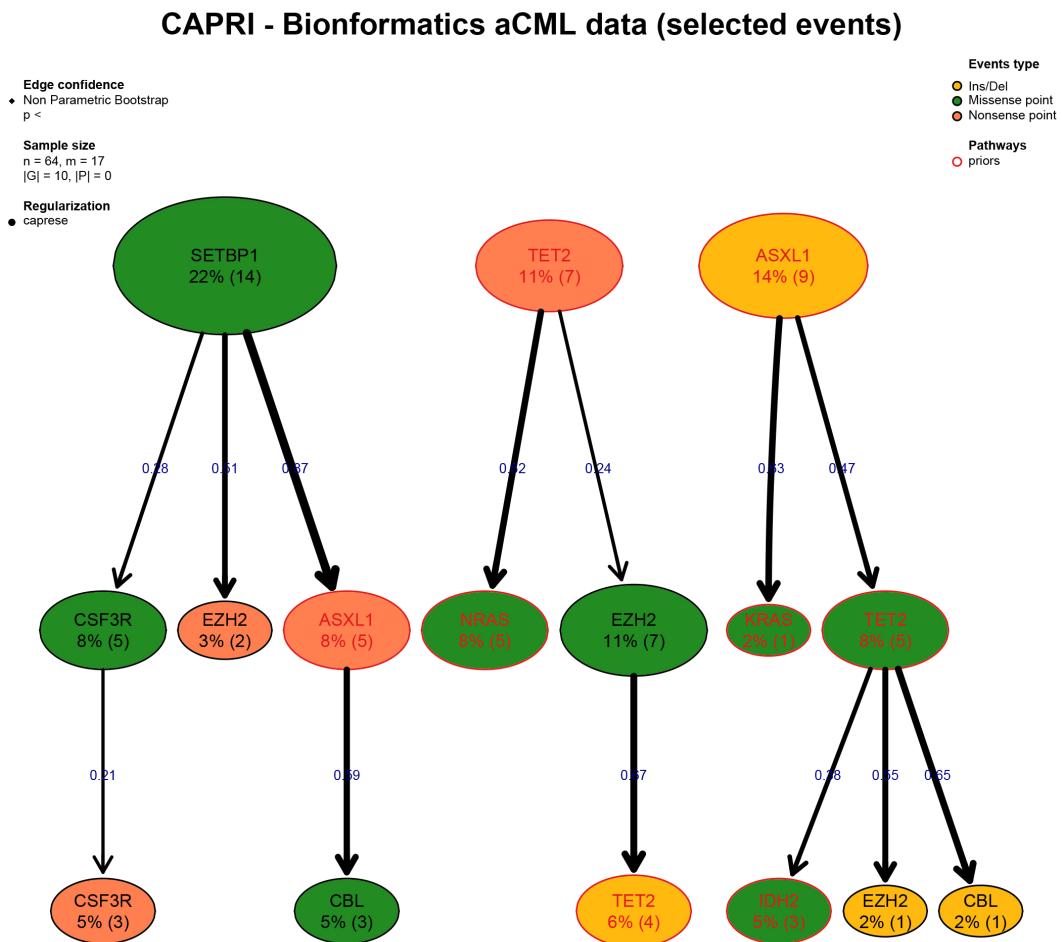
**Figure 8:** Reconstruction by CAPRI and Bootstrap. Result of the reconstruction by CAPRI on the input data set with the assessment by non-parametric bootstrap.

```

legend.cex = 0.5,
pathways = list(priors = gene.hypotheses),
label.edge.size = 10,
legend.pos = "top")
*** Expanding hypotheses syntax as graph nodes:
*** Rendering graphics
Nodes with no incoming/outgoing edges will not be displayed.
Annotating nodes with pathway information.
Annotating pathways with RColorBrewer color palette Set1 .
Adding confidence information: npb
RGraphviz object prepared.
Plotting graph and adding legends.
  
```

## Conclusions

We have described **TRONCO**, an R package that provides state-of-the-art techniques to support the user during the analysis of cross-sectional genomic data with the aim of understanding cancer evolution. In the current version, **TRONCO** implements the CAPRESE and CAPRI algorithms for cancer progression inference together with functionalities to load input cross-sectional data, set up the execution of the algorithms, assess the statistical confidence in the results, and visualize the inferred models.



**Figure 9:** Reconstruction by CAPRESE and Bootstrap. Result of the reconstruction by CAPRESE on the input data set with the assessment by non-parametric bootstrap.

**Financial support.** MA, GM, GC, AG, DR acknowledge Regione Lombardia (Italy) for the research projects RetroNet through the ASTIL Program [12-4-5148000-40]; U.A 053 and Network Enabled Drug Design project [ID14546A Rif SAL-7], Fondo Accordi Istituzionali 2009. BM acknowledges funding by the NSF grants CCF-0836649, CCF-0926166 and a NCI-PSOC grant.

## Bibliography

- Ö. Babur, M. Gönen, B. A. Aksoy, N. Schultz, G. Ciriello, C. Sander, and E. Demir. Systematic identification of cancer driving signaling pathways based on mutual exclusivity of genomic alterations. *bioRxiv*, page 009878, 2014. [p45]
- R. Beroukhim, G. Getz, L. Nghiemphu, J. Barretina, T. Hsueh, D. Linhart, I. Vivanco, J. C. Lee, J. H. Huang, S. Alexander, et al. Assessing the significance of chromosomal aberrations in cancer: methodology and application to glioma. *Proceedings of the National Academy of Sciences*, 104(50): 20007–20012, 2007. [p41]
- G. Caravagna, A. Graudenzi, D. Ramazzotti, R. Sanz-Pamplona, L. De Sano, G. Mauri, V. Moreno, M. Antoniotti, and B. Mishra. Algorithmic Methods to Infer the Evolutionary Trajectories in Cancer Progression. *Submitted. Available on bioRxiv.org, http://dx.doi.org/10.1101/027359.*, 2015. [p39, 40]
- E. Cerami, J. Gao, U. Dogrusoz, B. E. Gross, S. O. Sumer, B. A. Aksoy, A. Jacobsen, C. J. Byrne, M. L. Heuer, E. Larsson, et al. The cBio Cancer Genomics Portal: An Open Platform for Exploring Multidimensional Cancer Genomics Data. *Cancer discovery*, 2(5):401–404, 2012. [p41, 42, 43]

- J. Edmonds. Optimum Branchings. *Journal of Research of the National Bureau of Standards B*, 71B(4):233–240, 1967. [p41]
- M. Hofree, J. P. Shen, H. Carter, A. Gross, and T. Ideker. Network-based stratification of tumor mutations. *Nature methods*, 10(11):1108–1115, 2013. [p45]
- A. Jacobsen. R-Based API for Accessing the MSKCC Cancer Genomics Data Server. <https://cran.r-project.org/web/packages/cgdssr/>, 2011. [p43]
- D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009. [p42]
- NCI and the NHGRI. The Cancer Genome Atlas. <http://cancergenome.nih.gov/>, 2005. [p39]
- L. Olde Loohuis, G. Caravagna, A. Graudenzi, D. Ramazzotti, G. Mauri, M. Antoniotti, and B. Mishra. Inferring Tree Causal Models of Cancer Progression with Probability Raising. *PLoS One*, 9(12), 2014. [p39, 40, 46]
- R. Piazza, S. Valletta, N. Winkelmann, S. Redaelli, R. Spinelli, A. Pirola, L. Antolini, L. Mologni, C. Donadoni, E. Papaemmanuil, et al. Recurrent SETBP1 mutations in atypical chronic myeloid leukemia. *Nature Genetics*, 45(1):18–24, 2013. [p48]
- D. Ramazzotti, G. Caravagna, L. Olde-Loohuis, A. Graudenzi, I. Korsunsky, G. Mauri, M. Antoniotti, and B. Mishra. CAPRI: Efficient Inference of Cancer Progression Models from Cross-sectional Data. *Bioinformatics*, page btv296, 2015. [p39, 40, 41, 46, 48, 49, 53]
- C.-F. J. Wu. Jackknife, bootstrap and other resampling methods in regression analysis. *The Annals of Statistics*, 14(4):1261–1295, 1986. [p47]

*Marco Antoniotti, marco.antoniotti@unimib.it  
Dipartimento di Informatica Sistemistica e Comunicazione,  
Università degli Studi Milano-Bicocca  
Milano, Italy*

*Milan Center for Neuroscience,  
Università degli Studi Milano-Bicocca  
Milano, Italy*

*Giulio Caravagna, giulio.caravagna@unimib.it  
Dipartimento di Informatica Sistemistica e Comunicazione,  
Università degli Studi Milano-Bicocca  
Milano, Italy*

*Luca De Sano, luca.desano@gmail.com  
Dipartimento di Informatica Sistemistica e Comunicazione,  
Università degli Studi Milano-Bicocca  
Milano, Italy*

*Alex Graudenzi, alex.graudenzi@unimib.it  
Dipartimento di Informatica Sistemistica e Comunicazione,  
Università degli Studi Milano-Bicocca  
Milano, Italy*

*Institute of Molecular Bioimaging and Physiology of the Italian National Research Council (IBFM-CNR),  
Milano, Italy*

*Giancarlo Mauri, mauri@disco.unimib.it  
Dipartimento di Informatica Sistemistica e Comunicazione,  
Università degli Studi Milano-Bicocca  
Milano, Italy*

*Bud Mishra, mishra@nyu.edu  
Courant Institute of Mathematical Sciences,  
New York University  
New York, USA*

*Daniele Ramazzotti, daniele.ramazzotti@disco.unimib.it  
Dipartimento di Informatica Sistemistica e Comunicazione,  
Università degli Studi Milano-Bicocca  
Milano, Italy*

# diverse: an R Package to Analyze Diversity in Complex Systems

by Miguel R. Guevara, Dominik Hartmann, and Marcelo Mendoza

**Abstract** The package **diverse** provides an easy-to-use interface to calculate and visualize different aspects of diversity in complex systems. In recent years, an increasing number of research projects in social and interdisciplinary sciences, including fields like innovation studies, scientometrics, economics, and network science have emphasized the role of diversification and sophistication of socioeconomic systems. However, so far no dedicated package exists that covers the needs of these emerging fields and interdisciplinary teams. Most packages about diversity tend to be created according to the demands and terminology of particular areas of natural and biological sciences. The package **diverse** uses interdisciplinary concepts of diversity—like variety, disparity and balance—as well as ubiquity and revealed comparative advantages, that are relevant to many fields of science, but are in particular useful for interdisciplinary research on diversity in socioeconomic systems. The package **diverse** provides a toolkit for social scientists, interdisciplinary researcher, and beginners in ecology to (i) import data, (ii) calculate different data transformations and normalization like revealed comparative advantages, (iii) calculate different diversity measures, and (iv) connect **diverse** to other specialized R packages on similarity measures, data visualization techniques, and statistical significance tests. The comprehensiveness of the package, from matrix import and transformations options, over similarity and diversity measures, to data visualization methods, makes it a useful package to explore different dimensions of diversity in complex systems.

## Introduction

While measuring diversity is a natural topic in ecology and biology, the rise of complexity and big data research has also provided new opportunities for researchers in various fields of social sciences to understand the evolution of diversity in social, economic, and political systems (Nature, 2014).

Today, a large range of scientific fields make use of diversity measures, including ecologists calculating the diversity of species (Humphries et al., 1994), sociologists measuring the structure of communities (Haughton and Mukerjee, 1995), economists studying the diversification of exports or financial assets (Hidalgo and Hausmann, 2009), scientometrists analyzing the diversity and interdisciplinarity of research fields (Rafols, 2014; Chavarro et al., 2014; Wagner et al., 2011), and computer scientists searching for new diversity methods to ensemble algorithms (Kuncheva and Whitaker, 2003).

Consequently, different fields of science have created several specialized R packages on diversity. This includes packages that allow for the analysis of species and biodiversity, (e.g. **entropart** (Marcon and Hérault, 2015), **vegan** (Oksanen et al., 2016), **biodiversityR** (Kindt and Coe, 2005)), social distances (e.g. **Blaunet** (Wang et al., 2016)), genetics (e.g. **diveRsity** (Keenan et al., 2013)), biological systems (e.g. **divo** (Pietrzak et al., 2016)), functional ecology (e.g. **FD**, (Laliberté and Legendre, 2010)), species complexity (e.g. **hierDiversity** (Marion et al., 2015)), bootstrapping diversity indices (e.g. **simboot** (Scherer and Pallmann, 2014)), disparity of phylogenetic trees (e.g. **treescape** (Jombart et al., 2016)) or phylogenetic patterns (e.g. **SYNCSA** (Debastiani and Pillar, 2012)).

Most packages on diversity are in the fields of ecology, biology and other natural sciences. Each discipline and respective package uses the particular terminology of its scientific field. It is important to translate the existing mathematical diversity formulas into the relevant concepts and language of each community, and thereby also helps to create new specialized measures considering the particular research topics and demands of each community. But the thematic specialization can also make interdisciplinary communication difficult and reduces the chances of the adoption of these new measures, concepts and specialized packages by researchers outside of the particular scientific field.

In recent years, an increasingly large number of research projects in social and interdisciplinary sciences are exploring the role of diversity in complex socioeconomic systems. These new approaches in social and interdisciplinary sciences use existing diversity concepts from biological and natural sciences, but also have their own particular needs and concepts. For instance, recent work in economics, scientometrics and network science has highlighted the importance of diversification processes in complex systems, such as research, financial and energy portfolios, cultural diversity, the diversity of ties in social and economic networks, or the emergence of new or related scientific and economic fields (Hidalgo et al., 2007; Frenken et al., 2007; Rafols et al., 2010; Chavarro et al., 2014; Guevara et al., 2016; Eagle et al., 2010; Farchy and Ranaivoson, 2011).

Here we present the package **diverse** which aims to provide a useful toolkit for social scientists and interdisciplinary teams to measure and visualize diversity in socioeconomic systems, by providing

several of the most used measures of diversity and allowing for versatility with existing R packages on diversity, focusing, for example, on the calculation of similarity and distance measures ([proxy](#) (Meyer and Buchta, 2015)); bias corrected diversity measures ([entropart](#) (Marcon and Héault, 2015)); or the visualization of diversity in matrices, treemaps and networks ([pheatmap](#) (Kolde, 2015), [treemap](#) (Kindt and Coe, 2005), and [igraph](#) (Kindt and Coe, 2005)).

The package applies a diversity taxonomy that includes the variety, balance and disparity of complex systems (Stirling, 2007). The package **diverse** allows researchers to:

1. Read input and process data from complex systems in a simple manner.
2. Compute some of the most commonly used measures of diversity across sciences—including Shannon-Entropy, Herfindahl-Hirschman Index, Gini-Simpson Index or Berger-Parker Index.
3. Calculate complementary measures that are related to diversity, such as ubiquity, disparity or similarity between categories and entities.
4. Apply advanced diversity measures such as Rao-Stirling diversity and other diversity measures including weighting parameters.
5. Visualize different dimensions of diversity as variety, balance, or disparity.

The package **diverse** is available within CRAN. The newest development version is accessible at the branch *development* of the Git repository [github.com/mguevara/diverse](https://github.com/mguevara/diverse). In this Git repository interested users are also very welcome to submit issues.

The remainder of the article is organized as follows. In Section [Diversity](#) we describe different dimensions of diversity. In Section [Input data](#) we explain which type of data can be read/imported into the package and how it can be normalized, using for instance either binary, absolute or relative values. In Section [Measuring diversity](#) we present the measures available in this package discussing how researchers can use them to calculate different dimensions of diversity. In Section [Synthetic data and performance tests](#) we explain functions that are included in the package to simulate data and conduct bias, coverage, and performance tests. In Section [Conclusions](#) we summarize and briefly discuss the limitations and advantages of the package.

## Diversity

In this section we explain key properties of diversity with the help of example datasets.

### Example data

To illustrate the use of the package **diverse**, we will work with three datasets: Pantheon, Scidat and Geese.

- Pantheon is a sample of 10 countries from MIT's Pantheon project ([Macro Connections MIT MediaLab, 2014](#)). This dataset allows for a comparison of the diversity of occupations of the globally famous people from each country. The complete dataset includes 11341 persons classified in 88 distinct occupations and assigned to 195 countries (Yu et al., 2016).
- Scidat is an aggregation of the number of scientific publications assigned to 27 areas of science. This dataset was aggregated over the raw data of [SCImago](#) (2007). Scidat includes a sample of 10 countries from the year 2013.
- The third dataset is on the geese population in the Netherlands and was published by the Sovon Dutch Centre for Field Ornithology ([Nederland, 2015](#)). This dataset presents observations of 4 species of geese over a period of 11 years.

The three datasets are included in the package **diverse**. The subset of the Pantheon dataset is included as a "data.frame" object and both the Scidat and Geese datasets are included as a "matrix" object.

### The actors and concepts of diversity

We use the term *entity* to describe the systems or agents that host a set of categories. Entities could be, for example, persons, companies, countries, regions, institutions, or years.

We also use the term *category* to identify the different types of species that define the diversity of an entity. Categories could include types of animals, species of plants, fields of research, taxonomies of products, or technologies. The package assumes that the imported dataset has a previously given classification scheme.

The terms *value* or *value of abundance* is used for the amount of a category in each entity. This could be the quantity of each species in an ecosystem, or the total value of the different types of export goods of an economy.

In Pantheon, entities are countries, categories are different types of occupations, and values are the respective number of globally famous persons a country has in each category. In Scidat, entities are countries, categories are SCImago's areas of science, and values are the total number of citable documents that a country has published in each area in 2013. In Geese, entities are years, categories are species of geese, and values are the number of each species of geese observed in the Netherlands in the respective year.

Pantheon is a good example of data where some entities have missing values in some categories. Scidat is a useful example where most entities have values in each category yet have very large absolute differences between their values. The Geese dataset is a good example of the temporal evolution of natural species.

It must be noted that in most diversity measures (e.g. variety or Shannon entropy) the information about the number and types of categories of a single entity is sufficient to calculate this entity's diversity. However **diverse** is oriented to also work with multiple entities. Therefore it allows for the calculation of different distance and similarity matrices across categories and entities, and uses these distance measures in diversity measures like the Rao-Stirling Index ([Stirling, 2007](#)). Moreover, **diverse** allows for the calculation of relative specialization measures like the activity index or revealed comparative advantages (RCAs) that takes the portfolio and size of other entities into account when evaluating their relative specialization or comparative advantages (e.g. Belgium versus USA) (see Section [Data transformation and normalization](#)). Subsequently, we will mainly use data examples with multiple entities and categories. Nonetheless, many measures embedded in **diverse** can also be used to track the evolution of the diversity within a single entity.

Regarding the concept of diversity, previous interdisciplinary studies on diversity ([Rao, 1982](#); [Stirling, 1998](#); [McDonald and Dimmick, 2003](#); [Stirling, 2007](#)) showed that the concept of diversity is related to three main questions:

1. How many categories does an entity (and/or does each entity in a system) have?
2. How much of each category does an entity (and/or each entity in a system) have?
3. How distinct are the categories of an entity (and/or the categories of each entity in a system)?

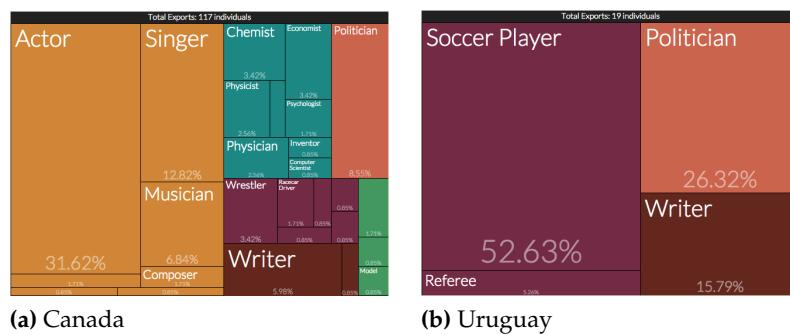
[Stirling \(1998, 2007\)](#) categorized these three properties of diversity as *variety*, *balance*, and *disparity*. Most diversity measures combine and emphasize these aspects with varying weights. Comprehensive measures take all three dimensions deliberately into account. Moreover data visualization methods and R packages like, for instance, **treemap** ([Tennekes, 2016](#)) or **igraph** ([Csardi and Nepusz, 2006](#)) can help to visualize these three dimensions of diversity. For instance, treemaps—allow for an emphasis on variety and balance ([Hausmann et al., 2011](#), p.105)—or network overlays maps allow for an emphasis on disparity ([Hidalgo et al., 2007](#); [Rafols et al., 2010](#)). The disparity dimension is often implied by a previous classification scheme, like a given classification of types of animals, scientific fields or exports, phylogenetic trees and/or it can alternatively be calculated based on a similarity or distance matrix (see also Section [Matrix of dissimilarities between entities](#)).

As an example, Figure 1 presents treemaps about the diversity of occupations of globally famous individuals from Canada and Uruguay according to MIT's Pantheon. Variety is represented by the number of boxes, balance is indicated by the differences in the size of the boxes (= percentage of the category), and disparity is represented by different colors.

First, regarding the *variety*, it is clear that Canada has a larger number of different occupations (27 boxes in Figure 1a) than Uruguay (4 boxes in Figure 1b). Second, regarding the *balance*, we can observe that Uruguay's concentration in terms of soccer players is very high (52.63%), while Canada's balance is less concentrated on one category, but is spread across more occupations of the Pantheon dataset. The R package **treemap** or other specialized data visualization programs like **D3plus** allows for the creation of such treemaps with different colors, text sizes and further visualization options.

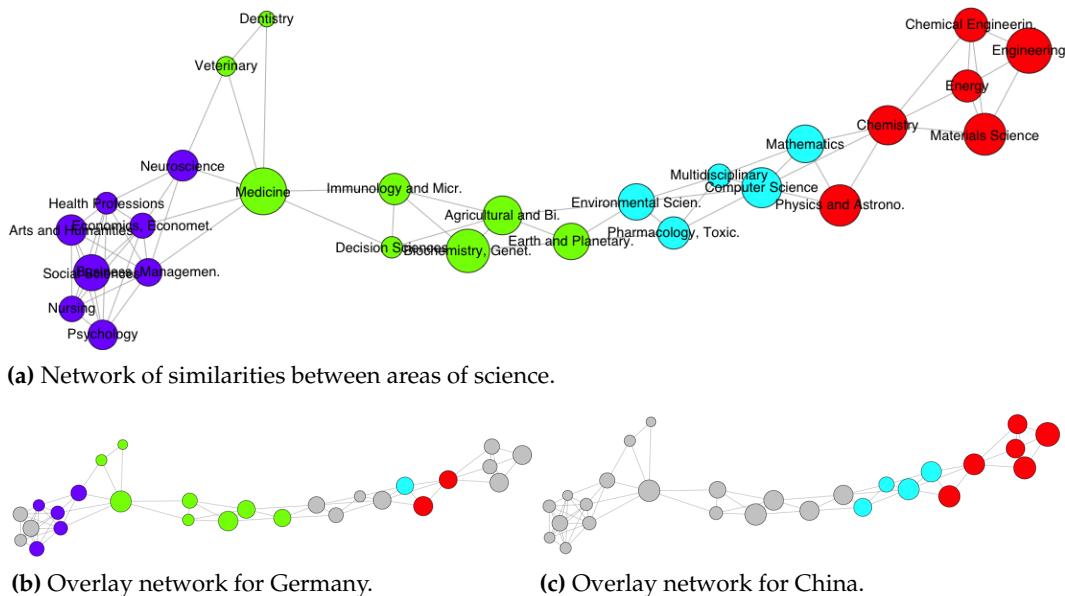
To illustrate disparity, Figure 2a shows a similarity network between areas of science that was obtained by considering each column as a vector of features (i.e. the number of articles of each country in that category) and then computing the cosine (dis-)similarity between those vectors. The pre-process options embedded in the package allows for the calculation of different types of similarity and distance matrices and considers both the absolute shares as well as relative strengths/specializations/comparative advantages of entities (see Section on [Data transformation and normalization](#)).

As expected, we can observe that "Social Sciences" are close to "Arts and Humanities," but more distant from "Mathematics" or "Engineering." We can also see the clustering between natural sciences and technological fields, like "Chemistry" and "Material Sciences." The disparity within a country is high if the dissimilarities between the areas/categories, in which the country has values, are also high.



**Figure 1:** Globally famous people according to Pantheon dataset. The size of the boxes is proportional to the number of people assigned to each occupation and born in that country. The color is according to main domains of occupations. Source <http://pantheon.media.mit.edu>.

In so-called overlay networks (Rafols et al., 2010), the values of entities (which are countries in the present dataset) are overlaid on the global network structure. Moreover, the variety is represented by the number of colored nodes, and the dimension balance can be represented by the size of the nodes. Figure 2b illustrates that in the Scidat example, Germany has *comparative advantages* (see the section on [Data transformation and normalization](#) for details) in many fields of science across the network, while China (Figure 2c) is more specialized (concentrated) in technological areas and engineering. In consequence the disparity, variety and balance in Germany are higher than in China. Such network overlay maps can, for instance, be made with the R package *igraph*.



**Figure 2:** Cosine similarity network of 27 areas of science obtained with Scidat dataset. Links represent the (dis)similarities between areas. Links below the threshold of 0.015 of Cosine similarity are not illustrated. The force-directed algorithm Fruchterman-Reingold was used for the network layout. The size of the nodes in 2a represents the total number of papers authored by the 10 countries included in Scidat; the size of the nodes in 2b and 2c is proportional to the papers authored for each country. Colors are according to communities detected by the algorithm *fastgreedy*. The grey-colored nodes identify areas with *Revealed Comparative Advantages RCA* below 1 (see section on [Data transformation and normalization](#) for details).

In the following sections we will detail how to import and transform data, and how to use the **diverse** package to quantify the described properties of diversity.

## Input data

This section details the type of the data object that is required by the package, how to import data from an external data file, and how to pre-process or normalize the raw data.

### Input formats

Since **diverse** was created to be able to work with multiple categories ( $N$ ) and multiple entities ( $M$ ) simultaneously, the data objects used for most of the functions in the package **diverse** can be either a data frame or a matrix of size  $M \times N$ .

In the case of a data frame—meaning that the data is shaped as an edge list—it has to have three columns in this order: entity, category and value. The first two columns are of the type "factor" and the third column is of type "numeric". The pantheon data frame is an example of this type of data object.

```
str(pantheon)
'data.frame':      119 obs. of  3 variables:
 $ Country    : Factor w/ 10 levels "Canada","Chile",...: 10 5 4 9 2 8 7 6 3 1 ...
 $ Occupation: Factor w/ 52 levels "Actor","Architect",...: 40 40 40 40 40 40 40 40 40 40
 $ Value       : int  6 2 8 5 10 9 17 36 38 10 ...
```

When the data is in a "matrix" format, each cell has to contain numeric values and the rownames and colnames must be defined with the names of entities and the names of categories. Non-existent values (NA) or 0 have to be used to indicate the lack of a category in an entity. The matrix of the scidat dataset is an example of this type of object.

```
str(scidat)
num [1:10, 1:27] 3507 35351 15603 1346 4158 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:10] "Argentina" "China" "Germany" "Hungary" ...
..$ : chr [1:27] "Agricultural and Biological Sciences" "Arts and Humanities"...
```

If the matrix has categories in the rows and entities in the columns, the parameter `category_row` must be set to TRUE when using the functions included in **diverse**. The matrix of the geese dataset is an example of this kind of object.

```
str(geese)
num [1:4, 1:11] 274 10788 4786 39273 247 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:4] "Little Grebe" "Crested Grebe" "Mute Swan" "Greylag Goose"
..$ : chr [1:11] "1996" "1997" "1998" "1999" ...
```

### Importing data

To simplify the input of data from external files, **diverse** includes the function `read_data()`. This function reads CSV files and automatically detects whether the data file is a matrix or an edge shape. Moreover, it retrieves a data frame ready to be passed to the parameter `data` of all the functions included in **diverse**. The user has to provide the path to the external CSV file by using the argument `path`.

In addition, to facilitate the import of data from different software formats, the function `read_data()` includes the parameter `type` that can be used to indicate whether the external data file comes for instance from *Stata* or *SPSS*. This functionality depends on the package [foreign](#) (R Core Team, 2015).

### Data transformation and normalization

Depending on the characteristics of the dataset, researchers often need to normalize, transform or filter the data before measuring diversity. For instance, in some cases the absolute quantity or share of each category in the portfolio of an entity is important. In other cases, the relative specialization and diversification of entities in comparison to a set of other entities (e.g. revealed comparative advantages of countries, or the relative activity in certain research fields) is more important. There are also cases where binary values (e.g. are certain categories present or not present) or discrete steps (e.g. not present, low, middle, high value) are important, depending on the respective research question. For this purpose, **diverse** includes the function `values()` which allows for the filtering and exclusion of data below a certain threshold value, and to binarize or normalize the data.

The normalization process could include forming proportion values, *Revealed Comparative Advantages* (RCA) (Balassa, 1986) and normalized RCAs (which is also called Activity Index). With the term *proportions* we refer to normalization within an entity (dividing the value of an entity in each category by the sum of values of the entity in all categories). The calculation of RCAs or the Activity Index is a normalization related to the other entities. For instance in economics, RCA computes the ratio between the proportion of a category within an entity, e.g. a country or region, and the proportion that represents that category in the global system (e.g. the world economy). The purpose of this measure is to understand in which categories an entity is relatively more specialized than others and thus seems to have a comparative advantage (Balassa, 1965). Typically, values of RCA greater than 1 are considered to "reveal" comparative advantages in the respective categories. Values below 1 reveal comparative disadvantages. The same idea can be found in Scientometrics where the RCAs are normalized between  $-1$  and  $1$ , and named "Activity Index." Both options, RCA and Activity Index are included in **diverse**.

To use these functionalities, the arguments `norm`, `filter`, and `binary` should be used. Argument `norm` can be set, for instance, to '`p`' for proportions, '`rca`' for RCAs, or '`ai`' for Activity Index. The argument `filter` allows the user to indicate a threshold, below which all the values are discarded (replaced with NA). The argument `binary` has to be set to TRUE if binary values are required. If the three arguments are applied, then the function `values()` first applies the normalization, then the filter and finally creates binary values.

The following matrix visualizations show the importance of the normalization process in datasets like Scidat where most entities produce all categories and the absolute differences (e.g. between the values of a small and a large country) are very large.

```
library(pheatmap)
colfunc <- colorRampPalette(c("deepskyblue4", "deepskyblue", "cyan"))
plot_mat <- function(data)
  pheatmap(data, colfunc(100), cluster_rows = FALSE, cluster_cols = FALSE)

col_1 <- names(sort(colSums(values(scidat)))) #order
row_1 <- names(sort(rowSums(values(scidat)), decreasing = TRUE))
plot_mat(values(scidat)[row_1,col_1])
plot_mat(values(scidat, norm = 'p')[row_1,col_1])
plot_mat(values(scidat, norm = 'rca')[row_1,col_1])
plot_mat(values(scidat, norm = 'rca', filter = 1)[row_1,col_1])
```

In Figure 3a we see the absolute values of authored papers by country in each area. The large number of papers from the United States in "Medicine" and "Biochemistry", as well as from China in "Engineering" and "Material Sciences" are the outstanding features of this matrix. If we consider proportions instead of absolute values, we can observe that "Medicine" is an important field of science for most countries, while a large proportion of the publication portfolios in Argentina or Mexico are in agricultural and biological sciences (see Figure 3b). Moreover, if we want to compare the relative specialization and comparative advantages of each country within the global system, an RCA based matrix will be more useful. Figure 3c presents the values of all RCAs, while Figure 3d presents the values of an RCA matrix in which values below 1 are represented by empty cells.

## Measuring diversity

In this section we explain the measures included in the package **diverse** by illustrating their use with our sample datasets.

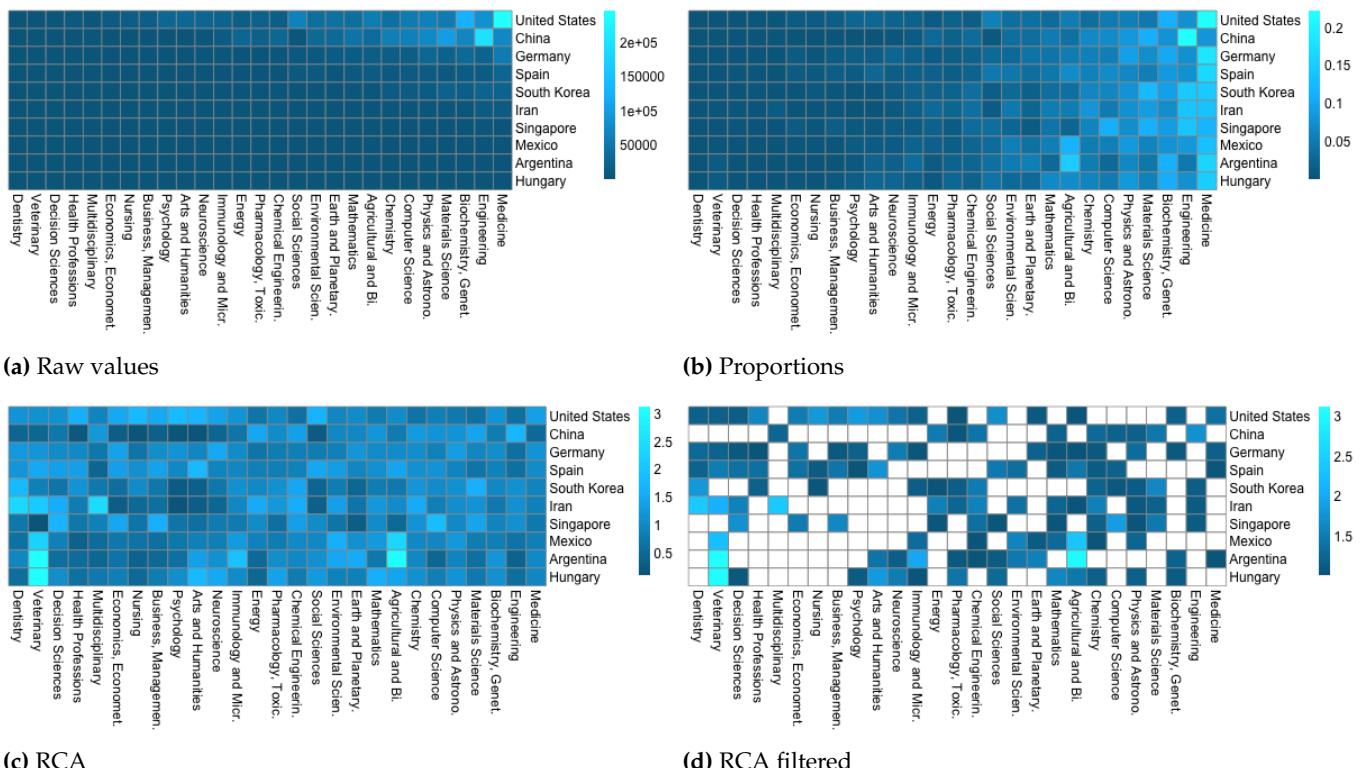
### Measures included

The diversity measures included in the package **diverse** allow for different dimensions of diversity—like variety, balance and disparity—to be analyzed separately or jointly.

To compute these measures, the main function `diversity()` must be used. All diversity measures available in the package **diverse** are listed in Table 1. These measures are organized from simple to complex, considering the properties of diversity they take into account.

Regarding the parameters of the function `diversity()`, the dataset to be analyzed should be provided in the `data` parameter, and the required diversity measure(s) should be provided in the `type` parameter.

The argument `data` has to fulfill the characteristics analyzed in the previous section on [Input data](#). The argument `type` can be a single `string` or a vector of strings, with either the complete name of the



**Figure 3:** Pheatmaps for matrices. The lighter the color, the higher the value. The white color represents empty cells.

ID	Measure	Formula	Reference
v	Variety	$v = \sum_i (p_i^0)$	
hh	Herfindahl–Hirschman Index	$HHI = \sum_i (p_i^2)$	Rhoades (1993)
b, gs	Blau Index, Gini-Simpson	$B = 1 - \sum_i (p_i^2) = 1 - HHI$	Blau (1977); Gini (1912)
s	Simpson	$D_S = \sum_i n_i(n_i - 1)/N_t(N_t - 1)$	Simpson (1949)
bp	Berger-Parker	$D_{BP} = \max_i (p_i)$	Berger and Parker (1970)
e	Shannon Entropy	$H = -\sum_i (p_i \log p_i)$	Shannon (1948)
ev	Pielou Evenness	$J = -\sum_i (p_i \log p_i) / \log v$	Pielou (1970)
re	Rényi-Entropy	${}^q H = (1 - q)^{-1} \log \left( \sum_i p_i^q \right)$	Rényi (1961)
hcdt	HCDT Entropy	${}^q H = (q - 1)^{-1} \left( 1 - \sum_i p_i^q \right)$	Havrda and Charvát (1967); Tsallis (1988)
hn	Hill Numbers	${}^q D_{HN} = \left( \sum_i p_i^q \right)^{1/(1-q)}$	Hill (1973)
d	Disparity	$DIS = \sum_{ij} d_{ij} / N$	
rao	Rao	$D_{RAO} = \sum_{ij} d_{ij} p_i p_j$	Rao (1982)
rs	Rao-Stirling	$\Delta = \sum_{ij} d_{ij}^\alpha (p_i p_j)^\beta$	Stirling (2007)

**Table 1:** Summary of measures available in the package **diverse**. The first block of measures are associated mainly with the dimensions variety and balance of the diversity, while the second block presents measures that use also the dimension disparity.  $C$  is the set of categories present in the entity.  $i, j \in C$ .  $i \neq j$  and  $ij \neq ji$ ;  $n_i$  is the value of abundance and  $p_i$  the proportion of the category  $i$  in the entity.  $v = n(C)$  is the number of categories present in the entity—the variety.  $N_t = \sum n_i$ . Here  $\log$  is the logarithm usually natural, and  $q, \alpha, \beta \geq 0$ . For HCDT and Rényi entropies when  $q \rightarrow 1$  converge to the Shannon entropy. Additionally, for Hill numbers, when  $q \rightarrow 1$ , it results in the exponential of Shannon Entropy.

measure or a mnemonic term (see column *ID* in Table 1). In the following sections we will explain the function `diversity()` and the related functions `variety()`, `balance()`, and `disparity()`.

### Variety or richness

Variety measures *how many categories or types an entity has*. Variety is useful as a first approach to the diversity of an entity since the number of categories (e.g. species, scientific fields or export categories) is easy to understand and calculate. Users can compute variety both within the function `diversity()` indicating `type='v'`, or with the function `variety()`. Both options return a data frame with the values of variety. In the case of the function `variety()` values are sorted in an decreasing order. For an increasing order, the argument `decreasing` should be set to `FALSE`.

For instance, we can compare the variety of the 10 countries included in our sample of Pantheon. Canada and China rank at the top of variety, while Uruguay and Vietnam rank at the bottom. In Scidata, US and Germany (see Figure 2b) have the highest level of variety, while China (see Figure 2c) and Mexico have the lowest variety. It is important to note that we are only considering fields of science in which these countries have Revealed Comparative Advantages (RCAs) equal to or higher than 1.

```
variety(data = pantheon)
      variety
Canada       27
China        24
...
Uruguay      4
Vietnam      4

#using function values() to normalize the dataset
scidat_rca_fil <- values(data = scidat, norm = 'rca', filter = 1)

variety(scidat_rca_fil)
      variety
United States 17
Germany       16
...
China          10
Mexico         9
```

Being related to the concept of variety, it is helpful in some cases to know the *ubiquity* or rareness of each category by considering its presence in all entities. Ubiquity could also be considered as the variety of entities that each category has (Hidalgo et al., 2007). We include this concept and measure through the function `ubiquity()` that returns the number of entities in which the category is present. A decreasing order is retrieved by default. In our sample of Pantheon “politicians” and “soccer players” are more common (ubiquitous) than “referees” or “wrestlers.”

```
ubiquity(data = pantheon)

      ubiquity
Politician    10
Writer         8
Soccer.Player  6
...
Referee        1
Wrestler       1
```

### Diversity measures that emphasize abundance and balance

Balance measures *how much of each category the entity has*. The raw indicators of balance are the values of abundance or the relative values of abundance which are the proportions  $p_i$  of each  $i$ -th category. The package **diverse** includes the function `balance()` which retrieves the matrix of entities-categories with their correspondent shares, proportions or probabilities.

The word *balance* is used when the values of abundance are more equally distributed across the categories. For a given variety, a more balanced system is considered more diverse. Extreme cases are those where the quantity of elements for each category is exactly the same (i.e. perfect balance) or conversely, where all the elements are concentrated in just one category (i.e. total concentration).

As pointed out by Tuomisto (2012), measuring balance alone is a complicated task because it is difficult to remove the effect of variety on it. The only measure of "balance" in a strict sense that is facilitated in this package is Pielou's evenness (Pielou, 1970), which according to Jost (2010) is also the best measure of balance.

However, **diverse** also allows for the calculation of a series of commonly used "balance" measures like the Herfindahl-Hirschman Index (HHI), Gini-Simpson or Blau-Index that emphasize the evenness or balance of a system (while also being affected by the variety of categories).

Diversity measures related to the property "balance" could be understood as statistical dispersion and are mainly a function of  $p_i$ . While some of them measure the evenness or *heterogeneity* of the distribution such as the *Blau Index*, others emphasize the *concentration*, such as the *Herfindahl-Hirschman Index (HHI)*.

The Herfindahl-Hirschman Index (HHI), for example, computes the probability that two individuals taken randomly belong to the same category. This probability is calculated with replacement, which means that after taking the first individual into account, it is replaced with an identical one; and thus neither affecting the total number of individuals in that category ( $n_i$ ) nor the total amount of individuals in the entity ( $N_t$ ). HHI is used in economics, for instance, to estimate the concentration of markets or wealth (Ceriani and Verme, 2011).

Taking into account that balance is the opposite to concentration, the *Gini-Simpson Index* ( $1 - HHI$ ) subtracts *HHI* from 1 to estimate balance. The same idea is behind the *Blau Index*. The Blau Index was created to measure the heterogeneity of social communities and its use is very common in sociology and other social sciences.

Similar to HHI, *Simpson* measure  $D_s$  has the same probabilistic idea of measuring concentration, but it computes the probability without replacement—meaning that the values of  $N_t$  and  $n_i$  decreases in 1 after the first probability is calculated (see Table 1). This measure of concentration and its equivalent balance or index of diversity ( $1 - D_s$ ) are widespread in ecology. Moreover, the reciprocal index ( $R_s = 1/D_s$ ) can be calculated.

In the following example, the *Herfindahl-Hirschman Index (HHI)*, the *Gini-Simpson Index* and the *Blau Index* from Pantheon are computed by using the function `diversity()`. We can observe that Uruguay and Vietnam have a higher HHI value and are thus more concentrated and less balanced than Canada and Chile. Note that the opposite occurs with the Gini-Simpson or Blau indexes. Besides the Gini-Simpson index, the concentration `gini.simpson.C` and the reciprocal of the concentration `gini.simpson.R` are also retrieved.

```
round(diversity(data = pantheon, type = c('hh', 'gs', 'b', 'ev')), 3)
      HHI gini.simpson gini.simpson.C gini.simpson.R blau.index evenness
Canada    0.372      0.628      0.372      2.689      0.628      0.843
Chile     0.133      0.867      0.133      7.538      0.867      0.959
...
Uruguay   0.235      0.765      0.235      4.263      0.765      0.820
Vietnam   0.139      0.861      0.139
```

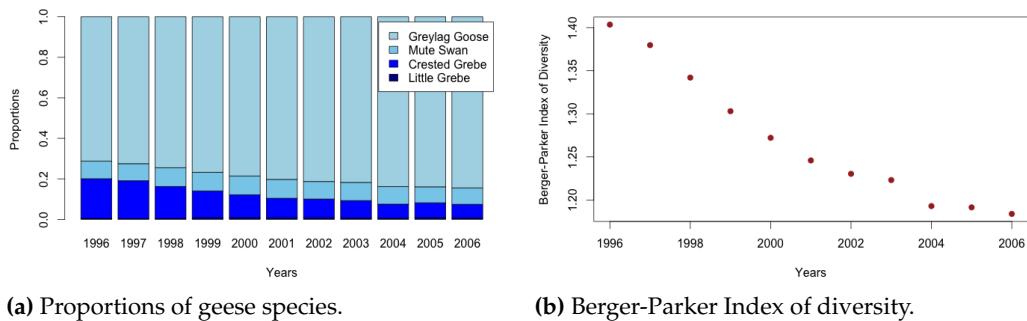
Graphical representations can help to understand the importance of balance and how it is captured by specific diversity measures. Figure 4a illustrates how the share of the dominant species Greylag Goose increases over time and how the share of Crested Grebe declines. The result is an unbalance between the species in this ecosystem.

The decrease in diversity of geese—understood here mainly as the balance of the abundance of different types of geese—can, for instance, be captured by Berger-Parker measures. The Berger-Parker Dominance Index ( $D_{BP}$ ) is a measure based on the dominant category ( $\max(p_i)$ ) and thus captures the dominance of the Greylag Goose. On the other hand, the Berger-Parker Index of Diversity ( $I_{BP} = 1/D_{BP}$ ) captures the balance between the species. Figure 4b shows how the Berger-Parker Index of Diversity decreases over time.

```
bal <- balance(geese, category_row = TRUE) #note the function balance
barplot(t(bal), legend = TRUE, xlab = "Years", ylab = "Proportions",
        col=c("darkblue","blue","sky blue", "light blue") )

bp <- diversity(geese, type = 'bp', category_row = TRUE)
plot(bp$berger.parker.I~rownames(bp), xlab = "Years",
      ylab = "Berger-Parker Index of Diversity", pch = 19, col = "brown")

diversity(data = geese, type = c('e','ev','s','bp'), category_row = TRUE)
```



**Figure 4:** Analysis of balance for Geese dataset.

	entropy	evenness	simpson.D	simpson.I	simpson.R	berger.parker.D	berger.parker.I
1996	0.7993160	0.5765846	0.5534977	0.4465023	1.806692	0.7124871	1.403534
1997	0.7764028	0.5600563	0.5674638	0.4325362	1.762227	0.7247953	1.379700
...							
2005	0.5790954	0.4177290	0.7160910	0.2839090	1.396471	0.8392823	1.191494
2006	0.5633026	0.4063369	0.7245616	0.2754384	1.380145	0.8446653	1.183901

### Entropy measures and Hill numbers

*Shannon Entropy* is a frequently used measure of balance and diversity. Entropy is a measure first created and used in information theory ([Shannon, 1948](#)) and has been widely adopted by other disciplines such as computation, ecology, and economy.

Entropy  $H$  measures the minimum volume of communication required to code a message. Furthermore, as pointed by [Hidalgo \(2015, p.17\)](#), “entropy is a measure of the multiplicity of states”. A high value of multiplicity of states (categories) implies more evenness and less concentration: as a consequence, the higher the variety and the balance, the higher the entropy. In the example above we could observe how the entropy in Geese decreased from 0.78 in 1996 to 0.56 in 2006. This is a consequence of an increase of the population of the dominant species.

A generalization of Shannon Entropy that is also included in **diverse** is Rényi’s entropy (see Table 1). Rényi’s entropy allows the users to give more or less relative importance to rare categories through the parameter  $q$ .

Another parameterized entropy is the HCDT entropy ([Havrda and Charvát, 1967; Daróczy, 1970; Tsallis, 1988](#)). It is noteworthy that Variety, Shannon, Blau’s and Berger-Parker’s indexes are special cases of HCDT (respectively with the parameter  $q = 0, 1, 2$  and infinity).

Finally, Hill numbers ([Hill, 1973](#)) are a mathematically unified family of diversity indexes that differ only by a parameter  $q$  and that take the effective number of categories into account, i.e. the number of equally abundant species that would be needed to give the same value of a diversity measure ([Chao et al., 2014b](#)). Hill numbers are of particular interest since entropy is not linear to the number of categories hosted by an entity [Jost \(2006\)](#). Moreover, several widely used diversity indexes, like variety/richness, Shannon entropy, Gini-Simpson Index, Rényi’s or HDCT entropy, can be obtained from Hill numbers ([Chao et al., 2014a](#)).

By using **diverse** we can observe the similarities between entropy measures and Hill numbers, when  $q$  has values of 0, 1 and 2.

When  $q = 0$ , variety, HCDT entropy and Hill numbers are the same. Rényi entropy is equal to  $\log(\text{variety})$ . When  $q = 1$ , Rényi entropy and HCDT entropy are equal to Shannon entropy ( $H$ ), while Hill numbers are equal to the exponential of  $H$ . When  $q = 2$ , HCDT entropy is equal to Gini-Simpson, while the Hill numbers index is equal to the reciprocal (`gini.simpson.R`) of the index of concentration of Gini (or Herfindahl-Hirschman Index (`gini.simpson.C`)).

```
diversity.pantheon, type=c("v", "hcdt", "hn", "re"), q=0)[1,]
  variety hcdt.entropy hill.numbers renyi.entropy
Canada      27          27          27      3.295837

diversity.pantheon, type=c("e", "re", "hcdt", "hn"), q=1)[2,]
  entropy renyi.entropy hcdt.entropy hill.numbers
Chile 1.626709     1.626709     1.626709      5.087107

diversity.pantheon, type=c("hcdt", "gs", "hn"), q=2)[3,]
```

```
hcdf.entropy gini.simpson gini.simpson.C gini.simpson.R hill.numbers
China      0.8168554    0.8168554     0.1831446      5.460167      5.460167
```

Generally for the three parameterized measures that are included in **diverse** (i.e. Rényi Entropy, HCDT and Hill numbers), the parameter value  $q = 0$  calculates variety,  $q < 1$  considers rare categories more important for diversity,  $q = 1$  considers all categories as equally important, and  $q > 1$  mainly shows the impact of dominant categories in diversity.

## Disparity

Another significant dimension of diversity is the disparity or dissimilarity between categories or entities (Stirling, 2007; Rafols and Meyer, 2009). Disparity is important for all diversity measures, though, often pre-given in form of classification schemes, like, for example, phylogenetic trees or types of species in ecology, or the type of research fields in scientometrics. Here, we explicitly take the diversity dimension disparity into account.

The dimension of disparity provides a notion of *how different the categories of an entity are*. For example the areas “Mathematics” and “Physics” are arguably more similar than “Mathematics” and “Nursing.” Measures of diversity, therefore, are also closely related to distance and similarity measures like Euclidean distances, cosine similarity, Jaccard-Index, or expert classifications of different categories.

## Matrix of dissimilarities between entities

Beside computing disparity, the dissimilarity matrix between entities is also useful for the visualization of networks, such as those proposed to evaluate economic complexity (Hidalgo et al., 2007; Hartmann et al., 2016) or the research capabilities of scholars (Guevara et al., 2016). Moreover, it helps to analyze the portfolio of entities in so-called network overlay maps and to explore the path of diversification as a function of the disparity in the network (Rafols et al., 2010; Guevara et al., 2016).

Based on the 10 countries included in Scidat, we calculate the dissimilarities between categories and then we create a network of areas of science in the following example. The resulting network is the one presented in Figure 2a in Section [Diversity](#).

```
adj <- dis_categories(data = scidat, method = 'cosine')
adj[adj > 0.015] <- 0 #filter

library(igraph)
g <- graph.adjacency(adjmatrix = adj, mode = 'undirected', weighted = TRUE)
totals <- colSums(values(scidat))
V(g)$size = log(totals[match(V(g)$name, names(totals))], base = 2) - 9
fc <- fastgreedy.community(g); colors <- rainbow(max(membership(fc)))
V(g)$color = colors[membership(fc)]
set.seed(67)
g$layout <- layout.fruchterman.reingold(g)
plot.igraph(g, vertex.label.cex = 0.9, vertex.label.font = 0,
            vertex.label.family = 'Helvetica', vertex.label.color='black', asp = FALSE)
```

## Calculating dissimilarities between entities

The function `dis_entities()` can be used to calculate a matrix of dissimilarities between entities. The following example computes the matrices of dissimilarities between countries (entities) for the 10 countries included in Scidat. In this example, Argentina is more similar to Mexico (0.04) and less similar to China (0.32). In addition, Germany is more similar to Hungary (0.02) and less similar to Singapore (0.10).

```
round(dis_entities(scidat, method = 'cosine'), 2)
          Argentina China Germany Hungary Iran Mexico Singapore...
Argentina      0.00  0.32   0.09   0.07  0.17   0.04    0.25
China         0.32  0.00   0.20   0.19  0.06   0.18    0.06
Germany       0.09  0.20   0.00   0.02  0.07   0.05    0.10
Hungary        0.07  0.19   0.02   0.00  0.07   0.03    0.11
Iran           0.17  0.06   0.07   0.07  0.00   0.07    0.05
Mexico         0.04  0.18   0.05   0.03  0.07   0.00    0.13
...
```

## Average or Sum Disparity

The function `disparity()` computes the average and/or the sum of dissimilarities among categories, either based on a given dissimilarity matrix of the user or through calculating the dissimilarity matrix within the function. The first case is based on a matrix of dissimilarities that the user provides in the argument `dis`. The dissimilarity matrix has to include the same names of the categories in the rownames and in the colnames.

In the second case, when the argument `dis` is not provided, `diverse` computes the disparities by using the dissimilarity matrix calculated by using the previously detailed function `dis_categories()`, as in the following example where the argument `dis` is not defined.

In this example with Scidat, we note that the average dissimilarities of categories in the US are greater than the disparities in Argentina or China.

```
scidat_rca_fil <- values(scidat, norm = 'rca', filter = 1)
disparity(scidat_rca_fil)
  disparity.sum disparity.mean
Argentina      121.12704    0.3450913
China          54.86895    0.1563218
...
Spain          147.35440    0.4198131
United States  190.86552    0.5437764
```

## Diversity measures that explicitly take variety, disparity and balance into account

The package includes also "full" measures of diversity that are able to capture variety, balance and disparity at the same time. These measures are *Rao* and *Rao-Stirling*, where the former is widespread in ecology, while the latter is more commonly applied in social sciences and scientometrics (Rafols, 2014; Wang et al., 2015).

Both measures compute the sum of the multiplication of the distances (disparity) and the proportions (balance) between the pairs of two distinct categories  $i$  and  $j$  (see Table 1). However, Rao-Stirling diversity allows users to assign the weights/parameters  $\alpha$  and  $\beta$  according to the importance of the disparity or balance, respectively.

Rao diversity is equivalent to Rao-Stirling diversity with the parameter values  $\alpha = \beta = 1$ . These values are also the default values in the function `diversity()`. Note that when the argument `dis` is not provided, the default method 'Euclidean distances' is used for the calculation of the dissimilarity matrix. Users can also provide their own dissimilarity matrix by using the argument `dis` in the function `diversity()`.

In the following example from Scidat, we calculate Rao diversity as well as the Rao-Stirling diversity with the parameter values  $\alpha = 0.7$  and  $\beta = 0.3$  and cosine dissimilarities between the entities. This example shows that Rao-Stirling diversity provides the possibility to emphasize different aspects of diversity. When we use the Rao Index, then Spain is considered to be more "diverse" than the US, but when we assign more importance to disparity, by increasing the parameter  $\alpha$  in the Rao-Stirling index, then the US is more "diverse."

```
scidat_rca_fil <- values(scidat, norm = 'rca', filter = 1)
diversity(data = scidat_rca_fil, type = c('rao', 'rs') ,
  alpha=0.7, beta = 0.3, method = 'cosine')
  rao.stirling      rao
  rao rao.stirling
Argentina     0.1526983    7.072576
China        0.1346935    4.814975
...
Spain         0.2137356   12.842799
United States 0.1874783   12.864261
```

Thus, the Rao-Stirling index and the package `diverse` allows the user to analyze the impact of different similarity measures as well as different weights of disparity and balance on the resulting diversity values and rankings.

It must be noted that, so far, we focus in "diverse" on the Stirling taxonomy (Stirling, 2007, 1998). In ecology another set of "similarity-based" measures has been developed and can be accessed in the package `entropart` and `treescape`.

## Synthetic data and performance tests

In this Section, we show how to use **diverse** to create synthetic data to the level of *individuals*, entities and datasets. An individual is an independent object that belongs to a category (e.g. a paper in a certain discipline of Scidat or a person in a certain type of occupation in Pantheon). Entities are constituted by a set of categories and their values of abundance. A set of several entities constitutes a synthetic dataset for the type of diversity measures we apply in **diverse**.

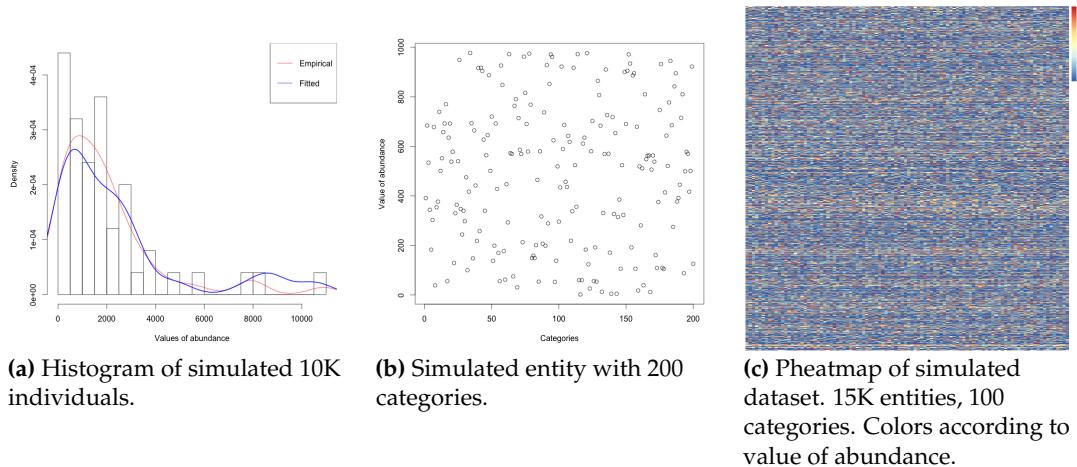
The functions included in **diverse** to simulate data are `sim_individuals()`, `sim_entities()` and `sim_dataset()`. In each function the user can define the size (i.e. number of individuals), the required level of variety or richness and the method to define the distribution or values of abundance.

For example, if we want to create individuals of an entity, we can use the function `sim_individuals()` to generate synthetic data with 10000 individuals assigned to 50 different species (categories). Moreover, the values of absolute abundance of species can be, for instance, distributed according to a log normal distribution with  $\mu = 0.50$  and  $\sigma = 1.183$  (see histogram in Figure 5a and (Beck and Schwanghart, 2010)).

```
set.seed(99)
synt_ind <- sim_individuals(n_categ=50, size=100000,
  category_prefix='ctg', type='log-normal', mean=0.507, sd=1.183)
hist(table(synt_ind), breaks = 30, xlab = "Values of abundance",
  probability = TRUE, main = NULL)
lines(density(table(synt_ind)), col="red")

library(fitdistrplus)
f <- fitdist(as.vector(table(synt_ind)), "lnorm")
x = rlnorm(50, mean=f$estimate['meanlog'][[1]], sd = f$estimate['sdlog'][[1]])
lines(density(x), col="blue", lwd=2)
legend("topright", legend = c('Empirical', 'Fitted'), col = c("red", "blue"), lty=1)
head(synt_ind)

[1] "ctg49" "ctg3"  "ctg41" "ctg49" "ctg4"   "ctg25"
```



**Figure 5:** Analysis of simulated data.

If the user wants to generate a simulated entity with values of abundance produced by the aggregation of individuals in categories, **diverse** provides the function `sim_entity()`. This function allows the user to define a distribution and/or a required number of categories (`n_categ()`). See Figure 5b.

```
sim_ent <- sim_entity(n_categ=200, values=sample(1:1000, replace=TRUE))
plot(sim_ent$value, ylab = "Value of abundance", xlab="Categories")
head(sim_ent)
Category Value
1          1    757
2          2    124
...
...
```

The simulation of a full dataset is also provided with **diverse**. The function `sim_dataset()` allows for users to define the number of categories in each entity (*variety*) as well as the number of required entities. A crucial argument of the function `sim_dataset()` is the vector of integers with the desired values of variety for each entity (`n_categ`). In the following example, we create a dataset of 1500 entities and 100 categories with random integer values of abundance between 10 and 5000. The values of variety for each entity are also randomly sampled between 1 and 100. The resulting dataset is retrieved as a data frame of values of abundance. By using the function `values()` we can plot this dataset as a matrix (see Figure 5c).

```

n_entities <- 1500
v_values <- sample(10:5000, size= n_entities, replace=TRUE)
v_n_categ <- sample(1:100, size = n_entities, replace=TRUE)
data_set <- sim_dataset(n_categ = v_n_categ, values= v_values,
  category_prefix = "C", category_random = TRUE)
pheatmap(values(data_set),cluster_rows = FALSE, cluster_cols = FALSE,
  show_rownames = FALSE, show_colnames = FALSE)
head(data_set)

...

```

## Performance

To test the performance of **diverse** we use the previously generated synthetic dataset (1500 entities by 100 categories), then we calculate the time used to perform two measures, namely Shannon entropy and Rao-Stirling diversity. Note that the second one is more time consuming since it involves the computation of a distance matrix. However, the time necessary to compute both measures is reasonable (0.021 and 2.697 seconds respectively). The time used to create the simulated dataset is more time consuming (35 secs.) since the dataset must ensure that the assigned number of categories for each entity accomplishes the requirements. Still it is a reasonable amount of time considering the dimensions and characteristics of the obtained data.

```

system.time(data_set <- sim_dataset(n_categ = v_n_categ, values= v_values,
  category_prefix = "C", entity_prefix = "E"))

  user  system elapsed
29.590   5.245  34.871
system.time(diversity(data_set, type=c("e")))
  user  system elapsed
  0.019   0.001   0.021
system.time(diversity(data_set, type=c("rs")))
  user  system elapsed
  2.478   0.206   2.697

```

## Coverage, biases and caveats

The package **diverse** is designed to work with datasets with a known number of categories and a comparatively low level of variety (i.e. scientific fields, occupations, or industrial sectors, in comparison to datasets in ecology with millions of species, including many unknown species). For instance, in ecology it has been demonstrated that diversity measures are biased in cases of small samples (e.g. in a very limited spatial area, limited amount of soil, etc.). Accordingly, in datasets on biodiversity, it is difficult to sample rare species appropriately (Beck and Schwanghart, 2010). To solve this issue associated to this type of datasets, measures of bias correction, e.g. of Shannon Entropy, have been proposed (Chao and Shen, 2003). These measures, mainly used in the area of ecology and biodiversity, are not yet implemented in **diverse**. Furthermore, considering for example phylogenetic diversity or functional diversity, other advanced measures, such as the generalization of the Rao's quadratic entropy (Chao et al., 2014a), are not yet included. To address these current limitations it must be noted that **diverse** can also be used in combination with several specialized packages such as **entropart**, **vegan** or **spadeR**. For instance, **diverse** provides a function (`to_entropart()`) that allows the user to transform the datasets from the package **diverse** into values of abundance to be used in **entropart**. Here we present a simple example to compute the richness of the *metacommunity* (see the **entropart** manual for details (Marcon and Héault, 2015)) generated with our synthetic dataset (the variable `data_set` of a previous example).

```

library(entropart)
abundance <- to_entropart(data_set)

```

```
mc <- MetaCommunity(abundance)
Richness(mc$Ps); Shannon(mc$Ps)
```

It must also be noted that the ability to statistically test differences in diversity measures across time and across systems could provide important insights for researchers. The following example shows a strong linear correlation between the Shannon Entropy in two different time steps of Scidat (dataframes scidat for 2013, and scidat\_2 for 2003). Further statistical test functions need to be implemented in subsequent versions of **diverse**, in order to also allow for the testing of the differences across systems. **diverse** will continue to learn from other disciplines, with the aim of implementing and adapting statistical test functions to the particular needs of researchers exploring the diversity in complex socioeconomic systems.

```
d_1 <- diversity(scidat, type="e")
d_2 <- diversity(scidat_2, type="e")
cor.test(d_1[,1], d_2[,1])
```

Pearson's product-moment correlation

```
data: d_1[, 1] and d_2[, 1]
t = 3.7171, df = 8, p-value = 0.005896
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.3330683 0.9496172
sample estimates:
cor
0.795807
```

## Conclusions

This paper introduced the package **diverse** which allows users to compute some of the most common measures of diversity from different fields of science. In summary, measuring diversity has become an important topic in many disciplines which analyze complex systems. The R package **diverse** allows for a combination of common measures from several disciplines and recent approaches from interdisciplinary research.

It must be noted that **diverse** has limitations that we aim to address in subsequent versions of the package. Possible future improvements include methods for considering diversity at different levels of aggregation (in hierarchical classification schemes, like mammals and insects, or agricultural or industrial goods, natural or social sciences, and their respective subcategories). Moreover, further emphasis on the role of different similarity measures at different levels of aggregations, as well as analyzing estimation error biases in incomplete samples are important future research areas in the measurement of diversity in socioeconomic systems, where social sciences can significantly learn from ecology and biology. Finally **diverse** can also continue to learn from ecology, biology, and other disciplines about how to apply statistical tests on the differences of diversity measures across systems.

In general, **diverse** offers a toolkit to analyze and visualize the diversity of entities, categories, and complex systems that is useful in particular for social scientists and interdisciplinary social research, as well as beginners in ecology and natural sciences. The package **diverse** provides different data import and export options and allows for the calculation of the different data transformations and similarity matrices, diversity measures, and diversity visualization options.

In order to present the functions provided by the package, we took advantage of an interdisciplinary taxonomy of diversity that defines variety, balance and disparity as three dimensions of diversity (Stirling, 2007). This taxonomy favors the creation of interdisciplinary bridges and helps in understanding how each diversity measure captures different aspects of diversity.

## Acknowledgments

We would like to thank Ismael Rafols, Diego Chavarro, Daniele Rotolo, and Andy Stirling for the example codes and valuable comments on diversity measures. We are also grateful for the valuable comments made by two anonymous reviewers. MG and MM acknowledge the Program of Incentives to Scientific Initiation (PIIC) from DGIP at Universidad Técnica Federico Santa María. MG thanks internal project ING01-1516 from Universidad de Playa Ancha. DH acknowledges support from the Marie Curie International Outgoing Fellowship No. 328828 within the 7th European Community Framework Programme. MM and MG acknowledges support from project FONDECYT 11121435.

## Bibliography

- B. Balassa. Trade Liberalisation and “Revealed” Comparative Advantage. *The Manchester School*, 33(2): 99–123, 1965. ISSN 1467-9957. doi: 10.1111/j.1467-9957.1965.tb00050.x. URL <http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9957.1965.tb00050.x/abstract>. [p65]
- B. Balassa. Comparative advantage in manufactured goods: a reappraisal. *The Review of Economics and Statistics*, pages 315–319, 1986. [p65]
- J. Beck and W. Schwanghart. Comparing measures of species diversity from incomplete inventories: an update. *Methods in Ecology and Evolution*, 1(1):38–44, Mar. 2010. ISSN 2041-210X. doi: 10.1111/j.2041-210X.2009.00003.x. URL <http://onlinelibrary.wiley.com/doi/10.1111/j.2041-210X.2009.00003.x/abstract>. [p72, 73]
- W. H. Berger and F. L. Parker. Diversity of Planktonic Foraminifera in Deep-Sea Sediments. *Science*, 168(3937):1345–1347, Dec. 1970. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.168.3937.1345. URL <http://www.sciencemag.org/content/168/3937/1345>. [p66]
- P. M. Blau. *Inequality and heterogeneity: A primitive theory of social structure*, volume 7. Free Press New York, 1977. [p66]
- L. Ceriani and P. Verme. The origins of the Gini index: extracts from Variabilità e Mutabilità (1912) by Corrado Gini. *The Journal of Economic Inequality*, 10(3):421–443, June 2011. ISSN 1569-1721, 1573-8701. doi: 10.1007/s10888-011-9188-x. URL <http://link.springer.com/article/10.1007/s10888-011-9188-x>. [p68]
- A. Chao and T.-J. Shen. Nonparametric estimation of Shannon’s index of diversity when there are unseen species in sample. *Environmental and Ecological Statistics*, 10(4):429–443, Dec. 2003. ISSN 1352-8505, 1573-3009. doi: 10.1023/A:1026096204727. [p73]
- A. Chao, C.-H. Chiu, and L. Jost. Unifying Species Diversity, Phylogenetic Diversity, Functional Diversity, and Related Similarity and Differentiation Measures Through Hill Numbers. *Annual Review of Ecology, Evolution, and Systematics*, 45(1):297–324, 2014a. doi: 10.1146/annurev-ecolsys-120213-091540. URL <http://dx.doi.org/10.1146/annurev-ecolsys-120213-091540>. [p69, 73]
- A. Chao, N. J. Gotelli, T. Hsieh, E. L. Sander, K. Ma, R. K. Colwell, and A. M. Ellison. Rarefaction and extrapolation with Hill numbers: a framework for sampling and estimation in species diversity studies. *Ecological Monographs*, 84(1):45–67, 2014b. [p69]
- D. Chavarro, P. Tang, and I. Rafols. Interdisciplinarity and research on local issues: evidence from a developing country. *Research Evaluation*, 23(3):195–209, 2014. [p60]
- G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*:1695, 2006. URL <http://igraph.org>. [p62]
- Z. Daróczy. Generalized information functions. *Information and control*, 16(1):36–51, 1970. [p69]
- V. Debastiani and V. Pillar. SYNCSA - R tool for analysis of metacommunities based on functional traits and phylogeny of the community components. *Bioinformatics*, 28:2067–2068, 2012. [p60]
- N. Eagle, M. Macy, and R. Claxton. Network Diversity and Economic Development. *Science*, 328 (5981):1029–1031, May 2010. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1186605. URL <http://science.sciencemag.org/content/328/5981/1029>. [p60]
- J. Farchy and H. Ranaivoson. Measuring the Diversity of Cultural Expressions: Applying the Stirling Model of Diversity in Culture. Technical Report 6, Unesco Institute for Statistics, 2011. [p60]
- K. Frenken, F. V. Oort, and T. Verburg. Related variety, unrelated variety and regional economic growth. *Regional Studies*, 41(5):685–697, July 2007. ISSN 0034-3404. doi: 10.1080/00343400601120296. URL <http://dx.doi.org/10.1080/00343400601120296>. [p60]
- C. Gini. *Variabilità e Mutuabilità. Contributo allo Studio delle Distribuzioni e delle Relazioni Statistiche*. Bologna, c. cuppini edition, 1912. [p66]
- M. R. Guevara, D. Hartmann, M. Aristarán, M. Mendoza, and C. A. Hidalgo. The research space: using career paths to predict the evolution of the research output of individuals, institutions, and nations. *Scientometrics*, pages 1–15, Sept. 2016. ISSN 0138-9130, 1588-2861. doi: 10.1007/s11192-016-2125-9. URL <http://link.springer.com/article/10.1007/s11192-016-2125-9>. [p60, 70]

- D. Hartmann, M. R. Guevara, C. Jara-Figueroa, M. Aristarán, and C. A. Hidalgo. Linking Economic Complexity, Institutions and Income Inequality. *arXiv:1505.07907 [physics, q-fin]*, 2016. URL <http://arxiv.org/abs/1505.07907>. arXiv: 1505.07907. [p70]
- D. M. A. Haughton and S. Mukerjee. The economic measurement and determinants of diversity. *Social Indicators Research*, 36(3):201–225, Nov. 1995. ISSN 0303-8300, 1573-0921. doi: 10.1007/BF01078814. URL <http://link.springer.com/article/10.1007/BF01078814>. [p60]
- R. Hausmann, C. A. Hidalgo, S. Bustos, M. Coscia, S. C. (M.A.), J. Jimenez, A. Simões, and M. A. Yıldırım. *The Atlas of Economic Complexity: Mapping Paths to Prosperity*. Center for International Development, Harvard University, 2011. ISBN 978-0-615-54662-9. [p62]
- J. Havrda and F. Charvát. Quantification method of classification processes. concept of structural  $\alpha$ -entropy. *Kybernetika*, 03(1):(30)–35, 1967. URL <http://eudml.org/doc/28681>. [p66, 69]
- C. Hidalgo. *Why Information Grows: The Evolution of Order, from Atoms to Economies*. Basic Books, June 2015. ISBN 978-0-465-04899-1. [p69]
- C. A. Hidalgo and R. Hausmann. The building blocks of economic complexity. *Proceedings of the National Academy of Sciences*, 106(26):10570–10575, June 2009. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.0900943106. URL <http://www.pnas.org/content/106/26/10570>. [p60]
- C. A. Hidalgo, B. Klinger, A.-L. Barabási, and R. Hausmann. The product space conditions the development of nations. *Science*, 317(5837):482–487, July 2007. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1144581. URL <http://www.sciencemag.org/content/317/5837/482>. [p60, 62, 67, 70]
- M. O. Hill. Diversity and Evenness: A Unifying Notation and Its Consequences. *Ecology*, 54(2):427–432, Mar. 1973. ISSN 0012-9658. doi: 10.2307/1934352. URL <http://www.esajournals.org/doi/abs/10.2307/1934352>. [p66, 69]
- C. J. C. J. Humphries, P. L. Forey, R. I. Vane-Wright, and S. Association. Systematics and conservation evaluation, 1994. URL [http://bvbr.bib-bvb.de:8991/F?func=service&doc\\_library=BVB01&local\\_base=BVB01&doc\\_number=006426531&line\\_number=0001&func\\_code=DB\\_RECORDS&service\\_type=MEDIA](http://bvbr.bib-bvb.de:8991/F?func=service&doc_library=BVB01&local_base=BVB01&doc_number=006426531&line_number=0001&func_code=DB_RECORDS&service_type=MEDIA). Published for the Systematics Association. [p60]
- T. Jombart, M. Kendall, J. Almagro-Garcia, and C. Colijn. *treescape: Statistical Exploration of Landscapes of Phylogenetic Trees*. 2016. URL <http://CRAN.R-project.org/package=treescape>. R package version 1.8.16. [p60]
- L. Jost. Entropy and diversity. *Oikos*, 113(2):363–375, May 2006. ISSN 1600-0706. doi: 10.1111/j.2006.0030-1299.14714.x. URL <http://onlinelibrary.wiley.com/doi/10.1111/j.2006.0030-1299.14714.x/abstract>. [p69]
- L. Jost. The Relation between Evenness and Diversity. *Diversity*, 2(2):207–232, Feb. 2010. doi: 10.3390/d2020207. URL <http://www.mdpi.com/1424-2818/2/2/207>. [p68]
- K. Keenan, P. McGinnity, T. F. Cross, W. W. Crozier, and P. A. Prodöhl. diveRsiTy: An R package for the estimation of population genetics parameters and their associated errors. *Methods in Ecology and Evolution*, 4(8):782–788, 2013. doi: 10.1111/2041-210X.12067. URL <http://onlinelibrary.wiley.com/doi/10.1111/2041-210X.12067/abstract>. R package version 1.9.89. [p60]
- R. Kindt and R. Coe. *Tree diversity analysis. A manual and software for common statistical methods for ecological and biodiversity studies*. World Agroforestry Centre (ICRAF), Nairobi (Kenya), 2005. URL [http://www.worldagroforestry.org/treesandmarkets/tree\\_diversity\\_analysis.asp](http://www.worldagroforestry.org/treesandmarkets/tree_diversity_analysis.asp). ISBN 92-9059-179-X. [p60, 61]
- R. Kolde. *pheatmap: Pretty Heatmaps*. 2015. URL <http://CRAN.R-project.org/package=pheatmap>. R package version 1.0.7. [p61]
- L. I. Kuncheva and C. J. Whitaker. Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. *Machine Learning*, 51(2):181–207, May 2003. ISSN 0885-6125, 1573-0565. doi: 10.1023/A:1022859003006. [p60]
- E. Laliberté and P. Legendre. A distance-based framework for measuring functional diversity from multiple traits. *Ecology*, 91(1):299–305, 2010. ISSN 0012-9658. doi: 10.1890/08-2244.1. URL <http://www.esajournals.org.ezproxy.canterbury.ac.nz/doi/abs/10.1890/08-2244.1>. [p60]
- Macro Connections MIT MediaLab. Pantheon - Mapping historical cultural production, 2014. URL <http://pantheon.media.mit.edu/methods>. [p61]

- E. Marcon and B. Héault. *entropart*: An R Package to Measure and Partition Diversity. *Journal of Statistical Software*, 67(8):1–26, 2015. doi: 10.18637/jss.v067.i08. [p60, 61, 73]
- Z. Marion, J. Fordyce, and B. Fitzpatrick. *hierDiversity: Hierarchical Multiplicative Partitioning of Complex Phenotypes*. 2015. URL <http://CRAN.R-project.org/package=hierDiversity>. R package version 0.1. [p60]
- D. G. McDonald and J. Dimmick. The Conceptualization and Measurement of Diversity. *Communication Research*, 30(1):60–79, Jan. 2003. ISSN 0093-6502, 1552-3810. doi: 10.1177/0093650202239026. URL <http://crx.sagepub.com/content/30/1/60>. [p62]
- D. Meyer and C. Buchta. *proxy: Distance and Similarity Measures*. 2015. URL <http://CRAN.R-project.org/package=proxy>. R package version 0.4-15. [p61]
- Nature. Diversity challenge. *Nature*, 513(7518):279–279, Sept. 2014. ISSN 0028-0836, 1476-4687. doi: 10.1038/513279a. URL <http://www.nature.com/doifinder/10.1038/513279a>. [p60]
- S. V. Nederland. Sovon Home Page, 2015. URL <https://www.sovon.nl/en>. [p61]
- J. Oksanen, F. G. Blanchet, R. Kindt, P. Legendre, P. R. Minchin, R. B. O'Hara, G. L. Simpson, P. Solymos, M. H. H. Stevens, and H. Wagner. *vegan: Community Ecology Package*. 2016. URL <http://CRAN.R-project.org/package=vegan>. R package version 2.3-5. [p60]
- E. C. Pielou. *Introduction to Mathematical Ecology*. John Wiley & Sons Inc, New York, Jan. 1970. ISBN 978-0-471-68918-8. [p66, 68]
- M. Pietrzak, M. Seweryn, and G. Rempala. *dovo: Tools for Analysis of Diversity and Similarity in Biological Systems*. 2016. URL <http://CRAN.R-project.org/package=dovo>. R package version 0.1.2. [p60]
- R Core Team. *foreign: Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ...* 2015. URL <http://CRAN.R-project.org/package=foreign>. R package version 0.8-65. [p64]
- I. Rafols. Knowledge Integration and Diffusion: Measures and Mapping of Diversity and Coherence. In Y. Ding, R. Rousseau, and D. Wolfram, editors, *Measuring Scholarly Impact*, pages 169–190. Springer International Publishing, 2014. ISBN 978-3-319-10376-1 978-3-319-10377-8. URL [http://link.springer.com/chapter/10.1007/978-3-319-10377-8\\_8](http://link.springer.com/chapter/10.1007/978-3-319-10377-8_8). [p60, 71]
- I. Rafols and M. Meyer. Diversity and network coherence as indicators of interdisciplinarity: case studies in bionanoscience. *Scientometrics*, 82(2):263–287, June 2009. ISSN 0138-9130, 1588-2861. doi: 10.1007/s11192-009-0041-y. URL <http://link.springer.com/article/10.1007/s11192-009-0041-y>. [p70]
- I. Rafols, A. L. Porter, and L. Leydesdorff. Science overlay maps: A new tool for research policy and library management. *Journal of the American Society for Information Science & Technology*, 61(9):1871–1887, Sept. 2010. ISSN 15322882. doi: 10.1002/asi.21368. URL <http://libproxy.mit.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=53286068&site=eds-live>. [p60, 62, 63, 70]
- C. R. Rao. Diversity and dissimilarity coefficients: A unified approach. *Theoretical Population Biology*, 21(1):24–43, Feb. 1982. ISSN 0040-5809. doi: 10.1016/0040-5809(82)90004-1. URL <http://www.sciencedirect.com/science/article/pii/0040580982900041>. [p62, 66]
- S. A. Rhoades. Herfindahl-Hirschman Index, The. *Federal Reserve Bulletin*, 79:188, 1993. URL <http://heinonline.org/HOL/Page?handle=hein.journals/fedred79&id=376&div=&collection=>. [p66]
- A. Rényi. On Measures of Entropy and Information. The Regents of the University of California, 1961. URL <http://projecteuclid.org/euclid.bsmsp/1200512181>. [p66]
- R. Scherer and P. Pallmann. *simboot: Simultaneous inference for diversity indices*. 2014. URL <http://CRAN.R-project.org/package=simboot>. R package version 0.2-5. [p60]
- SCImago. Scimago Journal & Country Rank, 2007. URL <http://www.scimagojr.com/>. [p61]
- C. E. Shannon. A mathematical theory of communication. *The Bell System technical Journal*, 27: 379–423, 623–656, Oct. 1948. [p66, 69]
- E. H. Simpson. Measurement of diversity. *Nature*, 163, 1949. [p66]
- A. Stirling. On the economics and analysis of diversity. *Science Policy Research Unit (SPRU), Electronic Working Papers Series, Paper*, 28:1–156, 1998. [p62, 71]

- A. Stirling. A general framework for analysing diversity in science, technology and society. *Interface The Journal of Royal Society*, 4(15):707–719, Aug. 2007. [p61, 62, 66, 70, 71, 74]
- M. Tennekes. *treemap: Treemap Visualization*. 2016. URL <http://CRAN.R-project.org/package=treemap>. R package version 2.4-1. [p62]
- C. Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *Journal of Statistical Physics*, 52(1-2):479–487, 1988. [p66, 69]
- H. Tuomisto. An updated consumer's guide to evenness and related indices. *Oikos*, 121(8):1203–1218, Aug. 2012. ISSN 1600-0706. doi: 10.1111/j.1600-0706.2011.19897.x. URL <http://onlinelibrary.wiley.com/doi/10.1111/j.1600-0706.2011.19897.x/abstract>. [p68]
- C. S. Wagner, J. D. Roessner, K. Bobb, J. T. Klein, K. W. Boyack, J. Keyton, I. Rafols, and K. Börner. Approaches to understanding and measuring interdisciplinary scientific research (IDR): A review of the literature. *Journal of Informetrics*, 5(1):14–26, Jan. 2011. ISSN 1751-1577. doi: 10.1016/j.joi.2010.06.004. URL <http://www.sciencedirect.com/science/article/pii/S1751157710000581>. [p60]
- C. Wang, M. Genkin, G. Berry, L. Chen, and M. Brashearswork. *Blaunet: Calculate and Analyze Blau Status for Measuring Social Distance*. 2016. URL <http://CRAN.R-project.org/package=Blaunet>. R package version 2.0.4. [p60]
- J. Wang, B. Thijs, and W. Gläzel. Interdisciplinarity and Impact: Distinct Effects of Variety, Balance, and Disparity. *PLoS ONE*, 10(5):e0127298, May 2015. doi: 10.1371/journal.pone.0127298. URL <http://dx.doi.org/10.1371/journal.pone.0127298>. [p71]
- A. Z. Yu, S. Ronen, K. Hu, T. Lu, and C. A. Hidalgo. Pantheon 1.0, a manually verified dataset of globally famous biographies. *Scientific data*, 3, 2016. [p61]

*Miguel R. Guevara*

*Computer Science Department, Universidad de Playa Ancha, and  
Department of Informatics, Universidad Técnica Federico Santa María  
Valparaíso  
Chile*

[miguel.guevara@upla.cl](mailto:miguel.guevara@upla.cl)

*Dominik Hartmann*

*Chair of Innovation Management and Innovation Economics, University of Leipzig  
Grimmaische Straße 12, 04109, Leipzig  
Fraunhofer Center for International Management and Knowledge Economy  
Neumarkt 9-19, 04109, Leipzig  
Germany*

[dominik.hartmann@uni-leipzig.de](mailto:dominik.hartmann@uni-leipzig.de)

*Marcelo Mendoza*

*Department of Informatics, Universidad Técnica Federico Santa María  
Av. Vicuña Mackenna 3939, San Joaquín, Santiago  
Chile*

[mmendoza@inf.utfsm.cl](mailto:mmendoza@inf.utfsm.cl)

# Simulating Correlated Binary and Multinomial Responses under Marginal Model Specification: The SimCorMultRes Package

by Anestis Touloumis

**Abstract** We developed the R package **SimCorMultRes** to facilitate simulation of correlated categorical (binary and multinomial) responses under a desired marginal model specification. The simulated correlated categorical responses are obtained by applying threshold approaches to correlated continuous responses of underlying regression models and the dependence structure is parametrized in terms of the correlation matrix of the latent continuous responses. This article provides an elaborate introduction to the **SimCorMultRes** package demonstrating its design and usage via three examples. The package can be obtained via CRAN.

## Introduction

Fitting marginal models with correlated binary or multinomial responses is required in many applications in which the responses are assumed to be correlated. The obvious instance of such studies is longitudinal studies (Diggle et al., 2002) where the categorical responses for each subject are collected across time points and form a cluster. For each cluster, the associated covariates are also recorded as they might influence the true marginal probabilities. Ordinary regression models designed for independent responses might not lead to consistent estimators of the marginal regression parameters or of their standard errors. For this reason, many authors have developed and proposed procedures for estimating the regression parameters of a marginal model with categorical responses that are robust to misspecification of the dependence structure, including maximum likelihood methods (Fitzmaurice and Laird, 1993; Glonek and McCullagh, 1995), copula approaches (Masarotto and Varin, 2012), quasi-least squares approaches (Shults and Chaganty, 1998), generalized quasi-likelihood methods (Sutradhar and Das, 1999; Sutradhar, 2003) and generalized estimating equations (GEE) approaches (Lipsitz et al., 1991; Chaganty and Joe, 2004; Touloumis et al., 2013). Although the asymptotic properties of these methods are well-established, the evaluation of their performance in finite samples under misspecification of the correlation structure relies on simulations. The crucial step of these empirical studies is to simulate correlated categorical responses that satisfy a desired marginal model and dependence structure specification.

Motivated by this, we present the R package **SimCorMultRes** (Touloumis, 2016) which makes it easy to simulate correlated categorical responses under a given marginal model and dependence structure configuration. The package implements marginal models for correlated binary responses (two response categories) as well as for correlated multinomial responses (three or more response categories) while taking into account the nature of the response categories (ordinal or nominal). In summary, the correlated binary/multinomial responses are obtained as realizations of an underlying continuum. This means that latent regression models with correlated continuous responses are utilized so as to generate the correlated categorical responses that satisfy the desired marginal model specification. The categorical responses are obtained by applying threshold approaches to the correlated continuous responses. In order to avoid theoretical pitfalls outlined in the next paragraph, the desired dependence structure is expressed in terms of the correlation matrix of the latent responses. To the best of our knowledge, **SimCorMultRes** is the first package in R that allows direct simulation of correlated categorical responses under a marginal model specification with categorical and/or continuous covariates.

To fully appreciate the features of **SimCorMultRes**, we briefly compare it with two R packages: i) **GenOrd** (Barbiero and Ferrari, 2015), that implements the methods presented by Ferrari and Barbiero (2012) and its features being discussed in greater detail in Barbiero and Ferrari (in press), and ii) **MultiOrd** (Amatya and Demirtas, 2016), that is described in Amatya and Demirtas (2015) and relies on the simulation techniques proposed by Demirtas (2006). These packages are designed to simulate random vectors of correlated binary or ordinal responses subject to fixed but common marginal probabilities across all subjects and a predefined correlation matrix for the correlated categorical responses. Therefore, unlike **SimCorMultRes**, it is not straightforward to utilize **GenOrd** or **MultiOrd** for simulating categorical responses conditional on a regression model specification for the marginal probabilities, especially when the marginal probabilities vary across subjects. In addition, **SimCorMultRes** has the

unique feature (to the best of our knowledge) to simulate correlated nominal responses. Another difference between **SimCorMultRes** and the R packages **GenOrd** and **MultiOrd** is that the former requires the association among the categorical responses to be directly expressed via their correlation matrix and that the joint specification of the marginal probabilities and of the correlation matrix leads to a valid joint distribution for the correlated categorical responses. A necessary condition for this is that the so-called Fréchet-Hoeffding bounds are satisfied, which can be verified by employing the method of Demirtas and Hedeker (2011). As noted by one of the reviewers, both **GenOrd** and **MultiOrd** have built-in mechanisms to check the restrictions imposed by the Fréchet-Hoeffding bounds. Unfortunately, even if these restrictions are met, it is still theoretically possible that a legitimate joint distribution does not exist for the correlated categorical responses (Bergsma and Rudas, 2002). To circumvent this difficulty, the methodology implemented in **SimCorMultRes** always defines the joint distribution of the correlated categorical responses in terms of the joint distribution of correlated latent random variables and thus, it allows the user to generate correlated categorical responses under any configuration of the marginal probabilities provided that the user-defined correlation matrix of the latent continuous responses is positive definite, a condition that can be more easily verified.

The remainder of this paper is organized as follows. First we present the theoretical background of the threshold approaches implemented in **SimCorMultRes**. In particular, we introduce the general two-stage algorithm for simulating correlated categorical responses, focusing on the threshold approaches that give rise to the marginal models with correlated categorical responses and on the modified version of the NORmal To Anything (NORTA) method (Cario and Nelson, 1997), the default simulation method of correlated latent random variables in **SimCorMultRes**. Next, we describe the core and utility functions of the package. Then, we demonstrate the use of **SimCorMultRes** by considering the problems of evaluating two estimation methods for marginal models with correlated nominal multinomial responses, of assessing the quality of an approximation that links the uniform local odds ratios structure with the correlation parameter of an underlying bivariate normal distribution, and of simulating correlated categorical random variables under no marginal model specification. Finally, we summarize the features of **SimCorMultRes** and discuss future extensions.

## Theoretical background

In this section, we introduce the threshold approaches that give rise to marginal models with correlated binary, ordinal or nominal responses. Since the thresholds are applied to correlated continuous responses, simulation of correlated continuous responses is required. This step can be performed in various ways, eg, directly from an appropriate multivariate distribution, by utilizing distributional properties about the sum or the difference of random vectors or by employing copula approaches. Herein we discuss a simple and straightforward simulation method that is based on the NORTA method, and we present a general algorithm that combines the threshold approaches with the modified NORTA method, enabling us to generate correlated categorical responses subject to a marginal model specification in a unified manner. However, we underline that the use of the NORTA method is optional in the general algorithm and that it can be replaced with another simulation method/technique as long as the distributional restrictions regarding the correlated continuous variables that are imposed by the thresholds are met.

For notational ease, adopt a longitudinal set-up for generating the correlated binary or multinomial variables. Let  $Y_{it}$  be the random variable of subject  $i$  ( $i = 1, \dots, N$ ) at time  $t$  ( $t = 1, \dots, T$ ) and let  $\mathbf{x}_{it}$  denote the associated covariates vector. To be consistent with the notation in the majority of the literature, let  $Y_{it} \in \{0, 1\}$  when there are two response categories and let  $Y_{it} \in \{1, 2, \dots, J \geq 3\}$  for at least three categories.

### Binary responses

Suppose the aim is to simulate correlated binary variables such that the marginal probabilities satisfy the model

$$\Pr(Y_{it} = 1 | \mathbf{x}_{it}) = F(\beta_{t0} + \boldsymbol{\beta}_t' \mathbf{x}_{it}) \quad (1)$$

where  $\beta_{t0}$  is the intercept and  $\boldsymbol{\beta}_t$  is the covariates parameter vector at time  $t$ , respectively, and where  $F$  is a cumulative distribution function (c.d.f.).

Now, consider the multivariate latent regression model

$$\mathbf{U}_i^B = \begin{pmatrix} U_{i1}^B \\ \vdots \\ U_{iT}^B \end{pmatrix} = \begin{pmatrix} \mu_{i1}^B \\ \vdots \\ \mu_{iT}^B \end{pmatrix} + \begin{pmatrix} e_{i1}^B \\ \vdots \\ e_{iT}^B \end{pmatrix} = \boldsymbol{\mu}_i^B + \mathbf{e}_i^B$$

where  $\mu_{it}^B = \beta_t' \mathbf{x}_{it}$  and  $\{\mathbf{e}_i^B : i = 1, \dots, N\}$  are independent random vectors such that  $e_{it}^B \sim F$  for all  $i$  and  $t$ . Under these assumptions, generation of binary responses under the threshold

$$Y_{it} = I(e_{it}^B \leq \beta_{t0} + \mu_{it}^B) = I(U_{it}^B \leq \beta_{t0} + 2\mu_{it}^B)$$

gives rise to the marginal model (1), where  $I(A)$  denotes the indicator function of the event  $A$ . This approach is a straightforward extension of the gold-standard simulation method proposed by Emrich and Piedmonte (1991), in the sense that it also permits marginal modeling of the univariate probabilities through covariates. Implementation of the method of Emrich and Piedmonte (1991) can be found in the orphaned R package `mvtBinaryEP` (By and Qaqish, 2011).

### Ordinal responses

Options for marginal modelling of correlated ordinal responses include the marginal cumulative link model

$$\Pr(Y_{it} \leq j | \mathbf{x}_{it}) = F(\beta_{tj0} + \beta_t' \mathbf{x}_{it}) \quad (2)$$

and the marginal continuation-ratio model

$$\Pr(Y_{it} = j | Y_{it} \geq j, \mathbf{x}_{it}) = F(\beta_{tj0} + \beta_t' \mathbf{x}_{it}). \quad (3)$$

In both models,  $F$  is a c.d.f. and  $\beta_t$  is the parameter vector at time  $t$  when the corresponding  $(J - 1)$  category-specific intercepts  $(\beta_{t10}, \beta_{t20}, \dots, \beta_{t(J-1)0})$  are excluded.

First, consider the marginal cumulative link model (2) and suppose the multivariate latent regression model

$$\mathbf{U}_i^{O1} = \begin{pmatrix} U_{i1}^{O1} \\ \vdots \\ U_{iT}^{O1} \end{pmatrix} = \begin{pmatrix} \mu_{i1}^{O1} \\ \vdots \\ \mu_{iT}^{O1} \end{pmatrix} + \begin{pmatrix} e_{i1}^{O1} \\ \vdots \\ e_{iT}^{O1} \end{pmatrix} = \boldsymbol{\mu}_i^{O1} + \mathbf{e}_i^{O1}$$

holds, where  $\mu_{it}^{O1} = -\beta_t' \mathbf{x}_{it}$ , and  $\{\mathbf{e}_i^{O1} : i = 1, \dots, N\}$  are independent random vectors such that  $e_{it}^{O1} \sim F$  for all  $i$  and  $t$ . To generate an ordinal response  $Y_{it}$  that satisfies model (2), one can categorize  $U_{it}^B$  by using the corresponding category-specific intercepts according to the threshold

$$Y_{it} = j \Leftrightarrow \beta_{t(j-1)0} < U_{it}^{O1} \leq \beta_{tj0}$$

where

$$-\infty = \beta_{t00} < \beta_{t10} < \beta_{t20} < \dots < \beta_{t(J-1)0} < \beta_{tJ0} = \infty.$$

This threshold approach extends the approach discussed in McCullagh (1980) from cumulative link models with independent ordinal responses to marginal cumulative link models with correlated ordinal responses.

Next, consider the marginal continuation-ratio model (3) and suppose the following multivariate latent regression model holds

$$\mathbf{U}_i^{O2} = \begin{pmatrix} \mathbf{U}_{i1}^{O2} \\ \vdots \\ \mathbf{U}_{iT}^{O2} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\mu}_{i1}^{O2} \\ \vdots \\ \boldsymbol{\mu}_{iT}^{O2} \end{pmatrix} + \begin{pmatrix} \mathbf{e}_{i1}^{O2} \\ \vdots \\ \mathbf{e}_{iT}^{O2} \end{pmatrix} = \boldsymbol{\mu}_i^{O2} + \mathbf{e}_i^{O2}$$

where  $\mathbf{U}_{it}^{O2} = (U_{it1}^{O2}, \dots, U_{itJ}^{O2})'$ ,  $\boldsymbol{\mu}_{it}^{O2} = -(\beta_t' \mathbf{x}_{it}, \dots, \beta_t' \mathbf{x}_{it})'$  and  $\mathbf{e}_{it}^{O2} = (e_{it1}^{O2}, \dots, e_{itJ}^{O2})'$  for all  $i$  and  $t$ , and  $\{\mathbf{e}_i^{O2} : i = 1, \dots, N\}$  are independent random vectors such that:

1.  $e_{itj}^{O2} \sim F$  for all  $i, t$  and  $j$ ,
2.  $e_{itj}^{O2}$  and  $e_{itj'}^{O2}$  are independent for all  $j \neq j'$  (local independence assumption).

The marginal continuation-ratio model (3) arises by applying the threshold

$$Y_{it} = j, \text{ given } Y_{it} \geq j \Leftrightarrow U_{itj}^{O2} \leq \beta_{tj0}$$

to the components of  $\mathbf{U}_{it}$ 's in a sequential order. This approach extends the latent variable representation described in Tutz (1991) which gives rise to the continuation-ratio model for independent ordinal responses (see Agresti, 2013).

## Nominal responses

Consider the marginal baseline-category logit model

$$\log \left[ \frac{\Pr(Y_{it} = j | \mathbf{x}_{it})}{\Pr(Y_{it} = J | \mathbf{x}_{it})} \right] = (\beta_{tj0} - \beta_{tJ0}) + (\beta_{tj} - \beta_{tJ})' \mathbf{x}_{it} = \beta_{tj0}^* + \beta_{tj}' \mathbf{x}_{it} \quad (4)$$

where  $\beta_{tj0}$  and  $\beta_{tj}$  is the  $j$ -th category-specific intercept and parameter vector at time  $t$ , respectively. For identifiability reasons, restrictions such as  $\beta_{tJ0} = 0$  and  $\beta_{tJ} = \mathbf{0}$  for all  $t$  are required, which imply that  $\beta_{tj0}^* = \beta_{tj0}$  and  $\beta_{tj}^* = \beta_{tj}$  for all  $t$  and for all  $j = 1, \dots, J - 1$ . Note that model (4) relates with the baseline-category logit model (see Agresti, 2013) and hence it is appropriate for marginal modelling of correlated nominal responses.

To connect the marginal baseline-category logit model (4) with underlying regression models, consider the multivariate latent regression model

$$\mathbf{U}_i^{NO} = \begin{pmatrix} \mathbf{U}_{i1}^{NO} \\ \vdots \\ \mathbf{U}_{iT}^{NO} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\mu}_{i1}^{NO} \\ \vdots \\ \boldsymbol{\mu}_{iT}^{NO} \end{pmatrix} + \begin{pmatrix} \mathbf{e}_{i1}^{NO} \\ \vdots \\ \mathbf{e}_{iT}^{NO} \end{pmatrix} = \boldsymbol{\mu}_i^{NO} + \mathbf{e}_i^{NO}$$

where  $\mathbf{U}_{it}^{NO} = (U_{it1}^{NO}, \dots, U_{itJ}^{NO})'$ ,  $\boldsymbol{\mu}_i^{NO} = (\beta_{t10} + \beta_{t1}' \mathbf{x}_{it}, \dots, \beta_{t(J-1)0} + \beta_{tJ}' \mathbf{x}_{it})'$  and  $\mathbf{e}_i^{NO} = (e_{i1}^{NO}, \dots, e_{itJ}^{NO})'$  for all  $i$  and  $t$ , and  $\{\mathbf{e}_i^{NO} : i = 1, \dots, N\}$  are independent random vectors such that:

1.  $e_{itj}^{NO}$  follow a standard extreme distribution for all  $i, t$  and  $j$ ,
2. the assumption of choice independence is met at each measurement occasion, that is  $e_{itj}^{NO}$  and  $e_{itj'}^{NO}$  are independent for all  $j \neq j'$ .

The threshold

$$Y_{it} = j \Leftrightarrow U_{itj} = \max\{U_{it1}, \dots, U_{itJ}\}$$

extends the principle of maximum random utility (McFadden, 1974) and it generates correlated nominal responses that give rise to the marginal baseline-category logit model (4).

## Simple version of the NORTA method

Li and Hammond (1975) proposed a simple method for generating continuous random vectors with given marginal distributions and a prescribed correlation matrix. Cario and Nelson (1997) introduced the NORTA method which essentially modifies the approach of Li and Hammond (1975) to account for any type of marginal distributions (discrete, continuous or mixed). Here, we describe a simple version of the NORTA method in which the desired marginal distributions are continuous and identical which is required by all the threshold approaches implemented in **SimCorMultRes**.

Let  $F$  be the c.d.f. of the target marginal distribution. To generate a  $p$ -variate random vector  $\mathbf{W} = (W_1, \dots, W_p)'$  with correlation matrix  $\text{cor}(\mathbf{W}) = \mathbf{R}_W$  such that  $W_k \sim F$  for all  $k = 1, \dots, p$ , the following NORTA transformation can be utilized:

1. Generate a random vector  $\mathbf{Z} = (Z_1, \dots, Z_p)'$  from a standard multivariate normal distribution with correlation matrix  $\text{cor}(\mathbf{Z}) = \mathbf{R}_Z$ . The elements of  $\mathbf{R}_Z$  are calculated by solving numerically  $p(p-1)/2$  equations, such that each equation relates  $\text{cor}(Z_k, Z_{k'})$  with  $\text{cor}(W_k, W_{k'})$  for all  $k < k'$ . The exact formulae are given by Li and Hammond (1975).
2. Apply the transformation  $W_k = F^{-1}[\Phi(Z_k)]$  for all  $k$ , where  $\Phi$  is the cumulative distribution of the standard normal distribution.

If  $F = \Phi$ , then the second step of the above modified NORTA algorithm is not needed. Otherwise, the correlation matrices  $\mathbf{R}_Z$  and  $\mathbf{R}_W$  are expected to differ. In fact, Cario and Nelson (1997) showed that under mild conditions it is possible to have  $\mathbf{R}_Z \approx \mathbf{R}_W$ . For example, if  $F$  is the cumulative distribution function of the standard logistic distribution (which might be the case in the marginal models for correlated binary and ordinal responses), then  $\mathbf{R}_Z \approx \mathbf{R}_W$  due to the well-known approximation  $\Phi(x) = F(x\pi/3)$  for all  $x \in \mathbb{R}$ . This simplifies the computational task as the  $p(p-1)/2$  equations are not needed to be solved and issues regarding non-existence of a valid correlation matrix  $\mathbf{R}_Z$  for a given choice of the correlation matrix  $\mathbf{R}_W$  (Li and Hammond, 1975) are avoided provided that  $\mathbf{R}_Z$  is positive-definite under mild conditions (Cario and Nelson, 1997).

## General generative process

We propose a simple and efficient two-staged general algorithm for generating correlated categorical responses:

Stage 1. *Marginal model specification*: Provide the covariates, the regression parameters and the link function (if required) of the desired marginal model (1), (2), (3), or (4).

Stage 2. *Simulation of continuous random vectors via the NORTA method and threshold approach*: Define the desired dependence structure by fixing  $\mathbf{R}_Z$  to generate the continuous random vectors  $\mathbf{U}_i$ 's from the multivariate latent regression model implied by the marginal model specification selected in Stage 1. Apply the corresponding threshold approach to obtain the correlated binary, ordinal or nominal responses.

In the marginal models described above, we usually choose  $F$  to be the c.d.f. of a standard normal, logistic or extreme value distribution. In either of these cases, it can be shown that the simulated categorical responses are independent if and only if  $\mathbf{R}_Z$  is the identity matrix, which is true if and only if the random variables in the latent random vectors  $\mathbf{e}_i$ 's are independent (Carlo and Nelson, 1997). For all other forms of  $\mathbf{R}_Z$ , correlated categorical responses will be generated.

Expressing the association structure in terms of  $\mathbf{R}_Z$  ensures the existence of a joint distribution for the correlated categorical responses regardless of the marginal model specification which is not the case when the association is expressed directly via the correlation matrix of the correlated categorical responses. This well-known fact has been mentioned by Bergsma and Rudas (2002) among others, and it has been exemplified in the case of correlated binary and multinomial responses by Chaganty and Joe (2004), Chaganty and Joe (2006) and Toulaoumis et al. (2013), respectively. The simplest scenario where adopting a common correlation matrix for the correlated categories responses across subjects is problematic is when the linear predictor in the marginal model is allowed to vary freely on the real line. In this case, only the identity matrix is a feasible value for the correlation matrix.

As mentioned before, the proposed version of the NORTA method is not the only option to simulate continuous random vectors in Stage 2 and instead, alternative simulation techniques can be easily employed. However, the user must be cautious in order to respect the corresponding marginal distributional assumptions and the assumption of local independence or choice independence whenever the marginal models (3) or (4) are used, respectively.

We emphasize that the proposed algorithm can also handle the situation in which no marginal model specification is provided. For more details, please refer to the third example below.

## Description of SimCorMultRes

**SimCorMultRes** contains four core functions (`rbin`, `rmult.bcl`, `rmult.clm` and `rmult.crm`) that enable the user to generate correlated categorical responses and two utility functions (`rnorta` and `rsmvnorm`) initially designed for internal use in the core functions. We describe in detail the arguments and the output of the core and utility functions.

### Core functions

Each core function in **SimCorMultRes** simulates correlated categorical responses under a marginal model specification. In particular, `rbin` simulates correlated binary responses that satisfy the marginal model (1), `rmult.clm` simulates correlated ordinal responses that satisfy the marginal cumulative link model (2), `rmult.crm` simulates correlated ordinal responses that satisfy the marginal continuation-ratio model (3) and `rmult.bcl` simulates correlated nominal responses that satisfy the marginal baseline-category logit model (4).

The common cluster size (`clsize`) of the subjects is required in all core functions.

The `ncategories` argument in `rmult.bcl` indicates the number of nominal response categories. The number of ordinal response categories in `rmult.clm` and `rmult.crm` is indirectly defined by the `intercepts` argument. It contains the values of the threshold parameters which can be provided either as a  $T \times (J - 1)$  matrix or as a vector of length  $J - 1$ . In the first case, the  $(t, j)$ -th element of `intercepts` corresponds to  $\beta_{tj0}$  and in the second case, it is assumed that  $\beta_{tj0} = \beta_{j0}$  for all  $t$  in the marginal models (2) or (3). The `intercepts` argument is also employed in `rbin` to specify whether the intercepts in the marginal model (1) are time-dependent. If  $\beta_{t0} = \beta_0$  for all  $t$ , then `intercepts` should be a single number that reflects the value of  $\beta_0$ . Otherwise, it should be a vector of size  $T$  with the  $t$ -th element equal to the value of  $\beta_{t0}$ .

The values for the marginal regression parameters (`betas`) should be provided as a numeric vector whenever  $\beta_t = \beta$  for all  $t$  in models (1), (2) or (3), and whenever  $\beta_{tj} = \beta_j$  and  $\beta_{tj} = \beta_j$  for all  $t$  in

model (4). In all other cases, betas should be provided as a matrix with  $T$  rows such that the  $t$ -th row contains the value of the marginal parameter vector at time  $t$ . It is important to emphasize that (category-specific) intercept values should not be included in betas unless the function `rmult.bcl` is used.

The functional relationship of the covariates in the marginal model (`xformula`) is specified similarly as in other regression models with the single difference that no response variable should be provided. The covariates defined in `xformula` can be imported via the `xdata` argument in “long” format, meaning that each row contains all the subject-specific covariates information at a given time. When `xdata` is missing, then the covariates are extracted from the environment that the core function is called.

The `link` argument in `rbin`, `rmult.clm` or `rmult.crm` determines the c.d.f.  $F$  in the marginal models (1), (2) or (3) respectively, i. e., the link function. Options for the link function include the probit (“probit”), the logit (“logit”), the complimentary log-log (“cloglog”) and the cauchit (“cauchit”). It is worth mentioning that there is no `link` argument in the function `rmult.bcl` because the marginal distribution of the latent continuous random variables  $e_{itj}^{NO}$ ’s is always the standard extreme value distribution.

In all core functions, the latent random vectors  $\mathbf{e}_i$ ’s can be either simulated using the proposed NORTA approach or provided by the user via the `rlatent` argument. In the first case, the correlation matrix  $\mathbf{R}_Z$  of the multivariate normal distribution (`cor.matrix`) in the modified NORTA method and the `link` argument, wherever present, are required. Checks are carried out to ensure that `cor.matrix` is a positive-definite correlation matrix and whenever `rmult.crm` or `rmult.bcl` is employed, `cor.matrix` is forced to satisfy the restrictions of the latent dependence structure that are implied by the threshold approach associated with models (3) or (4), respectively. In the case where the preferred simulation method is not the NORTA method, `rlatent` should contain the values of the latent random vectors while `cor.matrix` and `link` are ignored. Examples of using the `rlatent` argument can be found in the help files and the vignette of **SimCorMultRes**.

The output of any core function is displayed as a list with three items: (i) a matrix with the simulated responses such that the  $(i, t)$ -th element corresponds to the realization of  $Y_{it}$  (`Ysim`), (ii) a data frame (`simdata`) that contains the simulated responses (`y`), the covariates specified by `xformula`, subjects’ identities (`id`) and the measurement occasions (`time`), and (iii) the NORTA generated or user-defined latent random vectors (`rlatent`).

## Utility functions

The utility function `rnorta` offers a more general implementation of the NORTA method described earlier. The user needs to specify the number of random vectors (`R`), the correlation matrix  $\mathbf{R}_Z$  of the multivariate normal distribution (`cor.matrix`) and the names of the quantile functions of the desired marginal distributions (`distr`). The optional `qparameters` argument permits users to consider parameter values for the marginal distributions other than the default (obtained when `qparameters = NULL`). The function returns `R` random vectors with marginal distributions specified by `distr` (and `qparameters`) when `cor.matrix` is the correlation matrix of the multivariate normal distribution in the NORTA method. We highlight that `rnorta` has been extended to handle situations that are beyond the scope of simulation of correlated categorical responses subject to a marginal model specification. Unlike the simple version of the NORTA method needed for our purposes, `rnorta` does not require marginal distributions to be identical. In fact, any univariate discrete or continuous distribution whose quantile function is available in R can be employed in `distr` provided that the required R package is available.

The function `rsmvnorm` generates `R` random vectors from a multivariate normal distribution with mean vector the zero vector and covariance matrix `cor.matrix`.

Note that an error message is returned whenever `cor.matrix` in functions `rnorta` or `rsmvnorm` is not a positive-definite correlation matrix.

## Empirical illustration

We now illustrate the use of **SimCorMultRes** to: i) evaluate the performance of GEE approaches for estimating the regression parameters of a marginal baseline-category logit model, ii) to verify approximations that relate a uniform local odds ratios structure to the correlation coefficient of a bivariate normal distribution (Goodman, 1979) and, iii) to simulate correlated categorical random variables with fixed arbitrary univariate probabilities that are not subject to a marginal model specification.

## Parameter estimation of marginal models

The motivation behind the creation of **SimCorMultRes** lies on evaluating statistical methods that estimate the regression coefficients of marginal models with correlated binary or multinomial responses. To exemplify this, we employ two GEE models for estimating a marginal model with correlated nominal responses: i) the local odds ratios GEE approach (Touloumis et al., 2013) and ii) the independence “working” model, which treats all observations as independent when solving the estimating equations. Although the two competing GEE models are asymptotically equally efficient, in the sense that they both produce consistent estimators for the marginal regression parameters and of their standard errors, the regression coefficient estimators of the independence “working” model are expected to be slightly less precise than those of the local odds ratios GEE approach in small and moderate sample sizes due to the fact that the independence “working” model does not account for the dependence among the correlated responses (Touloumis et al., 2013).

To investigate this assertion for the case of correlated nominal responses, we employed the marginal baseline-category logit model

$$\log \left[ \frac{\Pr(Y_{it} = j | \mathbf{x}_{it})}{\Pr(Y_{it} = 5 | \mathbf{x}_{it})} \right] = \beta_{j0} + \beta_{j1} x_{it} \quad (5)$$

where  $(\beta_{10}, \beta_{11}, \beta_{20}, \beta_{21}, \beta_{30}, \beta_{31}, \beta_{40}, \beta_{41}) = (2, 1, 1, 2, 1.5, 1.5, 2.5, 0.5)$  and  $x_{it} \stackrel{i.i.d.}{\sim} N(0, 1)$  for all  $i = 1, \dots, 100$  and  $t = 1, 2, 3, 4$ . Further, the correlation matrix  $\mathbf{R}_Z$  among the normally distributed variables  $Z_{itj}$ 's in the NORTA method was given by

$$\text{cor}(Z_{itj}, Z_{it'j'}) = \begin{cases} 1 & \text{if } t = t' \text{ and } j = j' \\ 0 & \text{if } t = t' \text{ and } j \neq j' \\ 0.56^{tj-t'j'} & \text{if otherwise.} \end{cases}$$

```
> library("SimCorMultRes")
> library("multgee")
Loading required package: gnm
Loading required package: VGAM
Loading required package: stats4
Loading required package: splines
> set.seed(1)
> N <- 100
> clsizes <- 4
> ncategories <- 5
> betas <- c(2, 1, 1, 2, 1.5, 1.5, 2.5, 0.5, 0, 0)
> x <- rnorm(N * clsizes)
> cor.matrix <- toeplitz(0.56^seq(0, clsizes * ncategories - 1))
> for (i in 1:clsizes) {
+   diag.index <- 1:ncategories + (i - 1) * ncategories
+   cor.matrix[diag.index, diag.index] <- diag(1, ncategories)
+ }
```

Conditional on the above marginal model specification and dependence structure, we simulated correlated nominal responses and we fitted the local odds ratios GEE approach with an RC-type dependence structure and the independence “working” model using the R package **multgee** (Touloumis, 2015). We replicated this procedure 1000 times and at each iteration we recorded the estimates of the marginal regression parameter vector of the two competing models:

```
> B <- 1000
> indeGEEcoefs <- matrix(NA_real_, B, 8)
> RCGEEcoefs <- matrix(NA_real_, B, 8)
> for (b in 1:B) {
+   SimNomRes <- rmult.bcl(clsizes = clsizes, ncategories = ncategories,
+                           betas = betas, xformula = ~x, cor.matrix = cor.matrix)
+   fitRC <- try(nomLORgee(y ~ x, id = id, repeated = time, data = SimNomRes$simdata,
+                           LORstr = "RC", add = 0.05), silent = TRUE)
+   if (!inherits(fitRC, "try-error")) {
+     if (fitRC$convergence$conv)
+       RCGEEcoefs[b, ] <- coef(fitRC)
+   }
+   fitinde <- try(nomLORgee(y ~ x, id = id, repeated = time, data = SimNomRes$simdata,
+                             LORstr = "independence"), silent = TRUE)
```

**Table 1:** Simulation results for the local odds GEE approach (LOR) and the independence “working” model (IEE) for estimating the regression parameter vector and their standard errors (second row) in the marginal model (5).

Model	$\beta_{10} = 2$	$\beta_{11} = 1$	$\beta_{20} = 1$	$\beta_{21} = 2$	$\beta_{30} = 1.5$	$\beta_{31} = 1.5$	$\beta_{40} = 2.5$	$\beta_{41} = 0.5$
IEE	2.0701	1.0308	1.0494	2.0530	1.5682	1.5441	2.5702	0.5209
	0.3567	0.3232	0.4011	0.3601	0.3740	0.3412	0.3627	0.2995
LOR	2.0471	0.9951	1.0378	1.9832	1.5487	1.4943	2.5473	0.4946
	0.3512	0.3105	0.3944	0.3490	0.3673	0.3319	0.3562	0.2873
SRE	1.0522	1.0927	1.0405	1.0852	1.0526	1.0738	1.0573	1.0921

```
+ if (!inherits(fitinde, "try-error")) {
+   if (fitinde$convergence$conv)
+     indeGEEcoefs[b, ] <- coef(fitinde)
+ }
+ }
```

Although the local odds GEE approach did not always converge, the convergence rate for the local odds ratios GEE model was high

```
> convergence <- c(mean(!is.na(indeGEEcoefs)), mean(!is.na(RCGEEcoefs))) * 100
> convergence
[1] 100.0 99.7
```

and therefore, we can conduct a fair comparison by excluding the results from those 3 iterations in which the local odds ratios GEE approach failed to converge.

Table 1 summarizes the simulation results by displaying the simulated mean and standard error of the regression estimates from the two competing GEE models and the simulated relative efficiency (SRE) for each regression parameter of model (5). For a given coefficient of model (5), the SRE criterion was defined as the ratio of the simulated mean square error of the corresponding Monte Carlo estimate based on the local odds ratios GEE approach to that based on the independence “working” model. Values of the SRE criterion greater (less) than 1.0 imply that the local odds ratios GEE approach is more (less) efficient than the independence “working” model in estimating this specific regression parameter. As expected, the two GEE models seem to estimate consistently the marginal model (5), with the local odds ratios GEE approach being 4.05%–9.27% more efficient in estimating each regression coefficient.

The results of Table 1 were calculated using the following R commands:

```
> simindemean <- colMeans(indeGEEcoefs, na.rm = TRUE)
> simindesd <- apply(indeGEEcoefs, 2, function(x) sd(x, na.rm = TRUE))
> simRCmean <- colMeans(RCGEEcoefs, na.rm = TRUE)
> simRCsd <- apply(RCGEEcoefs, 2, function(x) sd(x, na.rm = TRUE))
> simindesmse <- (betas[-c(9:10)] - simindemean)^2 + simindesd^2
> simRCsmse <- (betas[-c(9:10)] - simRCmean)^2 + simRCsd^2
> SRE <- simindesmse/simRCsmse
> rbind(simindemean, simindesd, simRCmean, simRCsd, SRE)
```

### Uniform association model and bivariate normal distribution

Let  $f_{ab}$  denote the observed frequency of the cell  $(a, b)$  in a two-way contingency table and let  $F_{ab}$  be the corresponding expected frequency under some model, for  $a = 1, \dots, A$  and  $b = 1, \dots, B$ . Goodman (1979) proposed the uniform association model

$$\log(F_{ab}) = \nu + \kappa_a + \lambda_b + \phi \quad (6)$$

where the parameters  $\nu, \{\kappa_a : a = 1, \dots, A\}, \{\lambda_b : b = 1, \dots, B\}$  and  $\phi$  are identifiable once restrictions, such as sum to zero constraints (Agresti, 2013), are applied to  $\{\kappa_a : a = 1, \dots, A\}$  and  $\{\lambda_b : b = 1, \dots, B\}$ . The association between the row and column variables is modelled parsimoniously by assuming a common value  $\phi$  for the  $(A - 1) \times (B - 1)$  log local odds ratios. The key property of the uniform association model is that  $\phi$  relates to the correlation parameter  $\rho$  of an underlying bivariate normal distribution (Goodman, 1979) via the approximations

$$\phi \approx \frac{\rho}{1 - \rho^2} \frac{11}{12} \quad (7)$$

or

$$\rho \approx \left( \sqrt{1 + \eta^2} - \eta \right) \times 13/12 \quad (8)$$

where  $\eta = (2\phi)^{-1}$ . The validity of these approximations has been explored only for  $\rho = 0.5$  by Goodman (1979). Here, we perform a more detailed empirical investigation by considering a grid of values for  $\rho$ , namely  $\rho = 0.05, 0.10, \dots, 0.95$ .

For each value of  $\rho$ , we simulated 1000 random vectors from a bivariate normal distribution with mean vector the zero vector and covariance matrix the correlation matrix

$$\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}.$$

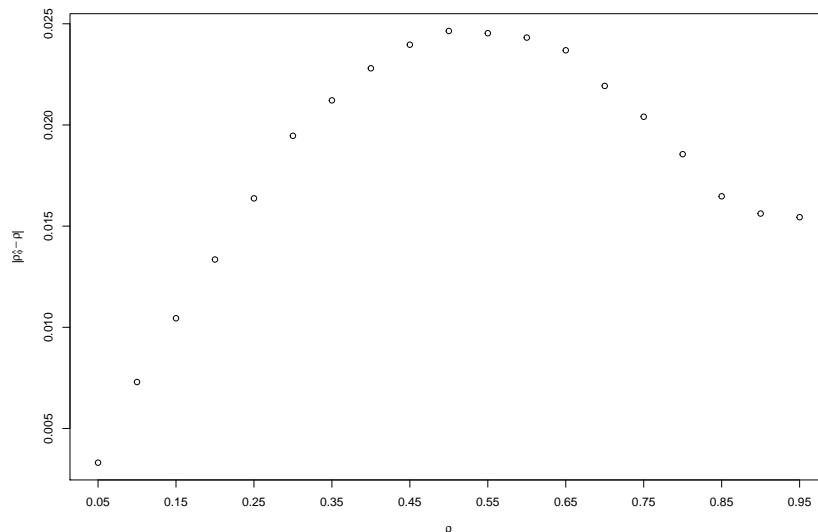
In a similar fashion as in Goodman (1979), correlated ordinal responses were generated by applying the threshold approach linked to model (2), with  $F = \Phi$  and equi-distanced category-specific intercepts  $(\beta_{10}, \beta_{20}, \beta_{30}, \beta_{40}, \beta_{50}, \beta_{60}, \beta_{70}) = (-3, -2, -1, 0, 1, 2, 3)$ . The sampling scheme does not involve any covariates, that is  $\beta_t = \mathbf{0}$  and  $\mathbf{x}_{it} = \mathbf{0}$  for all  $t$ . Next, we cross-classified the correlated simulated responses to obtain a  $8 \times 8$  contingency table and we estimated  $\phi$  by fitting the uniform association model (6). We repeated this procedure 10000 times:

```
> library("SimCorMultRes")
> set.seed(123)
> commonlogoddsratio <- function(N, rho, intercepts, B) {
+   cor.matrix <- toeplitz(c(1, rho))
+   x <- rep(0, 2 * N)
+   ans <- rep(0, B)
+   for (b in 1:B) {
+     CorOrdRes <- rmult.clm(clsize = 2, intercepts = intercepts, betas = 0,
+                             xformula = ~x, link = "probit", cor.matrix = cor.matrix)
+     simdata <- data.frame(table(CorOrdRes$Ysim[, 1], CorOrdRes$Ysim[, 2]))
+     if (any(simdata[, 3] == 0))
+       simdata[, 3] <- simdata[, 3] + 0.001
+     colnames(simdata) <- c("x", "y", "Freq")
+     fit <- glm(Freq ~ x + y + as.numeric(x):as.numeric(y), family = poisson(),
+                data = simdata)
+     ans[b] <- as.numeric(coef(fit)[length(coef(fit))])
+   }
+   ans
+ }
> N <- 1000
> intercepts <- c(-3, -2, -1, 0, 1, 2, 3)
> B <- 10000
> rho <- seq(0.05, 0.95, 0.05)
> logoddsratio <- rep(0, length(rho))
> for (i in seq_along(rho)) {
+   simdata <- commonlogoddsratio(N, rho[i], intercepts, B)
+   logoddsratio[i] <- mean(simdata)
+ }
There were 50 or more warnings (use warnings() to see the first 50)
> eta <- 1/(2 * logoddsratio)
> rhophi <- (sqrt(1 + eta^2) - eta) * 13/12
```

The produced `warnings()` reflect the fact that we have added 0.001 to each cell of the two-way contingency table whenever an observed zero count occurred to ensure the existence of the maximum likelihood estimator of  $\phi$  (Birch, 1963). We estimated the underlying correlation parameter  $\rho$  with  $\rho_{\hat{\phi}}$  obtained by replacing  $\phi$  in (8) with its Monte Carlo counterpart  $\hat{\phi}$ . The following R commands were run to obtain Figure 1.

```
> absdif <- abs(rhophi - rho)
> plot(rho, absdif, xlab = expression(rho), ylab = expression(abs(rho[hat(phi)] -
+ rho)), xaxt = "n")
> axis(1, at = seq(0.05, 0.95, 0.1), labels = seq(0.05, 0.95, 0.1))
```

Figure 1 displays the absolute difference between the true correlation parameter  $\rho$  and  $\rho_{\hat{\phi}}$ . In general, approximation (8) seems to work well for weak correlation patterns, that is when  $\rho \leq 0.20$ . The



**Figure 1:** The absolute difference between the true correlation parameter  $\rho$  and  $\rho_{\hat{\phi}}$ , the correlation implied by the association model (6).

simulated absolute difference increases slightly for  $0.25 \leq \rho \leq 0.55$  and then it decreases as  $0.6 \leq \rho \leq 0.95$ . In addition, the Monte Carlo estimates of  $\phi$  increase as the true value of  $\rho$  increases, which suggests that  $\phi$  does capture the strength of the underlying correlation structure. Therefore, we may conclude that approximations (7) and (8) can adequately describe the relationship between the uniform local odds ratios parameter  $\phi$  in the uniform association model (6) and the correlation parameter  $\rho$  of an underlying bivariate normal distribution.

### Simulating correlated categorical responses under no marginal model specification

For completeness' sake, we illustrate how to utilize **SimCorMultRes** in order to generate correlated categorical random variables conditional on a desired dependence structure and known marginal probabilities that are not determined by a regression model.

Suppose the goal is to simulate 5000 trivariate vectors  $\mathbf{Y}_i = (Y_{i1}, Y_{i2}, Y_{i3})'$  of multinomial responses such that  $Y_{it} \in \{1, 2, 3, 4\}$ ,

$$\begin{array}{llll} \Pr(Y_{i1} = 1) = 0.1 & \Pr(Y_{i1} = 2) = 0.3 & \Pr(Y_{i1} = 3) = 0.4 & \Pr(Y_{i1} = 4) = 0.2 \\ \Pr(Y_{i2} = 1) = 0.2 & \Pr(Y_{i2} = 2) = 0.2 & \Pr(Y_{i2} = 3) = 0.2 & \Pr(Y_{i2} = 4) = 0.4 \\ \Pr(Y_{i3} = 1) = 0.2 & \Pr(Y_{i3} = 2) = 0.4 & \Pr(Y_{i3} = 3) = 0.3 & \Pr(Y_{i3} = 4) = 0.1 \end{array}$$

and a common uniform local odds ratio structure

$$\phi_{tt'} = \frac{\Pr(Y_{it} = j, Y_{it'} = j') \Pr(Y_{it} = j+1, Y_{it'} = j'+1)}{\Pr(Y_{it} = j, Y_{it'} = j'+1) \Pr(Y_{it} = j+1, Y_{it'} = j')} = 2$$

holds for all  $i = 1, \dots, 5000$ ,  $t < t'$  and  $j, j' = 1, 2, 3$ . The above sampling scheme can be reparametrized in terms of the threshold approach related to the marginal cumulative link model (2) while utilizing the conclusions of the previous example to obtain the desired dependence structure.

To this direction, first define  $\beta_{tj0} = \Phi^{-1}[\Pr(Y_{it} \leq j)]$  for all  $t$  ( $t = 1, 2, 3$ ) and  $j$  ( $j = 1, 2, 3$ ) as the category-specific intercepts of a marginal cumulative probit model with no covariates:

```
> library(SimCorMultRes)
> set.seed(123)
> N <- 5000
> clsizes <- 3
> mprobs_1 <- c(0.1, 0.3, 0.4, 0.2)
> mprobs_2 <- c(0.2, 0.2, 0.2, 0.4)
> mprobs_3 <- c(0.2, 0.4, 0.3, 0.1)
> cprobs_1 <- cumsum(mprobs_1[-4])
> cprobs_2 <- cumsum(mprobs_2[-4])
> cprobs_3 <- cumsum(mprobs_3[-4])
> intercepts <- qnorm(rbind(cprobs_1, cprobs_2, cprobs_3))
```

```
> x <- rep(0, clsizes * N)
```

Next, for the pairwise dependence structure, approximate the desired pairwise uniform local odds ratios  $\phi_{12}, \phi_{13}, \phi_{23}$  via the correlation parameters of an underlying trivariate normal distribution with mean vector the zero vector. Since the desired pairwise local odds ratios are all equal ( $\phi_{12} = \phi_{13} = \phi_{23} = 2$ ), we may assume that the corresponding correlation parameters are all equal. This common correlation parameter  $\rho$  can be sufficiently approximated by equation (8), which suggests that  $\rho \approx 0.5543$ :

```
> CommomLOR <- log(2)
> eta <- 1/(2 * CommomLOR)
> rhophi <- (sqrt(1 + eta^2) - eta) * 13/12
> rhophi
[1] 0.5543136
```

To sum up, the desired correlated multinomial responses can be simulated under a cumulative probit model with no covariates and an exchangeable correlation matrix for the underlying trivariate normal distribution with correlation parameter equal to 0.5543:

```
> cor.matrix <- toeplitz(c(1, rhophi, rhophi))
> simdata <- rmult.clm(clsizes = clsizes, intercepts = intercepts, betas = 0,
+                         xformula = ~x, link = "probit", cor.matrix = cor.matrix)
```

The simulated category-specific probabilities satisfy the desired marginal configuration

```
> t(apply(simdata$Ysim, 2, function(x) table(x)/N))
      1     2     3     4
[1,] 0.0970 0.2986 0.4068 0.1976
[2,] 0.1996 0.2046 0.1978 0.3980
[3,] 0.2068 0.3868 0.3068 0.0996
```

and a simulated correlation matrix for the latent random variables

```
> cor(simdata$rlatent)
      [,1]    [,2]    [,3]
[1,] 1.0000000 0.5476759 0.5527712
[2,] 0.5476759 1.0000000 0.5534464
[3,] 0.5527712 0.5534464 1.0000000
```

This approach can also be employed to generate correlated binary random variables with known marginal probabilities provided that the desired correlation structure of the binary responses can be expressed in terms of a correlation matrix in the NORTA method. In this case **SimCorMultRes** is essentially implementing the simulation method of [Emrich and Piedmonte \(1991\)](#) without performing the first step of their algorithm.

## Summary

We have presented the R package **SimCorMultRes** that simulates correlated binary or multinomial random variables conditional on a marginal model specification while expressing the dependence structure via the correlation structure of latent random variables. We outlined the underlying theory that **SimCorMultRes** is based on and illustrated the use of the package with three examples. To the best of our knowledge, **SimCorMultRes** is the first R package that targets specifically on the generation of correlated binary, nominal or ordinal responses under marginal model specification. In some instances, it could also be used to simulate correlated categorical responses even when no model specification is provided for the marginal probabilities by exploiting the relationship of association measures for discrete variables and the bivariate normal distribution. This can be achieved by following a similar approach as the one adopted in the third example herein. The results in this paper were obtained using **SimCorMultRes** version 1.4.1 and R 3.3.1.

Although the NORTA method is the default tool for simulating the latent random vectors denoted by  $e_i$ 's, it is extremely important to emphasize that these can be provided by the user via the `rlatent` argument in the core functions. For example, generating correlated binary responses under a marginal logit model specification and with an exchangeable correlation matrix, can be accomplished by taking the difference of two independent random vectors from the multivariate Gumbel distribution each with correlation matrix the desired correlation matrix. This approach can be found in standard textbooks, such as [Balakrishnan \(1992\)](#). A working example, can be found in the vignette of this package.

A future direction is to increase the scope of marginal regression models for nominal and ordinal responses, e.g., by including threshold approaches that give rise to a marginal adjacent-categories logit model and allowing category-specific regression parameters in the marginal models with ordinal responses.

## Acknowledgments

The author wishes to thank the associate editor and two anonymous referees for their valuable comments and suggestions which significantly improved this manuscript and led to a more flexible implementation of the related methodologies in **SimCorMultRes**.

## Bibliography

- A. Agresti. *Categorical Data Analysis*. John Wiley & Sons, 3rd edition, 2013. [p81, 82, 86]
- A. Amatya and H. Demirtas. Multiord: An R package for generating correlated ordinal data. *Communications in Statistics-Simulation and Computation*, 44:1683–1691, 2015. [p79]
- A. Amatya and H. Demirtas. *MultiOrd: Generation of multivariate ordinal variates*, 2016. URL <http://CRAN.R-project.org/package=MultiOrd>. R package version 2.2. [p79]
- N. Balakrishnan. *Handbook of the Logistic Distribution*. CRC Press, 1992. [p89]
- A. Barbiero and P. A. Ferrari. *GenOrd: Simulation of ordinal and discrete variables with given correlation matrix and marginal distributions*, 2015. URL <http://CRAN.R-project.org/package=GenOrd>. R package version 1.4.0. [p79]
- A. Barbiero and P. A. Ferrari. An R package for the simulation of correlated discrete variables. *Communications in Statistics-Simulation and Computation*, in press. doi: 10.1080/03610918.2016.1146758. [p79]
- W. Bergsma and T. Rudas. Marginal models for categorical data. *The Annals of Statistics*, 30:140–159, 2002. [p80, 83]
- M. W. Birch. Maximum likelihood in three-way contingency tables. *Journal of the Royal Statistical Society B*, 25:220–233, 1963. [p87]
- K. By and B. Qaqish. *mvtBinaryEP: Generates correlated binary data*, 2011. URL <https://CRAN.R-project.org/package=mvtBinaryEP>. R package version 1.0.1. [p81]
- M. C. Cario and B. L. Nelson. Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix. Technical report, Northwestern University, IEMS Technical Report, 1997. [p80, 82, 83]
- N. R. Chaganty and H. Joe. Efficiency of generalized estimating equations for binary responses. *Journal of the Royal Statistical Society B*, 66:851–860, 2004. [p79, 83]
- N. R. Chaganty and H. Joe. Range of correlation matrices for dependent Bernoulli random variables. *Biometrika*, 93:197–206, 2006. [p83]
- H. Demirtas. A method for multivariate ordinal data generation given marginal distributions and correlations. *Journal of Statistical Computation and Simulation*, 76:1017–1025, 2006. [p79]
- H. Demirtas and D. Hedeker. A practical way for computing approximate lower and upper correlation bounds. *The American Statistician*, 65:104–109, 2011. [p80]
- P. Diggle, K. Y. Liang, and S. L. Zeger. *Longitudinal Data Analysis*. Oxford Statistical Science Series, London, 2002. [p79]
- L. J. Emrich and M. R. Piedmonte. A method for generating high-dimensional multivariate binary variables. *American Statistician*, 49:302–304, 1991. [p81, 89]
- P. A. Ferrari and A. Barbiero. Simulating ordinal data. *Multivariate Behavioral Research*, 47:566–589, 2012. [p79]
- G. M. Fitzmaurice and N. M. Laird. A likelihood-based method for analysing longitudinal binary responses. *Biometrika*, 80:141–151, 1993. [p79]

- G. F. V. Glonek and P. McCullagh. Multivariate logistic models. *Journal of the Royal Statistical Society B*, 57:533–546, 1995. [p<sup>79</sup>]
- L. A. Goodman. Simple models for the analysis of association in cross-classifications having ordered categories. *Journal of the American Statistical Association*, 74:537–552, 1979. [p<sup>84</sup>, 86, 87]
- S. T. Li and J. L. Hammond. Generation of pseudorandom numbers with specified univariate distributions and correlation coefficients. *IEEE Transactions on Systems, Man and Cybernetics*, 5:557–561, 1975. [p<sup>82</sup>]
- S. R. Lipsitz, N. M. Laird, and D. P. Harrington. Generalized estimating equations for correlated binary data: Using the odds ratio as a measure of association. *Biometrika*, 78:153–160, 1991. [p<sup>79</sup>]
- G. Masarotto and C. Varin. Gaussian copula marginal regression. *Electronic Journal of Statistics*, 6: 1517–1549, 2012. [p<sup>79</sup>]
- P. McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society B*, 42:109–142, 1980. [p<sup>81</sup>]
- D. McFadden. Conditional logit analysis of qualitative choice behaviour. In P. Zarembka, editor, *Frontiers in Econometrics*, pages 105–142. Academic Press, New York, 1974. [p<sup>82</sup>]
- J. Shults and N. R. Chaganty. Analysis of serially correlated data using quasi-least squares. *Biometrics*, 54:1622–1630, 1998. [p<sup>79</sup>]
- B. C. Sutradhar. An overview on regression models for discrete longitudinal responses. *Statistical Science*, 18:377–393, 2003. [p<sup>79</sup>]
- B. C. Sutradhar and K. Das. On the accuracy of efficiency of estimating equation approach. *Biometrika*, 86:459–465, 1999. [p<sup>79</sup>]
- A. Touloumis. R package multgee: A generalized estimating equations solver for multinomial responses. *Journal of Statistical Software*, 64:1–14, 2015. [p<sup>85</sup>]
- A. Touloumis. *SimCorMultRes: Simulates correlated multinomial responses*, 2016. URL <http://CRAN.R-project.org/package=SimCorMultRes>. R package version 1.4.1. [p<sup>79</sup>]
- A. Touloumis, A. Agresti, and M. Kateri. Generalized estimating equations for multinomial responses using a local odds ratios parameterization. *Biometrics*, 69:633–640, 2013. [p<sup>79</sup>, 83, 85]
- G. Tutz. Sequential models in categorical regression. *Computational Statistics & Data Analysis*, 11: 275–295, 1991. [p<sup>81</sup>]

Anestis Touloumis  
Computing, Engineering and Mathematics  
University of Brighton  
Brighton, United Kingdom  
[A.Touloumis@brighton.ac.uk](mailto:A.Touloumis@brighton.ac.uk)

# eiCompare: Comparing Ecological Inference Estimates across EI and EI:R $\times$ C

by Loren Collingwood, Kassra Oskooii, Sergio Garcia-Rios, and Matt Barreto

**Abstract** Social scientists and statisticians often use aggregate data to predict individual-level behavior because the latter are not always available. Various statistical techniques have been developed to make inferences from one level (e.g., precinct) to another level (e.g., individual voter) that minimize errors associated with ecological inference. While ecological inference has been shown to be highly problematic in a wide array of scientific fields, many political scientists and analysts employ the techniques when studying voting patterns. Indeed, federal voting rights lawsuits now require such an analysis, yet expert reports are not consistent in which type of ecological inference is used. This is especially the case in the analysis of racially polarized voting when there are multiple candidates and multiple racial groups. The **eiCompare** package was developed to easily assess two of the more common ecological inference methods: EI and EI:R $\times$ C. The package facilitates a seamless comparison between these methods so that scholars and legal practitioners can easily assess the two methods and whether they produce similar or disparate findings.

## Introduction

Ecological inference is a widely debated methodology for attempting to understand individual, or micro behavior from aggregate data. Ecological inference has come under fire for being unreliable, especially in the fields of biological sciences, ecology, epidemiology, public health and many social sciences. For example, Freedman (1999) explains that when confronted with individual level data, many ecological aggregate estimates in epidemiology have been proven to be wrong. In the field of ecology Martin et al. (2005) expose the problem of zero-inflation in studies of the presence or absence of specific species of different animals and note that ecological techniques can lead to incorrect inference. Greenland (2001) describes the many pitfalls of ecological inference in public health due to the nonrandomization of social context across ecological units of analysis. Elsewhere, Greenland and Robins (1994) have argued that the problem of ecological confounder control leads to biased estimates of risk in epidemiology. Related, Frair et al. (2010) argue that while some ecological analysis can be informative when studying animal habitat preference, existing methods of ecological inference provide imprecise information on variation in the outcome variables and that considerable improvements are necessary. Wakefield (2004) provides a nice comparison of how ecological inference performs across epidemiological versus social scientific research. He concludes that in epidemiological applications individual-level data are required for consistently accurate statistical inference.

However, within the narrow subfield of racial voting patterns in American elections ecological inference is regularly used. This is especially common in scholarly research on the voting rights act where the United States Supreme Courts directly recommended ecological inference analysis as the main statistical method to estimate voting preference by racial group (e.g. *Thornburg v. Gingles* 478 U.S. 30, 1986). Because Courts in the U.S. have so heavily relied on ecological inference, it has gained prominence in political science research. The American Constitution Society for Law and Policy explains that ecological inference is one of the three statistical analyses that must be performed in voting rights research on racial voting patterns.<sup>1</sup> As ecological inference evolved a group of scholars developed the **eiPack** package and published an article in *R News* announcing the new package (Lau et al., 2006).

This article does not conclude that ecological inference is appropriate or reliable outside the specific domain of American elections. Indeed, scholars in the fields of epidemiology and public health have correctly pointed out the limitations of individual level inference from aggregate data. However, its application to voting data in the United States represents one area where it may have utility, if model assumptions are met (Tam Cho and Gaines, 2004). Indeed, the main point of our article is not to settle the debate on the accuracy of ecological inference in the sciences writ large, but rather to assess the degree of similarity or difference with respect to two heavily used R packages within the field of political science, **ei** and **eiPack**. Our package, **eiCompare** offers scholars who regularly use ecological inference in analyses of voting patterns the ability to easily compare, contrast and diagnose estimates across two different ecological methods that are recommended statistical techniques in voting rights litigation.

Today, although there is continued debate among social scientists (Greiner, 2007, 2011; Cho, 1998)—the courts generally rely on two statistical approaches to ecological data. The first, ecological inference

<sup>1</sup>[http://www.acslaw.org/sites/default/files/VRI\\_Guide\\_to\\_Section\\_2\\_Litigation.pdf](http://www.acslaw.org/sites/default/files/VRI_Guide_to_Section_2_Litigation.pdf)

(EI), developed by King (1997), is said to be preferred when there are only two racial or ethnic groups, and ideally only two candidates contesting office. However, Wakefield (2004) notes that EI methods can be improved with the use of survey data as Bayesian priors. The second, ecological inference  $R \times C$  ( $R \times C$ ) developed by Rosen et al. (2001), is said to be preferred when there are multiple racial or ethnic groups, or multiple candidates contesting office. However, it is not clear that when faced with the exact same dataset, they would produce different results. In one case, analysis of the same dataset across multiple ecological approaches found they tend to produce the same conclusion (Grofman and Barreto, 2009). However, others have argued that using King's EI iterative approach with multiple racial groups or multiple candidates will fail and should not be relied on (Ferree, 2004). Still others have gone further and stated that EI cannot be used to analyze multiple racial group or multiple candidate elections, stating that "it biases the analysis for finding racially polarized voting," going on to call this approach "problematic" and stating that "no valid statistical inferences can be drawn" (Katz, 2014).

As with any methodological advancement, there is a healthy and rigorous debate in the literature. However, very little real election data has been brought to bear in this debate. Ferree (2004) offers a simulation of Black, White, and Latino turnout and voting patterns, and then examines real data from a parliamentary election in South Africa using a proportional representation system. (Grofman and Barreto, 2009) compare an exit poll to precinct election data in Los Angeles, but only compare Goodman's ecological regression against King's EI, using the single-equation versus double-equation approach, and do not examine the  $R \times C$  approach at all.

## Debates Over Ecological Inference

The challenges surrounding ecological inference are well documented. Robinson (2009) pointed out that relying on aggregate data to infer the behavior of individuals can result in the ecological fallacy, and since then scholars have applied different methods to discern more accurately individual correlations from aggregate data. Goodman (1953, 1959) advanced the idea of ecological regression where individual patterns can be drawn from ecological data under certain conditions. However Goodman's logic assumed that group patterns were consistent across each ecological unit, and in reality that may not be the case.

Eventually, systematic analysis revealed that these early methods could be unreliable (King, 1997). Ecological inference is King's (1997) solution to the ecological fallacy problem inherent in aggregate data, and since the late 1990s has been the benchmark method courts use in evaluating racial polarization in voting rights lawsuits, and has been used widely in comparative politics research on group and ethnic voting patterns. Critics claim that King's EI model was designed primarily for situations with just two groups (e.g., blacks and whites; Hispanics and Anglos, etc.). While many geographic areas (e.g., Mississippi, Alabama) still contain essentially two groups and hence pose no threat to traditional EI estimation procedures, the growth of racial groups such as Latinos and Asians have challenged the historical biracial focus on race in the United States (thereby challenging traditional EI model assumptions). Rosen et al. (2001) suggest a rows by columns ( $R \times C$ ) approach which allows for multiple racial groups, and multiple candidates; however, their Bayesian approach suffered computational difficulties and was not employed at a mass level. Since then, computing power has steadily improved, making  $R \times C$  a realistic solution for many scenarios and accessible packages now exist in R that are widely used. These two methodological approaches are now both regularly used in political science; however, there is no consistent evidence how they perform side-by-side, and are different.

Ferree (2004) critiques King's EI model, arguing that the conditions for iterative estimation (e.g., black vs. non black, white vs. non-white, Hispanic vs. non-Hispanic) can be considerably biased due to aggregation bias and multimodality in the data. In a hypothetical simulation dataset, Ferree shows that combining blacks and whites into a single "non-Hispanic" group in order to estimate Hispanic turnout can vastly overestimate Hispanic turnout, for example. However, the analysis did not provide any clues as to the specific conditions when and how  $R \times C$  is significantly better or preferred to EI. For example, if there are three racial groups in equal thirds of the electorate, does aggregation bias create more error in EI than a scenario in which two dominant groups comprise 90% and a small group is just 10%? Likewise, is EI's iterative approach to candidates more stable when analyzing three candidates and far less stable when eight candidates contest the election? These questions have not been considered empirically. Instead, the existing scholarship uses simulation data to prove theoretically that EI might create bias and that  $R \times C$  is preferred. We argue that real election data should be considered in a side-by-side comparison.

Despite some critiques, other political scientists have defended ecological inference and even ecological regression using both simulations and real data. Owen and Grofman (1997) assess whether or not ecological fallacy in ecological regression is a theoretical problem only, a real problem for

empirical analysis. In an extensive review, Owen and Grofman conclude that despite the valid theoretical concerns, linear ecological regression still holds up and provides meaningful and accurate estimates of racially polarized voting. A decade later, [Grofman and Barreto \(2009\)](#) again take up the question of how ecological models compare to one another using a combination of simulation, actual election precinct data, and an accompanying individual-level exit poll. Their analysis argues that there is general consistency across all ecological models and that once voter turnout rates are accounted for, ecological regression and King's EI lead scholars to the same results. However, Grofman and Barreto did not consider R×C in their comparison.

[Greiner and Quinn \(2010\)](#) combine R×C methods with individual level exit poll data, and argue that this hybrid model can be preferable to a straight aggregation model. However, using exit poll data is not always available to all researchers and practitioners. Indeed, in most county or city elections, exit poll data does not exist which is why scholars often attempt to infer voting patterns through aggregate data. [Herron and Shotts \(2003\)](#) also criticize EI estimates when used for second-stage regression - given that error is baked into the second-level regression estimation. However [Adolph and King \(2003\)](#) respond by adjusting the EI procedure to reduce inconsistencies when estimating second-stage regressions. Nevertheless, these issues with EI do not speak specifically to R×C methods.

[Greiner and Quinn \(2009\)](#) extend the 2×2 EI contingency problem to 3×3 and estimate voting preferences simultaneously for three candidates across three racial groups (but using counts instead of percentages). We extend this work by analyzing real-world datasets with sizes greater than 3×3 (multiple candidates and at least three racial groups). In all of this, our main goal is to assess whether using iterative EI or simultaneous R×C approaches change the conclusions social scientists can make from the data.

Finally, some have gone even further in arguing that EI is ill-equipped to handle complex datasets with multiple candidates and multiple racial groups, and that only R×C can produce reliable results ([Katz, 2014](#)). In explaining the theoretical reasons why EI cannot accurately process such elections Katz argues "adding additional groups and vote choices to King's (1997) EI is not straightforward," and also adds "given the estimation uncertainty, it may not be possible to infer which candidate is preferred by members of the group." The argument against EI in multiple racial group, or especially multiple candidate elections is that EI takes an iterative approach pitting candidate A versus all others who are not candidate A. If the election features four candidates (A, B, C, D) critics state that you cannot accurately estimate vote choice quantities if you compare the vote for candidate A against the combined vote for B, C, D. The iterative approach would then move on to estimate the vote share for candidate B against the combined vote for A, C, D and so on, so that four separate equations are run. [Katz \(2014\)](#) claims that EI biases the findings in favor of bloc-voting stating "this jerry rigged approach to dealing with more than two vote choices stacks the deck in favor of finding statistical evidence for racially polarized." Given these debates, our package allows scholars to quite easily make side-by-side comparisons and evaluate these competing claims.

While important advancements have been made in ecological inference techniques by [King \(1997\)](#) and [Rosen et al. \(2001\)](#) there is no consistency in which technique is used and how results are presented. What's more, legal experts and social scientists often argue during voting rights lawsuits that one technique is superior to the other, or that their results are more accurate. There is no question that both social scientists and legal experts would greatly benefit from a standardized software package that presents both ecological inference results (EI and R×C) simultaneously and metrics to compare each set of results. Thus, **eiCompare** was designed to compare the most commonly used methods today, EI and R×C, but also incorporates Goodman methods. The package lets analysts seamlessly assess whether EI and R×C estimates are similar (see [King \(1997\)](#) and [Rosen et al. \(2001\)](#) for a methodological description of the techniques). It incorporates functions from **ei** ([King and Roberts, 2013](#)) and **eiPack** ([Lau et al., 2012](#)) into a new package that relatively quickly compares ecological inference estimates across the two routines.

The package includes several functions that ultimately produce tables of results from the different ecological inference methods. Thus, in the case of racially polarized voting, analysts can quickly assess whether different racial groups preferred different candidates, according to the EI, R×C, and Goodman approaches. The **eiCompare** package wraps the **ei()** procedure ([King and Roberts, 2012](#)) into a generalized function, has a variety of table-making functions, and a plotting method that graphically depicts the difference between estimates for the two main EI methods (EI and R×C). Below, we use a working example of a voter precinct dataset in Corona, CA. To use the package, the process is simple: 1) Load the package, the appropriate data, run the EI generalized function, and create an EI table of results, 2) Run the R×C function (from **eiPack**) and create a table of results, 3) Run the Goodman regression generalized function if the user chooses, 4) Combine the results of all the algorithms together into a comparison table, and 5) Plot the comparison results. Before we conclude, we also compare EI and R×C findings against exit poll data from a 2005 Los Angeles mayoral run-off election. The rest of the paper follows this aforementioned outline.

## EI Generalize

To begin, we install (`install.packages("eiCompare")`) and load the **eiCompare** package (`library(eiCompare)`) from the CRAN repository. First, we load the aggregate-level dataset (`data(cor_06)`) into R, in this case a precinct (voting district) dataset from a 2006 election in the city of Corona, CA. Table 1 below displays the first five rows and column headers of the dataset. This dataset includes all the necessary variables to run the code in the **eiCompare** package. The first column is "precinct", which essentially operates as a unique identifier. The second column, "totvote", is the total number of votes cast within the precinct. Columns three and four are the two racial groups of whom we seek to determine their mean voting preference. The rest of the columns are the percent of the total vote for each respective candidate.

precinct	totvote	pct_latino	pct_other	pct_breitenbucher	pct_montanez	pct_spiegel	pct_skipworth
1	22000	0.21	0.79	0.20	0.21	0.29	0.30
2	22002	0.16	0.84	0.22	0.22	0.29	0.27
3	22003	0.21	0.79	0.22	0.22	0.30	0.26
4	22004	0.45	0.55	0.18	0.27	0.30	0.24
5	22008	0.31	0.69	0.23	0.25	0.30	0.22
6	22012	0.21	0.79	0.20	0.24	0.32	0.24

**Table 1:** Precinct dataset of Corona, CA, used for ecological inference. Each row is a precinct, the dataset must have a total column, racial/ethnic percentages of people living in the precinct, and vote percent for each candidate.

We are interested in how the four candidates (Breitenbucher, Montanez, Spiegel, Skipworth) performed with Latino voters and non-Latino voters (mostly non-Hispanic white), so we can assess whether racially polarized voting exists. The process begins with the `ei_est_gen()` function, which is a generalized version of the `ei()` function from the **ei** package. Instead of having to estimate EI results for each candidate and each racial group separately, `ei_est_gen()` automates this process.

The `ei_est_gen()` function takes a vector of candidate names, a character vector of tilde-prefixed racial group names, the name of the column representing the total number of people in the jurisdiction (e.g., registered voters, ballots cast), the "data.frame" object holding the data, and the table names used to display the results. The function also has four optional arguments, `rho`, `sample`, `tomog`, and `density_plot`. The former two can be used to adjust the parameters of the `ei()` algorithm. These are especially useful when the initial run does not compile or warnings are produced. The latter two plot out tomography and density plots, respectively, into the working directory but are set to off by default. These plots can be used to assess the stability – and thus veracity – of the EI procedure (see King and Roberts (2012) and King (1997) for details). Finally, the ... argument passes additional arguments onto the `ei()` function from the **ei** package.

One final note, given its iterative nature, the `ei_est_gen()` function can take a while to execute. This typically depends on features unique to the dataset, including the number of candidates and groups, the amount of racial/ethnic segregation within the city/area, as well as the number of precincts. This particular example does not take especially long, executing in about a minute on a standard Macbook Pro.

```
# LOAD DATA
data(cor_06)
# SET SEED FOR REPRODUCIBILITY
set.seed(294271)
# CREATE CHARACTER VECTORS REQUIRED FOR FUNCTION
cands <- c("pct_breitenbucher", "pct_montanez", "pct_spiegel", "pct_skipworth")
race_group2 <- c("~/ pct_latino", "~/ pct_other")
table_names <- c("EI: Pct Lat", "EI: Pct Other")
# RUN EI GENERALIZED FUNCTION
results <- ei_est_gen(cand_vector=cands, race_group = race_group2,
                      total = "totvote", data = cor_06, table_names = table_names)
# LOOK AT TABLE OF RESULTS
results
```

The call to the `results` object produces a table of results indicating the mean estimated voting preferences for Latinos and non-Latinos within the city of Corona (see Table 2). The results strongly suggest the presence of racially polarized voting, as Latinos prefer Montanez as their number one choice, whereas non-Latinos do not.

Candidate	EI: Pct Lat	EI: Pct Other
pct_breitenbucher	19.68	21.12
se	0.75	0.13
pct_montanez	35.95	20.13
se	0.03	0.08
pct_spiegel	28.43	31.01
se	0.57	0.23
pct_skipworth	18.64	26.84
se	0.71	0.23
Total	102.69	99.10

**Table 2:** EI mean estimates for Latino and Non-Latino candidate vote preferences in Corona, 2006**EI: R×C**

The R×C builds off of code from the **eiPack** package, where **eiCompare** simply takes the former's results and puts them into a similar "data.frame"/"table" object similar to the results from the **ei\_est\_gen()** function. First, the user follows the code from the **eiPack** package (here we use the **ei.reg.bayes()** function), and creates a formula object including all candidates and all groups. The user must ensure that the percentages on both signs of the ~ symbol add to 1. Thus, the initial **table()** code is a simple data check to ensure that this rule is followed. The R×C model is then run using the **ei.reg.bayes()** model. Users can read the **eiPack** documentation to familiarize themselves with this procedure. Depending on the nature of one's data, the R×C code can take a while to run. Finally, the results are passed onto the **bayes\_table\_make()** function, along with a vector of candidate names, and a vector of table names, similar to what was passed to **ei\_est\_gen()**.

```
# CHECK TO MAKE SURE DATA SUMS TO 1 FOR EACH PRECINCT
with(cor_06, pct_latino + pct_other)
with(cor_06, pct_breitenbucher + pct_montanez + pct_spiegel + pct_skipworth)
# SET SEED FOR REPRODUCIBILITY
set.seed(124271)
#RxC GENERATE FORMULA
form <- formula(cbind(pct_breitenbucher,pct_montanez,
                       pct_spiegel, pct_skipworth) ~ cbind(pct_latino, pct_other))
# RUN EI:RxC MODEL
ei_bayes <- ei.reg.bayes(form, data = cor_06, sample = 10000, truncate = TRUE)
# CREATE TABLE NAMES
table_names <- c("RxC: Pct Lat", "RxC: Pct Other")

# TABLE CREATION
ei_bayes_res <- bayes_table_make(ei_bayes, cand_vector = cands, table_names = table_names)
# LOOK AT TABLE OF RESULTS
ei_bayes_res
```

Candidate	RxC: Pct Lat	RxC: Pct Other
pct_breitenbucher	18.22	21.58
se	1.62	0.53
pct_montanez	34.96	20.44
se	1.72	0.56
pct_spiegel	28.24	31.05
se	1.08	0.35
pct_skipworth	18.61	26.91
se	1.73	0.56
Total	100.03	99.99

**Table 3:** EI:R×C mean estimates for Latino and Non-Latino candidate vote preferences in Corona, 2006

The results are presented in Table 3, and look remarkably similar to those presented in Table 2. Indeed, the exact same conclusions would be drawn from an analysis of both tables: Latinos prefer Montanez as their first choice and non-latinos prefer Spiegel as their top choice.

## Goodman Generalize

While many users will skip over the Goodman regression when conducting ecological inference, given the documented issues with the method (Shively, 1969; King, 1997), `eiCompare` nevertheless has a Goodman regression generalized function, similar to the `ei_est_gen()` function. This function takes a character vector of candidate names, a character vector of racial groups, the name of the column, a data object, and a character vector of table names. Because Goodman is simply a linear regression, the execution is very fast.

```
table_names <- c("Good: Pct Lat", "Good: Pct Other")
good <- goodman_generalize(cands, race_group2, "totvote", cor_06, table_names)
good
```

Table 4 shows the Goodman regression results. In this particular case, these results align quite closely with results from the two EI models. All three approaches essentially tell us the same thing.

Candidate	Good: Pct Lat	Good: Pct Other
pct_breitenbucher	17.51	20.34
se	3.18	3.74
pct_montanez	35.00	20.48
se	3.41	4.01
pct_spiegel	28.52	31.61
se	2.16	2.54
pct_skipworth	18.97	27.57
se	3.45	4.05
Total	100.00	100.00

**Table 4:** Goodman regression estimates for Latino and Non-Latino candidate vote preferences in Corona, 2006

## Combining Results

The last two sections address the comparison component of the package. The function, `ei_rc_good_table()`, takes the objects from the EI, R×C, and Goodman regression, and puts them into a "data.frame""table" object. To simplify comparison, the table adds an EI-R×C column differential for each racial group. This format lets the user quickly assess how the EI and R×C methods stack up against one another. The function takes the following arguments: EI results object (e.g., `results`), an R×C object (e.g., `ei_bayes_res`), and a character vector groups (e.g., `c("Latino", "Other")`) argument. The `good` argument for the Goodman regression is set to NULL, and the `include_good` argument defaults to FALSE. If the user wants to include a Goodman regression in the comparison of results they need to change the latter to TRUE and specify the the `good` argument as the object name from the `goodman_generalize()` call.

Candidate	EI: Pct Lat	RxC: Pct Lat	EI_Diff	EI: Pct Other	RxC: Pct Other	EI_Diff
pct_breitenbucher	19.68	18.22	-1.46	21.12	21.58	0.46
se	0.75	1.62		0.13	0.53	
pct_montanez	35.95	34.96	-0.99	20.13	20.44	0.31
se	0.03	1.72		0.08	0.56	
pct_spiegel	28.43	28.24	-0.19	31.01	31.05	0.04
se	0.57	1.08		0.23	0.35	
pct_skipworth	18.64	18.61	-0.02	26.84	26.91	0.07
se	0.71	1.73		0.23	0.56	
Total	102.69	100.03	-2.66	99.10	99.99	0.88

**Table 5:** EI and R×C comparisons for Latino and Non-Latino candidate vote preferences in Corona, 2006

The results of `ei_rc_good_table()` is a new class "ei\_compare", which includes a "data.frame" and groups character vector. This output is ultimately passed to `plot()`.

```
ei_rc_combine <- ei_rc_good_table(results, ei_bayes_res,
                                    groups = c("Latino", "Other"))
ei_rc_combine@data
```

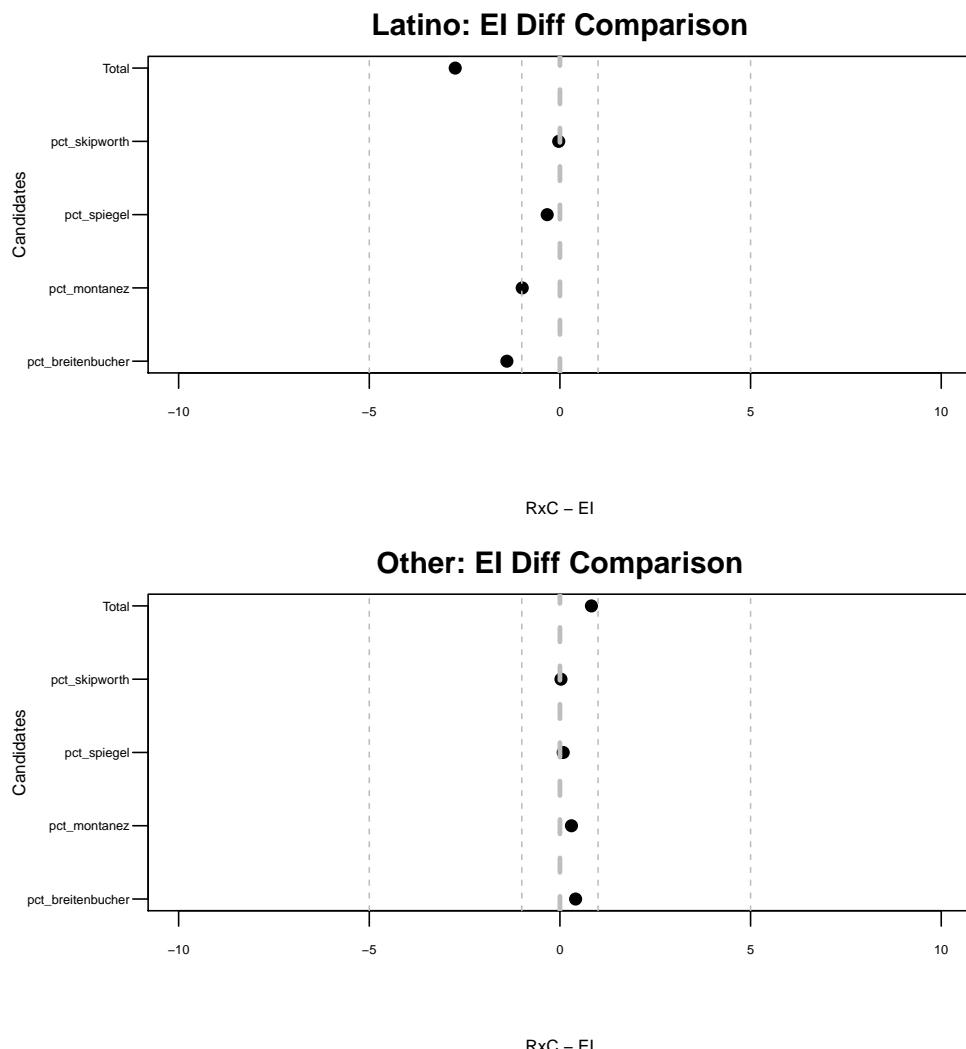
```
ei_rc_g_combine <- ei_rc_good_table(results, ei_bayes_res, good,
                                     groups = c("Latino", "Other"), include_good = TRUE)
ei_rc_g_combine
```

Table 5 displays the output of a call to the `ei_rc_good_table()` function for the first line of code above. The user must include the code @data onto the outputted table name to extract just the table. This table basically summarizes the results of the EI and R $\times$ C analyses. Clearly, very little difference emerges between the two methods in this particular instance.

## Plotting Results

Finally, users can plot the results of the EI, and R $\times$ C comparison to more visually determine whether the two methods are similar. Plotting is simple, as plot methods have been developed for the "ei\_compare" class. The code below produces the plot depicted in Figure 1.

```
# PLOT COMPARISON -- adjust the axes labels slightly
plot(ei_rc_combine, cex.axis = .5, cex.lab = .7)
```



**Figure 1:** Comparison of EI and R $\times$ C methods for Corona 06 precinct data

## Comparing Ecological and Individual-Level Data

One possible question remains, whether or not ecological estimates line up with individual level estimates. Many studies have pointed out that ecological fallacy and aggregation bias can produce ecological inference results that are highly questionable. In this section we implement the **eiCompare** package for a mayoral election in a multiethnic setting in which an individual-level exit poll survey was also administered. The **eiCompare** package provides EI and R $\times$ C results for the 2005 Los Angeles mayoral runoff election between Antonio Villaraigosa and James Hahn, and we also add results for the Los Angeles Times exit poll. Results are displayed in Table 6.

	EI: AV	EI: JH	RxC: AV	RxC: JH	Exit: AV	Exit: JH	MOE
White	45	54	48	52	50	50	+/- 2.5
Black	58	40	50	50	48	52	+/- 4.2
Latino	82	17	81	19	84	16	+/- 3.6
Asian	48	51	47	53	44	56	+/- 6.1

**Table 6:** Percent voting for Antonio Villaraigosa (AV) and James Hahn (JH) by ethnic group. Comparison between EI, R $\times$ C, and exit poll methods, Los Angeles mayoral election runoff, May 2005. Exit poll taken from Los Angeles Times.

The results presented in Table 6 demonstrate that not only do EI and R $\times$ C produce remarkably consistent results, but they very closely match the individual level estimates for the Los Angeles Times. The EI R $\times$ C estimates are all within the confidence range of the individual level data reported by the exit poll.

## Summary

**eiCompare** is a new package that builds on the work of King and others that attempts to address the ecological inference problem of making individual-level assessments based on aggregate-level data. As we have reviewed above, there is considerable debate in the sciences about the utility and accuracy of ecological techniques. Despite these well documented questions, ecological inference is widely used in political science and will continue to grow in importance when the constitutionally mandated redistricting in 2021 occurs. The redistricting cycle will bring with it extensive academic, legislative, and legal research using ecological inference to assess racial voting patterns across all 50 states.

While this new package does not develop a new method, *per se*, it improves analysts' ability to quickly compare different commonly used EI algorithms to assess the veracity of the methods and also produce tables of their findings. While R $\times$ C has been touted as the method necessary in situations with multiple groups and multiple candidates, the results do not always demonstrate face validity. In these scenarios – and others – analysts may want to incorporate original EI methods so they can compare how the two approaches stack up. Ultimately, this approach provides a needed assessment between two commonly used methods in voting behavior research.

## Bibliography

- C. Adolph and G. King. Analyzing second-stage ecological regressions: Comment on Herron and Shotts. *Political Analysis*, 11(1):65–76, 2003. [p94]
- W. K. T. Cho. Iff the assumption fits?: A comment on the King ecological inference solution. *Political Analysis*, 7(1):143–163, 1998. [p92]
- K. E. Ferree. Iterative approaches to R $\times$ C ecological inference problems: where they can go wrong and one quick fix. *Political Analysis*, 12(2):143–159, 2004. [p93]
- J. L. Frair, J. Fieberg, M. Hebblewhite, F. Cagnacci, N. J. DeCesare, and L. Pedrotti. Resolving issues of imprecise and habitat-biased locations in ecological analyses using GPS telemetry data. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 365(1550):2187–2200, 2010. [p92]
- D. A. Freedman. Ecological inference and the ecological fallacy. *International Encyclopedia of the social & Behavioral sciences*, 6:4027–4030, 1999. [p92]
- L. A. Goodman. Ecological regressions and behavior of individuals. *American sociological review*, 1953. [p93]

- L. A. Goodman. Some alternatives to ecological correlation. *American Journal of Sociology*, pages 610–625, 1959. [p93]
- S. Greenland. Ecologic versus individual-level sources of bias in ecologic estimates of contextual health effects. *International journal of epidemiology*, 30(6):1343–1350, 2001. [p92]
- S. Greenland and J. Robins. Invited commentary: ecologic studies — biases, misconceptions, and counterexamples. *American Journal of Epidemiology*, 139(8):747–760, 1994. [p92]
- D. J. Greiner. Ecological inference in voting rights act disputes: Where are we now, and where do we want to be? *Jurimetrics*, pages 115–167, 2007. [p92]
- D. J. Greiner. The quantitative empirics of redistricting litigation: Knowledge, threats to knowledge, and the need for less districting. *Yale Law & Policy Review*, 29(2):527–542, 2011. [p92]
- D. J. Greiner and K. M. Quinn. Exit polling and racial bloc voting: Combining individual-level and R×C ecological data. *The Annals of Applied Statistics*, pages 1774–1796, 2010. [p94]
- J. D. Greiner and K. M. Quinn. R×C ecological inference: bounds, correlations, flexibility and transparency of assumptions. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 172(1):67–81, 2009. [p94]
- B. Grofman and M. A. Barreto. A reply to Zax’s (2002) critique of Grofman and Migalski (1988) double-equation approaches to ecological inference when the independent variable is misspecified. *Sociological Methods & Research*, 37(4):599–617, 2009. [p93, 94]
- M. C. Herron and K. W. Shotts. Using ecological inference point estimates as dependent variables in second-stage linear regressions. *Political Analysis*, 11(1):44–64, 2003. [p94]
- J. N. Katz. Expert report on voting in the city of Whittier. March 5, 2014. [p93, 94]
- G. King. A solution to the ecological inference problem, 1997. [p93, 94, 95, 97]
- G. King and M. Roberts. Ei: a (n r) program for ecological inference. *Harvard University*. Retrieved from <http://gking.harvard.edu/files/ei.pdf>, 2012. [p94, 95]
- G. King and M. Roberts. ei: ei, 2013. URL <http://gking.harvard.edu/zelig>. R package version 1.3. [p94]
- O. Lau, R. T. Moore, and M. Kellermann. eipack: R×C ecological inference and higher-dimension data management. *New Functions for Multivariate Analysis*, 18(1):43, 2006. [p92]
- O. Lau, R. T. Moore, and M. Kellermann. eiPack: eiPack: Ecological Inference and Higher-Dimension Data Management, 2012. URL <http://CRAN.R-project.org/package=eiPack>. R package version 0.1-7. [p94]
- T. G. Martin, B. A. Wintle, J. R. Rhodes, P. M. Kuhnert, S. A. Field, S. J. Low-Choy, A. J. Tyre, and H. P. Possingham. Zero tolerance ecology: improving ecological inference by modelling the source of zero observations. *Ecology letters*, 8(11):1235–1246, 2005. [p92]
- G. Owen and B. Grofman. Estimating the likelihood of fallacious ecological inference: linear ecological regression in the presence of context effects. *Political Geography*, 16(8):675–690, 1997. [p93]
- W. S. Robinson. Ecological correlations and the behavior of individuals. *International journal of epidemiology*, 38(2):337–341, 2009. [p93]
- O. Rosen, W. Jiang, G. King, and M. A. Tanner. Bayesian and frequentist inference for ecological inference: The R×C case. *Statistica Neerlandica*, 55(2):134–156, 2001. [p93, 94]
- W. P. Shively. “Ecological” inference: the use of aggregate data to study individuals. *American Political Science Review*, 63(04):1183–1196, 1969. [p97]
- W. K. Tam Cho and B. J. Gaines. The limits of ecological inference: The case of split-ticket voting. *American Journal of Political Science*, 48(1):152–171, 2004. [p92]
- J. Wakefield. Ecological inference for 2×2 tables (with discussion). *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 167(3):385–445, 2004. [p92, 93]

*Loren Collingwood*  
*Department of Political Science*  
*University of California, Riverside*  
*900 University Avenue*  
*Riverside, CA 92521*  
*USA*  
[loren.collingwood@ucr.edu](mailto:loren.collingwood@ucr.edu)

*Kassra Oskooii*  
*Department of Political Science*  
*University of Washington*  
*101 Gowen Hall*  
*Seattle, WA 98195*  
*USA*  
[kassrao@uw.edu](mailto:kassrao@uw.edu)

*Sergio Garcia-Rios*  
*Department of Government*  
*Cornell University*  
*212 White Hall*  
*Ithaca, NY 14853*  
*USA*  
[garcia.rios@cornell.edu](mailto:garcia.rios@cornell.edu)

*Matt A. Barreto*  
*Department of Political Science*  
*University of California, Los Angeles*  
*Bunche Hall 3284*  
*Los Angeles, CA 90095*  
*USA*  
[barretom@ucla.edu](mailto:barretom@ucla.edu)

# rnrfa: An R package to Retrieve, Filter and Visualize Data from the UK National River Flow Archive

by Claudia Vitolo, Matthew Fry and Wouter Buytaert

**Abstract** The UK National River Flow Archive (NRFA) stores several types of hydrological data and metadata: daily river flow and catchment rainfall time series, gauging station and catchment information. Data are served through the NRFA web services via experimental RESTful APIs. Obtaining NRFA data can be unwieldy due to complexities in handling HTTP GET requests and parsing responses in JSON and XML formats. The **rnrfa** package provides a set of functions to programmatically access, filter, and visualize NRFA data using simple R syntax. This paper describes the structure of the **rnrfa** package, including examples using the main functions `gdf()` and `cmr()` for flow and rainfall data, respectively. Visualization examples are also provided with a **shiny** web application and functions provided in the package. Although this package is regional specific, the general framework and structure could be applied to similar databases.

## Introduction

The increasing volume of environmental data available online poses non-trivial challenges for efficient storage, access and share of this information (Vitolo et al., 2015). An integrated and consistent use of data is achieved by extracting data directly from web services and processing them on-the-fly. This improves the flexibility of modelling applications allowing a more seamless workflow integration, and also avoids the need to store local copies that would need to be periodically updated, therefore reducing maintenance issues in the system.

In the hydrology domain, various data providers are adopting web services and Application Programming Interfaces (APIs) to allow users a fast and efficient access to public datasets, such as the National River Flow Archive (NRFA) hosted by the Centre for Ecology and Hydrology in the United Kingdom. The NRFA is a primary source of information for hydrologists, modellers, researchers and practitioners operating on UK catchments. It stores several types of hydrological data and metadata: gauged daily flow and catchment mean rainfall time series as well as gauging station and catchment information. Data are typically served through the NRFA web services via a web-based graphical user interface (<http://nrfa.ceh.ac.uk/>) and, more recently, via experimental RESTful APIs. REST (Representational State Transfer) is an architectural style that uses the HyperText Transfer Protocol (HTTP) to perform operations such as accessing resources on the web via a Uniform Resource Identifier (URI). In simple terms, the location of a NRFA dataset on the web is a unique string of characters that follows a pattern. This string is assembled using the rules described in the API documentation and can be tested by typing the string in the address bar of a web browser.

This paper describes the technical implementation of the **rnrfa** package (Vitolo, 2016). The **rnrfa** package takes the complexities related to web development and data transfer away from the user, providing a set of functions to programmatically access, filter, and visualize NRFA data using simple R syntax. Although the NRFA APIs are still in their infancy and prone to further consolidation and refinement, the experimental implementation of the **rnrfa** package can be used to test these data services and provide useful feedback to the provider.

The package is in line with a Virtual Observatory approach (Beven et al., 2012) as it can be used as back-end tool to link data and models in a seamless fashion. It complements R's growing functionality in environmental web technologies (Leeper et al., 2016), amongst which are **moaa** (Chamberlain, 2015, interface to NOAA climate data API), **waterData** (Ryberg and Vecchia, 2014, interface to the U.S. Geological Survey daily hydrologic data services) and **RNCEP** (Kemp et al., 2011, interface to NASA NCEP weather data).

This paper first presents the NRFA archive, its web services and related APIs. We then illustrate the design and implementation of the **rnrfa** package, and how it can be used in synergy with existing R packages such as **shiny** (Chang et al., 2016), **leaflet** (Cheng and Xie, 2015), **rmarkdown** (Allaire et al., 2016), **DT** (Xie, 2015a), **dplyr** (Wickham and Francois, 2015) and **parallel** to generate interactive mapping applications, dynamic reports and big data analytics experiments.

## NRFA web services

The NRFA web services allow to view, filter and download data via a graphical user interface. This approach has a number of limitations. Firstly, time series of daily streamflow discharge and catchment rainfall can only be downloaded one at the time. Therefore, for large scale analyses, downloading datasets for hundreds of sites becomes a rather tedious task. Secondly, metadata can only be visualised (in table format) but not be downloaded. Metadata analyses may require copying and pasting large amounts of information introducing potential errors. Due to the above limitations, the NRFA is also accessible programmatically via a set of RESTful APIs. The API documentation is not in the public domain yet, therefore it must be considered experimental and subject to changes.

Station metadata (called *catalogue* hereafter) is available in JavaScript Object Notation (JSON) format. The catalogue contains a total of 18 attributes, which are listed in Table 1. The NRFA also provides time series of Gauged Daily Flow (*gdf*, in  $m^3/s$ ) and Catchment Mean Rainfall (*cmr*, in mm per month), formatted in an XML variant called WaterML2 (<http://www.opengeospatial.org/standards/waterml>). WaterML2 is an Open Geospatial Consortium (OGC) standard used worldwide to rigorously and unambiguously describe hydrological time series. It builds upon existing standards such as Observations & Measurements (Cox et al., 2011) for the metadata section and GML (Open Geospatial Consortium, 2013) for the observed time series. It is typically defined as a “Collection” and made up of five sections:

- The *metadata* section contains the document metadata (e.g., the generation date, the version, the generation system and the list of profiles utilised).
- The *temporalExtent* contains a description of the time period for which there are recordings (with a time stamp of start and end date).
- The *localDictionary* is a gml-based dictionary which stores the identifier (e.g., United Kingdom National River Flow Archive) and two dictionary entries: The first one describes the type of measurement (e.g., Gauged Daily Flow) with details on the variable measured (e.g., flow), units (e.g.,  $m^3/s$ ) and frequency of measurements (e.g., daily); the second entry describes the gauging site with details on ratings and their limitations.
- The *samplingFeatureMember* describes the monitoring point (e.g., vertical datum and time zone) and the station owner.
- The *observationMember* contains a set of nodes which schema is borrowed from the OGC Observation and Measurement standard. This section contains a gml-based identifier (station identification number) and additional information, such as *ObservationMetadata* (contact info, identification info, etc.), *phenomenonTime* (beginning and end of recordings), *ObservationProcess* (process type and reference). Finally the sub-section *result* contains the measurements in a gml-based format.

The nested structure of the WaterML2 files makes parsing of long time series and related metadata relatively slow and complex. In order to improve access to NRFA’s public data and metadata, we implemented a set of functions to assemble HTTP GET requests and parse XML/JSON responses from/to the catalogue and WaterML2 services using simple R syntax.

## Package availability and dependencies

The **rnrfa** package is a package designed to extend basic R functionalities to interact with the NRFA. It builds on the following packages that should be installed beforehand: **cowplot** (Wilke, 2016), **plyr** (Wickham, 2011), **httr** (Wickham, 2016a), **xml2** (Wickham and Hester, 2016), **stringr** (Wickham, 2016b), **xts** (Ryan and Ulrich, 2014), **rjson** (Couture-Beil, 2014), **ggmap** (Kahle and Wickham, 2013), **ggplot2** (Wickham, 2009), **rgdal** (Bivand et al., 2016), **sp** (Pebesma and Bivand, 2005; Bivand et al., 2013) and **parallel**<sup>1</sup>. The stable version of the package is available on the Comprehensive R Archive Network repository (CRAN; <https://CRAN.R-project.org/package=rnrfa/>) and can be downloaded and installed by typing the following command in the R console:

```
> install.packages("rnrfa")
```

The development version is available from a GitHub repository (<https://github.com/cvitolo/rnrfa>) and can be installed via **devtools** (Wickham and Chang, 2016), using the following commands:

```
> install.packages("devtools")
> devtools::install_github("cvitolo/rnrfa")
```

<sup>1</sup>In order to run the examples in this manuscript, the following packages should also be installed: **shiny**, **leaflet**, **DT**, **gridr** (Slowikowski, 2016), **knitr** (Xie, 2016, 2015b, 2014) and **rmarkdown**.

Column number	Column name	Description
1	<i>id</i>	Station identification number.
2	<i>name</i>	Name of the station.
3	<i>location</i>	Area in which the station is located.
4	<i>river</i>	Name of the river catchment.
5	<i>stationDescription</i>	General station description, containing information on weirs, ratings, etc.
6	<i>catchmentDescription</i>	Information on topography, geology, land cover, etc.
7	<i>hydrometricArea</i>	UK hydrometric area identification number, the related map is based on the Surface Water Survey designed in the 1930s and available at <a href="http://www.ceh.ac.uk/data/nrfa/hydrometry/has.html">http://www.ceh.ac.uk/data/nrfa/hydrometry/has.html</a> .
8	<i>operator</i>	UK measuring authorities, the related map is available at <a href="http://www.ceh.ac.uk/data/nrfa/hydrometry/mas.html">http://www.ceh.ac.uk/data/nrfa/hydrometry/mas.html</a> .
9	<i>haName</i>	Name of the hydrometric area.
10	<i>gridReference</i>	The Ordnance Survey grid reference number.
11	<i>stationType</i>	Type of station (e.g., flume, weir, etc.).
12	<i>catchmentArea</i>	Catchment area in $Km^2$ .
13	<i>gdfStart</i>	First year of monitoring.
14	<i>gdfEnd</i>	Last year of monitoring.
15	<i>farText</i>	Information on the regime (e.g., natural, regulated, etc.).
16	<i>categories</i>	Various tags (e.g., FEH_POOLING, FEH_QMED, HI-FLOWS_INCLUDED).
17	<i>altitude</i>	Altitude measured in metres above Ordnance Datum or, in Northern Ireland, Malin Head.
18	<i>sensitivity</i>	Sensitivity index calculated as the percentage change in flow associated with a 10 mm increase in stage at the $Q_{95}$ flow.

**Table 1:** Gauging station metadata, more detail is provided at [http://www.ceh.ac.uk/data/nrfa/data/gauging\\_stations.html](http://www.ceh.ac.uk/data/nrfa/data/gauging_stations.html).

The package is loaded using the following command:

```
> library(rnrrfa)
```

The package is fully documented and additional sample applications are available on the dedicated web page <http://cvitolo.github.io/rnrrfa/>. Feedbacks and contributions can be submitted through the GitHub issue tracking system (<https://github.com/cvitolo/rnrrfa/issues>) and pull requests (<https://github.com/cvitolo/rnrrfa/pulls>), respectively.

## Design and implementation

In many hydrological analyses the importance of efficient data retrieval is often underestimated with the consequence of allocating more time to this first task than to the data processing and analysis of results. The **rnrrfa** packages provides re-usable functions, based on a consistent syntax, that attempts to simplify data retrieval and makes it scalable to multiple data requests.

### Catalogue metadata

The full list of gauging stations is in JSON format and can be retrieved using the function `catalogue()`, used with no inputs.

```
> allStations <- catalogue()
```

This converts the information into a data frame with one row per station and 18 columns (Table 1 contains a detailed description of the attributes). The reader should note that the server response includes the Ordnance Survey (OS) grid reference, not latitude and longitude coordinates. The

`catalogue()` function converts the grid reference to latitude and longitude, then joins the coordinates to the data frame containing the list of stations.

The conversion is handled by the `osg_parse()` function which can transform OS grid references of different lengths to: a) latitude and longitude, in the WGS84<sup>2</sup> coordinate system; b) easting and northing, in the BNG<sup>3</sup> coordinate system. This function accepts two arguments: `gridRef`, a character string containing the OS grid reference, and `CoordSystem`, that can be either "WGS84" (default) or "BNG". The code below shows how to convert an example OS grid reference, "NC581062", to the two types of coordinates.

```
> # Option a: from OS grid reference to WGS84
> osg_parse(gridRef = "NC581062", CoordSystem = "WGS84")

> # Option b: from OS grid reference to BNG
> osg_parse(gridRef = "NC581062", CoordSystem = "BNG")
```

## Filtering stations

The `catalogue()` function provides 5 optional arguments that can be used to filter metadata based on various criteria. The argument `all`, for instance, is TRUE by default and forces all the metadata to be retrieved. If `all` is set to FALSE, the resulting data frame contains only the following columns: `id`, `name`, `river`, `catchmentArea`, `lat`, `lon`. This can be used, for instance, to print a concise version of the table to the screen.

At the time of writing, 1539 stations are monitored within NRFA. Very rarely the full set of stations is used. Depending on the aim of the analysis, stations might need to be filtered based on a geographical bounding box, length of the recording period, thresholds, etc. Below are some examples showing how to filter stations based on one or multiple criteria.

**Filtering based on a geographical bounding box.** Stations can be filtered based on a bounding box thanks to the NRFA web service and a specific functionality of its API. A bounding box should be defined as a list of four named elements (minimum longitude, minimum latitude, maximum longitude and maximum latitude) and passed as input to the `catalogue()` function using the argument `bbox`. The following example shows how to define a bounding box for the Plynlimon area (mid-Wales, United Kingdom), filter the related stations and map their location using the `ggmap` package. In Figure 1 the location of each station is shown as a red dot, while the name of the station is used as a label.

```
> # Define a bounding box.
> bbox <- list(lonMin = -3.76, latMin = 52.43, lonMax = -3.67, latMax = 52.48)
> # Filter stations based on bounding box.
> someStations <- catalogue(bbox)
> # Map
> library(ggmap)
> library(ggrepel)
> m <- get_map(location = as.numeric(bbox), maptype = 'terrain')
> ggmap(m) + geom_point(data = someStations, aes(x = lon, y = lat),
+ col = "red", size = 3, alpha = 0.5) +
> geom_text_repel(data = someStations, aes(x = lon, y = lat, label = name),
+ size = 3, col = "red")
```

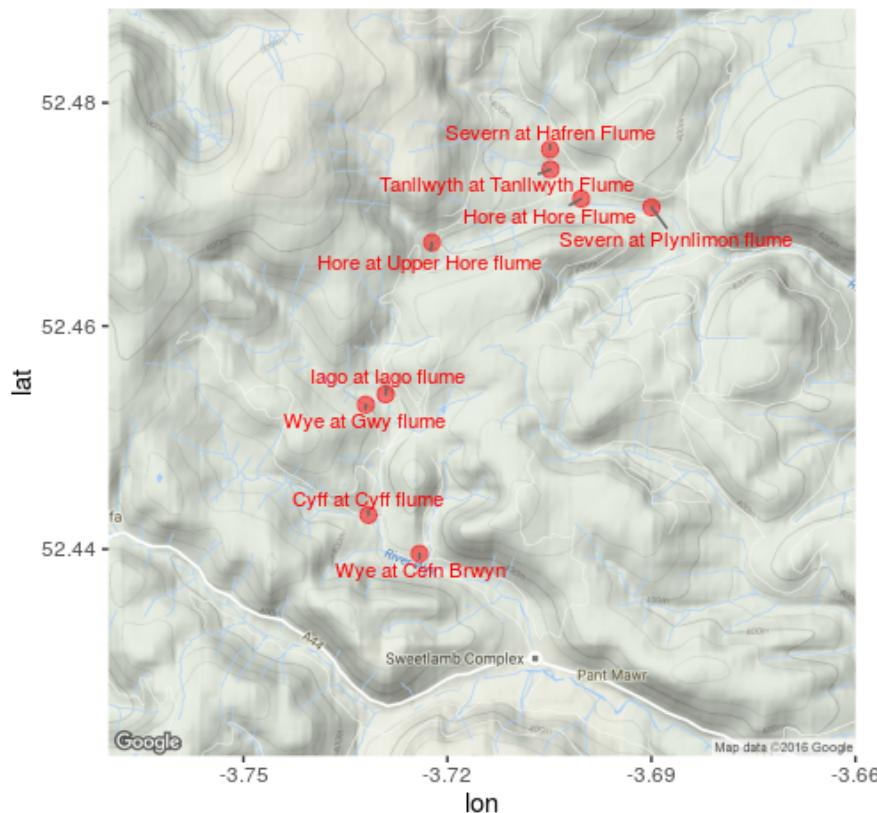
**Filtering based on recording period.** To calculate summary statistics, it is often useful to select only stations with at least  $x$  number of recording years. In the example below, we select only gauging stations with a minimum of 100 years of recordings, using the argument `minRec`. The result is a list of three stations, two of which are located in South England and one in Wales.

```
> # Select stations with more than 100 years of recordings.
> s100Y <- catalogue(minRec = 100, all = FALSE)
> # Print s100Y to the screen.
> s100Y
```

<code>id</code>	<code>name</code>	<code>river</code>	<code>catchmentArea</code>	<code>lat</code>	<code>lon</code>
-----------------	-------------------	--------------------	----------------------------	------------------	------------------

<sup>2</sup>World Geodetic System 1984, EPSG code: 4326.

<sup>3</sup>British/Irish National Grid, EPSG codes: 27700/29902.



**Figure 1:** Map of Plynlimon area with NRFA selected gauging stations (red dots).

636	38001	Lee at Feildes Weir	Lee	1036	51.76334	0.01277874
665	39001	Thames at Kingston	Thames	9948	51.41501	-0.30887638
1130	55032	Elan at Caban Dam	Elan	184	52.26907	-3.57239164

**Filtering based on metadata entries.** It is also possible to filter stations based on a number of metadata entries using the arguments: `columnName` (name of the column to filter) and `columnValue` (string or numeric value to match or compare). The function `catalogue()` looks for records containing the string `columnValue` in the column `columnName`. If `columnName` refers to a character field, the search is case sensitive and can be used to filter the stations based on the river name, catchment name, location and so on. In the example below we filter 34 stations falling within the Wye (Hereford) hydrometric area:

```
> stationsWye <- catalogue(columnName = "haName", columnValue = "Wye (Hereford)")
```

If `columnName` refers to a numeric field and `columnValue` contains special characters such as `>`, `<`,  `$\geq$`  and  `$\leq$`  followed by a number, stations are filtered using a threshold. For instance, there are 7 stations with drainage area smaller than  $1 \text{ Km}^2$ , which can be filtered using the command below:

```
> stations1KM <- catalogue(columnName = "catchmentArea", columnValue = "<1")
```

### Combined filtering

Filtering capabilities can also be combined. In the example below we filter all the stations within the above defined bounding box that belong to the Wye (Hereford) hydrometric area and have a minimum of 50 years of recordings. The only station that satisfies all the criteria is the Wye at Cefn Brwyn.

```
> catalogue(bbox, columnName = "haName", columnValue = "Wye (Hereford)",  
+           minRec = 50, all = FALSE)
```

id	name	river	catchmentArea	lat	lon
6	55008	Wye at Cefn Brwyn	Wye	10.6	52.43958 -3.724108

## WaterML2 services

Once a certain number of stations are selected, time series of gauged daily flow and catchment mean rainfall data can be obtained by requesting access to the NRFA WaterML2 service using the functions `gdf()` and `cmr()`, respectively. These functions assemble and send data requests to the WaterML2 service, parse responses and convert them to a time series object (of class from package `xts`). They use the same syntax and require the following arguments:

- `id`, the station identification numbers. This can either be a single string or a character vector.
- `metadata`, a logical variable. If set to FALSE (default), metadata are not parsed. If it is set to TRUE, the result for a single station is a list of two named elements: `data` (time series) and `meta` (metadata).

When `gdf()` and `cmr()` are executed, the assembled data request is printed to the screen. This is very useful if the user wants to understand how the API works behind the scenes, but not when incorporating the code in automated scripts. Although the NRFA API documentation is not public yet, the patterns are simple and can be easily extrapolated running a few examples.

### Get gauged daily flow

Raw flow data are typically measured in  $m^3/s$ , at 15-minute intervals. Data are first quality controlled, then the daily mean is calculated and stored in the NRFA public database. These data are typically collected for the monitoring of river networks but can also be used to calibrate hydrological models and build forecasting systems. The example below shows how to get the daily flow for the Tanllwyth at Tanllwyth Flume and the assembled data request (printed to the console).

```
> flow <- gdf(id = "54090")
```

```
http://nrfaapps.ceh.ac.uk/nrfa/xml/watertml2?db=nrfa\_public&stn=54090&dt=gdf
```

The result is a time series (of class “`xts`”). No station-specific information is stored, because the argument `metadata` is set to FALSE by default. An “`xts`” object can be easily converted into a data frame object and exported to a text file (e.g., `csv`) for use in other modelling software, as demonstrated in the example below.

```
> # Get gauged daily flow for station 54090.
> flow <- gdf(id = "54090")
> # Convert to csv.
> write.csv(as.data.frame(flow), "flowDF.csv", quote = FALSE)
```

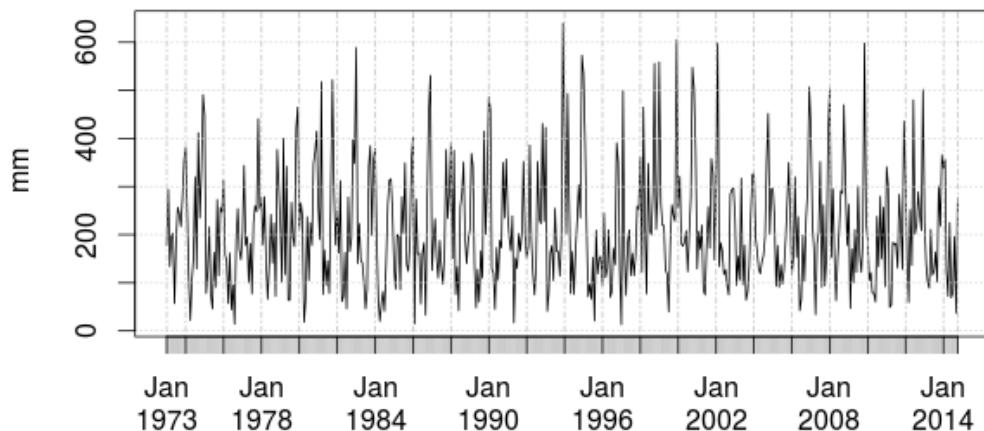
### Get catchment mean rainfall

The main forcing input in any hydrological model is rainfall. In many cases it is important to calculate the average rainfall over a catchment, this is achieved by using geospatial interpolation methods or, more simplistically, calculating the weighted average using a number of weather stations within the catchment and/or in the nearby areas. The NRFA provides pre-calculated monthly catchment mean rainfall, measured in  $mm$ , for a number of UK catchments. As the calculation is consistent across catchments, these datasets are a valuable resource to ensure reproducibility of hydrological analyses. Similar to `gdf()`, the function `cmr()` allows users to retrieve the catchment mean rainfall data by specifying the argument `id`. The example below shows that, if we set the argument `metadata` to TRUE, we can use metadata to automatically populate title and labels in a plot, as in Figure 2. The reader should note that `rain$data` is an “`xts`” object, therefore `plot(rain$data)` uses the S3 method for “`xts`”.

```
> rain <- cmr(id = "54090", metadata = TRUE)
> data <- rain$data
> meta <- rain$meta
> plot(data, main = paste(meta$variable, "-", meta$stationName),
+       xlab = "", ylab = meta$units)
```

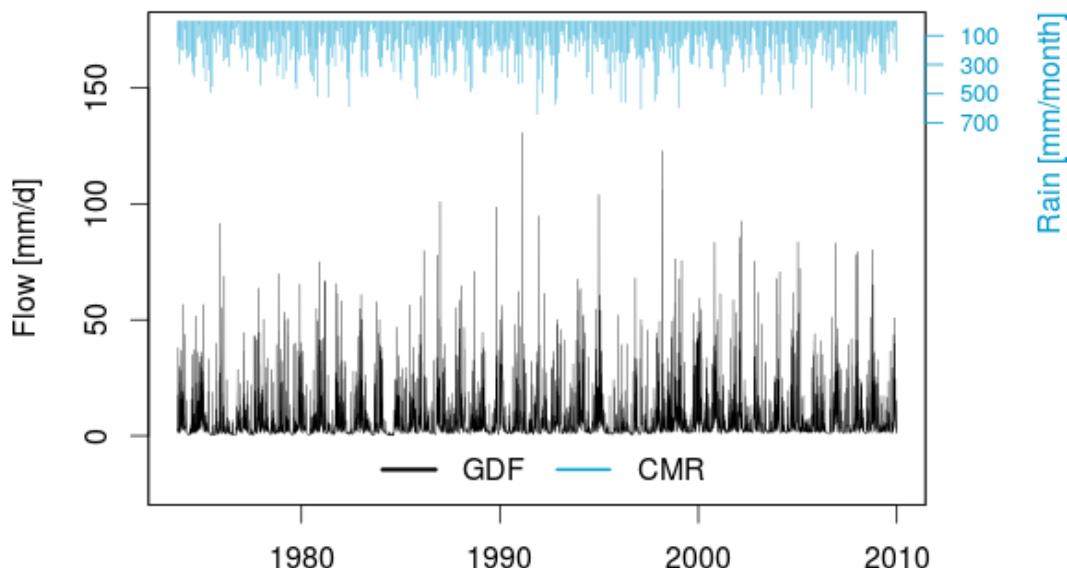
Station information consists of: the station name, location in latitude and longitude coordinates, the variable measured (i.e., rainfall), units (i.e.,  $mm$ ), aggregation function (i.e., accumulation), time step of recording (i.e., month) and time zone.

## Rainfall - Tanllwyth at Tanllwyth Flume



**Figure 2:** Monthly catchment mean rainfall for the Tanllwyth at Tanllwyth Flume catchment.

## Tanllwyth at Tanllwyth Flume



**Figure 3:** Monthly catchment mean rainfall and daily flow for the Tanllwyth at Tanllwyth Flume catchment.

### Convert and compare flow and rainfall for a given catchment

In the NRFA, flow and rainfall are stored in  $m^3/s$  and  $mm/month$ , respectively, therefore they are not directly comparable. However, given the catchment area (from the metadata catalogue), the flow can be easily converted into  $mm/day$  and then compared to the rainfall, for instance by plotting them on the same time line. Although the operations are trivial, it is a relatively lengthy procedure that can be simplified using the function `plot_rain_flow()`. This function uses the station id as input to request metadata as well as flow and rainfall time series for the given catchment, converts the flow from its original units to  $mm/day$  and then plots the converted flow and rainfall on two different  $y$ -axes so that they can be visually compared, as shown in Figure 3.

```
> plot_rain_flow(id = "54090")
```

### Multiple sites

The package `rnrrfa` is particularly useful for large scale data acquisition. If the `id` argument is a vector, the functions `gdf()` and `cmr()` can be used to sequentially fetch time series (meta)data from multiple

Test	min	lq	mean	median	uq	max	neval
a	17.598647	17.95601	18.419300	18.355630	19.037328	19.16267	10
b	3.564888	8.91512	8.411546	9.504491	9.666812	10.58291	10

**Table 2:** Benchmark tests comparing retrieval time for sequential (a) and simultaneous calls to the server (b). Results show time in seconds, obtained by averaging over 10 repetitions using the `microbenchmark` package (Mersmann, 2015).

sites. As the server can handle multiple requests, concurrent calls can be sent simultaneously using the `parallel` package. In order to send concurrent calls, a cluster object, created by the `parallel` package, should be passed to `gdf()` or `cmr()` using the argument `c1`. Below is a simple benchmark test in which we compare the processing time for collating flow time series data for the 9 stations in the Plynlimon area sending: a) 1 data request at the time and b) 9 simultaneous requests. The operations are repeated 10 times. The results are averaged and summarised in Table 2, which shows that (a) takes about 18 seconds, while (b) about 8 seconds. The reader should note that the time for retrieval does not reduce proportionally with the number of simultaneous requests because there is a limit in the number of calls the server can handle, which depends on the infrastructure and the number of incoming requests from other users.

```
> library(microbenchmark)
> library(parallel)
> cl <- makeClustergetOption("cl.cores", 9))
> microbenchmark(# sequential requests
+                 gdf(id = someStations$id, metadata = FALSE, cl = NULL),
+                 # concurrent requests
+                 gdf(id = someStations$id, metadata = FALSE, cl = cl), times = 10)
> stopCluster(cl)
```

## Some applications

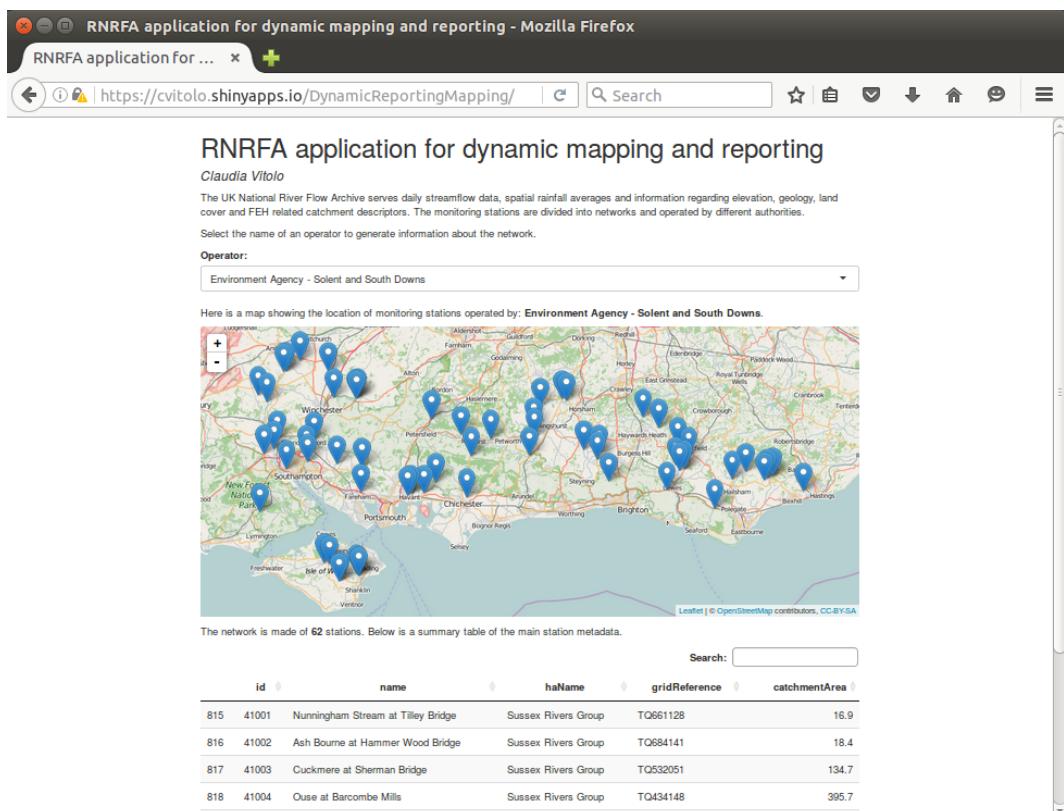
The `rnrrfa` package is an ideal building block for many scientific workflows but can also work as back-end tool for a number of web applications, from interactive mapping and dynamic reports that improve reproducibility of analysis, to the integration into more sophisticated big data analytics experiments. This can be achieved thanks to the intrinsic interoperability of the R environment. Some example applications are given in the following sections.

### Dynamic mapping and reporting application

Here we demonstrate the generation of a dynamic mapping and reporting application to summarise stations' metadata and map the spatial distribution of the monitoring network for each operator. The user can select the name of the operator using a drop-down menu and the dynamic document automatically renders an interactive map showing a marker for each station in the network on top of a background map based on OpenStreetMap. Users can zoom in/out and navigate to a specific area. Finally, the user can click on a marker to read name and station identification number from a pop-up window. Figure 4 shows a screenshot of the web application. At the bottom of the page is a dynamic table that summarises the metadata associated with the selected stations in the network. The table can be filtered using an interactive search box. The textual content also updates automatically the reporting of the number of stations within the selected network. The web application depends on the following packages: `rmarkdown`, `knitr`, `shiny`, `leaflet` and `DT` and its source code is available as gist at the following URL: <https://gist.github.com/cvitolo/d5d46b5e8f3676013857>.

### Geoprocessing based on user-defined areas

The NRFA web site does not allow users to execute geoprocessing tasks, for instance, to intersect the list of stations with user-defined or externally sourced areas. In some cases it might be of interest to explore the distribution of stations based on high-level administrative boundaries such as regions/countries. This is useful to understand whether there are differences in the reliability of the networks that can be explained by the different management approaches. Eurostat established a hierarchy of three levels of administrative divisions within each European country, called Nomenclature of Territorial Units for



**Figure 4:** RNRFA application for dynamic reporting and mapping.

Statistics (NUTS)<sup>4</sup>. At the first level, UK is divided into 12 regions: Northern Ireland, Scotland, Wales and 9 English sub-regions (East Midlands, East of England, Greater London, North East, North West, South East, South West, West Midlands, Yorkshire and the Humber). Calculating, for instance, the number/density of stations by region is not possible using the NRFA web site because the stations' metadata does not contain information on this type of administrative region and users cannot specify their own. However, these simple geoprocessing operations become relatively trivial using the **rnrfa** package.

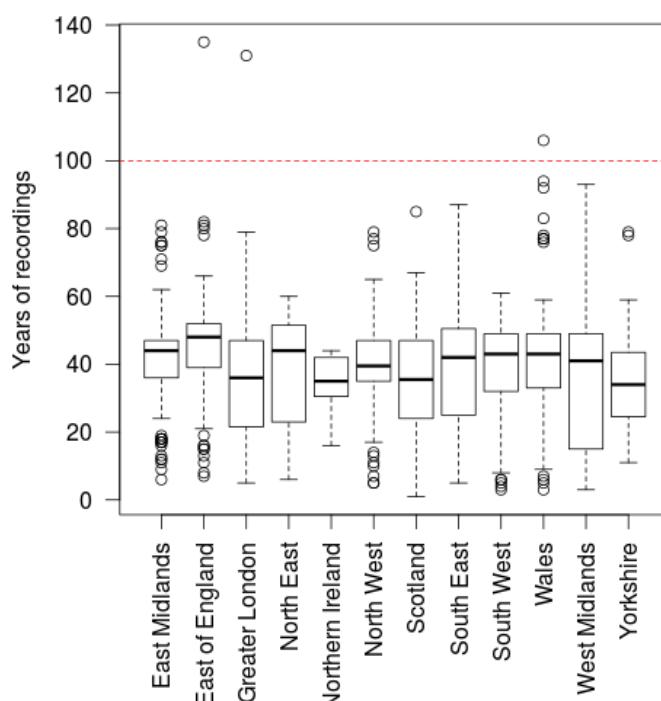
The procedure consists of five steps:

- retrieve the list of NRFA stations (using the `catalogue()` function);
- load the NUTS (level 1) shapefile and reproject the polygon to the geographic coordinate system WGS84 (using the **rgdal** and **sp** packages);
- transform the NRFA list of stations into a `SpatialPointDataFrame` ;
- spatially overlay NRFA stations (points) and NUTS1 regions (polygons);
- add a new column, containing the name of the NUTS1 regions, to the list of NRFA stations.

The updated list of stations is included, as sample dataset, in the data folder of this package, under the name `stationSummary`. Table 3 summarises the number of stations per region, the area of each region (in  $Km^2$ ), and the density of stations (number of stations/ $Km^2$ ). The metadata can now be easily summarised by NUTS1 region, for instance the boxplot in Figure 5 shows the distribution of years of recording. Northern Ireland seem to have the youngest network, with recording years in the range [16, 44]. Only three regions have stations with more than 100 years of recordings: East of England, London and Wales. Scotland and Northern Ireland have the lowest density of gauging stations, while Greater London the highest. The code to reproduce this example is available as gist at the following URL: <https://gist.github.com/cvitolo/aa3bc6f08a8394f653442e276568f9b3>.

<sup>4</sup>The Nomenclature of Territorial Units for Statistics (NUTS), is a standard for referencing the administrative divisions of European countries. There are three levels of NUTS and a shapefile is available from the Eurostat web site ([http://ec.europa.eu/eurostat/cache/GISCO/geodatafiles/NUTS\\_2013\\_01M\\_SH.zip](http://ec.europa.eu/eurostat/cache/GISCO/geodatafiles/NUTS_2013_01M_SH.zip)).

NUTS_ID	Region	# stations	Area (Km <sup>2</sup> )	Density
1	UKC	North East (England)	54.00	8601.77
2	UKD	North West (England)	137.00	14170.34
3	UKE	Yorkshire and the Humber	102.00	15418.70
4	UKF	East Midlands	101.00	15637.21
5	UKG	West Midlands	103.00	12999.97
6	UKH	East of England	149.00	19159.91
7	UKI	Greater London	36.00	1575.97
8	UKJ	South East (England)	169.00	19105.67
9	UKK	South West (England)	176.00	23912.24
10	UKL	Wales	132.00	20817.37
11	UKM	Scotland	324.00	78984.40
12	UKN	Northern Ireland	56.00	14175.46

**Table 3:** Summary of number of stations per NUTS1 region, area of each region and density of stations.**Figure 5:** Distribution of recording years for NRFA stations by NUTS1 regions.

### Big data analytics experiment

In the last few years, the UK MetOffice has reported “unusual warmth and lack of rainfall during March and April, particularly over England and Wales”<sup>5</sup>. Dry springs can affect water resources, because river flow below average translates, for instance, in reduced availability of drinking water. In this section we present a big data analytics experiment in which we try to understand if there is any evidence, in the NRFA data, that springs in the UK are becoming drier, both in terms of rainfall and river flow. This type of experiment consists of retrieving all the available rainfall and flow time series and find out, for each station, whether there is an increasing/decreasing trend.

Using the NRFA web site, the comparison of time series is only feasible for a limited number of sites. Time series should be first downloaded as text files and then compared manually. The biggest advantage of using the `rnrrfa` package, instead, is that multiple downloads can be automated using a single line of code.

In this experiment we used a cluster of 64 cores to download and analyse all the time series available from the NRFA stations with more than 10 years of recordings. The time series were first

<sup>5</sup>[http://www.metoffice.gov.uk/climate/uk/interesting/2011\\_spring](http://www.metoffice.gov.uk/climate/uk/interesting/2011_spring)



**Figure 6:** Map and boxplot of rainfall trend during spring.

downloaded, then summarised in terms of annual averages over the spring period. Seasonal averages can be calculated using the function `seasonal_averages()`, which takes as input a time series and a period of interest (e.g., spring) and calculates the related annual average. Using a very simplistic approach, a linear model was fit to the annual averages and the slope coefficient was used to estimate the trend. Negative slopes correspond to decreasing flow/rainfall, while positive slopes correspond to an increase of flow/rainfall over time. Once the fitted slope is calculated for each station, the results can be plotted using the function `plot_trend()`. Figures 6 and 7 show only the statistically significant trends for rainfall and flow respectively. Each figure is divided into two plots: Plot A shows the spatial distribution of negative trends with red dots and positive trends with blue dots; plot B shows the variability of trends over NUTS1 regions. In the latter plot, outliers are removed by showing only values between the 5th and 95th quantiles. From a meteorological perspective (Figure 6), there are only positive statistically significant trends and Scotland shows the largest. In terms of hydrological responses (Figure 7), trends are more subtle as the interquartile range is concentrated around zero. The most extreme negative trends were found in Scotland and North East England.

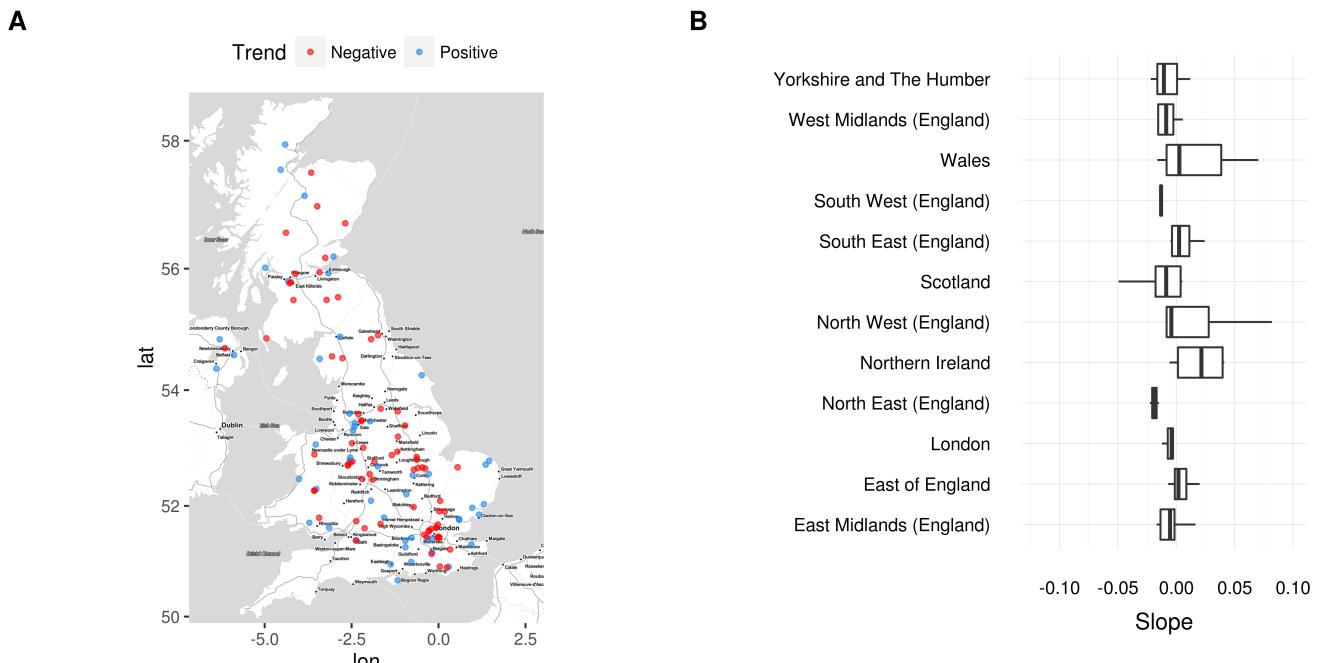
The entire run took about 31 minutes, the code to reproduce this example is available as gist at the following URL: <https://gist.github.com/cvitolo/612eb2ae9b47fe8f11a1ed8d06e3b434>. There are certainly more rigorous methodologies to estimate seasonal trends. This experiment was just an attempt to demonstrate that the `rnrfa` can simplify large scale data acquisition tasks.

## A note on package usage

The `cranlogs` (Csardi, 2015) package provides an API interface to download logs from the RStudio CRAN mirror which contains download counts from unique IP addresses and can be used as a proxy to estimate the volume of package users. By September 2016, the `rnrfa` package had been downloaded from CRAN 6372 times, just from this mirror, following a trend very similar to the `waterData` package (see Figure 8). Because the RStudio mirror is located in the US, it is expected that the download counts from UK mirrors could be even higher. We derive that this package is of interest for a large community of users, which gives us scope for future developments.

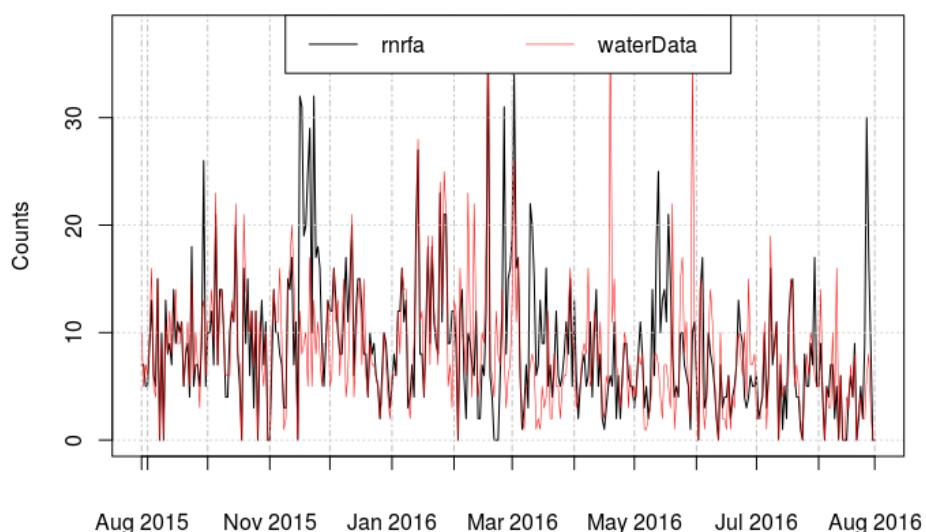
## Summary

This article describes the `rnrfa` package for interacting programmatically with the UK National River Flow Archive. It allows to access web resources such as the catalogue of stations' metadata and the WaterML2 service to retrieve gauged daily flow and (monthly) catchment mean rainfall. The package provides functions to query the catalogue based on various criteria (e.g., geographical bounding



**Figure 7:** Map and boxplot of flow trend during spring.

#### Download counts from RStudio CRAN mirror



**Figure 8:** Comparison between **rnrfa** and **waterData** download counts from independent IP addresses.

box, minimum number of recording years, river catchment/hydrometric area/operators amongst many other options), retrieve and visualise flow and rainfall time series, convert coordinates and flow measurements, and plot basic seasonal trends grouped on user defined regions. Some of these capabilities are strongly linked to the particular content of the NRFA database and are not directly transferable/applicable to other data sources. However the `gdf()` and `cmr()` functions could be re-used, with minimal changes, to get data/metadata from other providers adopting the WaterML2 standard.

The package is a convenient standalone application that allows NRFA users a more efficient access to the public database, compared to the web interface, e.g., the possibility to efficiently retrieve data from multiple sites. The **rnrfa** package can also be used as back-end tool for web applications. Amongst the existing R interfaces to data APIs, **rnrfa** follows a logic similar to **waterData**: Sites are first identified through a catalogue, streamflow data are imported via the station identification number,

then data are visualised and/or used in analyses. However, our package does not implement any function for data cleanup, because NRFA data are highly quality controlled. Users can currently take advantage of other packages such as `xts` to calculate aggregate variables, `evd` (Stephenson, 2002) for the analysis of extreme events, `outliers` (Komsta, 2011) to identify possible outliers and `sp` and `spacetime` (Pebesma, 2012; Bivand et al., 2013) for more advanced spatio-temporal processing.

In the future, we plan to implement additional processing functions (e.g., to compare gdf with flow in bankfull condition which is highly important for flood frequency estimations). Further developments are also scheduled on the NRFA side to include Web Feature Service (WFS), Sensor Observation Services (SOS) and updates to WaterML2 OGC standards. WFS layers can already be loaded and manipulated using `rgdal` (Bivand et al., 2016), while `sos4R` (Nüst et al., 2011) can be used as client for SOS.

## Acknowledgments

This work was carried out when Claudia Vitolo was working at the Imperial College London and supported by the Natural Environment Research Council pilot Probability, Uncertainty & Risk in the Environment (PURE) NE/I004017/1. Comments from two referees are gratefully acknowledged.

## Bibliography

- J. Allaire, J. Cheng, Y. Xie, J. McPherson, W. Chang, J. Allen, H. Wickham, A. Atkins, and R. Hyndman. *rmarkdown: Dynamic Documents for R*, 2016. URL <https://CRAN.R-project.org/package=rmarkdown>. R package version 0.9.2. [p102]
- K. Beven, W. Buytaert, and L. Smith. On virtual observatories and modelled realities (or why discharge must be treated as a virtual variable). *Hydrological Processes*, 26:1906–1909, 2012. doi: 10.1002/hyp.9261. [p102]
- R. Bivand, T. Keitt, and B. Rowlingson. *rgdal: Bindings for the Geospatial Data Abstraction Library*, 2016. URL <https://CRAN.R-project.org/package=rgdal>. R package version 1.1-10. [p103, 114]
- R. S. Bivand, E. Pebesma, and V. Gomez-Rubio. *Applied Spatial Data Analysis with R*. Springer-Verlag, 2nd edition, 2013. doi: 10.1007/978-1-4614-7618-4. [p103, 114]
- S. Chamberlain. *rnoaa: 'NOAA' Weather Data from R*, 2015. URL <https://CRAN.R-project.org/package=rnoaa>. R package version 0.5.0. [p102]
- W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2016. URL <https://CRAN.R-project.org/package=shiny>. R package version 0.13.2. [p102]
- J. Cheng and Y. Xie. *leaflet: Create Interactive Web Maps with the JavaScript "Leaflet" Library*, 2015. URL <https://CRAN.R-project.org/package=leaflet>. R package version 1.0.0. [p102]
- A. Couture-Beil. *rjson: JSON for R*, 2014. URL <https://CRAN.R-project.org/package=rjson>. R package version 0.2.15. [p103]
- S. Cox et al. Observations and measurements – XML implementation. *OGC document*, 2011. [p103]
- G. Csardi. *cranlogs: Download Logs from the "RStudio" CRAN Mirror*, 2015. URL <https://CRAN.R-project.org/package=cranlogs>. R package version 2.1.0. [p112]
- D. Kahle and H. Wickham. *ggmap: Spatial visualization with ggplot2*. *The R Journal*, 5(1):144–161, 2013. URL <https://journal.R-project.org/archive/2013-1/kahle-wickham.pdf>. [p103]
- M. U. Kemp, E. E. van Loon, J. Shamoun-Baranes, and W. Bouten. RNCEP: Global weather and climate data at your fingertips. *Methods in Ecology and Evolution*, 3(1):65–70, 2011. doi: 10.1111/j.2041-210x.2011.00138.x. [p102]
- L. Komsta. *outliers: Tests for Outliers*, 2011. URL <https://CRAN.R-project.org/package=outliers>. R package version 0.14. [p114]
- T. Leeper, S. Chamberlain, P. Mair, K. Ram, and C. Gandrud. CRAN Task View: Web technologies and services, 2016. URL <https://CRAN.R-project.org/view=WebTechnologies>. Version 2016-08-18. [p102]

- O. Mersmann. *microbenchmark: Accurate Timing Functions*, 2015. URL <https://CRAN.R-project.org/package=microbenchmark>. R package version 1.4-2.1. [p109]
- D. Nüst, C. Stasch, and E. J. Pebesma. Connecting R to the sensor web. Lecture Notes in Geoinformation and Cartography, pages 227–246. Springer, 2011. [p114]
- Open Geospatial Consortium. OpenGIS geography markup language (GML) encoding standard. Technical report, 2013. URL <http://www.opengeospatial.org/standards/gml>. [p103]
- E. Pebesma. spacetemp: Spatio-temporal data in R. *Journal of Statistical Software*, 51(7):1–30, 2012. doi: 10.18637/jss.v051.i07. [p114]
- E. J. Pebesma and R. S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, 2005. URL [https://CRAN.R-project.org/doc/Rnews/Rnews\\_2005-2.pdf](https://CRAN.R-project.org/doc/Rnews/Rnews_2005-2.pdf). [p103]
- J. A. Ryan and J. M. Ulrich. *xts: eXtensible Time Series*, 2014. URL <https://CRAN.R-project.org/package=xts>. R package version 0.9-7. [p103]
- K. R. Ryberg and A. V. Vecchia. *waterData: An R Package for Retrieval, Analysis, and Anomaly Calculation of Daily Hydrologic Time Series Data*, 2014. URL <https://CRAN.R-project.org/package=waterData>. R package version 1.0.4. [p102]
- K. Slowikowski. *ggrepel: Repulsive Text and Label Geoms for “ggplot2”*, 2016. URL <https://CRAN.R-project.org/package=ggrepel>. R package version 0.5. [p103]
- A. G. Stephenson. *evd: Extreme value distributions*. *R News*, 2(2):31–32, 2002. URL [https://CRAN.R-project.org/doc/Rnews/Rnews\\_2002-2.pdf](https://CRAN.R-project.org/doc/Rnews/Rnews_2002-2.pdf). [p114]
- C. Vitolo. *rnrfa: UK National River Flow Archive Data from R*, 2016. URL <https://CRAN.R-project.org/package=rnrfa>. R package version 1.3.0. [p102]
- C. Vitolo, Y. Elkhattib, D. Reusser, C. J. A. Macleod, and W. Buytaert. Web technologies for environmental big data. *Environmental Modelling & Software*, 63:185–198, 2015. doi: 10.1016/j.envsoft.2014.10.007. [p102]
- H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, 2009. URL <http://ggplot2.org>. [p103]
- H. Wickham. The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40(1): 1–29, 2011. doi: 10.18637/jss.v040.i01. [p103]
- H. Wickham. *httr: Tools for Working with URLs and HTTP*, 2016a. URL <https://CRAN.R-project.org/package=httr>. R package version 1.1.0. [p103]
- H. Wickham. *stringr: Simple, Consistent Wrappers for Common String Operations*, 2016b. URL <https://CRAN.R-project.org/package=stringr>. R package version 1.1.0. [p103]
- H. Wickham and W. Chang. *devtools: Tools to Make Developing R Packages Easier*, 2016. URL <https://CRAN.R-project.org/package=devtools>. R package version 1.12.0. [p103]
- H. Wickham and R. Francois. *dplyr: A Grammar of Data Manipulation*, 2015. URL <https://CRAN.R-project.org/package=dplyr>. R package version 0.4.3. [p102]
- H. Wickham and J. Hester. *xml2: Parse XML*, 2016. URL <https://CRAN.R-project.org/package=xml2>. R package version 1.0.0. [p103]
- C. O. Wilke. *cowplot: Streamlined Plot Theme and Plot Annotations for “ggplot2”*, 2016. URL <https://CRAN.R-project.org/package=cowplot>. R package version 0.6.2. [p103]
- Y. Xie. *Implementing Reproducible Computational Research*, chapter knitr: A Comprehensive Tool for Reproducible Research in R. Chapman and Hall/CRC, 2014. [p103]
- Y. Xie. *DT: A Wrapper of the JavaScript Library “DataTables”*, 2015a. URL <https://CRAN.R-project.org/package=DT>. R package version 0.1. [p102]
- Y. Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, 2nd edition, 2015b. ISBN 978-1498716963. [p103]
- Y. Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*, 2016. URL <https://CRAN.R-project.org/package=knitr>. R package version 1.14. [p103]

*Claudia Vitolo*  
*Forecast Department*  
*European Centre for Medium-range Weather Forecast*  
*Reading*  
*United Kingdom*  
[claudia.vitolo@ecmwf.int](mailto:claudia.vitolo@ecmwf.int)

*Matthew Fry*  
*Centre for Ecology and Hydrology*  
*Wallingford*  
*United Kingdom*  
[mfry@ceh.ac.uk](mailto:mfry@ceh.ac.uk)

*Wouter Buytaert*  
*Department of Civil and Environmental Engineering*  
*Imperial College London*  
*United Kingdom*  
[w.buytaert@imperial.ac.uk](mailto:w.buytaert@imperial.ac.uk)

# Qtools: A Collection of Models and Tools for Quantile Inference

by Marco Geraci

**Abstract** Quantiles play a fundamental role in statistics. The quantile function defines the distribution of a random variable and, thus, provides a way to describe the data that is specular but equivalent to that given by the corresponding cumulative distribution function. There are many advantages in working with quantiles, starting from their properties. The renewed interest in their usage seen in the last years is due to the theoretical, methodological, and software contributions that have broadened their applicability. This paper presents the R package **Qtools**, a collection of utilities for unconditional and conditional quantiles.

## Introduction

Quantiles have a long history in applied statistics, especially the median. The analysis of astronomical data by Galileo Galilei in 1632 (Hald, 2003, p.149) and geodic measurements by Roger Boscovich in 1757 (Koenker, 2005, p.2) are presumably the earliest examples of application of the least absolute deviation ( $L_1$ ) estimator in its, respectively, unconditional and conditional forms. The theoretical studies on quantiles of continuous random variables started to appear in the statistical literature of the 20th century. In the case of discrete data, studies have somewhat lagged behind most probably because of the analytical drawbacks surrounding the discontinuities that characterise discrete quantile functions. Some forms of approximation to continuity have been recently proposed to study the large sample behavior of quantile estimators. For example, Ma et al. (2011) have demonstrated the asymptotic normality of unconditional sample quantiles based on the definition of the mid-distribution function (Parzen, 2004). Machado and Santos Silva (2005) proposed inferential approaches to the estimation of conditional quantiles for counts based on data jittering.

Functions implementing quantile methods can be found in common statistical software. A considerable number of R packages that provide such functions are available on the Comprehensive R Archive Network (CRAN). The base package **stats** contains basic functions to estimate sample quantiles or compute quantiles of common parametric distributions. The **quantreg** package (Koenker, 2013) is arguably a benchmark for distribution-free estimation of linear quantile regression models, as well as the base for other packages which make use of linear programming (LP) algorithms (Koenker and D’Orey, 1987; Koenker and Park, 1996). Other contributions to the modelling of conditional quantile functions include packages for Bayesian regression, e.g. **bayesQR** (Benoit et al., 2014) and **BSSquare** (Smith and Reich, 2013), and the **lqmm** package (Geraci and Bottai, 2014; Geraci, 2014) for random-effects regression.

The focus of this paper is on the R package **Qtools**, a collection of models and tools for quantile inference. These include commands for

- quantile-based analysis of the location, scale and shape of a distribution;
- transformation-based quantile regression;
- goodness of fit and restricted quantile regression;
- quantile regression for discrete data;
- quantile-based multiple imputation.

The emphasis will be put on the first two topics listed above as they represent the main contribution of the package, while a short description of the other topics is given for completeness.

## Unconditional quantiles

### Definition and estimation of quantiles

Let  $Y$  be a random variable with cumulative distribution function (CDF)  $F_Y$  and support  $S_Y$ . The CDF calculated at  $y \in S_Y$  returns the probability  $F_Y(y) \equiv p = \Pr(Y \leq y)$ . The quantile function (QF) is defined as  $Q(p) = \inf_y \{F_Y(y) \geq p\}$ , for  $0 < p < 1$ . (Some authors consider  $0 \leq p \leq 1$ . For practical purposes, it is simpler to exclude the endpoints 0 and 1.) When  $F_Y$  is continuous and strictly monotone (hence,  $f_Y(y) \equiv F'_Y(y) > 0$  for all  $y \in S_Y$ ), the quantile function is simply the inverse of  $F_Y$ . In other cases, the quantile  $p$  is defined, by convention, as the smallest value  $y$  such that  $F_Y(y)$  is at least  $p$ .

Quantiles enjoy a number of properties. An excellent overview is given by Gilchrist (2000). In particular, the **Q-transformation rule** (Gilchrist, 2000) or equivariance to monotone transformations states that if  $h(\cdot)$  is a non-decreasing function on  $\mathbb{R}$ , then  $Q_{h(Y)}(p) = h\{Q_Y(p)\}$ . Hence  $Q_Y(p) = h^{-1}\{Q_{h(Y)}(p)\}$ . Note that this property does not generally hold for the expected value.

Sample quantiles for a random variable  $Y$  can be calculated in a number of ways, depending on how they are defined (Hyndman and Fan, 1996). For example, the function `quantile()` in the base package `stats` provides nine different sample quantile estimators, which are based on the sample order statistics or the inverse of the empirical CDF. These estimators are distribution-free as they do not depend on any parametric assumption about  $F$  (or  $Q$ ).

Let  $Y_1, Y_2, \dots, Y_n$  be a sample of  $n$  independent and identically distributed (iid) observations from the population  $F_Y$ . Let  $\xi_p$  denote the  $p$ th population quantile and  $\hat{\xi}_p$  the corresponding sample quantile. (The subscripts will be dropped occasionally to ease notation, e.g.  $F$  will be used in place of  $F_Y$  or  $\xi$  in place of  $\xi_p$ .) In the continuous case, it is well known that  $\sqrt{n}(\hat{\xi}_p - \xi_p)$  is approximately normal with mean zero and variance

$$\omega^2 = \frac{p(1-p)}{\{f_Y(\xi_p)\}^2}. \quad (1)$$

A more general result is obtained when the  $Y_i$ 's,  $i = 1, \dots, n$ , are independent but not identically distributed (*nid*). The density evaluated at the  $p$ th quantile,  $f(\xi_p)$ , is called the *density-quantile function* by Parzen (1979). Its reciprocal,  $s(p) \equiv 1/f(\xi_p)$ , is called the *sparsity function* (Tukey, 1965) or *quantile-density function* (Parzen, 1979).

As mentioned previously, the discontinuities of  $F_Y$  when  $Y$  is discrete represent a mathematical inconvenience. Ma et al. (2011) derived the asymptotic distribution of the sample mid-quantiles, that is, the sample quantiles based on the mid-distribution function (mid-CDF). The latter is defined as  $F_Y^{mid}(y) = F_Y(y) - 0.5p_Y(y)$ , where  $p_Y(y)$  denotes the probability mass function (Parzen, 2004). In particular, they showed that, as  $n$  becomes large,  $\sqrt{n}(\hat{\xi}_p^{mid} - \xi_p)$  is approximately normal with mean 0. Under iid assumptions, the expression for the sampling variance is similar to that in (1); see Ma et al. (2011) for details.

The package `Qtools` provides the functions `midecdf()` and `midquantile()`, which return objects of class "midecdf" or "midquantile", respectively, containing: the values or the probabilities at which mid-cumulative probabilities or mid-quantiles are calculated (`x`), the mid-cumulative probabilities or the mid-quantiles (`y`), and the functions that linearly interpolate those coordinates (`fn`). An example is shown below using data simulated from a Poisson distribution.

```
> library("Qtools")
> set.seed(467)
> y <- rpois(1000, 4)
> pmid <- midecdf(y)
> xmid <- midquantile(y, probs = pmid$y)
> pmid

Empirical mid-ECDF
Call:
midecdf(x = y)

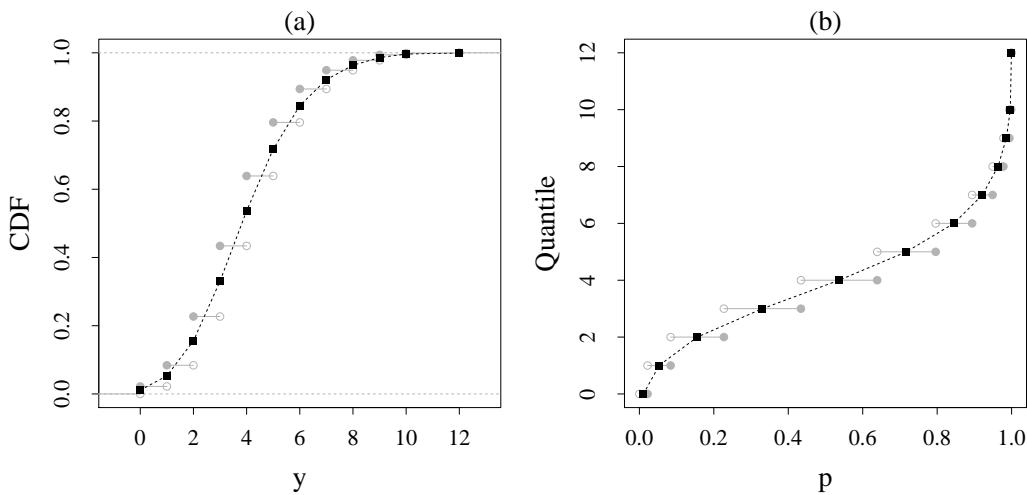
> xmid

Empirical mid-ECDF
Call:
midquantile(x = y, probs = pmid$y)
```

A confidence interval for sample mid-quantiles can be obtained using `confint.midquantile()`. This function is applied to the output of `midquantile()` and returns an object of class "data.frame" containing sample mid-quantiles, lower and upper bounds of the confidence intervals of a given level (95% by default), along with standard errors as an attribute named `stderr`. This is shown below using the sample `y` generated in the previous example.

```
> xmid <- midquantile(y, probs = 1:3/4)
> x <- confint(xmid, level = 0.95)
> x

  midquantile      lower      upper
1          1.0  0.9500000  1.0500000
2          2.0  1.9500000  2.0500000
3          3.0  2.9500000  3.0500000
4          4.0  3.9500000  4.0500000
```



**Figure 1:** Cumulative distribution (a) and quantile (b) functions for simulated Poisson data. The ordinary cumulative distribution function (CDF) and quantile function (QF) are represented by step-functions (grey lines), with the convention that, at the point of discontinuity or *jump*, the function takes its value corresponding to the ordinate of the filled circle as opposed to that of the hollow circle. The mid-CDF and mid-QF are represented by filled squares, while the piecewise linear functions (dashed lines) connecting the squares represent continuous versions of, respectively, the ordinary CDF and QF.

```
25%    2.540000 2.416462 2.663538
50%    3.822816 3.693724 3.951907
75%    5.254902 5.072858 5.436946
```

```
> attr(x, "stderr")
[1] 0.06295447 0.06578432 0.09276875
```

Finally, a plot method is available for both `midecdf()` and `midquantile()` objects. An illustration is given in Figure 1. The mid-distribution and mid-quantile functions are discrete and their values are marked by filled squares. The piecewise linear functions connecting the filled squares represent continuous versions of the CDF and QF which interpolate between the steps of, respectively, the ordinary CDF and quantile functions. Note that the argument `jumps` is a logical value indicating whether values at jumps should be marked.

```
> par(mfrow = c(1,2))
> plot(pmid, xlab = "y", ylab = "CDF", jumps = TRUE)
> points(pmid$x, pmid$y, pch = 15)
> plot(xmid, xlab = "p", ylab = "Quantile", jumps = TRUE)
> points(xmid$x, xmid$y, pch = 15)
```

### LSS - Location, scale and shape of a distribution

Since the cumulative distribution and quantile functions are two sides of the same coin, the location, scale, and shape (LSS) of a distribution can be examined using one or the other. Well-known quantile-based measures of location and scale are the median and inter-quartile range (IQR), respectively. Similarly, there are also a number of quantile-based measures for skewness and kurtosis (Groeneveld and Meeden, 1984; Groeneveld, 1998; Jones et al., 2011).

Define the *central* portion of the distribution as that delimited by the quantiles  $Q(p)$  and  $Q(1-p)$ ,  $0 < p < 0.5$ , and define the *tail* portion as that lying outside these quantiles. Let  $IPR(p) = Q(1-p) - Q(p)$  denote the inter-quantile range at level  $p$ . Building on the results by Horn (1983) and Ruppert (1987), Staudte (2014) considered the following identity:

$$\underbrace{\frac{IPR(p)}{IPR(r)}}_{\text{kurtosis}} = \underbrace{\frac{IPR(p)}{IPR(q)}}_{\text{tail-weight}} \cdot \underbrace{\frac{IPR(q)}{IPR(r)}}_{\text{peakedness}}, \quad (2)$$

where  $0 < p < q < r < 0.5$ . These quantile-based measures of shape are sign, location and scale invariant. As compared to moment-based indices, they are also more robust to outliers and easier to interpret (Groeneveld, 1998; Jones et al., 2011).

It is easy to verify that a quantile function can be written as

$$Q(p) = \underbrace{Q(0.5)}_{\text{median}} + \frac{1}{2} \underbrace{IQR(0.25)}_{\text{IQR}} \cdot \underbrace{\frac{IPR(p)}{IPR(0.25)}}_{\text{shape index}} \cdot \left( \underbrace{\frac{Q(p) + Q(1-p) - 2Q(0.5)}{IPR(p)}}_{\text{skewness index}} - 1 \right). \quad (3)$$

This identity establishes a relationship between the location (median), scale (IQR) and shape of a distribution. (This identity appears in Gilchrist (2000, p.74) with an error of sign. See also Benjamini and Krieger (1996, eq.1).) The quantity  $IPR(p)/IPR(0.25)$  in (3) is loosely defined as the *shape index* (Gilchrist, 2000, p.72), although it can be seen as the tail-weight measure given in (2) when  $p < 0.25$ . For symmetric distributions, the contribution of the skewness index vanishes. Note that the skewness index not only is location and scale invariant, but is also bounded between  $-1$  and  $1$  (as opposed to the Pearson's third standardised moment which can be infinite or even undefined). When this index is near the bounds  $-1$  or  $1$ , then  $Q(1-p) \approx Q(0.5)$  or  $Q(p) \approx Q(0.5)$ , respectively.

The function `qlss()` provides a quantile-based LSS summary with the indices defined in (3) of either a theoretical or an empirical distribution. It returns an object of class "qlss", which is a list containing measures of location (median), scale (IQR and IPR), and shape (skewness and shape indices) for each of the probabilities specified in the argument `probs` (by default, `probs = 0.1`). The quantile-based LSS summary of the normal distribution is given in the example below for  $p = 0.1$ . The argument `fun` can take any quantile function whose probability argument is named 'p' (this is the case for many standard quantile functions in R, e.g. `qt()`, `qchisq()`, `qf()`, etc.).

```
> qlss(fun = "qnorm", probs = 0.1)

call:
qlss.default(fun = "qnorm", probs = 0.1)

Unconditional Quantile-Based Location, Scale, and Shape

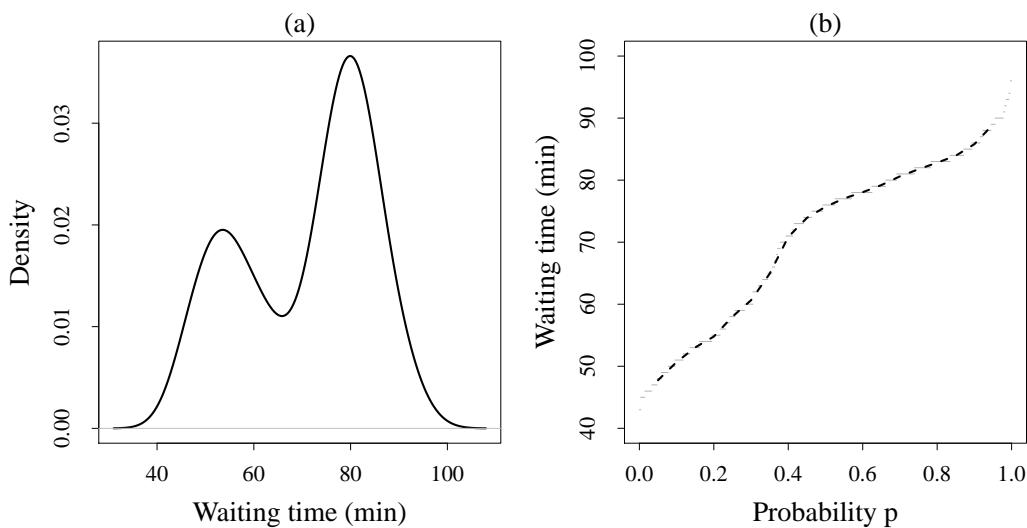
** Location **
Median
[1] 0
** Scale **
Inter-quartile range (IQR)
[1] 1.34898
Inter-quartile range (IPR)
[1] 0.1
2.563103
** Shape **
Skewness index
[1] 0
Shape index
[1] 0.1
1.900031
```

An empirical example is now illustrated using the `faithful` data set, which contains 272 observations on waiting time (minutes) between eruptions and the duration (minutes) of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA. Summary statistics are given in Table 1.

	Minimum	Q1	Q2	Q3	Maximum
Waiting time	43.0	58.0	76.0	82.0	96.0
Duration	1.6	2.2	4.0	4.5	5.1

**Table 1:** Minimum, maximum and three quartiles (Q1, Q2, Q3) for waiting time and duration in the Old Faithful Geyser data set.

Suppose the interest is in describing the distribution of waiting times. The density is plotted in Figure 2, along with the mid-quantile function. The distribution is bimodal with peaks at around 54



**Figure 2:** Estimated density (a) and empirical mid-quantile (b) functions of waiting time between eruptions in the Old Faithful Geyser data set.

and 80 minutes. Note that the arguments of the base function `quantile()`, including the argument `type`, can be passed on to `qlss()`.

```
> y <- faithful$waiting
> par(mfrow = c(1,2))
> plot(density(y))
> plot(midquantile(y, probs = p), jumps = FALSE)
> qlss(y, probs = c(0.05, 0.1, 0.25), type = 7)

call:
qlss.numeric(x = y, probs = c(0.05, 0.1, 0.25), type = 7)

Unconditional Quantile-Based Location, Scale, and Shape

** Location **
Median
[1] 76
** Scale **
Inter-quartile range (IQR)
[1] 24
Inter-quartile range (IPR)
0.05 0.1 0.25
41 35 24
** Shape **
Skewness index
      0.05      0.1      0.25
-0.3658537 -0.4285714 -0.5000000
Shape index
      0.05      0.1      0.25
1.708333 1.458333 1.000000
```

At  $p = 0.1$ , the skewness index is approximately  $-0.43$ , which denotes a rather strong left asymmetry. As for the shape index, which is equal to  $1.46$ , one could say that the tails of this distribution weigh less than those of a normal distribution ( $1.90$ ), though of course a comparison between unimodal and bimodal distributions is not meaningful.

## Conditional quantiles

### Linear models

In general, the  $p$ th linear QR model is of the form

$$Q_{Y|X}(p) = \mathbf{x}^\top \boldsymbol{\beta}(p) \quad (4)$$

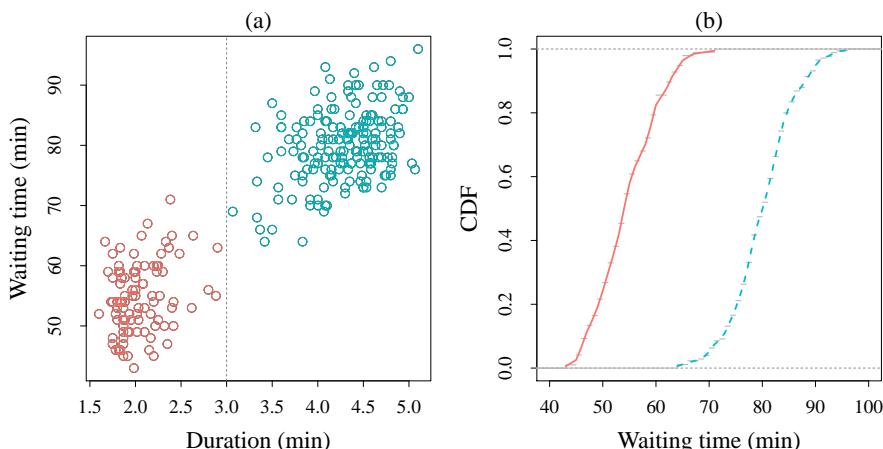
where  $\mathbf{x}$  is a  $k$ -dimensional vector of covariates (including 1 as first element) and  $\boldsymbol{\beta}(p) = [\beta_0(p), \beta_1(p), \dots, \beta_{k-1}(p)]^\top$  is a vector of coefficients. The slopes  $\beta_j(p)$ ,  $j = 1, \dots, k-1$ , have the usual interpretation of partial derivatives. For example, in case of the simple model  $Q_{Y|X}(p) = \beta_0(p) + \beta_1(p)x$ , one obtains

$$\frac{\partial Q_{Y|X}(p)}{\partial x} = \beta_1(p).$$

If  $x$  is a dummy variable, then  $\beta_1(p) = Q_{Y|X=1}(p) - Q_{Y|X=0}(p)$ , i.e. the so-called *quantile treatment effect* (Doksum, 1974; Lehmann, 1975; Koenker and Xiao, 2002). Estimation can be carried out using LP algorithms which, given a sample  $(\mathbf{x}_i, y_i)$ ,  $i = 1, \dots, n$ , solve

$$\min_{\mathbf{b} \in \mathbb{R}^k} \sum_{i=1}^n \kappa_p(y_i - \mathbf{x}_i^\top \mathbf{b}),$$

where  $\kappa_p(u) = u(p - I(u < 0))$ ,  $0 < p < 1$ , is the check loss function. Large- $n$  approximation of standard errors can be obtained from the sampling distribution of the linear quantile estimators (Koenker and Bassett, 1978).



**Figure 3:** (a) Waiting times between eruptions against durations of eruptions (dashed vertical line drawn at 3 minutes) in the Old Faithful Geyser data set. (b) Mid-CDF of waiting time by duration of eruption (solid line, shorter than 3 minutes; dashed line, longer than 3 minutes).

Waiting times between eruptions are plotted against the durations of the eruptions in Figure 3. Two clusters of observations can be defined for durations below and above 3 minutes (see also Azzalini and Bowman, 1990). The distribution shows a strong bimodality as already illustrated in Figure 2. A dummy variable for durations equal to or longer than 3 minutes is created to define the two distributions and included as covariate  $X$  in a model as the one specified in (4). The latter is then fitted to the Old Faithful Geyser data using the function `rq()` in the package `quantreg` for  $p \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$ .

```
> require("quantreg")
> y <- faithful$waiting
> x <- as.numeric(faithful$eruptions >= 3)
> fit <- rq(formula = y ~ x, tau = c(0.1, 0.25, 0.5, 0.75, 0.9))
> fit
```

```
Call:
rq(formula = y ~ x, tau = c(0.1, 0.25, 0.5, 0.75, 0.9))
```

**Coefficients:**

	tau= 0.10	tau= 0.25	tau= 0.50	tau= 0.75	tau= 0.90
(Intercept)	47	50	54	59	63
x	26	26	26	25	25

Degrees of freedom: 272 total; 270 residual

From the output above, it is quite evident that the distribution of waiting times is shifted by an approximately constant amount at all considered values of  $p$ . The location-shift hypothesis can be tested by using the Khmaladze test. The null hypothesis is that two distributions, say  $F_0$  and  $F_1$ , differ by a pure location shift (Koenker and Xiao, 2002), that is

$$H_0 : F_1^{-1}(p) = F_0^{-1}(p) + \delta_0,$$

where  $\delta_0$  is the quantile treatment effect, constant over  $p$ . The location-scale-shift specification of the test considers

$$H_0 : F_1^{-1}(p) = \delta_1 F_0^{-1}(p) + \delta_0.$$

The alternative hypothesis is that the model is more complex than the one specified in the null hypothesis. The Khmaladze test is implemented in **quantreg** (see ?quantreg::KhmaladzeTest for further details). The critical values of the test and corresponding significance levels (Koenker, 2005) are not readily available in the same package. These have been hardcoded in the **Qtools** function **KhmaladzeFormat()** which can be applied to "KhmaladzeTest" objects. For the Old Faithful Geyser data, the result of the test is not statistically significant at the 10% level.

```
> kt <- KhmaladzeTest(formula = y ~ x, taus = seq(.05, .95, by = .01),
> KhmaladzeFormat(kt, 0.05)
```

```
Khmaladze test for the location-shift hypothesis
Joint test is not significant at 10% level
Test(s) for individual slopes:
  not significant at 10% level
```

### Goodness of fit

Distribution-free quantile regression does not require introducing an assumption on the functional form of the error distribution (Koenker and Bassett, 1978), but only weaker quantile restrictions (Powell, 1994). Comparatively, the linear specification of the conditional quantile function in Equation 4 is a much stronger assumption and thus plays an important role for inferential purposes.

The problem of assessing the goodness of fit (GOF) is rather neglected in applications of QR. Although some approaches to GOF have been proposed (Zheng, 1998; Koenker and Machado, 1999; He and Zhu, 2003; Khmaladze and Koul, 2004), there is currently a shortage of software code available to users. The function **GOFTest()** implements a test based on the cusum process of the gradient vector (He and Zhu, 2003). Briefly, the test statistic is given by the largest eigenvalue of

$$n^{-1} \sum_i^n \mathbf{R}_n(\mathbf{x}_i) \mathbf{R}_n^\top(\mathbf{x}_i)$$

where  $\mathbf{R}_n(\mathbf{t}) = n^{-1/2} \sum_{j=1}^n \psi_p(r_j) \mathbf{x}_j I(\mathbf{x}_j \leq \mathbf{t})$  is the residual cusum (RC) process and  $\psi_p(r_j)$  is the derivative of the loss function  $\kappa_p$  calculated for residual  $r_j = y_j - \mathbf{x}_j^\top \boldsymbol{\beta}(p)$ . The sampling distribution of this test statistic is non-normal (He and Zhu, 2003) and a resampling approach is used to obtain the  $p$ -value under the null hypothesis.

An example is provided further below using the New York Air Quality data set, which contains 111 complete observations on daily mean ozone (parts per billion – ppb) and solar radiation (Langley – Ly). For simplicity, wind speed and maximum daily temperature, also included in the data set, are not analysed here.

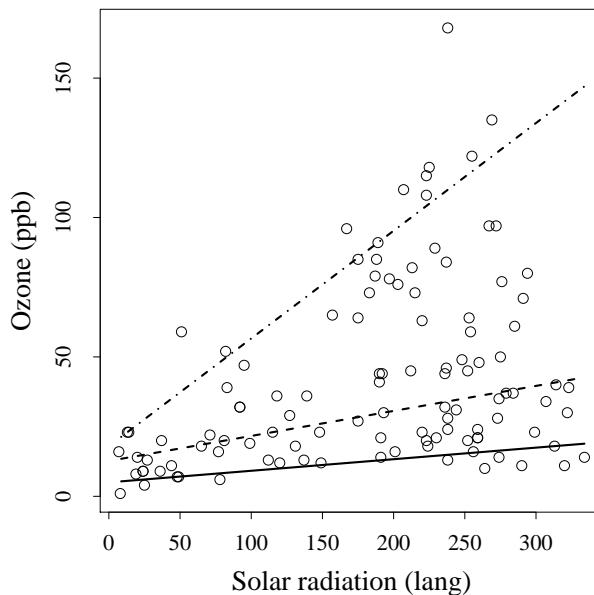
Suppose that the model of interest is

$$Q_{\text{ozone}}(p) = \beta_0(p) + \beta_1(p) \cdot \text{Solar.R}. \quad (5)$$

Three conditional quantiles ( $p \in \{0.1, 0.5, 0.9\}$ ) are estimated and plotted using the following code:

```
> dd <- airquality[complete.cases(airquality), ]
> dd <- dd[order(dd$Solar.R), ]
> fit.rq <- rq(Ozone ~ Solar.R, tau = c(.1, .5, .9), data = dd)
```

```
> x <- seq(min(dd$Solar.R), max(dd$Solar.R), length = 200)
> yhat <- predict(fit.rq, newdata = data.frame(Solar.R = x))
> plot(Ozone ~ Solar.R, data = dd)
> apply(yhat, 2, function(y, x) lines(x, y), x = x)
```



**Figure 4:** Predicted 10th (solid line), 50th (dashed line), and 90th (dot-dashed line) centiles of ozone conditional on solar radiation in the New York Air Quality data set.

As a function of solar radiation, the median of the ozone daily averages increases by 0.09 ppb for each Ly increase in solar radiation (Figure 4). The 90th centile of conditional ozone shows a steeper slope at 0.39 ppb/Ly, about nine times larger than the slope of the conditional 10th centile at 0.04 ppb/Ly.

The RC test applied to the object `fit.rq` provides evidence of lack of fit for all quantiles considered, particularly for  $p = 0.1$  and  $p = 0.5$ . Therefore the straight-line model in Equation 5 for these three conditional quantiles does not seem to be appropriate. The New York Air Quality data set will be analysed again in the next section, where a transformation-based approach to nonlinear modelling is discussed.

```
> gof.rq <- GOFTest(fit.rq, alpha = 0.05, B = 1000, seed = 987)
> gof.rq
```

```
Goodness-of-fit test for quantile regression based on the cusum process
Quantile 0.1: Test statistic = 0.1057; p-value = 0.001
Quantile 0.5: Test statistic = 0.2191; p-value = 0
Quantile 0.9: Test statistic = 0.0457; p-value = 0.018
```

### Transformation models

Complex dynamics may result in nonlinear effects in the relationship between the covariates and the response variable. For instance, in kinesiology, pharmacokinetics, and enzyme kinetics, the study of the dynamics of an agent in a system involves the estimation of nonlinear models; phenomena like human growth, certain disease mechanisms and the effects of harmful environmental substances such as lead and mercury, may show strong nonlinearities over time. In this section, the linear model is abandoned in favor of a more general model of the type

$$Q_{Y|X}(p) = g \left\{ \mathbf{x}^\top \boldsymbol{\beta}(p) \right\}, \quad (6)$$

for some real-valued function  $g$ . If  $g$  is nonlinear, the alternative approaches to conditional quantile modelling are

**Nonlinear parametric models** which may provide substantive interpretability, possibly parsimonious (in general more parsimonious than polynomials), and valid beyond the observed range

of the data. A nonlinear model depends on either prior knowledge of the phenomenon or the introduction of new, strong theory to explain the observed relationship with potential predictive power. Estimation may present challenges;

**Polynomial models and smoothing splines** falling under the label of *nonparametric regression*, in which the complexity of the model is approximated by a sequence of locally linear polynomials (a naïve global polynomial trend can be considered to be a special case). A nonparametric model need not introducing strong assumptions about the relationship and is essentially data-driven. Estimation is based on linear approximations and, typically, requires the introduction of a penalty term to control the degree of smoothing; and

**Transformation models** a flexible, parsimonious family of parametric transformations is applied to the response seeking to obtain approximate linearity on the transformed scale. The data provide information about the “best” transformation among a family of transformations. Estimation is facilitated by the application of methods for linear models.

The focus of this section is on the third approach. More specifically the functions available in **Qtools** refer to the methods for transformation-based QR models developed by Powell (1991), Chamberlain (1994), Mu and He (2007), Dehbi et al. (2016) and Geraci and Jones (2015). Examples of approaches to nonlinear QR based on parametric models or splines can be found in Koenker and Park (1996) and Yu and Jones (1998), respectively.

The goal of the transformation-based QR is to fit the model

$$Q_{h(Y;\lambda_p)}(p) = \mathbf{x}^\top \boldsymbol{\beta}(p). \quad (7)$$

The assumption is that the transformation  $h$  is the inverse of  $g$ ,  $h(Y;\lambda_p) \equiv g^{-1}(Y)$ , so that the  $p$ th quantile function of the transformed response variable is linear. (In practice, it is satisfactory to achieve approximate linearity.) The parameter  $\lambda_p$  is a low-dimensional parameter that gives some flexibility to the shape of the transformation and is estimated from the data. In general, the interest is on predicting  $Q_{Y|X}(p)$  and estimating the effects of the covariates on  $Q_{Y|X}(p)$ . If  $h$  is a non-decreasing function on  $\mathbb{R}$  (as is the case for all transformations considered here), predictions can be easily obtained from (7) by virtue of the equivariance property of quantiles,

$$Q_{Y|X}(p) = h^{-1} \left\{ \mathbf{x}^\top \boldsymbol{\beta}(p); \lambda_p \right\}. \quad (8)$$

The marginal effect of the  $j$ th covariate  $x_j$  can be obtained by differentiating the quantile function  $Q_{Y|X}(p)$  with respect to  $x_j$ . This can be written as the derivative of the composition  $Q \circ \eta$ , i.e.

$$\frac{\partial Q(p)}{\partial x_j} = \frac{\partial Q(p)}{\partial \eta(p)} \cdot \frac{\partial \eta(p)}{\partial x_j}, \quad (9)$$

$\eta(p) = \mathbf{x}^\top \boldsymbol{\beta}(p)$ . Once the estimates  $\hat{\boldsymbol{\beta}}(p)$  and  $\hat{\lambda}_p$  are obtained, these can be plugged in Equations 8 and 9.

The package **Qtools** provides several transformation families, namely the Box–Cox (Box and Cox, 1964), Aranda-Ordaz (Aranda-Ordaz, 1981), and Jones (Jones, 2007; Geraci and Jones, 2015) transformations. A distinction between these families is made in terms of the support of the response variable to which the transformation is applied and the number of transformation parameters. The Box–Cox model is a one-parameter family of transformations which applies to singly bounded variables,  $y > 0$ . The Aranda-Ordaz symmetric and asymmetric transformations too have one parameter and are used when responses are bounded on the unit interval,  $0 < y < 1$  (doubly bounded). Geraci and Jones (2015) developed two families of transformations which can be applied to either singly or doubly bounded responses:

**Proposal I transformations** with one parameter and assuming both symmetric and asymmetric forms;

**Proposal II transformations** with two parameters, with one parameter modelling the symmetry (or lack thereof) of the transformation.

Originally, Box and Cox (1964) proposed using power transformations to address lack of linearity, homoscedasticity and normality of the residuals in mean regression modelling. Sakia (1992, p.175) reported that “seldom does this transformation fulfil the basic assumptions of linearity, normality and homoscedasticity simultaneously as originally suggested by Box & Cox (1964). The Box-Cox transformation has found more practical utility in the empirical determination of functional relationships in a variety of fields, especially in econometrics”.

Indeed, the practical utility of power transformations has been long recognised in QR modelling (Powell, 1991; Buchinsky, 1995; Chamberlain, 1994; Mu and He, 2007). Model 7 is the Box–Cox QR

model if

$$h_{BC}(Y; \lambda_p) = \begin{cases} \frac{Y^{\lambda_p} - 1}{\lambda_p} & \text{if } \lambda_p \neq 0 \\ \log Y & \text{if } \lambda_p = 0. \end{cases} \quad (10)$$

Note that when  $\lambda_p \neq 0$ , the range of this transformation is not  $\mathbb{R}$  but the singly bounded interval  $(-1/\lambda_p, \infty)$ . This implies that the inversion in (8) is defined only for  $\lambda_p \mathbf{x}^\top \boldsymbol{\beta}(p) + 1 > 0$ .

The symmetric Aranda-Ordaz transformation is given by

$$h_{AOs}(Y; \lambda_p) = \begin{cases} \frac{2}{\lambda_p} \frac{Y^{\lambda_p} - (1-Y)^{\lambda_p}}{Y^{\lambda_p} + (1-Y)^{\lambda_p}} & \text{if } \lambda_p \neq 0, \\ \log\left(\frac{Y}{1-Y}\right) & \text{if } \lambda_p = 0. \end{cases} \quad (11)$$

(The symmetry here is that  $h_{AOs}(\theta; \lambda_p) = -h_{AOs}(1-\theta; \lambda_p) = h_{AOs}(\theta; -\lambda_p)$ .) There is a range problem with this transformation too since, for all  $\lambda_p \neq 0$ , the range of  $h_{AOs}$  is not  $\mathbb{R}$ , but  $(-2/|\lambda_p|, 2/|\lambda_p|)$ . The asymmetric Aranda-Ordaz transformation is given by

$$h_{AOa}(Y; \lambda_p) = \begin{cases} \log\left\{\frac{(1-Y)^{-\lambda_p} - 1}{\lambda_p}\right\} & \text{if } \lambda_p \neq 0, \\ \log\{-\log(1-Y)\} & \text{if } \lambda_p = 0. \end{cases} \quad (12)$$

For  $\lambda_p = 0$ , this is equivalent to the complementary log-log. The asymmetric Aranda-Ordaz transformation does have range  $\mathbb{R}$ . Note that  $h_{AOa}(Y; 1) = \log(Y/(1-Y))$ , i.e. the transformation is symmetric.

To overcome range problems, which give rise to computational difficulties, Geraci and Jones (2015) proposed to use instead one-parameter transformations with range  $\mathbb{R}$ . Proposal I is written in terms of the variable (say)  $W$ , where

$$h_I(W; \lambda_p) = \begin{cases} \frac{1}{2\lambda_p} \left(W^{\lambda_p} - \frac{1}{W^{\lambda_p}}\right) & \text{if } \lambda_p \neq 0 \\ \log W & \text{if } \lambda_p = 0, \end{cases} \quad (13)$$

which takes on four forms depending on the relationship of  $W$  to  $Y$ , as described in Table 2. For each of domains  $(0, \infty)$  and  $(0, 1)$ , there are symmetric and asymmetric forms.

Support of $Y$	Symmetric	Asymmetric
$(0, \infty)$	$W = Y$ $h_{Is}(Y; \lambda_p)$	$W = \log(1+Y)$ $h_{Ia}(Y; \lambda_p)$
$(0, 1)$	$W = Y/(1-Y)$ $h_{Is}(Y; \lambda_p)$	$W = -\log(1-Y)$ $h_{Ia}(Y; \lambda_p)$

**Table 2:** Choices of  $W$  and corresponding notation for transformations based on (13).

Since the transformation in (13) has range  $\mathbb{R}$  for all  $\lambda_p$ , it admits an explicit inverse transformation. In addition, in the case of a single covariate, every estimated quantile that results will be monotone increasing, decreasing or constant, although different estimated quantiles can have different shapes from this collection. Geraci and Jones (2015) also proposed a transformation that unifies the symmetric and asymmetric versions of  $h_I$  into a single two-parameter transformation, namely

$$h_{II}(W; \lambda_p) = h_I(W_{\delta_p}; \lambda_p), \quad (14)$$

where  $h_I$  is given in (13) and

$$W_{\delta_p} = h_{BC}(1 + W; \delta_p) = \begin{cases} \frac{(1 + W)^{\delta_p} - 1}{\delta_p} & \text{if } \delta_p > 0 \\ \log(1 + W) & \text{if } \delta_p = 0, \end{cases}$$

with  $W = Y$ , if  $Y > 0$ , and  $W = Y/(1 - Y)$ , if  $Y \in (0, 1)$ . The additional parameter  $\delta_p$  controls the asymmetry: symmetric forms of  $h_I$  correspond to  $\delta_p = 1$  while asymmetric forms of  $h_I$  to  $\delta_p = 0$ .

All transformation models discussed above can be fitted using a two-stage (TS) estimator (Chamberlain, 1994; Buchinsky, 1995) whereby  $\beta(p)$  is estimated conditionally on a fine grid of values for the transformation parameter(s). Alternatively, point estimation can be approached using the RC process (Mu and He, 2007), which is akin to the process that leads to the RC test introduced in the previous section. The RC estimator avoids the troublesome inversion of the Box-Cox and Aranda-Ordaz transformations, but it is computationally more intensive than the TS estimator.

There are several methods for interval estimation, including those based on large- $n$  approximations and the ubiquitous bootstrap. Both the TS and RC estimators have an asymptotic normal distribution. The large-sample properties of the TS estimator for monotonic quantile regression models have been studied by Powell (1991) (see also Chamberlain, 1994; Machado and Mata, 2000). Under regularity conditions, it can be shown that the TS estimator is unbiased and will converge to a normal distribution with a sandwich-type limiting covariance matrix which is easy to calculate. In contrast, the form of the covariance matrix of the sampling distribution for the RC estimator is rather complicated and its estimation requires resampling (Mu and He, 2007). Finally, if the transformation parameter is assumed to be known, then conditional inference is appropriate. In this case, the estimation procedures simplify to those for standard quantile regression problems.

In **Qtools**, model fitting for one-parameter transformation models can be carried out using the function `tsrq()`. The `formula` argument specifies the model for the linear predictor as in (7), while the argument `tsf` provides the desired transformation  $h$  as specified in Equations 10-13: "bc" for the Box-Cox model, "ao" for Aranda-Ordaz families, and "mcjI" for proposal I transformations. Additional arguments in the function `tsrq()` include

`symmetry` a logical flag to specify the symmetric or asymmetric version of "ao" and "mcjI";

`dbounded` a logical flag to specify whether the response variable is doubly bounded (default is strictly positive, i.e. singly bounded);

`lambda` a numerical vector to define the grid of values for estimating  $\lambda_p$ ; and `conditional`, a logical flag indicating whether  $\lambda_p$  is assumed to be known (in which case the argument `lambda` provides such known value).

There are other functions to fit transformation models. The function `rqrq()` fits one-parameter transformation models using the RC estimator. The functions `tsrq2()` and `n1rq2()` are specific to Geraci and Jones's (2015) Proposal II transformations. The former employs a two-way grid search while the latter is based on Nelder-Mead optimization as implemented in `optim()`. Simulation studies in Geraci and Jones (2015) suggest that, although computationally slower, a two grid search is numerically more stable than the derivative-free approach.

A summary of the basic differences between all fitting functions is given in Table 3. The table also shows the available methods in `summary.rqt()` to estimate standard errors and confidence intervals for the model's parameters. Unconditional inference is carried out jointly on  $\beta(p)$  and the transformation parameter by means of bootstrap using the package `boot` (Canty and Ripley, 2014; Davison and Hinkley, 1997). Large- $n$  approximations (Powell, 1991; Chamberlain, 1994; Machado and Mata, 2000) are also available for the one-parameter TS estimator under iid or nid assumptions.

When `summary.rqt()` is executed with the argument `conditional = TRUE`, confidence interval estimation for  $\beta_p$  is performed with one of the several methods developed for linear quantile regression estimators (Koenker, 2005, p.110) (see options "rank", "iid", "nid", "ker", and "boot" in `quantreg::summary.rq()`).

In the New York Air Quality data example, a linear model was found unsuitable to describe the relationship between ozone and solar radiation. At closer inspection, Figure 4 reveals that the conditional distribution of ozone may in fact be nonlinearly associated with solar radiation, at least for some of the conditional quantiles. The model

$$Q_{h_{Is}\{\text{ozone}\}}(p) = \beta_0(p) + \beta_1(p) \cdot \text{Solar.R}, \quad (15)$$

where  $h_{Is}$  denotes the symmetric version of (13) for a singly bounded response variable, is fitted for the quantiles  $p \in \{0.1, 0.5, 0.9\}$  using the following code:

Function name	Transformation parameters	Estimation	Standard errors or confidence intervals	
			Unconditional	Conditional
tsrq()	1	Two-stage	"iid", "boot"	"nid", All types
rcrq()	1	Residual process	cusum "boot"	All types
tsrq2()	2	Two-stage	"boot"	All types
nlrq2()	2	Nelder-Mead	"boot"	-

**Table 3:** Transformation-based quantile regression in package **Qtools**. All types consists of options "rank", "iid", "nid", "ker", and "boot" as provided by function summary() in package **quantreg**.

```
> system.time(fit.rqt <- tsrq(Ozone ~ Solar.R, data = dd, tsf = "mcjI",
+   symm = TRUE, dbounded = FALSE, lambda = seq(1, 3, by = 0.005),
+   conditional = FALSE, tau = c(.1, .5, .9)))
    user   system elapsed
    0.5     0.0     0.5
> fit.rqt

call:
tsrq(formula = Ozone ~ Solar.R, data = dd, tsf = "mcjI", symm = TRUE,
      dbounded = FALSE, lambda = seq(1, 3, by = 0.005), conditional = FALSE,
      tau = c(0.1, 0.5, 0.9))

Proposal I symmetric transformation (singly bounded response)

Optimal transformation parameter:
tau = 0.1 tau = 0.5 tau = 0.9
  2.210    2.475    1.500

Coefficients linear model (transformed scale):
      tau = 0.1 tau = 0.5 tau = 0.9
(Intercept) -3.3357578 -48.737341 16.557327
Solar.R       0.4169697  6.092168 1.443407

Degrees of freedom: 111 total; 109 residual
```

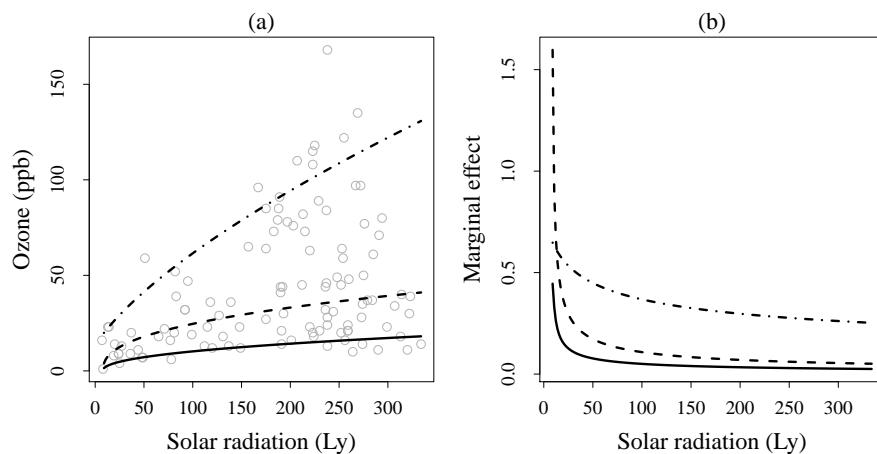
The TS estimator makes a search for  $\lambda_p$  over the grid 1.000, 1.005, ..., 2.995, 3.000. The choice of the search interval usually results from a compromise between accuracy and performance: the coarser the grid, the faster the computation but the less accurate the estimate. A reasonable approach would be to start with a coarse, wide-ranging grid (e.g. seq(-5, 5, by = 0.5)), then center the interval about the resulting estimate using a finer grid, and re-fit the model.

The output above reports the estimates  $\hat{\beta}(p)$  and  $\hat{\lambda}_p$  for each quantile level specified in tau. Here, the quantities of interest are the predictions on the ozone scale and the marginal effect of solar radiation, which can be obtained using the function predict.rqt().

```
> x <- seq(9, 334, length = 200)
> qhat <- predict(fit.rqt, newdata = data.frame(Solar.R = x),
+   type = "response")
> dqhat <- predict(fit.rqt, newdata = data.frame(Solar.R = x),
+   type = "margref", namevec = "Solar.R")
The linear component of the marginal effect is calculated as derivative of
Ozone ~ beta1 * Solar.R
with respect to Solar.R
```

The calculations above are based on a sequence of 200 ozone values in the interval [9, 334] Ly, as provided via the argument newdata (if this argument is missing, the function returns the fitted values). There are three types of predictions available:

link predictions of conditional quantiles on the transformed scale (7), i.e.  $\hat{Q}_{h(Y;\hat{\lambda}_p)}(p) = \mathbf{x}^T \hat{\beta}(p)$ ;  
 response predictions of conditional quantiles on the original scale (8), i.e.  $\hat{Q}_{Y|X}(p) = h^{-1} \left\{ \mathbf{x}^T \hat{\beta}(p); \hat{\lambda}_p \right\}$ ; and



**Figure 5:** Predicted 10th (solid line), 50th (dashed line), and 90th (dot-dashed line) centiles of ozone conditional on solar radiation (a) and corresponding estimated marginal effects (b) using the symmetric proposal I transformation in the New York Air Quality data set.

`maref` predictions of the marginal effect (9).

In the latter case, the argument `namevec` is used to specify the name of the covariate with respect to which the marginal effect has to be calculated. The function `maref.rqt()` computes derivatives symbolically using the `stats` function `deriv()` and these are subsequently evaluated numerically. While the nonlinear component of the marginal effect in Equation 9 (i.e.  $\partial Q(p)/\partial \eta(p)$ ) is rather straightforward to derive for any of the transformations (10)-(13), the derivative of the linear predictor (i.e.  $\partial \eta(p)/\partial x_j$ ) requires parsing the `formula` argument in order to obtain an expression suitable for `deriv()`. The function `maref.rqt()` can handle simple expressions with common functions like `log()`, `exp()`, etc., interaction terms, and "AsIs" terms (i.e. `I()`). However, using functions that are not recognised by `deriv()` will trigger an error.

The predicted quantiles of ozone and the marginal effects of solar radiation are plotted in Figure 5 using the following code:

```
> par(mfrow = c(1, 2))
> plot(Ozone ~ Solar.R, data = dd, xlab = "Solar radiation (lang)",
+       ylab = "Ozone (ppb)")
> for(i in 1:3) lines(x, qhat[, i], lty = c(1, 2, 4)[i], lwd = 2)
> plot(range(x), range(dqhat), type = "n", xlab = "Solar radiation (lang)",
+       ylab = "Marginal effect")
> for(i in 1:3) lines(x, dqhat[, i], lty = c(1, 2, 4)[i], lwd = 2)
```

The effect of solar radiation on different quantiles of ozone levels shows a nonlinear behavior, especially at lower ranges of radiation (below 50 Ly) and on the median ozone. It might be worth testing the goodness-of-fit of the model. In the previous analysis, it was found evidence of lack of fit for the linear specification (5). In contrast, the output reported below indicates that, in general, the goodness of fit of the quantile models based on the transformation model (15) has improved since the test statistics are now smaller at all values of  $p$ . However, such improvement is not yet satisfactory for the median.

```
> GOFTest(fit.rqt, alpha = 0.05, B = 1000, seed = 416)
```

```
Goodness-of-fit test for quantile regression based on the cusum process
Quantile 0.1: Test statistic = 0.0393; p-value = 0.025
Quantile 0.5: Test statistic = 0.1465; p-value = 0.005
Quantile 0.9: Test statistic = 0.0212; p-value = 0.127
```

The TS and RC estimators generally provide similar estimates and predictions. However, computation based on the cusum process tends to be somewhat slow, as shown further below. This is also true for the RC test provided by `GOFTest()`.

```
> system.time(fit.rqt <- rcrq(Ozone ~ Solar.R, data = dd, tsf = "mcjI",
+      symm = TRUE, dbounded = FALSE, lambda = seq(1, 3, by = 0.005),
+      tau = c(.1, .5, .9)))
```

```
user    system elapsed
36.88     0.03   37.64
```

An example using doubly bounded transformations is demonstrated using the A-level Chemistry Scores data set. The latter is available from [Qtools](#) and it consists of 31022 observations of A-level scores in Chemistry for England and Wales students, 1997. The data set also includes information of prior academic achievement as assessed with General Certificate of Secondary Education (GCSE) average scores. The goal is to evaluate the ability of GCSE to predict A-level scores. The latter are based on national exams in specific subjects (e.g. chemistry) with grades ranging from A to F. For practical purposes, scores are converted numerically as follows: A = 10, B = 8, C = 6, D = 4, E = 2, and F = 0. The response is therefore doubly bounded between 0 ad 10. It should be noted that this variable is discrete, although, for the sake of simplicity, here it is assumed that the underlying process is continuous.

The model considered here is

$$Q_{h_{AOa}\{\text{score}\}}(p) = \beta_0(p) + \beta_1(p) \cdot \text{gcse}, \quad (16)$$

where  $h_{AOa}$  denotes the asymmetric Aranda-Ordaz transformation in (12). This model is fitted for  $p = 0.9$ :

```
> data(Chemistry)
> fit.rqt <- tsrq(score ~ gcse, data = Chemistry, tsf = "ao", symm = FALSE,
+ lambda = seq(0, 2, by = 0.01), tau = 0.9)
```

The predicted 90th centile of A-level scores conditional on GCSE and the marginal effect of GCSE are plotted in Figure 6. There is clearly a positive, nonlinear association between the two scores. The nonlinearity is partly explained by the floor and ceiling effects which result from the boundedness of the measurement scale. Note, however, that the S-shaped curve is not symmetric about the inflection point. As a consequence, the marginal effect is skewed to the left. Indeed, the estimate  $\hat{\lambda}_{0.9} = 0$  and the narrow confidence interval give support to a complementary log-log transformation:

```
> summary(fit.rqt, conditional = FALSE, se = "nid")
call:
summary.rqt(object = fit.rqt, se = "nid", conditional = FALSE)

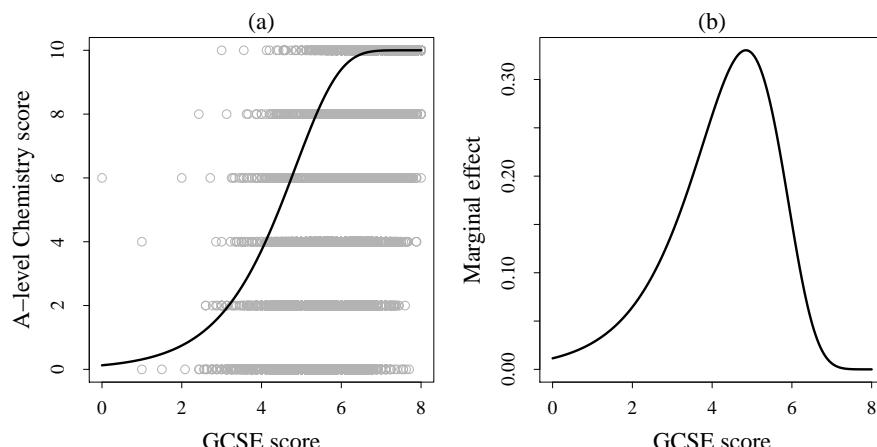
Aranda-Ordaz asymmetric transformation (doubly bounded response)

Summary for unconditional inference

tau = 0.9

Optimal transformation parameter:
      Value   Std. Error Lower bound  Upper bound

```



**Figure 6:** Predicted 90th centile of A-level scores conditional on GCSE scores (a) and corresponding estimated marginal effect (b) using the asymmetric Aranda-Ordaz transformation in the A-level Chemistry Scores data set.

```
0.000000000 0.001364422 -0.002674218 0.002674218
```

Coefficients linear model (transformed scale):

	Value	Std. Error	Lower bound	Upper bound
(Intercept)	-4.3520060	0.015414540	-4.3822179	-4.3217941
gcse	0.8978072	0.002917142	0.8920898	0.9035247

Degrees of freedom: 31022 total; 31020 residual

Alternatively, one can estimate the parameter  $\delta_p$  using a two-parameter transformation:

```
> coef(tsrq2(score ~ gcse, data = chemsub, dbounded = TRUE,
+   lambda = seq(0, 2, by = 0.1), delta = seq(0, 2, by = 0.1),
+   tau = 0.9), all = TRUE)

(Intercept)      gcse      lambda      delta
-4.1442274    0.8681246   0.0000000   0.0000000
```

These results confirm the asymmetric nature of the relationship since  $\hat{\delta}_{0.9} = 0$ . Similar results (not shown) were obtained with nlrq2().

In conclusion, the package **Qtools** offers several options in terms of transformations and estimation algorithms, the advantages and disadvantages of which are discussed by [Geraci and Jones \(2015\)](#). In particular, they found that the symmetric Proposal I transformation improves considerably on the Box-Cox method and marginally on the Aranda-Ordaz transformation in terms of mean squared error of the predictions. Also, asymmetric transformations do not seem to improve sufficiently often on symmetric transformations to be especially recommendable. However, the Box-Cox and the symmetric Aranda-Ordaz transformations should not be used when individual out-of-range predictions represent a potential inconvenience as, for example, in multiple imputation (see section further below). Finally, in some situations transformation-based quantile regression may be competitive as compared to methods based on smoothing, as demonstrated by a recent application to anthropometric charts ([Boghossian et al., 2016](#)).

## Conditional LSS

Quantile-based measures of location, scale, and shape can be assessed *conditionally* on covariates. A simple approach is to fit a linear model as in (4) or a transformation-based model as in (7), and then predict  $\hat{Q}_{Y|X}(p)$  to obtain the conditional LSS measures in Equation 3 for specific values of  $x$ .

Estimation of conditional LSS can be carried out by using the function qlss.formula(). The conditional model is specified in the argument formula, while the probability  $p$  is given in probs. (As seen in Equation 3, the other probabilities of interest to obtain the decomposition of the conditional quantiles are  $1 - p$ , 0.25, 0.5, and 0.75.) The argument type specifies the required type of regression model, more specifically "rq" for linear models and "rqt" for transformation-based models. The function qlss.formula() will take any additional argument to be passed to quantreg::rq() or tsrq() (e.g. subset, weights, etc.).

Let's consider the New York Air Quality data example discussed in the previous section and assume that the transformation model (15) holds for the quantiles  $p \in \{0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95\}$ . Then the conditional LSS summary of the distribution of ozone conditional on solar radiation for  $p = 0.05$  and  $p = 0.1$  is calculated as follows:

```
> fit.qlss <- qlss(formula = Ozone ~ Solar.R, data = airquality, type =
+   "rqt", tsf = "mcjI", symm = TRUE, dbounded = FALSE, lambda =
+   seq(1, 3, by = 0.005), probs = c(0.05, 0.1))
> fit.qlss

call:
qlss.formula(formula = Ozone ~ Solar.R, probs = c(0.05, 0.1),
  data = airquality, type = "rqt", tsf = "mcjI", symm = TRUE,
  dbounded = FALSE, lambda = seq(1, 3, by = 0.005))

Conditional Quantile-Based Location, Scale, and Shape
-- Values are averaged over observations --

** Location **
Median
```

```
[1] 30.2258
** Scale **
Inter-quartile range (IQR)
[1] 43.40648
Inter-quantile range (IPR)
  0.05      0.1
88.02909 73.93430
**Shape**
Skewness index
  0.05      0.1
0.5497365 0.5180108
Shape index
  0.05      0.1
1.960315 1.661648
```

The output, which is of class "qlss", is a named list with the same LSS measures seen in the case of unconditional quantiles. However, these are now conditional on solar radiation. By default, the predictions are the fitted values, which are averaged over observations for printing purposes. An optional data frame for predictions can be given via the argument newdata in predict.qlss(). If interval = TRUE, the latter computes confidence intervals at the specified level using R bootstrap replications (it is, therefore, advisable to set the seed before calling predict.qlss()). The conditional LSS measures can be conveniently plotted using the plot.qlss() function as shown in the code below. The argument z is required and specifies the covariate used for plotting. Finally, the argument whichp specifies one probability (and one only) among those given in probs that should be used for plotting (e.g.  $p = 0.1$  in the following example).

```
> set.seed(567)
> x <- seq(9, 334, length = 200)
> qhat <- predict(fit.qlss, newdata = data.frame(Solar.R = x),
+   interval = TRUE, level = 0.90, R = 500)
> plot(qhat, z = x, whichp = 0.1, interval = TRUE, type = "l",
+   xlab = "Solar radiation (lang)", lwd = 2)
```

Figure 7 shows that both the median and the IQR of ozone increase nonlinearly with increasing solar radiation. The distribution of ozone is skewed to the right and the degree of asymmetry is highest at low values of solar radiation. This is due to the extreme curvature of the median which takes on values close to the 10th centile (Figure 5). (Recall that the index approaches 1 when  $Q(p) \approx Q(0.5)$ .) However, the sparsity of observations at the lower end of the observed range of solar radiation determines substantial uncertainty as reflected by the wider confidence interval (Figure 7). At  $p = 0.1$ , the conditional shape index is on average equal to 1.66 and it increases monotonically from 1.32 to about 1.85, remaining always below the tail-weight threshold of a normal distribution (1.90).

## Other functions in Qtools

### Restricted quantile regression

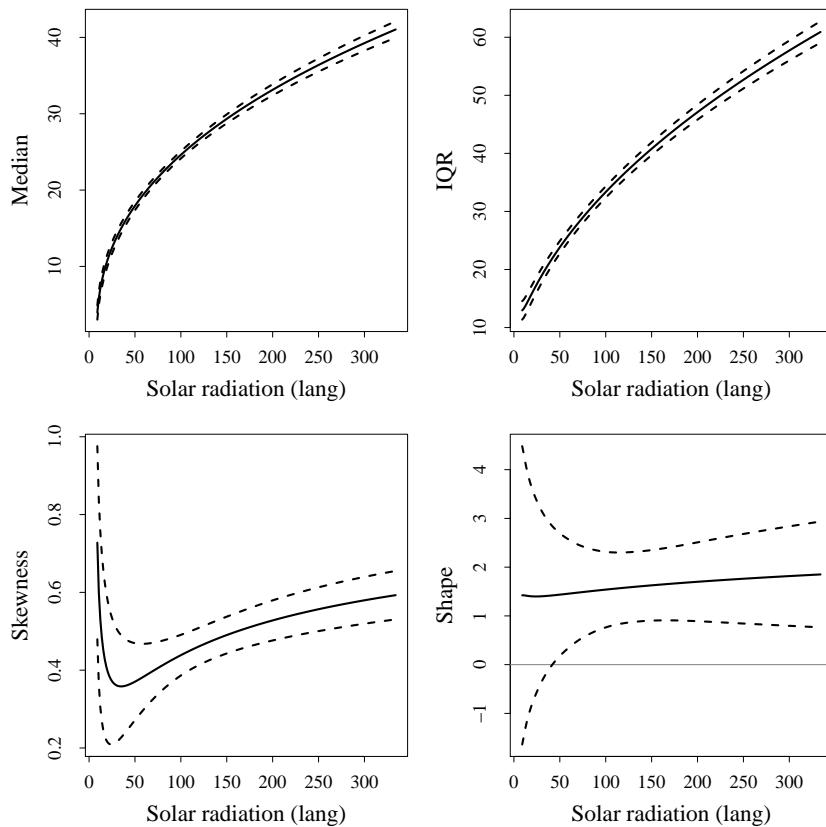
Besides a loss of precision, high sparsity (low density) might also lead to a violation of the basic property of monotonicity of quantile functions. Quantile crossing occurs when  $\mathbf{x}_i^\top \hat{\beta}(p) > \mathbf{x}_i^\top \hat{\beta}(p')$  for some  $\mathbf{x}_i$  and  $p < p'$ . This problem typically occurs in the outlying regions of the design space (Koenker, 2005) where also sparsity occurs more frequently. Balanced designs with larger sample sizes would then offer some assurance against quantile crossing, provided, of course, that the QR models are correctly specified. Model misspecification, indeed, can still be a cause of crossing of the quantile curves. Restricted regression quantiles (RRQ) (He, 1997) might offer a practical solution when little can be done in terms of modelling. This approach applies to a subclass of linear models

$$Y = \mathbf{x}^\top \beta + \epsilon$$

and linear heteroscedastic models

$$Y = \mathbf{x}^\top \beta + (\mathbf{x}^\top \gamma) \epsilon,$$

where  $\mathbf{x}^\top \gamma > 0$  and  $\epsilon \sim F$ . Basically, it consists in fitting a reduced regression model passing through the origin. The reader is referred to He (1997) for details. Here, it is worth stressing that when the restriction does not hold, i.e. if the model is more complex than a location-scale-shift model, then RRQ may yield unsatisfactory results He (1997). See also Zhao (2000) for an examination of the asymptotic



**Figure 7:** Location, scale and shape of ozone levels conditional on solar radiation in the New York Air Quality data set. Dashed lines denote the bootstrapped 90% point-wise confidence intervals.

properties of the restricted QR estimator. In particular, the relative efficiency of RRQ as compared to RQ depends on the error distribution. For some common unimodal distributions, Zhao (2000) showed that RRQ in iid models is more efficient than RQ. This property is lost when the error is asymmetric. In contrast, the efficiency of RRQ in heteroscedastic models is comparable to that of RQ even for small samples.

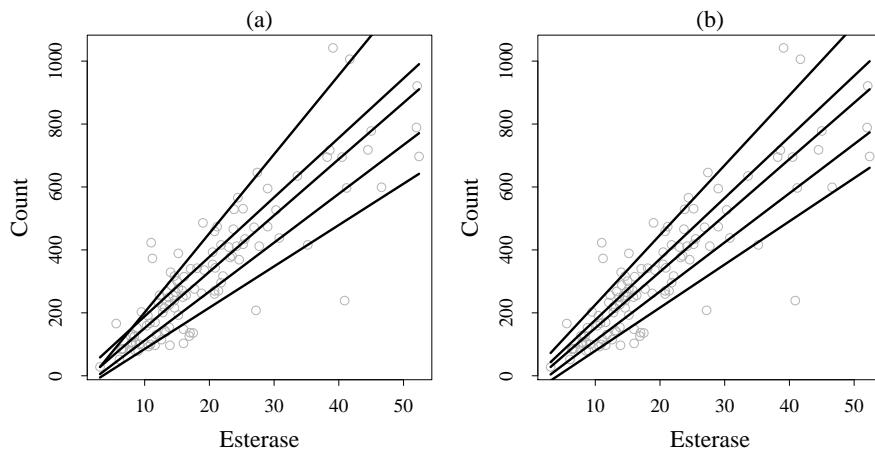
The package **Qtools** provides the functions `rrq()`, `rrq.fit()` and `rrq.wfit()` which are, respectively, the *restricted* analogs of `rq()`, `rq.fit()`, and `rq.wfitv` in **quantreg**. S3 methods `print()`, `coef()`, `predict()`, `fitted()`, `residuals()`, and `summary()` are available for objects of class "rrq". In particular, confidence intervals are obtained using the functions `boot()` and `boot.ci()` from package **boot**. Future versions of the package will develop the function `summary.rrq()` to include asymptotic standard errors (Zhao, 2000). An application is shown below using an example discussed by Zhao (2000). The data set, available from **Qtools**, consists of 118 measurements of esterase concentrations and number of bindings counted in binding experiments.

```
> data("esterase")
> taus <- c(.1, .25, .5, .75, .9)
> fit.rq <- rq(Count ~ Esterase, data = esterase, tau = taus)
> yhat1 <- fitted(fit.rq)
> fit.rrq <- rrq(Count ~ Esterase, data = esterase, tau = taus)
> yhat2 <- fitted(fit.rrq)
```

The predicted 90th centile curve crosses the 50th and 75th curves at lower esterase concentrations (Figure 8). The crossing is removed in predictions based on RRQs.

As discussed above, the reliability of the results depends on the validity of the restriction carried by RRQ. A quick check can be performed using the location-scale-shift specification of the Khmaladze test.

```
> kt <- KhmaladzeTest(formula = Count ~ Esterase, data = esterase,
+ taus = seq(.05,.95,by = .01), nullH = "location-scale")
> KhmaladzeFormat(kt, 0.05)
```



**Figure 8:** Predicted quantiles of number of bindings conditional on esterase concentration using regression quantiles (a) and restricted regression quantiles (b) in the Esterase data set.

Khmaladze test for the location-shift hypothesis

Joint test is not significant at 10% level

Test(s) for individual slopes:

not significant at 10% level

The quantile crossing problem can be approached also by directly rearranging the fitted values  $\hat{Q}_{Y|X=x}(p)$  to obtain monotone (in  $p$ ) predictions for each  $x$  (Chernozhukov et al., 2010). This method is implemented in the package **Rearrangement** (Graybill et al., 2016). As compared to RRQ, this approach is more general as it is not confined to, for example, location-scale-shift models (Chernozhukov et al., 2010); however, in contrast to RRQ, it does not yield estimates of parameters (e.g. slopes) of the model underlying the final monotonised curves. Such estimates, available from "rrq" objects, may be of practical utility when summarising the results.

### Conditional quantiles of discrete data

Modelling conditional functions of discrete data is less common and, on a superficial level, might even appear as an unnecessary complication. However, a deeper look at its rationale will reveal that a distribution-free analysis can provide insightful information in the discrete case as it does in the continuous case. Indeed, methods for conditional quantiles of continuous distributions can be—and have been—adapted to discrete responses.

The package **Qtools** offers some limited functionalities for count and binary data. Further research is needed to develop the theory of QR for discrete data and to improve computational algorithms. Therefore, the user should use these functions with caution.

Let  $Y$  be a count variable such as, for example, the number of car accidents during a week or the number of times a patient visits their doctor during a year. As usual,  $X$  denotes a vector of covariates. Poisson regression, which belongs to the family of generalised linear models (GLMs), is a common choice for this kind of data, partly because of its availability in many statistical packages. Symbolically,  $Y \sim \text{Pois}(\theta)$ , where  $\theta \equiv \mathbb{E}(Y|X = x) = h^{-1}(x^\top \beta)$  and  $h$  is the logarithmic link function. Note that the variance also is equal to  $\theta$ . Indeed, moments of order higher than 2 governing the shape of the distribution depend on the same parameter. Every component of the conditional LSS in a Poisson model is therefore controlled by  $\theta$ . If needed, more flexibility can be achieved using a distribution-free approach.

Machado and Santos Silva (2005) proposed the model

$$Q_{h(Z;p)}(p) = x^\top \beta(p), \quad (17)$$

where  $Z = Y + U$  is obtained by jittering  $Y$  with a  $[0, 1]$ -uniform noise  $U$ , independent of  $Y$  and  $X$ . In principle, any monotone transformation  $h$  can be considered. Given the continuity between counts induced by jittering, standard inference for linear quantile functions (Koenker and Bassett, 1978) can be applied to fit (17). In practice, a sample of  $M$  jittered responses  $Z$  is taken to estimate  $\hat{\beta}_m(p)$ ,  $m = 1, \dots, M$ ; the noise is then averaged out,  $\hat{\beta}(p) = \frac{1}{M} \sum_m \hat{\beta}_m(p)$ .

Machado and Santos Silva's (2005) methods, including large- $n$  approximations for standard errors, are implemented in the function `rq.counts()`. The formula argument specifies a linear model as

in (17), while the argument `tsf` provides the desired transformation  $h$ . By default, this is the log transformation (i.e. Box-Cox with parameter  $\lambda_p = 0$ ) but other transformations are allowed. Note that `GOFTest()` can be applied to "rq.counts" objects as well.

**Qtools** provides functions for modelling binary responses as well. First of all, it is useful to note that the classical GLM for a binary response  $Y \sim \text{Bin}(1, \pi)$  establishes a relationship between the probability  $\Pr(Y = 1) = \pi$  and a set of predictors  $x$ . The application of QR to binary outcomes relies on the continuous latent variable regression formulation

$$Y^* = x^\top \beta + \epsilon \quad (18)$$

and assumes that the binary observations are the result of the dichotomization  $Y = I(Y^* > 0)$ , with  $Y^*$  unobserved.

Maximum score estimation, originally developed by Manski (1975, 1985), is equivalent to estimating the conditional quantiles of the latent variable  $Y^*$ . However, the optimization problem offers numerical challenges due to the piecewise linearity of the indicator function and the nonconvexity of the loss function. The function `rq.bin()` is the main function to obtain binary regression quantiles. It is a wrapper for the function `rqbin.fit()` which calls Fortran code written for simulated annealing estimation (Goffe et al., 1994). **Qtools** offers a limited number of functions for objects of class "rq.bin" including `coef()` and `predict()`. These methods should be considered still experimental. In particular, the user should be aware that the estimates obtained from the fitting procedure may be sensitive to different settings of the simulated annealing algorithm. The latter can be controlled using `rqbinControl()`.

### Quantile-based multiple imputation

Regression models play an important role in conditional imputation of missing values. QR can be used as an effective approach for multiple imputation (MI) when location-shift models are inadequate (Muñoz and Rueda, 2009; Bottai and Zhen, 2013; Geraci, 2016).

In **Qtools**, `mice.impute.rq()` and `mice.impute.rrq()` are auxiliary functions written to be used along with the functions of the R package **mice** (van Buuren and Groothuis-Oudshoorn, 2011). The former is based on the standard QR estimator (`rq.fit()`) while the latter on the restricted counterpart (`rrq.fit()`). Both imputation functions allow for the specification of the transformation-based QR models discussed previously. The equivariance property is useful to achieve linearity of the conditional model and to ensure that imputations lie within some interval when imputed variables are bounded. An example is available from the help file `?mice.impute.rq` using the `nhanes` data set. See also Geraci (2016) for a thorough description of these methods.

### Final remarks

Quantiles have long occupied an important place in statistics. The package **Qtools** builds on recent methodological and computational developments of quantile functions and related methods to promote their application in statistical data modelling.

### Acknowledgements

This work was partially supported by an ASPIRE grant from the Office of the Vice President for Research at the University of South Carolina. I wish to thank anonymous reviewers for their helpful comments and Alexander McLain for his help with revising the final draft of the manuscript.

### Bibliography

- F. J. Aranda-Ordaz. On two families of transformations to additivity for binary response data. *Biometrika*, 68(2):357–363, 1981. [p125]
- A. Azzalini and A. W. Bowman. A look at some data on the Old Faithful Geyser. *Journal of the Royal Statistical Society C*, 39(3):357–365, 1990. [p122]
- Y. Benjamini and A. M. Krieger. Concepts and measures for skewness with data-analytic implications. *Canadian Journal of Statistics*, 24(1):131–140, 1996. [p120]

- D. F. Benoit, R. Al-Hamzawi, K. Yu, and D. V. den Poel. *bayesQR: Bayesian Quantile Regression*, 2014.  
URL <http://CRAN.R-project.org/package=bayesQR>. R package version 2.2. [p117]
- N. S. Boghossian, M. Geraci, E. M. Edwards, K. A. Morrow, and J. D. Horbar. Anthropometric charts for infants born between 22 and 29 weeks' gestation. *Pediatrics*, 2016. doi:10.1542/peds.2016-1641. [p131]
- M. Bottai and H. Zhen. Multiple imputation based on conditional quantile estimation. *Epidemiology, Biostatistics, and Public Health*, 10(1):e8758, 2013. [p135]
- G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society B*, 26(2):211–252, 1964. [p125]
- M. Buchinsky. Quantile regression, Box-Cox transformation model, and the US wage structure, 1963–1987. *Journal of Econometrics*, 65(1):109–154, 1995. [p125, 127]
- A. Canty and B. D. Ripley. *boot: Bootstrap R (S-PLUS) Functions*, 2014. URL <http://CRAN.R-project.org/package=boot>. R package version 1.3-15. [p127]
- G. Chamberlain. *Quantile Regression, Censoring, and the Structure of Wages*, volume 1. Cambridge University Press, Cambridge, UK, 1994. [p125, 127]
- V. Chernozhukov, I. Fernandez-Val, and A. Galichon. Quantile and probability curves without crossing. *Econometrica*, 78(3):1093–1125, 2010. [p134]
- A. C. Davison and D. V. Hinkley. *Bootstrap Methods and Their Applications*. Cambridge University Press, Cambridge, 1997. [p127]
- H.-M. Dehbi, M. Cortina-Borja, and M. Geraci. Aranda-Ordaz quantile regression for student performance assessment. *Journal of Applied Statistics*, 43(1):58–71, 2016. [p125]
- K. Doksum. Empirical probability plots and statistical inference for nonlinear models in the two-sample case. *The Annals of Statistics*, 2(2):267–277, 1974. [p122]
- M. Geraci. Linear quantile mixed models: The lqmm package for Laplace quantile regression. *Journal of Statistical Software*, 57(13):1–29, 2014. [p117]
- M. Geraci. Estimation of regression quantiles in complex surveys with data missing at random: An application to birthweight determinants. *Statistical Methods in Medical Research*, 25(4):1393–1421, 2016. [p135]
- M. Geraci and M. Bottai. Linear quantile mixed models. *Statistics and Computing*, 24(3):461–479, 2014. [p117]
- M. Geraci and M. C. Jones. Improved transformation-based quantile regression. *Canadian Journal of Statistics*, 43(1):118–132, 2015. [p125, 126, 127, 131]
- W. Gilchrist. *Statistical Modelling with Quantile Functions*. Chapman & Hall/CRC, Boca Raton, FL, 2000. [p118, 120]
- W. L. Goffe, G. D. Ferrier, and J. Rogers. Global optimization of statistical functions with simulated annealing. *Journal of Econometrics*, 60(1):65–99, 1994. [p135]
- W. Graybill, M. Chen, V. Chernozhukov, I. Fernandez-Val, and A. Galichon. *Rearrangement: Monotonize Point and Interval Functional Estimates by Rearrangement*, 2016. URL <http://CRAN.R-project.org/package=Rearrangement>. R package version 2.1. [p134]
- R. A. Groeneveld. A class of quantile measures for kurtosis. *The American Statistician*, 52(4):pp. 325–329, 1998. [p119, 120]
- R. A. Groeneveld and G. Meeden. Measuring skewness and kurtosis. *Journal of the Royal Statistical Society D*, 33(4):pp. 391–399, 1984. [p119]
- A. Hald. *A History of Probability and Statistics and their Applications before 1750*. John Wiley & Sons, New York, NY, 2003. [p117]
- X. He. Quantile curves without crossing. *The American Statistician*, 51(2):186–192, 1997. [p132]
- X. M. He and L. X. Zhu. A lack-of-fit test for quantile regression. *Journal of the American Statistical Association*, 98(464):1013–1022, 2003. [p123]

- P. S. Horn. A measure for peakedness. *The American Statistician*, 37(1):55–56, 1983. [p119]
- R. J. Hyndman and Y. Fan. Sample quantiles in statistical packages. *The American Statistician*, 50(4):361–365, 1996. [p118]
- M. C. Jones. Connecting distributions with power tails on the real line, the half line and the interval. *International Statistical Review*, 75(1):58–69, 2007. [p125]
- M. C. Jones, J. F. Rosco, and A. Pewsey. Skewness-invariant measures of kurtosis. *The American Statistician*, 65(2):89–95, 2011. [p119, 120]
- E. V. Khmaladze and H. L. Koul. Martingale transforms goodness-of-fit tests in regression models. pages 995–1034, 2004. [p123]
- R. Koenker. *Quantile Regression*. Cambridge University Press, New York, NY, 2005. [p117, 123, 127, 132]
- R. Koenker. *quantreg: Quantile Regression*, 2013. URL <http://CRAN.R-project.org/package=quantreg>. R package version 5.05. [p117]
- R. Koenker and G. Bassett. Regression quantiles. *Econometrica*, 46(1):33–50, 1978. [p122, 123, 134]
- R. Koenker and V. D’Orey. Algorithm AS 229: Computing regression quantiles. *Journal of the Royal Statistical Society C*, 36(3):383–393, 1987. [p117]
- R. Koenker and J. A. F. Machado. Goodness of fit and related inference processes for quantile regression. *Journal of the American Statistical Association*, 94(448):1296–1310, 1999. [p123]
- R. Koenker and B. J. Park. An interior point algorithm for nonlinear quantile regression. *Journal of Econometrics*, 71(1-2):265–283, 1996. [p117, 125]
- R. Koenker and Z. J. Xiao. Inference on the quantile regression process. *Econometrica*, 70(4):1583–1612, 2002. [p122, 123]
- E. L. Lehmann. *Nonparametrics: Statistical Methods Based on Ranks*. Holden-Day, San Francisco, CA, 1975. [p122]
- Y. Ma, M. G. Genton, and E. Parzen. Asymptotic properties of sample quantiles of discrete distributions. *Annals of the Institute of Statistical Mathematics*, 63(2):227–243, 2011. [p117, 118]
- J. A. F. Machado and J. Mata. Box–Cox quantile regression and the distribution of firm sizes. *Journal of Applied Econometrics*, 15(3):253–274, 2000. [p127]
- J. A. F. Machado and J. M. C. Santos Silva. Quantiles for counts. *Journal of the American Statistical Association*, 100(472):1226–1237, 2005. [p117, 134]
- C. F. Manski. Maximum score estimation of the stochastic utility model of choice. *Journal of Econometrics*, 3(3):205–228, 1975. [p135]
- C. F. Manski. Semiparametric analysis of discrete response: Asymptotic properties of the maximum score estimator. *Journal of Econometrics*, 27(3):313–333, 1985. [p135]
- Y. M. Mu and X. M. He. Power transformation toward a linear regression quantile. *Journal of the American Statistical Association*, 102(477):269–279, 2007. [p125, 127]
- J. F. Muñoz and M. Rueda. New imputation methods for missing data using quantiles. *Journal of Computational and Applied Mathematics*, 232(2):305–317, 2009. [p135]
- E. Parzen. Nonparametric statistical data modeling. *Journal of the American Statistical Association*, 74(365):105–121, 1979. [p118]
- E. Parzen. Quantile probability and statistical data modeling. *Statistical Science*, 19(4):652–662, 2004. [p117, 118]
- J. L. Powell. *Estimation of Monotonic Regression Models Under Quantile Restrictions*, pages 357–384. Cambridge University Press, New York, NY, 1991. [p125, 127]
- J. L. Powell. *Estimation of Semiparametric Models*, volume Volume 4, chapter 41, pages 2443–2521. Elsevier, 1994. [p123]
- D. Ruppert. What is kurtosis?: An influence function approach. *The American Statistician*, 41(1):1–5, 1987. [p119]

- R. M. Sakia. The Box–Cox transformation technique: A review. *Journal of the Royal Statistical Society D*, 41(2):169–178, 1992. [p125]
- L. Smith and B. Reich. *BSquare: Bayesian Simultaneous Quantile Regression*, 2013. URL <http://CRAN.R-project.org/package=BSquare>. R package version 1.1. [p117]
- R. G. Staudte. Inference for quantile measures of kurtosis, peakedness and tail-weight. *arXiv preprint arXiv:1047.6461v1 [math.ST]*, 2014. doi: 10.1080/03610926.2015.1056366. [p119]
- J. W. Tukey. Which part of the sample contains the information? *Proceedings of the National Academy of Sciences of the United States of America*, 53(1):127–134, 1965. [p118]
- S. van Buuren and K. Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, 45(3):1–67, 2011. [p135]
- K. M. Yu and M. C. Jones. Local linear quantile regression. *Journal of the American Statistical Association*, 93(441):228–237, 1998. [p125]
- Q. S. Zhao. Restricted regression quantiles. *Journal of Multivariate Analysis*, 72(1):78–99, 2000. [p132, 133]
- J. X. Zheng. A consistent nonparametric test of parametric regression models under conditional quantile restrictions. *Econometric Theory*, 14(1):123–138, 1998. [p123]

*Marco Geraci*

*Department of Epidemiology and Biostatistics  
Arnold School of Public Health, University of South Carolina  
915 Greene Street, Columbia SC 29204  
United States of America* [geraci@mailbox.sc.edu](mailto:geraci@mailbox.sc.edu)

# Two-Tier Latent Class IRT Models in R

by Silvia Bacci, Francesco Bartolucci

**Abstract** In analyzing data deriving from the administration of a questionnaire to a group of individuals, Item Response Theory (IRT) models provide a flexible framework to account for several aspects involved in the response process, such as the existence of multiple latent traits. In this paper, we focus on a class of semi-parametric multidimensional IRT models, in which these traits are represented through one or more discrete latent variables; these models allow us to cluster individuals into homogeneous latent classes and, at the same time, to properly study item characteristics. In particular, we follow a within-item multidimensional formulation similar to that adopted in the two-tier models, with each item measuring one or two latent traits. The proposed class of models may be estimated through the package **MLCIRTwithin**, whose functioning is illustrated in this paper with examples based on data about quality-of-life measurement and about the propensity to commit a crime.

## Introduction

Several fields of human knowledge require the measurement of unobservable constructs (or latent traits) through ad hoc methods based on questionnaires consisting of multiple items having dichotomously or ordered politomously scored response categories. This is the case of measurement of customer satisfaction, quality-of-life, level of physical and/or psychological disabilities, ability in certain subjects, and so on.

Item Response Theory (IRT) models (Hambleton and Swaminathan, 1985; Van der Linden and Hambleton, 1997; Bartolucci et al., 2015) are well-known statistical models to deal with these data. In their original formulation, these models are characterized by: (i) unidimensionality (i.e., only one latent trait is assumed to be measured by all items); (ii) a parametric (usually normal) distribution for the latent variables used to represent the trait of interest; and (iii) no effect of individual covariates on this latent trait. These elements often turn out to be restrictive in modern applications and, therefore, several extensions of IRT models have been proposed in the literature. Among the possible extensions, in this paper we consider the class of *multidimensional Latent Class (LC) IRT models* proposed by Bartolucci (2007) and von Davier (2008); see also Bacci et al. (2014). Models of this type are characterized by: (i) *multidimensionality*, in the sense that more latent traits may be measured by the set of items (Reckase, 2009); (ii) discreteness of the latent variables, so that homogeneous subpopulations (or latent classes; Lazarsfeld and Henry, 1968; Goodman, 1974) of individuals are detected with respect to the constructs measured by the questionnaire; and (iii) possible presence of individual covariates affecting the probabilities to belong to each latent class.

In particular, we focus on a specific extension of IRT models based on *within-item multidimensionality* (Adams et al., 1997), which is characterized by items affected by more than one latent variable. This is opposed to the more common *between-item multidimensionality*, where each item may measure only one latent variable as in the original approach of Bartolucci (2007). More in detail, the model here proposed represents a discrete version of the *item bifactor model* and of the more general *two-tier IRT model* (Bock et al., 1988; Gibbons and Hedeker, 1992; Gibbons et al., 2007; Cai, 2010; Cai et al., 2011; Reise, 2012; Bonifay, 2015), based on a particular within-item multidimensional formulation with each item loading on at most two latent variables that are mutually uncorrelated. With respect to traditional item bifactor and two-tier models, which assume the normality of the latent variables, the discreteness assumption increases the flexibility of the approach and allows us to cluster individuals in homogeneous latent classes. Formann and Kohlmann (2002) propose a general approach based on latent classes that includes the discrete two-tier model here proposed as special case. However, different from the proposal of these authors, we let the class membership probability depend on individual covariates and we also allow for more flexibility in terms of specification of model link function. Limited to binary items, a recent example of application of the proposed two-tier LC-IRT model is provided in Bacci and Bartolucci (2015) to jointly study certain students' abilities and the propensity to skipping item responses.

The procedures to estimate the proposed class of two-tier LC-IRT models are implemented in the R package **MLCIRTwithin** (Bartolucci and Bacci, 2016), downloadable from <http://CRAN.R-project.org/package=MLCIRTwithin>, whose illustration is the primary focus of the present paper. In particular, we are interested in providing a detailed description of the main functions of this package, named `est_multi_poly_within` and `search.model_within`, also through some applications.

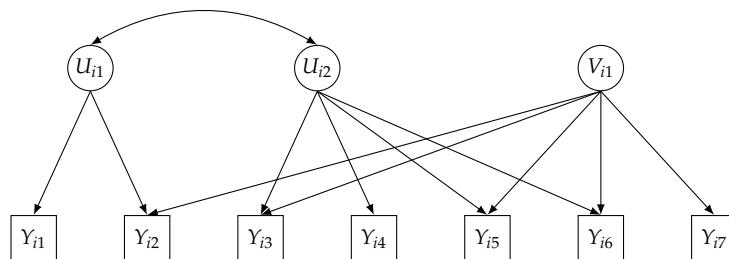
The remainder of the paper is organized as follows. In the next section we provide the formulation of the proposed class of two-tier LC-IRT models and, then, some details about likelihood inference for these models. Furthermore, we describe the main functions implemented in the R package **MLCIRTwithin** for model estimation. In the following, the functioning of the package is illustrated

through two applications: the first one concerns the measurement of Heath-related Quality Of Life (HQOL) on cancer patients and the second one is about the measurement of propensity to commit crimes. Some final remarks conclude the work.

## The class of models

The proposed class of models is formulated on the basis of two independent vectors of latent variables representing the unobservable individual characteristics measured by the test items. For each unit  $i = 1, \dots, n$ , these vectors are denoted by  $\mathbf{U}_i = (U_{i1}, \dots, U_{iD_1})'$  and  $\mathbf{V}_i = (V_{i1}, \dots, V_{iD_2})'$  and are of dimension  $D_1$  and  $D_2$ , respectively. Similarly to the item bifactor model, we assume that each item response  $Y_{ij}$ , with  $i = 1, \dots, n$  and  $j = 1, \dots, r$ , where  $r$  is the number of items, may depend on (and then measures) at most two latent variables, under the constraint that these two variables do not belong to the same vector. This is formalized by introducing the disjoint subsets  $\mathcal{U}_1, \dots, \mathcal{U}_{D_1}$  and  $\mathcal{V}_1, \dots, \mathcal{V}_{D_2}$  of  $\mathcal{J} = \{1, \dots, r\}$ , where  $\mathcal{U}_{d_1}$  contains the indices of the items depending on latent variable  $U_{id_1}$  and  $\mathcal{V}_{d_2}$  is the set of those depending on latent variable  $V_{id_2}$ . Equivalently,  $Y_{ij}$  depends on  $U_{id_1}$  if and only if  $j \in \mathcal{U}_{d_1}$  and on  $V_{id_2}$  if and only if  $j \in \mathcal{V}_{d_2}$ . Note that, even if the subsets  $\mathcal{U}_1, \dots, \mathcal{U}_{D_1}$  cannot overlap, and the same is assumed for  $\mathcal{V}_1, \dots, \mathcal{V}_{D_2}$ , the same item  $j$  may belong both to a set of the first type and to a set of the second type (within-item multidimensionality); more formally, there may exist  $d_1$  and  $d_2$  such that  $j \in \mathcal{U}_{d_1}$  and  $j \in \mathcal{V}_{d_2}$ . In practice, some items belonging to  $\mathcal{U}_{d_1}$ ,  $d_1 = 1, \dots, D_1$ , will be present also in  $\mathcal{V}_{d_2}$ ,  $d_2 = 1, \dots, D_2$ . With respect to the specification commonly encountered in the literature on item bifactor and two-tier models, our proposal is more general, as any value of  $D_1$  and  $D_2$  is allowed, whereas  $D_1 = 1$  (or, alternatively,  $D_2 = 1$ ) in the item bifactor model and  $D_1 = 2$  (or, alternatively,  $D_2 = 2$ ) in the two-tier model. Moreover, components of  $\mathbf{U}_i$  are allowed to be correlated; the same holds for components of  $\mathbf{V}_i$ .

An illustrative example of the above assumptions is provided in Figure 1, where  $D_1 = 2$ ,  $D_2 = 1$ , and four items out of  $r = 7$  measure two latent traits (item 2 measures dimensions  $U_{i1}$  and  $V_{i1}$ ; items 3, 5, and 6 measure dimensions  $U_{i2}$  and  $V_{i1}$ ); the two dimensions  $U_{i1}$  and  $U_{i2}$  do not share any item.



**Figure 1:** Path diagram of the proposed two-tier model for two latent vectors with two and one dimension, respectively, and seven items ( $\mathcal{U}_1 = \{1, 2\}$ ,  $\mathcal{U}_2 = \{3, 4, 5, 6\}$ ,  $\mathcal{V}_1 = \{2, 3, 5, 6, 7\}$ ).

Adopting a semi-parametric approach for the latent distribution, the first latent vector  $\mathbf{U}_i$  is assumed to have a discrete distribution based on  $k_1$  support points  $u_1, \dots, u_{k_1}$  and, in absence of individual covariates, common mass probabilities  $\lambda_1, \dots, \lambda_{k_1}$ . Similarly, the distribution of the second latent vector  $\mathbf{V}_i$  has  $k_2$  support points  $v_1, \dots, v_{k_2}$  and, again in absence of individual covariates, common mass probabilities  $\pi_1, \dots, \pi_{k_2}$ . In both cases, the support points identify classes of individuals that are homogeneous with respect to the latent traits represented by  $\mathbf{U}_i$  and  $\mathbf{V}_i$ . Note that cases with  $k_1 = 1$  or  $k_2 = 1$  detect a special situation in which vector of latent variables  $\mathbf{U}_i$  or  $\mathbf{V}_i$ , respectively, has no role in explaining the observed item responses.

For binary response variables, the *measurement model* assumes that, for  $i = 1, \dots, n$ ,  $j = 1, \dots, r$ ,  $h_1 = 1, \dots, k_1$ , and  $h_2 = 1, \dots, k_2$ ,

$$\text{logit } p(Y_{ij} = 1 | \mathbf{U}_i = \mathbf{u}_{h_1}, \mathbf{V}_i = \mathbf{v}_{h_2}) = \gamma_{1j} \sum_{d_1=1}^{D_1} 1\{j \in \mathcal{U}_{d_1}\} u_{h_1 d_1} + \gamma_{2j} \sum_{d_2=1}^{D_2} 1\{j \in \mathcal{V}_{d_2}\} v_{h_2 d_2} - \beta_j, \quad (1)$$

where  $1\{\cdot\}$  is the indicator function and  $\gamma_{1j}$ ,  $\gamma_{2j}$ , and  $\beta_j$  are suitable item parameters. As usual for IRT models,  $\gamma_{1j}$  and  $\gamma_{2j}$  represent the discrimination power of item  $j$  with respect to the latent variables in  $\mathbf{U}_i$  and  $\mathbf{V}_i$ , respectively, whereas  $\beta_j$  denotes the difficulty level of item  $j$ . In the previous expression,  $u_{h_1 d_1}$  denotes the  $d_1$ -th element of  $\mathbf{u}_{h_1}$ , whereas  $v_{h_2 d_2}$  denotes the  $d_2$ -th element of  $\mathbf{v}_{h_2}$ .

Different from traditional LC models characterized by constant mass probabilities, a more general approach is based on assuming that the probabilities to belong to every latent class defined by the

distribution of  $\mathbf{U}_i$  and  $\mathbf{V}_i$  depend on individual covariates, when such covariates are observed. For this aim, we denote the vector of covariates for individual  $i = 1, \dots, n$  by  $\mathbf{X}_i$  and we assume that  $\mathbf{U}_i$  and  $\mathbf{V}_i$  are conditionally independent given  $\mathbf{X}_i$ . Moreover, we adopt the following multinomial logit parametrization (Formann, 2007) for each latent vector:

$$\log \frac{\lambda_{h_1}(\mathbf{x}_i)}{\lambda_1(\mathbf{x}_i)} = \mathbf{x}'_i \boldsymbol{\delta}_{1h_1}, \quad h_1 = 2, \dots, k_1, \quad (2)$$

$$\log \frac{\pi_{h_2}(\mathbf{x}_i)}{\pi_1(\mathbf{x}_i)} = \mathbf{x}'_i \boldsymbol{\delta}_{2h_2}, \quad h_2 = 2, \dots, k_2, \quad (3)$$

with  $\lambda_{h_1}(\mathbf{x}_i) = p(\mathbf{U}_i = \mathbf{u}_{h_1} | \mathbf{X}_i = \mathbf{x}_i)$  and  $\pi_{h_2}(\mathbf{x}_i) = p(\mathbf{V}_i = \mathbf{v}_{h_2} | \mathbf{X}_i = \mathbf{x}_i)$ , where  $\mathbf{x}_i$  contains the constant term. The vectors of coefficients  $\boldsymbol{\delta}_{1h_1}$  and  $\boldsymbol{\delta}_{2h_2}$  measure the effect of the covariates on the logit to belong to class  $h_1 = 2, \dots, k_1$  and  $h_2 = 2, \dots, k_2$ , with respect to class  $h_1 = 1$  and  $h_2 = 1$ , respectively. Alternatively, a global logit formulation may be adopted. This is related to a cumulative logit formulation (Agresti, 2013), where the logits in equations (2) and (3) are substituted with

$$\log \frac{p(\mathbf{U}_i \geq \mathbf{u}_{h_1} | \mathbf{X}_i = \mathbf{x}_i)}{p(\mathbf{U}_i < \mathbf{u}_{h_1} | \mathbf{X}_i = \mathbf{x}_i)} = \log \frac{\lambda_{h_1}(\mathbf{x}_i) + \dots + \lambda_{k_1}(\mathbf{x}_i)}{\lambda_1(\mathbf{x}_i) + \dots + \lambda_{h_1-1}(\mathbf{x}_i)}, \quad h_1 = 2, \dots, k_1$$

and

$$\log \frac{p(\mathbf{V}_i \geq \mathbf{v}_{h_2} | \mathbf{X}_i = \mathbf{x}_i)}{p(\mathbf{V}_i < \mathbf{v}_{h_2} | \mathbf{X}_i = \mathbf{x}_i)} = \log \frac{\pi_{h_2}(\mathbf{x}_i) + \dots + \pi_{k_2}(\mathbf{x}_i)}{\pi_1(\mathbf{x}_i) + \dots + \pi_{h_2-1}(\mathbf{x}_i)}, \quad h_2 = 2, \dots, k_2,$$

respectively. The main advantage of the global logit parametrization is the easier interpretation of the regression coefficients that now refer to the effect of the covariates on the logit to belong to a specific class (or higher) with respect to a lower class. However, this parameterization requires the latent classes to be ordered according to a specific criterion (e.g., requiring an increasing trend of the support points for a given dimension).

In the case of polytomously scored items with ordered categories indexed from 0 to  $l_j - 1$ , the model based on Equation (1) may be extended according to a global logit link function, so that a graded response model (Samejima, 1969) results in:

$$\log \frac{p(Y_{ij} \geq y | \mathbf{U}_i = \mathbf{u}_{h_1}, \mathbf{V}_i = \mathbf{v}_{h_2})}{p(Y_{ij} < y | \mathbf{U}_i = \mathbf{u}_{h_1}, \mathbf{V}_i = \mathbf{v}_{h_2})} = \gamma_{1j} \sum_{d_1=1}^{D_1} 1\{j \in \mathcal{U}_{d_1}\} u_{h_1 d_1} + \gamma_{2j} \sum_{d_2=1}^{D_2} 1\{j \in \mathcal{V}_{d_2}\} v_{h_2 d_2} - \beta_{jy}. \quad (4)$$

Alternatively, using a local logit link function, we may assume a partial credit model (Masters, 1982):

$$\log \frac{p(Y_{ij} = y | \mathbf{U}_i = \mathbf{u}_{h_1}, \mathbf{V}_i = \mathbf{v}_{h_2})}{p(Y_{ij} = y-1 | \mathbf{U}_i = \mathbf{u}_{h_1}, \mathbf{V}_i = \mathbf{v}_{h_2})} = \gamma_{1j} \sum_{d_1=1}^{D_1} 1\{j \in \mathcal{U}_{d_1}\} u_{h_1 d_1} + \gamma_{2j} \sum_{d_2=1}^{D_2} 1\{j \in \mathcal{V}_{d_2}\} v_{h_2 d_2} - \beta_{jy}. \quad (5)$$

In the above expressions,  $y = 1, \dots, l_j - 1$  and the difficulty parameter  $\beta_{jy}$  is now specific of item  $j$  and response category  $y$ . A more parsimonious model is obtained by expressing  $\beta_{jy}$  as the sum of two components (rating scale parametrization; Andrich, 1978), that is,

$$\beta_{jy} = \beta_j + \tau_y, \quad j = 1, \dots, r; y = 1, \dots, l_j - 1, \quad (6)$$

so that the distance in terms of difficulty from category to category (i.e.,  $\tau_y$ ) is the same for all items. Note that the rating scale parametrization is allowed only when items have the same number of response categories (i.e.,  $l_j = l$ ,  $j = 1, \dots, r$ ). For more details about the possible item parametrizations in the presence of ordinal items see Bacci et al. (2014) and Bartolucci et al. (2015).

In order to ensure the identification of the proposed class of models, two necessary conditions must hold. First, as usual in the IRT modeling, we must constrain one discriminant index to be equal to 1 and one difficulty parameter to be equal to 0 for each dimension. More in detail, let  $j_{d_1}$  be a specific element of  $\mathcal{U}_{d_1}$  and  $j_{d_2}$  a specific element of  $\mathcal{V}_{d_2}$  for  $d_1 = 1, \dots, D_1$  and  $d_2 = 1, \dots, D_2$ . Then we assume  $\gamma_{1j_{d_1}} = 1$ ,  $\gamma_{2j_{d_2}} = 1$ , and, when item difficulties are free,  $\beta_{j_{d_1}1} = 0$  and  $\beta_{j_{d_2}1} = 0$ , whereas in the presence of a rating scale parametrization we assume  $\beta_{j_{d_1}1} = 0$ ,  $\beta_{j_{d_2}1} = 0$ , and  $\tau_1 = 0$ . In the case of binary items, constraints on difficulties simplify to  $\beta_{j_{d_1}1} = 0$  and  $\beta_{j_{d_2}1} = 0$ . Generally speaking,  $j_{d_1}$  and  $j_{d_2}$  may be chosen in an arbitrary way, paying attention to select a different item for each dimension. So, in the example illustrated in Figure 1, if we constrain item  $j = 1$  for dimension  $U_{i1}$  and item  $j = 3$  for dimension  $U_{i2}$ , then for dimension  $V_{i1}$  we may constrain any one of the items in the subset  $\mathcal{V}_1$  with the only exception of item  $j = 3$ . As an alternative to constraining item parameters, we may fix the support points, as in the general diagnostic model of von Davier (2008).

A further identification condition requires that at least one item belongs to one of the subsets  $\mathcal{U}_{d_1}$  or

to one of the subsets  $\mathcal{V}_{d_2}$ ; more formally, the union of  $\mathcal{U}_1, \dots, \mathcal{U}_{D_1}$  must be different from the union of  $\mathcal{V}_1, \dots, \mathcal{V}_{D_2}$ . In other words, we restrict  $\gamma_{1j} = 0$  or  $\gamma_{2j} = 0$  for at least one  $j$  and the maximum number of items shared by  $\mathbf{U}_i$  and  $\mathbf{V}_i$  is equal to  $r - 1$ . Alternatively, we may skip this restrictive condition by specifying in a suitable way linear constraints (e.g., equality restrictions) on some discriminant parameters (for some examples see Cai, 2010; Cai et al., 2011).

To specify in a flexible way constraints on the support points and item parameters, we denote the complete vectors of support points by  $\mathbf{u} = (u_{11}, u_{12}, \dots, u_{k_1 D_1})'$  for latent variable  $\mathbf{U}_i$  and  $\mathbf{v} = (v_{11}, v_{12}, \dots, v_{k_2 D_2})'$  for latent variable  $\mathbf{V}_i$ , the complete vectors of item discriminating indices as  $\boldsymbol{\gamma}_1 = (\gamma_{11}, \dots, \gamma_{1r})'$  for items affected by  $\mathbf{U}_i$  (then  $\gamma_{1j}$  is missing if item  $j$  does not belong to  $\mathcal{U}_1, \dots, \mathcal{U}_{D_1}$ ) and  $\boldsymbol{\gamma}_2 = (\gamma_{21}, \dots, \gamma_{2r})'$  for items affected by  $\mathbf{V}_i$  (then  $\gamma_{2j}$  is missing if item  $j$  does not belong to  $\mathcal{V}_1, \dots, \mathcal{V}_{D_2}$ ), and the complete vector of item difficulties as  $\boldsymbol{\beta} = (\beta_{11}, \dots, \beta_{r, l_r - 1})'$  (or  $\boldsymbol{\beta} = (\beta_1, \dots, \beta_r)'$  in the binary case). The corresponding vectors of free support points and free item parameters are denoted by  $\tilde{\mathbf{u}}$ ,  $\tilde{\mathbf{v}}$ ,  $\tilde{\boldsymbol{\gamma}}_1$ ,  $\tilde{\boldsymbol{\gamma}}_2$ , and  $\tilde{\boldsymbol{\beta}}$ , respectively. A wide range of linear constraints and fixed values of the parameters are specified through a suitable definition of matrices  $\mathbf{Z}_u$ ,  $\mathbf{Z}_v$ ,  $\mathbf{Z}_{\boldsymbol{\gamma}_1}$ ,  $\mathbf{Z}_{\boldsymbol{\gamma}_2}$ , and  $\mathbf{Z}_{\boldsymbol{\beta}}$  and vectors  $\mathbf{z}_u$ ,  $\mathbf{z}_v$ ,  $\mathbf{z}_{\boldsymbol{\gamma}_1}$ ,  $\mathbf{z}_{\boldsymbol{\gamma}_2}$ , and  $\mathbf{z}_{\boldsymbol{\beta}}$ , as follows:

$$\mathbf{u} = \mathbf{Z}_u \tilde{\mathbf{u}} + \mathbf{z}_u, \quad (7)$$

$$\mathbf{v} = \mathbf{Z}_v \tilde{\mathbf{v}} + \mathbf{z}_v, \quad (8)$$

$$\boldsymbol{\gamma}_1 = \mathbf{Z}_{\boldsymbol{\gamma}_1} \tilde{\boldsymbol{\gamma}}_1 + \mathbf{z}_{\boldsymbol{\gamma}_1}, \quad (9)$$

$$\boldsymbol{\gamma}_2 = \mathbf{Z}_{\boldsymbol{\gamma}_2} \tilde{\boldsymbol{\gamma}}_2 + \mathbf{z}_{\boldsymbol{\gamma}_2}, \quad (10)$$

$$\boldsymbol{\beta} = \mathbf{Z}_{\boldsymbol{\beta}} \tilde{\boldsymbol{\beta}} + \mathbf{z}_{\boldsymbol{\beta}}. \quad (11)$$

For instance, according to the usual IRT parametrization with free support points and constraints on the item parameters (i.e., one discriminant index equal to 1 and one difficulty parameter equal to 0 for each dimension),  $\mathbf{Z}_u$  and  $\mathbf{Z}_v$  are identity matrices of dimensions  $k_1 D_1 \times k_1 D_1$  and  $k_2 D_2 \times k_2 D_2$ , respectively, and  $\mathbf{z}_u$  and  $\mathbf{z}_v$  are null vectors. Moreover, matrices  $\mathbf{Z}_{\boldsymbol{\gamma}_1}$ ,  $\mathbf{Z}_{\boldsymbol{\gamma}_2}$ , and  $\mathbf{Z}_{\boldsymbol{\beta}}$  are defined as identity matrices without those columns corresponding to the constrained item parameters, whereas  $\mathbf{z}_{\boldsymbol{\gamma}_1}$  and  $\mathbf{z}_{\boldsymbol{\gamma}_2}$  are vectors with ones in correspondence of constrained item discrimination parameters and zeros otherwise;  $\mathbf{z}_{\boldsymbol{\beta}}$  is a vector of zeros. Further examples of specification of constraints on model parameters are provided in the sequel, when the functioning of the estimation functions of the proposed R package and an example on criminal data (Example 2) are illustrated.

## Likelihood inference

The proposed two-tier LC-IRT model can be estimated by maximizing the *marginal log-likelihood*

$$\ell(\boldsymbol{\eta}) = \sum_{i=1}^n \log L_i(\mathbf{y}_i | \mathbf{x}_i), \quad (12)$$

where  $\boldsymbol{\eta}$  is the vector of free model parameters, that is, support points of  $\mathbf{U}_i$  and  $\mathbf{V}_i$ , item difficulty and discrimination parameters, and regression coefficients for the covariates; in the previous expression,  $\mathbf{y}_i = (y_{i1}, \dots, y_{ir})'$  is the vector of observed item responses for subject  $i$ . Due to the *local independence* assumption, the marginal likelihood  $L_i(\mathbf{y}_i | \mathbf{x}_i)$  for subject  $i$  (or manifest probability of  $\mathbf{y}_i$ ) used in equation (12) is given by:

$$L_i(\mathbf{y}_i | \mathbf{x}_i) = \sum_{h_1=1}^{k_1} \sum_{h_2=1}^{k_2} \lambda_{h_1}(\mathbf{x}_i) \pi_{h_2}(\mathbf{x}_i) \prod_{j=1}^r p_{h_1 h_2}(y_{ij}),$$

where,  $p_{h_1 h_2}(y) = p(Y_{ij} = y | \mathbf{U}_i = \mathbf{u}_{h_1}, \mathbf{V}_i = \mathbf{v}_{h_2})$ , which depends on (1) in the case of binary items and on (4) or (5) in the case of ordinal items, if a global logit or a local logit parametrization is adopted.

We maximize  $\ell(\boldsymbol{\eta})$  through the Expectation Maximization (EM) algorithm (Dempster et al., 1977), which is based on alternating two steps until convergence:

**E-step:** the expected value of the *complete data log-likelihood* (i.e., the log-likelihood for the observed and latent variables) is computed, given the current parameter vector  $\boldsymbol{\eta}$ . In practice, this consists in computing the posterior probability  $q_{h_1 h_2 i}$  for  $h_1 = 1, \dots, k_1$ ,  $h_2 = 1, \dots, k_2$ , and  $i = 1, \dots, n$ ; this is the probability that unit  $i$  belongs to latent class  $h_1$ , according to the first vector of latent variables, and to latent class  $h_2$ , according to the second vector of latent variables, given the observed data, that is,  $p(\mathbf{U}_i = \mathbf{u}_{h_1}, \mathbf{V}_i = \mathbf{v}_{h_2} | \mathbf{x}_i, \mathbf{y}_i)$ . By the Bayes' theorem we have that

$$q_{h_1 h_2 i} = \frac{\lambda_{h_1}(\mathbf{x}_i) \pi_{h_2}(\mathbf{x}_i) \prod_{j=1}^r p_{h_1 h_2}(y_{ij})}{L_i(\mathbf{y}_i | \mathbf{x}_i)}.$$

The resulting complete data log-likelihood is, in expected value, equal to

$$\ell^*(\boldsymbol{\eta}) = \sum_{h_1=1}^{k_1} \sum_{h_2=1}^{k_2} \sum_{i=1}^n q_{h_1 h_2 i} \log \left[ \lambda_{h_1}(\mathbf{x}_i) \pi_{h_2}(\mathbf{x}_i) \prod_{j=1}^r p_{h_1 h_2}(y_{ij}) \right]. \quad (13)$$

**M-step:** the parameter vector  $\boldsymbol{\eta}$  is updated by maximizing function (13) obtained at the previous step. Note that single parameter subvectors of  $\boldsymbol{\eta}$  may be updated separately, as this function factorizes in three components involving the mass probabilities  $\lambda_{h_1}(\mathbf{x}_i)$ , the mass probabilities  $\pi_{h_2}(\mathbf{x}_i)$ , and the conditional response probabilities  $p_{h_1 h_2}(y_{ij})$ , respectively. Iterative algorithms of Newton-Raphson type are necessary to maximize all components (for details see [Bacci and Bartolucci, 2015](#), and references therein) with the exception of the first two in the case of absence of individual covariates. In fact, in this case we have the following explicit expressions to update the class weights:

$$\begin{aligned} \lambda_{h_1} &= \frac{1}{n} \sum_{h_2=1}^{k_2} \sum_{i=1}^n q_{h_1 h_2 i}, \quad h_1 = 1, \dots, k_1, \\ \pi_{h_2} &= \frac{1}{n} \sum_{h_1=1}^{k_1} \sum_{i=1}^n q_{h_1 h_2 i}, \quad h_2 = 1, \dots, k_2. \end{aligned}$$

Similar to the other iterative algorithms, the first iteration of the EM algorithm needs to be initialized through suitable values for the model parameters that can be chosen according to certain deterministic or random rules. A common problem with finite mixture models, and then with the proposed model, is due to the presence of several local maximum points of the log-likelihood function. Therefore, in order to avoid a solution that does not correspond to the global maximum, a good practice consists in repeating the estimation process for a specific model a certain number of times using random starting values and, in the presence of different values of the log-likelihood at convergence, the solution corresponding to the highest log-likelihood value is selected.

A crucial point is that of model selection, mainly as concerns the choice of the number of support points (or latent classes) for both latent vectors (i.e.,  $k_1$  and  $k_2$ ). For this aim, a likelihood-ratio test cannot be directly used, as the regularity conditions for having an asymptotic null distribution of  $\chi^2$ -type are not satisfied for this type of test when it is applied to compare two models with different values of  $k_1$  and  $k_2$ . We then suggest to rely on suitable forms of penalization of the maximum log-likelihood, such as the Akaike Information Criterion (AIC; [Akaike, 1973](#)), which is related to Kullback-Leibler distance between the true density and the estimated density of a model. This criterion is based on the following index:

$$AIC = -2\hat{\ell}(\boldsymbol{\eta}) + 2\#\text{par},$$

with  $\hat{\ell}(\cdot)$  denoting the estimated maximum log-likelihood and  $\#\text{par}$  the number of free parameters. Alternatively, we suggest the use of the Bayesian Information Criterion (BIC; [Schwarz, 1978](#)) based on the index

$$BIC = -2\hat{\ell}(\boldsymbol{\eta}) + \log(n)\#\text{par}.$$

According to both these criteria, one should select the model with the minimum value of AIC or BIC.

Other selection criteria may be based on the entropy, whose computation involves the individual posterior probabilities. Entropy is a measure of the capability of the model to provide a neat partition of the sample units, which is computed as

$$E = - \sum_{h_1=1}^{k_1} \sum_{h_2=1}^{k_2} \sum_{i=1}^n q_{h_1 h_2 i} \log q_{h_1 h_2 i},$$

based on the posterior probabilities  $q_{h_1 h_2 i}$ . If the components are well separated, the posterior probabilities tend to define a clear partition of the units, assuming values close to one, and, as a consequence, the entropy will be close to zero. Usually, the entropy is not directly used to assess the number of support points and, to also account for the goodness of fit of the model, a normalized version of entropy is used by [Celeux and Soromenho \(1996\)](#). This is defined as

$$NEC = \frac{E}{\hat{\ell}_{k_1 k_2} - \hat{\ell}_{11}}, \quad k_1 > 1, k_2 > 1,$$

where  $\hat{\ell}_{k_1 k_2}$  is the maximum log-likelihood of the model with  $k_1$  and  $k_2$  support points and  $\hat{\ell}_{11}$  is the maximum log-likelihood value of the model with just one component for both latent variables. According to this criterion, the optimal number of components is the one that minimizes the NEC index. Note that NEC is not defined when  $k_1 = k_2 = 1$ , in which case  $NEC = 1$  by convention.

In practice, we propose to fit a series of models with similar specifications that distinguish one other for values assigned to  $k_1$  and  $k_2$  and, then, to make comparisons through one or more of the mentioned criteria. In more detail, given  $k_1$ , we consider increasing values of  $k_2$  and, similarly, given  $k_2$  we consider increasing values of  $k_1$  until *AIC*, *BIC*, or *NEC* do not start to increase and, then, the previous value of support points is taken as the optimal one.

More in general, for certain values of  $k_1$  and  $k_2$  we suggest that the choice between two competing models is driven by the likelihood-ratio test in the presence of nested models (i.e., when one model is obtained by the other one through constraints on the parameters), whereas the selection criteria above mentioned are suitable in the presence of non-nested models. The likelihood-ratio test is also used to evaluate the global fit of a model, when this is compared with the saturated model, that is, the largest model one can fit. Note that in the context at issue the saturated model is defined only for model specifications without covariates. Other proposals, coming from the literature on logistic regression models and on IRT models, consist of parametric and non-parametric tests that allow us to verify specific hypotheses concerning, among others, the unidimensionality of the questionnaire, the validity of the Rasch paradigm, the validity of the local independence assumption. In addition to the global fit of a model, also item-specific fit statistics, which are usually based on the comparison between observed and expected item responses, are useful to evaluate the goodness of each item and the need of removing it from the questionnaire. For a wide review of the mentioned methods see [Bartolucci et al. \(2015\)](#), Chap. 5.7, and the references therein.

Finally, in order to facilitate the interpretation of the results, we suggest to standardize the estimated support points  $\hat{u}_{d_1 h_1}$  and  $\hat{v}_{d_2 h_2}$ , so as to obtain latent variables that have mean 0 and variance equal to 1, and coherently transform the estimated item parameters  $\hat{\gamma}_{1j}$ ,  $\hat{\gamma}_{2j}$ , and  $\hat{\beta}_{jy}$ .

Let  $\hat{\mu}_{U_{d_1}}$  and  $\hat{\sigma}_{U_{d_1}}$  denote the mean and the standard deviation of  $\hat{u}_{d_1 1}, \dots, \hat{u}_{d_1 k_1}$  and let  $\hat{\mu}_{V_{d_2}}$  and  $\hat{\sigma}_{V_{d_2}}$  denote the mean and the standard deviation of  $\hat{v}_{d_2 1}, \dots, \hat{v}_{d_2 k_2}$ . Then,  $\hat{u}_{d_1 h_1}$  and  $\hat{v}_{d_2 h_2}$  are standardized as follows:

$$\hat{u}_{d_1 h_1}^* = \frac{\hat{u}_{d_1 h_1} - \hat{\mu}_{U_{d_1}}}{\hat{\sigma}_{U_{d_1}}}, \quad d_1 = 1, \dots, D_1, \quad (14)$$

$$\hat{v}_{d_2 h_2}^* = \frac{\hat{v}_{d_2 h_2} - \hat{\mu}_{V_{d_2}}}{\hat{\sigma}_{V_{d_2}}}, \quad d_2 = 1, \dots, D_2. \quad (15)$$

Moreover,  $\hat{\gamma}_{1j}$ ,  $\hat{\gamma}_{2j}$ , and  $\hat{\beta}_{jy}$  are transformed as

$$\hat{\gamma}_{1j}^* = \hat{\gamma}_{1j} \sum_{d_1=1}^{D_1} 1\{j \in \mathcal{U}_{d_1}\} \hat{\sigma}_{U_{d_1}}, \quad (16)$$

$$\hat{\gamma}_{2j}^* = \hat{\gamma}_{2j} \sum_{d_2=1}^{D_2} 1\{j \in \mathcal{V}_{d_2}\} \hat{\sigma}_{V_{d_2}}, \quad (17)$$

$$\hat{\beta}_{jy}^* = \hat{\beta}_{jy} - \hat{\gamma}_{1j} \sum_{d_1=1}^{D_1} 1\{j \in \mathcal{U}_{d_1}\} \hat{\mu}_{U_{d_1}} - \hat{\gamma}_{2j} \sum_{d_2=1}^{D_2} 1\{j \in \mathcal{V}_{d_2}\} \hat{\mu}_{V_{d_2}}, \quad (18)$$

for  $j = 1, \dots, r$  and  $y = 0, \dots, l_j - 1$ , with  $l_j = 2$  for a dichotomously scored item.

## The R package MLCIRTwithin

The class of two-tier LC-IRT models previously described may be estimated through the R package **MLCIRTwithin**; for technical details see the official documentation provided in CRAN ([Bartolucci and Bacci, 2016](#)).

Before illustrating the main functions in the package at issue, it is worth mentioning some alternative R packages, which estimate models with a formulation resembling the one proposed. A first example is provided by the R package **MultiLCIRT** ([Bartolucci et al., 2014, 2016](#)), whose functions are similar, in terms of input and output, to those of **MLCIRTwithin**; however, **MultiLCIRT** is limited to the estimation of LC-IRT models under between-item multidimensionality, in the sense that items loading on more than one latent trait are not allowed and then a single vector of latent variables  $\mathbf{U}_i$  is used. Moreover, constraints on the item parameters or fixed values for the support points cannot be specified. Package **CDM** ([Robitzsch et al., 2016](#)) performs the estimation of the class of cognitive diagnostic models ([Tatsuoka, 1983; Jang, 2008; Rupp and Templin, 2008](#)), in which the proposed, discrete, two-tier model may be included. The class of models estimated through **CDM** may be characterized, among the main options, by normally distributed latent traits or, alternatively, discrete

latent traits whose support points may be freely estimated or may be specified as fixed values. As concerns item parameters, item-by-category specific slopes as well as linear constraints are allowed. However, different from our proposal, individual covariates affecting the class membership as well as the specification of a global logit link are not allowed. Moreover, attention must be paid to the interpretation of the latent classes, which are defined in a quite different and general way with respect to our proposal. The estimation of within-item multidimensional IRT models and item bifactor models is also performed through packages **mirt** (Chalmers, 2012; Chalmers et al., 2016) and **flirt** (Jeon et al., 2014), under the assumption of normally distributed latent variables. Package **mirt** also allows for discrete latent variables; however, in such a case just the multidimensional LC model without a classical IRT parametrization (mainly, without item difficulties) and without covariates is estimated. A major flexibility with respect to **mirt** is provided by package **covLCA** (Bertrand and Hafner, 2013), which is focused on multidimensional LC models with covariates affecting both the class membership and the manifest variables. Other two packages to mention are **lavaan** (Rosseel et al., 2015) and **OpenMx** (Neale et al., 2016) that perform the estimation of the wide class of structural equation models, in which unidimensional and multidimensional IRT models are included, under the assumption of normality of the latent variables. Finally, we mention two general and flexible softwares that may accommodate the estimation of the model here proposed, that is, **Mplus** (Muthén and Muthén, 2012) and **LatentGold** (Vermunt and Magidson, 2005): the former is tailored to the estimation of latent variable models under the assumption of normal or discrete latent variables, whereas the latter is focused on LC models. In both cases, the user may formulate IRT models with a variety of features, among which multidimensionality and presence of covariates.

### Functions `est_multi_poly_within` and `est_multi_poly_between`

The main function of **MLCIRTwithin** is `est_multi_poly_within`, which performs the maximum likelihood estimation of the model specified through equations (1) to (5), allowing for several options.

Function `est_multi_poly_within` requires the following main input arguments:

- `S`: matrix of item response configurations listed row-by-row; items with a different number of categories and missing responses are allowed.
- `yv`: vector of the frequencies of every row in `S`; by default, `yv` is a vector of ones.
- `k1`: number of latent classes for latent variable  $U_i$ .
- `k2`: number of latent classes for latent variable  $V_i$ .
- `X`: matrix of covariates affecting the class weights; by default, `X` is `NULL`.
- `start`: method of initialization of the algorithm: "deterministic" (default value) for values chosen according to a deterministic rule, "random" for values randomly drawn from suitable distributions (continuous uniform between 0 and 1 for the class weights and standard normal for the other parameters), and "external" for values provided by the researcher through inputs `Phi`, `ga1t`, `ga2t`, `De1`, and `De2`.
- `link`: type of link function: "global" for global logits as in Equation (4) and "local" for local logits as in Equation (5). With binary items, any type of link function may be specified, resulting in a Rasch (Rasch, 1960) or a two-parameter logit (2PL; Birnbaum, 1968) type model depending on the value assigned to input `disc`.
- `disc`: constraints on the discriminating item parameters: `FALSE` (default value) for parameters  $\gamma_{1j}$  and  $\gamma_{2j}$  all equal to one and `TRUE` for free values. With binary items, option `disc = FALSE` results in a Rasch model, whereas a 2PL model is obtained when option `disc = TRUE`.
- `dif1`: constraints on the difficulty item parameters, in the case of ordinal polytomously scored items: `FALSE` (default value) for unconstrained parameters  $\beta_{jy}$  and `TRUE` for a rating scale parametrization as in (6). This option is not allowed in the presence of items with a different number of response categories.
- `multi1`: matrix with one row for each component of  $U_i$  and elements in each cell indicating the indices of the items measuring the dimension corresponding to that row; the number of rows is  $D_1$  and that of columns equals the number of items in the largest dimension. If dimensions differ in the number of items, zeros are inserted in the empty cells. Each item corresponding to the first column of each row has discriminating index constrained to 1 and difficulty parameter constrained to 0 to ensure model identifiability. For instance, in the presence of 6 items, with items 1 and 2 measuring the first dimension of latent variable  $U_i$  and the remaining items 3 to 6 measuring another dimension of  $U_i$ , as in Figure 1, matrix `multi1` is specified as

```
(multi1 <- rbind(c(1,2,0,0), c(3,4,5,6)))
```

```
[,1] [,2] [,3] [,4]
[1,]    1    2    0    0
[2,]    3    4    5    6
```

- `multi2`: same as `multi1` for latent variable  $V_i$ . For model identifiability, attention must be payed on the item indices in the first column of `multi2` that cannot be the same as the indices in the first column of `multi1`. For instance, in the situation described in Figure 1, the matrix `multi2` specified as

```
(multi2 <- c(7, 2:3, 5:6))
```

```
[1] 7 2 3 5 6
```

implies  $\gamma_{27} = 1$  and  $\beta_7 = 0$ . A particular case is when the intersection between matrices `multi1` and `multi2` is empty: in such a case a between-item multidimensional LC-IRT model is specified, based on two completely independent latent vectors  $U_i$  and  $V_i$ .

- `fort`: if TRUE, Fortran routines are used whenever possible to speed up computation.
- `tol`: level of tolerance of the algorithm in terms of relative difference between the log-likelihood corresponding to two consecutive algorithm iterations (default value is  $10^{-10}$ ).
- `disp`: if TRUE, the log-likelihood evolution is displayed step-by-step.
- `output`: if TRUE, additional output arguments are returned.
- `out_se`: if TRUE, standard errors and variance-covariance matrix for the parameter estimates are returned.
- `glob`: type of parametrization for the sub-model assumed on the individual-specific latent class weights: FALSE (default value) for a multinomial logit model as in (2)-(3) and TRUE for a global logit model.
- `Zth1, Zth2`: matrices for the specification of linear constraints on the support points, according to (7) and (8), respectively; by default these are identity matrices with a number of rows (and columns) equal to the total number of support points, that is,  $k_1 D_1$  and  $k_2 D_2$ , respectively.
- `zth1, zth2`: vectors of length  $k_1 D_1$  and  $k_2 D_2$ , respectively, for the specification of linear constraints and fixed support points, according to (7) and (8), respectively; by default they are null vectors.

Under the default specifications of `Zth1`, `Zth2`, `zth1`, and `zth2`, the support points are freely estimated and, for the model identification, certain constraints are assumed on the item parameters. On the contrary, to fix the values of the support points, `Zth1`, `Zth2`, `zth1`, and `zth2` must be supplied by the user. For instance, in the situation described in Figure 1 under the assumption  $k_1 = k_2 = 2$ , we define  $u = (-1, -1, 1, 1)'$  and  $v = (-0.5, 0.5)'$  as follows:

```
Zth1 <- matrix(0,2*2,0)
zth1 <- c(rep(-1, times=2), rep(1, times = 2))
Zth2 <- matrix(0,2,0)
zth2 <- c(-0.5,0.5)
```

- `Zga1, Zga2, Zbe`: matrices for the specification of linear constraints on the vectors of item parameters  $\gamma_1$ ,  $\gamma_2$ , and  $\beta$ , as in (9), (10) and (11), respectively. In more detail, the number of rows of `Zga1` and `Zga2` is equal to the number of non-null entries in `multi1` and `multi2`, respectively, and coincides with the length of vectors  $\gamma_1$  and  $\gamma_2$ ; whereas the number of rows of `Zbe` corresponds to the total number of item difficulties and coincides with the length of vector  $\beta$ . The number of columns of `Zga1`, `Zga2`, `Zbe` is equal to the total number of free parameters, corresponding to the length of vectors  $\tilde{\gamma}_1$ ,  $\tilde{\gamma}_2$ , and  $\tilde{\beta}$ , respectively. By default these are identity matrices without those columns corresponding to the constrained parameters. For instance, in the situation described in Figure 1 with the usual IRT constraints  $\gamma_{11} = \gamma_{13} = 1$  and  $\gamma_{27} = 1$  resulting from matrices `multi1` and `multi2` defined above, and  $\beta_1 = \beta_3 = \beta_7 = 0$  in the case of binary items, the following matrices are used by default in function `est_multi_poly_within`:

```
(Zga1 <- diag(6)[, -c(1,3)])
[,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0
[2,]    1    0    0    0
[3,]    0    0    0    0
[4,]    0    1    0    0
[5,]    0    0    1    0
[6,]    0    0    0    1
```

```
(Zga2 <- diag(5)[, -5])
[,1] [,2] [,3] [,4]
[1,] 1 0 0 0
[2,] 0 1 0 0
[3,] 0 0 1 0
[4,] 0 0 0 1
[5,] 0 0 0 0

(Zbe <- diag(7)[, -c(1,3,7)])
[,1] [,2] [,3] [,4]
[1,] 0 0 0 0
[2,] 1 0 0 0
[3,] 0 0 0 0
[4,] 0 1 0 0
[5,] 0 0 1 0
[6,] 0 0 0 1
[7,] 0 0 0 0
```

Whenever we are interested in introducing further constraints, then the matrices at issue must be supplied by the user. For instance, to restrict  $\gamma_{14} = \gamma_{15}$ , then matrix Zga1 must be defined as

```
Zga1 <- diag(6)[ , -c(1, 3, 5)]; Zga1[5, 2] <- 1
Zga1
```

```
[,1] [,2] [,3]
[1,] 0 0 0
[2,] 1 0 0
[3,] 0 0 0
[4,] 0 1 0
[5,] 0 1 0
[6,] 0 0 1
```

- zga1, zga2, and zbe: vectors whose length is equal to the number of rows of Zga1, Zga2, and Zbe, respectively. In other words, length of zga1, zga2, and zbe is given by the number of elements in  $\gamma_1$ ,  $\gamma_2$ , and  $\beta$ . The suitable specification of these vectors, combined with that of matrices Zga1, Zga2, and Zbe, allows for linear constraints and fixed values of the item parameters, as in (9), (10), and (11). By default, zga1 and zga2 are vectors with elements 1 for each constrained item and 0 otherwise; by default zbe is a null vector. For instance, in the situation depicted in Figure 1 and matrices multi1 and multi2, default values assumed for vectors at issue are

```
zga1 <- c(1, 0, 1, 0, 0, 0)
zga2 <- c(0, 0, 0, 0, 1)
zbe <- rep(0, times = 7)
```

Any other constraint may be defined by modifying in a suitable way these three vectors and matrices Zga1, Zga2, Zbe. For instance, if we are interested in fixing the difficulty of (binary) item 4 to be equal to 2 (i.e.,  $\beta_4 = 2$ ), then we define Zbe and zbe as follows:

```
Zbe <- diag(7)[, -c(1,3,4,7)]
Zbe
```

```
[,1] [,2] [,3]
[1,] 0 0 0
[2,] 1 0 0
[3,] 0 0 0
[4,] 0 0 0
[5,] 0 1 0
[6,] 0 0 1
[7,] 0 0 0
```

```
zbe <- c(0, 0, 0, 2, 0, 0, 0)
```

Function est\_multi\_poly\_within supplies the following output:

- piv1 and piv2: vectors of the estimated weights of latent classes for  $U_i$  and  $V_i$ , respectively; in the presence of individual covariates, these are averages of the individual-specific mass probabilities.

- Th1 and Th2: matrices of estimated and constrained support points for each dimension (by row) and each latent class (by column) for  $U_i$  and  $V_i$ , respectively.
- Bec: matrix of estimated and constrained item difficulty parameters; exact zeros correspond to identifiability constraints.
- ga1c and ga2c: vectors of estimated and constrained item discriminating parameters for  $U_i$  and  $V_i$ , respectively; exact ones correspond to identifiability constraints and NA to items that do not load on the latent variable.
- th1t, th2t, bet, ga1t, and ga2t: estimated parameters (i.e., parameters without constraints) related to Th1, Th2, Bec, ga1c, ga2c, respectively.
- Th1s, Th2s, Becs, ga1cs, and ga2cs: standardized values of Th1, Th2, Bec, ga1c, and ga2c, respectively; in Th1s and Th2s classes are re-ordered according to the increasing values of the support points for the first dimension.
- piv1s and piv2s: the same as piv1 and piv2, but re-ordered according to Th1s and Th2s.
- fv1 and fv2: vectors indicating the reference items for each dimension of  $U_i$  and  $V_i$ , respectively.
- Phi: conditional response probabilities for every item and each pair of latent classes of  $U_i$  and  $V_i$ .
- De1 and De2: matrices of estimated regression coefficients for the model on the class weights for  $U_i$  and  $V_i$ , respectively, in the presence of individual covariates; for each covariate and the constant term, the number of estimated coefficients is equal to the number of latent classes minus one.
- Piv1, Piv2, Pp1, Pp2, and 1kv: optional output (obtained if output = TRUE) referred to the matrices of weights for every covariate configuration for latent variables  $U_i$  and  $V_i$ , the matrices of the posterior probabilities for each response configuration and latent class for latent variables  $U_i$  and  $V_i$ , and the values of the log-likelihood during the estimation process, respectively.
- XX1dis and XX2dis: design matrices for the covariates affecting the first and the second vector of latent variables, respectively (optional output obtained if output = TRUE).
- 1k: value of the log-likelihood at convergence.
- np: number of estimated model parameters.
- aic, bic, and ent: AIC, BIC, and entropy indices, respectively.
- seDe1, seDe2, seTh1, seTh2, seBec, sega1, sega2, seth1t, seth2t, sebet, sega1t, sega2t, and Vn: standard errors of the corresponding estimated parameters and estimated variance-covariance matrix (if out\_se = TRUE).

Some relevant commands to display output from function `est_multi_poly_within` are based on the S3 methods summary for the main estimates; `coef` and `confint` for the point estimates and confidence intervals (at a specified level of confidence) of support points, item parameters, and regression coefficients; `logLik` for the value of log-likelihood at convergence; and `vcov` for the estimated variance-and-covariance matrix.

Another relevant function of package **MLCIRTwithin** is `est_multi_poly_between`, which performs the maximum likelihood estimation of an LC-IRT model under between-item multidimensionality. The main differences with respect to function `est_multi_poly` of the R package **MultiLCIRT** are that the latter does not allow for items with a different number of response categories and refers to a slightly different specification of item difficulties (for details see [Bacci et al., 2014](#)).

Input arguments required by `est_multi_poly_between` are very similar to those of function `est_multi_poly_within`. The main difference is that only one vector of latent variables is involved in the model specification. Consequently, the number of latent classes (input `k`) is common to all the dimensions and the multidimensional structure of the items is specified through one matrix (input `multi`), having one row for each dimension. Constraints on model parameters are also possible through a suitable definition of arguments `Zth`, `zth`, `Zbe`, `zbe`, `Zga`, and `zga`, whose functioning is the same as the corresponding arguments in function `est_multi_poly_within`. The function provides as its main output argument a vector of estimated average weights of the latent classes (output `piv`) and a matrix of estimated support points for each dimension and each latent class of the latent trait before (output `Th`) and after the standardization (output `Ths`). Besides, a matrix of difficulty item parameters (outputs `Bec` and, in the case of standardization, `Becs`), a vector of discriminating indices (output `gac` and, in the case of standardization, `gacs`), and a matrix of regression coefficients (output `De`) are provided, other than other output arguments similar to those above described for `est_multi_poly_within`, included the S3 methods.

Finally, we clarify that a model specification of type

```
out1 <- est_multi_poly_between(S, k = k0, link = "global", multi = rbind(1:3, 4:6)),
```

with  $k_0$  latent classes, is substantially different from a model specification of type

```
out2 <- est_multi_poly_within(S, k1 = k0, k2 = k0, link = "global", multi1 = c(1:3),
                                multi2 = c(4:6)).
```

In fact, the model corresponding to out2 involves two completely independent latent variables, having incidentally the same number of latent classes: thus, an individual belonging to a specific class (say, class 1) according to the first latent variable may belong to any latent class under the second latent variable (say, class 2). On the contrary, model out1 involves only one latent variable decomposed in two dimensions: thus, belonging to a given latent class under one dimension implies belonging to the same class under the other dimension. Overall, model out2 has  $k_0 - 1$  free parameters more than model out1.

### Functions `search.model_within` and `search.model_between`

As outlined in Section “Likelihood inference,” the selection of a two-tier LC-IRT model may be a quite demanding procedure, requiring the choice of the number of support points for the latent variables and a check for the possible presence of local maxima. Function `search.model_within` allows us to search for the global maximum of the log-likelihood of a model with a specific formulation (in terms of multidimensional structure, link function, and constraints on the item parameters) given a vector of possible number of latent classes to try for.

In practice, function `search.model_within` applies function `est_multi_poly_within` a given number of times for each pair of values for  $k_1$  and  $k_2$ , initializing the estimation algorithm with deterministic and random values of the model parameters and holding, for each pair of  $k_1$  and  $k_2$ , that model with the highest value of the log-likelihood at convergence. To make the entire process computationally less demanding, the search of the global maximum may be performed with a relatively large tolerance level for checking convergence of the estimation algorithm. Then, in order to improve the precision of parameters estimates, the estimates provided by the model with the best value of the log-likelihood are used as starting values in the last step of the model selection process, using an augmented tolerance level. Note that when  $k_1 = 1$  or  $k_2 = 1$  the model estimation is actually performed by the function `est_multi_poly_between`, which is automatically retrieved by `search.model_within`.

The function at issue requires the following main input arguments:

- $S$ ,  $yv$ ,  $X$ ,  $link$ ,  $disc$ ,  $difl$ ,  $multi1$ ,  $multi2$ ,  $fort$ ,  $disp$ ,  $output$ ,  $out\_se$ ,  $Zth1$ ,  $zth1$ ,  $Zth2$ ,  $zth2$ ,  $Zbe$ ,  $zbe$ ,  $Zga1$ ,  $zga1$ ,  $Zga2$ , and  $zga2$ : are the same as in function `est_multi_poly_within`.
- $kv1$  and  $kv2$ : vectors of number of latent classes to try for latent variable  $U_i$  and  $V_i$ , respectively; single values are also allowed for.
- $tol1$  and  $tol2$ : tolerance levels (default value are  $10^{-6}$  and  $10^{-10}$ , respectively) for checking convergence of the algorithm as relative difference between consecutive log-likelihoods. The value of  $tol1$  is used for checks based on deterministic and random starting values, whereas the value of  $tol2$  is used for improving the precision of estimates for the model with the best log-likelihood level.
- $nrep$ : constant value that drives the number of estimations of each model with random starting values, given by  $nrep(k_1 k_2 - 1)$ ; the default value for  $nrep$  is 2. In the case of  $nrep$  equal to 0, only the estimation with deterministic starting values is performed.

Note that if single values for  $kv1$  and  $kv2$  are specified and  $nrep$  equals 0, function `search.model_within` performs just one call of function `est_multi_poly_within` (or function `est_multi_poly_between` if  $kv1$  or  $kv2$  equal 1) with option `start = "deterministic"`.

Function `search.model_within` supplies the following output:

- $aicv$ ,  $bicv$ , and  $necv$ : vectors of AIC, BIC, and NEC indices, respectively, for each of the estimated models.
- $errv$ : trace of any error occurred during the estimation process.
- $1kv$ : values of log-likelihood at convergence for each of the estimated models.
- $out.single$ : output of each single model, similar to the output of `est_multi_poly_within`, with the addition of values of  $k_1$  (output `k1`);  $k_2$  (output `k2`); and the sequence of log-likelihoods (output `lktrace`) for the deterministic start, for each random start, and for the final estimation provided by a tolerance level equal to  $tol2$  (if  $tol2 > tol1$ ).

Finally, we outline that a function with input and output arguments similar to those of the function `search.model_within` is available to perform the model selection in the case of between-item multidimensional LC-IRT models. This function is named `search.model_between` and it relies on `est_multi_poly_between`.

## Examples in R

In the following we illustrate package **MLCIRTwithin** through two data analysis examples. In Example 1 we describe the model selection procedure, as well as the interpretation of the output, considering a set of ordered items measuring two latent variables. In Example 2, the specification of constraints on the support points and on the item parameters is illustrated through the analysis of data concerning repeated item responses along two time occasions. The detailed software scripts to implement the two examples, named Example1.R and Example2.R, are available in the Supplementary Online Material at [https://sites.google.com/site/bartstatistics/sm\\_mlcirtwithin.zip](https://sites.google.com/site/bartstatistics/sm_mlcirtwithin.zip).

### Example 1: analysis of multidimensionality

Data set SF12\_nomiss, already provided in the R package **MLCIRTwithin**, refer to a sample of 493 oncological Italian patients who were asked to fill in the Italian validated Short Form 12 version 2 questionnaire (SF-12; [Stewart and Ware, 1992](#); [Ware et al., 2002](#)) concerning the assessment of HQOL. The questionnaire is comprised by 12 items having five ordered response modalities, except items 2 and 3 having only three modalities; a high score means a worse level of HQOL and vice-versa (note that, in the original scoring system, modalities of items 9 and 10 are reversed). Also the age is available for each patient. In the following we show the first few records of the data set and the related summaries.

```
library(MLCIRTwithin)

data(SF12_nomiss)
head(SF12_nomiss)

  Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8 Y9 Y10 Y11 Y12      age
1  1  0  1  1  0  2  2  2  1   1   1   0 74.94593
2  0  0  1  1  2  1  2  1   0   2   1   1 84.49829
3  1  1  1  2  1  0   0  2  1   1   1   1 77.44285
4  3  2  2  4  4  4   4  3  4   4   4   4 80.55305
6  1  1  2  2  2  2  2  2  2   2   2   2 81.68104
7  1  0  1  3  2  2  1  2  1   2   1   3 78.55168

str(SF12_nomiss)

'data.frame':      493 obs. of  13 variables:
 $ Y1 : num  1 0 1 3 1 1 1 1 2 2 ...
 $ Y2 : num  0 0 1 2 1 0 0 2 1 1 ...
 $ Y3 : num  1 1 1 2 2 1 1 2 1 2 ...
 $ Y4 : num  1 1 2 4 2 3 1 3 2 3 ...
 $ Y5 : num  0 2 1 4 2 2 2 2 2 3 ...
 $ Y6 : num  2 1 0 4 2 2 2 2 3 3 ...
 $ Y7 : num  2 2 0 4 2 1 1 2 3 3 ...
 $ Y8 : num  2 1 2 3 2 2 2 1 3 3 ...
 $ Y9 : num  1 0 1 4 2 1 2 3 3 3 ...
 $ Y10: num  1 2 1 4 2 2 1 2 2 2 ...
 $ Y11: num  1 1 1 4 2 1 2 1 4 3 ...
 $ Y12: num  0 1 1 4 2 3 1 2 3 3 ...
 $ age: num  74.9 84.5 77.4 80.6 81.7 ...

# For the description of each item see the online documentation
?SF_nomiss
```

According to the main current literature (see, mainly, [Ware et al., 2002](#)), the SF-12 questionnaire may be used to properly evaluate two main aspects of HQOL: physical and emotional. The standard scoring algorithm for summarizing these two latent dimensions is based on an orthogonal factor analysis, on the basis of which positive and negative weights are assigned to each item. More in detail, items 1 to 5 and item 8 have positive weights for physical HQOL and negative weights for emotional HQOL, whereas items 6, 7, 9, 11, and 12 have negative weights for physical HQOL and positive weights for emotional HQOL; item 10 has positive weights for both components. According to this scoring system, the scores of physical and emotional HQOL result by a suitable weighted average of the item responses.

A low score on physical HQOL has the following meaning (Ware and Gandek, 1998): substantial limitations in self-care, physical, social and role activities; severe bodily pain; frequent tiredness; health rated as poor. On the contrary, a high level of physical component corresponds to: no physical limitations, disabilities or decrements in well-being; high energy level; health rated as excellent. As regards the emotional component, a low score implies: frequent psychological distress, social and role disability due to emotional problems; health rated poor. On the other hand, a high level of emotional component corresponds to: frequent positive affect; absence of psychological distress and limitations in usual social activities due to emotional problems; health rated excellent.

The main drawback of the above algorithm based on the orthogonal factor analysis is that the summary score may be inconsistent due to weights of opposite sign for the same items, as higher emotional health scores drive physical health scores down and, similarly, higher physical health scores drive emotional health scores down (Farivar et al., 2007). An alternative approach for clustering patients according to their physical and emotional health status is based on IRT analysis (see, among others, Hays et al., 1993). In such a context, we analyze the multidimensional structure of SF-12 questionnaire through a two-dimensional model allowing items measuring both latent variables. In more detail, we compare several plausible multidimensional structures, defined through the following matrices, with the first one referred to the physical HQOL and the second one referred to the emotional HQOL:

**Type 1:** within-item multidimensional model with two independent latent variables and no shared item; items are allocated according to the sign of weights resulting by the factor analysis mentioned above:

```
(multi1_dim1 <- c(1:5, 8))
[1] 1 2 3 4 5 8
(multi1_dim2 <- c(6:7, 9:12))
[1] 6 7 9 10 11 12
```

**Type 2:** model with two latent variables sharing items that do not explicitly affect a specific dimension

```
(multi2_dim1 <- c(1:5, 8:12))
[1] 1 2 3 4 5 8 9 10 11 12
(multi2_dim2 <- c(6:12, 1))
[1] 6 7 8 9 10 11 12 1
```

**Type 3:** multidimensional structure similar to the previous one, but with three items (9, 10, and 11) assigned only to the emotional HQOL

```
(multi3_dim1 <- c(1:5, 8, 12))
[1] 1 2 3 4 5 8 12
(multi3_dim2 <- c(6:12, 1))
[1] 6 7 8 9 10 11 12 1
```

**Type 4:** multidimensional structure similar to that defined through multi21 and multi22, but one more item (number 8), concerning the presence of pain, is assigned only to physical HQOL, since pain is usually intended in terms of physical health (i.e., bodily pain)

```
(multi4_dim1 <- c(1:5, 8, 12))
[1] 1 2 3 4 5 8 12
(multi4_dim2 <- c(6:7, 9:12, 1))
[1] 6 7 9 10 11 12 1
```

The allocation of every item according to one of the above proposed structures is suggested by the more or less explicit reference of the item text to physical or emotional component of HQOL (or to both of them).

Considering the possible multidimensional structures above defined, we focus on models with global logit link function and free discriminating item parameters; also the effect of age on the mass probabilities is investigated.

```

# Item responses and covariates
S <- SF12_nomiss[, 1:12]
X <- SF12_nomiss[, 13]

For each type of multidimensional structure, we select the optimal number of latent classes on
the basis of BIC index according to the procedure described in Section "Likelihood inference". Also
the check of local maxima solutions follows the same lines described therein. For these aims we use
function search.model_within, as follows:

#### Model selection
maxk1 <- 6
maxk2 <- 6

tol1 <- 10^-3
tol2 <- 10^-6

# Multidimensional structure of Type 1
set.seed(0)
out1 <- search.model_within(S, kv1 = 1:maxk1, kv2 = 1:maxk2, X = X, link = "global",
                             disc = TRUE, multi1 = multi1_dim1, multi2 = multi1_dim2,
                             fort = TRUE, tol1 = tol1, tol2 = tol2, nrep = 1)

# Multidimensional structure of Type 2
set.seed(0)
out2 <- search.model_within(S, kv1 = 1:maxk1, kv2 = 1:maxk2, X = X, link = "global",
                             disc = TRUE, multi1 = multi2_dim1, multi2 = multi2_dim2,
                             fort = TRUE, tol1 = tol1, tol2 = tol2, nrep = 1)

# Multidimensional structure of Type 3
set.seed(0)
out3 <- search.model_within(S, kv1 = 1:maxk1, kv2 = 1:maxk2, X = X, link = "global",
                             disc = TRUE, multi1 = multi3_dim1, multi2 = multi3_dim2,
                             fort = TRUE, tol1 = tol1, tol2 = tol2, nrep = 1)

# Multidimensional structure of Type 4
set.seed(0)
out4 <- search.model_within(S, kv1 = 1:maxk1, kv2 = 1:maxk2, X = X, link = "global",
                             disc = TRUE, multi1 = multi4_dim1, multi2 = multi4_dim2,
                             fort = TRUE, tol1 = tol1, tol2 = tol2, nrep = 1)

```

We advise that the entire estimation process may take a very long computational time; then, we suggest to reduce the tolerance level of the algorithm (we adopted  $10^{-3}$  instead of the default value  $10^{-6}$  for argument tol1 and  $10^{-6}$  instead of the default value  $10^{-10}$  for argument tol2) and the number of repetitions with random initializations (we specified nrep = 1 instead of the default value nrep = 2). Outputs out1, out2, out3, and out4 are contained in the file 'Example1.RData', available in the supplementary online material at [https://sites.google.com/site/bartstatistics/sm\\_mlciirtwithin.zip](https://sites.google.com/site/bartstatistics/sm_mlciirtwithin.zip).

In order to select the optimal model, values of BIC index are displayed in the following 36-by-4 matrix, having one row for each model and one column for each multidimensional structure:

```

# BIC indices
BIC <- cbind(out1$bicv, out2$bicv, out3$bicv, out4$bicv)
colnames(BIC) <- c("Type 1", "Type 2", "Type 3", "Type 4")
k1 <- rep(1:6, times = 1, each = 6)
k2 <- rep(1:6, times = 6)
BIC <- cbind(k1, k2, BIC)
BIC

```

	k1	k2	Type 1	Type 2	Type 3	Type 4
[1,]	1	1	16041.95	16041.95	16041.95	16041.95
[2,]	1	2	14952.57	14758.90	14758.90	14857.40
[3,]	1	3	14637.43	14397.05	14397.05	14523.30
[4,]	1	4	14457.90	14186.90	14186.90	14323.75
[5,]	1	5	14457.16	14182.29	14182.29	14322.70
[6,]	1	6	14456.62	14181.76	14181.76	14323.05

```
[7,] 2 1 15245.74 14755.76 15106.47 15106.47
[8,] 2 2 14180.93 14010.29 14071.65 14097.89
[9,] 2 3 13865.85 13721.04 13750.05 13784.53
[10,] 2 4 13686.27 13543.49 13554.24 13595.26
[11,] 2 5 13685.48 13543.77 13553.30 13599.42
[12,] 2 6 13684.91 13559.40 13563.59 13604.91
[13,] 3 1 14945.39 14292.43 14748.77 14748.77
[14,] 3 2 13880.61 13674.91 13783.16 13791.13
[15,] 3 3 13565.50 13435.72 13482.17 13495.04
[16,] 3 4 13385.96 13291.34 13303.73 13319.80
[17,] 3 5 13385.83 13294.18 13306.15 13324.50
[18,] 3 6 13392.96 13308.42 13318.83 13334.18
[19,] 4 1 14897.97 14183.55 14678.01 14678.01
[20,] 4 2 13837.48 13606.69 13728.08 13732.33
[21,] 4 3 13522.40 13369.83 13441.26 13451.80
[22,] 4 4 13342.82 13245.49 13262.09 13271.06
[23,] 4 5 13342.45 13251.35 13269.85 13277.34
[24,] 4 6 13341.56 13265.77 13271.07 13293.80
[25,] 5 1 14858.93 14122.61 14610.54 14610.54
[26,] 5 2 13851.00 13549.28 13744.75 13674.28
[27,] 5 3 13538.57 13353.08 13383.57 13387.67
[28,] 5 4 13356.25 13196.32 13215.00 13219.66
[29,] 5 5 13298.70 13207.16 13216.58 13226.12
[30,] 5 6 13298.20 13212.85 13237.74 13242.02
[31,] 6 1 14932.45 14110.19 14613.70 14613.70
[32,] 6 2 13805.78 13557.76 13683.30 13683.43
[33,] 6 3 13495.08 13339.91 13393.67 13410.61
[34,] 6 4 13311.17 13205.40 13231.35 13230.88
[35,] 6 5 13312.05 13214.68 13237.86 13240.88
[36,] 6 6 13310.90 13222.19 13241.50 13243.07
```

On the basis of the above matrix we observe that the minimum value of the BIC index, that is, 13196.32, is displayed in column 2, denoting the multidimensional structure of type 2, and row 28, which refers to models with  $k_1 = 5$  and  $k_2 = 4$  latent classes. The output of the selected model is contained in the object `out2$out.single[[28]]`.

```
# Minimum BIC
min(BIC[, 3:6])

[1] 13196.32

# Detect model with the minimum BIC
arrayInd(which.min(BIC[, -c(1:2)]), .dim = dim(BIC))

[,1] [,2]
[1,] 28    2

# Number of support points for the best model
out2$out.single[[28]]$k1

[1] 5

out2$out.single[[28]]$k2
[1] 4

# Selected model
outsel = out2$out.single[[28]]
```

We also observe that the same number of support points is selected for the other multidimensional structures with shared items (i.e., structures of types 3 and 4).

```
# Minimum BIC and selected model for multidimensional structures of types 1, 3, 4
min(BIC[, "Type 1"])

[1] 13298.2
```

```

which.min(BIC[, "Type 1"])

[1] 30

min(BIC[, "Type 3"])

[1] 13215

which.min(BIC[, "Type 3"])

[1] 28

min(BIC[, "Type 4"])

[1] 13219.66

which.min(BIC[, "Type 4"])

[1] 28

```

After selecting the number of latent classes and the type of within-item multidimensional structure, we estimate again the selected model `out` with option `out_se = T` for the computation of standard errors. To reduce the computational time, we use function `est_multi_poly_within` with option `start = "external"` and specified as input for options `Phi`, `ga1c`, `ga2c`, `De1`, and `De2` the corresponding output from the selected model.

```

# Re-estimate model to compute standard errors
out <- est_multi_poly_within(S = S, k1 = outsel$k1, k2 = outsel$k2, X = X,
                               start = "external",
                               multi1 = multi2_dim1, multi2 = multi2_dim2,
                               Phi = outsel$Phi, ga1t = outsel$ga1t, ga2t = outsel$ga2t,
                               De1 = outsel$De1, De2 = outsel$De2,
                               link = "global", disc = TRUE, fort = TRUE, output = TRUE,
                               out_se = TRUE, disp = TRUE)

```

Details of the output of the estimated model may be displayed through the usual methods `summary`, `coef`, and `confint`.

```

### Display output of the estimated model
# summary(out)
# coef(out)
# confint(out)

# Estimates of support points and average mass probabilities for physical HQOL
lv1 <- rbind(out$Th1, t(out$piv1))
rownames(lv1) <- c("Physical HQOL", "Average prob.")
round(lv1, 3)

```

	1	2	3	4	5
Physical HQOL	0.823	1.684	2.854	0.093	-2.024
Average prob.	0.199	0.387	0.310	0.083	0.021

```

# Estimates of support points and average mass probabilities for emotional HQOL
lv2 <- rbind(out$Th2, t(out$piv2))
rownames(lv2) <- c("Emotional HQOL", "Average prob.")
round(lv2, 3)

```

	1	2	3	4
Emotional HQOL	0.471	11.526	7.585	4.151
Average prob.	0.149	0.141	0.390	0.321

According to the estimated model, patients are clustered in 5 latent classes denoting different levels of physical HQOL (output `out$Th1`) and in 4 latent classes denoting different levels of emotional HQOL (output `out$Th2`). To simplify the interpretation of the latent classes, it is useful to re-order and standardize the estimated support points. For this aim, function `est_multi_poly_within` provides the values of support points, which are standardized according to equations (14)-(18) and re-ordered

according to increasing values of the first dimension of the corresponding latent variable (outputs `out$Th1s` and `out$Th2s` with related weights `out$piv1s` and `out$piv2s`, respectively).

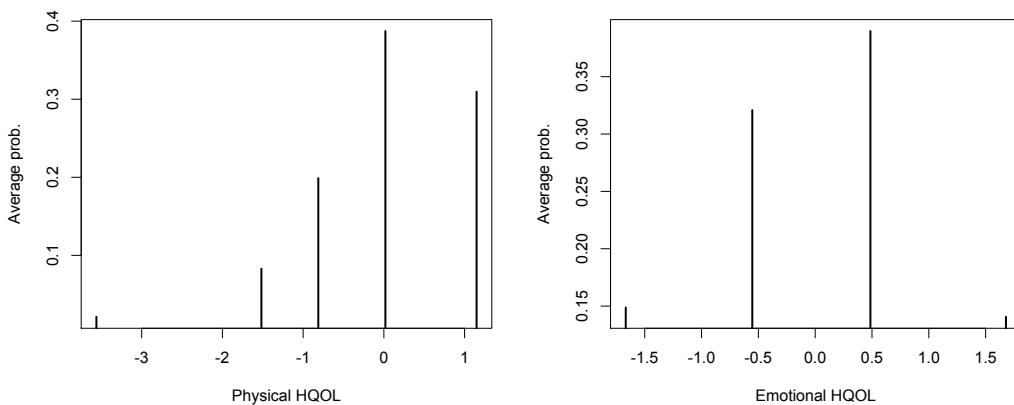
```
# Standardized support points of physical HQOL
lv1s <- rbind(out$Th1s, t(out$piv1s))
rownames(lv1s) <- c("Stand. Physical HQOL", "Average prob.")
round(lv1s, 3)

      5       4       1       2       3
Stand. Physical HQOL -3.561 -1.517 -0.812  0.019  1.149
Average prob.        0.021  0.083  0.199  0.387  0.310

# Standardized support points of emotional HQOL
lv2s <- rbind(out$Th2s, t(out$piv2s))
rownames(lv2s) <- c("Stand. Emotional HQOL", "Average prob.")
round(lv2s, 3)

      1       4       3       2
Stand. Emotional HQOL -1.667 -0.553  0.486  1.679
Average prob.        0.149  0.321  0.390  0.141
```

We observe that classes 5, 4, and 1 of physical HQOL and classes 1 and 4 of emotional HQOL collect patients with negative levels of the related latent trait, whereas patients with levels of HQOL above the mean belong to the remaining classes (i.e., classes 2 and 3 for physical HQOL and classes 3 and 2 for emotional HQOL). We also observe that the distribution of physical HQOL is strongly skewed to negative values, whereas that of emotional HQOL is symmetric (Figure 2).



**Figure 2:** Standardized mass probability distribution of physical (left panel) and emotional (right panel) HQOL.

As concerns item parameters, the discriminating indices and the corresponding standard errors and confidence intervals are displayed as follows:

```
# Item discriminating parameters and related standard errors
gamma1 <- cbind(out$ga1c, out$segal1c)
colnames(gamma1) <- c("gamma1", "st.err.")
round(gamma1, 3)

gamma1 st.err.
[1,] 1.000  0.000
[2,] 1.988  0.329
[3,] 1.193  0.199
[4,] 3.519  0.570
[5,] 3.394  0.582
[6,]     NA    NA
[7,]     NA    NA
[8,] 1.579  0.255
```

```
[9,] -0.006  0.117
[10,]  0.732  0.151
[11,]  0.145  0.125
[12,]  1.002  0.184

gamma2 <-cbind(out$ga2c, out$sega2c)
colnames(gamma2) <- c("gamma2", "st.err.")
round(gamma2, 3)

  gamma2 st.err.
[1,]  0.212  0.042
[2,]     NA    NA
[3,]     NA    NA
[4,]     NA    NA
[5,]     NA    NA
[6,]  1.000  0.000
[7,]  0.809  0.076
[8,]  0.128  0.037
[9,]  0.599  0.079
[10,] 0.429  0.059
[11,] 0.793  0.093
[12,] 0.606  0.074

# Confidence intervals at 95% of item discriminating parameters (columns 1-4)
confint(out)

[...] Output omitted
```

Confidence interval for the item parameters:

	gamma1_1	gamma1_2	gamma2_1	gamma2_2	beta1_1	beta1_2	beta2_1	beta2_2	beta3_1	beta3_2	beta4_1	beta4_2
1	1.0000	1.0000	0.1293	0.2939	0.0000	0.0000	2.2927	3.0763	5.4703	6.6560	7.0976	8.9132
2	1.3425	2.6326	NA	NA	-0.2428	1.8308	3.6945	6.1378	NA	NA	NA	NA
3	0.8036	1.5825	NA	NA	-1.4885	-0.0974	1.1539	2.5872	NA	NA	NA	NA
4	2.4010	4.6368	NA	NA	-1.3097	2.1941	1.2833	4.9066	5.2791	9.4358	8.5939	13.4278
5	2.2542	4.5348	NA	NA	-0.8427	2.5507	1.9437	5.5126	5.3510	9.4330	8.2434	12.8277
6	NA	NA	1.0000	1.0000	0.0000	0.0000	2.4866	3.9477	5.4657	7.6420	8.4771	11.2498
7	NA	NA	0.6612	0.9574	-0.4397	0.7189	1.7730	3.0777	4.1602	5.8143	7.1608	9.3353
8	1.0790	2.0796	0.0556	0.2010	-2.8325	-0.8067	0.3711	2.0693	3.0574	4.9355	4.5989	6.6225
9	-0.2348	0.2235	0.4451	0.7529	-1.9089	-0.6523	1.0767	2.2309	3.5534	5.0154	6.9565	9.1180
10	0.4366	1.0274	0.3135	0.5451	-0.7823	0.3940	1.9615	3.1901	4.6207	6.1860	7.3341	9.3617
11	-0.0999	0.3905	0.6100	0.9762	-1.1600	0.1005	1.6032	2.9576	4.8700	6.7528	7.4612	9.8398
12	0.6411	1.3637	0.4611	0.7503	-1.5644	0.0124	1.4481	2.8845	4.6337	6.4619	6.9167	9.0782

[...] Output omitted

In the previous output, the first two tables show the estimates of item discriminating parameters  $\hat{\gamma}_{1j}$  and  $\hat{\gamma}_{2j}$ , respectively, and the related standard errors, whereas the last table shows the corresponding inferior and superior limits of the confidence intervals at 95% level. Entries denoted by NA refer to those items that do not load on the corresponding latent variable.

We observe that discriminating parameters are generally highly significant, with two notable exceptions: discriminating indices for items 9 and 11 referred to physical HQOL (i.e., parameters  $\gamma_{19}$  and  $\gamma_{1,11}$ ). This result suggests that these two items do not contribute in a significant way to the measurement of physical HQOL.

The standardized estimates of parameters  $\hat{\gamma}_{1j}^*$  and  $\hat{\gamma}_{2j}^*$  are provided by

```
# Standardized discriminating parameters
gammas <- rbind(out$ga1cs, out$ga2cs)
rownames(gammas) <- c("Physical HQOL", "Emotional HQOL")
round(gammas, 3)
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
Physical HQOL	1.036	2.059	1.236	3.645	3.516	NA	NA	1.636	-0.006	0.758	0.151	1.038
Emotional HQOL	0.699	NA	NA	NA	NA	3.304	2.674	0.424	1.979	1.419	2.621	2.001

Note that the judgment about general health (item 1) is affected by both components of HQOL, but the physical dimension has a more relevant role ( $\hat{\gamma}_{11}^* = 1.036$  vs  $\hat{\gamma}_{21}^* = 0.699$ ). On the contrary, the self-evaluation of the consequences of physical or emotional health on social activities (item 12) is mainly affected by the emotional component of HQOL ( $\hat{\gamma}_{1,12}^* = 1.038$  vs  $\hat{\gamma}_{2,12}^* = 2.001$ ).

We also highlight relevant differences among items in terms of difficulty, as results by the related standardized item parameters:

```
# Standardized difficulty parameters
round(out$Becs, 3)
```

	cutoff			
item	1	2	3	4
1	-2.929	-0.245	3.134	5.076
2	-2.514	1.608	NA	NA
3	-2.778	-0.115	NA	NA
4	-5.414	-2.761	1.501	5.155
5	-4.795	-1.921	1.743	4.887
6	-5.979	-2.762	0.575	3.884
7	-4.700	-2.414	0.148	3.409
8	-5.215	-2.175	0.601	2.215
9	-4.853	-1.918	0.712	4.465
10	-3.979	-1.209	1.618	4.563
11	-5.514	-2.704	0.827	3.667
12	-6.066	-3.124	0.258	2.707

In particular, positive responses to items related only (items 4 and 5) or mainly (item 1) to physical HQOL require a level of the latent trait in average higher than items related only (items 6 and 7) or mainly (items 9, 11, and 12) to emotional HQOL. In other words, the emotional component of HQOL interferes less than physical component of HQOL on work, daily activities, and social life.

In the estimated model we assume an effect of patient's age on the HQOL, according to the multinomial logit parametrization, as in equation (3). Estimates of regression coefficients  $\delta_{1h_1}$  and  $\delta_{2h_2}$ , with  $h_1 = 2, 3, 4, 5$  and  $h_2 = 2, 3, 4$ , and related standard errors, are provided by:

```
# Effect of covariate age on physical HQOL
De1 <- cbind(out$De1, out$seDe1)
colnames(De1) <- c("delta12", "delta13", "delta14", "delta15", "se(delta12)",
                   "se(delta13)", "se(delta14)", "se(delta15)")
round(De1, 3)

      delta12 delta13 delta14 delta15 se(delta12) se(delta13) se(delta14) se(delta15)
intercept  2.960   3.021  -0.997  -4.773      1.035      0.937      1.862     2.453
X1        -0.038  -0.043   0.002   0.039      0.017      0.015      0.026     0.035

# Effect of covariate age on emotional HQOL
De2 <- cbind(out$De2, out$seDe2)
colnames(De2) <- c("delta22", "delta23", "delta24", "se(delta22)", "se(delta23)",
                   "se(delta24)")
round(De2, 3)

      delta22 delta23 delta24 se(delta22) se(delta23) se(delta24)
intercept  0.743   2.581   1.905      0.951      0.791      0.850
X1        -0.013  -0.027  -0.019      0.015      0.013      0.014

# Confidence intervals at 95% of regression coeffic. for physical and emotional HQOL
confint(out)

[...] Output omitted

Confidence interval for the regression coefficients for the 1st latent variable:
logit
      2_1      2_2      3_1      3_2      4_1      4_2      5_1      5_2
intercept  0.9301  4.9890  1.1833  4.8581 -4.6451  2.6520 -9.5808  0.0355
X1        -0.0709 -0.0055 -0.0717 -0.0146 -0.0489  0.0528 -0.0295  0.1075

Confidence interval for the regression coefficients for the 2nd latent variable:
logit
      2_1      2_2      3_1      3_2      4_1      4_2
intercept -1.1204  2.606   1.0306  4.1313  0.2383  3.5707
X1        -0.0434  0.017  -0.0521 -0.0021 -0.0458  0.0081
```

We observe a negative, although not particularly significant, effect of age on the probability to belong to a specific latent class with respect to the worst one (i.e., class 1), for both components of HQOL. A more parsimonious and more easily interpretable solution is provided by specifying a global logit parametrization through option `glob = TRUE` of the function `est_multi_poly_within`, as follows:

```
# Model with global logit parametrization for covariates
outgl <- est_multi_poly_within(S = S, k1 = 5, k2 = 4, X = X,
                                multi1 = multi2_dim1, multi2 = multi2_dim2, link = "global",
                                disc = TRUE, fort = TRUE, tol = 10^-8, disp = TRUE, output = TRUE,
                                out_se = TRUE, glob = TRUE)

# Effect of covariate age on physical HQOL
De1glob <- cbind(outgl$De1, outgl$seDe1)
colnames(De1glob) <- c("coef", "se")
round(De1glob, 3)

      coef     se
cutoff1 5.093 0.544
cutoff2 2.827 0.467
cutoff3 1.050 0.443
cutoff4 -0.777 0.477
X1      -0.028 0.007

# Effect of covariate age on emotional HQOL
De2glob <- cbind(outgl$De2, outgl$seDe2)
colnames(De2glob) <- c("coef", "se")
round(De2glob, 3)

      coef     se
cutoff1 2.306 0.463
cutoff2 0.664 0.453
cutoff3 -1.312 0.461
X1      -0.010 0.007

# Confidence intervals at 95% of regression coeffic. for physical and emotional HQOL
confint(outgl)

[...] Output omitted

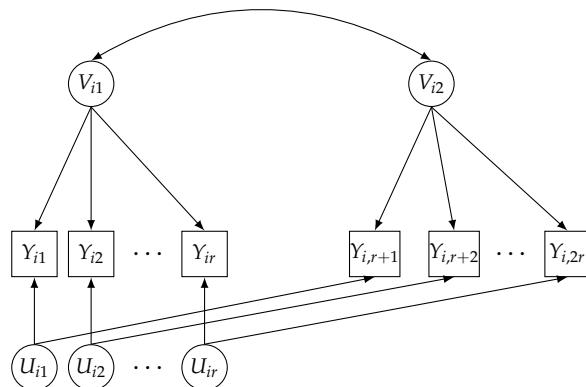
Confidence interval for the regression coefficients for the 1st latent variable:
logit
      _1      _2
cutoff1 4.0265 6.1586
cutoff2 1.9112 3.7427
cutoff3 0.1816 1.9187
cutoff4 -1.7110 0.1575
X1      -0.0422 -0.0132

Confidence interval for the regression coefficients for the 2nd latent variable:
logit
      _1      _2
cutoff1 1.3987 3.2124
cutoff2 -0.2229 1.5509
cutoff3 -2.2153 -0.4088
X1      -0.0240 0.0049
```

In such a case just one regression coefficient for each latent variable is estimated instead of  $k_1 - 1$  or  $k_2 - 1$ , denoting a negative effect of patient's age on the logit to belong to a specific class or higher with respect to a lower class.

### Example 2: analysis of repeated item responses

An interesting example of a real problem that can be solved through the discrete two-tier models is encountered when a questionnaire is used on multiple waves to measure the same latent variable at each time occasion. In such a situation, following Cai (2010) we distinguish a “specific” dimension  $U_{id_1}$  for each item within the questionnaire to capture the dependence between an individual’s responses to the same item  $d_1$  ( $d_1 = 1, \dots, D_1$ ,  $D_1 = r$ ) at the different occasions, and we introduce one “primary” dimension  $V_{id_2}$  for each wave  $d_2$  ( $d_2 = 1, \dots, D_2$ ), representing the latent variable of interest at a given occasion. Through a suitable choice of identifiability and equality constraints on the item parameters, it is possible to estimate the support points of  $V_i$  and, then, obtain a class-specific time trajectory of the latent variable measured by the questionnaire. The resulting multidimensional structure is illustrated in Figure 3 for the case  $D_2 = 2$ .



**Figure 3:** Path diagram of the discrete two-tier model for longitudinal data for  $r$  items and  $D_2 = 2$  waves.

To illustrate specification and estimation of the discrete two-tier model for longitudinal data we refer to a simulated data set about 10 different types of crime committed by a cohort of 10,000 hypothetical subjects on 6 waves; each crime corresponds to a binary item, which assumes value 1 if the crime was committed and 0 otherwise. The latent variable of interest denotes the propensity to commit a crime at each time occasion. The data set is available from the R package **LMest** (Bartolucci and Pandolfi, 2016).

```
library(MLCIRTwithin)

### Load and prepare data
library(LMest)
data(data_crimeal_sim)
data_crimeal_sim[1:12,]

  id sex time y1 y2 y3 y4 y5 y6 y7 y8 y9 y10
[1,] 1   1    1  0  0  0  0  0  0  0  0  0  0  0
[2,] 1   1    2  0  0  0  0  0  0  0  0  0  0  0
[3,] 1   1    3  0  0  0  0  0  0  0  0  0  0  0
[4,] 1   1    4  0  0  0  0  0  0  0  0  0  0  0
[5,] 1   1    5  0  0  0  0  0  0  0  0  0  0  0
[6,] 1   1    6  0  0  0  0  0  0  0  0  0  0  0
[7,] 2   1    1  0  0  0  0  0  0  0  0  0  0  0
[8,] 2   1    2  0  0  0  0  0  0  0  0  0  0  0
[9,] 2   1    3  0  0  0  0  0  0  0  0  0  0  0
[10,] 2  1    4  0  0  0  0  0  0  0  0  0  0  0
[11,] 2  1    5  0  0  0  0  0  0  0  0  0  0  0
[12,] 2  1    6  0  0  0  0  0  0  0  0  0  0  0
```

In order to simplify the illustration of the application, we keep the first two waves and five types of crime, as follows:

```
# Keep items: y1,y3,y5,y7,y10; keep occasions: 1, 2
criminal_red <- data_crimeal_sim[(data_crimeal_sim[,3]==1 | data_crimeal_sim[,3]==2),
c(1:3,4,6,8,10,13)]
```

Moreover, in order to estimate the model at issue through the R package **MLCIRTwithin**, we need to reshape the data from “long” to “wide” format; we also aggregate records corresponding to the same patterns. Note that, after reshaping the data set, the total amount of items is 10, that is, 5 items observed at time 1 ( $j = 1, \dots, 5$ ) and the same 5 items observed again at time 2 ( $j = 6, \dots, 10$ ).

```
# Data reshape in wide format
criminal_red <- data.frame(criminal_red)
crim_wide <- reshape(criminal_red, v.names = c("y1", "y3", "y5", "y7", "y10"),
                      timevar = "time", idvar = "id", direction = "wide")
head(crim_wide)

  id sex y1.1 y3.1 y5.1 y7.1 y10.1 y1.2 y3.2 y5.2 y7.2 y10.2
1  1   1     0     0     0     0     0     0     0     0     0     0
3  2   1     0     0     0     0     0     0     0     0     0     0
5  3   1     0     0     0     0     0     0     0     0     0     0
7  4   1     0     0     0     0     0     0     0     0     0     0
9  5   1     0     0     0     0     0     0     0     0     0     0
11 6   1     0     0     0     0     0     0     0     0     0     0
dim(crim_wide)
[1] 10000    12

# Aggregate records with the same pattern
crim_wide <- as.matrix(crim_wide)
crim_wide2 <- aggr_data(crim_wide[, -1])

# Item responses, covariates, and vector of weights
S <- crim_wide2$data_dis[, -1]
X <- crim_wide2$data_dis[, 1]; X <- X - 1
yv <- crim_wide2$freq

The multidimensional structure is defined through the specification of matrices multi1 and multi2, with multi1 referring to specific dimensions capturing the dependence between responses to the same item at the two different waves, whereas multi2 refers to the propensity to commit a crime at time 1 and at time 2. Then,  $\mathbf{U}_i = (U_{i1}, \dots, U_{i5})'$  and  $\mathbf{V}_i = (V_{i1}, V_{i2})'$ ; we also assume  $k_1 = k_2 = 2$ .
```

```
### Define the multidimensional structure
multi1 <- matrix(0, nrow=5, ncol=2)
multi2 <- matrix(0, nrow=2, ncol=5)
multi1[1,] <- c(6, 1)
multi1[2,] <- c(2, 7)
multi1[3,] <- c(3, 8)
multi1[4,] <- c(4, 9)
multi1[5,] <- c(5, 10)
multi2[1,] <- c(1:5)
multi2[2,] <- c(7, 6, 8:10)
multi1
 [,1] [,2]
[1,]    6    1
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
multi2
 [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    7    6    8    9   10

# Number of latent classes
k1 <- 2
k2 <- 2
```

The next step of the model specification consists in fixing some support points to identify the model and adding equality constraints on difficulties and discriminating indices to properly account for the longitudinal data structure, similarly to Cai (2010). In more detail, we fix the support points of the latent variable  $\mathbf{U}_i$ , that is,  $\mathbf{u} = (-1, -1, -1, -1, -1, 1, 1, 1, 1, 1)'$ , and on the first dimension of the latent variable  $\mathbf{V}_i$ , whereas the support points on the second dimension of  $\mathbf{V}_i$  are freely estimated, that is,

$v = (-1, v_{12}, 1, v_{22})'$ : in such a way, we estimate how the propensity to commit a crime changes over the time. Moreover, we constrain difficulty and discriminating parameters of each item at time 1 to be equal to the parameters of the same item at time 2, that is,  $\beta_j = \beta_{5+j}$ ,  $\gamma_{1j} = \gamma_{1,5+j}$ ,  $\gamma_{2j} = \gamma_{2,5+j}$ , with  $j = 1, \dots, 5$ .

```
### Specification of model constraints
# Fix support points on latent variable U
# Fix support points on the first dimension of latent variable V
# Free support points on the second dimension of latent variable V

(Zth1 <- matrix(0, nrow(multi1)*k1, 0))

[1,]
[2,]
[3,]
[4,]
[5,]
[6,]
[7,]
[8,]
[9,]
[10,]

(zth1 <- c(rep(-1, times = nrow(multi1)), rep(1, times = nrow(multi1))))
[1] -1 -1 -1 -1 -1  1  1  1  1  1

(Zth2 <- diag(nrow(multi2)*k2)[ , -c(1,3)])
[,1] [,2]
[1,]    0    0
[2,]    1    0
[3,]    0    0
[4,]    0    1

(zth2 <- c(-1, 0, 1, 0))
[1] -1  0  1  0

# Equality constraints on difficulties and discriminating indices to account
# for the longitudinal data structure

(Zbe <- matrix(1, nrow(multi2), 1) %x% diag(nrow(multi1)))
[,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    1    0    0    0
[3,]    0    0    1    0    0
[4,]    0    0    0    1    0
[5,]    0    0    0    0    1
[6,]    1    0    0    0    0
[7,]    0    1    0    0    0
[8,]    0    0    1    0    0
[9,]    0    0    0    1    0
[10,]   0    0    0    0    1

Zga2 <- Zbe
Zga1 <- Zbe
```

The final step consists in estimating the model according to the structure specified above. The main difference with respect to the model presented in the previous section (Example 1) is that now all items are shared by both the latent variables  $U_i$  and  $V_i$  and suitable constraints on the model parameters are introduced through arguments Zth1, zth1, Zth2, zth2, Zbe, Zga1, and Zga2, according to equations (7)-(11).

```
out <- est_multi_poly_within(S = S, yv = yv, k1 = k1, k2 = k2, X = X, link = "global",
                               disc = TRUE, multi1 = multi1, multi2 = multi2, disp = TRUE,
                               output = TRUE, out_se = TRUE, Zth1 = Zth1, zth1 = zth1, Zth2 = Zth2,
                               zth2 = zth2, Zbe = Zbe, Zga1 = Zga1, Zga2 = Zga2)
```

Estimates of support points and item parameters may be displayed through the usual methods

```

summary, coef, and confint:

### Display output
# summary(out)
# coef(out)
# confint(out)

# Support points and average weights of latent variable V
out$Th2

      class
dimension     1       2
  1 -1.0000000 1.000000
  2 -0.6925198 1.679367

out$piv2

[,1]
[1,] 0.91757919
[2,] 0.08242081

# Conditional response probabilities for latent variable V
# Note that the latent classes are numbered following an increasing order,
# starting from latent variable U
round(out$Phi[,, 3:4], 3)

, , class = 3

      item
category   1   2   3   4   5   6   7   8   9   10
  0 0.994 0.982 0.944 0.986 0.998 0.991 0.974 0.919 0.981 0.997
  1 0.006 0.018 0.056 0.014 0.002 0.009 0.026 0.081 0.019 0.003

, , class = 4

      item
category   1   2   3   4   5   6   7   8   9   10
  0 0.897 0.799 0.551 0.874 0.882 0.758 0.618 0.335 0.758 0.628
  1 0.103 0.201 0.449 0.126 0.118 0.242 0.382 0.665 0.242 0.372

```

We observe that 91.8% of the individuals belong to class 1, which is characterized by the smallest propensity to commit a crime, whereas the remaining 8.24% of individuals are allocated in class 2. Both classes present a tendency to increase the propensity to commit a crime from time 1 to time 2. Indeed, the estimated support points of the propensity to commit a crime at time 2 (i.e., estimates of  $v_{12}$  and  $v_{22}$ ) are higher than the corresponding values at time 1, for both the latent classes ( $-0.693$  vs  $-1$  for class 1 and  $1.679$  vs  $1$  for class 2). Moreover, the conditional probabilities of observing a given crime is higher for items observed at time 2, that is, items 6 to 10, with respect to the same items observed at time 1, that is, items 1 to 5, mainly in the case of latent class 2. For instance, for an individual belonging to class 2 the conditional probability of observing a crime of type 1 equals 10.3% at time 1 and it increases to 24.2% at time 2.

Finally, we outline that the constraints specified through matrices Zbe, Zga1, and Zga2 result in the following estimates, which are equal for every pair of items referring to the same type of crime (i.e., items 1-6, 2-7, 3-8, 4-9, 5-10):

```

# Estimates of item parameters
beta <- cbind(out$Bec[1:5], out$Bec[6:10])
colnames(beta) <- c("time 1", "time 2")
gamma1 <- cbind(out$ga1c[1:5], out$ga1c[6:10])
colnames(gamma1) <- c("time 1", "time 2")
gamma2 <- cbind(out$ga2c[1:5], out$ga2c[6:10])
colnames(gamma2) <- c("time 1", "time 2")

beta
      time 1   time 2
[1,] 4.994018 4.994018

```

```
[2,] 4.891019 4.891019
[3,] 2.700745 2.700745
[4,] 5.991825 5.991825
[5,] 5.544927 5.544927
```

```
gamma1
      time 1   time 2
[1,] 1.318531 1.318531
[2,] 2.195804 2.195804
[3,] 1.184268 1.184268
[4,] 2.883623 2.883623
[5,] 1.353772 1.353772
```

```
gamma2
      time 1   time 2
[1,] 1.510292 1.510292
[2,] 1.317424 1.317424
[3,] 1.311341 1.311341
[4,] 1.169848 1.169848
[5,] 2.183823 2.183823
```

## Summary

In this paper we illustrate the R package **MLCIRTwithin**, whose main function `est_multi_poly_within` implements an Expectation Maximization based approach to estimate the parameters of two-tier latent class item response theory (IRT) models. This class of models, which extends in a flexible way the class of basic IRT models, is based on two independent vectors of latent variables. Moreover, two main assumptions hold: (i) items are allowed to measure one or two latent variables (within-item multidimensionality) and (ii) latent variables are assumed to have a discrete distribution with a finite number of support points, which identify homogeneous latent classes of individuals, and related mass probabilities that may depend on individual covariates.

We illustrate the R package through two examples based on data about the measurement of health-related quality of life in cancer patients (Example 1) and about the measurement of the propensity to commit a crime in two time occasions (Example 2). The first example investigates the multidimensional structure of the questionnaire and is focused on the interpretation of the estimated model parameters. The second example illustrates how to treat longitudinal item responses through the specification of suitable constraints on support points and item parameters.

## Acknowledgements

Both authors acknowledge the financial support from the grant FIRB (“Futuro in ricerca”) 2012 on “Mixture and latent variable models for causal inference and analysis of socio-economic data”, which is funded by the Italian Government (RBFR12SHVV). Authors are also grateful to Dr. G. Miccinesi of the Istituto per lo Studio e la Prevenzione Oncologica (Florence, IT) for making available the data used in Example 1.

## Bibliography

- R. J. Adams, M. Wilson, and W.-C. Wang. The multidimensional random coefficients multinomial logit. *Applied Psychological Measurement*, 21:1–23, 1997. [p139]
- A. Agresti. *Categorical Data Analysis*. Wiley, Hoboken, NJ, 2013. [p141]
- H. Akaike. Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csaki, editors, *Second International Symposium of Information Theory*, pages 267–281, Budapest, 1973. Akademiai Kiado. [p143]
- D. Andrich. A rating formulation for ordered response categories. *Psychometrika*, 43:561–573, 1978. [p141]
- S. Bacci and F. Bartolucci. A multidimensional finite mixture structural equation model for nonignorable missing responses to test items. *Structural Equation Modeling: A Multidisciplinary Journal*, 22: 352–365, 2015. [p139, 143]

- S. Bacci, F. Bartolucci, and M. Gnaldi. A class of multidimensional latent class IRT models for ordinal polytomous item responses. *Communications in Statistics - Theory and Methods*, 43:787–800, 2014. [p139, 141, 148]
- F. Bartolucci. A class of multidimensional IRT models for testing unidimensionality and clustering items. *Psychometrika*, 72:141–157, 2007. [p139]
- F. Bartolucci and S. Bacci. MLCIRTwithin: Latent Class Item Response Theory (LC-IRT) Models under Within-Item Multidimensionality. R package version 2.1, URL <https://cran.r-project.org/web/packages/MLCIRTwithin>, 2016. [p139, 144]
- F. Bartolucci and S. Pandolfi. LMest: Latent Markov Models with and without Covariates. R package version 2.2-0, URL <https://cran.r-project.org/web/packages/LMest>, 2016. [p159]
- F. Bartolucci, S. Bacci, and M. Gnaldi. MultiLCIRT: An R package for multidimensional latent class item response models. *Computational Statistics & Data Analysis*, 71:971–985, 2014. [p144]
- F. Bartolucci, S. Bacci, and M. Gnaldi. *Statistical Analysis of Questionnaires: a Unified Approach based on R and Stata*. Chapman & Hall, CRC Press, Boca Raton, FL, 2015. [p139, 141, 144]
- F. Bartolucci, S. Bacci, and M. Gnaldi. MultiLCIRT: Multidimensional latent class Item Response Theory models. R package version 2.10, URL <https://cran.r-project.org/web/packages/MultiLCIRT/>, 2016. [p144]
- A. Bertrand and C. M. Hafner. covLCA: Latent Class Models with Covariate Effects on Underlying and Measured Variables. R package version 1.0, URL <https://cran.r-project.org/web/packages/covLCA/>, 2013. [p145]
- A. Birnbaum. Some latent trait models and their use in inferring an examinee's ability. In F. M. Lord and M. R. Novick, editors, *Statistical Theories of Mental Test Scores*, pages 395–479. Addison-Wesley, Reading, MA, 1968. [p145]
- R. D. Bock, R. Gibbons, and E. Muraki. Full-information item factor analysis. *Applied Psychological Measurement*, 12:261–280, 1988. [p139]
- W. E. Bonifay. An illustration of the two-tier item factor analysis model. In S. P. Reise and D. A. Revicki, editors, *Handbook of Item Response Theory Modeling*, pages 207–225. Routledge, 2015. [p139]
- L. Cai. A two-tier full-information item factor analysis model with applications. *Psychometrika*, 75: 581–612, 2010. [p139, 142, 159, 160]
- L. Cai, J. S. Yang, and M. Hansen. Generalized full-information item bifactor analysis. *Psychological Methods*, 16:221–248, 2011. [p139, 142]
- G. Celeux and G. Soromenho. An entropy criterion for assessing the number of clusters in a mixture model. *Journal of Classification*, 13:195–212, 1996. [p143]
- P. Chalmers, J. Pritikin, A. Robitzsch, M. Zoltak, K. Kim, C. F. Falk, and A. Meade. mirt: Multidimensional Item Response Theory. R package version 1.18, URL <https://cran.r-project.org/web/packages/mirt/>, 2016. [p145]
- P. R. Chalmers. mirt: a multidimensional item response theory package for the R environment. *Journal of Statistical Software*, 48:1–29, 2012. [p145]
- A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38, 1977. [p142]
- S. S. Farivar, W. E. Cunningham, and R. D. Hays. Correlated physical and mental health summary scores for the SF-36 and SF-12 health survey, V. 1. *Health and Quality of Life Outcomes*, 5:doi: 10.1186/1477-7525-5-54, 2007. [p151]
- A. K. Formann. Mixture analysis of multivariate categorical data with covariates and missing entries. *Computational Statistics & Data Analysis*, 51:5236–5246, 2007. [p141]
- A. K. Formann and T. Kohlmann. Three-parameter linear logistic latent class analysis. In J. A. Hagenaars and A. L. McCutcheon, editors, *Applied latent class analysis*, pages 183–210. Cambridge University Press, Cambridge, 2002. [p139]
- R. D. Gibbons and D. R. Hedeker. Full-information item bi-factor analysis. *Psychometrika*, 57:423–436, 1992. [p139]

- R. D. Gibbons, R. D. Bock, D. Hedeker, D. J. Weiss, E. Segawa, D. K. Bhaumik, D. J. Kupfer, E. Frank, V. J. Grochocinski, and A. Stover. Full-information item bifactor analysis of graded response data. *Applied Psychological Measurement*, 31:4–19, 2007. [p139]
- L. A. Goodman. Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika*, 61:215–231, 1974. [p139]
- R. K. Hambleton and H. Swaminathan. *Item Response Theory: Principles and Applications*. Kluwer Nijhoff Publishing, Boston, 1985. [p139]
- R. D. Hays, C. D. Sherbourne, and R. M. Mazel. The RAND 36-item health survey 1.0. *Health Economics*, 2:217–227, 1993. [p151]
- E. E. Jang. A framework for cognitive diagnostic assessment. In C. A. Chapelle, Y.-R. Chung, and J. Xu, editors, *Towards adaptive CALL: Natural language processing for diagnostic language assessment*, pages 117–131. Iowa State University, Ames, IA, 2008. [p144]
- M. Jeon, F. Rijmen, and S. Rabe-Hesketh. Flexible item response theory modeling with FLIRT. *Applied Psychological Methods*, 38:404–405, 2014. [p145]
- P. F. Lazarsfeld and N. W. Henry. *Latent Structure Analysis*. Houghton Mifflin, Boston, 1968. [p139]
- G. N. Masters. A Rasch model for partial credit scoring. *Psychometrika*, 47:149–174, 1982. [p141]
- L. Muthén and B. Muthén. *Mplus user's guide*. Los Angeles, CA, Muthén & Muthén edition, 2012. [p145]
- M. C. Neale, M. D. Hunter, J. N. Pritikin, M. Zahery, T. R. Brick, R. M. Kickpatrick, R. Estabrook, T. C. Bates, H. H. Maes, and S. M. Boker. OpenMx 2.0: Extended structural equation and statistical modeling. *Psychometrika*, 81(2):535–549, 2016. doi: 10.1007/s11336-014-9435-8. [p145]
- G. Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests*. Danish Intitute for Educational Reserch, Copenhagen, 1960. [p145]
- M. D. Reckase. *Multidimensional Item Response Theory*. Springer, 2009. [p139]
- S. P. Reise. The rediscovery of bifactor measurement models. *Multivariate Behavioral Research*, 47: 667–696, 2012. [p139]
- A. Robitzsch, T. Kiefer, A. C. George, and A. Uenue. CDM: Cognitive Diagnosis Modeling. R package version 5.0-0, URL <https://cran.r-project.org/web/packages/CDM/>, 2016. [p144]
- Y. Rosseel, D. Oberski, J. Byrnes, L. Vanbrabant, V. Savalei, E. Merkle, and et al. lavaan: Latent Variable Analysis. R package version 0.5-20, URL <http://lavaan.org>, 2015. [p145]
- A. A. Rupp and J. L. Templin. Unique characteristics of diagnostic classification models: A comprehensive review of the current state-of-the-art. *Measurement: Interdisciplinary Research and Perspectives*, 6:219–262, 2008. [p144]
- F. Samejima. Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph Supplement*, 34, 1969. [p141]
- G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978. [p143]
- A. L. Stewart and J. E. Ware. *Measuring Functioning and Well-being*. Duke University Press: Durham, 1992. [p150]
- K. K. Tatsuoka. Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement*, 20:345–354, 1983. [p144]
- W. J. Van der Linden and R. K. Hambleton. *Handbook of Modern Item Response Theory*. Springer, New York, 1997. [p139]
- J. Vermunt and J. Magidson. *Latent GOLD 4.0 user's guide*. Statistical Innovations Inc., Belmont, Massachusetts, 2005. [p145]
- M. von Davier. A general diagnostic model applied to language testing data. *British Journal of Mathematical and Statistical Psychology*, 61:287–307, 2008. [p139, 141]
- J. E. Ware and B. Gandek. Overview of the SF-36 health survey and the international quality of life assessment (IQOLA) project. *Journal of Clinical Epidemiology*, 51:903–912, 1998. [p151]

J. E. Ware, M. Kosinski, D. M. Turner-Bowker, and B. Gandek. *How to Score Version 2 of the SF-12 Health Survey*. QualityMetric Incorporated: Lincoln, RI, 2002. [p150]

*Silvia Bacci*

*Department of Economics, University of Perugia  
Via A. Pascoli 20, 06123 Perugia  
Italy*

[silvia.bacci@unipg.it](mailto:silvia.bacci@unipg.it)

*Francesco Bartolucci*

*Department of Economics, University of Perugia  
Via A. Pascoli 20, 06123 Perugia  
Italy*

[francesco.bartolucci@unipg.it](mailto:francesco.bartolucci@unipg.it)

# Variants of Simple Correspondence Analysis

by Rosaria Lombardo and Eric J. Beh

**Abstract** This paper presents the R package **CAvariants** (Lombardo and Beh, 2017). The package performs six variants of correspondence analysis on a two-way contingency table. The main function that shares the same name as the package – **CAvariants** – allows the user to choose (via a series of input parameters) from six different correspondence analysis procedures. These include the classical approach to (symmetrical) correspondence analysis, singly ordered correspondence analysis, doubly ordered correspondence analysis, non symmetrical correspondence analysis, singly ordered non symmetrical correspondence analysis and doubly ordered non symmetrical correspondence analysis. The code provides the flexibility for constructing either a classical correspondence plot or a biplot graphical display. It also allows the user to consider other important features that allow to assess the reliability of the graphical representations, such as the inclusion of algebraically derived elliptical confidence regions. This paper provides R functions that elaborates more fully on the code presented in Beh and Lombardo (2014).

## Introduction

Computational procedures for detecting the association between two or more categorical variables are important aspects of statistical theory and practice. In particular, correspondence analysis provides a quick and simple graphical summary of how categories and variables are associated with one another. The theoretical aspects of the analysis are well documented in the statistical and allied disciplines; see, for example, Benzécri (1973), Greenacre (1984), Lebart et al. (1984), Beh (2004a), Nishisato (2007), and Beh and Lombardo (2014). Despite the necessity for programs and functions that allow their user to perform correspondence analysis, the availability for many of the varied approaches is generally limited. Commercially available statistical software, such as MATLAB, Minitab, SAS and SPSS provide a means of carrying out correspondence analysis, although their procedures often provide only the most basic of features as part of their output. Generally nothing beyond the calculation of principal inertia values, profile coordinates, contribution to inertia and a two-dimensional correspondence plot are provided. Other popular statistical languages, such as R, provide some packages for performing simple and multiple correspondence analysis of the classical (symmetrical) type, (Murtagh, 2005; Nenadic and Greenacre, 2007; Alberti, 2015; De Leeuw, 2006; De Leeuw and Mair, 2009a; Ringrose, 2012; Kostov et al., 2015). Nevertheless, at present, no popular statistical packages provide functions to perform ordered variants of symmetrical and non symmetrical correspondence analysis.

## Overview of correspondence analysis in R

Since the mid 2000's the programming environment of R has proven to be extremely popular in all areas of theoretical and applied statistics. This is due in part to the free availability of the program from the Comprehensive R Archive Network (CRAN; <http://CRAN.R-project.org/>), the versatility of the coding environment and the ever increasing number of packages that are now available on the CRAN.

Various R packages have received a great deal of attention for their contribution to the computing of correspondence analysis (CA). One of the first is the **MASS** package (Venables and Ripley, 2002; Ripley, 2016). It provides the user with a means of performing simple and multiple correspondence analysis with the option of including supplementary points onto a display. More recently the **ca** package of Nenadic and Greenacre (2007) includes functions for performing simple, multiple and joint correspondence analysis using two and three dimensions for the graphical displays. Supplementary points were incorporated into the R code of Murtagh (2005) while the **anacor** package of De Leeuw and Mair (2009a) allows the user to perform classical and canonical correspondence analysis with missing values (De Leeuw, 2006; De Leeuw and Mair, 2009b). Further, one may refer to the CA or MCA functions in the **FactoMineR** package by Lé et al. (2008). For lexical tables, the CaGalt function incorporated into the **FactoMineR** package by Kostov et al. (2015) may be used. Another recent package – **cabootcrs** – by Ringrose (2012) checks the reliability of association by superimposing onto a plot bootstrap confidence regions. The **CAinterprTools** package by Alberti (2015) makes use of graphical features to enrich a visual interpretation of CA results. Alternatively, De Leeuw and Mair (2009a) prepared the **homals** package for performing Gifi's approach to correspondence analysis. As well also, Clavel et al. (2014) presented **dualScale** package for doing dual scaling (i.e., multiple correspondence analysis) of multiple

	<i>Variants of correspondence analysis</i>										
package	CA	NSCA	MCA	JCA	SOCA	DOCA	SONSCA	DONSCA	CCA	CNSCA	DCA
<b>ade4</b>	x	x	x						x	x	
<b>anacor</b>	x			x					x		
<b>ca</b>	x			x	x						
<b>cabootcrs</b>	x										
<b>CAinterprTools</b>	x			x							
<b>CAvariants</b>	x	x			x	x	x	x	x	x	
<b>cncaGUI</b>											
<b>dualScale</b>	x		x								
<b>ExPosition</b>	x		x							x	
<b>FactoMineR</b>	x		x								
<b>homals</b>	x		x								
<b>MASS</b>	x		x								
<b>PTAk</b>	x										
<b>vegan</b>	x		x					x			

**Table 1:** R packages and some CA variants. CA: simple CA; NSCA: non symmetrical CA; MCA: multiple CA; JCA: joint CA; SOCA: singly ordered CA; DOCA: doubly ordered CA; SONSCA: singly ordered NSCA; DONSCA: doubly ordered NSCA; CCA: canonical CA; CNSCA: canonical NSCA; DCA: discriminant CA

choice data. [Baxter and Cool \(2010\)](#) and [Alberti \(2015\)](#) provide a good overview of correspondence analysis using R with an archaeological focus. Another R based package that can be downloaded freely from CRAN is [ExPosition](#) ([Beaton et al., 2014](#)). It is written by Herve Abdi and his team and performs a variety of different multivariate data analysis techniques, including correspondence analysis and multiple correspondence analysis. Abdi's group has also been responsible for other variations of correspondence analysis including *multi-block discriminant correspondence analysis* ([Williams et al., 2010](#)) and *discriminant correspondence analysis* ([Abdi, 2007](#)). Furthermore, another suite of R functions that enables the user to perform a variety of correspondence analysis techniques is [vegan](#) ([Oksanen et al., 2016](#)), which was developed primarily for vegetation ecologists. It includes functions that provide the user with a large array of techniques to choose from including classic correspondence analysis, canonical correspondence analysis and detrended correspondence analysis. One may also consider the [ade4](#) package ([Dray and Dufour, 2007](#); [Chessel et al., 2004](#); [Thioulouse et al., 1997](#)), which also includes non symmetric correspondence analysis, to analyze ecological and environmental data in the framework of numerous euclidean exploratory methods. Further, the [cncaGUI](#) package ([Librero et al., 2015](#)) allows canonical correspondence analysis and canonical non symmetrical correspondence analysis providing inferential results by using bootstrap methods. The [PTAk](#) package includes ([Leibovici, 2010, 2015](#)) functions for doing multiway data decomposition, and in particular, it also allows simple correspondence analysis and a generalization of correspondence analysis for  $k$ -way tables. Lastly, but certainly not least, the R code of [Murtagh \(2005\)](#) for performing simple and multiple correspondence analysis may also be considered.

An overview of the broad areas of correspondence analysis that these packages cover is summarized in Table 1. While non symmetrical correspondence analysis for nominal variables is included in some of the R packages on the CRAN that perform correspondence analysis, the remaining ordered variants have not yet been made available in any R package. However, fragments of R code for some of these CA variants are available in [Beh and Lombardo \(2014\)](#). Therefore, this paper provides a comprehensive description of R code that enhances, beyond the classics, the type of correspondence analysis that one may use. The advantages of these variants is that they enable the user to incorporate categorical predictor/response associations and the ordinal structure of a variable. For ordered variables we can easily identify any linear and non-linear sources of association that may exist in the data. The ordered variants also provide a visualization of non-linear trends of association; the classical approaches to correspondence analysis do not encompass these features.

The theoretical aspects underlying all the six variants of correspondence analysis considered in this paper can be found in [Beh and Lombardo \(2014\)](#) and [Lombardo et al. \(2016\)](#). However, here we will provide the reader with a brief overview of the theoretical aspects of these analyses. We also describe how the algebraic confidence ellipses for polynomial biplots can be derived; this aspect of the analysis has not been described elsewhere.

## Some theory

### Symmetrical and non symmetrical correspondence analysis

Consider a two-way contingency table  $\mathbf{N}$  of dimension  $I \times J$  such that it is a cross-classification of two variables consisting of  $I$  row categories and  $J$  column categories, respectively. Denote the matrix of the joint relative frequencies by  $\mathbf{P} = (p_{ij})$  so that  $\sum_{i=1}^I \sum_{j=1}^J p_{ij} = 1$ . Let  $p_{i\bullet} = \sum_{j=1}^J p_{ij}$  and  $p_{\bullet j} = \sum_{i=1}^I p_{ij}$  be the  $i$ th marginal row proportion and the  $j$ th marginal column proportion, general elements of the diagonal matrices,  $\mathbf{D}_I$  and  $\mathbf{D}_J$ , respectively.

There are many ways that correspondence analysis can be performed and Nishisato (2007, Chapter 2) provides an excellent overview of some of them. Here, we present the chi-squared statistic expressed in terms of the weighted sum-of-squares of the centered column profiles since this alternative expression of  $X^2$  is useful when comparing symmetrical correspondence analysis with its non symmetrical variant. Therefore, consider the chi-squared statistic of  $\mathbf{N}$  which is defined as

$$X^2 = n \sum_{i=1}^I \sum_{j=1}^J \frac{p_{\bullet j}}{p_{i\bullet}} \left( \frac{p_{ij}}{p_{\bullet j}} - p_{i\bullet} \right)^2 = n \sum_{i=1}^I \sum_{j=1}^J \frac{p_{\bullet j}}{p_{i\bullet}} \pi_{ij}^2,$$

where  $\mathbf{\Pi} = (\pi_{ij})$  is the  $I \times J$  matrix of centered column profiles. In this case, the weight matrices in  $\Re^I$  and  $\Re^J$  are defined by the elements of the matrices  $\mathbf{D}_I^{-1}$  and  $\mathbf{D}_J$ , respectively.

Suppose we now treat the column variable as a predictor variable and the row variable as its response variable. When such an asymmetric association structure exists between the two categorical variables one may consider non symmetrical correspondence analysis (Lauro and D'Ambra, 1984; D'Ambra and Lauro, 1989; Kroonenberg and Lombardo, 1999). To quantify this asymmetric association, consider the Goodman-Kruskal (1954) tau index

$$\tau = \frac{\sum_{i=1}^I \sum_{j=1}^J p_{\bullet j} \left( \frac{p_{ij}}{p_{\bullet j}} - p_{i\bullet} \right)^2}{1 - \sum_{i=1}^I p_{i\bullet}^2} = \frac{\sum_{i=1}^I \sum_{j=1}^J p_{\bullet j} \pi_{ij}^2}{1 - \sum_{i=1}^I p_{i\bullet}^2} = \frac{\tau_{num}}{\tau_{den}}.$$

For this asymmetric case, the weight matrices are  $\mathbf{I}$  (an  $I \times I$  identity matrix) and  $\mathbf{D}_J$  respectively. Notice that the denominator can be treated as a constant term since it does not depend on the predictor variable. For this reason it can be neglected without losing any information about the structure of the association. Therefore  $\tau_{num}$  is the measure of association considered in non symmetrical correspondence analysis.

In order to graphically depict the association or the prediction of the rows given the columns in a low dimensional space, we may consider the generalized singular value decomposition of the centered column matrix  $\mathbf{\Pi}$  using the suitable weight matrices (Kroonenberg and Lombardo, 1999).

Suppose we consider a general framework for the symmetrical and non symmetrical variants of CA (Lombardo et al., 2016), that considers generic weight matrices,  $\mathbf{V}_I$  and  $\mathbf{W}_J$ , in  $\Re^I$  and  $\Re^J$ . This general framework may be defined by considering the weighted centered column profile matrix

$$\tilde{\mathbf{\Pi}} = \mathbf{V}^{1/2} \mathbf{\Pi} \mathbf{W}^{1/2}.$$

Therefore, symmetric (or classical) correspondence analysis may be performed by considering  $\mathbf{V} = \mathbf{D}_I^{-1}$  and  $\mathbf{W} = \mathbf{D}_J$ , while non symmetrical correspondence analysis is defined when  $\mathbf{V} = \mathbf{I}$  and  $\mathbf{W} = \mathbf{D}_J$ . Doing so leads to the generalized singular value decomposition (GSVD) of

$$\text{GSVD}(\tilde{\mathbf{\Pi}}) = \mathbf{A} \mathbf{\Pi} \mathbf{B}^T.$$

where the right and left singular vectors are  $\mathbf{A}(= a_{im})$  and  $\mathbf{B}(= b_{jm})$ , respectively. These quantities have the orthonormality properties with metrics  $\mathbf{D}_I^{-1}$  or  $\mathbf{I}$  (identity) (in  $\Re^I$ , depending on the symmetric or asymmetric relationship between the rows and columns) and  $\mathbf{D}_J$  (in  $\Re^J$ ), respectively. As usual, the elements of the diagonal matrix of singular values,  $\mathbf{\Lambda} = \text{diag}(\lambda_m)$ , are arranged in descending order.

## Ordered symmetrical and non symmetrical correspondence analysis

When both variables are ordered, we adapt SVD by using basis vectors for the row and column spaces by performing the bivariate moment decomposition (BMD) on the matrix  $\tilde{\mathbf{m}}$ . The BMD of  $\tilde{\mathbf{m}}$  is expressed as

$$\text{BMD}(\tilde{\mathbf{m}}) = \mathcal{A}\mathbf{Z}\mathcal{B}^T,$$

where  $\mathcal{A}$  and  $\mathcal{B}$  are the row and column polynomial matrices defined by Emerson (1968), respectively, and  $\mathbf{Z}$  is the matrix of the generalized correlations (Rayner and Beh, 2009). The construction of polynomials  $\mathcal{A}$  and  $\mathcal{B}$  requires the specification of *a priori* scores,  $s_X(i)$  and  $s_Y(j)$  (defined by  $\mathfrak{m}_i$  and  $\mathfrak{m}_j$  in CAvariants, respectively), to reflect the ordinal structure of the row and column variables. These polynomials are orthonormal with respect to the weight matrices. For the analysis of nominal variables, when a symmetrical association between the variables is considered, the weights in  $\mathfrak{N}^I$  and  $\mathfrak{N}^J$  are  $\mathbf{D}_I^{-1}$  and  $\mathbf{D}_J$ , respectively. When an asymmetric association is considered, the weights are given by  $\mathbf{I}$  and  $\mathbf{D}_J$ , respectively.

When only one of the two variables consists of ordered categories, rather than considering the BMD or the GSVD of  $\tilde{\mathbf{m}}$ , one may consider instead its hybrid decomposition (HD) (Beh, 2001, 2008; Lombardo et al., 2016). This method of decomposition consists of singular vectors for the nominal variable and orthogonal polynomials for the ordered variable. Consider the case, as does the package CAvariants, where the column variable consists of ordered categories and the row variable consists of nominal categories. Then the HD of  $\tilde{\mathbf{m}}$  takes the form

$$\text{HD}(\tilde{\mathbf{m}}) = \mathbf{A}\mathbf{Z}\mathcal{B}^T,$$

where  $\mathbf{A}$  is the column matrix of singular vectors for the nominal row categories and  $\mathcal{B}$  is the column matrix of orthogonal polynomials for the ordered column categories. The generic elements of  $\mathbf{Z}$ ,  $z_{mv}$ , are the *hybrid* generalized correlations; for further details on these elements see Beh and Lombardo (2014) and Lombardo et al. (2016).

## Generalized correlations in ordered CA variants

The generalized correlation matrix,  $\mathbf{Z}$ , in the BMD of  $\tilde{\mathbf{m}}$  reflects the various sources of association between the variables and is derived using orthogonal polynomials (Best and Rayner, 1996; Beh, 1997; Rayner and Beh, 2009). For example, when the row and column scores are defined as consecutive integers such that  $s_X(i) = i$  for  $i = 1, \dots, I$  and  $s_Y(j) = j$  for  $j = 1, \dots, J$ , then  $z_{11}$  is Pearson's product moment correlation of N. A simple generalization of this correlation is  $z_{12}$  which is a measure of the correlation between any change in the location of the row categories and dispersion of the column categories. For this reason,  $z_{12}$  is a generalized correlation describing the linear-by-quadratic association between the row and column categories.

For ordered CA variants, the total inertia is

$$\text{Inertia}(\tilde{\mathbf{m}}) = \sum_{u=1}^{I-1} \sum_{v=1}^{J-1} z_{uv}^2,$$

which can also be written in matrix form as

$$\text{Inertia}(\tilde{\mathbf{m}}) = \text{trace}(\mathbf{Z}^T \mathbf{Z}) = \text{trace}(\mathbf{Z} \mathbf{Z}^T) = \text{trace}(\tilde{\mathbf{m}}^2).$$

From the matrix of generalized correlations  $\mathbf{Z}$ , we can obtain the inertia of each polynomial axis by considering the sum-of-squares of  $z_{uv}$  over either  $u$  or  $v$ . Using BMD or HD, the symmetric and asymmetric measures of association ( $X^2$  and  $\tau$ ) can be partitioned into polynomial components that reflect various sources of variation for each of the categories. The inertias of the polynomial components will henceforth be referred to as *sources of inertia* and are akin to the principal inertia values in (symmetrical or non symmetrical) correspondence analysis.

A formal statistical test of the  $X^2$  or  $\tau$  index can be made. To test the statistical significance of the total inertia in the symmetrical and non symmetrical case, we can compare the chi-squared statistic, or the C-statistic,  $C = \tau \cdot (n - 1) \cdot (I - 1)$  (Light and Margolin, 1971), with the  $\chi^2$  distribution with  $(I - 1)(J - 1)$  degree of freedom; see, for example, Beh and Lombardo (2014) for further details.

**Unequal inertias of the row and column polynomials.** When considering the BMD of  $\tilde{\mathbf{m}}$ , the total inertia of the row and column spaces ( $\mathfrak{N}^I$  and  $\mathfrak{N}^J$ , respectively) will be identical. However, the inertia associated with each of the row and column polynomials will often be different. For the row categories,

there are  $I - 1$  row inertia values – one for each of the axes – where the inertia of the  $u$ th polynomial axis is  $z_{\bullet u}^2$ . Similarly, there are  $J - 1$  column inertia values – one for each of the axes – where the inertia of the  $v$ th axis is  $z_{\bullet v}^2$ . For this reason, we recommend constructing polynomial biplots for the ordered variants of correspondence analysis instead of the traditional correspondence plots constructed using principal coordinates. See [Beh and Lombardo \(2014\)](#) and [Lombardo et al. \(2016\)](#) for more details on these features.

For the HD of  $\mathbf{\tilde{Z}}$ , the interpretation and properties of the  $\mathbf{Z}$  matrix are a mixture (or hybrid) of  $\mathbf{\tilde{Z}}$  from the GSVD and  $\mathbf{Z}$  from the BMD. When considering the space  $\Re^J$ , calculating  $\sum_{m=1}^{M^*} z_{m1}^2 = z_{\bullet 1}^2$  gives the location component of the ordered columns and represents the principal inertia for this variable along the first polynomial axis. Similarly in  $\Re^I$ , computing  $\sum_{v=1}^{J-1} z_{1v}^2 = z_{1\bullet}^2 = \lambda_1^2$  yields the principal inertia of the first principal axis for the nominal row variable. Like BMD, HD yields different sets of inertia values for each axis in the  $\Re^I$  and  $\Re^J$  spaces.

## Polynomial biplots and elliptical confidence regions

When constructing a polynomial biplot, the ordered row and column categories can be displayed in a single plot since the row and column coordinates are computed with respect to the same set of polynomial axes. For example, in a polynomial *row metric preserving* (or row isometric) biplot, the column standard polynomial coordinates are

$$\mathbf{G} = \mathcal{B} \quad (g_{jv} = \beta_{jv}),$$

while the principal polynomial coordinates for the row categories are

$$\mathbf{F} = \mathcal{A}\mathbf{Z} = \mathbf{\tilde{Z}}\mathbf{W}_J\mathcal{B} \quad \left( f_{iv} = \alpha_{iu}z_{uv} = \sum_{j=1}^J w_{\bullet j} \tilde{\pi}_{ij} \beta_{jv} \right).$$

In practice, the coordinates for both the row and column categories are computed using the same orthonormal polynomial axes, i.e., the column polynomials.

The plot method for objects returned by CAvariants provides the user with the option of constructing parametric (or algebraic) elliptical confidence regions for all the six CA variants not only for the nominal CA variants as originally proposed by [Beh \(2010\)](#). We compute the semi-major and semi-minor axis lengths of the elliptical region for the row and column categories. Here, we provide the ellipse axes lengths for the ordered symmetric variants of correspondence analysis. For example, the semi-major axis length of the confidence ellipse for the  $i$ th row category is

$$x_{i(\alpha)} = z_{11}^2 \sqrt{\frac{\chi_\alpha^2}{n \times \text{trace}(\mathbf{Z}'\mathbf{Z})} \left( \frac{1}{p_{i\bullet}} - \sum_{m=3}^{I-1} a_{im}^2 \right)}, \quad (1)$$

while the semi-minor axis length for this row is

$$y_{j(\alpha)} = z_{22}^2 \sqrt{\frac{\chi_\alpha^2}{n \times \text{trace}(\mathbf{Z}'\mathbf{Z})} \left( \frac{1}{p_{i\bullet}} - \sum_{m=3}^{I-1} a_{im}^2 \right)}. \quad (2)$$

Similar semi-axis lengths can also be derived for the column categories and for the non-symmetrical CA variants. Furthermore, note that ellipsoids can be constructed for three- or higher-dimensional correspondence plots by considering the input parameter  $M > 2$  in the plot method. For further details on this issue see [Beh \(2010\)](#); [Beh and Lombardo \(2014\)](#).

Unlike the confidence circles of [Lebart et al. \(1984\)](#) and the more computationally intensive bootstrap techniques proposed in the literature ([Markus, 1994](#); [Linting et al., 2007](#); [Ringrose, 2012](#); [Greenacre, 1984](#); [Lombardo and Ringrose, 2012](#)), constructing confidence ellipses in this manner takes into consideration the contribution of the  $i$ th row principal polynomial coordinate in dimensions higher than the second. In fact, since all  $I$  dimensions are reflected in the semi-major and semi-minor axis lengths, all of the contribution that a point has to the symmetrical or asymmetrical association can be accounted for in a two-dimensional plot using equations (1) and (2). Additional information for how to compute the  $p$ -values of each category point can be easily found by considering a similar theoretical development of the  $p$ -values described in [Beh and Lombardo \(2014, 2015\)](#) for a correspondence analysis of a contingency table with nominal variables.

## An overview of the CAvariants package

The primary function discussed in this paper is CAvariants. It allows the user to select which analysis to perform from a suite of six correspondence analysis techniques. These include symmetrical (or classical) correspondence analysis, non symmetrical correspondence analysis and their ordered variants, described in Beh and Lombardo (2014, 2015).

The six variations of simple correspondence analysis included in the package CAvariants are:

- The classical approach to simple correspondence analysis (the default analysis which is defined by the input parameter catype = "CA").
- Two-way, or doubly ordered, symmetrical correspondence analysis (the user can perform this analysis by defining the input parameter catype = "DOCA").
- One-way, or singly ordered, correspondence analysis for tables of symmetrically related variables, where the column variable is ordered (the user can perform this analysis by defining the input parameter catype = "SOCA").
- Non symmetrical correspondence analysis where the nominal column variable is a predictor of the nominal row variable (the user can perform this analysis by defining the input parameter catype = "NSCA")
- Two-way, or doubly ordered, non symmetrical correspondence analysis where the ordered column variable is a predictor of the ordered row variable (the user can perform this analysis by defining the input parameter catype = "DONSCA").
- One-way, or singly ordered, non symmetrical correspondence analysis, where the ordered column variable is a predictor of the nominal row variable (the user can perform this analysis by defining the input parameter catype = "SONSCA")

The input parameters of the function CAvariants are:

- The two-way contingency table, Xtable.
- The assigned ordered scores for the row categories. By default, mi = NULL which gives consecutive integer valued (natural) scores.
- The assigned ordered scores for the column categories. By default, mj = NULL which gives consecutive integer valued (natural) scores.
- The horizontal polynomial or principal axis. By default firstaxis = 1.
- The vertical polynomial or principal axis. By default lastaxis = 2.
- The input parameter for specifying what variant of correspondence analysis is considered. By default catype = "CA", other possible values are: catype = "SOCA", catype = "DOCA", catype = "NSCA", catype = "SONSCA", catype = "DONSCA".
- The input parameter, ellcomp, ensures that the characteristics of the algebraic confidence ellipses are computed and stored. When ellcomp = TRUE (which is the default), the output includes the characteristics of the ellipses. The eccentricity of the confidence ellipses is summarized by the quantity eccentricity, this is the distance between the center and either of its two foci, which can be thought of as a measure of how much the conic section deviates from being circular (when it is equal to zero then the region becomes circular). The semi-major axis length of the ellipse for each row and column point is given by HL Axis 1 while HL Axis 2 gives the semi-minor axis length of the points along the second axis. The area of the ellipse for each row and column category is given by Area while the p-value of each category is defined by P-value.
- The number of axes Mell considered in determining the structure of the elliptical confidence regions. By default, Mell = min(nrow(Xtable), ncol(Xtable)), i.e., the rank of the data matrix.
- The confidence level, alpha, of the elliptical regions. By default, alpha = 0.05.

To visually portray and assess the statistical significance of the categories to the association between the variables of a contingency table, the plot method can be called by the user. As well as displaying the classic correspondence plot or biplot, this function allows one to superimpose onto the plot algebraically derived elliptical confidence regions for each of the principal coordinates (Lebart et al., 1984; Beh, 2010; Lombardo and Ringrose, 2012; Beh and Lombardo, 2015) for all CA variants. There are other features of the plot, i.e., through the plot method for "CAvariants" objects, the user may utilize. Some of these are applicable to all of the analyses and some are applicable to only a few. The input parameters of the plot method for "CAvariants" objects are:

- The name of the output object, for example say res, used with the main function CAvariants.
- The horizontal polynomial or principal axis, firstaxis. By default, firstaxis = 1.

- The vertical polynomial or principal axis, `lastaxis`. By default, `lastaxis = 1`.
- The size of characters, `cex`, displayed on the correspondence plot or biplot. By default, `cex = 0.8`.
- The parameter, `cex.lab` that specifies the size of character labels of axes in graphical displays. By default, `cex.lab = 0.8`.
- The scaling parameter, `prop`, for specifying the limits of the plotting area. By default, `prop = 1`.
- The type of graphical display required (either a classical correspondence plot or a biplot). The user can look at a classical correspondence plot by defining the input parameter `plottype = "classic"`. When `plottype = "biplot"`, it produces biplot graphical displays, or polynomial biplots in case of an ordered analysis. Note that for an ordered analysis only polynomial biplots are suitable. In particular for the singly ordered variants, only row isometric polynomial biplots make sense, as we assume that the ordered variable is the column variable (the column coordinates are standard polynomial coordinates and the row coordinates are principal polynomial coordinates). By default, `plottype = "biplot"`.
- For a biplot, one may specify that it be a row-isometric biplot (`biptype = "row"`) or a column-isometric biplot (`biptype = "column"`). This feature is available for the nominal symmetrical and the non symmetrical correspondence analyses. By default, a row-isometric biplot, `biptype = "row"`, is produced.
- The parameter for scaling the biplot coordinates, `scaleplot`, originally proposed in Section 2.3.1 of [Gower et al. \(2011\)](#) and described on page 135 of [Beh and Lombardo \(2014\)](#). By default, `scaleplot = 1`.
- The parameter `posleg` for specifying the position of the legend when portraying trends of ordered categories in ordered variants of correspondence analysis. By default, `posleg = "topleft"`.
- The parameter `pos` for specifying the position of point symbols in the graphical displays. By default, `pos = 2`.
- The logical parameter, `ell` which specifies whether algebraic confidence ellipses are to be included in the plot or not. Setting the input parameter to `ell = TRUE` will allow the user to assess the statistical significance of each category to the association between the variables. The ellipses will be included when the plot is constructed using principal coordinates (being either row and column principal coordinates or row and column principal polynomial coordinates). By default, this input parameter is set to `ell = FALSE`. See also the input parameter `ellcomp` of the function `CAvariants` for a description of the numeric characteristics of the confidence ellipses (eccentricity, area, etc.), as well as the input parameter `ellprint` of the `print` method for “`CAvariants`” objects for getting a print of these parameters.
- The number of axes `Mell` considered when portraying the elliptical confidence regions. By default, it is equal to `Mell = min(nrow(Xtable), ncol(Xtable))`, i.e., the rank of the data matrix. This parameter is identical to the input parameter `Mell` of the function `CAvariants`.
- The confidence level of the elliptical regions. By default, `alpha = 0.05`.

The `print` method for “`CAvariants`” objects included in the package, `CAvariants`, and displays the main results of the analysis specified by the user. The results displayed depends on the type of analysis being performed. The principal inertia values, total inertia and  $p$ -values are included as part of its output when `catype = "CA"`, `catype = "SOCA"` or `catype = "DOCA"` and are based on Pearson’s chi-squared statistic. The Goodman Kruskal tau-index is the association measure of interest when `catype = "NSCA"`, `catype = "SONSCA"` or `catype = "DONSCA"`. When an ordered analysis is specified – such as when `catype = "DOCA"`, `catype = "SOCA"`, `catype = "SONSCA"` or `catype = "DONSCA"` – a table describing the significant polynomial components of inertia will also be reported.

The input parameters of the `print` method for “`CAvariants`” objects are:

- The name of the output object, for example say `res`, used with the main function `CAvariants`.
- The number of dimensions, `printdims`, that are used to generate the correspondence plot, or biplot, and for summarizing the numerical output of the analysis. By default, `printdims = 2`.
- The flag parameter, `ellprint`, allows that the characteristics of the confidence ellipses (eccentricity, semi-axis, area,  $p$ -values) are displayed. By default, `ellprint = TRUE`.
- The number of axes, `Mell`, used for the construction of the confidence ellipses. By default, it is equal to its maximum value, `Mell = min(nrow(Xtable), ncol(Xtable))`, i.e., the rank of the data matrix. This input parameter is identical to the parameter `Mell` of both, function `CAvariants` and the `plot` method for “`CAvariants`” objects.

- The level of significance used for the construction of the elliptical regions, `alpha`. By default, `alpha = 0.05`.
- The minimum number of decimal places, `digits`, used for displaying the numerical summaries of the analysis. By default, `digits = 3`.

In general, this function produces the following output:

- The two-way contingency table, `Xtable`.
- The matrix of row weights, `Row weights`: `Imass`. These weights depend on the type of analysis performed.
- The matrix of column weights, `Column weights`: `Jmass`. These weights are equal to the data column margins for all types of analysis performed.
- The total inertia, `Total inertia`, of the analysis performed. For example, when considering the variants of non symmetrical correspondence analysis, the numerator of the Goodman-Kruskal tau index, the associated C-statistic and its *p*-value are produced.
- The inertia values, their percentage contribution to the total inertia and the cumulative percent inertias of the row and column space `Inertias`. When performing an ordered correspondence analysis, this output summary describes both the row and column spaces for each principal or polynomial axis. When `catype` is "CA" or "NSCA", the associated inertia values in the row and column spaces are identical.
- The generalized correlation matrix `Generalized correlation matrix`, when performing an ordered correspondence analysis, `catype` should be "DOCA", "DONSCA", "SOCA" or "SONSCA".
- The row principal coordinates, `Row principal coordinates`, when `catype` is "CA" or "NSCA".
- The column principal coordinates, `Column principal coordinates`, when `catype` is "CA" or "NSCA".
- The row standard coordinates, `Row standard coordinates`, when `catype` is "CA" or "NSCA".
- The column standard coordinates, `Column standard coordinates`, when `catype` is "CA" or "NSCA".
- The row principal polynomial coordinates, `Row principal polynomial coordinates`, when performing an ordered correspondence analysis.
- The column principal polynomial coordinates, `Column principal polynomial coordinates`, when performing a doubly ordered correspondence analysis.
- The Row standard polynomial coordinates, i.e., standard polynomial coordinates for the row categories when performing a doubly ordered correspondence analysis.
- The Column standard polynomial coordinates, i.e., standard polynomial coordinates for the column categories when performing an ordered correspondence analysis.
- The Euclidean distance of the row categories from the origin of the plot, `Row distances from the origin of the plot`.
- The Euclidean distance of the column categories from the origin of the plot, `Column distances from the origin of the plot`.
- The polynomial components of the total inertia and their *p*-values, `Polynomial components`. The total inertia of the column space is partitioned to identify polynomial components when `catype` is "SOCA" or "SONSCA". When `catype` is "DOCA" or "DONSCA", the total inertia of both the row and column space is partitioned to identify of polynomial components.
- The inner product, `Inner product`, of the biplot coordinates (concerning the first two axes when `firstraxis = 1` and `lastaxis = 2`).
- When the input flag parameter is `ellprint = TRUE`, then the print includes the eccentricity of the confidence ellipses, the semi-major axis length of the ellipse for each row and column point, `HL Axis 1`, the semi-minor axis length for the ellipse for each row and column point, `HL Axis 2`, the area of the ellipse for each row and column point, `Area` and the *p*-value for each row and column point, `P-value`, see also the parameter `ellcomp` of the function `CAvariants` for a detailed description of these parameters.

Furthermore, package `CAvariants` contains a `summary` method for the objects returned by `CAvariants`. This method provides the list of the objects names of the output and a selection of the main output objects described in the `print` method for objects returned by `CAvariants`.

## Numerical outputs

As an example of the complete set of numerical results that is obtained from performing a particular variant of correspondence analysis, consider the case where a singly ordered non symmetrical correspondence analysis is performed on the data table shopdataM available in the package **CAvariants**. This object is the contingency table being analyzed and is described more fully in Section [Application](#). The output object name of the main function is called res and is the execution of the CAvariants function on the shopdataM. The object res is obtained using

```
R> res <- CAvariants(shopdataM, catype = "SONSCA")
```

The results are available in the following entries which can be obtained using

```
R> names(res)
```

which gives

```
[1] "Xtable"      "rows"        "cols"        "r"           "rowlabels"
[6] "collabels"   "Rprinccoord" "Cprinccoord" "Rstdcoord"   "Cstdcoord"
[11] "tauden"      "tau"         "inertiasum"  "inertias"    "inertias2"
[16] "comps"       "catype"     "mj"          "mi"          "pcc"
[21] "Jmass"       "Imass"     "Trend"      "Z"           "ellcomp"
[26] "risell"      "Mell"
```

These results may be printed to the screen by using

```
R> print(res)
```

while a summary of each of these numerical features is produced by using

```
R> summary(res)
```

## Application

To demonstrate the application of a variant of simple correspondence analysis described in the **CAvariants** package, we present the following example. We shall confine our attention to the non symmetrical correspondence analysis of a singly ordered contingency table. The contingency table that we are examining is concerned with shoplifting in The Netherlands and summarizes, in part, the results of a survey of the Dutch Central Bureau of Statistics ([Israëls, 1987](#)). The data considers a sample of 20819 men who were suspected of shoplifting in Dutch stores between 1977 and 1978. The predictor variable consists of the age groups of the perpetrators (less than 12yrs, 12 to 14yrs, 15 to 17yrs, 18 to 20yrs, 21 to 29yrs, 30 to 39yrs, 40 to 49yrs, 50 to 64yrs, 65yrs and over) while the response variable of the table consists of the items stolen. These items are *clothing, clothing accessory, tobacco and/or provisions, stationary, books, records, household goods, candy, toys, jewelry, perfume, hobby and/or tools* and *other items*. For an extensive description of this example, and the application of correspondence analysis, see [Lombardo et al. \(2016\)](#).

After choosing the suitable variant of correspondence analysis, we create the object res that consists of the complete features of the analysis by running the command

```
R> res <- CAvariants(shopdataM, catype = "SONSCA")
```

`print(res)` will return as part of its output the following numerical features:

RESULTS for SONSCA Correspondence Analysis

	M12<	M13	M16	M19	M25	M35	M45	M57	M65+
clothing	81	138	304	384	942	359	178	137	45
accessories	66	204	193	149	297	109	53	68	28
tobacco	150	340	229	151	313	136	121	171	145
stationary	667	1409	527	84	92	36	36	37	17
books	67	259	258	146	251	96	48	56	41
records	24	272	368	141	167	67	29	27	7
household	47	117	98	61	193	75	50	55	29
candy	430	637	246	40	30	11	5	17	28
toys	743	684	116	13	16	16	6	3	8
jewelry	132	408	298	71	130	31	14	11	10
perfumes	32	57	61	52	111	54	41	50	28

hobby	197	547	402	138	280	200	152	211	111
other	209	550	454	252	624	195	88	90	34

## Row Weights: Imass

	clothing	accessories	tobacco	stationary	books	records	household
clothing	1	0	0	0	0	0	0
accessories	0	1	0	0	0	0	0
tobacco	0	0	1	0	0	0	0
stationary	0	0	0	1	0	0	0
books	0	0	0	0	1	0	0
records	0	0	0	0	0	1	0
household	0	0	0	0	0	0	1
candy	0	0	0	0	0	0	0
toys	0	0	0	0	0	0	0
jewelry	0	0	0	0	0	0	0
perfumes	0	0	0	0	0	0	0
hobby	0	0	0	0	0	0	0
other	0	0	0	0	0	0	0
	candy	toys	jewelry	perfumes	hobby	other	
clothing	0	0	0	0	0	0	
accessories	0	0	0	0	0	0	
tobacco	0	0	0	0	0	0	
stationary	0	0	0	0	0	0	
books	0	0	0	0	0	0	
records	0	0	0	0	0	0	
household	0	0	0	0	0	0	
candy	1	0	0	0	0	0	
toys	0	1	0	0	0	0	
jewelry	0	0	1	0	0	0	
perfumes	0	0	0	1	0	0	
hobby	0	0	0	0	1	0	
other	0	0	0	0	0	1	

## Column Weights: Jmass

	12<	13	16	19	25	35	45	57	65+
12<	0.137	0.00	0.000	0.0000	0.000	0.0000	0.0000	0.0000	0.0000
13	0.000	0.27	0.000	0.0000	0.000	0.0000	0.0000	0.0000	0.0000
16	0.000	0.00	0.171	0.0000	0.000	0.0000	0.0000	0.0000	0.0000
19	0.000	0.00	0.000	0.0808	0.000	0.0000	0.0000	0.0000	0.0000
25	0.000	0.00	0.000	0.0000	0.166	0.0000	0.0000	0.0000	0.0000
35	0.000	0.00	0.000	0.0000	0.000	0.0665	0.0000	0.0000	0.0000
45	0.000	0.00	0.000	0.0000	0.000	0.0000	0.0394	0.0000	0.0000
57	0.000	0.00	0.000	0.0000	0.000	0.0000	0.0000	0.0448	0.0000
65+	0.000	0.00	0.000	0.0000	0.000	0.0000	0.0000	0.0000	0.0255

Total inertia 0.038

Inertias, percent inertias and cumulative percent inertias of the row space

	inertia	inertiapc	cuminertiapc
1	0.0300	79.88	79.88
2	0.0037	9.86	89.74
3	0.0032	8.44	98.18
4	0.0003	0.92	99.10
5	0.0003	0.67	99.77
6	0.0001	0.17	99.94
7	0.0000	0.05	99.99
8	0.0000	0.01	100.00

Inertias, percent inertias and cumulative percent inertias of the column space

	inertia2	inertiapc2	cuminertiapc2
1	0.0225	59.83	59.83
2	0.0096	25.58	85.41
3	0.0028	7.33	92.74

4	0.0019	5.18	97.92
5	0.0003	0.82	98.74
6	0.0003	0.74	99.48
7	0.0001	0.35	99.83
8	0.0001	0.17	100.00

Predictability Index for Variants of Non symmetrical Correspondence Analysis:

Numerator of Tau Index predicting the rows given the column categories

[1] 0.038

Tau Index predicting the rows given the column categories

[1] 0.041

C-statistic 10331.51 and p-value 0

Polynomial Components of Inertia

\*\* Column Components \*\*

	Component	Value	P-value
Location		6181.536	0
Dispersion		2642.363	0
Cubic		757.192	0
Error		750.418	0
** C-Statistic **		10331.509	0

Generalized correlation matrix of Hybrid Decomposition

	v1	v2	v3	v4	v5	v6	v7	v8
m1	-0.147	0.084	0.018	-0.030	0.011	0.005	-0.005	0.003
m2	-0.028	-0.034	-0.032	0.024	0.005	-0.010	0.003	0.001
m3	-0.013	-0.037	0.036	-0.016	-0.004	0.006	-0.006	0.002
m4	-0.001	0.002	0.006	0.014	-0.010	0.005	-0.001	-0.001
m5	-0.001	-0.001	-0.007	-0.006	-0.007	0.009	-0.004	-0.004
m6	0.000	0.000	-0.001	0.000	0.000	-0.001	-0.006	0.005
m7	0.000	0.000	0.000	-0.001	-0.002	-0.003	-0.001	-0.002
m8	0.000	0.000	0.000	0.000	-0.001	0.000	0.001	0.001
m9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Column standard polynomial coordinates = column polynomial axes

	Axis 1	Axis 2
M12<	-1.232	1.352
M13	-0.759	0.142
M16	-0.285	-0.652
M19	0.188	-1.029
M25	0.661	-0.991
M35	1.135	-0.536
M45	1.608	0.336
M57	2.081	1.624
M65+	2.554	3.328

Row principal polynomial coordinates

	Axis 1	Axis 2
clothing	0.072	-0.056
accessories	0.017	-0.014
tobacco	0.039	0.017
stationary	-0.084	0.033
books	0.012	-0.012
records	0.000	-0.021
household	0.015	-0.004
candy	-0.045	0.027
toys	-0.067	0.049
jewelry	-0.017	-0.006

perfumes	0.014	0.000
hobby	0.030	0.015
other	0.014	-0.030

Column distances from the origin of the plot

	Axis 1	Axis 2
M12<	0.057	0.002
M13	0.027	0.000
M16	0.000	0.000
M19	0.027	0.002
M25	0.046	0.004
M35	0.041	0.000
M45	0.031	0.005
M57	0.021	0.022
M65+	0.010	0.047

Row distances from the origin of the plot

	Axis 1	Axis 2
clothing	0.005	0.003
accessories	0.000	0.000
tobacco	0.001	0.000
stationary	0.007	0.001
books	0.000	0.000
records	0.000	0.000
household	0.000	0.000
candy	0.002	0.001
toys	0.005	0.002
jewelry	0.000	0.000
perfumes	0.000	0.000
hobby	0.001	0.000
other	0.000	0.001

Inner product of coordinates (first two axes when 'firstaxis=1' and 'lastaxis=2')

	M12<	M13	M16	M19	M25	M35	M45	M57	M65+
clothing	0.111	0.097	0.014	-0.112	-0.150	-0.118	-0.065	-0.012	0.044
accessories	0.029	0.022	0.002	-0.024	-0.031	-0.027	-0.019	-0.011	-0.002
tobacco	0.057	0.017	-0.010	-0.003	0.004	-0.023	-0.059	-0.094	-0.122
stationary	-0.132	-0.089	-0.003	0.089	0.114	0.110	0.098	0.085	0.064
books	0.023	0.015	0.000	-0.014	-0.018	-0.018	-0.018	-0.017	-0.015
records	0.011	0.008	0.000	-0.008	-0.010	-0.010	-0.008	-0.006	-0.004
household	0.023	0.014	0.000	-0.013	-0.016	-0.018	-0.018	-0.018	-0.017
candy	-0.074	-0.049	-0.001	0.048	0.061	0.061	0.055	0.049	0.039
toys	-0.122	-0.070	0.004	0.061	0.074	0.088	0.101	0.113	0.115
jewelry	-0.021	-0.013	0.000	0.012	0.015	0.016	0.016	0.016	0.015
perfumes	0.021	0.010	-0.001	-0.008	-0.009	-0.013	-0.018	-0.023	-0.026
hobby	0.048	0.007	-0.012	0.010	0.021	-0.011	-0.055	-0.098	-0.135
other	0.026	0.030	0.007	-0.039	-0.054	-0.036	-0.010	0.017	0.043

Eccentricity of ellipses

[1] 0.757

Ellipse axes, Area, p-values of rows

	HL Axis 1	HL Axis 2	Area	P-value
clothing	0.013	0.009	0	0.000
accessories	0.010	0.007	0	0.000
tobacco	0.011	0.007	0	0.000
stationary	0.010	0.007	0	0.000
books	0.012	0.008	0	0.000
records	0.008	0.005	0	0.000
household	0.015	0.010	0	0.000
candy	0.013	0.008	0	0.000
toys	0.011	0.007	0	0.000
jewelry	0.013	0.008	0	0.000
perfumes	0.015	0.010	0	0.297

hobby	0.011	0.007	0	0.000
other	0.011	0.007	0	0.000

	Ellipse axes, Area, p-values of columns			
	HL Axis 1	HL Axis 2	Area	P-value
M12<	0.034	0.022	0.002	0
M13	0.020	0.013	0.001	0
M16	0.020	0.013	0.001	0
M19	0.023	0.015	0.001	0
M25	0.025	0.016	0.001	0
M35	0.026	0.017	0.001	0
M45	0.031	0.020	0.002	0
M57	0.046	0.030	0.004	0
M65+	0.070	0.046	0.010	0

The total inertia of data, defined by the Goodman-Kruskal tau index (which may also be referred to as the index of predictability) when performing a non symmetrical correspondence analysis, is  $\tau = 0.0414$ ; in the output this is reflected by Tau Index predicting the rows given the column. To determine whether this index is statistically significant, we compute the C-statistic and find that it is equal to 10331.5 (with 96 degrees of freedom). Therefore, with a  $p$ -value that is less than 0.0001, the age of the perpetrators is a strong predictor of the items that are stolen. The Goodman-Kruskal tau index and the statistical significance of the C-statistic are summarized as part of the output, together with the partition of the C-statistic, which identifies significant sources of variation in the ordered column categories. Indeed, we can look at the inertia explained by each polynomial axis to mark differences with the other non-ordered analysis. We can see that the most dominant contribution to the total inertia of the data is due to the component associated with the linear polynomial of the columns. This location component is 6182 and explains 59.8% of the total inertia. The next most dominant is the dispersion component of 2642 and reflects that 25.6% of the variation in the column categories is due to their difference in dispersion. Similarly, the cubic component is 757 and accounts for about 7% of the column variation. Even if the remaining, higher order, components are all statistically significant (their associated  $p$ -value is less than 0.001), they will be not taken into consideration since polynomials with degree higher than three (and more commonly, four) show limited information about the association structure and variation of the variables. Hence, collectively, components higher than the fourth are referred to as the *error* polynomial component. Note that the first two components (linear and dispersion) explain 85.4% of the total inertia, so the first two polynomial axes will provide a sufficient graphical display of the variation of the categories. Furthermore, with the specification of `ellprint = TRUE` in the `print` method for "CAvariants" objects, the output consists of the eccentricity value of the ellipses, the semi-axis lengths of the ellipse for each of the categories, the area of each ellipse and the associated  $p$ -values.

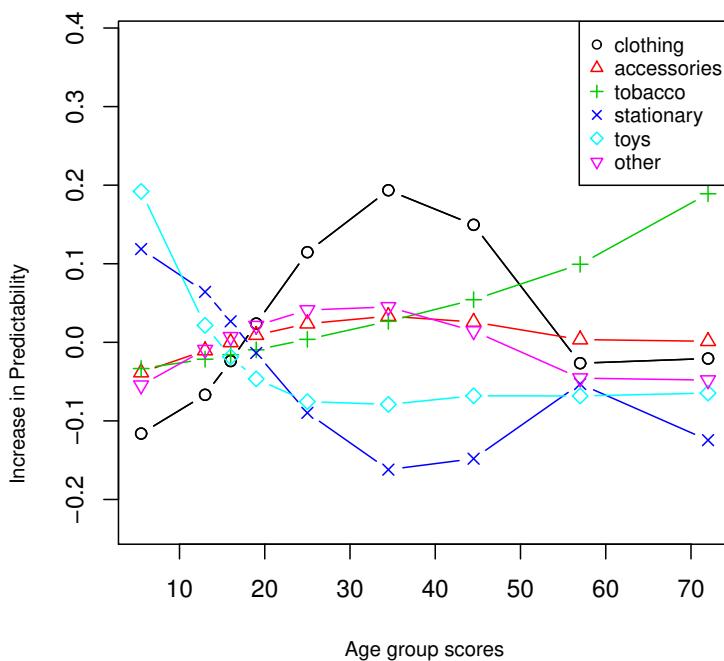
### Polynomial biplot: Portraying the predictability

When an ordered analysis is performed, the trend plots of the row and column categories are depicted. For example, when performing a singly ordered NSCA, the variation, or trend, of the row categories is examined by observing how it is affected by the ordered column categories when using a polynomial transformation. Figure 1 shows a parabolic trend of the row category *clothing*. This trend highlights that there is a greater propensity to steal clothing by people aged 25 to 45 years than those of a younger, or older, age. Figure 2 provides an alternative visual display of these trends and is constructed by depicting the row (items) categories using principal coordinates and the column (age) categories using standard coordinates. Hence a row isometric biplot is constructed. Since the analysis also incorporates the ordered nature of the column categories and the nominal structure of the row categories, Figure 2 is referred to as the row isometric polynomial biplot of the data.

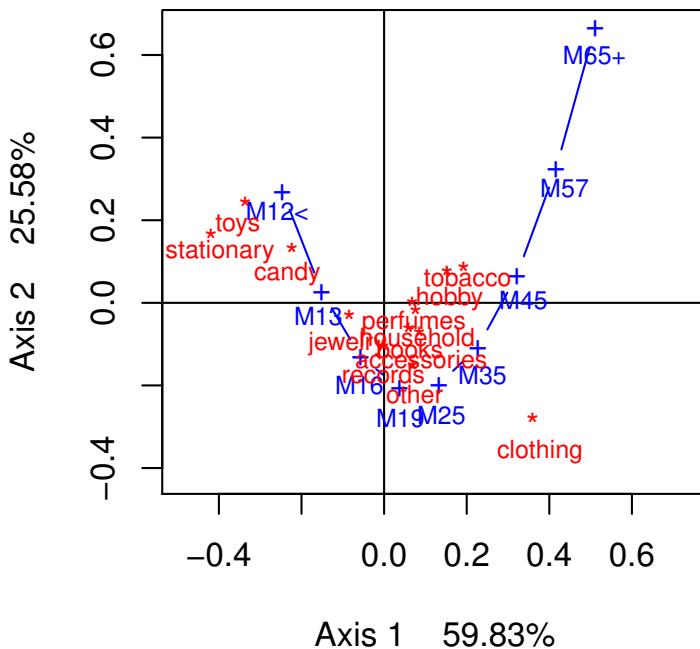
The trend plot of Figure 1 and the polynomial biplot given by Figure 2 can be obtained using the following command:

```
R> plot(res, plottype = "biplot", biptype = "row", scaleplot = 5, pos = 1)
```

When the first two polynomial axes are used to construct the biplot of Figure 2, the resulting configuration has a parabolic shape. Observe that the explained inertia of the polynomial axes is as follows: The first polynomial axis accounts for 59.8% of the inertia and the second polynomial axis for 25.6% of the inertia. We can therefore see that the novelty of the polynomial biplot is based on the polynomial representation of the predictor variable. The first linear polynomial axis represents the deviation from the mean centered profile accounting for the ordered structure of the age groups, which is reflected in the correct ordering of the age categories along the first polynomial axis. The second polynomial axis shows a parabolic shape of the categories with positive concavity. Furthermore, note that the left-hand



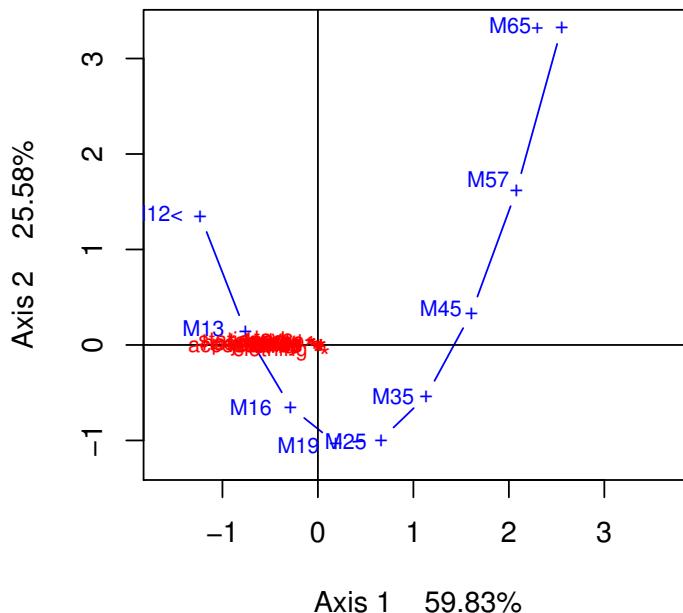
**Figure 1:** Trend of rows: A selection of rows of the centered column profile table reconstructed by using the first two polynomials.



**Figure 2:** Row-isometric polynomial biplot of singly ordered NSCA of shoplifting data: first two polynomial components, *Stolen goods* and *Age*.

side of the first axis is dominated by the young age groups with adolescents and young adults at the center of the display (who steal items consistent with the average number of thefts of all items). The mid-adult and older age groups are on the right-hand side of Figure 2.

The magnitude of the coordinates indicate the importance of the first two polynomial components for modeling the trends of the items. In particular, we see that the first two polynomial coordinates are sufficient to model the trends for most stolen goods. The reliability of the graphical representation can be assessed by constructing elliptical confidence regions for the row categories which are depicted using row principal polynomial coordinates. These ellipses can be obtained using the plot method for "CAvariants" objects such that



**Figure 3:** 95% confidence ellipses in the row isometric polynomial biplot of singly ordered NSCA of the shoplifting data: *Stolen goods and Age*.

```
R> plot(res, scaleplot = 1, ell = TRUE, alpha = 0.05)
```

Figure 3 gives the 95% confidence ellipses for the row categories and are constructed so that the weights of the semi-axes are expressed in terms of the hybrid generalized correlations rather than the squared singular values associated to each axis. These ellipses are constructed so that the information contained in all of the dimensions is depicted so that, for the plot method for “CAvariants” objects,  $M = 8$ . Since this figure does not show clearly ellipses for a scale problem of coordinates, we can focus our attention more closely to those points closer to the origin of Figure 3 by specifying that

```
R> plot(res, scaleplot = 1, ell = TRUE, alpha = 0.05, prop = 60)
```

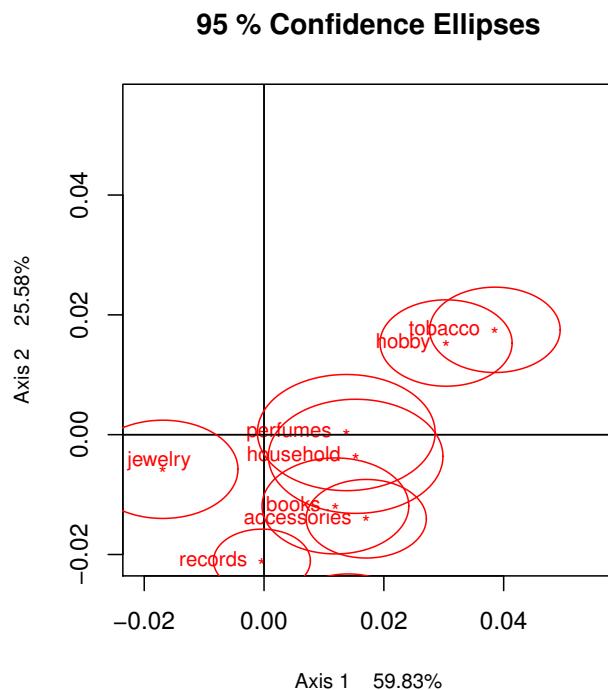
By zooming closer to the origin, the configuration of points near the origin is given by Figure 4. It shows the overlap of the confidence region for perfumes with the origin. It means that all of the items, except *perfumes*, are important contributors to the asymmetric association since their confidence ellipses do not overlap with the origin of the plot.

The contribution of all items to the association structure is also reflected in the  $p$ -values that are summarized as part of the output of the print method for “CAvariants” objects with  $M = 8$  and appear in the last column of the table, titled Ellipse axes,Area,p-values of rows where  $\alpha = 0.05$ . These results show that the only non-statistically significant row category is *perfumes*, as expected from its ellipse, with a  $p$ -value of 0.297. If we now consider the age of the males in the sample, and specify  $M = 8$  when constructing confidence ellipses and calculating  $p$ -values, see the last column of the table titled Ellipse axes,Area,p-values of columns, all age groups are useful predictors of the items that are stolen.

## Conclusion

There are many freely downloadable programs/code available for performing classical correspondence analysis. For example, the R code of Nenadic and Greenacre (2007) and De Leeuw and Mair (2009a) may be considered for performing simple and joint correspondence analysis. However, the CAvariants package provides variants of correspondence analysis which are not offered by other correspondence analysis R packages on CRAN. To the best of these authors' knowledge, CAvariants is the only package available that provides the user with the option of performing six variants of two-way correspondence analysis and, in particular, ordered symmetrical and non symmetrical correspondence analysis variants. Indeed, symmetrical correspondence analysis for ordered variables was implemented in SPLUS by Beh (2004b) and has been easily adapted for R.

Subsequent versions of the function may allow for more flexibility by giving the user more tools to assess the reliability of graphical results. These may include bootstrap confidence regions to



**Figure 4:** A zoomed view of the origin of the row-isometric polynomial biplot given by Figure 3.

complement the algebraic regions developed by these authors, or three-dimensional polynomial biplots. While Beh and Lombardo (2014) and Lombardo et al. (2016) describe the theoretical aspects of these variants of correspondence analysis for two-way contingency tables in detail, they also provide fragments of R code to undertake the relevant calculations. However, this paper has described the CAvariants package by demonstrating the applicability of one variant and providing new insight into the development of elliptical regions for ordered variants of correspondence analysis.

## Bibliography

- H. Abdi. Discriminant correspondence analysis. In N. J. Salkind, editor, *Encyclopedia of Measurement and Statistics*, pages 270–275. Sage Publications, Inc., 2007. [p168]
- G. Alberti. CAinterprTools: An R package to help interpreting correspondence analysis results. *SoftwareX*, 1–2:26–31, 2015. doi: 10.1016/j.softx.2015.07.001. [p167, 168]
- M. J. Baxter and H. E. M. Cool. Correspondence analysis in R for archaeologists: An educational account. *Archeologia e Calcolatori*, 21:211–228, 2010. [p168]
- D. Beaton, C. R. C. Fatt, and H. Abdi. An ExPosition of multivariate analysis with the singular value decomposition in R. *Computational Statistics & Data Analysis*, 72:176–189, 2014. doi: 10.1016/j.csda.2013.11.006. [p168]
- E. J. Beh. Simple correspondence analysis of ordinal cross-classifications using orthogonal polynomials. *Biometrical Journal*, 39:589–613, 1997. doi: 10.1002/bimj.4710390507. [p170]
- E. J. Beh. Partitioning Pearson’s chi-squared statistic for singly ordered two-way contingency tables. *The Australian and New Zealand Journal of Statistics*, 43:327–333, 2001. doi: 10.1111/1467-842x.00179. [p170]
- E. J. Beh. Simple correspondence analysis: A bibliographic review. *International Statistical Review*, 72: 257–284, 2004a. doi: 10.1111/j.1751-5823.2004.tb00236.x. [p167]
- E. J. Beh. S-PLUS code for ordinal correspondence analysis. *Computational Statistics*, 19:593–612, 2004b. doi: 10.1007/bf02753914. [p181]
- E. J. Beh. Simple correspondence analysis of nominal-ordinal contingency tables. *Journal of Applied Mathematics and Computer Sciences*, 8:1–12, 2008. doi: 10.1155/2008/218140. [p170]

- E. J. Beh. Elliptical confidence regions for simple correspondence analysis. *Journal of Statistical Planning and Inference*, 140:2582–2588, 2010. doi: 10.1016/j.jspi.2010.03.018. [p171, 172]
- E. J. Beh and R. Lombardo. *Correspondence Analysis: Theory, Practice and New Strategies*. John Wiley & Sons, 2014. doi: 10.1002/9781118762875. [p167, 168, 170, 171, 172, 173, 182]
- E. J. Beh and R. Lombardo. Confidence regions and  $p$ -values for classical and non-symmetric correspondence analysis. *Communications in Statistics – Theory and Methods*, 44:95–114, 2015. doi: 10.1080/03610926.2013.768665. [p171, 172]
- J. P. Benzécri. *Analyse des Données*. Dunod, Paris, 1973. [p167]
- D. J. Best and J. C. W. Rayner. Nonparametric analysis for doubly ordered two-way contingency tables. *Biometrics*, 52:1153–1156, 1996. doi: 10.2307/2533077. [p170]
- D. Chessel, A. B. Dufour, and J. Thioulouse. The ade4 package I: One-table methods. *R News*, 4(1):5–10, 2004. URL [https://www.R-project.org/doc/Rnews/Rnews\\_2004-1.pdf](https://www.R-project.org/doc/Rnews/Rnews_2004-1.pdf). [p168]
- J. G. Clavel, S. Nishisato, and A. Pita. *dualScale: Dual Scaling Analysis of Multiple Choice Data*, 2014. URL <https://CRAN.R-project.org/package=dualScale>. R package version 0.9.1. [p167]
- L. D’Ambra and N. C. Lauro. Non-symmetrical correspondence analysis for three-way contingency table. In R. Coppi and S. Bolasco, editors, *Multiway Data Analysis*, pages 301–315. Elsevier, Amsterdam, 1989. [p169]
- J. De Leeuw. Correspondence analysis in R. [www.cuddyvalley.org/psychoR/ca](http://www.cuddyvalley.org/psychoR/ca), 2006. [p167]
- J. De Leeuw and P. Mair. Simple and canonical correspondence analysis using the R package anacor. *Journal of Statistical Software*, 31(5):1–18, 2009a. doi: 10.18637/jss.v031.i01. [p167, 181]
- J. De Leeuw and P. Mair. Gifi methods for optimal scaling in R: The package homals. *Journal of Statistical Software*, 31(4):1–20, 2009b. doi: 10.18637/jss.v031.i04. [p167]
- S. Dray and A. B. Dufour. The ade4 package: Implementing the duality diagram for ecologists. *Journal of Statistical Software*, 22(4):1–20, 2007. doi: 10.18637/jss.v022.i04. [p168]
- P. L. Emerson. Numerical construction of orthogonal polynomials from a general recurrence formula. *Biometrics*, 24:696–701, 1968. doi: 10.2307/2528328. [p170]
- J. Gower, S. Lubbe, and N. le Roux. *Understanding Biplots*. John Wiley & Sons, Chichester, 2011. doi: 10.1002/9780470973196. [p173]
- M. Greenacre. *Theory and Application of Correspondence Analysis*. London Academic Press, London, 1984. [p167, 171]
- A. Israëls. *Eigenvalue Techniques for Qualitative Data*. DSWO Press, Leiden, 1987. [p175]
- B. Kostov, M. Bécue-Bertaut, and F. Husson. Correspondence analysis on generalised aggregated lexical tables (CA-GALT) in the FactoMineR package. *The R Journal*, 7(1):109–117, 2015. URL <https://journal.r-project.org/archive/2015-1/kostov-becuebertaut-husson.pdf>. [p167]
- P. M. Kroonenberg and R. Lombardo. Nonsymmetric correspondence analysis: A tool for analysing contingency tables with a dependence structure. *Multivariate Behavioral Research Journal*, 34:367–397, 1999. doi: 10.1207/s15327906mbr3403\_4. [p169]
- N. C. Lauro and L. D’Ambra. L’analyse non symmetrique des correspondances. In E. Diday, editor, *Data Analysis and Informatics III*, pages 433–446. Elsevier, Amsterdam, 1984. [p169]
- S. Lê, J. Josse, and F. Husson. FactoMineR: An R package for multivariate analysis. *Journal of Statistical Software*, 25(1):1–18, 2008. doi: 10.18637/jss.v025.i01. [p167]
- L. Lebart, A. Morineau, and K. M. Warwick. *Multivariate Descriptive Statistical Analysis*. John Wiley & Sons, New-York, USA, 1984. [p167, 171, 172]
- D. G. Leibovici. Spatio-temporal multiway decomposition using principal tensor analysis on  $k$ -modes: The R package PTAk. *Journal of Statistical Software*, 34(10):1–34, 2010. doi: 10.18637/jss.v034.i10. [p168]
- D. G. Leibovici. Principal tensor analysis on  $k$  modes. <http://c3s2i.free.fr/>, 2015. [p168]

- A. B. N. Librero, P. Willems, and P. G. Villardon. *cncaGUI: Canonical Non-Symmetrical Correspondence Analysis in R*, 2015. URL <https://CRAN.R-project.org/package=cncaGUI>. R package version 1.0. [p168]
- R. J. Light and B. H. Margolin. An analysis of variance for categorical data. *Journal of the American Statistical Association*, 66(335):534–544, 1971. doi: 10.1080/01621459.1971.10482297. [p170]
- M. Linting, J. J. Meulman, P. F. J. Groenen, and A. J. Van der Kooij. Stability of nonlinear principal components analysis: An empirical study using the balanced bootstrap. *Psychological Methods*, 12(3):359–379, 2007. doi: 10.1037/1082-989x.12.3.359. [p171]
- R. Lombardo and E. J. Beh. *CAvariants: Correspondence Analysis Variants*, 2017. URL <https://CRAN.R-project.org/package=CAvariants>. R package version 3.4. [p167]
- R. Lombardo and T. J. Ringrose. Bootstrap confidence regions in non-symmetrical correspondence analysis. *Electronic Journal of Applied Statistical Analysis*, 5:413–417, 2012. doi: 10.1080/00949655.2011.579968. [p171, 172]
- R. Lombardo, E. J. Beh, and P. M. Kroonenberg. Modelling trends in ordered correspondence analysis using orthogonal polynomials. *Psychometrika*, 81:325–349, 2016. doi: 10.1007/s11336-015-9448-y. [p168, 169, 170, 171, 175, 182]
- M. T. Markus. *Bootstrap Confidence Regions in Non-Linear Multivariate Analysis*. DSWO Press, 1994. [p171]
- F. Murtagh. *Correspondence Analysis and Data Coding with Java and R*. Chapman & Hall/CRC, Boca Raton, FL, 2005. doi: 10.1201/9781420034943. [p167, 168]
- O. Nenadic and M. Greenacre. Correspondence analysis in R, with two- and three-dimensional graphics: The ca package. *Journal of Statistical Software*, 20:1–13, 2007. doi: 10.18637/jss.v020.i03. [p167, 181]
- S. Nishisato. *Multidimensional Nonlinear Descriptive Analysis*. Taylor & Francis Group, LLC, 2007. [p167, 169]
- J. Oksanen, F. G. Blanchet, M. Friendly, R. Kindt, P. Legendre, D. McGlinn, P. R. Minchin, R. B. O'Hara, G. L. Simpson, P. Solymos, M. H. H. Stevens, E. Szoecs, and H. Wagner. *vegan: Community Ecology Package*, 2016. URL <https://CRAN.R-project.org/package=vegan>. R package version 2.4-1. [p168]
- J. C. W. Rayner and E. J. Beh. Towards a better understanding of correlation. *Statistica Neerlandica*, 63:324–333, 2009. doi: 10.1111/j.1467-9574.2009.00425.x. [p170]
- T. J. Ringrose. Bootstrap confidence regions for correspondence analysis. *Journal of Statistical Computation and Simulation*, 83:1397–1413, 2012. doi: 10.1080/00949655.2011.579968. [p167, 171]
- B. Ripley. *MASS: Support Functions and Datasets for Venables and Ripley's MASS*, 2016. URL <https://CRAN.R-project.org/package=MASS>. R package version 7.3-45. [p167]
- J. Thioulouse, D. Chessel, S. Dolédec, and J. M. Olivier. ADE-4: A multivariate analysis and graphical display software. *Statistics and Computing*, 7:75–83, 1997. doi: 10.1023/a:1018513530268. [p168]
- W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, 4th edition, 2002. doi: 10.1007/978-0-387-21706-2. [p167]
- J. L. Williams, H. Abdi, R. French, and B. J. Orange. A tutorial on multi-block discriminant correspondence analysis (MUDICA): A new method for analyzing discourse data from clinical populations. *Journal of Speech Language and Hearing Research*, 53:1372–1393, 2010. [p168]

Rosaria Lombardo

Department of Economics, University of Naples Campania “Luigi Vanvitelli”  
via Gran Priorato di Malta, Capua 81043  
Italy  
[rosaria.lombardo@unina2.it](mailto:rosaria.lombardo@unina2.it)

Eric J. Beh

School of Mathematical & Physical Sciences, University of Newcastle  
University Drive, Callaghan, NSW, 2308 Australia  
[eric.beh@newcastle.edu.au](mailto:eric.beh@newcastle.edu.au)

# hdm: High-Dimensional Metrics

by Victor Chernozhukov, Chris Hansen, Martin Spindler

**Abstract** In this article the package High-dimensional Metrics **hdm** is introduced. It is a collection of statistical methods for estimation and quantification of uncertainty in high-dimensional approximately sparse models. It focuses on providing confidence intervals and significance testing for (possibly many) low-dimensional subcomponents of the high-dimensional parameter vector. Efficient estimators and uniformly valid confidence intervals for regression coefficients on target variables (e.g., treatment or policy variable) in a high-dimensional approximately sparse regression model, for average treatment effect (ATE) and average treatment effect for the treated (ATET), as well for extensions of these parameters to the endogenous setting are provided. Theory grounded, data-driven methods for selecting the penalization parameter in Lasso regressions under heteroscedastic and non-Gaussian errors are implemented. Moreover, joint/ simultaneous confidence intervals for regression coefficients of a high-dimensional sparse regression are implemented. Data sets which have been used in the literature and might be useful for classroom demonstration and for testing new estimators are included.

## Introduction

Analysis of high-dimensional models, models in which the number of parameters to be estimated is large relative to the sample size, is becoming increasingly important. Such models arise naturally in readily available high-dimensional data which have many measured characteristics available per individual observation as in, for example, large survey data sets, scanner data, and text data. Such models also arise naturally even in data with a small number of measured characteristics in situations where the exact functional form with which the observed variables enter the model is unknown, and we create many technical variables, a dictionary, from the raw characteristics. Examples of this scenario include semiparametric models with nonparametric nuisance functions. More generally, models with many parameters relative to the sample size often arise when attempting to model complex phenomena.

For R, many packages for estimation high-dimensional models are already available. For example, **glmnet** (Friedman et al., 2010) and **lars** (Hastie and Efron, 2013) are popular for Lasso estimation. The section "Regularized & Shrinkage Methods" in the task view on "Machine Learning and Statistical Learning" contains further implementation of Lasso and related methods.

The methods which are implemented in this package (**hdm**) are chiefly distinct from already available methods in other packages in offering the following four major features:

- 1) First, we provide a version of Lasso regression that expressly handles and allows for non-Gaussian and heteroscedastic errors.
- 2) Second, we implement a theoretically grounded, data-driven choice of the penalty level  $\lambda$  in the Lasso regressions. To underscore this choice, we call the Lasso implementation in this package "rigorous" Lasso (=rlasso). The prefix r in function names should underscore this. In high-dimensional settings cross-validation is very popular; but it lacks a theoretical justification for use in the present context and some theoretical proposals for the choice of  $\lambda$  are often not feasible. Moreover, the theoretically grounded, data-driven choice makes cross-validation redundant, as it is time-consuming particularly in large data sets.
- 3) Third, we provide efficient estimators and uniformly valid confidence intervals for various low-dimensional causal/structural parameters appearing in high-dimensional approximately sparse models.<sup>1</sup> For example, we provide efficient estimators and uniformly valid confidence intervals for a regression coefficient on a target variable (e.g., a treatment or policy variable) in a high-dimensional sparse regression model. Target variable in this context means the object not interest, e.g. a prespecified regression coefficient. We also provide estimates and confidence intervals for average treatment effect (ATE) and average treatment effect for the treated (ATET), as well extensions of these parameters to the endogenous setting.
- 4) Fourth, joint/ simultaneous confidence intervals for estimated coefficients in a high-dimensional approximately sparse models are provided, based on the methods and theory developed in Belloni et al. (2014b). They proposed uniformly valid confidence regions for regressions coefficients in a high-dimensional sparse Z-estimation problems, which include median, mean, and many other regression problems as special cases. In this article we apply this method to the coefficients of a Lasso regression and highlight this method with an empirical example.

<sup>1</sup>A formal definition of approximately sparse models can be found in the accompanying vignette.

In the following we will give a short overview over the functionality of the package with simple examples. A detailed introduction on how to use the functions is given in the accompanying vignette.

## A very short guide to the literature

In this section we give a very short introduction to the literature with a focus on the methods which are implemented in the package.

Lasso under heteroscedastic and non-Gaussian errors was analysed in [Belloni et al. \(2012\)](#). Post-Lasso, i.e. a least squares fit with the variables selected by a Lasso regression, is introduced and analysed in [Belloni and Chernozhukov \(2013\)](#). Inference for a low-dimensional component of a potentially high-dimensional vector has been conducted in [Belloni et al. \(2014a\)](#). [Belloni et al. \(2014a\)](#) also consider inference on average treatment effects in a heterogeneous treatment effect setting after selection amongst high-dimensional controls. Inference in high-dimensional settings is enabled by the so-called orthogonality condition. The systematic development of the orthogonality condition for inference on low-dimensional parameters in high-dimensional settings is given in [Chernozhukov et al. \(2015a\)](#).

Instrumental variables estimation is a central topic in Econometrics and also becoming more popular in fields such as Biostatistics, Epidemiology, and Sociology. Good introductions to instrumental variables and treatment effects are the books [Angrist and Pischke \(2008\)](#) and [Imbens and Rubin \(2015\)](#). The case of selection on high-dimensional instrumental variables is given in [Belloni et al. \(2012\)](#), the case of selection on the instruments and control variables in [Chernozhukov et al. \(2015b\)](#). For further discussion of estimation of treatment effects in a high-dimensional setting including cases with endogenous treatment assignment, we refer to [Belloni et al. \(2013\)](#).

## Estimation of Lasso under heteroscedastic and non-Gaussian errors and inference for low-dimensional subcomponents

An important feature of our package is that it allows Lasso estimation under heteroscedastic and non-Gaussian errors. This distinguishes the package **hdm** from other already available software implementations of Lasso. As an additional benefit, the theoretical grounded choice of the penalty does not require cross-validation which might lead to a considerable saving of computation time in many applications.

### Prediction in linear models using approximate sparsity

Consider high dimensional approximately sparse linear regression models. These models have a large number of regressors  $p$ , possibly much larger than the sample size  $n$ , but only a relatively small number  $s = o(n)$  of these regressors are important for capturing accurately the main features of the regression function. The latter assumption makes it possible to estimate these models effectively by searching for approximately the right set of regressors.

The model reads

$$y_i = x_i' \beta_0 + \varepsilon_i, \quad \mathbb{E}[\varepsilon_i x_i] = 0, \quad \beta_0 \in \mathbb{R}^p, \quad i = 1, \dots, n$$

where  $y_i$  are observations of the response variable,  $x_i = (x_{i,1}, \dots, x_{i,p})'$ 's are observations of  $p$ -dimensional regressors, and  $\varepsilon_i$ 's are centered disturbances, where possibly  $p \gg n$ . Assume that the data sequence is i.i.d. for the sake of exposition, although the framework covered is considerably more general. An important point is that the errors  $\varepsilon_i$  may be non-Gaussian or heteroskedastic ([Belloni et al., 2012](#)).

The model can be exactly sparse, namely

$$\|\beta_0\|_0 \leq s = o(n),$$

or approximately sparse, namely that the values of coefficients, sorted in decreasing order,  $(|\beta_0|_{(j)})_{j=1}^p$  obey,

$$|\beta_0|_{(j)} \leq A j^{-a(\beta_0)}, \quad a(\beta_0) > 1/2, \quad j = 1, \dots, p.$$

An approximately sparse model can be well-approximated by an exactly sparse model with sparsity index

$$s \propto n^{1/(2a(\beta_0))}.$$

In order to get theoretically justified performance guarantees, we consider the Lasso estimator

with data-driven penalty loadings:

$$\hat{\beta} = \arg \min_{\beta \in \mathbb{R}^p} \mathbb{E}_n[(y_i - x'_i \beta)^2] + \frac{\lambda}{n} \|\hat{\Psi}\beta\|_1$$

where  $\|\beta\|_1 = \sum_{j=1}^p |\beta_j|$ ,  $\hat{\Psi} = \text{diag}(\hat{\psi}_1, \dots, \hat{\psi}_p)$  is a diagonal matrix consisting of penalty loadings, and  $\mathbb{E}_n$  abbreviates the empirical average. The penalty loadings are chosen to insure basic equivariance of coefficient estimates to rescaling of  $x_{i,j}$  and can also be chosen to address heteroskedasticity in model errors. We discuss the choice of  $\lambda$  and  $\hat{\Psi}$  below.

Regularization by the  $\ell_1$ -norm naturally helps the Lasso estimator to avoid overfitting, but it also shrinks the fitted coefficients towards zero, causing a potentially significant bias. In order to remove some of this bias, consider the Post-Lasso estimator that applies ordinary least squares to the model  $\hat{T}$  selected by Lasso, formally,

$$\hat{T} = \text{support}(\hat{\beta}) = \{j \in \{1, \dots, p\} : |\hat{\beta}| > 0\}.$$

The Post-Lasso estimate is then defined as

$$\tilde{\beta} \in \arg \min_{\beta \in \mathbb{R}^p} \mathbb{E}_n \left( y_i - \sum_{j=1}^p x_{i,j} \beta_j \right)^2 : \beta_j = 0 \quad \text{if } \hat{\beta}_j = 0, \quad \forall j.$$

In words, the estimator is ordinary least squares applied to the data after removing the regressors that were not selected by Lasso. The Post-Lasso estimator was introduced and analysed in [Belloni and Chernozhukov \(2013\)](#).

A crucial matter is the choice of the penalization parameter  $\lambda$ . With the right choice of the penalty level, Lasso and Post-Lasso estimators possess excellent performance guarantees: They both achieve the near-oracle rate for estimating the regression function, namely with probability  $1 - \gamma - o(1)$ ,

$$\sqrt{\mathbb{E}_n[(x'_i(\hat{\beta} - \beta_0))^2]} \lesssim \sqrt{(s/n) \log p}.$$

In high-dimensions setting, cross-validation is very popular in practice but lacks theoretical justification and so may not provide such a performance guarantee. In sharp contrast, the choice of the penalization parameter  $\lambda$  in the Lasso and Post-Lasso methods in this package is theoretical grounded and feasible. Therefore we call the resulting method the “rigorous”Lasso method and hence add a prefix **r** to the function names.

In the case of homoscedasticity, we set the penalty loadings  $\hat{\psi}_j = \sqrt{\mathbb{E}_n x_{i,j}^2}$ , which insures basic equivariance properties. There are two choices for penalty level  $\lambda$ : the X-independent choice and X-dependent choice. In the X-independent choice we set the penalty level to

$$\lambda = 2c\sqrt{n}\hat{\sigma}\Phi^{-1}(1 - \gamma/(2p)),$$

where  $\Phi$  denotes the cumulative standard normal distribution,  $\hat{\sigma}$  is a preliminary estimate of  $\sigma = \sqrt{\mathbb{E}\varepsilon^2}$ , and  $c$  is a theoretical constant, which is set to  $c = 1.1$  by default for the Post-Lasso method and  $c = .5$  for the Lasso method, and  $\gamma$  is the probability level, which is set to  $\gamma = .1$  by default. The parameter  $\gamma$  can be interpreted as the probability of mistakenly not removing X’s when all of them have zero coefficients. In the X-dependent case the penalty level is calculated as

$$\lambda = 2c\hat{\sigma}\Lambda(1 - \gamma|X),$$

where

$$\Lambda(1 - \gamma|X) = (1 - \gamma) - \text{quantile of } n\|\mathbb{E}_n[x_i e_i]\|_\infty |X|,$$

where  $X = [x_1, \dots, x_n]'$  and  $e_i$  are iid  $N(0, 1)$ , generated independently from  $X$ ; this quantity is approximated by simulation. The X-independent penalty is more conservative than the X-dependent penalty. In particular the X-dependent penalty automatically adapts to highly correlated designs, using less aggressive penalization in this case [Belloni et al. \(2010\)](#).

In the case of heteroskedasticity, the loadings are set to  $\hat{\psi}_j = \sqrt{\mathbb{E}_n[x_{ij}^2 \hat{\varepsilon}_i^2]}$ , where  $\hat{\varepsilon}_i$  are preliminary estimates of the errors. The penalty level can be X-independent ([Belloni et al., 2012](#)):

$$\lambda = 2c\sqrt{n}\Phi^{-1}(1 - \gamma/(2p)),$$

or it can be X-dependent and estimated by a multiplier bootstrap procedure ([Chernozhukov et al.,](#)

2013):

$$\lambda = c \times c_W(1 - \gamma),$$

where  $c_W(1 - \gamma)$  is the  $1 - \gamma$ -quantile of the random variable  $W$ , conditional on the data, where

$$W := n \max_{1 \leq j \leq p} |2\mathbb{E}_n[x_{ij}\hat{\varepsilon}_i e_i]|,$$

where  $e_i$  are iid standard normal variables distributed independently from the data, and  $\hat{\varepsilon}_i$  denotes an estimate of the residuals.

Estimation proceeds by iteration. The estimates of residuals  $\hat{\varepsilon}_i$  are initialized by running least squares of  $y_i$  on five regressors that are most correlated to  $y_i$ . This implies conservative starting values for  $\lambda$  and the penalty loadings, and leads to the initial Lasso and Post-Lasso estimates, which are then further updated by iteration. The resulting iterative procedure is fully justified in the theoretical literature.

## R Implementation

The core of the package are the functions `rlasso` and `rlassoEffects`. They allow estimation of a Lasso regression under heteroscedastic and non-Gaussian errors and inference on a set of prespecified, low-dimensional objects. As Lasso regression introduces shrinkage bias, a post-Lasso option, which can be described as Lasso estimation followed by a final refit with `ols` including only the selected variables, is available.

The function `rlasso` implements Lasso and post-Lasso. The default option is to use post-Lasso, `post=TRUE`. The user can also decide if an unpenalized intercept should be included (`TRUE` by default), `LassoShooting.fit` contains the computational algorithm that underlies the estimation procedure. This function implements a version of the Shooting Lasso Algorithm (Fu, 1998). The option `penalty` of the function `rlasso` allows different choices for the penalization parameter and loadings. It allows for homoscedastic or heteroscedastic errors with default `homoscedastic = FALSE`. Moreover, the dependence structure of the design matrix might be taken into consideration for calculation of the penalization parameter with `X.dependent.lambda = TRUE`. With the option `lambda.start` initial values for the algorithm can be set.<sup>2</sup>

The option `penalty` allows to set the constants  $c$  and  $\gamma$  which are necessary for the calculation of the penalty. For a detailed description how the penalty and variable loadings are calculated, we refer to the accompanying vignette. The maximum number of iterations and the tolerance when the algorithms should stop can be set with `control`.

`rlasso` returns an object of S3 class `rlasso` for which methods like `predict`, `print`, `summary` are provided. The methods `print` and `summary` have the option `all`. By setting this option to `FALSE` only the coefficients estimated to be non-zero are shown.

The function `rlassoEffects` does inference for prespecified target variables. Those can be specified either by the variable names, an integer valued vector giving their position in `x`, or by a logical vector indicating the variables for which inference should be conducted. It returns an object of S3 class `rlassoEffects` for which the methods `summary`, `print`, `confint`, and `plot` are provided. `rlassoEffects` is a wrapper function for `rlassoEffect` which does inference for a single target regressor.

The function `rlassoEffects` might either be used in the form `rlassoEffects(x, y, index)` where `x` is a matrix, `y` denotes the outcome variable and `index` specifies the variables of `x` for which inference is conducted. This can done by an integer vector (position of the variables), a logical vector or the name of the variables. An alternative usage is as `rlassoEffects(formula, data, I)` where `I` is a one-sided formula which specifies the variables for which inference is conducted. For further details we refer to the help page of the function and the following examples where both methods for usage are shown.

For logistic regression the corresponding functions are implemented in an analogous manner, named `rlassologit` and `rlassologitEffects`.

## Example

First, we generate a data set on which we demonstrate the basic functions:

---

<sup>2</sup>In some cases the user might want to use a predefined, fixed penalization parameter. This can be done by a special option in the following way: set `homoscedastic` to 'none' and supply a value (vector) `lambda.start`. Then this value is used as penalty parameter with independent design and heteroscedastic errors to weight the regressors.

```

set.seed(12345)
> n <- 100 #sample size
> p <- 100 # number of variables
> s <- 3 # number of variables with non-zero coefficients
> X <- matrix(rnorm(n*p), ncol=p)
> beta <- c(rep(5,s), rep(0,p-s))
> Y <- X%*%beta + rnorm(n)

```

Next we use Lasso for fitting and prediction, show the results and make predictions for the old and for new X-variables

```

> lasso.reg <- rlasso(Y~X, post=FALSE, intercept=TRUE) # use Lasso, not-post-Lasso
> # lasso.reg <- rlasso(X,Y, post=FALSE, intercept=TRUE) # alternative use
> summary(lasso.reg, all=FALSE)

```

Call:

```
rlasso.formula(formula = Y ~ X, post = FALSE, intercept = TRUE)
```

Post-Lasso Estimation: FALSE

Total number of variables: 100

Number of selected variables: 11

Residuals:

	Min	1Q	Median	3Q	Max
	-2.09008	-0.45801	-0.01237	0.50291	2.25098

	Estimate
(Intercept)	0.057
1	4.771
2	4.693
3	4.766
13	-0.045
15	-0.047
16	-0.005
19	-0.092
22	-0.027
40	-0.011
61	0.114
100	-0.025

Residual standard error: 0.8039

Multiple R-squared: 0.9913

Adjusted R-squared: 0.9902

Joint significance test:

the sup score statistic for joint significance test is 64.02 with a p-value of 0

```

> yhat.lasso <- predict(lasso.reg) #in-sample prediction
> Xnew <- matrix(rnorm(n*p), ncol=p) # new X
> Ynew <- Xnew%*%beta + rnorm(n) #new Y
> yhat.lasso.new <- predict(lasso.reg, newdata=Xnew) #out-of-sample prediction

```

To reduce bias, now use post-Lasso for fitting and prediction

```

> post.lasso.reg <- rlasso(Y~X, post=TRUE, intercept=TRUE)
> summary(post.lasso.reg, all=FALSE)

```

Call:

```
rlasso.formula(formula = Y ~ X, post = TRUE, intercept = TRUE)
```

Post-Lasso Estimation: TRUE

Total number of variables: 100

Number of selected variables: 3

```

Residuals:
    Min      1Q  Median      3Q     Max 
-2.83981 -0.80339  0.02063  0.75573  2.30421 

            Estimate
(Intercept) 0.000
1            5.150
2            4.905
3            4.912

Residual standard error: 1.059
Multiple R-squared:  0.9855
Adjusted R-squared:  0.9851
Joint significance test:
  the sup score statistic for joint significance test is 66.87 with a p-value of      0

> yhat.postlasso <- predict(post.lasso.reg) #in-sample prediction
> yhat.postlasso.new <- predict(post.lasso.reg, newdata=Xnew) #out-of-sample prediction

```

We can do inference on a set of variables of interest, e.g. the first, second, third, and the fiftieth:

```

> lasso.effect <- rlassoEffects(x=X, y=Y, index=c(1,2,3,50))
> print(lasso.effect)

```

Call:  
`rlassoEffects.default(x = X, y = Y, index = c(1, 2, 3, 50))`

Coefficients:

	V1	V2	V3	V50
5.0890	4.7781	4.8292	0.1384	

```

> summary(lasso.effect)
[1] "Estimates and significance testing of the effect of target variables"
   Estimate Std. Error t value Pr(>|t|)    
V1      5.0890    0.1112  45.781  <2e-16 ***
V2      4.7781    0.1318  36.264  <2e-16 ***
V3      4.8292    0.1314  36.752  <2e-16 ***
V50     0.1384    0.1122   1.234    0.217   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The confidence intervals for the coefficients are given by

```

> confint(lasso.effect)
  2.5 %   97.5 %
> confint(lasso.effect)
  2.5 %   97.5 %
V1    4.8710981 5.306834
V2    4.5198436 5.036332
V3    4.5716439 5.086726
V50   -0.0814167 0.358284

```

Moreover, simultaneous / joint confidence intervals can be calculated which will be discussed in more detail later.

```

> confint(lasso.effect, joint = TRUE)
  2.5 %   97.5 %
V1    4.8464078 5.3315239
V2    4.4315649 5.1246107
V3    4.4864563 5.1719131
V50   -0.1720149 0.4488822

```

Finally, we can also plot the estimated effects with their confidence intervals:

```
> plot(lasso.effect, main="Confidence Intervals")
```

A strength of the package is that it can also handle non-Gaussian and heteroscedastic error what we illustrate with a small example:

```

> Y <- X%*%beta + rnorm(n, sd=sin(X%*%beta)^2)
> Y <- X%*%beta + rt(n, df=5)
> lasso.reg = rlasso(Y~X, post=FALSE, intercept=TRUE) # use Lasso, not-post-Lasso
> summary(lasso.reg, all=FALSE)

Call:
rlasso.formula(formula = Y ~ X, post = FALSE, intercept = TRUE)

Post-Lasso Estimation: FALSE

Total number of variables: 100
Number of selected variables: 8

Residuals:
    Min      1Q  Median      3Q      Max 
-2.51449 -0.76567  0.09798  0.80007  2.11780 

Estimate
(Intercept) 0.032
1            4.959
2            4.686
3            4.671
8            0.083
21           0.010
36           0.047
58           -0.070
100          -0.133

Residual standard error: 1.051
Multiple R-squared:  0.9858
Adjusted R-squared:  0.9845
Joint significance test:
the sup score statistic for joint significance test is 66.87 with a p-value of 0

```

## Joint/ simultaneous confidence intervals

### Introduction

Belloni et al. (2014b) provide uniformly valid confidence intervals for  $p_1$  target parameters which are defined via Huber's Z-problems. The Z-framework encompasses, among other things, mean regression, median regression, generalized linear models, as well as many other methods. The dimension  $p_1$  of the target parameter might be potentially much larger than the sample size. Here we apply their results to the Lasso regression which can be embedded into this framework. This enables the provision of uniformly valid joint/ simultaneous confidence intervals for the target parameters. This functionality is implemented via the functions `rlassoEffects` and `confint`. To get joint confidence intervals the option `joint` of the latter function has to be set to `TRUE`. In the next section we illustrate this with an empirical illustration, analysing the effect of gender on wage, to quantify potential discrimination.

### Empirical application: the effect of gender on wage

In Labour Economics an important question is how the wage is related to the gender of the employed. We use US census data from the year 2012 to analyse the effect of gender and interaction effects of other variables with gender on wage jointly. The dependent variable is the logarithm of the wage, the target variable is `female` (in combination with other variables). All other variables denote some other socio-economic characteristics, e.g. marital status, education, and experience. For a detailed description of the variables we refer to the help page.

First, we load and prepare the data.

```

> library(hdm)
> data(cps2012)
> X <- model.matrix(~ -1 + female + female:(widowed + divorced + separated +
+ nevermarried + hsd08 + hsd911 + hsg + cg + ad + mw + so + we + exp1 + exp2 + exp3) +
+ (widowed + divorced + separated + nevermarried +

```

```
+ hsd08 + hsd911 + hsg + cg + ad + mw + so + we + exp1 + exp2 + exp3)^2, data=cps2012)
> dim(X)
[1] 29217 136
> X <- X[,which(apply(X, 2, var)!=0)] # exclude all constant variables
> dim(X)
[1] 29217 116
> index.gender <- grep("female", colnames(X))
> y <- cps2012$lnw
```

The parameter estimates for the target parameters, i.e. all coefficients related to gender (i.e. by interaction with other variables) are calculated and summarized by the following commands

```
> effects.female <- rlassoEffects(x=X, y=y, index=index.gender)
> summary(effects.female)
[1] "Estimates and significance testing of the effect of target variables"
      Estimate Std. Error t value Pr(>|t|)
female        -0.154923  0.050162 -3.088 0.002012 **
female:widowed  0.136095  0.090663  1.501 0.133325
female:divorced  0.136939  0.022182  6.174 6.68e-10 ***
female:separated  0.023303  0.053212  0.438 0.661441
female:nevermarried  0.186853  0.019942  9.370 < 2e-16 ***
female:hsd08     0.027810  0.120914  0.230 0.818092
female:hsd911    -0.119335  0.051880 -2.300 0.021435 *
female:hsg       -0.012890  0.019223 -0.671 0.502518
female:cg        0.010139  0.018327  0.553 0.580114
female:ad        -0.030464  0.021806 -1.397 0.162405
female:mw       -0.001063  0.019192 -0.055 0.955811
female:so        -0.008183  0.019357 -0.423 0.672468
female:we       -0.004226  0.021168 -0.200 0.841760
female:exp1      0.004935  0.007804  0.632 0.527139
female:exp2      -0.159519  0.045300 -3.521 0.000429 ***
female:exp3      0.038451  0.007861  4.891 1.00e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Finally, we estimate and plot confident intervals, first "pointwise" and then the joint confidence intervals.

```
> joint.CI <- confint(effects.female, level = 0.95, joint = TRUE)
> joint.CI
              2.5 %      97.5 %
female        -0.244219905 -0.06562666
female:widowed -0.036833704  0.30902467
female:divorced  0.097104065  0.17677471
female:separated -0.066454223  0.11305975
female:nevermarried  0.149932792  0.22377417
female:hsd08     -0.230085134  0.28570576
female:hsd911    -0.215291089 -0.02337899
female:hsg       -0.046383541  0.02060398
female:cg        -0.023079946  0.04335705
female:ad        -0.072370082  0.01144259
female:mw       -0.035451308  0.03332443
female:so        -0.043143587  0.02677690
female:we       -0.043587197  0.03513494
female:exp1      -0.008802488  0.01867301
female:exp2      -0.239408202 -0.07963045
female:exp3      0.024602478  0.05229868
> plot(effects.female, joint=TRUE, level=0.95)
```

This analysis allows a closer look how discrimination according to gender is related to other socio-economic variables.

As a side remark, the version 0.2 allows also now a formula interface for many functions including `rlassoEffects`. Hence, the analysis could also be done more compact as

```
> effects.female <- rlassoEffects(lnw ~ female + female:(widowed + divorced + separated +
+ nevermarried + hsd08 + hsd911 + hsg + cg + ad + mw + so + we + exp1 + exp2 + exp3) +
```

```
+ (widowed + divorced + separated + nevermarried +
+ hsd08 + hsd911 + hsg + cg + ad + mw + so + we + exp1 + exp2 + exp3)^2, data=cps2012,
+ I = ~ female + female:(widowed + divorced + separated + nevermarried +
+ hsd08 + hsd911 + hsg + cg + ad + mw + so + we + exp1 + exp2 + exp3))
```

The one-sided option `I` gives the target variables for which inference is conducted.

## Data sets

The package contains six data sets which are made available for researchers. The growth data set contains the so-called Barro-Lee data (Barro and Lee, 1994). It contains macroeconomics information for large set of countries over several decades to analyse the drivers of economic growth. Next, the package includes a data set on settler mortality and economic outcomes which was used in the literature to illustrate the effect of institutions on economic growth (Acemoglu et al., 2001). The data set pension contains information about the choice of 401(k) pension plans and savings behaviour in the United States (Chernozhukov and Hansen, 2004). A data set for analysing the effect of judicial decisions regarding eminent domain on economic outcomes (e.g. house prices), is included. More information is given in Chen and Sethi (2010) and Belloni et al. (2012). The BLP data (Berry et al., 1995) set was analysed in Berry, Levinsohn and Pakes (1995). The data stem from annual issues of the Automotive News Market Data Book. The data set inlcudes information on all models marketed during the the period beginning 1971 and ending in 1990 cotaining 2217 model/years from 997 distinct models. Finally, US census data (CPS2012) is included.

All data sets are described in their help pages where also further references can be found.

## Further functions for inference on structural parameters

Additionally, the package contains functions for estimation and inference on structural parameters in a high-dimensional setting. An important tool for estimation in the potential presence of endogeneity is instrumental variable (IV) estimation. The function `rlassoIV` implements methods for IV estimation in a high-dimensional setting. This function is a wrapper for several methods. The user can specify if selection shall be done on the instruments (z-variables), on the control variables (x-variables), or on both. For the low-dimensional case where no selection is done for the x- or z-variables `rlassoIV` performs classical 2SLS estimation by calling the function `tsls`.

The goal of many empirical analyses is to understand and estimate the causal effect of a treatment, e.g. the effect of an endogenous binary treatment  $D$ , on a outcome,  $Y$ , in the presence of a binary instrumental variable,  $Z$ , in settings with very many potential control variables. We provide functions to estimate the local average treatment effect (LATE), the local average treatment effect of the treated (LATET) and as special cases the average treatment effect (ATE) and the average treatment effect of the treated (ATET) in this setting in functions `rlassoLATE`, `rlassoATE`, `rlassoATE`, and `rlassoATET` respectively. These functions also allow calculation of treatment effect standard errors with different bootstrap methods ("wild", "normal", "Bayes") besides to the classical plug-in standard errors. Moreover, methods `print`, `summary`, and `confint`, are available.

Both the family of `rlassoIV`-functions and the family of the functions for treatment effects , which are introduced in the next section, allow use with both formula-interface and by handing over the prepard model matrices. Hence the general pattern for use with formula is `function(formula,data,...)` where formula consists of two-parts and is a member of the class `Formula`. These formulas are of the pattern  $y \sim d + x_1 | x_2 + z$  where  $y$  is the outcome variable,  $x$  are exogenous variables,  $d$  endogenous varialbes (if several ones are allowed depends on the concrete function), and  $z$  denote the instrumental variables. A more primitive use of the functions is by simply hand over the required group of variables as matrices: `function(x=x,d=d,y=y,z=z)`. In some of the following examples both alternatives are displayed.

### Example: IV estimation

In this section, we briefly present how instrumental variables estimation is conducted in a high-dimensional setting. The eminent domain example, for which the data set is contained in the package, serves as an illustration. The underlying goal is to estimate the effect of pro-plaintiff decisions in cases of eminent domain (government's takings of private property) on economic outcomes. The analysis of the effects of such decisions is complicated by the possible endogeneity between judicial decisions and potential economic outcomes. To address the potential endogeneity, we employ an instrumental

variables strategy based on the random assignment of judges to the federal appellate panels that make the decisions. Because judges are randomly assigned to three-judge panels, the judges and their demographics are randomly assigned conditional on the distribution of characteristics of federal circuit court judges in a given circuit-year.

First, we load the data and construct the matrices with the controls ( $x$ ), instruments ( $z$ ), outcome ( $y$ ), and treatment variables ( $d$ ). Here we consider regional GDP as the outcome variable.

```
> data(EminentDomain)
> z <- EminentDomain$logGDP$z
> x <- EminentDomain$logGDP$x
> y <- EminentDomain$logGDP$y
> d <- EminentDomain$logGDP$d
```

As mentioned above,  $y$  is the economic outcome, the logarithm of the GDP,  $d$  the number of pro plaintiff appellate takings decisions in federal circuit court  $c$  and year  $t$ ,  $x$  is a matrix with control variables, and  $z$  is the matrix with instruments. Here we consider socio-economic and demographic characteristics of the judges as instruments.

Next, we estimate the model with selection on the instruments.

```
> lasso.IV.Z <- rlassoIV(x=x, d=d, y=y, z=z, select.X=FALSE, select.Z=TRUE)
> # or lasso.IV.Z <- rlassoIVselectZ(x=X, d=d, y=y, z=Z)
> summary(lasso.IV.Z)
[1] "Estimates and significance testing of the effect of target variables in the
    IV regression model"
  coeff.    se. t-value p-value
d1 0.01543 0.01926  0.801   0.423
```

Finally, we do selection on both the  $x$  and  $z$  variables.

```
> lasso.IV.XZ <- rlassoIV(x=x, d=d, y=y, z=z, select.X=TRUE, select.Z=TRUE)
> summary(lasso.IV.XZ)
Estimates and Significance Testing of the effect of target variables in the
    IV regression model
  coeff.    se. t-value p-value
d1 -0.03488 0.15881 -0.22   0.826

> confint(lasso.IV.XZ)
  2.5 %  97.5 %
d1 -0.3461475 0.2763868
```

### Example: treatment effects

Here we apply the treatment functions to 401(k) plan participation. Though it is clear that 401(k) plans are widely used as vehicles for retirement saving, their effect on assets is less clear. The key problem in determining the effect of participation in 401(k) plans on accumulated assets is saver heterogeneity coupled with nonrandom selection into participation states. In particular, it is generally recognized that some people have a higher preference for saving than others. Thus, it seems likely that those individuals with the highest unobserved preference for saving would be most likely to choose to participate in tax-advantaged retirement savings plans and would also have higher savings in other assets than individuals with lower unobserved saving propensity. This implies that conventional estimates that do not allow for saver heterogeneity and selection of the participation state will be biased upward, tending to overstate the actual savings effects of 401(k) and IRA participation.

Again, we start first with the data preparation. For a detailed description of the data set we refer to the help page `help(pension)`.

```
data(pension)
y <- pension$tw; d = pension$p401; z = pension$e401
X <- pension[,c("i2", "i3", "i4", "i5", "i6", "i7", "a2", "a3", "a4", "a5",
               "fsize", "hs", "smcol", "col", "marr", "twoearn", "db", "pira", "hown")]
# simple model
xvar <- c("i2", "i3", "i4", "i5", "i6", "i7", "a2", "a3", "a4", "a5",
         "fsize", "hs", "smcol", "col", "marr", "twoearn", "db", "pira", "hown")
xpart <- paste(xvar, collapse = "+")
form <- as.formula(paste("tw ~ ", paste(c("p401", xvar), collapse = "+"), "|",
                        paste(xvar, collapse = "+")))
```

```
formZ <- as.formula(paste("tw ~ ", paste(c("p401", "xvar"), collapse = "+"), "|",
  paste(c("e401", "xvar"), collapse = "+")))
```

Now we can compute the estimates of the target treatment effect parameters:

```
> #pension.ate = rlassoATE(X,d,y)
> pension.ate = rlassoATE(form, data = pension)
> summary(pension.ate)
Estimation and significance testing of the treatment effect
Type: ATE
Bootstrap: not applicable
  coeff.  se. t-value p-value
TE 10490 1920  5.464 4.67e-08 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> #pension.atet <- rlassoATET(X,d,y)
> pension.atet <- rlassoATET(form, data = pension)
> summary(pension.atet)
Estimation and significance testing of the treatment effect
Type: ATET
Bootstrap: not applicable
  coeff.  se. t-value p-value
TE 11810 2844  4.152 3.29e-05 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> pension.late <- rlassoLATE(X,d,y,z)
> #pension.late <- rlassoLATE(formZ, data=pension)
> summary(pension.late)
Estimation and significance testing of the treatment effect
Type: LATE
Bootstrap: not applicable
  coeff.  se. t-value p-value
TE 12189 2734  4.458 8.27e-06 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> pension.latet <- rlassoLATET(X,d,y,z)
> #pension.latet <- rlassoLATET(formZ, data=pension)
> summary(pension.latet)
Estimation and significance testing of the treatment effect
Type: LATET
Bootstrap: not applicable
  coeff.  se. t-value p-value
TE 12687 3590  3.534 0.00041 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The results are summarized in Table 1. We see that in this example the estimates are quite similar, while the standard errors differ.

**Table 1:** Estimation of treatment effects

	Estimate	Std. Error
ATE	10490.07	1919.99
ATET	11810.45	2844.33
LATE	12188.66	2734.12
LATET	12686.87	3590.09

## Application: estimation of the effect of price on demand

### Introduction

A core problems in Economics is to estimate the effect of price on the demand of products. This is usually impeded by the fact that price are not exogenously given, but determined by demand and supply in a simultaneous system of equations, introducing endogeneity. A possible way to solve this problem is to employ instrumental variables - a technique which is also getting more popular outside of Economics. The results found here are a replication in R of the results in Chernozhukov et al. (2015b) and Chernozhukov et al. (2015a).

here we are interested in estimating the coefficients in a simple logit model of demand for automobiles using market share data. Our example is based on the data and most basic strategy from Berry et al. (1995). The goal is to estimate the influence / effect of the price on demand (=market share). Specifically,

$$\begin{aligned}\log(s_{it}) - \log(s_{0t}) &= \alpha_0 p_{it} + x'_{it} \beta_0 + \varepsilon_{it}, \\ p_{it} &= z'_{it} \delta_0 + x'_{it} \gamma_0 + u_{it},\end{aligned}$$

where  $s_{it}$  is the market share of product  $i$  in market  $t$  with product zero denoting the outside option,  $p_{it}$  is the price and is treated as endogenous,  $x_{it}$  are observed included product characteristics, and  $z_{it}$  are instruments. In the data set the variable  $y$  is defined as  $\log(s_{it}) - \log(s_{0t})$ .

In our example, we use the same set of product characteristics ( $x$ -variables) as used in obtaining the basic results in Berry et al. (1995). Specifically, we use five variables in  $x_{it}$ : a constant, an air conditioning dummy, horsepower divided by weight, miles per dollar, and vehicle size. We refer to these five variables as the baseline set of controls.

We also adopt the argument from Berry et al. (1995) to form our potential instruments. BLP argue that that characteristics of other products will satisfy an exclusion restriction,  $E[\varepsilon_{it}|x_{j\tau}] = 0$  for any  $\tau$  and  $j \neq i$ , and thus that any function of characteristics of other products may be used as instrument for price. This condition leaves a very high-dimensional set of potential instruments as any combination of functions of  $\{x_{j\tau}\}_{j \neq i, \tau \geq 1}$  may be used to instrument for  $p_{it}$ . To reduce the dimensionality, BLP use intuition and an exchangeability argument to motivate consideration of a small number of these potential instruments formed by taking sums of product characteristics formed by summing over products excluding product  $i$ . Specifically, we form baseline instruments by taking

$$z_{k,it} = \left( \sum_{r \neq i, r \in \mathcal{I}_f} x_{k,rt}, \sum_{r \neq i, r \notin \mathcal{I}_f} x_{k,rt} \right)$$

where  $x_{k,it}$  is the  $k^{th}$  element of vector  $x_{it}$  and  $\mathcal{I}_f$  denotes the set of products produced by firm  $f$ . This choice yields a vector  $z_{it}$  consisting of 10 instruments. We refer to this set of instruments as the baseline instruments.

While the choice of the baseline instruments and controls is motivated by good intuition and economic theory, it should be noted that theory does not clearly state which product characteristics or instruments should be used in the model. Theory also fails to indicate the functional form with which any such variables should enter the model. High-dimensional methods outlined offer one strategy to help address these concerns which complements the economic intuition motivating the baseline controls and instruments. As an illustration, we consider an expanded set of controls and instruments. We augment the set of potential controls with all first order interactions of the baseline variables, quadratics and cubics in all continuous baseline variables, and a time trend which yields a total of 24  $x$ -variables. We refer to these as the augmented controls. We then take sums of these characteristics as potential instruments following the original strategy which yields 48 potential instruments.

### Estimation

The data set is included in the package an can be accessed in the following way

```
data(BLP)
BLPData <- BLP$BLP
```

A detailed description of the data is given in Berry et al. (1995) and also on the help page.

In the base line model we consider five  $x$ -variables and ten instrumental variables as described above. First we process the data, in particular we construct the design matrices for the  $x$ - and  $z$ -variables. The matrix of instruments  $Z$  is shipped with the data set but could also be constructed with

the internal function `constructIV` in the package **hdm**.

```
attach(BLPData)
X <- as.matrix(cbind(1, BLPData[,c("hpwt", "air", "mpd", "space")]))
Z <- BLP$Z
#Z <- hdm:::constructIV(firm.id, cdid, id, X)
```

With the baseline x- and z- variables, we estimate the price-effect with ols, tsls, and finally with selection on the x- and z-variables.

```
ols.reg <- lm(y~ price + hpwt + air + mpd + space, data =BLPData)
tsls.reg <- tsls(x=X, d=price, y=y, z=Z, intercept =FALSE, homoscedastic = FALSE)
lasso.reg <- rlassoIV(x=X[,-1], y=y, z=Z, d=price, select.X=TRUE, select.Z=TRUE,
intercept = TRUE)
```

The results are summarized in the table below:

**Table 2:** Results baseline model

Method	Price coefficient	Standard error
Baseline OLS	-0.089	0.004
Baseline TSLS	-0.136	0.012
Baseline TSLS with Lasso selection	-0.174	0.013

Next, we estimate the augmented model which in total has 24 controls and 48 instruments First, we construct the data needed to estimate the augmented model. The set of augmented IVs can be accessed by

```
augZ <- BLP$augZ

tu <- trend/19
mpdu <- mpd/7
spaceu <- space/2
augX = cbind(1, hpwt, air, mpdu, spaceu, tu, hpwt^2, hpwt^3, mpdu^2, mpdu^3,
spaceu^2, spaceu^3, tu^2, tu^3, hpwt*air, mpdu*air, spaceu*air, tu*air,
hpwt*mpdu, hpwt*spaceu, hpwt*tu, mpdu*spaceu, mpdu*tu, spaceu*tu)
colnames(augX) <- c("constant", "hpwt", "air", "mpdu", "spaceu", "tu", "hpwt^2", "hpwt^3",
"mpdu^2", "mpdu^3", "spaceu^2", "spaceu^3", "tu^2", "tu^3", "hpwt*air", "mpdu*air",
"spaceu*air", "tu*air", "hpwt*mpdu", "hpwt*spaceu", "hpwt*tu", "mpdu*spaceu", "mpdu*tu",
"spaceu*tu")
# augZ <- hdm:::constructIV(firm.id, cdid, id, augX) # construction of augmented set of IVs
```

Next, we redo the analysis with augmented set of variables:

```
ols.reg <- lm(y~ -1 + cbind(price, augX))
tsls.reg <- tsls(x=augX, d=price, y=y, z=augZ, intercept =FALSE, homoscedastic = FALSE)
lasso.reg <- rlassoIV(x=augX[,-1], y=y, z=augZ, d=price, select.X=TRUE, select.Z=TRUE,
intercept = TRUE)
```

The results for the augmented model are presented in the table:

**Table 3:** Results augmented model

Method	Price coefficient	Standard error
Augmented OLS	-0.099	0.004
Augmented TSLS	-0.127	0.008
Augmented TSLS with Lasso selection	-0.286	0.02

## Results

We report estimation results from the baseline and augmented setting in the tables above. The estimations all give a negative effect of price on demand as expected, but they differ considerably in size of the effects. In the augmented model we get the strongest influence of price on the market share and we see that the choice of instruments is an important issue.

## Summary

The package **hdm** contains methods for estimation and inference in a high-dimensional setting. We have presented a short overview of the functions implemented. The package also contains data sets which might be useful for classroom presentations and as applications for newly developed estimators. More examples and a short introduction in the underlying theoretical concepts are provided in the accompanying vignette. It is planned to extend the functionality and to improve the performance of the functions in subsequent versions of the package.

## Bibliography

- D. Acemoglu, S. Johnson, and J. A. Robinson. The colonial origins of comparative development: An empirical investigation. *American Economic Review*, 91(5):1369–1401, 2001. [p193]
- J. D. Angrist and J.-S. Pischke. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press, 2008. [p186]
- R. J. Barro and J.-W. Lee. Data set for a panel of 139 countries. *NBER*, <http://www.nber.org/pub/barro.lee.html>, 1994. [p193]
- A. Belloni and V. Chernozhukov. Least squares after model selection in high-dimensional sparse models. *Bernoulli*, 19(2):521–547, 2013. ArXiv, 2009. [p186, 187]
- A. Belloni, V. Chernozhukov, and C. Hansen. Inference for high-dimensional sparse econometric models. *Advances in Economics and Econometrics. 10th World Congress of Econometric Society. August 2010*, III:245–295, 2010. ArXiv, 2011. [p187]
- A. Belloni, D. Chen, V. Chernozhukov, and C. Hansen. Sparse models and methods for optimal instruments with an application to eminent domain. *Econometrica*, 80:2369–2429, 2012. Arxiv, 2010. [p186, 187, 193]
- A. Belloni, V. Chernozhukov, I. Fernández-Val, and C. Hansen. Program evaluation with high-dimensional data. *arXiv:1311.2645*, 2013. ArXiv, 2013. [p186]
- A. Belloni, V. Chernozhukov, and C. Hansen. Inference on treatment effects after selection amongst high-dimensional controls. *Review of Economic Studies*, 81:608–650, 2014a. ArXiv, 2011. [p186]
- A. Belloni, V. Chernozhukov, and K. Kato. Uniform post-selection inference for least absolute deviation regression and other z-estimation problems. *Biometrika*, 2014b. doi: 10.1093/biomet/asu056. [p185, 191]
- S. Berry, J. Levinsohn, and A. Pakes. Automobile prices in market equilibrium. *Econometrica*, 63: 841–890, 1995. [p193, 196]
- D. L. Chen and J. Sethi. Does forbidding sexual harassment exacerbate gender inequality. unpublished manuscript, 2010. [p193]
- V. Chernozhukov and C. Hansen. The impact of 401(k) participation on the wealth distribution: An instrumental quantile regression analysis. *Review of Economics and Statistics*, 86(3):735–751, 2004. [p193]
- V. Chernozhukov, D. Chetverikov, and K. Kato. Gaussian approximations and multiplier bootstrap for maxima of sums of high-dimensional random vectors. *Annals of Statistics*, 41:2786–2819, 2013. [p187]
- V. Chernozhukov, C. Hansen, and M. Spindler. Valid post-selection and post-regularization inference: An elementary, general approach. *Annual Review of Economics*, 7(1):649–688, 2015a. doi: 10.1146/annurev-economics-012315-015826. [p186, 196]
- V. Chernozhukov, C. Hansen, and M. Spindler. Valid post-selection and post-regularization inference in linear models with many controls and instruments. *American Economic Review: Papers and Proceedings*, 2015b. [p186, 196]
- J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. [p185]
- W. J. Fu. Penalized regressions: The bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 1998. doi: 10.1080/10618600.1998.10474784. [p188]

- T. Hastie and B. Efron. *lars: Least Angle Regression, Lasso and Forward Stagewise*, 2013. URL <https://CRAN.R-project.org/package=lars>. R package version 1.2. [p185]
- G. W. Imbens and D. B. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press, New York, NY, USA, 2015. ISBN 0521885884, 9780521885881. [p186]

*Victor Chernozhukov*  
Massachusetts Institute of Technology  
Economics Department and Center for Statistics  
50 Memorial Drive, Cambridge, MA 02142  
USA  
[vchern@mit.edu](mailto:vchern@mit.edu)

*Chris Hansen*  
University of Chicago  
Booth School of Business  
5807 South Woodlawn Avenue, Chicago, Illinois 60637  
USA  
[chansen1@chicagobooth.edu](mailto:chansen1@chicagobooth.edu)

*Martin Spindler*  
University of Hamburg and Max Planck Society  
Von-Melle-Park 5  
20146 Hamburg  
Germany  
[spindler@mea.mpisoc.mpg.de](mailto:spindler@mea.mpisoc.mpg.de)

# Normal Tolerance Interval Procedures in the tolerance Package

by Derek S. Young

**Abstract** Statistical tolerance intervals are used for a broad range of applications, such as quality control, engineering design tests, environmental monitoring, and bioequivalence testing. **tolerance** is the only R package devoted to procedures for tolerance intervals and regions. Perhaps the most commonly-employed functions of the package involve normal tolerance intervals. A number of new procedures for this setting have been included in recent versions of **tolerance**. In this paper, we discuss and illustrate the functions that implement these normal tolerance interval procedures, one of which is a new, novel type of operating characteristic curve.

## Introduction and overview of the tolerance package

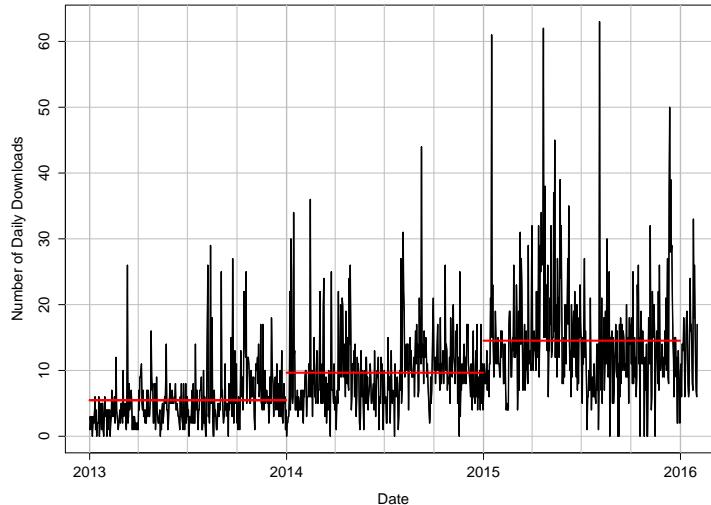
Statistical tolerance intervals of the form  $(1 - \alpha, P)$  provide bounds to capture at least a specified proportion  $P$  of the sampled population with a given confidence level  $1 - \alpha$ . The quantity  $P$  is called the *content* of the tolerance interval and the confidence level  $1 - \alpha$  reflects the sampling variability. There is an extensive literature on tolerance intervals with some of the earliest works being Wilks (1941, 1942) and Wald (1943). The texts by Guttman (1970) and Krishnamoorthy and Mathew (2009) are devoted to the theoretical development and application of tolerance intervals, while the text by Hahn and Meeker (1991) discusses their application in the broader context of statistical intervals.

**tolerance** (Young, 2010) is a popular R package for constructing exact and approximate tolerance intervals and regions. Since its initial release in 2009, the package has grown to include tolerance interval procedures for a large number of parametric distributions, nonparametric settings, and regression models. There are also tolerance region procedures for the multivariate normal and multivariate regression settings. Procedures for more specific settings are also included, such as one-sided tolerance limits for the difference between two independent normal random variables (Hall, 1984) and fiducial tolerance intervals for the function of parameters from discrete distributions (Mathew and Young, 2013). The package also includes some graphical capabilities for visualizing the tolerance intervals (regions) by plotting the limits (regions) on histograms, scatterplots, or control charts of the data.

**tolerance** has been used for a broad range of applications, including cancer research (Heck et al., 2014), wildlife biology (Pasquaretta et al., 2012), assessing the performance of genetic algorithms (Van der Borght et al., 2014), ratio editing in surveys (Young and Mathew, 2015), air quality assessment (Hafner et al., 2013), and instrumentation testing (Burr and Gavron, 2010). General interest in **tolerance** can be gauged by the **cranlogs** package (Csardi, 2015), which pulls download logs of the RStudio (RStudio Team, 2015) CRAN mirror. Figure 1 shows the daily number of downloads of **tolerance** from the beginning of 2013 to the beginning of 2016. There is clearly a general increasing trend over the years as the average number of daily downloads per year is approximately 5, 10, and 15 in 2013, 2014, and 2015, respectively.

Capabilities of **tolerance** have been discussed in Young (2010, 2014). Even with those varied capabilities, perhaps the most commonly used methods involve the normal distribution. Normal tolerance intervals are often required during design verification or process validation. The utility of normal tolerance intervals is further highlighted in documents published by the EPA (Environmental Protection Agency, 2006), the IAEA (International Atomic Energy Agency, 2008), and standard 16269-6 of the ISO (International Organization for Standardization, 2014). In this paper, we discuss new capabilities in **tolerance** specifically involving normal tolerance intervals. This includes the calculation of exact and equal-tailed normal tolerance intervals, Bayesian normal tolerance intervals, tolerance intervals for fixed-effects ANOVA, and sample size determination strategies. We also introduce novel pseudo-operating characteristic (OC) curves that illustrate how the  $k$ -factor, sample size, confidence level, and content level each change relative to one another. Such curves can be useful for planning design tests.

As noted earlier, **tolerance** also includes a function for constructing multivariate normal tolerance regions. The **mvtol.region()** function was included with the initial release of **tolerance**. **mvtol.region()** includes several Monte Carlo procedures developed in Krishnamoorthy and Mathew (1999) and Krishnamoorthy and Mondal (2006) for finding the  $k$ -factor of the multivariate normal tolerance region. The **plottol()** function can also be used to plot tolerance ellipses over bivariate normal data and tolerance ellipsoids over trivariate normal data. The latter is accomplished using **plot3d()** from the **rgl** package (Adler and Murdoch, 2014). We will not discuss the **mvtol.region()**



**Figure 1:** Number of daily downloads for **tolerance** from the RStudio CRAN mirror over a three-year time span (2013–2016).

function further since it is already well-documented (Young, 2010, 2014) and our present focus is on newer capabilities in **tolerance** for normal tolerance intervals.

For our discussion, we assume that the reader has already installed and loaded **tolerance** using the usual commands:

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

## Normal tolerance intervals - classical and Bayesian

Let  $\mathbf{X} = (X_1, X_2, \dots, X_n)$  be a random sample of continuous random variables that have cumulative distribution function  $F_X$ , which is parameterized by  $\theta \in \Theta \subset \mathbb{R}^d$ . Let  $X \sim F_X$ , independently of  $\mathbf{X}$ . In the classical set-up, a  $(1 - \alpha, P)$  one-sided *upper tolerance limit* ( $U_1(\mathbf{X})$ ) and one-sided *lower tolerance limit* ( $L_1(\mathbf{X})$ ) satisfy the expressions

$$P_{\mathbf{X}}(P_X[X \leq U_1(\mathbf{X}) | \mathbf{X}] \geq P) = 1 - \alpha \quad (1)$$

and

$$P_{\mathbf{X}}(P_X[L_1(\mathbf{X}) \leq X | \mathbf{X}] \geq P) = 1 - \alpha, \quad (2)$$

respectively. Similarly, a  $(1 - \alpha, P)$  two-sided *tolerance interval*,  $(L_2(\mathbf{X}), U_2(\mathbf{X}))$ , satisfies

$$P_{\mathbf{X}}(P_X[L_2(\mathbf{X}) \leq X \leq U_2(\mathbf{X}) | \mathbf{X}] \geq P) = 1 - \alpha. \quad (3)$$

Sometimes, controlling the proportion in the tails is required, in which case we have a  $(1 - \alpha, P)$  *equal-tailed tolerance interval*,  $(L_e(\mathbf{X}), U_e(\mathbf{X}))$ , that satisfies

$$P_{\mathbf{X}}(\{P_X[L_e(\mathbf{X}) \leq X | \mathbf{X}] \leq (1 - P)/2\} \cap \{P_X[U_e(\mathbf{X}) \geq X | \mathbf{X}] \leq (1 - P)/2\}) = 1 - \alpha. \quad (4)$$

Equal-tailed tolerance intervals ensure that we simultaneously have no more than  $(1 - P)/2$  of the sampled population below the lower tolerance limit and no more than  $(1 - P)/2$  of the sampled population above the upper tolerance limit.

Let  $X_1, \dots, X_n$  be *iid*  $\mathcal{N}(\mu, \sigma^2)$ ; i.e. a normal distribution with unknown mean  $\mu$  and unknown variance  $\sigma^2$ . Let  $\bar{X}$  and  $S^2$  denote the sample mean and sample variance, respectively. The formulas for  $(1 - \alpha, P)$  lower and upper normal tolerance limits are

$$L_h(\mathbf{X}) = \bar{X} - k_h(n, \alpha, P)S \quad \text{and} \quad U_h(\mathbf{X}) = \bar{X} + k_h(n, \alpha, P)S, \quad (5)$$

respectively, where  $h \in \{1, 2, e\}$ . In other words,  $h$  is an index specifying whether we want one-sided tolerance limits, two-sided tolerance intervals, or equal-tailed tolerance intervals.  $k_1(n, \alpha, P)$  and

$k_2(n, \alpha, P)$  are the  $k$ -factors for these two settings. The  $k$ -factor ensures that we capture at least a proportion  $P$  of the sampled population with confidence level  $(1 - \alpha)$ .  $k_1(n, \alpha, P)$  is calculated by

$$k_1(n, \alpha, P) = \frac{1}{\sqrt{n}} t_{n-1; 1-\alpha}(\sqrt{n}z_P), \quad (6)$$

where  $t_{f,q}(\delta)$  is the  $q^{\text{th}}$  quantile of a noncentral  $t$ -distribution with  $f$  degrees of freedom and noncentrality parameter  $\delta$  and  $z_q$  is the  $q^{\text{th}}$  quantile of the standard normal distribution.  $k_2(n, \alpha, P)$  is the solution to the integral equation

$$\sqrt{\frac{2n}{\pi}} \int_0^\infty P \left( \chi_{n-1}^2 > \frac{(n-1)\lambda_{1;P}^2(z^2)}{k_2(n, \alpha, P)^2} \right) e^{-\frac{1}{2}nz^2} dz = 1 - \alpha, \quad (7)$$

where  $\chi_f^2$  is the chi-square random variable with  $f$  degrees of freedom and  $\chi_{f,q}^2(\delta)$  is the  $q^{\text{th}}$  quantile of the noncentral chi-squared distribution with  $f$  degrees of freedom and noncentrality parameter  $\delta$ .

Owen (1964) was the first to propose equal-tailed tolerance intervals for the normal distribution. Equal-tailed normal tolerance intervals still take the form of (5), but where the tolerance factor  $k_e(n, \alpha, P)$  is found as the solution to the integral equation

$$\left( 2^{\frac{n-1}{2}} \Gamma \left( \frac{n-1}{2} \right) \right)^{-1} \int_{\frac{(n-1)\theta_n^2}{nk_e(n, \alpha, P)^2}}^\infty \left( 2\Phi \left( -\theta_n + \frac{k_e(n, \alpha, P)\sqrt{nz}}{\sqrt{n-1}} \right) - 1 \right) e^{-z/2} z^{\frac{n-1}{2}-1} dz = 1 - \alpha, \quad (8)$$

where  $\theta_n = \sqrt{nz}_{\frac{1+\alpha}{2}}$  and  $\Phi(\cdot)$  denotes the standard normal cumulative distribution function. A general discussion comparing the utility of two-sided tolerance intervals versus equal-tailed tolerance intervals is found in Jensen (2009).

The `normtol.int()` function in **tolerance** is able to calculate all of the one-sided tolerance limits, two-sided tolerance intervals, and equal-tailed tolerance intervals discussed above. In the past, challenges with computing noncentral distributions necessitated the use of approximations for  $k_1(n, \alpha, P)$ ,  $k_2(n, \alpha, P)$ , and  $k_e(n, \alpha, P)$ . For the two-sided tolerance intervals, early versions of **tolerance** simply used various approximations that appeared in the literature over the years for computing the  $k$ -factors. These are controlled through the `method` argument and their specific formulas are outlined in Young (2010), which utilized **tolerance** version 0.2.2. Since then, the exact  $k$ -factor in (7) and the exact equal-tailed  $k$ -factor in (8) have been included. These are implemented by setting `method = "EXACT"` and `method = "OCT"`, respectively. Both of these methods use adaptive quadrature via the `integrate()` function as well as box-constrained optimization via the `optim()` function. The original approximation methods are still available primarily to retain all functionality of previous versions of **tolerance**.

The dataset that we will use to illustrate most of the procedures in our discussion is a quality control dataset from Krishnamoorthy and Mathew (2009). The data are from a machine that fills plastic containers with a liter of milk. At the end of a particular shift, a sample of  $n = 20$  containers was selected and the actual amount of milk in each container was measured using a highly-accurate method. These measurements are as follows:

```
> milk <- c(0.968, 0.982, 1.030, 1.003, 1.046,
+         1.020, 0.997, 1.010, 1.027, 1.010,
+         0.973, 1.000, 1.044, 0.995, 1.020,
+         0.993, 0.984, 0.981, 0.997, 0.992)
```

A quick check of normality with the Shapiro-Wilk test confirms that this is an appropriate assumption:

```
> shapiro.test(milk)
```

Shapiro-Wilk normality test

```
data: milk
W = 0.96364, p-value = 0.6188
```

For the milk data, the  $(0.95, 0.90)$  one-sided tolerance limits, two-sided tolerance interval, and equal-tailed tolerance interval are found as follows:

```
> normtol.int(x = milk, alpha = 0.05, P = 0.90, side = 1)
  alpha   P  x.bar 1-sided.lower 1-sided.upper
1  0.05 0.9 1.0036      0.9610333     1.046167

> normtol.int(x = milk, alpha = 0.05, P = 0.90, side = 2, method = "EXACT", m = 50)
  alpha   P  x.bar 2-sided.lower 2-sided.upper
```

```

1 0.05 0.9 1.0036      0.9523519      1.054848

> normtol.int(x = milk, alpha = 0.05, P = 0.90, side = 2, method = "OCT", m = 50)
  alpha   P x.bar 2-sided.lower 2-sided.upper
1 0.05 0.9 1.0036      0.9471414      1.060059

```

Note that the equal-tailed tolerance interval is wider than the corresponding two-sided tolerance interval due to the more stringent requirement of controlling the proportions in the tails. For the two-sided tolerance intervals, the additional argument *m* is used to control the number of subintervals to use for performing the numerical integration via *integrate()*. While not illustrated above, there is an additional argument that can be used if one wishes to construct log-normal tolerance intervals. The argument *log.norm* is a logical argument set to FALSE by default. If set to TRUE, log-normal tolerance intervals are calculated using the fact that if  $X$  is log-normally distributed, then  $Y = \log(X)$  is normally distributed. Thus, the *normtol.int()* function simply takes the logarithm of the data in the *x* argument, constructs the desired normal tolerance limits, and then takes the anti-log of those limits.

Users of normal tolerance intervals are often interested in summarizing a variety of possible  $k$ -factors for given sample sizes  $n$ , confidence levels  $1 - \alpha$ , and content level  $P$ . The *K.table* function allows the user to specify a vector of possible values for each of these three quantities. A list is then returned whose elements are summarized according to the *by* argument. For example, suppose we are interested in the  $k_1(n, \alpha, P)$  values for all combinations of  $n \in \{10, 20\}$ ,  $\alpha \in \{0.01, 0.05\}$ , and  $P \in \{0.95, 0.99\}$ . Moreover, we would like to summarize the list by the levels of  $P$ . This is accomplished as follows:

```

> K.table(n = c(10, 20), alpha = c(0.01, 0.05), P = c(0.95, 0.99),
+           side = 1, by.arg = "P")
$`0.95`
    10      20
0.99 3.738315 2.807866
0.95 2.910963 2.396002

$`0.99`
    10      20
0.99 5.073725 3.831558
0.95 3.981118 3.295157

```

For example, the first entry of the first matrix is the  $k$ -factor for a one-sided (0.99, 0.95) tolerance limit when  $n = 10$ . One can also set *side* = 2, which requires the user to specify the *method* argument; e.g. "EXACT" for values of  $k_2(n, \alpha, P)$  or "OCT" for values of  $k_e(n, \alpha, P)$ . The *by.arg* argument can also be set to "alpha" or "n" depending on which quantity you want to represent the elements of the outputted list.

Bayesian tolerance intervals were first presented in [Aitchison \(1964\)](#). For the Bayesian set-up, let  $\mathbf{x}$  be a vector of realizations of  $X$ ,  $\mathcal{L}(\boldsymbol{\theta}|\mathbf{x})$  be the likelihood function,  $\pi(\boldsymbol{\theta})$  be a prior distribution for  $\boldsymbol{\theta}$ , and  $p(\boldsymbol{\theta}|\mathbf{x})$  be the posterior distribution of  $\boldsymbol{\theta}$  given by

$$p(\boldsymbol{\theta}|\mathbf{x}) = \frac{\mathcal{L}(\boldsymbol{\theta}|\mathbf{x})\pi(\boldsymbol{\theta})}{\int_{\Theta} \mathcal{L}(\boldsymbol{\theta}|\mathbf{x})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}}. \quad (9)$$

Then,  $(1 - \alpha, P)$  Bayesian one-sided upper and lower tolerance limits satisfy

$$P_{\Theta}(P_X[X \leq U_1(\boldsymbol{\theta})|\boldsymbol{\theta}] \geq P|X) = 1 - \alpha \quad (10)$$

and

$$P_{\Theta}(P_X[L_1(\boldsymbol{\theta}) \leq X|\boldsymbol{\theta}] \geq P|X) = 1 - \alpha, \quad (11)$$

respectively, a  $(1 - \alpha, P)$  Bayesian two-sided tolerance interval satisfies

$$P_{\Theta}(P_X[L_2(\boldsymbol{\theta}) \leq X \leq U_2(\boldsymbol{\theta})|\boldsymbol{\theta}] \geq P|X) = 1 - \alpha, \quad (12)$$

and a  $(1 - \alpha, P)$  Bayesian equal-tailed tolerance interval satisfies

$$P_{\Theta}(\{P_X[L_e(\boldsymbol{\theta}) \leq X|\boldsymbol{\theta}] \leq (1 - P)/2\} \cap \{P_X[U_e(\boldsymbol{\theta}) \geq X|\boldsymbol{\theta}] \leq (1 - P)/2\}|X) = 1 - \alpha. \quad (13)$$

Notice that the Bayesian set-up is calculated with respect to the probability measure  $P_{\Theta}$  while the classical set-up is calculated with respect to the distribution of the random sample  $\mathbf{X}$ . We refer the reader to the texts by [Guttman \(1970\)](#) and [Krishnamoorthy and Mathew \(2009\)](#) for more details on

both classical and Bayesian tolerance intervals.

The parameters  $\mu$  and  $\sigma^2$  are still assumed unknown. We use the conjugate priors  $\pi(\mu|\sigma^2)$  and  $\pi(\sigma^2)$ , which are

$$\mu|\sigma^2 \sim \mathcal{N}(\mu_0, \sigma^2/n_0) \text{ and } \sigma^2 \sim \text{Scale-inv-}\chi^2(m_0, \sigma_0^2), \quad (14)$$

respectively, where Scale-inv- $\chi^2(\nu, \tau^2)$  is the scaled inverse chi-squared distribution with  $\nu$  degrees of freedom and scaling parameter  $\tau^2$ . Thus, the joint prior density of  $(\mu, \sigma^2)$  is  $\pi(\mu, \sigma^2) = \pi(\mu|\sigma^2)\pi(\sigma^2)$ . The four hyperparameters for this prior structure are  $\mu_0 \in \mathbb{R}$ ,  $\sigma_0^2 > 0$ ,  $m_0 > 0$ ,  $n_0 > 0$ .  $m_0$  and  $n_0$  are not prior sample size quantities, but are tunable quantities to reflect the prior precision relative to the sample size. The joint posterior distribution is then

$$p(\mu, \sigma^2 | \mathbf{x}) = p(\mu|\sigma^2)p(\sigma^2), \quad (15)$$

where  $p(\mu|\sigma^2)$  and  $p(\sigma^2)$  are the distributions

$$\mu|\sigma^2 \sim \mathcal{N}\left(\bar{x}, \frac{\sigma^2}{n_0 + n}\right) \text{ and } \sigma^2 \sim \text{Scale-inv-}\chi^2(m_0 + n - 1, q^2), \quad (16)$$

respectively, such that

$$\bar{x} = \frac{n_0\mu_0 + n\bar{x}}{n_0 + n} \text{ and } q^2 = (m_0 + n - 1)^{-1} \left[ m_0\sigma_0^2 + (n - 1)s^2 + \frac{n_0n}{n_0 + n}(\bar{x} - \mu_0)^2 \right]. \quad (17)$$

Note that the formulas in the Bayesian set-up are written such that they are conditioned on realizations of the observed data; i.e.  $\mathbf{X} = \mathbf{x}$ . Furthermore, they are written in terms of the sample estimates of the mean ( $\bar{x}$ ) and variance ( $s^2$ ). Additional details on the above can be found, for example, in Chapter 3 of [Gelman et al. \(2013\)](#).

Similar to the classical setting,  $(1 - \alpha, P)$  Bayesian lower and upper normal tolerance limits are, respectively,

$$L_h(\bar{x}, s^2) = \bar{x} - k_h(n, n_0, m_0, \alpha, P)q \text{ and } U_h(\bar{x}, s^2) = \bar{x} + k_h(n, n_0, m_0, \alpha, P)q, \quad (18)$$

where, again,  $h$  is used as an index for one-sided limits, a two-sided interval, or an equal-tailed interval. Note that these limits are expressed in terms the maximum likelihood estimates of  $\mu$  and  $\sigma^2$ , which occur through how  $\bar{x}$  and  $q$  are defined. Thus, the one-sided  $k$ -factor is calculated by

$$k_1(n, n_0, m_0, \alpha, P) = \frac{1}{\sqrt{n_0 + n}} t_{m_0+n-1; 1-\alpha}(\sqrt{n_0 + n}z_p), \quad (19)$$

the two-sided  $k$ -factor  $k_2(n, n_0, m_0, \alpha, P)$  is calculated by finding the solution to

$$\sqrt{\frac{2(n_0 + n)}{\pi}} \int_0^\infty P\left(\chi_{m_0+n-1}^2 > \frac{(m_0 + n - 1)\chi_{1,P}^2(z^2)}{k_2(n, n_0, m_0, \alpha, P)^2}\right) e^{-\frac{1}{2}(n_0+n)z^2} dz = 1 - \alpha, \quad (20)$$

and the equal-tailed  $k$ -factor  $k_e(n, n_0, m_0, \alpha, P)$  is calculated by finding the solution to

$$\begin{aligned} \frac{2^{-\left(\frac{m_0+n-1}{2}\right)}}{\Gamma\left(\frac{m_0+n-1}{2}\right)} \int_{\frac{(m_0+n-1)\sigma_0^2}{(n_0+n)k_e(n, n_0, m_0, \alpha, P)^2}}^\infty & \left(2\Phi\left(-\vartheta_{n_0+n} + \frac{k_e(n, n_0, m_0, \alpha, P)\sqrt{n_0 + n}z}{\sqrt{m_0 + n - 1}}\right) - 1\right) \\ & \times e^{-z/2} z^{\frac{m_0+n-1}{2}-1} dz = 1 - \alpha. \end{aligned} \quad (21)$$

Finally, if one considers the non-informative prior distribution

$$\pi(\mu, \sigma^2) \propto \sigma^{-2}, \quad (22)$$

the solutions for the one-sided Bayesian normal tolerance limits and two-sided Bayesian normal tolerance intervals are the same as for the classical setting given in Equations (5)–(8); see Chapter 11 of [Krishnamoorthy and Mathew \(2009\)](#) for the details.

The `bayesnormtol.int()` function for computing Bayesian normal tolerance intervals is new as of **tolerance** version 1.1.1. It was composed to closely mirror the `normtol.int()` function. For the milk data, suppose we use the conjugate prior structure in (14). Assuming we have some historical knowledge about the milk filling process, the following hyperparameter values are used:  $\mu_0 = 1.000$ ,  $\sigma^2 = 0.001$ , and  $m_0 = n_0 = 20$ . Then, the Bayesian  $(0.95, 0.90)$  one-sided tolerance limits, two-sided

tolerance interval, and equal-tailed tolerance interval are found as follows:

```
> bayesnormtol.int(x = milk, alpha = 0.05, P = 0.90, side = 1,
+                     hyper.par = list(mu.0 = 1.000, sig2.0 = 0.001,
+                     m.0 = 20, n.0 = 20))
alpha   P 1-sided.lower 1-sided.upper
1  0.05 0.9      0.9551936      1.048406

> bayesnormtol.int(x = milk, alpha = 0.05, P = 0.90, side = 2, method = "EXACT",
+                     m = 50, hyper.par = list(mu.0 = 1.000, sig2.0 = 0.001,
+                     m.0 = 20, n.0 = 20))
alpha   P 2-sided.lower 2-sided.upper
1  0.05 0.9      0.9453603      1.05824

> bayesnormtol.int(x = milk, alpha = 0.05, P = 0.90, side = 2, method = "OCT",
+                     m = 50, hyper.par = list(mu.0 = 1.000, sig2.0 = 0.001,
+                     m.0 = 20, n.0 = 20))
alpha   P 2-sided.lower 2-sided.upper
1  0.05 0.9      0.9407625      1.062838
```

The `bayesnormtol.int()` function has the arguments `x`, `alpha`, `P`, `side`, `method`, and `m` just as in the `normtol.int()`. However, here we also have `hyper.par`, which is a list with elements for the four hyperparameters. The output is structured identically to the output obtained using `normtol.int()`, which allows for easy comparison between the classical and Bayesian results.

## Fixed-effects ANOVA tolerance intervals

The approach for classical normal tolerance intervals can be easily extended for the balanced fixed-effects ANOVA model

$$Y_{ij...kl} = \theta + \alpha_i + \beta_j + \dots + \gamma_k + \epsilon_{ij...kl}, \quad (23)$$

where  $\theta$  is the grand mean,  $\alpha_i, \beta_j, \dots, \gamma_k$  are the factor effects each subject to the constraint that the summation of the effects over the respective index is equal to 0,  $\epsilon_{ij...kl}$  are *iid*  $\mathcal{N}(0, \sigma^2)$  error terms, and the indices are  $i = 1, \dots, a$ ,  $j = 1, \dots, b, \dots, k = 1, \dots, c$ , and  $l = 1, \dots, n$ . The approach discussed below is for the classical setting. Currently, the **tolerance** package does not have a function for calculating Bayesian ANOVA tolerance intervals.

Let  $\mathbf{Y} \in \mathbb{R}^{ab \cdots cn}$  be a vector of all of the measured responses in (23), which are *iid*  $\mathcal{N}(0, \sigma^2)$ . The formulas for the tolerance limits in the fixed-effects ANOVA setting are:

$$\begin{aligned} L_{i;h}(\mathbf{Y}) &= \bar{Y}_{i,\dots} - k_h(n_i, f, \alpha, P) \sqrt{MSE} & \text{and} & U_{i;h}(\mathbf{Y}) = \bar{Y}_{i,\dots} + k_h(n_i, f, \alpha, P) \sqrt{MSE} \\ L_{j;h}(\mathbf{Y}) &= \bar{Y}_{\cdot j,\dots} - k_h(n_j, f, \alpha, P) \sqrt{MSE} & \text{and} & U_{j;h}(\mathbf{Y}) = \bar{Y}_{\cdot j,\dots} + k_h(n_j, f, \alpha, P) \sqrt{MSE} \\ &\vdots && \vdots \\ L_{k;h}(\mathbf{Y}) &= \bar{Y}_{\dots,k} - k_h(n_k, f, \alpha, P) \sqrt{MSE} & \text{and} & U_{k;h}(\mathbf{Y}) = \bar{Y}_{\dots,k} + k_h(n_k, f, \alpha, P) \sqrt{MSE} \end{aligned} \quad (24)$$

Conceptually, the formulas in (24) are similar to those in (5). We take a point estimate of the mean at each factor level (i.e. the quantities  $\bar{Y}_{i,\dots}, \bar{Y}_{\cdot j,\dots}, \dots, \bar{Y}_{\dots,k}$ ) and then add or subtract the  $k$ -factor times the standard deviation. The standard deviation is now estimated by the root mean square error,  $\sqrt{MSE}$ .

The  $k$ -factor in (24), again, has the subscript  $h$  to indicate an index for one-sided limits, a two-sided interval, or an equal-tailed interval. However, the formulas are modified for the ANOVA setting. In formulas (6)–(8), the quantity  $(n - 1)$  reflects the degrees of freedom when estimating the sample variance  $S^2$ . In the ANOVA setting, this is replaced by the degrees of freedom due to the error; i.e. the degrees of freedom associated with the  $MSE$ . Thus, we replace each occurrence of  $(n - 1)$  in (6)–(8) with  $f$ , the error degrees of freedom. Moreover, all occurrences of the sample size  $n$  are replaced with the number of observations at each factor level; i.e.  $n_i, n_j, \dots, n_k$ . Note that the tolerance intervals presented are only accurate for balanced (or nearly-balanced) ANOVA settings.

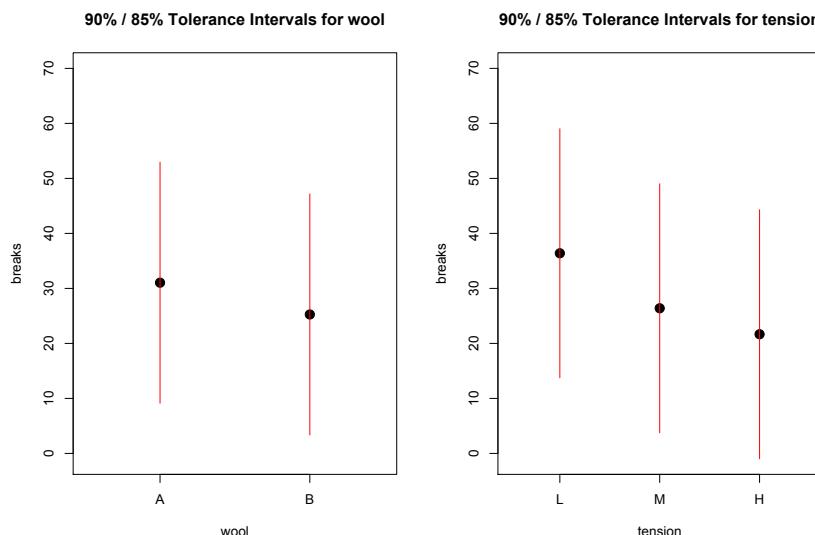
We analyze the well-known dataset that resulted from an experimental design regarding the effects of wool type and the amount of tension applied to a loom of yarn on the number of warp breaks that occur on that loom of yarn (Tippett, 1950). The first factor is wool type, which has two levels: A or B.

The second factor is tension level, which has three levels: low (L), medium (M), or high (H). The six treatments (i.e. factor level combinations) are randomly assigned to one of 54 looms of yarn. Thus, we have  $n = 9$  replicates per treatment. Suppose we want to construct  $(0.85, 0.90)$  equal-tailed tolerance intervals for each factor level's mean. Below is how we would do this in the **tolerance** package:

```
> lm.out <- lm(breaks ~ wool + tension, data = warpbreaks)
> out <- anovatol.int(lm.out, data = warpbreaks, alpha = 0.10,
+                      P = 0.85, side = 2, method = "OCT")
These are 90%/85% 2-sided tolerance intervals.
> out
$wool
  mean n      k 2-sided.lower 2-sided.upper
A 31.03704 27 1.886857     9.117165    52.95691
B 25.25926 27 1.886857    3.339387    47.17913

$tension
  mean n      k 2-sided.lower 2-sided.upper
L 36.38889 18 1.948567   13.7521219   59.02566
M 26.38889 18 1.948567   3.7521219   49.02566
H 21.66667 18 1.948567   -0.9701003   44.30343
```

In the `anovatol.int()` function, we have similar arguments as in the `normtol.int()` function, except now we input an object of class "`lm`" and we also tell the function the name of the original dataset using the `data` argument. The output is a list summarizing the tolerance interval results for each factor level. For example, the  $(0.90, 0.85)$  equal-tailed tolerance interval for the medium tension applied to the yarn is about  $(3.75, 49.03)$ .



**Figure 2:** Plot of the  $(0.85, 0.90)$  equal-tailed tolerance intervals for the yarn strength data.

We can also produce a figure of the above output by using the `plottol()` function as follows:

```
> plottol(out, x = warpbreaks)
```

The above produces the plot in Figure 2. This figure has a separate panel for each factor. The `y`-axis is the response and the `x`-axis is the levels of the respective factor. The solid black point is the factor level mean and the red lines extend to the lower and upper tolerance limits calculated earlier. Such a figure provides a relative comparison of the tolerance intervals for each factor level.

## Sample size determination strategies

As noted in Faulkenberry and Weeks (1968), an important question for statistical practitioners is "What sample size should be used to determine the tolerance limits?" Those same authors addressed this problem by developing an approach to ensure that the calculated tolerance intervals are "close" to the quantiles that result in a content level at least as large as  $P$ . Their solution was developed for

sample size determination of one-sided tolerance limits and two-sided tolerance intervals, but it is not applicable to equal-tailed tolerance intervals. In order to briefly present their approach, let  $C_{\mu,\sigma}(\mathbf{X})$  denote any of the three inner probabilities in Equations (1)–(3) or any of the three analogous inner probabilities for the Bayesian set-up in Equations (10)–(12). To ensure the "goodness" of the tolerance limits (interval), one must choose an arbitrary  $P' > P$  and small  $\delta > 0$  to determine a sample size  $n^*$  such that

$$P_{\mathbf{X}} \{ C_{\mu,\sigma}(\mathbf{X}) \geq P' \} \geq \delta \quad (25)$$

or

$$P_{\theta} \{ C_{\mu,\sigma}(\mathbf{X}) \geq P' \} \geq \delta \quad (26)$$

for the classical or Bayesian setting, respectively.

The `norm.ss` function for sample size determination of normal tolerance limits (intervals) is new as of **tolerance** version 1.1.1. The function finds the minimum sample size  $n^*$  for the approach due to Faulkenberry and Weeks (1968) discussed above. For our example, suppose the quality engineer overseeing the milk-filling process wants to submit a future sample of liters of milk to the highly-accurate measurement method. Per the company's guidelines, the engineer needs to know the minimum sample size to construct a (0.95, 0.90) two-sided tolerance interval such that  $P_{\mathbf{X}} \{ C_{\mu,\sigma}(\mathbf{X}) \geq 0.97 \} \geq 0.10$ . This is calculated as follows:

```
> norm.ss(alpha = 0.05, P = 0.90, delta = 0.10, P.prime = 0.97,
+           side = 2, m = 50, method = "FW")
alpha   P delta P.prime   n
1  0.05 0.9    0.1    0.97 60
```

Thus, the engineer would need a minimum sample size of  $n^* = 60$  to ensure that there is only a small probability  $\delta = 0.10$  that the (0.95, 0.90) tolerance interval will have a content exceeding  $P = 0.90$  by  $(P' - P) = 0.07$ .

In the `norm.ss()` function, the argument `method` is set to "FW". There are two additional sample size determination strategies that can be calculated, which are controlled through the `method` argument. Both of these strategies assume there is some historical data and specification limits for the process at-hand. We briefly illustrate these strategies below and refer the reader to Young et al. (2016) for further details.

The first alternative strategy is a simple "back-of-the-envelope" calculation. We consider the problem of designing a study to demonstrate that a process or product falls within the specification limits  $(S_L, S_U)$ . We are interested in the minimum sample size necessary such that a  $(1 - \alpha, P)$  one-sided lower tolerance limit exceeds  $S_L$ , a  $(1 - \alpha, P)$  one-sided upper tolerance limit falls below  $S_U$ , or a  $(1 - \alpha, P)$  two-sided tolerance interval is contained within  $(S_L, S_U)$ . In other words, this requires finding the minimum sample size  $n^*$  such that

$$S_L < \mu - k_1(n, \alpha, P)\sigma; \quad (27)$$

$$S_U > \mu + k_1(n, \alpha, P)\sigma; \text{ or} \quad (28)$$

$$\mu \pm k_e(n, \alpha, P)\sigma \subset (S_L, S_U), \quad (29)$$

for one-sided upper tolerance limits, one-sided lower tolerance limits, or equal-tailed tolerance intervals, respectively. As emphasized in Young et al. (2016), this approach is intended simply for planning purposes and it does not guarantee any specific bounds relative to the nominal coverage probability. Note that (29) is for an equal-tailed tolerance interval since we posit values for  $\mu$  and  $\sigma$  and, thus, the resulting tolerance interval would be built around a (hypothetically) true center of the normal population.

For our example, suppose that the quality engineer is overseeing the launch of a new process for filling the one-liter containers of milk, which is intended to be more accurate than the previous process. The company set specification limits at (0.990, 1.010). For determining the minimum sample size necessary to construct a (0.95, 0.90) two-sided tolerance interval that is within the specification limits, the engineer assumes the mean and variance from the data of the original process. This calculation can then be done as follows:

```
> norm.ss(alpha = 0.05, P = 0.90, side = 2, spec = c(0.990, 1.010),
+           method = "DIR", hyper.par = list(mu.0 = 1.004, sig2.0 = 0.001))
alpha   P delta P.prime   n
1  0.05 0.9    5
```

Thus, the minimum sample size is  $n^* = 5$ . This calculation was done by setting `method = "DIR"`, entering the specification limits in the `spec` argument, and entering the assumed  $\mu$  and  $\sigma^2$  in the

argument `hyper.par`.

The second alternative strategy presented in Young et al. (2016) is a method for providing data-dependent values of the precision quantities  $P'$  and  $\delta$  in the approach due to Faulkenberry and Weeks (1968). The approach assumes there is information on historical data, a set of current data, and specification limits that can be used for calculating values of  $P'$  and  $\delta$ . The approach is intended to be used when there is no practical guidance for setting these values other than using “rule-of-thumb” quantities suggested in Faulkenberry and Weeks (1968).

For the milk filling process, suppose the engineer has historical measurements, which have a combined mean 0.994 liters and variance 0.002. Suppose the specification limits on the original process are (0.900, 1.100) and that the engineer needs a minimum sample size to construct a (0.95, 0.90) two-sided tolerance interval to show that the process meets the specification limits. However, the engineer is unsure about levels to choose for  $\delta$  and  $P'$ . We can use the `norm.ss()` function as follows:

```
> norm.ss(x = milk, alpha = 0.05, P = 0.90, side = 2, spec = c(0.900, 1.100),
+           method = "YGZO", hyper.par = list(mu.0 = 0.994, sig2.0 = 0.002))
   alpha      P      delta      P.prime      n
1  0.05 0.9 0.1807489 0.9733307 42
```

Thus, the engineer would need a minimum sample size of  $n^* = 42$  to ensure that there is only a probability of about  $\delta = 0.181$  that the (0.95, 0.90) tolerance interval will have a content exceeding  $P = 0.90$  by about  $(P' - P) = 0.073$ .

## OC curves involving k-factors

Sometimes, engineers and industrial statisticians are interested in understanding how the confidence level or content level changes as a function of  $n$  for a given level of the  $k$ -factor. If one has normally distributed data that they intend to demonstrate meets certain specification limits, then it is important to understand the type of values for  $1 - \alpha$  and  $P$  that one can reasonably expect to use. In this section, we present OC curves for such planning purposes.

The first type of OC curve is used when one specifies a range of values of the sample size  $n$  and a target value of the  $k$ -factor. Then, one can either specify a set of  $1 - \alpha$  values and solve for  $P$  or one can specify a set of  $P$  values and solve for  $1 - \alpha$ . The values of  $n$  are plotted on the  $x$ -axis and the value being solved for – either  $P$  or  $1 - \alpha$  – is plotted on the  $y$ -axis. The different OC curves pertain to the set of specified values – either  $1 - \alpha$  or  $P$ . Since too many curves can become cumbersome, we have placed an upper limit of 10 curves that can be overlaid on a given plot. Also, the colors used for the curves were chosen using a colorblind-friendly palette that was established by Okabe and Ito (2002).

Suppose a company is designing a product and the engineer needs to collect enough data so that the resulting two-sided tolerance interval will have a  $k$ -factor of 4. Content levels under consideration are  $P \in \{0.90, 0.95, 0.99\}$  while the possible number of samples that can be used for the test are  $n = 10, 11, \dots, 20$ . In order to determine the resulting confidence levels that can be obtained under these conditions, the engineer can construct an OC curve for  $1 - \alpha$  using the following code:

```
norm.OC(k = 4, alpha = NULL, P = c(0.90, 0.95, 0.99), n = 10:20,
        side = 2, method = "EXACT", m = 25)
```

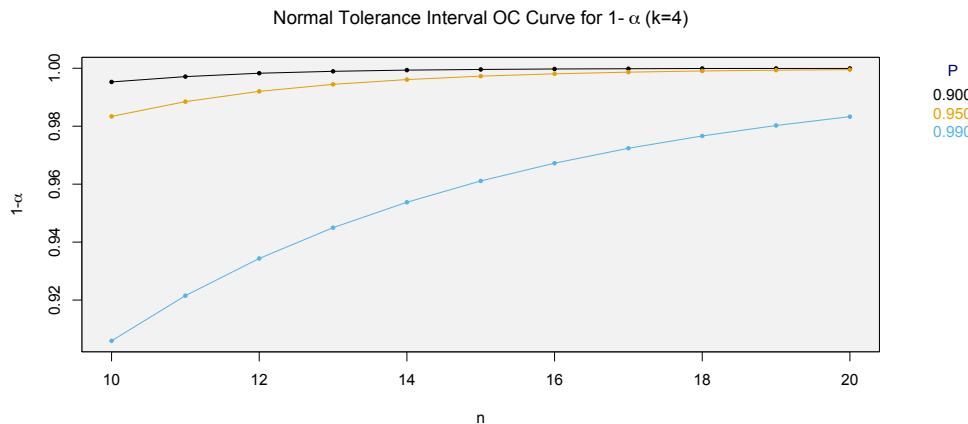
The resulting plot is given in Figure 3. For example, if the engineer chooses  $n = 15$ , then they can construct a two-sided tolerance interval with  $k = 4$  and content level of  $P = 0.99$  with confidence level near 0.96. However, if the engineer wishes to decrease the content of the tolerance interval to  $P = 0.95$  or  $P = 0.90$ , then a confidence level very near 1 can be achieved.

In the `norm.OC()` function, the arguments of `side`, `method`, and `m` are, again, passed down to the underlying `K.factor()` function. In order to generate Figure 3, we need to specify a single value for  $k$  (i.e. the  $k$ -factor) and at least one value for  $P$ . Since we are constructing curves where the sample size is on the  $x$ -axis, we need at least two values for  $n$ . Note that `alpha` must be left at its default `NULL` value.

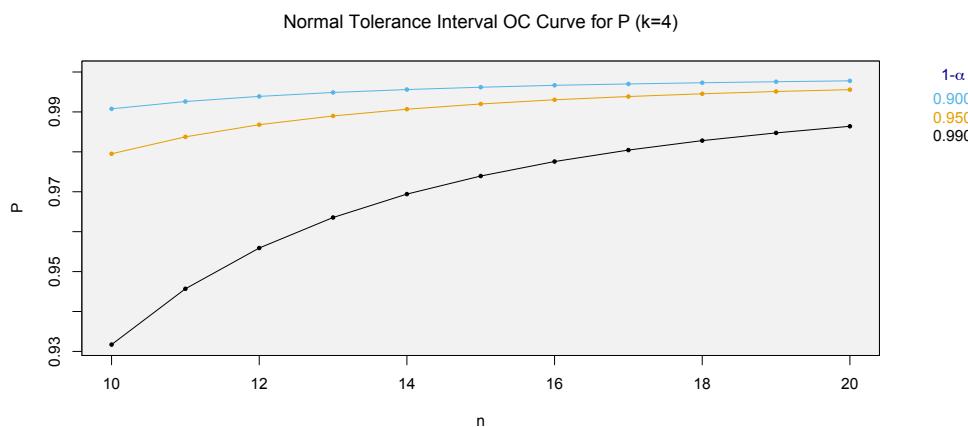
Suppose now that the same engineer considers confidence levels of  $1 - \alpha \in \{0.90, 0.95, 0.99\}$  with the same values of  $k$  and  $n$  from before. In order to determine the resulting content levels that can be obtained under these conditions, the engineer can construct an OC curve for  $P$  using the following code:

```
norm.OC(k = 4, alpha = c(0.01, 0.05, 0.10), P = NULL, n = 10:20,
        side = 2, method = "EXACT", m = 25)
```

The resulting plot is given in Figure 4. For example, if the engineer chooses  $n = 12$ , then they can construct a two-sided tolerance interval with  $k = 4$  and confidence level of  $1 - \alpha = 0.99$  that captures about 95% of the sampled population. However, if the engineer wishes to decrease the confidence level



**Figure 3:** OC curves for  $1 - \alpha$  given the set of content levels  $P \in \{0.90, 0.95, 0.99\}$ , sample sizes  $n = 10, 11, \dots, 20$ , and a two-sided  $k$ -factor of 4.



**Figure 4:** OC curves for  $P$  given the set of confidence levels  $1 - \alpha \in \{0.90, 0.95, 0.99\}$ , sample sizes  $n = 10, 11, \dots, 20$ , and a two-sided  $k$ -factor of 4.

of the tolerance interval to  $1 - \alpha = 0.95$  or  $1 - \alpha = 0.90$ , then a tolerance interval that captures about 99% of the sampled population can be achieved. Note that the code using the `norm.OC()` function is similar to the previous example, except that now we specify at least one value for `alpha` and leave `P` must at its default `NULL` value.

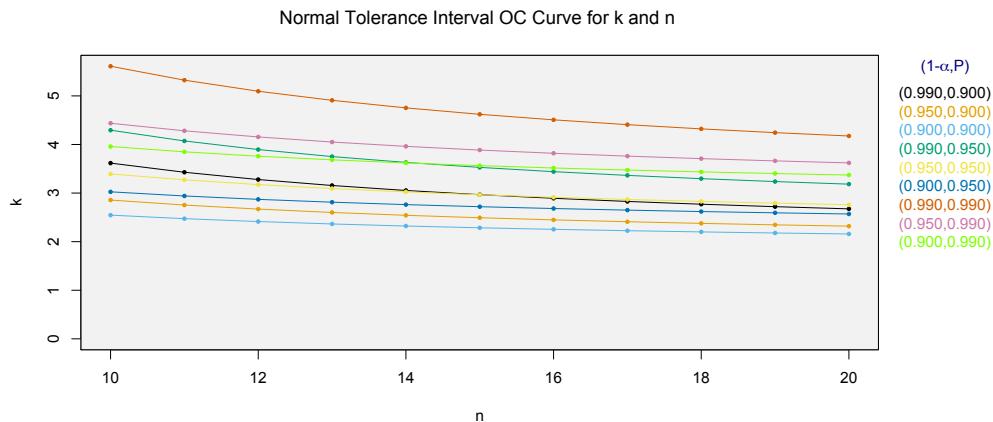
Finally, the `norm.OC()` function can also be used to construct an OC-curve where the  $k$ -factor is calculated for specified values of  $n$ ,  $1 - \alpha$ , and  $P$ . The different curves will be for each combination of the specified  $1 - \alpha$  and  $P$  levels. For our example, suppose the engineer is interested in the  $k$ -factors for two-sided tolerance intervals for the set of confidence levels  $1 - \alpha \in \{0.90, 0.95, 0.99\}$ , the set of content levels  $P \in \{0.90, 0.95, 0.99\}$ , and sample sizes  $n = 10, 11, \dots, 20$ . Then, we can specify the respective arguments in the `norm.OC()` function while leaving the `k` argument at its default `NULL` value:

```
norm.OC(k = NULL, P = c(0.90, 0.95, 0.99), alpha=c(0.01, 0.05, 0.10),
        n = 10:20, side = 2, method = "EXACT", m = 25)
```

The resulting plot is given in Figure 5. This OC-curve allows the user to assess the width of the tolerance interval as  $n$  changes for the given  $(1 - \alpha, P)$  tolerance levels.

## Summary

**tolerance** is the only R package devoted to the calculation of tolerance intervals and regions. Since its earlier versions (Young, 2010), there have been many updates to the package that include additional parametric tolerance interval procedures, improved nonparametric tolerance interval procedures, and some multivariate tolerance region procedures.



**Figure 5:** OC curves for the  $k$ -factor when given the set of confidence levels  $1 - \alpha \in \{0.90, 0.95, 0.99\}$ , the set of content levels  $P \in \{0.90, 0.95, 0.99\}$ , and sample sizes  $n = 10, 11, \dots, 20$ .

In this paper, we focused on the varied capabilities of **tolerance** pertaining to tolerance intervals for the normal distribution. Many of these procedures have been added to the package since the discussion presented in Young (2010). We discussed the calculation of one-sided normal tolerance limits, exact and equal-tailed normal tolerance intervals, Bayesian normal tolerance intervals, tolerance intervals for fixed-effects ANOVA, and sample size determination strategies. We also introduced novel operating characteristic (OC) curves that illustrate how the  $k$ -factor, sample size, confidence level, and content level each change relative to one another. As pointed out throughout our discussion, all of these procedures have a large degree of utility in a variety of practical contexts.

The **tolerance** package continues to expand the functions available for constructing tolerance intervals and regions. We note that some of the updates over the years have been a direct result of requests by end users of the package. Thus, one can expect additional capabilities in future versions of **tolerance**, both for the normal setting and other data settings.

## Bibliography

- D. Adler and D. Murdoch. *rgl: 3D visualization device system (OpenGL)*, 2014. URL <http://CRAN.R-project.org/package=rgl>. R package version 0.95.1201. [p200]
- J. Aitchison. Bayesian tolerance regions. *Journal of the Royal Statistical Society, Series B*, 26(2):161–175, 1964. [p203]
- T. Burr and A. Gavron. Pass/fail criterion for a simple radiation portal monitor test. *Modern Instrumentation*, 1(3):27–33, 2010. [p200]
- G. Csardi. *cranlogs: Download Logs from the 'RStudio' 'CRAN' Mirror*, 2015. URL <http://CRAN.R-project.org/package=cranlogs>. R package version 2.1.0. [p200]
- Environmental Protection Agency. *Data Quality Assessment: Statistical Methods for Practitioners*. U.S. Environmental Protection Agency, Washington, DC, USA, 2006. URL <http://www.epa.gov/sites/production/files/2015-08/documents/g9s-final.pdf>. [p200]
- G. D. Faulkenberry and D. L. Weeks. Sample size determination for tolerance limits. *Technometrics*, 10(2):343–348, 1968. [p206, 207, 208]
- A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis*. CRC Press, Boca Raton, FL, 3<sup>rd</sup> edition, 2013. [p204]
- I. Guttman. *Statistical Tolerance Regions: Classical and Bayesian*. Charles Griffin and Company, London, 1970. [p200, 203]
- S. D. Hafner, C. Howard, R. E. Muck, R. B. Franco, F. Montes, P. G. Green, F. Mitloehner, S. L. Trabue, and C. A. Rotz. Emission of volatile organic compounds from silage: Compounds, sources, and implications. *Atmospheric Environment*, 77:827–839, 2013. [p200]

- G. J. Hahn and W. Q. Meeker. *Statistical Intervals: A Guide for Practitioners*. Wiley-Interscience, New York, NY, 1991. [p200]
- I. J. Hall. Approximate one-sided tolerance limits for the difference or sum of two independent normal variates. *Journal of Quality Technology*, 16(1):15–19, 1984. [p200]
- M. M. Heck, M. Retz, M. Bandu, M. Souchay, E. Vitzthum, G. Weirich, M. Mollenhauer, T. Schuster, M. Autenrieth, H. Kübler, T. Maurer, M. Thalgott, K. Herkommer, J. E. Gschwend, and R. Nawroth. Topography of lymph node metastases in prostate cancer patients undergoing radical prostatectomy and extended lymphadenectomy: Results of a combined molecular and histopathologic mapping study. *European Urology*, 66(2):222–229, 2014. [p200]
- International Atomic Energy Agency. *Safety Report Series No. 52: Best Estimate Safety Analysis for Nuclear Plants: Uncertainty Evaluation*. IAEA Publishing Section, Vienna, Austria, 2008. URL [http://www-pub.iaea.org/MTCD/publications/PDF/Pub1306\\_web.pdf](http://www-pub.iaea.org/MTCD/publications/PDF/Pub1306_web.pdf). [p200]
- International Organization for Standardization. *ISO 16269-6: Statistical Interpretation of Data – Part 6: Determination of Statistical Tolerance Intervals*. International Organization for Standardization, Geneva, Switzerland, 2014. URL [http://www.iso.org/iso/catalogue\\_detail.htm?csnumber=57191](http://www.iso.org/iso/catalogue_detail.htm?csnumber=57191). [p200]
- W. A. Jensen. Approximations of tolerance intervals for normally distributed data. *Quality and Reliability Engineering International*, 25(5):571–580, 2009. [p202]
- K. Krishnamoorthy and T. Mathew. Comparison of approximation methods for computing tolerance factors for a multivariate normal population. *Technometrics*, 41(3):234–249, 1999. [p200]
- K. Krishnamoorthy and T. Mathew. *Statistical Tolerance Regions: Theory, Applications, and Computation*. Wiley, Hoboken, NJ, 2009. [p200, 202, 203, 204]
- K. Krishnamoorthy and S. Mondal. Improved tolerance factors for multivariate normal distributions. *Communications in Statistics - Simulation and Computation*, 35(2):461–478, 2006. [p200]
- T. Mathew and D. S. Young. Fiducial-based tolerance intervals for some discrete distributions. *Computational Statistics and Data Analysis*, 61:38–49, 2013. [p200]
- M. Okabe and K. Ito. Color Universal Design (CUD) - how to make figures and presentations that are friendly to colorblind people, 2002. URL <http://jfly.iam.u-tokyo.ac.jp/color/>. [p208]
- D. B. Owen. Control of percentages in both tails of the normal distribution. *Technometrics*, 6(4):377–387, 1964. [p202]
- C. Pasquaretta, G. Bogliani, L. Ranghetti, C. Ferrari, and A. von Hardenberg. The animal locator: A new method for accurate and fast collection of animal locations for visible species. *Wildlife Biology*, 18(2):202–214, 2012. [p200]
- RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA, 2015. URL <http://www.rstudio.com/>. [p200]
- L. H. C. Tippett. *Technological Applications of Statistics*. Wiley, New York, NY, 1950. [p205]
- K. Van der Borght, G. Verbeke, and H. van Vlijmen. Multi-model inference using mixed effects from a linear regression based genetic algorithm. *BMC Bioinformatics*, 15(88):1–11, 2014. [p200]
- A. Wald. An extension of Wilks' method for setting tolerance limits. *Annals of Mathematical Statistics*, 14(1):45–55, 1943. [p200]
- S. S. Wilks. Determination of sample sizes for setting tolerance limits. *The Annals of Mathematical Statistics*, 12(1):91–96, 1941. [p200]
- S. S. Wilks. Statistical prediction with special reference to the problem of tolerance limits. *The Annals of Mathematical Statistics*, 13(4):400–409, 1942. [p200]
- D. S. Young. tolerance: An R package for estimating tolerance intervals. *Journal of Statistical Software*, 36(5):1–39, 2010. URL <http://www.jstatsoft.org/v36/i05/>. [p200, 201, 202, 209, 210]
- D. S. Young. Computing tolerance intervals and regions using r. In M. B. Rao and C. R. Rao, editors, *Handbook of Statistics, Volume 32: Computational Statistics with R*, pages 309–338. North Holland - Elsevier, Amsterdam, Netherlands, 2014. [p200, 201]
- D. S. Young and T. Mathew. Ratio edits based on statistical tolerance intervals. *Journal of Official Statistics*, 31(1):77–100, 2015. [p200]

D. S. Young, C. M. Gordon, S. Zhu, and B. D. Olin. Sample size determination strategies for normal tolerance intervals using historical data. *Quality Engineering*, 28(3):335–349, 2016. [p<sup>207</sup>, <sup>208</sup>]

*Derek S. Young  
Department of Statistics  
University of Kentucky  
323 Multidisciplinary Science Building  
725 Rose Street  
Lexington, KY 40536-0082 USA  
derek.young@uky.edu*

# easyROC: An Interactive Web-tool for ROC Curve Analysis Using R Language Environment

by Dincer Goksuluk, Selcuk Korkmaz, Gokmen Zararsiz and A. Ergun Karaagaoglu

**Abstract** ROC curve analysis is a fundamental tool for evaluating the performance of a marker in a number of research areas, e.g., biomedicine, bioinformatics, engineering etc., and is frequently used for discriminating cases from controls. There are a number of analysis tools which are used to guide researchers through their analysis. Some of these tools are commercial and provide basic methods for ROC curve analysis while others offer advanced analysis techniques and a command-based user interface, such as the R environment. The R environments includes comprehensive tools for ROC curve analysis; however, using a command-based interface might be challenging and time consuming when a quick evaluation is desired; especially for non-R users, physicians etc. Hence, a quick, comprehensive, free and easy-to-use analysis tool is required. For this purpose, we developed a user-friendly web-tool based on the R language. This tool provides ROC statistics, graphical tools, optimal cutpoint calculation, comparison of several markers, and sample size estimation to support researchers in their decisions without writing R codes. easyROC can be used via any device with an internet connection independently of the operating system. The web interface of easyROC is constructed with the R package **shiny**. This tool is freely available through [www.biosoft.hacettepe.edu.tr/easyROC](http://www.biosoft.hacettepe.edu.tr/easyROC).

## Introduction

The receiver operating characteristics (ROC) curve is a graphical approach used to visualize and assess the performance of a binary classifier system. This unique feature of ROC curve analysis makes it one of the most extensively used methods in various fields of science. It was originally developed during World War II to detect whether a signal on the radar screen represented an object or a noise (Egan, 1975; Swets et al., 2000; Fan et al., 2006) and today it is widely used in medicine, radiology, biometrics, bioinformatics and various applications of machine learning and data mining research (Fawcett, 2006; Sonego et al., 2008). ROC curve analysis can be implemented for several reasons: (i) to assess the overall performance of a classifier using several performance measures, (ii) to compare the performances of classifiers, and (iii) to determine the optimal cutpoint for a given classifier, diagnostic test or marker/biomarker. For simplicity of language, we will use the terms classifier and diagnostic test throughout the manuscript. The performance of a classifier can be summarized using the point estimations and confidence intervals of several basic performance measures such as sensitivity, specificity or combined measures of sensitivity and specificity such as likelihood ratios, accuracy, area under the ROC curve (AUC), etc. A ROC curve is basically a plot of a classifier's true positive rates (TPR: sensitivity) versus false positive rates (FPR: 1 – specificity) where each point is generated by a different threshold value, i.e., cutpoint. For the simplicity of equations, we will use the terms TPR and FPR in the equations. One of the major tasks is to determine the optimum cutpoint value which corresponds to the reasonable TPR and FPR values. The determination of an optimum value is usually a trade-off between performance measures. The ROC curve is used to find the optimal cutpoint located on the curve which is the closest point to the top-left corner. However, finding the "optimum" cutpoint is not always based on maximizing the sensitivity and specificity. It is reasonable to select an optimum cutpoint value by regarding alternative selection criteria such as maximization of predictive values, diagnostic odds ratio, etc.

There are a number of commercial (e.g., IBM SPSS, MedCalc, Stata, etc.) and open-source (R) software packages which are used to guide researchers through their ROC curve analysis. Some of these software packages provide basic features for ROC curve analysis while others, such as R, offer advanced features but also a command-based user interface. The R environment includes comprehensive tools for ROC curve analysis, such as **ROCR** (Sing et al., 2005), **pROC** (Robin et al., 2011), **ROC** (Carey and Redestig, 2015) and **OptimalCutpoints** (Lopez-Raton et al., 2014).

All of the R packages mentioned above perform ROC curve analysis using the related package functions. Although these packages are comprehensive and flexible, they require a good programming knowledge of the R language. However, working with a command-based interface might be challenging and time consuming when a quick evaluation is desired especially for non-R users, such as physicians and other health care professionals. Fortunately, an R package **shiny** (Chang et al., 2015) allows users to create interactive web-tools with a nicely designed, user-friendly and easy-to-use user interface. In this context, we developed a web-tool, easyROC, for ROC curve analysis. The

user interface of easyROC is constructed via **shiny** and HTML codes. easyROC combines several R packages for ROC curve analysis. This tool has three main parts including ROC statistics, cutpoint calculations and sample size estimation. Detailed information about easyROC and the related methods together with mathematical background are given in Section [Material and methods](#). easyROC is freely available at <http://www.biosoft.hacettepe.edu.tr/easyROC> and all the source codes are on GitHub<sup>1</sup>.

## Material and methods

### Theory behind ROC analysis

Let us consider the binary classification problem where  $X$  denotes the value of the classifier for cases and controls. Consider the values of controls distributed as  $X_0 \sim G_0(\cdot)$  and cases as  $X_1 \sim G_1(\cdot)$ . Let  $\hat{Y} = \{0, 1\}$  be the estimated class labels of the subjects for a given threshold value  $c$  as given in Equation 1.

$$\hat{Y} = \begin{cases} 1, & \text{if } X \geq c \\ 0, & \text{if } X < c \end{cases} \quad (1)$$

**Parametric ROC curve.** The parametric ROC curve is plotted using the FPR (1 – Specificity) and TPR (Sensitivity) values given in Equation 2 for all possible cutpoints of a classifier.

$$\begin{aligned} FPR_c &= P(X \geq c | Y = 0) = \int_c^{\infty} G_0(x) dx \\ TPR_c &= P(X \geq c | Y = 1) = \int_c^{\infty} G_1(x) dx \end{aligned} \quad (2)$$

When the distribution of the classifier is *Normal*, the parametric ROC curve is fitted using *binormal* ROC properties. Suppose  $X_0 \sim \text{Normal}(\mu_0, \sigma_0^2)$  and  $X_1 \sim \text{Normal}(\mu_1, \sigma_1^2)$ . The ROC curve is the function of FPRs; as in Equation 3.

$$ROC(t) = \Phi(a + b\Phi^{-1}(t)), \quad (3)$$

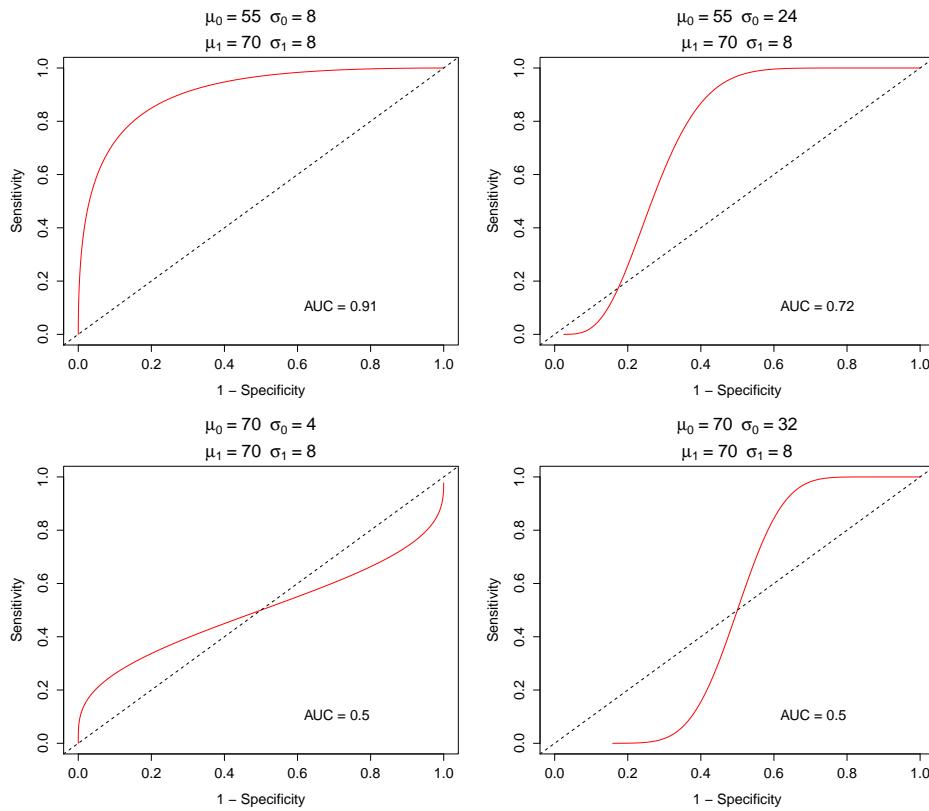
where  $a = (\mu_1 - \mu_0)/\sigma_1$ ,  $b = \sigma_0/\sigma_1$ ,  $t = FPR_c$  and  $\Phi$  is the cumulative distribution function of the standard normal distribution (Zhou et al., 2002). The area under the curve is calculated using Equation 4.

$$AUC = \int_0^1 ROC(t) dt = \Phi\left(\frac{a}{\sqrt{1+b^2}}\right) \quad (4)$$

Fitting the ROC curve by using Equation 3 has two major drawbacks: (i) incorrect ROC curves may arise when the underlying distribution is not normal, (ii) ROC lines are improper when within class variations are not similar, i.e., heteroscedasticity. An example of improper ROC curves is given in Figure 1. To overcome these problems, one may nonparametrically fit the ROC curve without considering distributional assumptions or use parametric/semiparametric alternatives to the binormal model (Gönen and Heller, 2010).

**Nonparametric ROC curve.** Consider the estimated class labels in Equation 1. The FPR and TPR given in Equation 2 are estimated; as given in Equation 5.

<sup>1</sup><http://www.github.com/dncR/easyROC>

**Figure 1:** Parametric ROC curves.

$$\begin{aligned}\widehat{FPR}_c &= \frac{1}{n_0} \sum_{j=1}^{n_0} I \left[ X_{0j} \geq c \right] \\ \widehat{TPR}_c &= \frac{1}{n_1} \sum_{i=1}^{n_1} I [ X_{1i} \geq c ]\end{aligned}\quad (5)$$

The empirical ROC curve is plotted using  $\widehat{FPR}_c$  and  $\widehat{TPR}_c$  and the area under the curve, given in Equation 6, is estimated by summing the trapezoids enclosed by the points of the ROC curve. The nonparametric AUC is related to the Mann-Whitney statistic of the rank-sum test (Bamber, 1975; Hanley and McNeil, 1982).

$$\widehat{AUC} = \frac{1}{n_0 n_1} \sum_{j=1}^{n_0} \sum_{i=1}^{n_1} \Psi(X_{1i}, X_{0j}), \quad (6)$$

where  $\Psi = 0$  if  $X_0 > X_1$ ,  $\Psi = 1$  if  $X_1 > X_0$  and  $\Psi = 1/2$  if  $X_0 = X_1$ .

**Performance measures and optimal cutpoints.** The predicted and actual classes, i.e., gold standard test results, can be shown with a  $2 \times 2$  cross table; as seen in Table 1. The performance of a classifier is basically measured using the total proportion of true positive (TP) and true negative (TN) cases. By using Table 1, several performance measures are also calculated. Among these performance measures, we focused on the measures given in Table 1 which are widely used and well-known. The optimal cutpoint is determined by using one or more performance measures together. An ideal cutpoint, for example, might be selected by maximizing the sensitivity and specificity of a classifier. A classifier with perfect discriminative ability would have sensitivity and specificity measures equal to 1. Hence, the area under the curve for a perfect separation will be equal to 1.

Although researchers are usually interested in the overall diagnostic performance of a classifier, it is sometimes useful to focus on a portion of the ROC curve to compute the partial AUCs (pAUC). pAUC is an extension of the AUC measure which considers the trapezoids within a given interval of

Predicted Labels	Actual Labels		Total
	Positive ( $Y = 1$ )	Negative ( $Y = 0$ )	
Positive ( $\hat{Y} = 1$ )	TP	FP	TP + FP
Negative ( $\hat{Y} = 0$ )	FN	TN	FN + TN
Total	TP + FN	FP + TN	$n$
TP: True positive FP: False positive TN: True negative FN: False negative NPV: Negative predictive value PPV: Positive predictive value PLR: Positive likelihood ratio NLR: Negative likelihood ratio	Sensitivity = $TP/(TP + FN)$ Specificity = $TN/(FP + TN)$ $PPV = TP/(TP + FP)$ $NPV = TN/(TN + FN)$ $PLR = Sensitivity/(1 - Specificity)$ $NLR = (1 - Sensitivity)/Specificity$		

**Table 1:** A  $2 \times 2$  classification table and performance measures.

sensitivity and/or specificity. Let us consider the pAUC where specificity (or sensitivity) lies within the interval  $[t_1, t_2]$ . The pAUC is calculated by taking the integral (parametric) as given in Equation 7 or by summing the trapezoids within the interval (nonparametric).

$$pAUC(t_1, t_2) = \int_{t_1}^{t_2} ROC(x)dx \quad (7)$$

As the interval  $[t_1, t_2]$  converges to  $[0, 1]$ , the pAUC will converge to the overall AUC. The best classifier can be selected using either AUC or pAUC values.

Identification of the optimal cutpoint is an important task to avoid incorrect conclusions. Various methods are available in the literature to determine the optimal cutpoint. Most of these methods are based on the sensitivity and specificity measures. However, other methods are also available based on cost-benefit, prevalence, predictive values and diagnostic likelihood ratios. Two popular methods are, for example, the Youden index and the minimization of the distance of the point on the curve to the top-left corner, i.e., the point indicating perfect discrimination.

$$\text{Youden}(c) = \max\{TPR_c - FPR_c\} \quad (8)$$

Table 1 gives the list of optimal cutpoint methods we consider in easyROC. For detailed information and mathematical background, see [Lopez-Raton et al. \(2014\)](#).

**Statistical inference.** A common subject of interest in ROC analysis is to compare the performances of several classifiers to select the best one to discriminate cases from controls. For a classifier with random chance discrimination ability, the equation  $TPR = FPR$  holds. In that case, the area under the curve is 0.50. Hence, the discrimination ability of a classifier is mostly tested against the value 0.50.

$$\begin{aligned} H_0 : AUC &= 0.50 \\ H_1 : AUC &\neq 0.50 \end{aligned}$$

Under the large sample theory, the significance of AUC is tested using the *Wald* test statistic as given in Equation 9.

$$z = \frac{\widehat{AUC} - AUC}{\text{Var}(\widehat{AUC})^{1/2}} \quad (9)$$

When the parametric approach is used, the variance of AUC is estimated using Equation 10 (McClish, 1989; Zhou et al., 2002).

$$\text{Var}(\widehat{AUC}) = f^2 \text{Var}(\hat{a}) + g^2 \text{Var}(\hat{b}) + 2fg \text{Cov}(\hat{a}, \hat{b}), \quad (10)$$

where

$$f = \frac{e^{-a^2/2(1+b^2)}}{\sqrt{2\pi(1+b^2)}} \quad \text{and} \quad g = -\frac{abe^{-a/2(1+b^2)}}{\sqrt{2\pi(1+b^2)^3}} \quad (11)$$

and the estimated variances for  $a$  and  $b$  as follows:

$$\begin{aligned} \widehat{\text{Var}}(\hat{a}) &= \frac{n_1(\hat{a}^2 + 2) + 2n_0\hat{b}^2}{2n_0n_1}, \\ \widehat{\text{Var}}(\hat{b}) &= \frac{(n_1 + n_0)\hat{b}^2}{2n_0n_1}, \\ \widehat{\text{Cov}}(\hat{a}, \hat{b}) &= \frac{\hat{a}\hat{b}}{2n_0}. \end{aligned} \quad (12)$$

The estimated values of  $a$  and  $b$  are used in Equation 11. A number of methods have been proposed for the estimation of the variance of AUC when the nonparametric approach is used. In this paper, we will focus on the methods described below:

### 1. Mann-Whitney version of rank-sum test:

Hanley and McNeil (1982) propose the variance estimation given in Equation 13. This method estimates the variance using an approximation based on exponential distribution as

$$\begin{aligned} \text{Var}(\widehat{\text{AUC}}) &= \frac{1}{n_0n_1} \left\{ \text{AUC}(1 - \text{AUC}) + (n_1 - 1)(Q_1 - \text{AUC}^2) \right. \\ &\quad \left. + (n_0 - 1)(Q_2 - \text{AUC}^2) \right\}, \end{aligned} \quad (13)$$

where  $Q_1 = \widehat{\text{AUC}}/(2 - \widehat{\text{AUC}})$  and  $Q_2 = 2\widehat{\text{AUC}}^2/(1 + \widehat{\text{AUC}})$ . The Mann-Whitney version might underestimate the variance when the area is nearly 0.5 and overestimate it when the area is close to 1 (Hanley and McNeil, 1982; Hanley and Hajian-Tilaki, 1997; Obuchowski, 1994). This estimate is mostly used in sample-size estimation.

### 2. DeLong et al. (1988)'s estimate:

Since the exponential distribution approximation in Equation 13 gives biased variance estimates, DeLong et al. (1988) suggest an alternative method which is free from distributional assumptions. Define the components  $T_{1i}$  for the  $i$ th subject from cases and  $T_{0j}$  for the  $j$ th subject from controls as follows:

$$\begin{aligned} \psi(T_{1i}) &= \frac{1}{n_0} \sum_{j=1}^{n_0} \Psi(X_{1i}, X_{0j}) \quad i = 1, 2, \dots, n_1 \\ \psi(T_{0j}) &= \frac{1}{n_1} \sum_{i=1}^{n_1} \Psi(X_{1i}, X_{0j}) \quad j = 1, 2, \dots, n_0 \end{aligned} \quad (14)$$

Using the Equation 14 the variance of AUC is estimated as

$$\text{Var}(\widehat{\text{AUC}}) = \frac{1}{n_1} S_{T_1}^2 + \frac{1}{n_0} S_{T_0}^2, \quad (15)$$

where  $S_{T_1}^2$  and  $S_{T_0}^2$  are variance estimates of  $T_1$  and  $T_0$  as in Equation 16.

$$S_{T_i}^2 = \frac{1}{n_i - 1} \sum_{j=1}^{n_i} [\psi(T_{ij}) - \widehat{\text{AUC}}]^2 \quad i = 0, 1 \quad (16)$$

### 3. Normal approximation of binomial proportion:

Another alternative for variance estimation is to use binomial approximation under the large sample theory, as given in Equation 17. For small samples, this method may give biased estimates.

$$\text{Var}(\widehat{\text{AUC}}) = \frac{\text{AUC}(1 - \text{AUC})}{n_0 + n_1} \quad (17)$$

The estimated variance derived from one of the methods described above is used to construct the confidence intervals of the AUC. A common method is to use large sample approximation as below:

$$\widehat{\text{AUC}} - z_{1-\alpha/2} \text{Var}(\widehat{\text{AUC}})^{1/2} < \text{AUC} < \widehat{\text{AUC}} + z_{1-\alpha/2} \text{Var}(\widehat{\text{AUC}})^{1/2}. \quad (18)$$

When the area under the curve is close to 1 or the sample size is relatively small, the large sample approximation in Equation 18 produces improper confidence intervals since the upper limit exceeds 1. To solve this problem, Agresti and Coull (1998) proposed the *score confidence interval* that guarantees the upper limit is less than or equal to 1. Another alternative is to construct the binomial exact confidence intervals given in Equation 19 using the relationship between binomial and F-distribution (Morisette and Khorram, 1998)

$$\frac{1}{1 + \frac{n-x+1}{x} F_{2(n-x+1), 2x, \alpha/2}} \leq p \leq \frac{\frac{x+1}{n-x} F_{2(x+1), 2(n-x), \alpha/2}}{1 + \frac{x+1}{n-x} F_{2(x+1), 2(n-x), \alpha/2}}, \quad (19)$$

where  $p = x/n$  is the binomial proportion such as sensitivity, specificity and AUC.

**Sample size calculation.** In most studies, determining the required sample size is an important step for the research to be able to detect significant results. Sample size determination is required for both constructing the confidence interval of the unknown population parameter and testing a research hypothesis. Obuchowski (1998) reviewed sample size determination for several study designs. In this paper, we cover the sample size determination for three types of studies based on AUCs. In addition, the following sample size calculations can be extended to other performance measures such as sensitivity, specificity, etc.

The variance estimates of AUCs can be obtained using one of the Equations 13, 15 and 17. While Equation 13 is a good approximation for a variety of underlying distributions, the estimated variance will be underestimated if the test results are in a discrete rating format. To overcome this problem, Obuchowski (1998) and Obuchowski et al. (2004) suggest an alternative variance estimation method for rating data using the variance function as given in Equation 20 which is based on an underlying binormal distribution. In this section, we focused on sample size calculation for discrete scale data. However, the same formulas are valid for continuous scale diagnostic tests since the only difference is about estimating the variance of diagnostic test accuracy.

$$\text{V}(\widehat{\text{AUC}}) = 0.0099 e^{-a^2/2} \times \left[ (5a^2 + 8) + (a^2 + 8)/R \right], \quad (20)$$

where  $a = \sqrt{2} \Phi^{-1}(\text{AUC})$  and  $R = n_0/n_1$  is the allocation ratio, i.e., the ratio of the number of controls to the number of cases. The estimated variance is then  $\text{Var}(\widehat{\text{AUC}}) = \text{V}(\widehat{\text{AUC}})/n_1$ . The total sample size is equal to  $n = n_1(1 + R)$ . One of the variance estimations from Equations 13, 15, 17 and 20 is used for the sample size calculations. The selection of the appropriate variance estimation method is based on the variable type of the test results and underlying distributions.

### 1. Hypothesis test to determine the AUC of a single classifier:

In most of the studies with a single classifier, the aim of the study is to determine whether the diagnostic test performs well for discriminating diseased patients from controls. Consider the hypotheses  $H_0 : \text{AUC} = 0.5$  versus  $H_1 : \text{AUC} > 0.5$  (i.e, one-sided test). The required number of cases is determined using Equation 21 (Obuchowski et al., 2004).

$$\begin{aligned}
n_1 &= \frac{\left[ z_{1-\alpha} \sqrt{\text{Var}_0(\widehat{AUC})} + z_{1-\beta} \sqrt{\text{Var}_1(\widehat{AUC})} \right]^2}{(AUC - 0.5)^2} \\
&= \frac{\left[ z_{1-\alpha} \sqrt{0.0792 \times (1 + 1/R)} + z_{1-\beta} \sqrt{\text{Var}_1(\widehat{AUC})} \right]^2}{(AUC - 0.5)^2}, \tag{21}
\end{aligned}$$

where  $\text{Var}_0$  and  $\text{Var}_1$  are the variance estimations under the null and alternative hypotheses using Equation 20.  $z_{1-\alpha}$  and  $z_{1-\beta}$  are lower-tailed percentile values of the cumulative standard normal distribution. Finally, the total sample size is obtained using  $n = n_1 + n_1 \times R$ .

## 2. Comparing the AUCs of two classifiers:

When the aim of a study is to compare two classifiers, one may consider the hypotheses  $H_0 : AUC_1 = AUC_2$  versus  $H_1 : AUC_1 \neq AUC_2$ . The two classifiers will be equally performing under the null hypothesis. The required number of cases is calculated using Equation 22.

$$n_1 = \frac{\left[ z_{1-\alpha/2} \sqrt{\text{Var}_0(\widehat{AUC}_1 - \widehat{AUC}_2)} + z_{1-\beta} \sqrt{\text{Var}_1(\widehat{AUC}_1 - \widehat{AUC}_2)} \right]^2}{(AUC_1 - AUC_2)^2}, \tag{22}$$

where  $\text{Var}_0$  and  $\text{Var}_1$  are the variance estimations under the null and alternative hypotheses; as given in Equation 23 (Zhou et al., 2002; Obuchowski et al., 2004).

$$\text{Var}(\widehat{AUC}_1 - \widehat{AUC}_2) = \text{Var}(\widehat{AUC}_1) + \text{Var}(\widehat{AUC}_2) - 2\text{Cov}(\widehat{AUC}_1, \widehat{AUC}_2) \tag{23}$$

The total sample size is calculated using the allocation ratio. When two classifiers are performed on the same subjects, the design will be *paired* yielding the covariance term to be a nonzero (usually positive) quantity. However, the covariance term will be zero (i.e., independent classifiers) if each test is performed on different subjects. Detailed information on the calculation of the covariance term can be found in Zhou et al. (2002).

## 3. Non-inferiority of a new classifier to a standard one:

In addition to comparing two classifiers, some studies are designed to explore the performance of a new classifier to that of a standard one. The new classifier should perform as well as but not necessarily better than the standard test (Obuchowski et al., 2004). The hypotheses are  $H_0 : AUC_{std} - AUC_{new} \geq \Delta$  versus  $H_1 : AUC_{std} - AUC_{new} < \Delta$ . The required number of cases is calculated using Equation 24

$$n_1 = \frac{(z_{1-\alpha} + z_{1-\beta})^2 \text{Var}_1(\widehat{AUC}_{std} - \widehat{AUC}_{new})}{(AUC_{std} - AUC_{new} - \Delta)^2}, \tag{24}$$

where  $\Delta$  is the non-inferiority margin, i.e., the minimum acceptable difference between the AUCs of the standard and new classifiers.

## Current ROC analysis tools and easyROC

ROC curve analysis is one of the standard procedures included in most statistical analysis tools such as IBM SPSS, Stata, MedCalc and R. Each tool offers different features within ROC curve analysis. Among commercial software packages, IBM SPSS, which is one of the most widely used commercial software packages, plots the ROC curve and computes some basic statistics such as AUC and its standard error, confidence interval and statistical significance. However, it does not provide any method for sample size calculation or cutpoint determination. Stata offers a variety of calculations for ROC curve analysis including partial AUC, multiple comparisons of ROC curves, optimal cutpoint determination using the Youden index and several performance measures. Another commercial software alternative for ROC curve analysis is MedCalc, which has comprehensive features compared to most of the other available commercial software packages and is especially developed for biomedical research. MedCalc provides sample size estimation for a single diagnostic test, but it does not have an option for pAUC calculation.

Unlike commercial software packages, R is an open source and free software package that includes all the features of commercial software packages and more through several packages such as **ROC**,

	IBM SPSS	Stata	MedCalc	ROC	ROCR	pROC	easyROC
Plots	Yes	Yes	Yes*	Yes	Yes*	Yes*	Yes*
Conf. intervals	Yes	Yes*	Yes	Yes	Yes	Yes*	Yes*
pAUC	No	Yes	No	Yes	Yes	Yes*	Yes*
Statistical tests	No	Yes	No	Yes	Yes	Yes*	Yes*
Diagnostic measures	No	Yes	Yes	No	Yes*	Yes	Yes
Multiple comp.	No	Yes	Yes*	No	No	Yes*	Yes
Cutpoints	No	Yes	Yes	No	No	Yes	Yes*
Sample size	No	No	Yes	No	No	Yes	Yes*
Free license	No	No	No	Yes*	Yes*	Yes*	Yes*
Open source	No	No	No	Yes*	Yes*	Yes*	Yes*
Web-tool access	No	No	No	No	No	No	Yes*
User interface	Yes	Yes*	Yes*	No	No	Yes*	Yes*

\* Comprehensive ones.

**Table 2:** Comparison of easyROC with other tools.

**ROCR, pROC and OptimalCutpoints.** **ROC** is an R/Bioconductor package which can plot the ROC curve and calculate the AUC. It also calculates pAUCs based on false positive rates. This package is originally developed to be used for the ROC analysis with DNA microarrays. **ROCR** is a comprehensive R package providing over 25 different performance measures (based on package version 1.0-7). It allows users to create two dimensional performance curves. Although **ROCR** is one of the most comprehensive packages for assessing the performance measures, it provides limited options to select the optimum cutpoint. One may use any of the two-dimensional performance graphs to determine the optimal cutpoint graphically. It computes the AUC and its confidence interval, however, it does not provide a statistical test for performance measures.

**pROC**, on the other hand, offers more comprehensive and flexible features than its free and commercial counterparts. It performs statistical tests for the comparison of ROC curves using DeLong et al. (1988), Venkatraman and Begg (1996) and Venkatraman (2000) for AUC, and Hanley and McNeil (1983) and Pepe et al. (2009) for both AUC and pAUC. It also calculates the confidence intervals for the sensitivity, specificity, ROC curves, pAUC, and smoothed ROC curves. The confidence intervals are computed using DeLong et al. (1988)'s method for AUCs and using bootstrap for pAUCs, sensitivity and specificity at given threshold(s). Bootstrap confidence intervals and pAUC regions are shown in the ROC curve plot. Several diagnostic measures, such as sensitivity, specificity, negative and positive predictive values, are computed for a given threshold. Like **ROCR**, **pROC** also offers limited features for detecting the optimal cutpoint. Two methods, i.e., Youden index and closest point to the top-left corner, are available to find the optimal cutpoint. In addition, **pROC** is an alternative among the ROC packages on CRAN to find the required sample size for a single diagnostic test or the comparison of two diagnostic tests. Two versions of **pROC** are available: (i) for the R programming language and (ii) with a graphical user interface for the S-PLUS statistical software package.

There are several packages providing optimal cutpoint calculations through R. **OptimalCutpoints** is a sophisticated R package specifically developed to determine the optimal cutpoint of a test or biomarker (Lopez-Raton et al., 2014). It includes 34 different cutpoint calculation methods based on sensitivity/specificity measures, cost-benefit analysis, predictive values, diagnostic likelihood ratios, prevalences and *p*-values. A brief description of these methods is given in Supplementary 1. Although these R packages, especially **pROC**, seem to be a perfect match for ROC curve analysis, none of them has a graphical user interface and all require coding knowledge, which makes them hard to use; especially for non-R users.

Another R package worth mentioning is **plotROC** (Sachs, 2016) which is available on CRAN and also for **shiny** platforms. **plotROC** is a flexible and sophisticated R package which can be used to create nice-looking and interactive ROC graphs. Unlike the packages described above, **plotROC** has a web-based user interface which is very useful for non-R users. Researchers can use its web service to create ROC graphics and download the figures to their local computer. However, it does not provide any statistical tests or sample size calculations.

easyROC aims to extend the features of several ROC packages in R and allows researchers to conduct their ROC curve analysis through a single and easy-to-use interface without writing any R code. This tool is a web-based application created via **shiny** and HTML programming. easyROC makes use of the R packages **plyr** (Wickham, 2011), **pROC** and **OptimalCutpoints** for conducting ROC analysis. **plyr** is used for manipulating data while **pROC** is used for estimation and hypothesis testing of pAUCs. easyROC has comprehensive options for ROC curve analysis which other tools do not have

Modules (Tab panels)	Features
ROC curve	<ul style="list-style-type: none"> <li>• Parametric/Nonparametric ROC</li> <li>• AUC, pAUC <ul style="list-style-type: none"> <li>– Confidence interval (Exact and Asymptotic)</li> <li>– Significance test (Wald)</li> </ul> </li> <li>• Standard error estimation <ul style="list-style-type: none"> <li>– DeLong (1988)</li> <li>– Mann-Whitney</li> <li>– Binomial approximation</li> </ul> </li> <li>• Multiple comparison of AUCs <ul style="list-style-type: none"> <li>– Bonferroni</li> <li>– False discovery rate</li> </ul> </li> <li>• ROC plot (customizable)</li> </ul>
Cutpoints	<ul style="list-style-type: none"> <li>• 34 different methods for optimum cutpoints (<a href="#">Lopez-Raton et al., 2014</a>)</li> <li>• Performance measures with confidence intervals <ul style="list-style-type: none"> <li>– Exact CIs</li> <li>– Asymptotic CIs</li> </ul> </li> <li>• Cutpoint graphs (fully customizable) <ul style="list-style-type: none"> <li>– ROC curve</li> <li>– Sensitivity &amp; specificity plot</li> <li>– Density plot</li> <li>– Scatter plot</li> </ul> </li> </ul>
Sample size	<ul style="list-style-type: none"> <li>• Single diagnostic test</li> <li>• Comparison of two diagnostic tests</li> <li>• Noninferiority of a new test to a standard test</li> </ul>

**Table 3:** Features of easyROC.

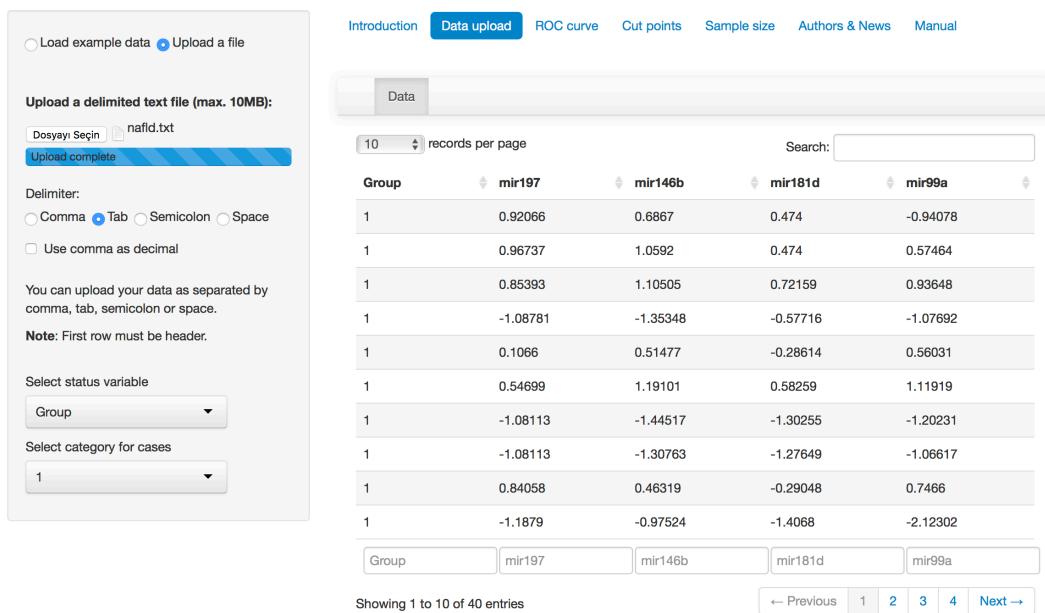
(or partially shares some features). The ROC curve can be estimated using parametric or nonparametric approaches. It offers four different methods for the calculation of the standard error and confidence interval of the AUC. Researchers can calculate the pAUCs based on sensitivity and specificity, if necessary. One may perform pairwise comparisons to find the classifiers which have similar or different discrimination ability. However, the pairwise comparison should be carried out carefully since the type I error increases with the increasing number of comparisons. easyROC offers multiple test corrections in order to keep type I error at a given level. Multiple comparisons of diagnostic tests can be applied using either Bonferroni or false discovery rate correction. Furthermore, the optimal cutpoints are determined using the methods from **OptimalCutpoints** and the corresponding measures at a given cutpoint, including sensitivity, specificity, positive and negative predictive values, and positive and negative likelihood ratios are also returned. One can determine the desired sample size for ROC curve analysis using this tool for three different cases. All these comprehensive features are accessible through a graphical user interface, which makes the analysis process easier for all users. The comparison with other tools is given in Table 2 and the features of each module are given in Table 3.

## Results

### Case study on non-alcoholic fatty liver disease

To illustrate our application, we used the non-alcoholic fatty liver disease (NAFLD) dataset of [Celikbilek et al. \(2014\)](#). This study was designed to identify the non-invasive miRNA biomarkers of NAFLD. The authors obtained the serum samples of 20 healthy and 20 NAFLD observations and quantified the expression levels of eight miRNAs using quantitative Real-Time PCR (qPCR) technology. After performing the necessary statistical analysis, the authors revealed that miR-197, miR-146b, miR-181d and miR-99a may be potential biomarkers in identifying NAFLD. The normalized expression values

## easyROC: a web-tool for ROC curve analysis (ver. 1.3)



**Figure 2:** Uploading data into easyROC.

of these miRNAs and the class information (the column named “Group”, where 0 refers to controls and 1 refers to cases) of each observation are given in Supplementary 2. This file can be directly used as input to the easyROC web-tool and users can arrange their own data based on this file. Two example datasets, Mayo and PBC (Murtaugh et al., 1994), are also available in the web-tool for users to practice the application. In our example, the aim is to investigate the discriminative performances of each miRNA, compare each other and identify the optimal cutpoints for each miRNA in identifying NAFLD.

### Implementation of easyROC web-tool

The data are uploaded to the easyROC interface using the *Data upload* tab (Figure 2). easyROC accepts a delimited text file with variable names in the first row. The status variable is also set by the same tab panel. easyROC automatically detects the variable names and exports them into related fields. When data are correctly uploaded, researchers may proceed with ROC curve analysis, cutpoint estimations or sample size calculations. The area under the curve, confidence intervals and significance tests for AUC, multiple comparisons (if multiple markers are selected) and pAUCs are calculated with the *ROC curve* tab (Figures 3 and 4). The ROC curve is estimated using the nonparametric approach. The advanced option allows researchers to select a method for standard error estimation and confidence intervals. easyROC selects the DeLong et al. (1988) method by default.

Here, we select mir197, mir146b, mir181d and mir99a miRNAs to assess their performances and to compare them with each other in identifying NAFLD. Since the expression levels of all miRNAs are underexpressed in the NAFLD group, lower values will indicate higher risk and therefore we should uncheck the “Higher values indicate higher risks” box. Using DeLong et al. (1988) standard error estimations, we obtained the ROC curves for each miRNA biomarker and AUC values as 0.86 (0.75–0.97), 0.77 (0.61–0.92), 0.76 (0.60–0.93) and 0.75 (0.59–0.91) for mir181d, mir197, mir99a and mir146b, respectively. The results revealed that all miRNAs’ predictive performances are significant and higher than random chance in identifying NAFLD (Figure 3). By controlling the type I error using Bonferroni correction, all pairwise comparisons showed non-significant results ( $p > 0.05$ ). This may be due to the small sample size of the data. Increasing the sample size, thus the statistical power of the test, may concretize the predictive ability of mir181d as compared to other miRNAs.

Finding a suitable cutpoint is one of the aims of ROC curve analysis. We made use of the **OptimalCutPoints** package from R (Lopez-Raton et al., 2014), which has 34 different methods, to calculate cutpoints for each marker. An optimal cutpoint can be computed via the *Cut point* tab by selecting a marker and a method. Then, the application will calculate an optimal cutpoint and

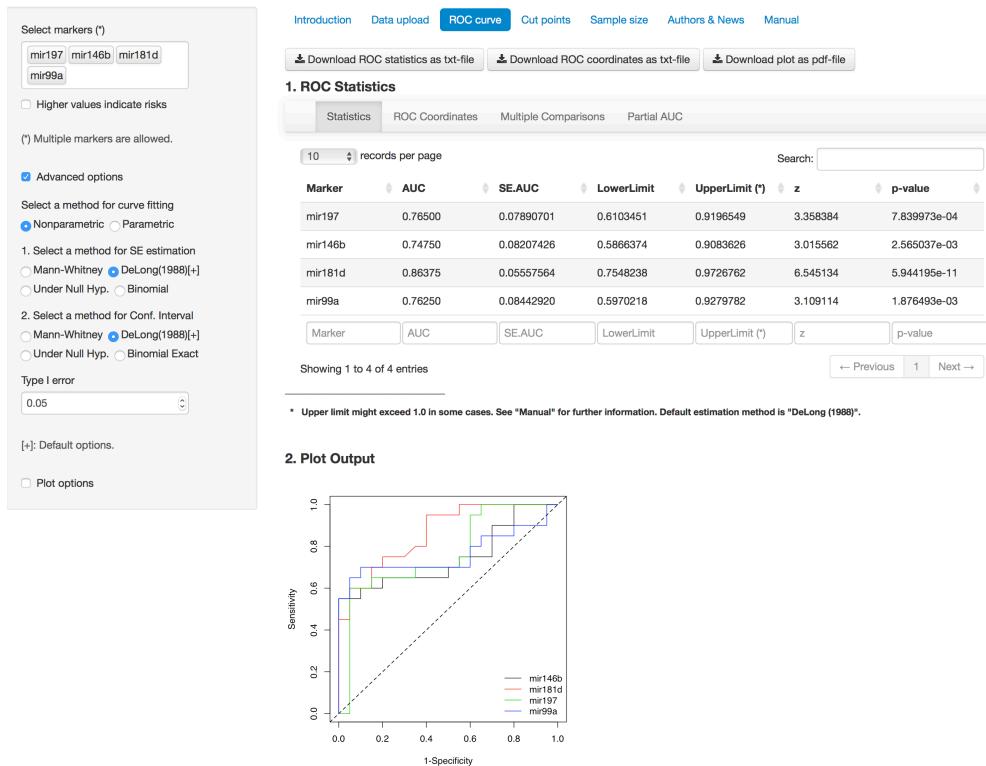


Figure 3: ROC curve analysis results.

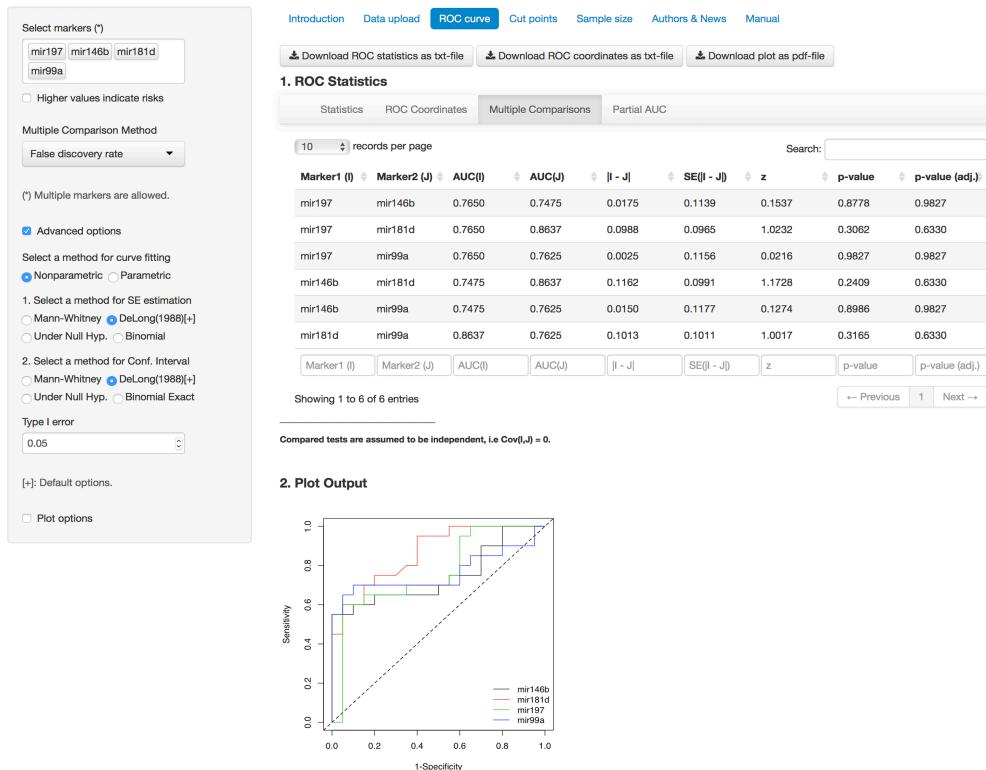
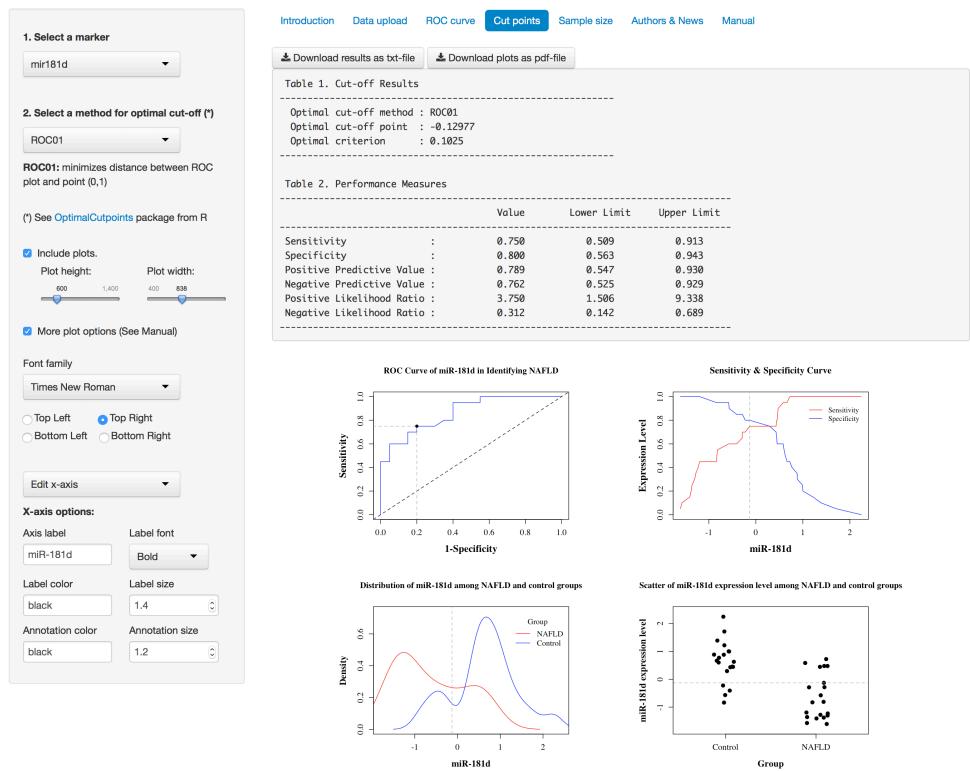


Figure 4: Multiple comparison of the diagnostic tests.

performance measures such as sensitivity, specificity, positive and negative predictive value, and positive and negative likelihood ratio based on the corresponding cutpoint value. The "ROC01" method, for example, determines the optimal cutpoint as  $-0.12977$  for mir181d. Using this cutpoint, a new test observation with a mir181d expression level lower than this value can be assigned as an



**Figure 5:** Determination of optimal cutpoint(s).

NAFLD patient. Based on the identified cutpoint, we obtained statistical diagnostic measures with 95% confidence intervals (Figure 5). We obtain a sensitivity of 0.75 (0.51–0.91) and specificity of 0.80 (0.56–0.94). If users select the “Include plots” option, four plots will appear under the statistics results. The first plot in the upper-left corner displays the optimal cutpoint on the ROC curve. Users can observe the change of sensitivity and specificity measures based on the value of the marker on the plot placed in the upper-right corner. The density and scatter of the expression values in each group are displayed in the bottom-left and bottom-right corners. The plots can be modified through the “More plot options” section. All the results and figures can be downloaded using the related “Download” buttons in each tab panel.

## Conclusion

Since ROC curve analysis is one of the principal statistical analysis methods, it is used by a wide range of the scientific community. Both commercial and free software tools are available for users to perform it. Generally, easy-to-use and nicely-designed interfaces are offered by commercial software packages whereas flexible and comprehensive tools are available in free, open-access, code-based software packages, such as R. The first novelty of our tool is that it allows the user to use free and open-access software with an easy-to-use interface. In other words, we combine the power of an open-source and free language with a nicely designed and easily accessible interface. This tool offers more comprehensive features and a wide variety of implementations for ROC curve analysis than its commercial and free counterparts, which is another novelty of this application. It is specifically constructed for ROC curve analysis, unlike the commercial software packages, such as IBM SPSS, Stata and MedCalc.

This web-based application is intended for research purposes only, not for clinical or commercial use. Since it is a non-profit service to the scientific community, it comes with no warranty and no data security. However, since this web server uses the R package **shiny**, each user performs his/her analyses in a new R session. After uploading data, the application only saves responses within its R session and prints the results instantly. After a user has quit the application, the corresponding R session will be closed and any uploaded data, responses or outputs will not be saved locally or remotely.

This tool is freely available through <http://www.biosoft.hacettepe.edu.tr/easyROC/> and all the source codes are available at <http://www.github.com/dncR/easyROC> under GPL version 3. It will

be regularly updated upon the dependent R packages used in this application, including **shiny** and **OptimalCutpoints**, and new features will be continually added as they are developed.

## Bibliography

- A. Agresti and B. A. Coull. Approximate is better than “exact” for interval estimation of binomial proportions. *The American Statistician*, 52(2):119–126, 1998. doi: 10.2307/2685469. [p218]
- D. Bamber. The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of Mathematical Psychology*, 12(4):387–415, 1975. doi: 10.1016/0022-2496(75)90001-2. [p215]
- V. Carey and H. Redestig. ROC: Utilities for ROC, with Uarray Focus, 2015. URL <https://www.bioconductor.org>. R package version 1.44.0. [p213]
- M. Celikbilek, M. Baskol, S. Taheri, K. Deniz, S. Dogan, G. Zararsiz, S. Gursoy, K. Guven, O. Ozbakir, M. Dundar, and M. Yucesoy. Circulating microRNAs in patients with non-alcoholic fatty liver disease. *World Journal of Hepatology*, 6(8):613–620, 2014. [p221]
- W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. shiny: Web Application Framework for R, 2015. URL <https://CRAN.R-project.org/package=shiny>. R package version 0.12.1. [p213]
- E. R. DeLong, D. M. DeLong, and D. L. Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics*, 44(3):837–845, 1988. doi: 10.2307/2531595. [p217, 220, 222]
- J. P. Egan. *Signal Detection Theory and ROC Analysis*. Series in Cognition and Perception. Academic Press, New York, NY, 1975. [p213]
- J. Fan, S. Upadhye, and A. Worster. Understanding receiver operating characteristic (ROC) curves. *Canadian Journal of Emergency Medicine*, 8:19–20, 2006. doi: 10.1017/s1481803500013336. [p213]
- T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. doi: 10.1016/j.patrec.2005.10.010. [p213]
- M. Gönen and G. Heller. Lehmann family of ROC curves. *Medical Decision Making*, 30(4):509–517, 2010. doi: 10.1177/0272989x09360067. [p214]
- J. A. Hanley and K. O. Hajian-Tilaki. Sampling variability of nonparametric estimates of the areas under receiver operating characteristic curves: An update. *Academic Radiology*, 4(1):49–58, 1997. doi: 10.1016/s1076-6332(97)80161-4. [p217]
- J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982. doi: 10.1148/radiology.143.1.7063747. [p215, 217]
- J. A. Hanley and B. J. McNeil. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843, 1983. doi: 10.1148/radiology.148.3.6878708. [p220]
- M. Lopez-Raton, M. X. Rodriguez-Alvarez, C. Cadarso-Suárez, and F. Gude-Sampedro. OptimalCutpoints: An R package for selecting optimal cutpoints in diagnostic tests. *Journal of Statistical Software*, 61(8):1–36, 2014. doi: 10.18637/jss.v061.i08. [p213, 216, 220, 221, 222]
- D. K. McClish. Analyzing a portion of the ROC curve. *Medical Decision Making*, 9(3):190–195, 1989. doi: 10.1177/0272989x8900900307. [p216]
- J. T. Morisette and S. Khorram. Exact binomial confidence interval for proportions. *Photogrammetric Engineering and Remote Sensing*, 64(4):281–283, 1998. [p218]
- P. A. Murtaugh, E. R. Dickson, G. M. Van Dam, M. Malinchoc, P. M. Grambsch, A. L. Langworthy, and C. H. Gips. Primary biliary cirrhosis: Prediction of short-term survival based on repeated patient visits. *Hepatology*, 20(1):126–134, 1994. doi: 10.1002/hep.1840200120. [p222]
- N. A. Obuchowski. Computing sample size for receiver operating characteristic studies. *Investigative Radiology*, 29(2):238–243, 1994. doi: 10.1097/00004424-199402000-00020. [p217]
- N. A. Obuchowski. Sample size calculations in studies of test accuracy. *Statistical Methods in Medical Research*, 7(4):371–392, 1998. [p218]

- N. A. Obuchowski, M. L. Lieber, and F. H. Wians. ROC curves in clinical chemistry: Uses, misuses, and possible solutions. *Clinical Chemistry*, 50(7):1118–1125, 2004. doi: 10.1373/clinchem.2004.031823. [p218, 219]
- M. Pepe, G. Longton, and H. Janes. Estimation and comparison of receiver operating characteristic curves. *The Stata Journal*, 9(1):1, 2009. [p220]
- X. Robin, N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.-C. Sanchez, and M. Müller. pROC: An open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics*, 12(1): 77, 2011. doi: 10.1186/1471-2105-12-77. [p213]
- M. C. Sachs. *plotROC: Generate Useful ROC Curve Charts for Print and Interactive Use*, 2016. URL <http://sachsmc.github.io/plotROC>. R package version 2.0.1. [p220]
- T. Sing, O. Sander, N. Beerenwinkel, and T. Lengauer. ROCR: Visualizing classifier performance in R. *Bioinformatics*, 21(20):3940–3941, 2005. [p213]
- P. Sonego, A. Kocsor, and S. Pongor. ROC analysis: Applications to the classification of biological sequences and 3D structures. *Briefings in Bioinformatics*, 9(3):198–209, 2008. [p213]
- J. A. Swets, R. M. Dawes, and J. Monahan. Better decisions through science. *Scientific American*, 283(4): 82–87, 2000. doi: 10.1038/scientificamerican1000-82. [p213]
- E. Venkatraman. A permutation test to compare receiver operating characteristic curves. *Biometrics*, 56 (4):1134–1138, 2000. doi: 10.1111/j.0006-341x.2000.01134.x. [p220]
- E. Venkatraman and C. B. Begg. A distribution-free procedure for comparing receiver operating characteristic curves from a paired experiment. *Biometrika*, 83(4):835–848, 1996. doi: 10.1093/biomet/83.4.835. [p220]
- 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/>. [p220]
- X.-H. Zhou, D. K. McClish, and N. A. Obuchowski. *Statistical Methods in Diagnostic Medicine*, volume 569. John Wiley & Sons, 2002. doi: 10.1002/9780470317082. [p214, 216, 219]

Dincer Goksuluk  
Department of Biostatistics, School of Medicine, Hacettepe University  
Sıhhiye Campus, 06100 - Ankara  
Turkey  
[dincer.goksuluk@hacettepe.edu.tr](mailto:dincer.goksuluk@hacettepe.edu.tr)

Selcuk Korkmaz  
Department of Biostatistics, School of Medicine, Hacettepe University  
Sıhhiye Campus, 06100 - Ankara  
Turkey  
[selcukkorkmaz@hotmail.com](mailto:selcukkorkmaz@hotmail.com)

Gokmen Zararsiz  
Department of Biostatistics, School of Medicine, Erciyes University  
Genome and Stem Cell Center, Erciyes University  
Talas, 38039 - Kayseri  
Turkey  
[gokmenzararsiz@hotmail.com](mailto:gokmenzararsiz@hotmail.com)

A. Ergun Karaagaoglu  
Department of Biostatistics, School of Medicine, Hacettepe University  
Sıhhiye Campus, 06100 - Ankara  
Turkey  
[ekaraaga@gmail.com](mailto:ekaraaga@gmail.com)

## Supplementary material

**Supplementary 1:** A brief description of optimal cutpoint methods.

Method	Description
Youden	Youden index identifies the cutpoint that maximizes the sum of <i>Sensitivity</i> and <i>Specificity</i> .
CB	CB is a measure based on the cost and benefit method, and is calculated from the slope of the ROC curve.
MinValueSp MinValueSe	For a given minimum value for <i>Specificity</i> , MinValueSp identifies the optimal value as the one that gives the maximum <i>Sensitivity</i> . In contrast, for a given minimum value for <i>Sensitivity</i> , MinValueSe identifies the optimal value as the one that gives the maximum <i>Specificity</i> .
ValueSp ValueSe	For a given particular value for <i>Specificity</i> , ValueSp identifies the optimal value as the one that gives the maximum <i>Sensitivity</i> . In contrast, for a given particular value for <i>Sensitivity</i> , ValueSe identifies the optimal value as the one that gives the maximum <i>Specificity</i> .
MinValueSpSe	For given minimum values for <i>Specificity</i> and <i>Sensitivity</i> measures, MinValueSpSe identifies the optimal value as the one that gives the maximum <i>Sensitivity</i> or <i>Specificity</i> (user-defined).
MaxSp MaxSe	MaxSp and MaxSe are two measures based on the maximization of <i>Specificity</i> and <i>Sensitivity</i> , respectively.
MaxSpSe	MaxSpSe is a measure based on the simultaneous maximization of both <i>Specificity</i> and <i>Sensitivity</i> measures.
MaxProdSpSe	MaxProdSpSe is a measure based on the maximization of the product of <i>Sensitivity</i> and <i>Specificity</i> .
ROC01	ROC01 identifies the optimal cutpoint that is closest to the upper-left corner (0,1) of the ROC graph.
SpEqualSe	SpEqualSe is a measure based on the minimization of the absolute difference between <i>Sensitivity</i> and <i>Specificity</i> .
MaxEfficiency	MaxEfficiency is a measure based on the minimization of the misclassification error, $(FP + FN)/n$ .

*Continued on next page*

**Supplementary 1 – Continued from previous page**

<b>Method</b>	<b>Description</b>
Minimax	Minimax is a measure based on the minimization of the most frequent error. Minimax is computed using the equation $\text{Minimax}_c = \min_c(\max(p(1 - Sensitivity) + (1 - p)(1 - Specificity)))$ where $c$ is the cutpoint and $p$ is the prevalence.
MaxDOR	MaxDOR is a measure based on the maximization of the diagnostic odds ratio, calculated using the equation $\text{MaxDOR}_c = \max_c[(Sensitivity \times Specificity) / ((1 - Sensitivity)(1 - Specificity))]$ .
MinValueNPV MinValuePPV	For a given minimum value for $NPV$ , MinValueNPV identifies the optimal value as the one that gives the maximum $PPV$ . In contrast, for a given minimum value for $PPV$ , MinValuePPV identifies the optimal value as the one that gives the maximum $NPV$ .
ValueNPV ValuePPV	For a given particular value for $NPV$ , ValueNPV identifies the optimal cutpoint as the one that gives the maximum $PPV$ . In contrast, for a given particular value for $PPV$ , ValuePPV identifies the optimal cutpoint as the one that gives the maximum $NPV$ .
MinValueNPVPPV	For given minimum values for predictive values, MinValueNPVPPV identifies the optimal value as the one that gives the maximum $NPV$ or $PPV$ (user-defined).
PROC01	PROC01 identifies the optimal cutpoint that is closest to the upper-left corner (0, 1) of the partial ROC (pROC) graph.
NPVEqualPPV	NPVEqualPPV is a measure based on the minimization of the absolute difference between $NPV$ and $PPV$ .
MaxNPVPPV	MaxNPVPPV is a measure based on the simultaneous maximization of both $NPV$ and $PPV$ measures.
MaxSumNPVPPV	MaxSumNPVPPV is a measure based on the maximization of the sum of $NPV$ and $PPV$ measures.
MaxProdNPVPPV	MaxProdNPVPPV is a measure based on the maximization of the product of $NPV$ and $PPV$ .
ValueDLR.Negative ValueDLR.Positive	These two measures are based on setting particular values for negative and positive diagnostic likelihood ratios, respectively.
MinPvalue	MinPvalue is a measure based on the minimization of the $p$ -value of the Chi-square test on assessing the independence between the diagnostic and gold standard test.

*Continued on next page*

**Supplementary 1 – Continued from previous page**

<b>Method</b>	<b>Description</b>
ObservedPrev	ObservedPrev is a measure which identifies the optimal cutpoint closest to the observed prevalence by minimizing the quantity $ c - p $ . This method is valid when the diagnostic test takes values within the interval $[0, 1]$ .
MeanPrev	MeanPrev is a measure which identifies the optimal cutpoint closest to the average of the diagnostic test values. It is suggested to use this measure if the diagnostic test takes values between 0 and 1.
PrevalenceMatching	PrevalenceMatching is a measure based on the equality of actual and predicted prevalence. The cutpoint minimizes the absolute quantity $ p(1 - Sensitivity) - (1 - p)(1 - Specificity) $ . This method is valid when the diagnostic test takes values within the interval $[0, 1]$ .

For details, see Lopez-Raton *et al.* (2014).

**Supplementary 2:** Non-alcoholic fatty liver disease (NAFLD) data (Çelikbilek et al., 2014).

Grup	mir197	mir146b	mir181d	mir99a	Grup	mir197	mir146b	mir181d	mir99a
1	0.921	0.687	0.474	-0.941	0	1.214	1.122	0.882	1.610
1	0.967	1.059	0.474	0.575	0	1.401	0.148	0.444	0.625
1	0.854	1.105	0.722	0.936	0	0.494	-0.179	1.386	0.134
1	-1.088	-1.353	-0.577	-1.077	0	1.608	1.386	2.242	0.926
1	0.107	0.515	-0.286	0.560	0	1.274	1.609	0.769	1.108
1	0.547	1.191	0.583	1.119	0	0.827	1.128	0.452	0.374
1	-1.081	-1.445	-1.303	-1.202	0	-0.147	-0.545	0.878	0.044
1	-1.081	-1.308	-1.276	-1.066	0	0.353	0.320	-0.225	0.367
1	0.841	0.463	-0.290	0.747	0	-1.635	-0.677	-0.838	-0.543
1	-1.188	-0.975	-1.407	-2.123	0	1.848	1.523	1.712	0.940
1	-1.014	-0.649	-1.194	-1.786	0	0.987	0.606	0.626	0.542
1	-1.081	-1.256	-1.229	-0.679	0	0.020	0.503	0.600	0.367
1	-1.295	-1.204	-1.607	-2.216	0	1.061	1.518	1.217	0.209
1	-1.081	-1.268	-0.829	-0.658	0	0.474	0.572	0.292	0.786
1	-1.081	-1.365	-1.376	-1.457	0	-0.868	-0.505	-0.408	-0.117
1	-1.081	-1.371	-0.812	-1.804	0	-0.414	-0.259	0.665	0.363
1	-1.081	-0.769	-1.359	-0.156	0	0.394	0.417	1.000	0.130
1	0.854	1.243	0.444	1.460	0	0.941	0.543	0.431	1.083
1	-1.074	-1.365	-1.572	-0.339	0	-0.387	-0.202	-0.568	0.345
1	-0.634	-0.276	-0.130	-0.081	0	-0.674	-0.689	0.995	0.893

# tigris: An R Package to Access and Work with Geographic Data from the US Census Bureau

by Kyle Walker

**Abstract** TIGER/Line shapefiles from the United States Census Bureau are commonly used for the mapping and analysis of US demographic trends. The **tigris** package provides a uniform interface for R users to download and work with these shapefiles. Functions in **tigris** allow R users to request Census geographic datasets using familiar geographic identifiers and return those datasets as objects of class "Spatial\*DataFrame". In turn, **tigris** ensures consistent and high-quality spatial data for R users' cartographic and spatial analysis projects that involve US Census data. This article provides an overview of the functionality of the **tigris** package, and concludes with an applied example of a geospatial workflow using data retrieved with **tigris**.

## Introduction

Analysis and visualization of geographic data are often core components of the analytical workflow for researchers and data scientists; as such, access to open and reliable geographic datasets are of paramount importance. The United States Census Bureau provides access to such data in the form of its TIGER/Line shapefile products ([United States Census Bureau, 2016a](#)). The files are extracts from the Census Bureau's Master Address File/Topologically Integrated Geographic Encoding and Referencing (TIGER) database, which in turn are released to the public as shapefiles, a common format for encoding geographic data as vectors (e.g. points, lines, and polygons). Available TIGER/Line shapefiles include all of the Census Bureau's areal enumeration units, such as states, counties, Census tracts, and Census blocks; transportation data such as roads and railways; and both linear and areal hydrography. The TIGER/Line files are updated annually, and include attributes that allow them to be joined with other tabular data, including demographic data products released by the Census Bureau.

The **tigris** package aims to simplify the process of working with these datasets for R users ([Walker, 2016](#)). With functions in **tigris**, R users can specify the data type and geography for which they would like to obtain geographic data, and return the corresponding TIGER/Line data as an R object of class "Spatial\*DataFrame". This article provides an overview of the **tigris** package, and gives examples that show how it can contribute to common geographic visualization and spatial analysis workflows in R. Examples in the article include a discussion of how **tigris** helps R users retrieve and work with data from the US Census Bureau, as well as an applied example of how **tigris** can fit within a common geospatial workflow in R, in which data from the United States Internal Revenue Service are visualized with both static and interactive cartography.

## Geographic data and Census visualization in R

The TIGER/Line files were first released by the US Census Bureau in ASCII format in 1989, and represented street centerline data for the entire United States. Since then, the Census Bureau has expanded the coverage of the TIGER/Line data, and transitioned the core format of the publicly-available files to the shapefile in 2007. TIGER/Line shapefiles include *boundary files*, which encompass the boundaries of governmental units or other areal units for which the Census Bureau tabulates data. This includes the core Census hierarchy of areal units from the Census block (analogous to a city block) to the entire United States, as well as common geographic entities such as city boundaries. The Census Bureau distinguishes between *legal entities*, which have official government standing, and *statistical entities*, which have no legal definition but are used for the tabulation of data. The Census Bureau also makes available shapefiles of *geographic features*, which include entities such as roads, rivers, and railroads. All TIGER/Line shapefiles are distributed in a geographic coordinate system using the North American Datum of 1983 (NAD83) ([United States Census Bureau, 2014](#)).

As the TIGER/Line datasets are available in shapefile format, they can be read into and translated to R objects by the **rgdal** package ([Bivand et al., 2015](#)). **rgdal** is an R interface to the open-sourced Geospatial Data Abstraction Library, or GDAL, an open-source translator that can convert between numerous common vector and raster spatial data formats ([GDAL Development Team, 2015](#)). When loaded into R, shapefiles will be represented as objects of class "Spatial\*" by the **sp** package ([Bivand et al., 2013](#)). Most Census datasets obtained by **tigris** will be loaded as objects of class

"`SpatialPolygonsDataFrame`" given that they represent Census areal entities; selected geographic features, such as roads, linear water features, and landmarks, may be represented as objects of class "`SpatialLinesDataFrame`" or "`SpatialPointsDataFrame`". "`Spatial*DataFrames`" are R objects that represent spatial data as closely as possible to regular R data frames, yet also contain information about the feature geometry and coordinate system of the data (see [Bivand et al., 2013](#), for more information).

Several R packages provide access to Census geographic and demographic data. The `UScensus2000` (no longer on CRAN) and `UScensus2010` packages by Zack Almquist allow for access to several geographic datasets for the 2000 and 2010 Censuses, including blocks, Census tracts, and counties ([Almquist, 2010](#)). These datasets can also be linked to demographic data from the 2000 and 2010 Censuses, which are stored in related, external packages. The `USAboundaries` package similarly provides access to some Census geographic boundary files such as zip code tabulation areas (ZCTAs) and counties; it also makes available historical boundary files dating back to 1629 ([Mullen, 2015](#)). Another R package, `choroplethr`, wraps `ggplot2` ([Wickham, 2009](#)) to map data from the US Census Bureau's American Community Survey aggregated to common Census geographies ([Lamstein and Johnson, 2015](#)). The purpose of the `tigris` package is to help R users work with US Census Bureau geographic data by granting direct access to the Census shapefiles via a simple, uniform interface. Further, as `tigris` interfaces directly with Census Bureau data stores, it ensures access to high-quality and up-to-date geographic data for R projects.

## Core functionality of `tigris`

The core functionality of `tigris` consists of a series of functions, each corresponding to a single Census Bureau geography of interest, that grant access to geographic data from the US Census Bureau. `tigris` allows R users to obtain both the core TIGER/Line shapefiles as well as the Census Bureau's Cartographic Boundary Files. Cartographic Boundary Files, following the [United States Census Bureau \(2015\)](#), "are simplified representations of selected geographic areas from the Census Bureau's MAF/TIGER geographic database" ([United States Census Bureau, 2015](#)).

To download geographic data using `tigris`, the R user calls the function corresponding to the desired geography. For example, to obtain a "`SpatialPolygonsDataFrame`" of US states from the TIGER/Line dataset, the user calls the `states()` function in `tigris`, which can then be plotted with the `plot()` function from the `sp` package, which is loaded by `tigris` automatically:

```
library(tigris)
us_states <- states()
plot(us_states)
```



**Figure 1:** Basic plot of US states retrieved from the Census TIGER/Line database.

The `states()` function call instructs **tigris** to fetch a TIGER/Line shapefile from the US Census Bureau that represents the boundaries of the 50 US states, the District of Columbia, and US territories. **tigris** then uses **rgdal** to load the data into the user's R session as an object of class "Spatial\*DataFrame". Many functions in **tigris** have a parameter, `cb`, that if set to TRUE will direct **tigris** to load a cartographic boundary file instead. Cartographic boundary files default to a simplified resolution of 1:500,000; in some cases, as with states, resolutions of 1:5 million and 1:20 million are available. For example, an R user could specify the following modifications to the `states()` function, and retrieve a simplified dataset.

```
us_states_20m <- states(cb = TRUE, resolution = "20m")
ri <- us_states[us_states$NAME == "Rhode Island", ]
ri_20m <- us_states_20m[us_states_20m$NAME == "Rhode Island", ]
plot(ri)
plot(ri_20m, border = "red", add = TRUE)
```



**Figure 2:** Difference between default TIGER/Line and 1:20 million cartographic boundary outlines of Rhode Island.

The plot illustrates some of the differences between the TIGER/Line shapefiles and the cartographic boundary files, in this instance for the state of Rhode Island. The TIGER/Line shapefiles are the most detailed datasets in interior areas, and represent the legal boundaries for coastal areas which extend three miles beyond the shoreline. The Cartographic Boundary Files have less detail in interior areas, but are clipped to the shoreline of the United States, which may be preferable for thematic mapping but can introduce additional detail for coastal features. A full list of the geographic datasets available through **tigris** is found in Table 1; datasets with an asterisk are available as both TIGER/Line and cartographic boundary files.

When Census data are available for download at sub-national levels, they are referenced by their Federal Information Processing Standard (FIPS) codes, which are codes that uniquely identify geographic entities in the Census database. When applicable, **tigris** uses smart state and county lookup to simplify the process of data acquisition for R users. This allows users to obtain data by supplying the name or postal code of the desired state – along with the name of the desired county, when applicable – rather than their FIPS codes. In the following example, the R user fetches roads data for Kalawao County Hawaii, the smallest county in the United States by area, located on the northern coast of the island of Moloka'i.

```
kw_roads <- roads("HI", "Kalawao")
plot(kw_roads)
```

While many Census shapefiles correspond to these common geographic identifiers in the United States, not all datasets are identifiable in this way. A good example is the Zip Code Tabulation

Family	Functions
General area functions	nation*; divisions*; regions*; states*; counties*; tracts*; block_groups*; blocks; places*; pumas*; school_districts; county_subdivisions*; zctas* congressional_districts*; state_legislative_districts*; voting_districts (2012 only)
Legislative district functions	
Water functions	area_water; linear_water; coastline
Metro area functions	core_based_statistical_areas*; combined_statistical_areas*; metro_divisions; new_england*; urban_areas*
Transportation functions	primary_roads; primary_secondary_roads; roads; rails
Native/tribal geometries functions	native_areas*; alaska_nativeRegional_corporations*; tribal_block_groups; tribal_census_tracts; tribal_subdivisions_national
Other	landmarks; military

**Table 1:** Functions available in the tigris package. Functions denoted with an asterisk are also available as cartographic boundary files.

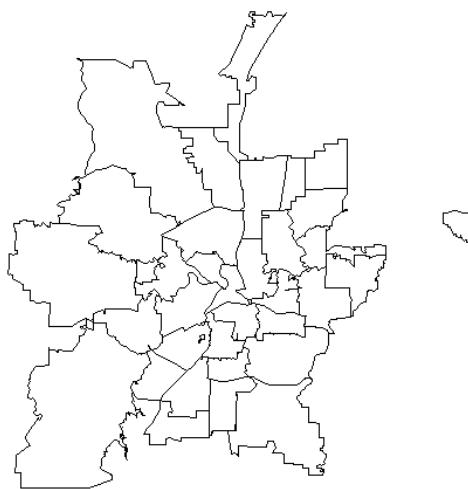


**Figure 3:** Roads in Kalawao County, Hawaii.

Area (ZCTA), a geographic dataset developed by the US Census Bureau to approximate zip codes, postal codes used by the United States Postal Service. Social data in the United States are commonly distributed at the zip code level, including an example later in this article; however, zip codes themselves are not coherent geographic entities, and change frequently. ZCTAs, then, function as proxies for zip codes, and are built from Census blocks in which a plurality of addresses on a given block have a given zip code ([United States Census Bureau, 2016b](#)).

ZCTAs commonly cross county lines and even cross state lines in certain instances; as such, the US Census Bureau only makes the entire ZCTA dataset of over 33,000 zip codes available for download. Often, analysts will not need all of these ZCTAs for a given project. **tigris** allows users to subset ZCTAs on load with the `starts_with` parameter, which accepts a vector of strings that contains the beginning digits of the ZCTAs that the analyst wants to load into R. The example below retrieves ZCTAs in the area around Fort Worth, Texas.

```
fw_zips <- zctas(cb = TRUE, starts_with = "761")
plot(fw_zips)
```



**Figure 4:** Zip Code Tabulation Areas that start with "761" (near Fort Worth, Texas).

When **tigris** downloads Census shapefiles to the R user's computer, it uses the `rappdirs` package to cache the downloads for future access ([Ratnakumar et al., 2014](#)). In turn, once the R user has downloaded the Census geographic data, **tigris** will know where to look for it and will not need to re-download. To turn off this behavior, a **tigris** user can set `options(tigris_use_cache = FALSE)` after loading the package; this will direct **tigris** to download shapefiles to a temporary directory on the user's computer instead, and load data into R from there.

The Census Bureau releases updated TIGER/Line shapefiles every year, and these yearly updates are available to **tigris** users. **tigris** defaults to the 2015 shapefiles, which at the time of this writing is the most recent year available.. However, **tigris** users can supply a different year to a **tigris** function as a named argument to obtain data for a different year; for example, `year = 2014` in the function call will fetch TIGER/Line shapefiles or cartographic boundary files from 2014. Additionally, R users can set this as a global option in their R session by entering the command `options(tigris_year = 2014)`.

## Data manipulation with **tigris**

The primary utility of the **tigris** package is for consistent and quick data access for R users with a minimum of code. As **tigris** loads objects of class "Spatial\*" from the `sp` package, data analysis and visualization using the acquired data can be handled with R's suite of cartographic and spatial analysis packages, which will be addressed later in the article. However, **tigris** does include two

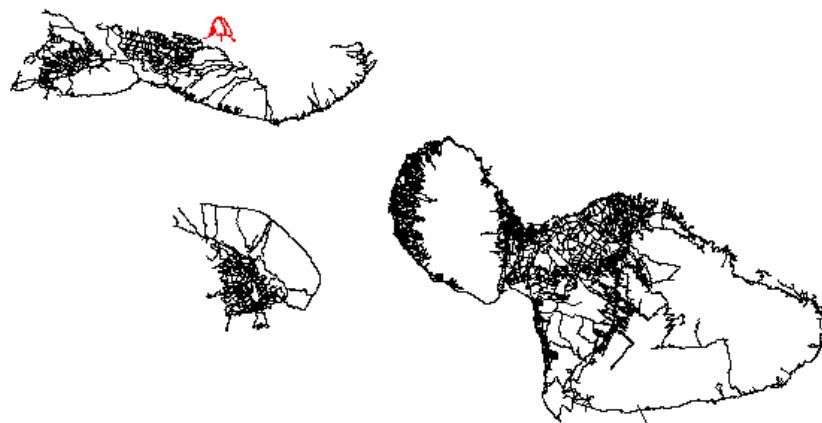
functions, `rbind_tigris()` and `geo_join()`, to assist with common operations when working with Census Bureau geographic data: combining datasets with one another, or merging them to tabular data.

Some Census shapefiles, like roads, are only available by county; however, an R user may want a roads dataset that represents multiple counties. `tigris` has built-in functionality to handle these circumstances. Data loaded into R by `tigris` functions are assigned a special "tigris" attribute that identifies the type of geographic data represented by the object. This attribute can be checked with the function `tigris_type()`:

```
> tigris_type(kw_roads)
[1] "road"
```

Objects with the same "tigris" attributes can then be combined into a single object using the function `rbind_tigris()`. In the example below, the user loads data for Maui County, which comprises the remainder of the island of Moloka'i as well as the islands of Maui and Lanai. As the roads data do not explicitly contain information about counties, an identifying "county" column is specified in the example below. The data are then plotted and colored by county; Kalawao County is colored red in the figure.

```
maui_roads <- roads("HI", "Maui")
kw_roads$county <- "Kalawao"
maui_roads$county <- "Maui"
maui_kw_roads <- rbind_tigris(kw_roads, maui_roads)
plot(maui_kw_roads, col = c("red", "black")[as.factor(maui_kw_roads$county)])
```

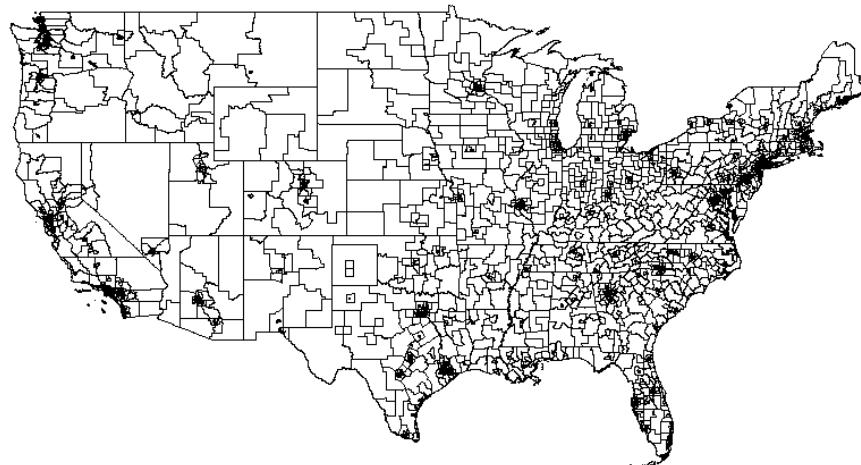


**Figure 5:** Roads in Maui County and Kalawao County, Hawaii.

The `rbind_tigris()` function also accepts a list of `sp` objects with the same "tigris" attributes. This is particularly useful in the event that an analyst needs a dataset that covers the United States, but is only available at sub-national levels. An example of this is the Public Use Microdata Area (PUMA), the Census geography at which individual-level microdata samples are associated. PUMAs are available by state in `tigris` via the `pumas()` function; however, an analyst will commonly want PUMA geography for the entire United States to facilitate national-level analyses. In this example, `rbind_tigris()` can be used with `lapply()` to fetch PUMA datasets for each state and then combine them into a dataset covering the continental United States.

**tigris** includes a built-in data frame named `fips_codes` that used to match state and county names with Census FIPS codes in the various functions in the package. It can also be used to generate vectors of codes to be passed to **tigris** functions as in this example, so that the analyst does not have to generate the full list of state codes by hand.

```
us_states <- unique(fips_codes$state)[1:51]
continental_states <- us_states[!us_states %in% c("AK", "HI")]
us_pumas <- rbind_tigris(
  lapply(
    continental_states, function(x) {
      pumas(state = x, cb = TRUE)
    }
  )
)
plot(us_pumas)
```



**Figure 6:** Public Use Microdata Areas (PUMAs) for the continental United States, generated with `rbind_tigris`.

The above code directs R to iterate through the state codes for the continental United States, fetching PUMA geography for each state and storing it in a list which `rbind_tigris()` then combines into a continental PUMA dataset.

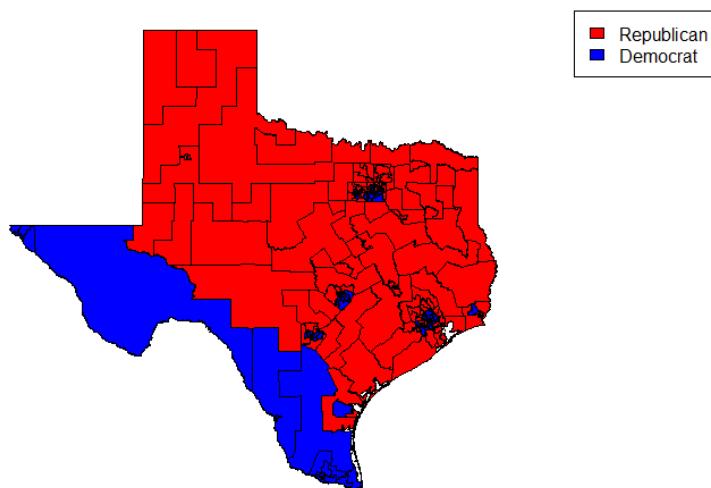
The other data manipulation function in **tigris**, `geo_join()`, is designed to assist with the common but sometimes-messy process of merging tabular data to US Census Bureau shapefiles. Such joined data can then be used for statistical mapping, such as a choropleth map that shows variation in an attribute by the shading of polygons.

In the example below, an analyst uses functions in **tigris** to help create a choropleth map that shows how the areas represented by legislators in the Texas State House of Representatives, the lower house of the Texas state legislature, vary by political party. By convention in the United States, areas represented by members of the Republican Party are shaded in red, and areas represented by members of the Democratic Party are shaded in blue.

To accomplish this, the analyst loads in a CSV containing information on party representation in Texas by legislative district, and uses the `state_legislative_districts()` function to retrieve boundaries for the legislative districts. The two datasets can then be joined with `geo_join()`. The first argument in the `geo_join()` call represents the object of class "Spatial\*DataFrame"; the second argument represents a regular R data frame. The third and fourth arguments specify the columns in the spatial data frame and regular data frame, respectively, to be used to match the rows; if the names of these columns are the same, that name can be passed as a named argument to the `by` parameter,

which is unused here. Once the two datasets are joined, they can be visualized with `sp` plotting functions.

```
df <- read.csv("http://personal.tcu.edu/kylewalker/data/txlege.csv",
               stringsAsFactors = FALSE)
districts <- state_legislative_districts("TX", house = "lower", cb = TRUE)
txlege <- geo_join(districts, df, "NAME", "District")
txlege$color <- ifelse(txlege$Party == "R", "red", "blue")
plot(txlege, col = txlege$color)
legend("topright", legend = c("Republican", "Democrat"),
       fill = c("red", "blue"))
```



**Figure 7:** State legislative districts for the Texas State House of Representatives, colored by the party affiliations of their representatives. Data derived from *The Texas Tribune*, <https://www.texastribune.org/directory/>

As the plot illustrates, Democrats in Texas tend to represent areas in and around major cities such as Dallas, Houston, and Austin, as well as areas along the United States-Mexico border. Republicans, on the other hand, tend to represent rural areas in addition to suburban areas on the edges of metropolitan areas in the state.

## Analytic visualization in R using data obtained with `tigris`

To this point, this paper has employed simplified examples to demonstrate the functionality of `tigris`; the following scenario combines these examples into an applied analytic and visualization workflow. The goal here is to show how `tigris` fits in with a broader spatial analysis workflow in R. R has a plethora of packages available for geographic visualization and spatial analysis; for analysts working with United States geographies, `tigris` can contribute to the analytic process by providing ample data access with a minimum of code, and without having to retrieve datasets outside of R.

This example demonstrates how to create metropolitan area maps of taxation data from the United States Internal Revenue Service (IRS), which are made available at the zip code level ([United States Internal Revenue Service, 2015](#)). As discussed earlier, zip codes are not physical areas but rather designations given by the United States Postal Service (USPS) to guide mail routes; as such, ZCTAs will be used instead, and accessed with the `zctas` function.

As ZCTAs do not have a clear correspondence between their boundaries and those of other Census units, ZCTA boundaries will commonly cross those of metropolitan areas, which are county-based. However, `tigris` provides programmatic access to metropolitan area boundaries as well, which in turn

can be used to identify intersecting ZCTAs through spatial overlay with the **sp** package. The resultant spatial data can then be merged to data from the IRS and visualized.

Such a workflow could resemble the following. An analyst reads in raw data from the IRS website as an R data frame, and uses the **dplyr** package (Wickham and Francois, 2015) to subset the data frame and identify the average total income reported to the IRS by zip code in thousands of dollars in 2013, assigning it to the variable `df`. In the original IRS dataset, `A02650` represents the aggregate total income reported to the IRS by zip code in thousands of dollars, and `N02650` represents the number of tax returns that reported total income in that zip code.

```
library(dplyr)
library(stringr)
library(readr)

# Read in the IRS data
zip_data <- "https://www.irs.gov/pub/irs-soi/13zpallnoagi.csv"
df <- read_csv(zip_data) %>%
  mutate(zip_str = str_pad(as.character(ZIPCODE), width = 5,
                          side = "left", pad = "0"),
        incpr = A02650 / N02650) %>%
  select(zip_str, incpr)
```

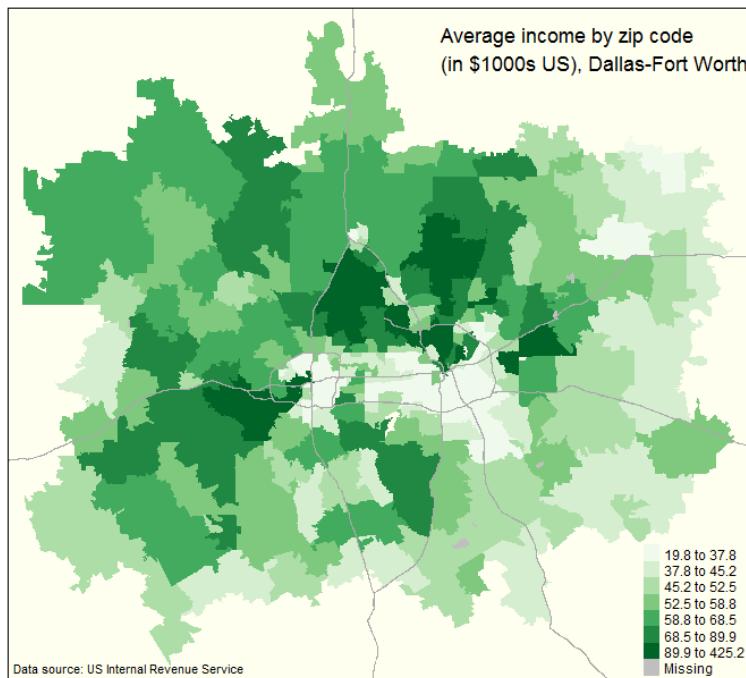
The analyst then defines a function that will leverage **tigris** to read in Census ZCTA and metropolitan area datasets as objects of class "`SpatialPolygonsDataFrame`", and return the ZCTAs that intersect a given metropolitan area as defined by the analyst.

```
library(tigris)
library(sp)

# Write function to get ZCTAs for a given metro
get_zips <- function(metro_name) {
  zips <- zctas(cb = TRUE)
  metros <- core_based_statistical_areas(cb = TRUE)
  # Subset for specific metro area
  # (be careful with duplicate cities like "Washington")
  my_metro <- metros[grepl(sprintf("^%s", metro_name),
                           metros$NAME, ignore.case = TRUE), ]
  # Find all ZCTAs that intersect the metro boundary
  metro_zips <- over(my_metro, zips, returnList = TRUE)[[1]]
  my_zips <- zips[zips$ZCTA5CE10 %in% metro_zips$ZCTA5CE10, ]
  # Return those ZCTAs
  return(my_zips)
}
```

The analyst can then fetch ZCTA geography for a given metropolitan area, which in this example will be Dallas-Fort Worth, Texas, and merge the IRS income data to it with `geo_join()`. For visualization, this example uses the **tmap** package (Tennekes, 2015), an excellent option for creating high-quality cartographic products in R, to create a choropleth map. To provide spatial reference to the Census tracts on the map, major roads obtained with the `primary_roads()` function in **tigris** are added to the map as well.

```
library(tmap)
rds <- primary_roads()
dfw <- get_zips("Dallas")
dfw_merged <- geo_join(dfw, df, "ZCTA5CE10", "zip_str")
tm_shape(dfw_merged, projection = "+init=epsg:26914") +
  tm_fill("incpr", style = "quantile", n = 7, palette = "Greens", title = "") +
  tm_shape(rds, projection = "+init=epsg:26914") +
  tm_lines(col = "darkgrey") +
  tm_layout(bg.color = "ivory",
            title = "Average income by zip code \n(in $1000s US), Dallas-Fort Worth",
            title.position = c("right", "top"), title.size = 1.1,
            legend.position = c(0.85, 0), legend.text.size = 0.75,
            legend.width = 0.2) +
  tm_credits("Data source: US Internal Revenue Service",
             position = c(0.002, 0.002))
```



**Figure 8:** Map of average reported total income to the Internal Revenue Service by zip code for the Dallas-Fort Worth, Texas metropolitan area in 2013, created with `tmap`.

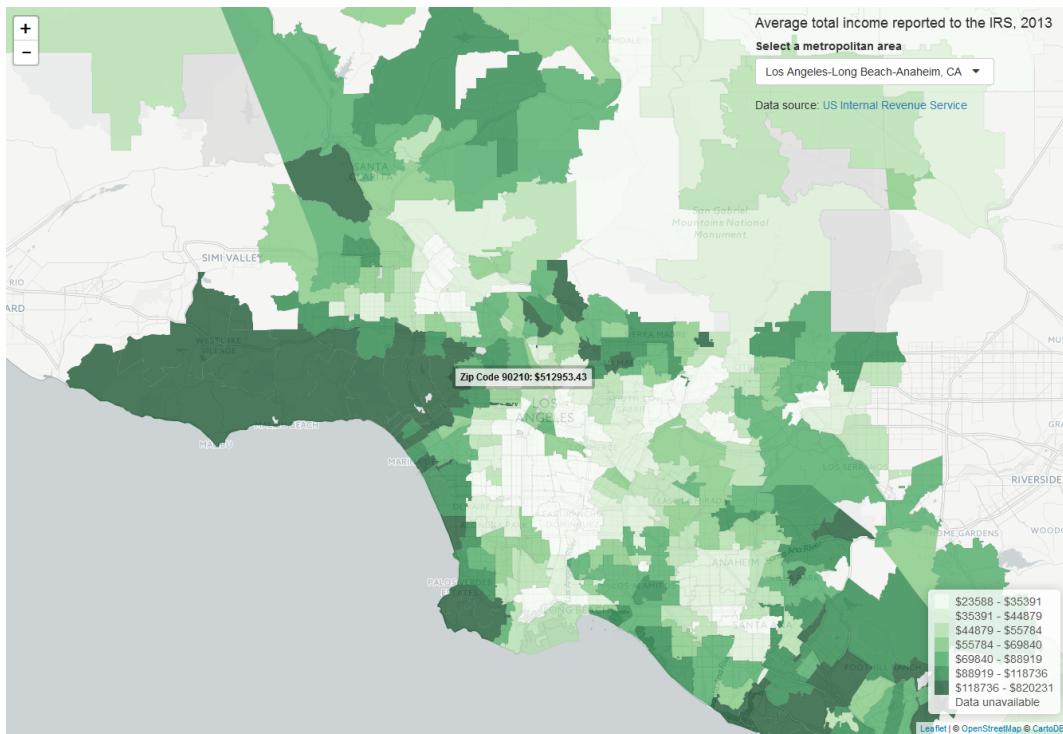
Given that both geographic data obtained through `tigris` and the IRS data are available for the entire country, an R developer could extend this example and create a web application that generates interactive income maps based on user input with the `shiny` package (Chang et al., 2016). Below is an example of such an application, which is viewable online at <http://walkerke.shinyapps.io/tigris-zip-income>; the code for the application can be viewed at <http://github.com/walkerke/tigris-zip-income>.

In the application, the user selects a metropolitan area from the drop-down menu, instructing the Shiny server to generate an interactive choropleth map of average reported total income by zip code from the IRS, as in the above example, but in this instance using the `leaflet` package (Cheng and Xie, 2015). The application uses the same process described above for the static map to subset the data; in this instance, however, the Shiny server makes these computations on-the-fly in response to user input. In the figure, the Los Angeles, California metropolitan area is selected; however, all metropolitan areas in the United States are available to users of the application. While both of these cartographic examples require considerable R infrastructure to process the data and ultimately create the visualizations, `tigris` plays a key role in each by providing direct access to reliable spatial data programmatically from R.

## Conclusion

This paper has summarized the functionality of the `tigris` package for retrieving and working with shapefiles from the United States Census Bureau. Access to high-quality spatial data is essential for the geospatial analyst, but can be difficult to access. For R users working on projects that can benefit from United States Census Bureau data, `tigris` provides direct access to the Census Bureau's TIGER/Line and cartographic boundary files using a simple and consistent API. In turn, tasks such as looking up FIPS codes to identify the correct datasets to download or combining several Census datasets are reduced to a few lines of R code in `tigris`.

More significantly, the utility of `tigris` is exemplified when included in a larger geospatial project that incorporates US Census data, such as the static and interactive maps of IRS data included in this article. These examples illustrate some clear advantages that R has over traditional desktop Geographic Information Systems software for geographic analysis and visualization. To create an interactive application showing IRS data by zip code as in Figure 9, a GIS analyst would traditionally have to search out and download the data from the web; load it into a desktop GIS for merging and calculating new columns; publish the data to a server; and build the application with a web mapping client and web application framework, often requiring several software applications. As shown in



**Figure 9:** Interactive Leaflet map of average reported total income to the Internal Revenue Service by zip code for the Los Angeles, California metropolitan area in 2013, built in Shiny.

this article, this entire process now can take place inside of an R script, helping ensure quality and reproducibility. For projects that require US Census Bureau geographic data, **tigris** aims to fit well within these types of workflows.

## Acknowledgments

I am indebted to Bob Rudis, whose key contributions to the **tigris** package were essential in improving its functionality. I would also like to thank Eli Knaap, who encouraged me to write the package; Hadley Wickham for writing *R Packages* and the **devtools** package (Wickham, 2015); (Wickham and Chang, 2015), which helped me significantly as I developed **tigris**; and Roger Bivand as well as the three anonymous reviewers for providing useful feedback on the manuscript.

## Bibliography

- Z. Almquist. US Census spatial and demographic data in R: The UScensus2000 suite of packages. *Journal of Statistical Software*, 37(6):1–31, 2010. [p232]
- R. Bivand, T. Keitt, and B. Rowlingson. *rgdal: Bindings for the Geospatial Data Abstraction Library*, 2015. URL <https://CRAN.R-project.org/package=rgdal>. R package version 1.1-1. [p231]
- R. S. Bivand, E. Pebesma, and V. Gomez-Rubio. *Applied spatial data analysis with R, Second edition*. Springer, New York, 2013. URL <http://www.asdar-book.org/>. [p231, 232]
- W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2016. URL <https://CRAN.R-project.org/package=shiny>. R package version 0.13.2. [p240]
- J. Cheng and Y. Xie. *leaflet: Create Interactive Web Maps with the JavaScript ‘Leaflet’ Library*, 2015. URL <http://rstudio.github.io/leaflet/>. R package version 1.0.1.9002. [p240]
- GDAL Development Team. *GDAL - Geospatial Data Abstraction Library, Version 2.0.1*. Open Source Geospatial Foundation, 2015. URL <http://www.gdal.org>. [p231]
- A. Lamstein and B. P. Johnson. *choroplethr: Simplify the Creation of Choropleth Maps in R*, 2015. URL <https://CRAN.R-project.org/package=choroplethr>. R package version 3.3.0. [p232]

- L. Mullen. *USAboundaries: Historical and Contemporary Boundaries of the United States of America*, 2015. URL <https://github.com/ropensci/USAboundaries>. R package version 0.1.1.9001. [p232]
- S. Ratnakumar, T. Mick, and T. Davis. *rappdirs: Application directories: determine where to save data, caches and logs.*, 2014. URL <https://CRAN.R-project.org/package=rappdirs>. R package version 0.3. [p235]
- M. Tennekes. *tmap: Thematic Maps*, 2015. URL <https://CRAN.R-project.org/package=tmap>. R package version 1.0. [p239]
- United States Census Bureau. TIGER/Line shapefiles: Technical documentation. Report, 2014. [p231]
- United States Census Bureau. Cartographic boundary file description, 2015. URL [https://www.census.gov/geo/maps-data/data/cbf/cbf\\_description.html](https://www.census.gov/geo/maps-data/data/cbf/cbf_description.html). Accessed: 2016-02-27. [p232]
- United States Census Bureau. Tiger/line® shapefiles and tiger/line® files, 2016a. URL <https://www.census.gov/geo/maps-data/data/tiger-line.html>. Accessed: 2016-05-22. [p231]
- United States Census Bureau. Zip code™ tabulation areas (zctas™), 2016b. URL <https://www.census.gov/geo/reference/zctas.html>. Accessed: 2016-05-22. [p235]
- United States Internal Revenue Service. Soi tax stats - individual income tax statistics - 2013 zip code data (soi), 2015. URL <https://www.irs.gov/uac/soi-tax-stats-individual-income-tax-statistics-2013-zip-code-data-soi>. Accessed: 2016-05-19. [p238]
- K. Walker. *tigris: Load Census TIGER/Line Shapefiles into R*, 2016. R package version 0.3. [p231]
- H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. ISBN 978-0-387-98140-6. URL <http://had.co.nz/ggplot2/book>. [p232]
- H. Wickham. *R Packages*. O'Reilly Media, Sebastopol, CA, 2015. ISBN 978-1491910597. URL <http://r-pkgs.had.co.nz/>. [p241]
- H. Wickham and W. Chang. *devtools: Tools to Make Developing R Packages Easier*, 2015. URL <https://CRAN.R-project.org/package=devtools>. R package version 1.8.0. [p241]
- H. Wickham and R. Francois. *dplyr: A Grammar of Data Manipulation*, 2015. URL <https://CRAN.R-project.org/package=dplyr>. R package version 0.4.3. [p239]

*Kyle Walker  
Texas Christian University  
2850 S University Dr  
Fort Worth, TX 76109  
kyle.walker@tcu.edu*

# Escape from Boxland

## Generating a Library of High-Dimensional Geometric Shapes

by Barret Schloerke, Hadley Wickham, Dianne Cook, and Heike Hofmann

**Abstract** A library of common geometric shapes can be used to train our brains for understanding data structure in high-dimensional Euclidean space. This article describes the methods for producing cubes, spheres, simplexes, and tori in multiple dimensions. It also describes new ways to define and generate high-dimensional tori. The algorithms are described, critical code chunks are given, and a large collection of generated data are provided. These are available in the R package **geozoo**, and selected movies and images, are available on the GeoZoo web site (<http://schloerke.github.io/geozoo/>).

## Introduction

This paper describes how to build a library of high-dimensional geometric shapes: cubes, spheres, simplexes, and tori. Data describing numerous 4D polytopes and polyhedra generated by other researchers are included in the library, a single location to describe the many different object structures. The purpose is to enable people to train their brains for understanding data structures residing in high-dimensional Euclidean space. This work extends the work described in Cook (1997) which concentrated on samples from statistical distributions.

The **geozoo** package in R contains the code to create the geometric shapes. Code fragments, describing the key components of the algorithms for generating the shapes, are included in this paper. The shapes in the library are best viewed using the dynamic graphical method called a tour (Asimov, 1985; Buja et al., 2005; Cook et al., 2007), such as that available in GGobi (Swayne et al., 2003) and the **tourr** R package (Wickham et al., 2011).

The structure of the paper is that basic shapes are described first followed by more complex shapes, in this order, cubes, spheres, simplexes, polyhedra, polytopes, and tori.

## Cubes

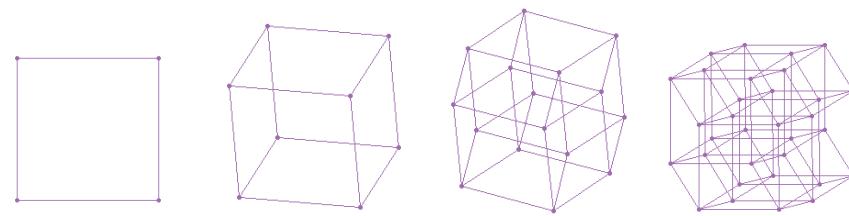
Cubes are the first shape that a person should examine when starting to learn about higher dimensions. Cubes are relatively simple to understand: they have orthogonal, uniform length sides and are convex shapes. A 0-D cube is a single point. A 1-D cube is a line segment. A 2-D cube is a square and a 3-D cube is a box.

The 4-D cube may be hard to imagine, partly because we are accustomed to describing our physical world using only three dimensions. The leap to 4-D is more understandable after watching the movie “Flatland” (Martin, 1965) or reading the novella of the same name (Abbott, 1884). In “Flatland”, the world is 2-D and characters struggle with the concept of 3-D.

Working from this name, we might think of our world as “Boxland”: we live in 3-D and struggle with the concept of higher dimensions. Shadows created by light sources help perceive the third dimension. In Flatland, the inhabitants see only 1-D line segments. For a Flatlander who has never seen the world we live in, the third dimension is hard to understand. Similarly, for inhabitants of our world, it might seem daunting to imagine the fourth dimension. But it’s not that difficult!

Figure 1 shows the evolution of the cube from 2-D to 5-D. Each figure is a 2-D projection of a wireframe cube from two to five dimensions. To increase the dimension of a cube, replicate and shift a cube of one dimension lower along a new orthogonal axis, connecting the corresponding vertices. The 3-D cube grows from  $2 \times 2$ -D squares, connected with four new edges. The 4-D cube is born from  $2 \times 3$ -D cubes and a 5-D cube emerges from  $2 \times 4$ -D cubes. Any object with more than 2 dimensions has infinitely many projections onto a 2-D plane. The projections chosen in Figure 1 (and the remaining figures of the paper) were done to highlight the overall structure or features that make each object distinct from each other.

Vertices of a high-dimensional cube can be considered as all permutations of the binary digits (0 and 1) in  $p$ -D. A line is defined by two points: (0), (1). A square is defined by four points: (0, 0), (0, 1), (1, 0), (1, 1), which are all of the permutations of 0 and 1 in two columns, that is, the Cartesian product of two lines. A 3-D cube is the Cartesian product of two squares, and has all of the permutations of 0 and 1 in three columns.



**Figure 1:** Wireframe cubes, (from left to right) 2-D, 3-D, 4-D, 5-D.

### Points on vertices

The two different ways to define a high-dimensional cubes leads to two different methods to create a  $p$ -D cube. Both methods yield the same result, which is shown for  $p = 1, 2, 3$  in Table 1.

1-D:			3-D:					
row #	1	edges	row #	1	2	3	edges	
1	0	2	1	0	0	0	2 3 5	
2	1		2	1	0	0	4 6	
2-D:			3-D:					
row #	1	2	edges	row #	1	2	3	edges
1	0	0	2 3	1	0	0	0	2 3 5
2	1	0	4	2	1	0	0	4 6
3	0	1	4	3	0	1	0	4 7
4	1	1		4	1	1	0	8

**Table 1:** 1-D, 2-D and 3-D cube vertices and edges.

- **Method 1:** Recursively double a lower-dimensional cube.

Using the standard coordinate system, the base is 0 and 1. After establishing the base, we recursively double the base in the first column(s), and add an additional column containing a 0 in the first half of the rows and a 1 in the second half of the rows. The process is repeated  $(p - 1)$  times, to obtain a  $p$ -D cube.

```
cube_iterate <- function(p) {
  if (p == 1) {
    return(rbind(0, 1))
  }
  lower_dim_cube <- cube_iterate(p - 1)
  rbind(
    cbind(lower_dim_cube, 0),
    cbind(lower_dim_cube, 1)
  )
}
```

- **Method 2:** Generate all permutations of 0,1 in  $p$  columns.

This method takes advantage of an existing function in R, `expand.grid`. It produces all permutations by generating the Cartesian product of a set of vectors. For our purposes, the number of columns is not fixed, so we use `do.call`, to convert a function call of the form `x(a,b,c)` to `do.call(x, list(a,b,c))`, allowing specification of an arbitrary number of arguments.

```
cube_permute <- function(p) {
  as.matrix(
    do.call(
      expand.grid,
      rep(list(c(0, 1)), p)
    )
  )
}
```

## Completing the wire frame

The wire frame for a cube draws the edges of the cube, connecting all points that differ in one of the values, e.g.  $(0, 0, 0)$  and  $(1, 0, 0)$ , or  $(0, 0, 0)$  and  $(0, 1, 0)$  for a 3-D cube. Each edge is a vector of length 1 and is defined by specifying the row numbers of the two corresponding elements of the vertex data, e.g. in a 3-D cube  $(2, 4)$  would connect rows  $(1, 0, 0)$  and  $(1, 1, 0)$ . Table 1 gives vertex and edge lists for  $p = 1, 2$ , and 3. Edges are not ordered  $(1, 2) = (2, 1)$ , and we use just one of the two, with the smaller number first. Presented below are three ways to generate an edge set, the last being the most computationally efficient but less intuitive.

- **Method 1:** Distance of 1.

The distances between all  $p * (p - 1)/2$  pairs of vertices are computed, and the pairs of vertices which have distance 1 are returned. This is the simplest approach to generate the edge set but obviously slow to compute as  $p$  increases.

```
cube_edges_length1 <- function(cube) {
  p <- ncol(cube)
  num_points <- 2 ^ p
  from_to <- matrix(NA, nrow = num_points * p / 2, ncol = 2)
  next_store_position <- 1
  for (i in 1:(num_points - 1)) {
    for (j in (i + 1):num_points) {
      d1 <- sum((cube[i, ] - cube[j, ]) ^ 2)
      if (d1 == 1) {
        from_to[next_store_position, ] <- c(i, j)
        next_store_position <- next_store_position + 1
      }
    }
  }
  from_to
}
```

- **Method 2:** The binomial approach.

This is faster to compute than the first method because it involves only a single loop over the cube vertices. For this approach to work, the vertices of the cube need to have been created using the methods described in Section 32.2.1. Each vertex, that has  $c$  elements equal to 0, will be connected to  $c$  other vertices, and we need to determine the row numbers for these other vertices. (The row number for a corresponding connected vertex is obtained by adding  $2^{(j-1)}$ ,  $j = 1, \dots, p$ , if column  $j$  contains a 0, to the row number,  $i, i = 1, \dots, (\#vertices - 1)$  of the originating vertex.) For example, for a 3-D cube, the first vertex  $(0, 0, 0)$  will be connected to vertices  $2^0 + 1 = 2, 2^1 + 1 = 3$  and  $2^2 + 1 = 5$ .

```
cube_edges_binomial <- function(cube) {
  p <- ncol(cube)
  num_points <- 2 ^ p
  from_to <- matrix(NA, nrow = num_points * p / 2, ncol = 2)
  next_store_position <- 1
  for (i in 1:(num_points - 1)) {
    for (j in 1:p) {
      if (cube[i, j] == 0) {
        from_to[next_store_position, ] <- c(i, 2 ^ (j - 1) + i)
        next_store_position <- next_store_position + 1
      }
    }
  }
  from_to
}
```

- **Method 3:** Binary relationships.

The final method is the most computationally efficient but the least intuitive. Here we will use the fact that the vertices of the cube can be represented as binary numbers, e.g.  $(0, 1, 1) = 011_2 = 3_{10}$ . This allows us to both vectorize the code and use the C bitwise operations provided by the [bitops](#) package.

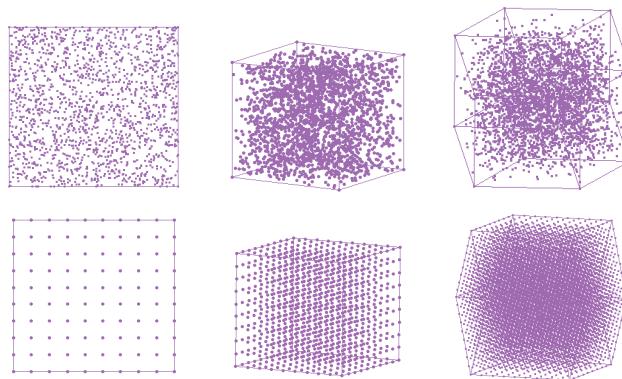
The key insight to note is that edges connect vertices which have a single bit flipped. For example, 011 connects to 111, 001 and 010 (vertex 3 connects to 7, 1, and 2). We can flip a single

bit with the *exclusive or* function,  $011 \oplus 100 = 111$ ,  $011 \oplus 010 = 001$ ,  $011 \oplus 001 = 010$ . This leads to a fast and efficient method for generating the edges.

```
library(bitops)
cube_edges_binary <- function(p) {
  vertices <- 0:(2 ^ p - 1)
  from_verts <- vertices[
    rep(1:(2 ^ p), each = p)
  ]
  from_to <- data.frame(
    from = from_verts,
    to = bitXor(from_verts, 2 ^ (0:(p - 1)))
  )
  from_to <- subset(from_to, from < to) + 1
  from_to
}
```

### Solid cube

A solid cube has points in the interior (Figure 2). It is easy to generate, using either random sampling or a fixed grid.



**Figure 2:** Solid cubes in 2-D, 3-D, and 4-D made of (top) independent random samples from  $p$  uniform distributions and (bottom) fixed grid points. As the dimensions increase, the vertices look sparse, more so with the random samples.

- **Method 1:** Random uniform.

The R function `runif` generates samples from a uniform distribution between 0 and 1. Generating  $p$  random uniform values creates a  $p$ -dimensional vector corresponding to a point inside a  $p$ -dimensional cube. The number of points needed to make the cube appear solid increases exponentially as  $p$  increases. For example, a 3-D cube with  $k$  points on each side has  $k^3$  total points, and a 4-D cube with the same  $k$  points per side  $k^4$  total points. Thus, every time the dimension is increased, the number of points must be increased substantially for the shape to look similarly solid. In our `cube_solid_random` function, we use a base of 850 points, and the total number of points is capped at 50000 points for speed of viewing.

```
cube_solid_random <- function(p, n = 850 * 2 ^ p) {
  matrix(runif(n * p), ncol = p)
}
```

- **Method 2:** Equidistant.

A solid cube can be generated using equidistant points. As with the second vertex generation method, the `expand.grid` function is used. The input  $n$  allows the number of grid points to be varied.

```
cube_solid_grid <- function(p, n) {
  grid <- list(seq(0, 1, length = n))
  do.call(expand.grid, rep(grid, p))
}
```

There are advantages and disadvantages to the methods provided. The first method, random uniform points, produces a solid cube that looks more solid, but as  $p$  increases, points near the vertices become more scarce. The second method, equidistant points, fills the vertex regions, but the structure produces regular patterns which can be distracting to the viewer.

### Hollow cube

The “face” of a cube is a surface that is one dimension lower than that of the cube. For example, a face of a 3-D cube is a 2-D square and a face of a 4-D cube is a 3-D cube.

To generate points on the faces of a cube, points are created in all dimensions except one. The remaining dimension is given the value 0 or 1, to create the opposing faces. Because the face of a cube is a  $(p - 1)$ -dimensional cube with a 0 or 1 in the remaining column, we may perform two different methods to produce the  $p$ -dimensional cube’s faces.

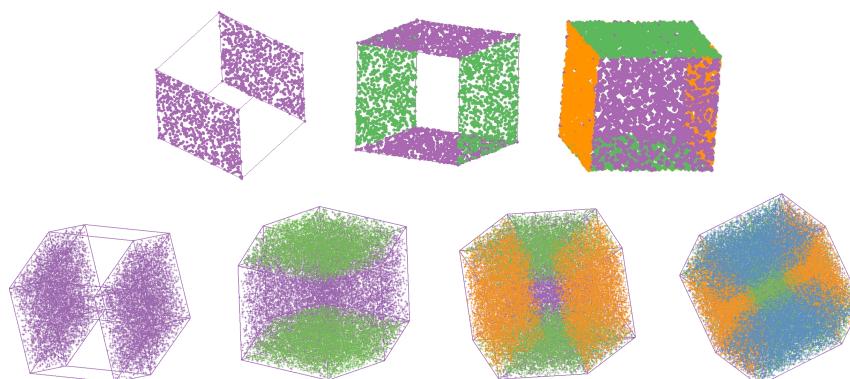
- **Method 1:** Equidistant faces.

Equidistant faces may leverage the fact that each “face” contains the same data. Therefore, we may calculate a single  $(p - 1)$ -dimensional equidistant cube and place it in the return matrix multiple times with the  $i$ th column missing from the return matrix. The number of rows of the return matrix is equivalent to  $2 * p * \text{nrow}(\text{face})$  with the first half of the return matrix being 0’s and the last half being 1’s. The input  $n$  is supplied directly to the *cube\_solid\_grid* function.

```
cube_face_grid <- function(p, n = 10) {
  face <- cube_solid_grid(p - 1, n)
  face_n <- nrow(face)
  faces <- do.call(data.frame, rep(list(X = rep(0:1, each = p * face_n)), p))
  for(i in seq_len(p)) {
    faces[(face_n * (i - 1) + 1):(face_n * i), -i] <- face
    faces[(face_n * (i - 1) + 1):(face_n * i) + (p * face_n), -i] <- face
  }
  return(as.matrix(faces))
}
```

- **Method 2:** Random uniform faces.

Naively creating a 3-D cube, the  $X_1$  and  $X_2$  components of the cube are given random values and the  $X_3$  components would be set to 0 in the first half and 1 in the second half. The process would then be repeated for the remaining columns, as shown in Figure 3. This will create six 2-D squares which form the faces of a 3-D cube. The bottom row shows the different faces of a 4-D cube. The left side plot shows the first pair of faces, a solid 3-D cube in  $X_1, X_2, X_3$ , with fixed values on the fourth dimension  $X_4$ . The subsequent plots show the remaining faces.



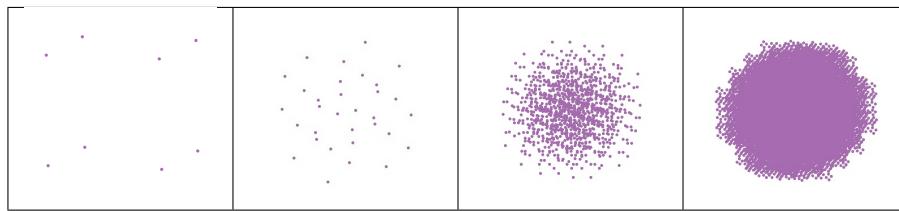
**Figure 3:** Faces of a 3-D cube (top row) and a 4-D cube (bottom row), obtained by fixing one column of values to be 0 or 1, and allowing the other columns to vary freely between 0 and 1.

Unlike the equidistant faces of a cube, each random uniform face must be different from every other face. To avoid repetitive calculations, we leverage the fact that a  $p$ -dimensional random uniform cube with one column missing is equivalent a  $(p - 1)$ -dimensional random uniform cube. Therefore, we generate a  $p$ -dimensional cube and insert 0’s or 1’s for each of the corresponding faces in each dimension. The input  $n$  matches the number of random points for a  $(p - 1)$ -dimensional cube.

```

cube_face_random <- function(p, n = 850 * 2 ^ (p - 1)) {
  faces <- cube_solid_random(p, 2 * p * n)
  for (i in seq_len(p)) {
    faces[(n * (i - 1) + 1):(n * i), i] <- 0
    faces[(n * (i - 1) + 1):(n * i) + (p * face_n), i] <- 1
  }
  faces
}

```



**Figure 4:** Vertices of cubes, (left to right) 3-D, 5-D, 10-D, and 15-D. Cubes look more rounded in projections as the dimension increases.

### High-D cubes look spherical!

As the dimension increases, the shape of the cube appears more rounded than square in a 2-D projection. Explanation is given in [Diaconis and Freedman \(1984\)](#) and is related to the Central Limit Theorem. When we use a tour to visualize high-dimensional data we examine low-dimensional projections. Consider the axes for a  $p$ -dimensional space labeled as  $X_1, \dots, X_p$ . A 1-D projection is generated by taking a linear combination of these axes, such as  $a_1 X_1 + \dots + a_p X_p$ . The squared values of  $a_j, j = 1, \dots, p$  are constrained to sum to 1. As  $p$  increases, combining the values operates like averaging the values in many dimensions, resulting in views that look Gaussian. Another way to think about it is that we are looking at rotated cubes rather than a cube through its square face and this gets increasingly rounded as the dimension increases.

## Spheres

A sphere can be described as all points within a fixed radius (for simplicity use 1) around a fixed point (for simplicity use  $\mathbf{zero}_p$ ),  $\{X : X_1^2 + \dots + X_p^2 \leq 1\}$ . A hollow sphere is the set of points with radius equal to 1,  $\{X : X_1^2 + \dots + X_p^2 = 1\}$ . Generating the hollow sphere is simpler than the solid sphere.

### Hollow sphere

To generate points uniformly distributed on the surface of a sphere we use the following trick: first, we generate a random vector from a multivariate standard normal distribution and then normalize its length. The normalized point is now a random point on a unit sphere ([Watson, 1983](#), 2.6). The top row of Figure 5 shows the results.

```

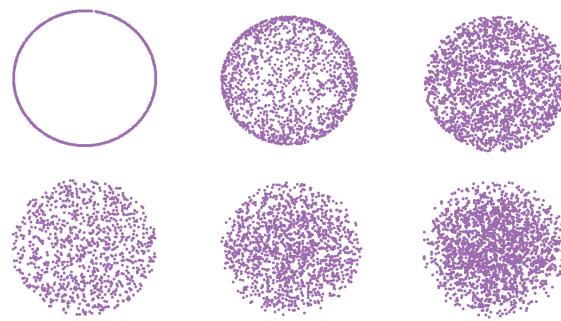
norm_vec <- function(x) {
  x / sqrt(sum(x ^ 2))
}

sphere_hollow <- function(p, n = p * 500) {
  x <- matrix(rnorm(n * p), ncol = p)
  t(apply(x, 1, norm_vec))
}

```

### Don't reject the solid sphere!

Solid spheres can be generated in much the same way as solid cubes; use random points to fill the object. While the solid cube fills the box, the sphere's points are inside a radius of 1 from the center. A simple approach would be to use a rejection method: generate points in the solid cube and discard those with radius more than 1. This is problematic as  $p$  increases, since most points will eventually

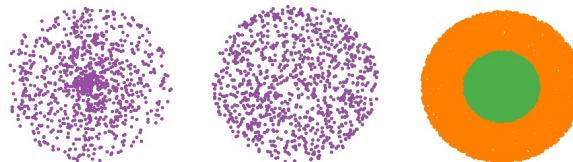


**Figure 5:** Hollow (top row) and solid spheres (bottom row) for 2-D, 3-D, 4-D.

be rejected. For example, to generate the points of a 3-D sphere, around 50% of the proposed points are accepted, but for a 10-D sphere only 0.25% of the proposed points are accepted. The space in the corners of the enveloping cube, outside the sphere, increases dramatically with  $p$ .

The approach we used is a minor modification to the method used to generate a hollow sphere. Figure 6 illustrates the process. The vector length is randomly sampled from a uniform distribution on  $(0, 1)$  (left plot). The result is raised to the power  $1/p$  to adjust for the volume increase with  $p$ , resulting in points spread evenly throughout the inside of the sphere (middle plot). Taking the sphere to the  $1/p$  power may seem ad hoc, but the operation ensures that the density is uniform within the sphere. To see this, compare circles of radius 1 and 2 (right plot). The area of the smaller circle equals  $\pi = 1^2\pi$ . The area of the whole circle equals  $4\pi = 2^2\pi$ , which is four times as large but only twice the radius. Thus, without accounting for radial distance, more points will be generated closer to the center than is warranted by the area. Raising the vector length to the power  $1/p$ ,  $1/2$  in our example, corrects for the volume.

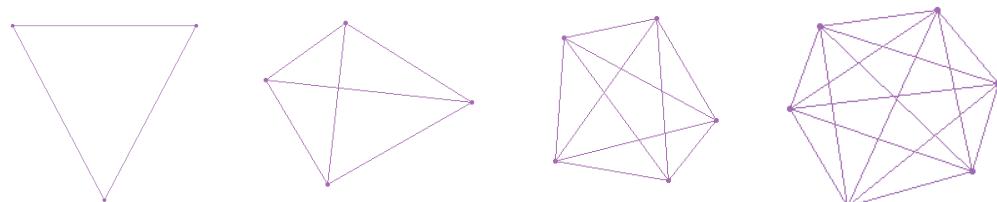
```
sphere_solid_random <- function(p, n = p * 500) {
  sphere_hollow(p, n) * runif(n) ^ (1 / p)
}
```



**Figure 6:** Solid sphere in 2-D: (left) generated randomly results in an over-concentration of points in the center, (middle) adjusted for volume generates a uniform distribution inside the sphere, and (right) comparison of areas of circles of radius 1 and 2.

## Simplexes

Simplexes are one of the simplest objects to create and view. A  $p$ -D simplex is a shape that is created in the  $(p + 1)$ th dimension with vertices corresponding to the coordinate axes. The simplex vertices are then projected, using a Helmert matrix, into the  $p$ -dimensional space in which it exists.



**Figure 7:** Wireframe simplexes: 2-D, 3-D, 4-D, 5-D.

For example, a 2-D simplex has unprojected 3-D vertices at (1,0,0), (0,1,0), (0,0,1), which are reduced by the Helmert transformation to points in a 2-D equilateral triangle, (0.7071,0.4082), (-0.7071,0.4082), (0.0000,-0.8165).

```
helmert <- function(d) {
  helmert_mat <- matrix(NA, nrow = d, ncol = d)
  helmert_mat[1, ] <- rep(1 / sqrt(d), d)
  for (i in 1:(d - 1)) {
    helmert_mat[i + 1, ] <- c(
      rep(1 / sqrt(i * (i + 1)), i),
      -i / sqrt(i * (i + 1)),
      rep(0, d - i - 1)
    )
  }
  helmert_mat
}

simplex <- function(p) {
  x <- diag(p)
  # center simplex
  x <- x - matrix(1 / p, p, p)
  hm <- helmert(p)
  final <- (x %*% t(hm))[, -1]
  final
}
```

The wire frame for a simplex connects every point to every other point, and can be computed in just two lines of code, following method 2 of the cube vertices. `simplex_wires` makes a list of all pairs of vertices and then removes the edges that connect a vertex to itself.

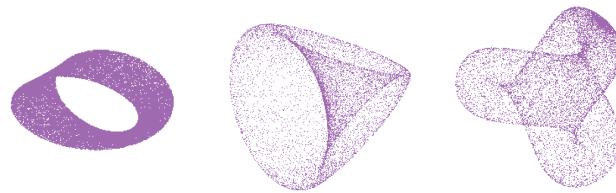
```
simplex_wires <- function(simplex) {
  wires <- do.call(
    expand.grid,
    list(
      c(1:nrow(simplex)),
      c(1:nrow(simplex))
    )
  )
  wires[!wires[,1] == wires[,2],]
}
```

## Polyhedra

A polyhedron is a three dimensional object that contains straight edges and has flat faces. Our polyhedra data comes from George W. Hart's website ([Hart, 2000](#)). Hart's website contains an extensive collection of polyhedra, ranging from Platonic Solids to Stellations of the Rhombic Triacontahedron. In our data sets, we used the information from Platonic Solids, Kepler-Poinsot Polyhedra, Archimedean Polyhedra and its duals, Prisms. The data was reformatted from VRML into XML. The vertices and wire frames from separate files were compiled into tables. Some reformatting of edges was also necessary.

## Surfaces and curves

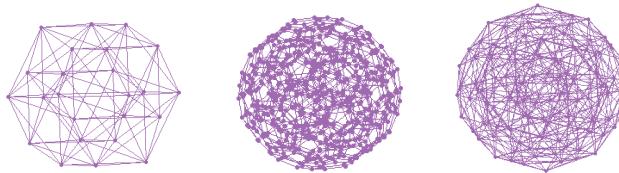
Paul Bourke's website ([Bourke, 1996](#)) has equations for generating several famous objects, including the Möbius Strip, Steiner's Roman Surface and the Klein Bottle (Figure 8). It is interesting to see how each object twists onto itself. R functions based on these equations were written to produce each object. For each shape, a set of random angles were generated to seed the equations, producing points on the surface. For some of the objects, to be displayed in its familiar form, the plot limits for each variable need to use the same global minimum and maximum values.



**Figure 8:** Möbius strip, Steiner's roman surface, and Boy's surface.

## Polytopes

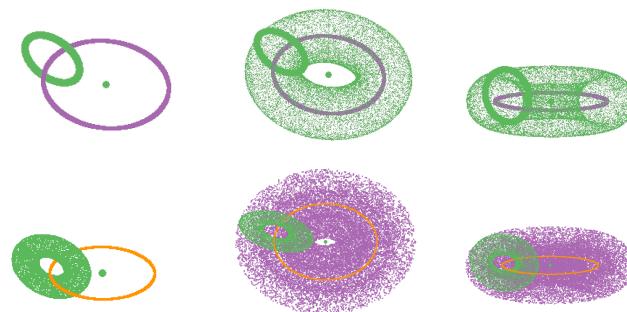
A *polytope* is a generalized term of a geometric shape in any dimension. A polygon is a 2-D polytope. A polyhedron is a 3-D polytope. A polychoron is a 4-D polytope. Beyond that, we typically use polytope to refer to any  $p$ -gon. Our polychoron data comes from Paul Bourke's website (Bourke, 2003), where there is also information on some of the objects that have been covered (cube, simplex) in the preceding sections of this paper, and new objects, 24-cell, 120-cell, 600-cell (Figure 9). The explanations of these polytopes is very clear. The data has been formatted into XML files allowing descriptions of the vertices and edges for each shape.



**Figure 9:** 24-Cell, 120-Cell, and 600-Cell.

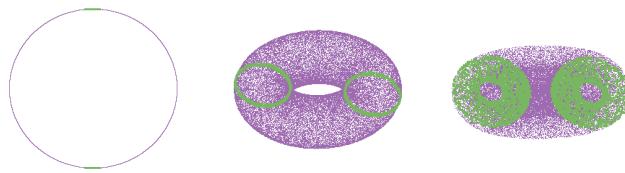
## Tori

A “doughnut” torus is known as a ring torus. Paul Bourke's website (Bourke, 1990) on “The Torus and Super Torus” provides the inspiration. The website explains how the 3-D torus is made. It also contains information that we used to develop the process of building high-dimensional tori. Figure 10 shows this process: a smaller circle that follows a larger circle, creating a doughnut. The points for the torus are formed by polar coordinates.



**Figure 10:** Generating a torus: 2-D to 3-D (top row), and 3-D to 4-D (bottom row).

To produce a 4-D torus, a 3-D torus is rotated around a circle into the fourth dimension (Figure 10 bottom row). This torus still has a hole in the center. The process can be thought of as a recursive circle system. For a 3-D torus, the smaller radius circle follows the larger radius circle. For a 4-D torus, a 3-D torus follows an even larger radius circle. That is, the lower-dimensional torus is shifted by a fixed distance and rotated about a *new* axis perpendicular to the axis of the hole. This algorithm for a ring torus has not been previously defined and will be explained in detail in the following section.



**Figure 11:** 2-D, 3-D, and 4-D tori with features highlighted.

### Ring torus

A ring torus has a hole in the center of the object and is generated recursively. A 2-D circle forms the base of the torus, which is defined by:

$$\begin{aligned} X_1 &: \cos(\theta_1) * r_1 \\ X_2 &: \sin(\theta_1) * r_1 \end{aligned}$$

using one radius and one angle. The 3-D torus is defined by:

$$\begin{aligned} X_1 &: \cos(\theta_1) * (r_1 + \cos(\theta_2) * r_2) \\ X_2 &: \sin(\theta_1) * (r_1 + \cos(\theta_2) * r_2) \\ X_3 &: \sin(\theta_2) * r_2 \\ r_1 &> r_2 \end{aligned}$$

Notice that 3-D torus builds from the 2-D:  $r_1$  is replaced by  $(r_1 + \cos(\theta_2)r_2)$ . The 3-D torus has four parameters: two angles and two radii. The third dimension is formed entirely by the additional angle and radius. A 4-D torus will be generated with the same pattern as the 3-D torus: a new dimension will be added and material will be inserted recursively into the formulas:

$$\begin{aligned} X_1 &: \cos(\theta_1) * (r_1 + \cos(\theta_2) * (r_2 + \cos(\theta_3) * r_3)) \\ X_2 &: \sin(\theta_1) * (r_1 + \cos(\theta_2) * (r_2 + \cos(\theta_3) * r_3)) \\ X_3 &: \sin(\theta_2) * (r_2 + \cos(\theta_3) * r_3) \\ X_4 &: \sin(\theta_3) * r_3 \\ r_1 &> r_2 > r_3 \end{aligned}$$

The steps to building a higher-dimensional torus are:

1. add a new dimension that equals  $\sin(\theta_i) * r_i$ .
2. in all other dimensions, replace  $r_{i-1}$  with  $r_{i-1} + \cos(\theta_i) * r_i$ .
3. repeat.

However, these steps are not easy to recurse. Replacing values in a formula after it has been realized is not simple. By rearranging the formulas, a better method to recurse is achieved:

$$\begin{aligned} X_1 &: ((\cos(\theta_3) * r_3 + r_2) * (\cos(\theta_2) + r_1) * \cos(\theta_1)) \\ X_2 &: ((\cos(\theta_3) * r_3 + r_2) * (\cos(\theta_2) + r_1) * \sin(\theta_1)) \\ X_3 &: (\cos(\theta_3) * r_3 + r_2) * \sin(\theta_2) \\ X_4 &: \sin(\theta_3) * r_3 \\ r_1 &> r_2 > r_3 \end{aligned}$$

The first step of the recursion, starting from the last dimension, is given as follows:

$$\begin{aligned} X_1 &: \cos(\theta_3) * r_3 \\ X_2 &: \cos(\theta_3) * r_3 \\ X_3 &: \cos(\theta_3) * r_3 \\ X_4 &: \sin(\theta_3) * r_3 \end{aligned}$$

which translates to this R code:

```
torus<-c(
  rep(cos(theta[p - 1]) * radius[p - 1], p - 1),
  sin(theta[p - 1]) * radius[p - 1]
)
```

From this start, we recurse backwards from  $p - 1$  to 2. A new radius is added at each iteration which is multiplied with the previous equation by the cosine of an angle. The final step adds a last radius and multiplies the result by the sine of the new angle.

```
for (i in (p - 1):2) {
  for (j in (i - 1):1) {
    torus[j] <- (torus[j] + radius[i - 1]) * cos(theta[i - 1])
  }
  torus[i] <- (torus[i] + radius[i - 1]) * sin(theta[i - 1])
}
```

The construction of a 4-D torus is shown below:

$$\begin{aligned} p &= 4 \\ \text{Base} \\ X_1 &: \cos(\theta_3) * r_3 \\ X_2 &: \cos(\theta_3) * r_3 \\ X_3 &: \cos(\theta_3) * r_3 \\ X_4 &: \sin(\theta_3) * r_3 \\ \\ i &= p - 1 = 3 \\ j &= 2 : 1 \\ X_1 &: (\cos(\theta_3) * r_3 + r_2) * \cos(\theta_2) \\ X_2 &: (\cos(\theta_3) * r_3 + r_2) * \cos(\theta_2) \\ X_3 &: (\cos(\theta_3) * r_3 + r_2) * \sin(\theta_2) \\ X_4 &: \sin(\theta_3) * r_3 \\ \\ i &= 2 \\ j &= 1 : 1 \\ X_1 &: ((\cos(\theta_3) * r_3 + r_2) * \cos(\theta_2) + r_1) * \cos(\theta_1) \\ X_2 &: ((\cos(\theta_3) * r_3 + r_2) * \cos(\theta_2) + r_1) * \sin(\theta_1) \\ X_3 &: (\cos(\theta_3) * r_3 + r_2) * \sin(\theta_2) \\ X_4 &: \sin(\theta_3) * r_3 \end{aligned}$$

This code results in one row of data for one point on the 4-D torus for each value of angle. By varying the angle and binding the result, we get points over the surface of the torus:

```

finished <- rbind(finished, torus.row)

or

matrix(
  do.call(rbind, as.list(
    replicate(
      n,
      torus.row(radius, p)
    )
  )),
  ncol = p, byrow = TRUE
)

```

The angles create the rings, and thus need to vary fully between 0 and  $2\pi$ :

```
theta <- runif(p - 1, min = 0, max = 2 * pi)
```

The radii for the process are fixed at the start. To produce a hole in each dimension the radii need to decrease with  $p$ . We set the hole to have a size of 1 in each dimension, and the radii are reduced by a power of 2 from the previous dimension.

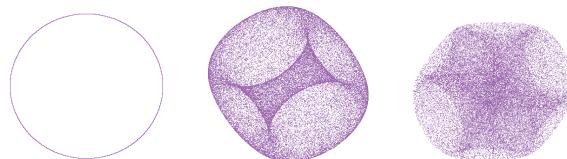
```
radius <- 2 ^ ((p - 2):0)
```

This produces points fairly evenly but not uniformly spread on the surface of the torus. A different way to generate the torus is to produce the angles at set intervals, resulting in more circular patterns.

### Flat torus

Another common hyper-torus is the flat torus. A flat torus is commonly seen expanding into infinity as a screen saver on some computers. The flat torus has multiple holes in its center and is easy to generate.

A flat torus is formed in pairs of dimensions, defined by a sine and cosine of one angle, for example the circle is generated in 2-D by  $\cos(\theta_1)$  and  $\sin(\theta_1)$ . A flat torus in any dimension is created from multiple pairs of sine and cosine, e.g. a 1-D torus is generated by one pair, a 2-D torus by two pairs, four variables, and a 3-D torus by three pairs, six variables (Figure 12). All values of sine and cosine are generated from angles,  $(-2\pi, 2\pi)$ , separately for each pair. The flat torus has an even number of dimensions, but an effective dimension half that size. Figure 13 illustrates the construction of a 6-D flat torus.



**Figure 12:** Flat tori in 2-D, 4-D, and 6-D.

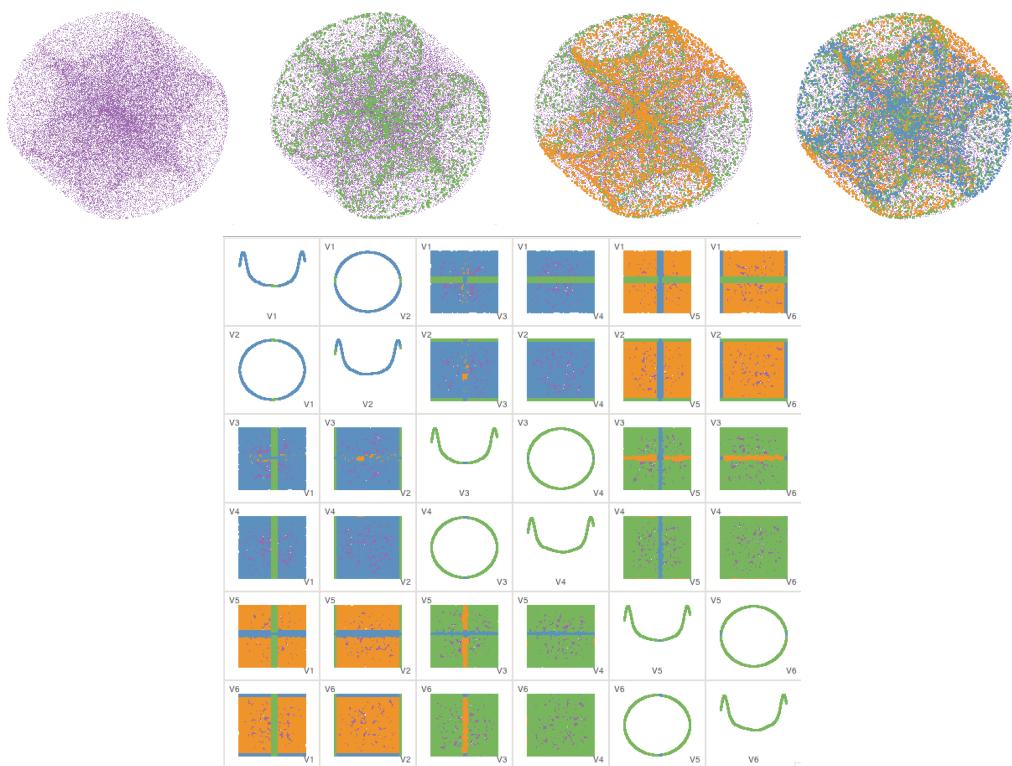
### Solid tori

These tori are all hollow. To create points in the interior of the tori one would randomly generate the radii length for each point.

## Conclusion

This paper has described how `geozoo` generates several types of high-dimensional geometric shapes. The result is a library designed to help conceptualize objects in high-dimensional spaces. It has also led to some new geometric shape definitions.

This library, although seemingly removed from real high-dimensional data has some strong connections. Much of our data analytic methods are based in high-dimensional Euclidean space. Developing some visual insight into this space can help to understand the methods that operate in



**Figure 13:** Views of the 6-D flat torus which illustrate its construction. The torus (top right) has its components highlighted with green, orange and blue, and the same torus is displayed as a scatterplot matrix, better revealing the construction.

higher dimensions. Some data problems can be closely mapped to the geometric shapes. For example, ranked data showing preferences for a fixed set of objects, can be mapped to high-dimensional polytopes (Thompson, 1993). Values in a sample are commonly constrained to sum to a fixed number, for example, 100%, forming compositional data. This type of data lies inside a  $p$ -D simplex. Good experimental designs commonly have a geometric structure (Hedayat et al., 1999). The ideas to examine boundaries of supervised classifiers described in Caragea et al. (2008) build on the geometric shapes described here.

We encourage the reader to look at the movies, and images, and download the data on the project web site, which can be accessed at <http://schloerke.github.io/geozoo/>. The **geozoo** package contains the R code to generate the geometric shapes and is available to download from CRAN (<http://www.R-project.org>). We especially encourage readers to experiment with creating new high-dimensional geometric shapes, or to contribute ideas and code back to this project.

## Acknowledgments

This work was supported by National Science Foundation grant DMS0706949. Andreas Buja provided some helpful feedback on the work. Dianne Cook thanks the Isaac Newton Institute, Cambridge, UK for support, where she was visiting while drafting this paper.

## Bibliography

- E. Abbott. *Flatland: A Romance of Many Dimensions*. Dover Publications, 1884. [p243]
- D. Asimov. The Grand Tour: A Tool for Viewing Multidimensional Data. *SIAM Journal of Scientific and Statistical Computing*, 6(1):128–143, 1985. [p243]
- P. Bourke. The Torus and Supertoroid. <http://paulbourke.net/geometry/torus/>; Date accessed: 2016-07-06, 1990. [p251]
- P. Bourke. Geometry, Surfaces, Curves, Polyhedra. <http://paulbourke.net/geometry/>; Date accessed: 2016-07-06, 1996. [p250]
- P. Bourke. Regular Polytopes (Platonic Solids) in 4D. <http://paulbourke.net/geometry/hyperspace/>; Date accessed: 2016-07-06, 2003. [p251]
- A. Buja, D. Cook, D. Asimov, and C. Hurley. Computational Methods for High-Dimensional Rotations in Data Visualization. In C. R. Rao, E. J. Wegman, and J. L. Solka, editors, *Handbook of Statistics: Data Mining and Visualization*, pages 391–413. Elsevier/North Holland, <http://www.elsevier.com>, 2005. [p243]
- D. Caragea, D. Cook, H. Wickham, and V. Honavar. Visual Methods for Examining SVM Classifiers. In S. Simonoff, M. Böhnen, and A. Mazeika, editors, *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics*, pages 136–153. Springer-Verlag, Berlin, 2008. [p255]
- D. Cook. Calibrate Your Eyes to Recognize High-Dimensional Shapes from Their Low-Dimensional Projections. *Journal of Statistical Software*, 2(6), 1997. <https://www.jstatsoft.org/article/view/v002i06> and <http://www.public.iastate.edu/~dicook/JSS/paper/paper.html>; Date accessed: 2016-07-06. [p243]
- D. Cook, E. K. Lee, A. Buja, and H. Wickham. Grand Tours, Projection Pursuit Guided Tours and Manual Controls. In C.-H. Chen and A. U. W. Härdle, W., editors, *Handbook of Computational Statistics (Volume III) Data Visualization*. Springer, New York, NY, 2007. URL <http://www.springer.com>. [p243]
- P. Diaconis and D. Freedman. Asymptotics of Graphical Projection Pursuit. *Annals of Statistics*, 12(3): 793–815, 1984. [p248]
- G. Hart. Virtual Polyhedra. <http://www.georgehart.com/virtual-polyhedra/vp.html>; Date accessed: 2016-07-06, 2000. [p250]
- A. S. Hedayat, N. J. A. Sloane, and J. Stufken, editors. *Orthogonal Arrays: Theory and Applications*. Springer, <http://www.springer.com>, 1999. [p255]
- E. Martin. Flatland. <http://www.der.org/films/flatland.html>; Date accessed: 2016-07-06, 1965. [p243]
- R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2003. URL <http://www.r-project.org>. ISBN 3-900051-00-3. [p]
- D. F. Swayne, D. Temple Lang, A. Buja, and D. Cook. GGobi: Evolving from XGobi into an Extensible Framework for Interactive Data Visualization. *Journal of Computational Statistics and Data Analysis*, 43:423–444, 2003. URL <http://authors.elsevier.com/sd/article/S0167947302002864>. [p243]
- G. L. Thompson. Generalized permutation polytopes and exploratory graphical methods for ranked data. *The Annals of Statistics*, 21(3):1401–1403, 1993. [p255]
- G. Watson. *Statistics on spheres*. University of Arkansas lecture notes in the mathematical sciences. Wiley, 1983. ISBN 9780471888666. URL <https://books.google.com/books?id=tBjvAAAQAAJ>. [p248]
- H. Wickham, D. Cook, H. Hofmann, and A. Buja. tourr: An R package for exploring multivariate data with projections. *Journal of Statistical Software*, 40(2):1–18, 2011. <http://www.jstatsoft.org/v40/i02/>. [p243]

*Barret Schloerke*  
*Department of Statistics*  
*Purdue University*  
*250 N. University Street*  
*West Lafayette, IN 47907*  
*USA*  
[schloerke@gmail.com](mailto:schloerke@gmail.com)

*Hadley Wickham*  
*RStudio Inc*  
[hadley@rstudio.com](mailto:hadley@rstudio.com)

*Dianne Cook*  
*Department of Econometrics and Business Statistics*  
*Monash University*  
*Wellington Road*  
*Clayton, Victoria 3800*  
*Australia*  
[dcook@monash.edu](mailto:dcook@monash.edu)

*Heike Hofmann*  
*Department of Statistics*  
*Iowa State University*  
*Snedecor Hall*  
*Ames, Iowa 50011*  
*USA*  
[hofmann@iastate.edu](mailto:hofmann@iastate.edu)

# Ake: An R Package for Discrete and Continuous Associated Kernel Estimations

by Wanbitching E. Wansouwé, Sobom M. Somé and Célestin C. Kokonendji

**Abstract** Kernel estimation is an important technique in exploratory data analysis. Its utility relies on its ease of interpretation, especially based on graphical means. The **Ake** package is introduced for univariate density or probability mass function estimation and also for continuous and discrete regression functions using associated kernel estimators. These associated kernels have been proposed due to their specific features of variables of interest. The package focuses on associated kernel methods appropriate for continuous (bounded, positive) or discrete (count, categorical) data often found in applied settings. Furthermore, optimal bandwidths are selected by cross-validation for any associated kernel and by Bayesian methods for the binomial kernel. Other Bayesian methods for selecting bandwidths with other associated kernels will complete this package in its future versions; particularly, a Bayesian adaptive method for gamma kernel estimation of density functions is developed. Some practical and theoretical aspects of the normalizing constant in both density and probability mass functions estimations are given.

## Introduction

Kernel smoothing methods are popular tools for revealing the structure of data that could be missed by parametric methods. For real datasets, we often encounter continuous (bounded, positive) or discrete (count, categorical) data types. The classical kernels methods assume that the underlying distribution is unbounded continuous, which is frequently not the case; see, for example, Duong (2007) for multivariate kernel density estimation and discriminant analysis. A solution is provided for categorical data sets by Hayfield and Racine (2008). In fact, they used kernels well adapted for these categorical sets (Aitchison and Aitken, 1976). Throughout the present paper, the unidimensional support  $\mathbb{T}$  of the variable of interest can be  $\{0, 1, \dots, N\}$ ,  $[a, b]$  or  $[0, \infty)$  for a given integer  $N$  and reals  $a < b$ .

The recently developed **Ake** package, implements associated kernels that seamlessly deal with continuous (bounded, positive) and discrete (categorical, count) data types often found in applied settings; see, for example, Libengué (2013) and Kokonendji and Senga Kiessé (2011). These associated kernels are used to smooth probability density functions (p.d.f.), probability mass functions (p.m.f.) or regression functions. The coming versions of this package will contain, among others, p.d.f. estimation of heavy tailed data (e.g., Ziane et al., 2015) and the estimation of other functionals. The bandwidth selection remains crucial in associated kernel estimations of p.d.f., p.m.f. or regression functions. Some methods have been investigated for selecting bandwidth parameters but the commonly used is the least squared cross-validation. A Bayesian approach has been also recently introduced by Zougab et al. (2012) in the case of a binomial kernel. This method can be extended to various associated kernels with other functionals. Despite the great number of packages implemented for nonparametric estimation in continuous cases with unbounded kernels, to the best of our knowledge, the R packages to estimate p.m.f. with categorical or count variables, p.d.f. with bounded or positive datasets, and regression functions have been far less investigated.

The rest of the paper is organized as follows. In Section [Non-classical associated kernels](#), we briefly describe the definition of associated kernels and then illustrate examples in both continuous and discrete cases which are discussed. Then, the associated kernel estimator for p.d.f. or p.m.f. is presented and illustrated with some R codes in Section [Density or probability mass function estimations](#). In particular, three bandwidth selection methods are available: cross-validation for any (continuous or discrete) associated kernel, the Bayesian local method for the binomial kernel and also a new theoretical Bayesian adaptive method for the gamma kernel. Also, some practical and theoretical aspects of the normalizing constant in both p.d.f. and p.m.f. estimations are given. Section [Bandwidth selection for kernel regression involving associated kernels](#) investigates the case of regression functions with two bandwidth selection techniques: cross-validation and also the Bayesian global method for the binomial kernel. Section [Summary and final remarks](#) concludes.

## Non-classical associated kernels

Recall that the support  $\mathbb{T}$  of the p.m.f., p.d.f. or regression function, to be estimated, is any set  $\{0, 1, \dots, N\}$ ,  $[a, b]$  or  $[0, \infty)$  for a given integer  $N$  and reals  $a < b$ . The associated kernel in both continuous and discrete cases is defined as follows.

**Definition 34.2.1.** ([Kokonendji and Senga Kiessé, 2011](#); [Libengué, 2013](#)) Let  $\mathbb{T} (\subseteq \mathbb{R})$  be the support of the p.m.f., p.d.f. or regression function, to be estimated,  $x \in \mathbb{T}$  a target and  $h$  a bandwidth. A parametrized p.m.f. (respectively p.d.f.)  $K_{x,h}(\cdot)$  of support  $S_{x,h} (\subseteq \mathbb{R})$  is called “associated kernel” if the following conditions are satisfied:

$$x \in S_{x,h}, \quad (1)$$

$$\mathbb{E}(\mathcal{Z}_{x,h}) = x + a(x, h), \quad (2)$$

$$\mathbb{V}(\mathcal{Z}_{x,h}) = b(x, h), \quad (3)$$

where  $\mathcal{Z}_{x,h}$  denotes the random variable with p.m.f. (respectively p.d.f.)  $K_{x,h}$  and both  $a(x, h)$  and  $b(x, h)$  tend to 0 as  $h$  goes to 0.

**Remark 34.2.2.** This definition has the following interesting interpretations:

- (i) The function  $K_{x,h}(\cdot)$  is not necessary symmetric and is intrinsically linked to  $x$  and  $h$ .
- (ii) The support  $S_{x,h}$  is not necessary symmetric around  $x$ ; it can depend or not on  $x$  and  $h$ .
- (iii) The condition (1) can be viewed as  $\cup_{x \in \mathbb{T}} S_{x,h} \supseteq \mathbb{T}$  and it implies that the associated kernel takes into account the support  $\mathbb{T}$  of the density  $f$ , to be estimated.
- (iv) If  $\cup_{x \in \mathbb{T}} S_{x,h}$  does not contain  $\mathbb{T}$  then this is the well-known problem of boundary bias.
- (v) Both conditions (2) and (3) indicate that the associated kernel is more and more concentrated around  $x$  as  $h$  goes to 0. This highlights the peculiarity of the associated kernel which can change its shape according to the target position.

In order to construct an associated kernel  $K_{x,h}(\cdot)$  from a parametric (discrete or continuous) probability distribution  $K_\theta$ ,  $\theta \in \Theta \subset \mathbb{R}^d$  on the support  $S_\theta$  such that  $S_\theta \cap \mathbb{T} \neq \emptyset$ , we need to establish a correspondence between  $(x, h) \in \mathbb{T} \times (0, \infty)$  and  $\theta \in \Theta$ ; see [Kokonendji and Senga Kiessé \(2011\)](#). In what follows, we will call  $K \equiv K_\theta$  the *type of kernel* to make a difference from the classical notion of a continuous symmetric (e.g., Gaussian) kernel. In this context, the choice of the associated kernel becomes important as well as that of the bandwidth. Moreover, we distinguish the associated kernels said sometimes of “second order” of those said of “first order” which verify the two first conditions (1) and (2). The rest of this section is devoted to discuss examples of associated kernels in both discrete and continuous cases.

### Discrete associated kernels

Among the discrete associated kernels found in literature, we here use the best in sense of Definition 34.2.1. Negative binomial and Poisson kernels are respectively overdispersed (i.e.,  $\mathbb{V}(\mathcal{Z}_{x,h}) > \mathbb{E}(\mathcal{Z}_{x,h})$ ) and equidispersed (i.e.,  $\mathbb{V}(\mathcal{Z}_{x,h}) = \mathbb{E}(\mathcal{Z}_{x,h})$ ) and thus are not recommended; see [Kokonendji and Senga Kiessé \(2011\)](#) for further details. The first associated kernel listed below, namely the binomial kernel, is the best of the first order or *standard kernels* which satisfies

$$\lim_{h \rightarrow 0} \mathbb{V}(\mathcal{Z}_{x,h}) \in \mathcal{V}(0), \quad (4)$$

where  $\mathcal{V}(0)$  is a neighborhood of 0 which does not depend on  $x$ . The two other discrete associated kernels satisfy all conditions of Definition 34.2.1.

- The binomial (`bino`) kernel is defined on the support  $S_x = \{0, 1, \dots, x+1\}$  with  $x \in \mathbb{T} := \mathbb{N} = \{0, 1, \dots\}$  and then  $h \in (0, 1]$ :

$$B_{x,h}(u) = \frac{(x+1)!}{u!(x+1-u)!} \left( \frac{x+h}{x+1} \right)^u \left( \frac{1-h}{x+1} \right)^{x+1-u} \mathbb{1}_{S_x}(u),$$

where  $\mathbb{1}_A$  denotes the indicator function of any given event  $A$ . Note that  $B_{x,h}$  is the p.m.f. of the binomial distribution  $\mathcal{B}(x+1; (x+h)/(x+1))$  with its number of trials  $x+1$  and its success probability in each trial  $(x+h)/(x+1)$ . It is appropriate for count data with small or moderate sample sizes and, also, it satisfies (4) rather than (3); see [Kokonendji and Senga Kiessé \(2011\)](#) and also [Zougab et al. \(2012\)](#) for a bandwidth selection by Bayesian method.

- The following class of symmetric discrete triangular kernels has been proposed in Kokonendji et al. (2007). The support  $\mathbb{T}$  of the p.m.f.  $f$  to be estimated, can be unbounded (e.g.,  $\mathbb{N}, \mathbb{Z}$ ) or finite (e.g.,  $\{0, 1, \dots, N\}$ ). Then, suppose that  $h$  is a given bandwidth parameter and  $a$  is an arbitrary and fixed integer. For fixed arm  $a \in \mathbb{N}$ , the discrete triangular (DTra) kernel is defined on  $S_{x,a} = \{x, x \pm 1, \dots, x \pm a\}$  with  $x \in \mathbb{T} = \mathbb{N}$ :

$$DT_{x,h;a}(u) = \frac{(a+1)^h - |u-x|^h}{P(a,h)} \mathbb{1}_{S_{x,a}}(u),$$

where  $P(a,h) = (2a+1)(a+1) - 2 \sum_{k=0}^a k^h$  is the normalizing constant. It is symmetric around the target  $x$ , satisfying Definition 34.2.1 and suitable for count variables; see Kokonendji and Zocchi (2010) for an asymmetric version. Note that  $h \rightarrow 0$  gives the Dirac kernel.

- A discrete kernel estimator for categorical data has been introduced in Aitchison and Aitken (1976). Its asymmetric discrete associated kernel version that we here label DiracDU (DirDU) as "Dirac Discrete Uniform" has been deduced in Kokonendji and Senga Kiessé (2011) as follows. For fixed  $c \in \{2, 3, \dots\}$  the number of categories, we define  $S_c = \{0, 1, \dots, c-1\}$  and

$$DU_{x,h;c}(u) = (1-h) \mathbb{1}_{\{x\}}(u) + \frac{h}{c-1} \mathbb{1}_{S_c \setminus \{x\}}(u),$$

where  $h \in (0, 1]$  and  $x \in \mathbb{T} = S_c$ . In addition, the target  $x$  can be considered as the reference point of  $f$  to be estimated; and, the smoothing parameter  $h$  is such that  $1-h$  is the success probability of the reference point. This DiracDU kernel is symmetric around the target, satisfying Definition 34.2.1 and appropriated for categorical set  $\mathbb{T}$ . See, e.g., Racine and Li (2004) for some uses. Note that  $h = 0$  provides the Dirac kernel.

### Continuous associated kernels

One can find several continuous associated kernels in literature among the Birnbaum-Saunders of Jin and Kawczak (2003). Here, we present seven associated kernels well adapted for the estimations of density or regression functions on any compact or nonnegative support of datasets. All these associated kernels satisfy Definition 34.2.1.

- The extended beta (BE) kernel is defined on  $S_{x,h,a,b} = [a, b] = \mathbb{T}$  with  $a < b < \infty$ ,  $x \in \mathbb{T}$  and  $h > 0$  such that

$$BE_{x,h,a,b}(u) = \frac{(u-a)^{(x-a)/\{(b-a)h\}} (b-u)^{(b-x)/\{(b-a)h\}}}{(b-a)^{1+h^{-1}} B(1+(x-a)/(b-a)h, 1+(b-x)/(b-a)h)} \mathbb{1}_{S_{x,h,a,b}}(u),$$

where  $B(r,s) = \int_0^1 t^{r-1} (1-t)^{s-1} dt$  is the usual beta function with  $r > 0, s > 0$ ; see Libengué (2013). For  $a = 0$  and  $b = 1$ , it corresponds to the beta kernel (Chen, 1999) which is the p.d.f. of the beta distribution with shape parameters  $1+x/h$  and  $(1-x)/h$ . The extended beta kernel is appropriate for any compact support of observations.

- The gamma (GA) kernel is given on  $S_{x,h} = [0, \infty) = \mathbb{T}$  with  $x \in \mathbb{T}$  and  $h > 0$ :

$$GA_{x,h}(u) = \frac{u^{x/h}}{\Gamma(1+x/h) h^{1+x/h}} \exp\left(-\frac{u}{h}\right) \mathbb{1}_{[0,\infty)}(u),$$

where  $\Gamma(v) = \int_0^\infty s^{v-1} \exp(-s) ds$  is the classical gamma function with  $v > 0$ ; see Chen (2000). It is the p.d.f. of the gamma distribution  $\mathcal{G.A}(1+x/h, h)$  with scale parameter  $1+x/h$  and shape parameter  $h$ . It is suitable for the non-negative real set  $\mathbb{T} = [0, \infty)$ .

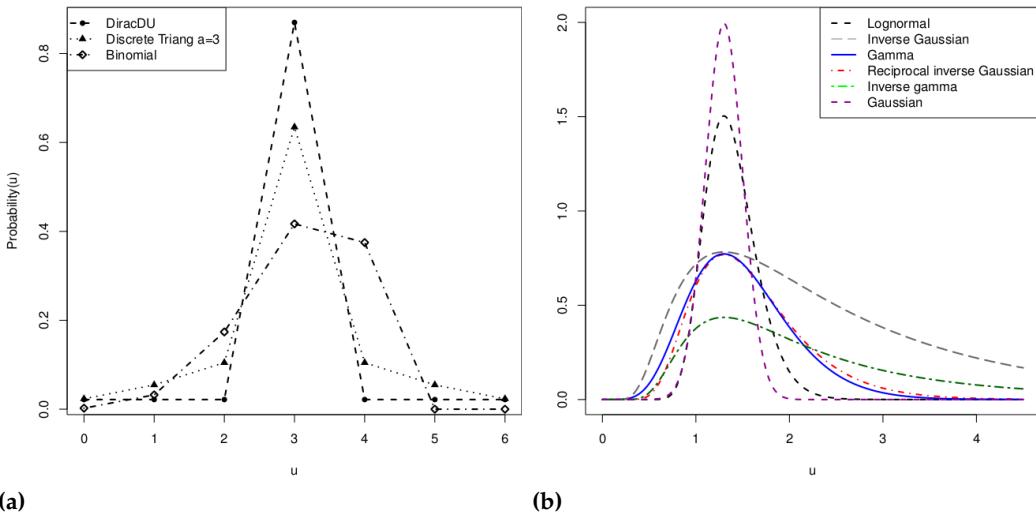
- The lognormal (LN) kernel is defined on  $S_{x,h} = [0, \infty) = \mathbb{T}$  with  $x \in \mathbb{T}$  and  $h > 0$  such that

$$LN_{x,h}(u) = \frac{1}{uh\sqrt{2\pi}} \exp\left\{-\frac{1}{2} \left(\frac{1}{h} \log\left(\frac{u}{x}\right) - h\right)^2\right\} \mathbb{1}_{S_{x,h}}(u);$$

see Libengué (2013) and also Igarashi and Kakizawa (2015). It is the p.d.f. of the classical lognormal distribution with mean  $\log(x) + h^2$  and standard deviation  $h$ .

- The reciprocal inverse Gaussian (RIG) kernel is given on  $S_{x,h} = (0, \infty) = \mathbb{T}$  with  $x \in \mathbb{T}$  and  $h > 0$ :

$$RIG_{x,h}(u) = \frac{1}{\sqrt{2\pi hu}} \exp\left\{-\frac{(x^2 + xh)^{1/2}}{2h} \left(\frac{u}{(x^2 + xh)^{1/2}} - 2 + \frac{(x^2 + xh)^{1/2}}{u}\right)\right\} \mathbb{1}_{S_{x,h}}(u);$$



**Figure 1:** Shapes of univariate (discrete and continuous) associated kernels: (a) DiracDU, discrete triangular  $a = 3$  and binomial with same target  $x = 3$  and bandwidth  $h = 0.13$ ; (b) lognormal, inverse Gaussian, gamma, reciprocal inverse Gaussian, inverse gamma and Gaussian with same target  $x = 1.3$  and  $h = 0.2$ .

see Scaillet (2004), Libengué (2013) and also Igarashi and Kakizawa (2015). It is the p.d.f. of the classical reciprocal inverse Gaussian distribution with mean  $1/\sqrt{x^2 + xh}$  and standard deviation  $1/h$ .

**Remark 34.2.3.** The three continuous associated kernels inverse gamma, inverse Gaussian and Gaussian are not adapted for density estimation on supports  $[0, \infty)$  and thus are not included in the **Ake** package; see Part (b) of Figure 1.

Indeed:

- The inverse gamma (IGA) kernel, defined on  $S_{x,h} = (0, \infty) = \mathbb{T}$  with  $x \in (0, 1/h)$  and  $h > 0$  such that

$$IGA_{x,h}(u) = \frac{h^{1-1/(xh)}}{\Gamma(-1+1/(xh))} u^{-1/(xh)} \exp\left(-\frac{1}{hu}\right) \mathbb{1}_{(0,\infty)}(u)$$

(Libengué, 2013), is graphically the worst since it does not well concentrate on the target  $x$ . Note that it is the p.d.f. of the inverse gamma distribution with scale parameter  $-1 + 1/(xh)$  and scale parameter  $1/h$ .

- Also, the inverse Gaussian (IG) kernel, defined on  $S_{x,h} = (0, \infty) = \mathbb{T}$  with  $x \in (0, 1/3h)$  and  $h > 0$  by

$$IG_{x,h}(u) = \frac{1}{\sqrt{2\pi hu}} \exp \left\{ -\frac{(1-3xh)^{1/2}}{2h} \left( \frac{u}{(1-3xh)^{1/2}} - 2 + \frac{(1-3xh)^{1/2}}{u} \right) \right\} \mathbb{1}_{S_{x,h}}(u)$$

(Scaillet, 2004; Libengué, 2013), has the same graphical properties as the inverse gamma. Note that it is the p.d.f. of the inverse Gaussian distribution  $\mathcal{IG}(1 + x/h, h)$  with scale parameter  $x/(1 - 3xh)^{1/2}$  and shape parameter  $1/h$ .

- From the well known Gaussian kernel  $K^G(u) = (h\sqrt{2\pi})^{-1} \exp(u^2/2h^2) \mathbb{1}_{\mathbb{R}}(u)$ , we define its associated version (Gaussian) on  $S_{x,h} = \mathbb{R}$  with  $x \in \mathbb{T} := \mathbb{R}$  and  $h > 0$ :

$$K_{x,h}^G(u) = \frac{1}{h\sqrt{2\pi}} \exp \left\{ \frac{1}{2} \left( \frac{u-x}{h} \right)^2 \right\} \mathbb{1}_{\mathbb{R}}(u).$$

It has the same shape at any target and thus is well adapted for continuous variables with unbounded supports but not for  $[0, \infty)$  or compact set of  $\mathbb{R}$ ; see also [Epanechnikov \(1969\)](#) for another example of a continuous symmetric kernel.

Figure 1 shows some forms of the above-mentioned univariate associated kernels. The plots highlight the importance given to the target point and around it in discrete (a) and continuous (b) cases. Furthermore, for a fixed bandwidth  $h$ , the Gaussian keeps its same shape along the support; however, they change according to the target for the other non-classical associated kernels. This

Arguments	Description
x	The target.
t	The single or the grid value where the function is computed.
h	The bandwidth or smoothing parameter.
ker	The associated kernel.
a0, a1	The left and right bounds of the support for the extended beta kernel.
a	The arm of the discrete triangular kernel. Default value is 1.
c	The number of categories in DiracDU kernel. Default value is 2.
Result	Description
	Returns a single value of the associated kernel function.

**Table 1:** Summary of arguments and results of kern.fun.

explains the inappropriateness of the Gaussian kernel for density or regression estimation in any bounded interval and of the DiracDU kernel for count regression estimation; see Part (b) of Figure 1. From Part (v) of Remark 34.2.2, the inverse gamma and inverse Gaussian are the worst since they do not well concentrate on the target  $x$ ; see Remark 34.2.3. These previous associated kernels can be applied to various functionals.

We have implemented in R the method kern.fun for both discrete and continuous associated kernels. Seven possibilities are allowed for the kernel function. We enumerate the arguments and results of the default kern.fun.default function in Table 1. The kern.fun is used as follows for the binomial kernel:

```
R> x <- 5
R> h <- 0.1
R> y <- 0:10
R> k_b <- kern.fun(x, y, h, "discrete", "bino")
```

## Density or probability mass function estimations

The p.d.f. or p.m.f. estimation is an usual application of the associated kernels. Let  $X_1, \dots, X_n$  be independent and identically distributed (i.i.d.) random variables with an unknown p.d.f. (respectively p.m.f.)  $f$  on  $\mathbb{T}$ . An associated kernel estimator  $\hat{f}_n$  of  $f$  is simply:

$$\hat{f}_n(x) = \frac{1}{n} \sum_{i=1}^n K_{x,h}(X_i), \quad x \in \mathbb{T}. \quad (5)$$

Here, we point out some pointwise properties of the estimator (5) in both discrete and continuous cases.

**Proposition 34.3.1.** (Kokonendji and Senga Kiessé, 2011; Libengué, 2013) Let  $X_1, X_2, \dots, X_n$  be an  $n$  random sample i.i.d. from the unknown p.m.f. (respectively p.d.f.)  $f$  on  $\mathbb{T}$ . Let  $\hat{f}_n = \hat{f}_{n,h,K}$  be an estimator (5) of  $f$  with an associated kernel. Then, for all  $x \in \mathbb{T}$  and  $h > 0$ , we have

$$\mathbb{E}\{\hat{f}(x)\} = \mathbb{E}\{f(\mathcal{Z}_{x,h})\},$$

where  $\mathcal{Z}_{x,h}$  is the random variable associated to the p.m.f. (respectively p.d.f.)  $K_{x,h}$  on  $\mathbb{S}_{x,h}$ . Furthermore, for a p.m.f. (respectively p.d.f.), we have respectively  $\hat{f}_n(x) \in [0, 1]$  (respectively  $\hat{f}_n(x) > 0$ ) for all  $x \in \mathbb{T}$  and

$$\int_{x \in \mathbb{T}} \hat{f}_n(x) v(dx) = C_n, \quad (6)$$

where  $C_n = C(n; h, K)$  is a positive and finite constant if  $\int_{\mathbb{T}} K_{x,h}(t) v(dx) < \infty$  for all  $t \in \mathbb{T}$ , and  $v$  is a count or Lebesgue measure on  $\mathbb{T}$ .

It is easy to see that  $C_n = 1$  for the estimators (5) with DiracDU kernel or any classical (symmetric)

associated kernel. Indeed, for the DiracDU kernel estimation we have

$$\begin{aligned}\sum_{x=0}^{c-1} \hat{f}_n(x) &= \sum_{x=0}^{c-1} \left\{ (1-h) \mathbb{1}_{\{x\}}(X_1) + \frac{h}{c-1} \mathbb{1}_{S_c \setminus \{x\}}(X_1) \right\} \\ &= (1-h) + \frac{h}{c-1}(c-1) \\ &= 1.\end{aligned}$$

In general we have  $C_n \neq 1$  for other discrete and also continuous associated kernels, but it is always close to 1. In practice, we compute  $C_n$  depending on observations before normalizing  $\hat{f}_n$  to be a p.m.f. or a p.d.f. The following code helps to compute the normalizing constant, e.g., for gamma kernel estimation:

```
R> data("faithful", package = "datasets")
R> x <- faithful$waiting
R> f <- dke.fun(x, ker = "GA", 0.1)
R> f$c_n
[1] 0.9888231
```

Without loss of generality, we study  $x \mapsto \hat{f}_n(x)$  up to a normalizing constant which is used at the end of the density estimation process. Notice that non-classical associated kernel estimators  $\hat{f}_n$  are improper density estimates or as kind of “balloon estimators”; see Sain (2002). There are two ways to normalize these estimators (5). The first method is the *global* normalization using  $C_n$  of (6):

$$\tilde{f}_n(x) = \frac{\hat{f}_n(x)}{\int_{inf(\mathbb{T})}^{sup(\mathbb{T})} \hat{f}_n(x) v(dx)}, \quad x \in \mathbb{T}. \quad (7)$$

Another alternative is to use an *adaptive* normalization of (5) according to each target  $x$ :

$$\tilde{\tilde{f}}_n(x) = \frac{1}{n} \sum_{i=1}^n \frac{K_{x,h}(X_i)}{\int_{inf(\mathbb{T})}^{sup(\mathbb{T})} K_{x,h}(X_i) v(dx)}, \quad x \in \mathbb{T},$$

but this approach, with similar results than (7), is not used here. The representations are done with the global normalization (7). In the package, we also compute the normalizing constant (6) for any data set.

In discrete cases, the integrated squared error (ISE) defined by

$$ISE_0 = \sum_{x \in \mathbb{N}} \{\tilde{f}_n(x) - f_0(x)\}^2,$$

is the criteria used to measure numerically the discrete smoothness of  $\tilde{f}_n$  from (7) with the empirical or naive p.m.f.  $f_0$  such that  $\sum_{x \in \mathbb{N}} f_0(x) = 1$ ; see, e.g., Kokonendji and Senga Kiessé (2011). Concerning the continuous variables, the histogram gives a graphical measure of comparison with  $\tilde{f}_n$ ; see, for example, Figure 2.

### Some theoretical aspects of the normalizing constant

In this section, we present some theoretical aspects of the normalizing constant  $C_n$  of (6) and two examples in the continuous and discrete cases. We first recall the following result on pointwise properties of the estimator (5).

**Lemma 34.3.2.** (Kokonendji and Senga Kiessé, 2011; Libengué, 2013) Let  $x \in \mathbb{T}$  be a target and  $h \equiv h_n$  a bandwidth. Assuming  $f$  is in the class  $\mathcal{C}^2(\mathbb{T})$  in the continuous case, then

$$Bias \left\{ \hat{f}_n(x) \right\} = A(x, h)f'(x) + \frac{1}{2} \left\{ A^2(x, h) + B(x, h) \right\} f''(x) + o(h^2). \quad (8)$$

Similar expressions (8) hold in the discrete case, except that  $f'$  and  $f''$  are finite differences of the first and second order respectively.

Furthermore, for the continuous case, if  $f$  is bounded on  $\mathbb{T}$  then there exists  $r_2 = r_2(K_{x,h}) > 0$  the largest

real number such that  $\|K_{x,h}\|_2^2 := \int_{S_{x,h}} K_{x,h}^2(u) du \leq c_2(x)h^{-r_2}$ ,  $0 \leq c_2(x) \leq \infty$  and

$$\mathbb{V} \left\{ \widehat{f}_n(x) \right\} = \frac{1}{n} f(x) \|K_{x,h}\|_2^2 + o \left( \frac{1}{nh^{r_2}} \right). \quad (9)$$

For discrete situations, the result (9) becomes

$$\mathbb{V} \left\{ \widehat{f}_n(x) \right\} = \frac{1}{n} f(x) [\{\mathbb{P}(\mathcal{Z}_{x,h} = x)\}^2 - f(x)],$$

where  $\mathcal{Z}_{x,h}$  denotes the discrete random variable with p.m.f.  $K_{x,h}$ .

It is noticeable that the bias (8) is bigger than the one with symmetric kernels and thus can be reduced; see, e.g., [Zhang \(2010\)](#), [Zhang and Karunamuni \(2010\)](#) and [Libengué \(2013\)](#).

**Proposition 34.3.3.** Following notations in Lemma 34.3.2, the mean and variance of  $C_n$  of (6) are respectively:

$$\mathbb{E}(C_n) \simeq 1 + \int_{\inf(\mathbb{T})}^{\sup(\mathbb{T})} \left\{ A(x, h)f'(x) + \frac{1}{2} [A^2(x, h) + B(x, h)] f''(x) \right\} v(dx), \quad (10)$$

$$\mathbb{V}(C_n) \simeq \begin{cases} \frac{1}{n} \int_{\inf(\mathbb{T})}^{\sup(\mathbb{T})} (f(x) \|K_{x,h}\|_2^2) dx & \text{if } \mathbb{T} \text{ is continuous,} \\ \frac{1}{n} \sum_{x \in \mathbb{T}} (f(x) [\{\mathbb{P}(\mathcal{Z}_{x,h} = x)\}^2 - f(x)]) & \text{if } \mathbb{T} \text{ is discrete,} \end{cases} \quad (11)$$

where  $\inf(\mathbb{T})$  and  $\sup(\mathbb{T})$  are respectively the infimum and supremum of  $\mathbb{T}$ , the measure  $v$  is Lebesgue or count on the support  $\mathbb{T}$ , and where “ $\simeq$ ” stands for approximation.

**Proof.** From Lemma 34.3.2 and the Fubini theorem, we successively show (10) as follows:

$$\begin{aligned} \mathbb{E}(C_n) &= \mathbb{E} \left( \int_{\inf(\mathbb{T})}^{\sup(\mathbb{T})} \widehat{f}_n(x) v(dx) \right) = \int_{\inf(\mathbb{T})}^{\sup(\mathbb{T})} \mathbb{E} \left( \widehat{f}_n(x) \right) v(dx) \\ &= \int_{\inf(\mathbb{T})}^{\sup(\mathbb{T})} \left( \text{Bias} \left\{ \widehat{f}_n(x) \right\} + f(x) \right) v(dx) \\ &\simeq \int_{\inf(\mathbb{T})}^{\sup(\mathbb{T})} \left\{ A(x, h)f'(x) + \frac{1}{2} [A^2(x, h) + B(x, h)] f''(x) + f(x) \right\} v(dx) \\ &\simeq \int_{\inf(\mathbb{T})}^{\sup(\mathbb{T})} f(x) v(dx) + \int_{\inf(\mathbb{T})}^{\sup(\mathbb{T})} \left\{ A(x, h)f'(x) + \frac{1}{2} [A^2(x, h) + B(x, h)] f''(x) \right\} v(dx) \\ &\simeq 1 + \int_{\inf(\mathbb{T})}^{\sup(\mathbb{T})} \left\{ A(x, h)f'(x) + \frac{1}{2} [A^2(x, h) + B(x, h)] f''(x) \right\} v(dx). \end{aligned}$$

The variance (11) is trivial from Lemma 34.3.2.  $\square$

**Example 1.** Let  $f$  be an exponential density with parameter  $\gamma > 0$ . Thus, one has:

$$\begin{aligned} f(x) &= \gamma \exp(-\gamma x), \\ f'(x) &= -\gamma^2 \exp(-\gamma x), \\ f''(x) &= \gamma^3 \exp(-\gamma x). \end{aligned}$$

Consider the lognormal kernel with  $A(x, h) = x(\exp(3h^2/2) - 1)$ ,  $B(x, h) = x^2 \exp(3h^2)(\exp(h^2) - 1)$  and  $\|LN_{x,h}\|_2^2 = 1/(2\pi h x^{1/2})$ ; see [Libengué \(2013\)](#). Then, using the Taylor formula around  $h$ , the expressions of  $\mathbb{E}(C_n)$  and  $\mathbb{V}(C_n)$  are:

$$\mathbb{E}(C_n) \simeq 1 + \left[ \left( -1 + \exp \frac{3h^2}{2} \right) + \frac{1}{2} \left\{ \left( -1 + \exp \frac{3h^2}{2} \right)^2 + \exp 3h^2 (-1 + \exp h^2) \right\} \right] \simeq 1 - \frac{h^2}{2}$$

and  $\mathbb{V}(C_n) \simeq \gamma(2nh\sqrt{\pi})^{-1} \int_0^\infty z^{-1} \exp(-z) dz$  with  $\int_0^\infty z^{-1} \exp(-z) dz \approx 16.2340$  by computation with R. Thus, the quantity  $C_n$  cannot be equal to 1.

**Example 2.** Let  $f$  be a Poisson p.m.f. with parameter  $\lambda$  and thus  $f(x) = \lambda^x \exp(-\lambda)/x!$ . The finite differences  $f^{(k)}(x)$  of order  $k \in \{1, 2, \dots\}$  at  $x \in \mathbb{N}$  are given by the recursive relation:

$$f^{(k)}(x) = \{f^{(k-1)}(x)\}^{(1)} \text{ with } f^{(1)}(x) = \begin{cases} \{f(x+1) - f(x-1)\}/2, & \text{if } x \in \mathbb{N} \setminus \{0\}, \\ f(1) - f(0), & \text{if } x = 0, \end{cases}$$

and

$$f^{(2)}(x) = \begin{cases} \{f(x+2) - 2f(x) + f(x-2)\}/4, & \text{if } x \in \mathbb{N} \setminus \{0,1\}, \\ \{f(3) - 3f(1) + f(0)\}/4, & \text{if } x = 1, \\ \{f(2) - 2f(1) + f(0)\}/2, & \text{if } x = 0. \end{cases}$$

Considering the binomial kernel with  $A(x, h) = x + h$ ,  $B(x, h) = (x + h)(1 - h)/(x + 1)$ , we successively obtain

$$\begin{aligned} \mathbb{E}(C_n) &\simeq 1 + \frac{h}{4} \left( f(3) + 3f(2) - 3f(1) - 2f(0) \right. \\ &\quad + \sum_{x=2}^{\infty} 2 \{f(x+1) - f(x-1)\} + \frac{(x+3)\{f(x+2) - 2f(x) + f(x-2)\}}{x+1} \Big) \\ &\quad + \left( \frac{3f(3) + 8f(2) - 17f(1) + 3f(0)}{2} \right. \\ &\quad \left. + \sum_{x=2}^{\infty} \frac{x\{f(x+1) - f(x-1)\}}{2} + \frac{(x^3 + x^2 + 1)\{f(x+2) - 2f(x) + f(x-2)\}}{x+1} \right) \\ &\simeq 1 + \frac{h \exp(-\lambda)}{4} \left[ \frac{\lambda^3}{3!} + 3 \frac{\lambda^2}{2!} - 3\lambda - 2 \right. \\ &\quad \left. + \sum_{x=2}^{\infty} \left\{ \frac{2\lambda^{x+1}}{(x+1)!} - \frac{2\lambda^{x-1}}{(x-1)!} + \frac{x+3}{x+1} \left( \frac{\lambda^{x+2}}{(x+2)!} - 2 \frac{\lambda^x}{x!} + \frac{\lambda^{x-2}}{(x-2)!} \right) \right\} \right] \\ &\quad + \left[ \frac{\lambda^3}{4} + 2\lambda^2 - \frac{17\lambda}{2} + 3 \right. \\ &\quad \left. + \sum_{x=2}^{\infty} \left\{ \frac{x}{2} \left( \frac{\lambda^{x+1}}{(x+1)!} - \frac{\lambda^{x-1}}{(x-1)!} \right) + \frac{x^3 + x^2 + 1}{x+1} \left( \frac{\lambda^{x+2}}{(x+2)!} - 2 \frac{\lambda^x}{x!} + \frac{\lambda^{x-2}}{(x-2)!} \right) \right\} \right] \end{aligned}$$

and

$$\mathbb{V}(C_n) \simeq \frac{\exp(-\lambda)}{n} \sum_{x \in \mathbb{T}} \left( \frac{\lambda^x}{x!} \left[ \left\{ (1+h) \left( \frac{x+h}{x+1} \right)^x \right\}^2 - \frac{\lambda^x \exp(-\lambda)}{x!} \right] \right).$$

## Bandwidth selection

Now, we consider the bandwidth selection problems which are generally crucial in nonparametric estimation. Several methods already existing for continuous kernels can be adapted to the discrete case as the classical least-squares cross-validation method; see, for example, [Bowman \(1984\)](#), [Marron \(1987\)](#) and references therein. Here, we simply propose three procedures for the bandwidth selection: cross-validation, Bayesian local for binomial and adaptive for the gamma kernel. Also, a review of bayesian bandwidth selection methods is presented. Each time, the smoothing parameter selection is done with the non-normalized version  $\hat{f}_n$  of the estimator (5) before the global normalization  $\tilde{f}_n$  of (7).

## Cross-validation for any associated kernel

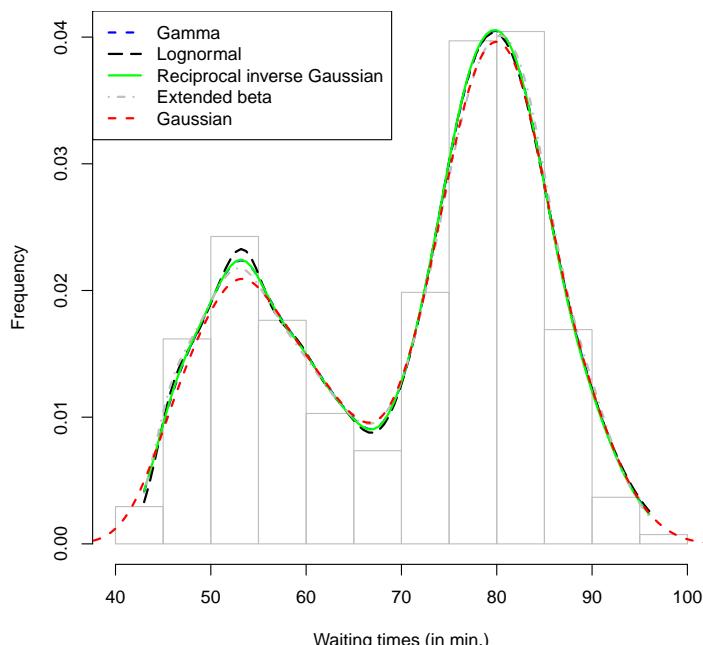
For a given associated kernel  $K_{x,h}$  with  $x \in \mathbb{T}$  and  $h > 0$ , the optimal bandwidth  $h_{cv}$  of  $h$  is obtained by cross-validation as  $h_{cv} = \arg \min_{h>0} CV(h)$  with

$$CV(h) = \int_{x \in \mathbb{T}} \left\{ \hat{f}_n(x) \right\}^2 \nu(dx) - \frac{2}{n} \sum_{i=1}^n \hat{f}_{n,-i}(X_i),$$

where  $\hat{f}_{n,-i}(X_i) = (n-1)^{-1} \sum_{j \neq i} K_{X_i,h}(X_j)$  is being computed as  $\hat{f}_n(X_i)$  by excluding the observation  $X_i$  and  $\nu$  is the Lebesgue or count measure. This method is applied to all estimators (5) with associated kernels cited in this paper, independently on the support  $\mathbb{T}$  of  $f$  to be estimated.

[Table 2](#) gives the arguments and results of the cross-validation function `hcvc.fun` defined for continuous data are below. The `hcvd.fun` is the corresponding function for discrete data. The `hcvc.fun` is performed with the Old Faithful geyser data described in [Azzalini and Bowman \(1990\)](#) and [Härdle \(2012\)](#). The dataset concerns waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA. The following codes and [Figure 2](#) give smoothing density estimation with various associated kernels of the waiting time variable.

Arguments	Description
Vec	The positive continuous data sample.
seq.bws	The sequence of bandwidths where to compute the cross-validation function.
ker	The associated kernel.
a0, a1	The bounds of the support of extended beta kernel. Default values are respectively 0 and 1.
a	The arm of the discrete triangular kernel. Default value is 1.
c	The number of categories in DiracDU kernel. Default value is 2.
Results	Description
hcv	The optimal bandwidth obtained by cross-validation.
seq.h	The sequence of bandwidths used to compute hcv.
CV	The values of the cross-validation function.

**Table 2:** Summary of arguments and results of hcvc.fun.**Figure 2:** Smoothing density estimation of the Old Faithful geyser data (Azzalini and Bowman, 1990) by some continuous associated kernels with the support of observations  $[43, 96] = \mathbb{T}$ .

```
R> data("faithful", package = "datasets")
R> x <- faithful$waiting
R> f1 <- dke.fun(x, 0.1, "continuous", ker = "GA")
R> f2 <- dke.fun(x, 0.036, "continuous", ker = "LN")
R> f3 <- dke.fun(x, 0.098, "continuous", ker = "RIG")
R> f4 <- dke.fun(x, 0.01, "continuous", ker = "BE", a0 = 40, a1 = 100)
R> t <- seq(min(x), max(x), length.out = 100)
R> hist(x, probability = TRUE, xlab = "Waiting times (in min.)",
+       ylab = "Frequency", main = "", border = "gray")
R> lines(t, f1$fn, lty = 2, lwd = 2, col = "blue")
R> lines(t, f2$fn, lty = 5, lwd = 2, col = "black")
R> lines(t, f3$fn, lty = 1, lwd = 2, col = "green")
R> lines(t, f4$fn, lty = 4, lwd = 2, col = "grey")
R> lines(density(x, width = 12), lty = 8, lwd = 2, col = "red")
R> legend("topleft", c("Gamma", "Lognormal", "Reciprocal inverse Gaussian",
+ "Extended beta", "Gaussian"), col = c("blue", "black", "green", "grey", "red"),
+ lwd = 2, lty = c(2, 5, 1, 4, 8), inset = .0)
```

## A review of Bayesian bandwidth selection

Bayesian inference grows out of the simple formula known as Bayes rule. Assume we have two random variables  $A$  and  $B$ . A principle rule of probability theory known as the chain rule allows us to specify the joint probability of  $A$  and  $B$  taking on particular values  $a$  and  $b$ ,  $P(a, b)$ , as the product of the conditional probability that  $A$  will take on value  $a$  given that  $B$  has taken on value  $b$ ,  $P(a|b)$ , and the marginal probability that  $B$  takes on value  $b$ ,  $P(b)$ . Which gives us:

$$\text{Joint probability} = \text{Conditional Probability} \times \text{Marginal Probability}.$$

$$\text{Thus we have: } P(a, b) = P(a|b)P(b).$$

This expression (Bayes rule) indicates that we can compute the conditional probability of a variable  $A$  given the variable  $B$  from the conditional probability of  $B$  given  $A$ . This introduces the notion of prior and posterior knowledge.

**Prior and posterior knowledge.** A prior probability is the probability available to us beforehand, and before making any additional observations. A posterior probability is the probability obtained from the prior probability after making additional observation to the prior knowledge available.

**Summarizing the Bayesian approach.** The Bayesian approach to parameter estimation works as follows:

1. Formulate our knowledge about a situation.
2. Gather data.
3. Obtain posterior knowledge that updates our beliefs.

How do we formulate our knowledge about a situation?

- a. Define a distribution model which expresses qualitative aspects of our knowledge about the situation. This model will have some unknown parameters, which will be dealt with as random variables.
- b. Specify a prior probability distribution which expresses our subjective beliefs and subjective uncertainty about the unknown parameters, before seeing the data.

After gathering the data, how do we obtain posterior knowledge?

- c. Compute posterior probability distribution which estimates the unknown parameters using the rules of probability and given the observed data, presenting us with updated beliefs.

[Zougab et al. \(2013\)](#) proposed a Bayesian approach based upon a likelihood cross-validation approximation and a Markov chain Monte Carlo (MCMC) method for deriving the global optimal bandwidth using the famous binomial kernel. However, a global bandwidth does not generally provide a good estimator for complex p.m.f.'s, in particular for small and moderate sample sizes. Generally, the global discrete associated kernel estimator tends to simultaneously under- and oversmooth  $f(x)$ .

In order to improve the global discrete associated kernel estimator, in particular for complex count data with small and moderate sample sizes, [Zougab et al. \(2012\)](#) and [Zougab et al. \(2013\)](#) adapted two versions of variable bandwidths for discrete associated kernel estimator and proposed Bayesian approaches for selecting these variable bandwidths. Note that these two versions are originally proposed for kernel density estimation (see, e.g., [Sain and Scott 1996](#); [Breiman et al. 1977](#); [Abramson 1982](#); [Brewer 2000](#); [Zhang et al. 2006](#); [Zougab et al. 2014a](#) and [Zhang et al. 2016](#)).

Recently, [Zougab et al. \(2012\)](#) have considered the local discrete associated kernel estimator (balloon estimator in discrete case) and have derived the closed form of the variable bandwidth at each point  $x$  for which the p.m.f. is estimated by considering the binomial kernel estimator and locally treating the bandwidth as a random quantity with a beta prior distribution. This approach outperforms existing classical global methods, namely, MISE and CV in particular for small and moderate sample sizes. [Zougab et al. \(2013\)](#) have also proposed the adaptive discrete associated kernel estimator (sample-point estimator in discrete situation), which replaces  $h$  by  $h_i$  for each observation  $x_i$  with  $i = 1, \dots, n$ , and then employs the Bayesian approach for estimating the adaptive bandwidths  $h_i$ . The authors have considered the binomial kernel and the beta prior for each variable bandwidth  $h_i$ , and have shown that this approach performs better than the popular classical global selectors.

### Bayesian estimation of localized bandwidth for the binomial kernel

An alternative to the cross-validation for bandwidth selection is by using Bayesian methods. These methods have been investigated with three different procedures: local, global and adaptive; see respectively Zougab et al. (2012, 2013, 2014b). In terms of integrated squared error and execution times, the local Bayesian outperforms the other Bayesian procedures. In the local Bayesian framework, the variable bandwidth is treated as parameter with prior  $\pi(\cdot)$ . Under squared error loss function, the Bayesian bandwidth selector is the posterior mean; see Zougab et al. (2012).

First, as we have mentioned above,  $f(x)$  can be approximated by

$$f(x|h) = f_h(x) = \sum_{u \in \mathbb{T}} f(u) B_{x,h}(u) = \mathbb{E}\{B_{x,h}(X)\},$$

where  $B_{x,h}$  is the binomial kernel and  $X$  is a random variable with p.m.f.  $f$ . Now, considering  $h$  as a scale parameter for  $f_h(x)$ , the local approach consists of using  $f_h(x)$  and constructing a Bayesian estimator for  $h(x)$ .

Indeed, let  $\pi(h)$  denote the beta prior density of  $h$  with positive parameters  $\alpha$  and  $\beta$ . By the Bayes theorem, the posterior of  $h$  at the point of estimation  $x$  takes the form

$$\pi(h|x) = \frac{f_h(x)\pi(h)}{\int f_h(x)\pi(h)dh}.$$

Since  $f_h$  is unknown, we use  $\hat{f}_h$  as natural estimator of  $f_h$ , and hence we can estimate the posterior by

$$\pi(h|x, X_1, X_2, \dots, X_n) = \frac{\hat{f}_h(x)\pi(h)}{\int \hat{f}_h(x)\pi(h)dh}.$$

Under the squared error loss, the Bayes estimator of the smoothing parameter  $h(x)$  is the posterior mean and is given by  $\hat{h}_n(x) = \int h\hat{\pi}(h|x, X_1, X_2, \dots, X_n)dh$ . Exact approximation is

$$\hat{h}_n(x) = \frac{\sum_{i=0}^n \sum_{k=0}^{X_i} \frac{x^k}{(x+1-X_i)!k!(X_i-k)!} B(X_i + \alpha - k + 1, x + \beta + 1 - X_i)}{\sum_{i=0}^n \sum_{k=0}^{X_i} \frac{x^k}{(x+1-X_i)!k!(X_i-k)!} B(X_i + \alpha - k, x + \beta + 1 - X_i)}, \quad \forall x \in \mathbb{N} \text{ with } X_i \leq x + 1,$$

where  $B(\cdot, \cdot)$  is the beta function; see Zougab et al. (2012) for more details.

### Bayesian estimation of adaptive bandwidth for the gamma kernel

The bandwidth  $h$  in the gamma kernel density estimation can be allowed to be adaptive. This approach gives a variable bandwidth  $h_i$  for each observation  $X_i$  in place of the initial fixed bandwidth  $h$ . Following Zougab et al. (2014a), we suggest using Bayesian methods to estimate such adaptive bandwidths or variable bandwidths  $h_i, i = 1, \dots, n$ . Thus, we treat  $h_i$  as a random variable with a prior distribution  $\pi(\cdot)$ . The estimator (5) with gamma kernel of Section Continuous associated kernels and variable bandwidths are reformulated as

$$\hat{f}_n(x) = \frac{1}{n} \sum_{i=1}^n GA_{x,h_i}(X_i). \quad (12)$$

The leave-one-out kernel estimator of  $f(X_i)$  deduced from (12) is

$$\hat{f}(X_i | \{X_{-i}\}, h_i) = \frac{1}{n-1} \sum_{j=1, j \neq i}^n GA_{X_i, h_i}(X_j), \quad (13)$$

where  $\{X_{-i}\}$  denotes the set of observations excluding  $X_i$ . The posterior distribution for each variable bandwidth  $h_i$  given  $X_i$  provided from the Bayesian rule is expressed as follow

$$\pi(h_i | X_i) = \frac{\hat{f}(X_i | \{X_{-i}\}, h_i)\pi(h_i)}{\int_0^\infty \hat{f}(X_i | \{X_{-i}\}, h_i)\pi(h_i)dh_i}. \quad (14)$$

We obtain the Bayesian estimator  $\tilde{h}_i$  of  $h_i$  by using the quadratic loss function

$$\tilde{h}_i = \mathbb{E}(h_i | X_i). \quad (15)$$

In the following, we assume that each  $h_i = h_i(n)$  has an inverse gamma prior distribution  $\text{IGA}(\alpha, \beta)$  with the shape parameter  $\alpha > 0$  and scale parameter  $\beta > 0$ . The density of  $\text{IGA}(a, b)$  with  $a, b > 0$  is defined as

$$\Phi_{a,b}(z) = \frac{b^a}{\Gamma(a)} z^{-a-1} \exp(-b/z) \mathbb{1}_{(0,\infty)}(z). \quad (16)$$

This allows us to obtain the closed form of the posterior density and the Bayesian estimator given by the following result.

**Theorem 34.3.4.** For fixed  $i \in \{1, 2, \dots, n\}$ , consider each observation  $X_i$  with its corresponding bandwidth  $h_i$ . Using the gamma kernel estimator (12) and the inverse gamma prior distribution  $\text{IGA}(\alpha, \beta)$  given in (16) with  $\alpha > 1/2$  and  $\beta > 0$  for each  $h_i$ , then:

- (i) The posterior density (14) is the following weighted sum of inverse gamma

$$\pi(h_i | X_i) = \frac{1}{D_{ij}} \sum_{j=1, j \neq i}^n \left\{ A_{ij} \Phi_{\alpha+1/2, B_{ij}}(h_i) \mathbb{1}_{(0,\infty)}(X_i) + C_j \Phi_{\alpha+1, X_j+\beta}(h_i) \mathbb{1}_{\{0\}}(X_i) \right\}$$

with  $A_{ij} = [\Gamma(\alpha + 1/2)] / (\beta^\alpha X_i^{1/2} \sqrt{2\pi} B_{ij}^{\alpha+1/2})$ ,  $B_{ij} = X_i \log X_i - X_i \log X_j + X_j - X_i + \beta$ ,

$C_j = \Gamma(\alpha + 1) / [\beta^{-\alpha} (X_j + \beta)^{\alpha+1}]$  and  $D_{ij} = \sum_{j=1, j \neq i}^n \left\{ A_{ij} \mathbb{1}_{(0,\infty)}(X_i) + C_j \mathbb{1}_{\{0\}}(X_i) \right\}$ .

- (ii) The Bayesian estimator  $\tilde{h}_i$  of  $h_i$ , given in (15), is

$$\tilde{h}_i = \frac{1}{D_{ij}} \sum_{j=1, j \neq i}^n \left\{ \frac{A_{ij} B_{ij}}{\alpha - 1/2} \mathbb{1}_{(0,\infty)}(X_i) + \frac{(X_j + \beta) C_j}{\alpha} \mathbb{1}_{\{0\}}(X_i) \right\}$$

according to the previous notations of  $A_{ij}$ ,  $B_{ij}$ ,  $C_j$  and  $D_{ij}$ .

**Proof.** (i) Let us represent  $\pi(h_i | X_i)$  of (14) as the ratio of  $N(h_i | X_i) := \hat{f}(X_i | \{X_{-i}\}, h_i) \pi(h_i)$  and  $\int_0^\infty N(h_i | X_i) dh_i$ . From (13) and (16) the numerator is, first, equal to

$$\begin{aligned} N(h_i | X_i) &= \left( \frac{1}{n-1} \sum_{j=1, j \neq i}^n G A_{X_j, h_i}(X_j) \right) \left( \frac{\beta^\alpha}{\Gamma(\alpha)} h_i^{-\alpha-1} \exp(-\beta/h_i) \right) \\ &= \frac{[\Gamma(\alpha)]^{-1}}{(n-1)} \sum_{j=1, j \neq i}^n \frac{G A_{X_j, h_i}(X_j)}{\beta^{-\alpha} h_i^{\alpha+1}} \exp(-\beta/h_i). \end{aligned} \quad (17)$$

Following Chen (2000), we assume that for all  $X_i \in (0, \infty)$  one has  $1 + (X_i/h_i) \rightarrow \infty$  as  $n \rightarrow \infty$ . Using the Stirling formula  $\Gamma(z+1) \simeq \sqrt{2\pi} z^{z+1/2} \exp(-z)$  as  $z \rightarrow \infty$ , the term of the sum in (17) can be successively calculated as

$$\begin{aligned} \frac{G A_{X_j, h_i}(X_j)}{\beta^{-\alpha} h_i^{\alpha+1}} \exp(-\beta/h_i) &= \frac{X_j^{(X_i/h_i)} \exp(-X_j/h_i)}{h_i^{1+(X_i/h_i)} \Gamma[1 + (X_i/h_i)] \beta^{-\alpha} h_i^{\alpha+1}} \exp(-\beta/h_i) \\ &= \frac{\exp[-(X_j + \beta - X_i \log X_j)/h_i]}{\beta^{-\alpha} h_i^{(X_i/h_i)+\alpha+2} \sqrt{2\pi} \exp(-X_i/h_i) (X_i/h_i)^{(X_i/h_i)+1/2}} \\ &= \frac{\Gamma(\alpha + 1/2)}{\beta^{-\alpha} X_i^{1/2} \sqrt{2\pi} B_{ij}^{\alpha+1/2}} \times \frac{B_{ij}^{\alpha+1/2} \exp[-B_{ij}/h_i]}{h_i^{\alpha+3/2} \Gamma(\alpha + 1/2)} \\ &= A_{ij} \Phi_{\alpha+1/2, B_{ij}}(h_i), \end{aligned} \quad (18)$$

with  $B_{ij} = X_i \log X_i - X_i \log X_j + X_j - X_i + \beta$ ,  $A_{ij} = [X_j^{-1} \Gamma(\alpha + 1/2)] / (\beta^{-\alpha} X_i^{-1/2} \sqrt{2\pi} B_{ij}^{\alpha+1/2})$  and  $\Phi_{\alpha+1/2, B_{ij}}(h_i)$  is given in (16).

Also, for  $X_i = 0$ , the term of the sum (17) can be expressed as follows

$$\begin{aligned} \frac{GA_{0,h_i}(X_j)}{\beta^{-\alpha}h_i^{\alpha+1}} \exp(-\beta/h_i) &= \frac{\exp(-X_j/h_i)}{\beta^{-\alpha}h_i^{\alpha+2}} \exp(-\beta/h_i) \\ &= \frac{\Gamma(\alpha+1)}{\beta^{-\alpha}(X_j+\beta)^{\alpha+1}} \times \frac{(X_j+\beta)^{\alpha+1} \exp[-(X_j+\beta)/h_i]}{h_i^{\alpha+2}\Gamma(\alpha+1)} \\ &= C_j \Phi_{\alpha+1,X_j+\beta}(h_i), \end{aligned} \quad (19)$$

with  $C_j = \Gamma(\alpha+1)/[\beta^{-\alpha}(X_j+\beta)^{\alpha+1}]$  and  $\Phi_{\alpha+1,X_j+\beta}(h_i)$  is given in (16). Combining (18) and (19), the expression of  $N(h_i | X_i)$  in (17) becomes

$$N(h_i | X_i) = \frac{[\Gamma(\alpha)]^{-1}}{(n-1)} \sum_{j=1,j \neq i}^n \left\{ A_{ij} \Phi_{\alpha+1/2,B_{ij}}(h_i) \mathbb{1}_{(0,\infty)}(X_i) + C_j \Phi_{\alpha+1,X_j+\beta}(h_i) \mathbb{1}_{\{0\}}(X_i) \right\}. \quad (20)$$

From (20), the denominator is successively computed as follows:

$$\begin{aligned} \int_0^\infty N(h_i | X_i) dh_i &= \frac{[\Gamma(\alpha)]^{-1}}{(n-1)} \sum_{j=1,j \neq i}^n \left( A_{ij} \int_0^\infty \Phi_{\alpha+1/2,B_{ij}}(h_i) \mathbb{1}_{(0,\infty)}(X_i) dh_i \right. \\ &\quad \left. + C_j \int_0^\infty \Phi_{\alpha+1,X_j+\beta}(h_i) \mathbb{1}_{\{0\}}(X_i) dh_i \right) \\ &= \frac{[\Gamma(\alpha)]^{-1}}{(n-1)} \sum_{j=1,j \neq i}^n \left\{ A_{ij} \mathbb{1}_{(0,\infty)}(X_i) + C_j \mathbb{1}_{\{0\}}(X_i) \right\} \\ &= \frac{[\Gamma(\alpha)]^{-1}}{(n-1)} D_{ij}, \end{aligned} \quad (21)$$

with  $D_{ij} = \sum_{j=1,j \neq i}^n (A_{ij} \mathbb{1}_{(0,\infty)}(X_i) + C_j \mathbb{1}_{\{0\}}(X_i))$ . Finally, the ratio of (20) and (21) leads to the result of Part (i).

(ii) Let us remember that the mean of the inverse gamma distribution  $\mathcal{IG}(\alpha, \beta)$  is  $\beta/(\alpha - 1)$ . Thus, the expression of  $\pi(h_i | X_i)$  in (14) is given by

$$\pi(h_i | X_i) = \frac{1}{D_{ij}} \sum_{j=1,j \neq i}^n \left\{ A_{ij} \Phi_{\alpha+1/2,B_{ij}}(h_i) \mathbb{1}_{(0,\infty)}(X_i) + C_j \Phi_{\alpha+1,X_j+\beta}(h_i) \mathbb{1}_{\{0\}}(X_i) \right\}$$

and, therefore,  $\tilde{h}_i = \mathbb{E}(h_i | X_i) = \int_0^\infty h_i \pi(h_i | X_i) dh_i$  is finally

$$\tilde{h}_i = \mathbb{E}(h_i | X_i) = \frac{1}{D_{ij}} \sum_{j=1,j \neq i}^n \left\{ \frac{A_{ij}B_{ij}}{\alpha - 1/2} \mathbb{1}_{(0,\infty)}(X_i) + \frac{(X_j + \beta)C_j}{\alpha} \mathbb{1}_{\{0\}}(X_i) \right\}.$$

□

This new method of selecting bandwidth by the Bayesian adaptive procedure will be implemented in a future version of the **Ake** package.

## Bandwidth selection for kernel regression involving associated kernels

One of the most often encountered models in nonparametric statistics is the regression model. The function that provides the best prediction of a dependent variable  $y$  in terms of an independent variable  $x$  is the conditional expectation  $\mathbb{E}(y/x) = m(x)$ . This is called regression function and its estimation from a sequence of  $n$  pairs  $(x_i, y_i)$ ,  $i = 1, \dots, n$  is a problem in statistics. We will consider the case  $(x, y) \in \mathbb{T} \times \mathbb{R}$ . For simplicity, we take  $\mathbb{T} = \mathbb{R}$  if  $x$  is a continuous variable and  $\mathbb{T} = \mathbb{N}$  in the discrete case. The classical non parametric regression model between two variables  $y$  and  $x$  is

$$y_i = m(x_i) + \epsilon_i, \quad (22)$$

where  $\mathbf{y} = (y_1, \dots, y_n)$  is a response vector,  $\mathbf{x} = (x_1, \dots, x_n)$  is an explanatory vector,  $\epsilon = (\epsilon_1, \dots, \epsilon_n)$  is the error following a Gaussian distribution with zero mean and finite variance  $\sigma^2$ , i.e.,  $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$  and  $m : \mathbb{T} \mapsto \mathbb{R}$  is the unknown regression function. Several methods have been proposed to estimate the regression function in the continuous case. We cite for example the histograms introduced by Tukey (1961) and studied by Geoffroy (1980) and Lecoutre (1990), the spline method which can be found in Reinsch (1967), Silverman (1985) and Wahba (1990) and also the regression using partition

proposed by Breiman et al. (1984).

As for the density or probability mass function, the estimate of the regression function by the kernel method is the most used because of its good asymptotic properties and interest in practice. Introduced initially for continuous density estimation by Rosenblatt (1956) and Parzen (1962), this method was adopted by Nadaraya (1964) and Watson (1964) for estimating the continuous regression function. It was also applied to smooth the discrete regression function  $m$  for  $x \in \mathbb{N}$ . Some studies have been done to estimate the discrete regression function, using the Dirac-type kernel (naive estimator) or discrete kernels of Aitchison and Aitken (1976). However, the naive estimator is appropriate only when the sample size is large, and the discrete kernel of Aitchison and Aitken (1976) is only suitable for categorical data; see Hayfield and Racine (2008) and also Hayfield and Racine (2014). Kokonendji et al. (2009) adapted the Nadaraya (1964) and Watson (1964) kernel to the discrete unknown function  $m$ , using the discrete associated kernels. In their work, using the integrated mean square error and the coefficient of determination  $R^2$ , they showed that the binomial or discrete triangular kernels are better compared to the optimal Epanechnikov kernel. In this section we present the theoretical foundations of the estimated regression function with continuous and discrete associated kernels.

Both in continuous and discrete cases, consider the relation between a response variable  $Y$  and an explanatory variable  $x$  given by

$$Y = m(x) + \epsilon, \quad (23)$$

where  $m$  is an unknown regression function from  $\mathbb{T} \subseteq \mathbb{R}$  to  $\mathbb{R}$  and  $\epsilon$  the disturbance term with null mean and finite variance. Let  $(X_1, Y_1), \dots, (X_n, Y_n)$  be a sequence of i.i.d. random variables on  $\mathbb{T} \times \mathbb{R} (\subseteq \mathbb{R}^2)$  with  $m(x) = \mathbb{E}(Y|X=x)$  of (23). Using (continuous or discrete) associated kernels, the Nadaraya (1964) and Watson (1964) estimator  $\hat{m}_n$  of  $m$  is

$$\hat{m}_n(x; h) = \sum_{i=1}^n \frac{Y_i K_{x,h}(X_i)}{\sum_{i=1}^n K_{x,h}(X_i)} = \hat{m}_n(x), \quad \forall x \in \mathbb{T} \subseteq \mathbb{R}, \quad (24)$$

where  $h \equiv h_n$  is the smoothing parameter such that  $h_n \rightarrow 0$  as  $n \rightarrow \infty$ .

Besides the criterion of kernel support, we retain the root mean squared error (RMSE) and also the practical coefficient of determination given respectively by

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n \{Y_i - \hat{m}_n(X_i)\}^2}$$

and

$$R^2 = \frac{\sum_{i=1}^n \{\hat{m}_n(X_i) - \bar{y}\}^2}{\sum_{i=1}^n (Y_i - \bar{y})^2},$$

with  $\bar{y} = n^{-1}(Y_1 + \dots + Y_n)$ .

In discrete cases, the reg.fun function for (24) is used with the binomial kernel on milk data as follows. This dataset is about average daily fat (kg/day) yields from milk of a single cow for each of the first 35 weeks.

```
R> data("milk", package = "Ake")
R> x <- milk$week
R> y <- milk$yield
R> h <- reg.fun(x, y, "discrete", "bino", 0.1)
R> h

Bandwidth h:0.1           Coef_det=0.9726

Number of points: 35;      Kernel = Binomial

data          y
Min. : 1.0  Min. :0.0100
1st Qu.: 9.5  1st Qu.:0.2750
Median :18.0  Median :0.3600
Mean   :18.0  Mean   :0.3986
3rd Qu.:26.5  3rd Qu.:0.6150
Max.   :35.0  Max.   :0.7200
eval.points      m_n
Min. : 1.0  Min. :0.01542
1st Qu.: 9.5  1st Qu.:0.27681
```

Arguments	Description
Vec	The explanatory data sample can be discrete or continuous.
y	The response variable.
ker	The associated kernel.
h	The sequence of bandwidths where to compute the optimal bandwidth.
a0, a1	The bounds of the support of extended beta kernel. Default values are respectively 0 and 1.
a	The arm of the discrete triangular kernel. Default value is 1.
c	The number of categories in DiracDU kernel. Default value is 2.
Results	Description
kernel	The associated kernel.
hcv	The optimal bandwidth obtained by cross-validation.
CV	The values of the cross-validation function.
seqbw	The sequence of bandwidths used to compute hcv.

**Table 3:** Summary of arguments and results of `hcvreg.fun`.

```
Median :18.0 Median :0.35065
Mean   :18.0 Mean   :0.39777
3rd Qu.:26.5 3rd Qu.:0.60942
Max.   :35.0 Max.   :0.70064
```

The above `reg.fun` is also used for continuous cases; see Figure 3 and Table 4 for the motorcycle impact data of [Silverman \(1985\)](#).

## Bandwidth selection

We present two bandwidth selection methods for the regression: the well-known cross-validation for any associated kernel and the Bayesian global for the binomial kernel.

### Cross-validation for any associated kernel

For a given associated kernel, the optimal bandwidth parameter is  $\hat{h}_{cv} = \arg \min_{h>0} LSCV(h)$  with

$$LSCV(h) = \frac{1}{n} \sum_{i=1}^n \{Y_i - \hat{m}_{-i}(X_i)\}^2, \quad (25)$$

where  $\hat{m}_{-i}(X_i)$  is computed as  $\hat{m}_n$  of (24) excluding  $X_i$ ; see, e.g., [Kokonendji et al. \(2009\)](#). The `hcvreg.fun` function to compute this optimal bandwidth is described in Table 3.

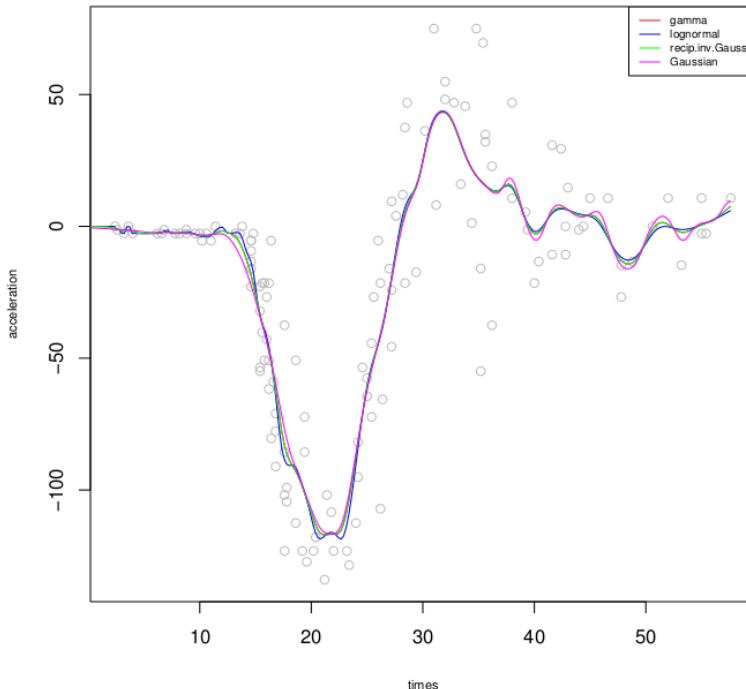
The following code helps to compute the bandwidth parameter by cross-validation on milk data. The associated kernel used is the discrete triangular kernel with arm  $a = 1$ .

```
R> data("milk", package = "Ake")
R> x <- milk$week
R> y <- milk$yield
R> f <- hcvreg.fun(x, y, type_data = "discrete", ker = "triang", a = 1)
R> f$hcv
[1] 1.141073
```

When we consider the continuous associated kernel, one needs to set the type of data parameter to “continuous” in the `hcvreg.fun` function. Thus, the `hcvreg.fun` and `reg.fun` functions are used with gamma, lognormal, reciprocal inverse Gaussian and Gaussian kernel on the motor cycle impact data described in [Silverman \(1985\)](#). The observations consist of accelerometer reading taken through time in an experimentation on the efficiency of crash helmets. The results in Table 4 agree with the shapes of continuous associated kernels of Part (b) of Figure 1; see also Figure 3. In fact, since the lognormal kernel is well concentrated around the target  $x$ , it gives the best  $R^2$  which is 75.9%. The gamma and the reciprocal inverse Gaussian kernels give similar  $R^2$  in the order 73%. Although the Gaussian kernel is well concentrated on the target, it gives the lower result of  $R^2 = 70.90\%$ . This is mainly due to the symmetry of the kernel which cannot change its shapes according to the target.

	Gamma	Lognormal	Rec. Inv. Gaussian	Gaussian
R <sup>2</sup>	0.7320	0.7591	0.7328	0.7090

**Table 4:** Some expected values of R<sup>2</sup> of nonparametric regressions of the motor cycle impact data (Silverman, 1985) by some continuous associated kernels.



**Figure 3:** Nonparametric regressions of the motors cycle impact data (Silverman, 1985) by some continuous associated kernels.

### Bayesian global for binomial kernel

Using Bayes theorem, the joint posterior distribution of  $h$  given the observations is

$$\pi(h|X_1, X_2, \dots, X_n) \propto h^{\alpha-1}(1-h)^{\beta-1} \left( \frac{1}{2} \sum_{i=1}^n \{y_i - \hat{m}_{-i}(X_i)\}^2 + b \right)^{-(n+2a)/2},$$

where  $\propto$  denotes proportional, the reals  $a$  and  $b$  are the parameters of the inverse gamma distribution  $IG(a, b)$ , and  $\alpha$  and  $\beta$  those of the beta distribution  $Be(\alpha, \beta)$ . The estimate  $\hat{h}_{bay}$  of the smoothing parameter  $h$  is given by Markov chain Monte Carlo (MCMC) techniques with Gibbs sampling:

$$\hat{h}_{bay} = \frac{1}{N - N_0} \sum_{N_0+1}^N h^{(t)},$$

where  $N_0$  is the burn-in period and  $N$  the number of iterations; see Zougab et al. (2014b) for further details. It will be implemented in a future version of the **Ake** package.

### Summary and final remarks

The **Ake** package offers easy tools for R users whose research involves kernel estimation of density functions and/or regression functions through associated kernels that are capable of handling all categorical, count and real positive datasets. Figure 1 shows the importance of the associated kernel choice as well as the bandwidth selection. In fact, symmetric (e.g., Gaussian) kernel estimators (respectively empirical estimators) are not suitable for bounded or positive continuous datasets (respectively discrete small samples). We then need an appropriate associated kernel. The binomial kernel is suitable for small size count data while the discrete triangular or the naive kernel are more indicated for large sample sizes. In continuous cases, the lognormal and gamma kernels give the best

estimation for positive data while the extended beta is suitable for any compact support.

This package includes various continuous and discrete associated kernels. It also contains functions to handle the bandwidth selection problems through cross-validation, local and global Bayesian procedures for binomial kernel and also the adaptive Bayesian procedure for the gamma kernel. In general, Bayesian choices of smoothing parameters will be better than their cross-validation counterparts. Future versions of the package will contain Bayesian methods with other associated kernels. Also, these associated kernels are useful for heavy tailed data p.d.f. estimation and can be added later in the package; see, e.g., Ziane et al. (2015). The case of multivariate data needs to be taken in consideration; see Kokonendji and Somé (2015) for p.d.f. estimation and Somé and Kokonendji (2016) for regression. We think that the **Ake** package can be of interest to nonparametric practitioners of different applied settings.

## Acknowledgments

We sincerely thank the CRAN editors and two anonymous reviewers for their valuable comments. Part of this work was done while the first author was at “Laboratoire de Mathématiques de Besançon” as a Visiting Scientist, with the support of “Agence Universitaire de la Francophonie” (AUF).

## Bibliography

- I. S. Abramson. On bandwidth variation in kernel estimates – A square root law. *The Annals of Statistics*, 10(4):1217–1223, 1982. doi: 10.1214/aos/1176345986. [p267]
- J. Aitchison and C. G. Aitken. Multivariate binary discrimination by the kernel method. *Biometrika*, 63 (3):413–420, 1976. doi: 10.2307/2335719. [p258, 260, 271]
- A. Azzaolini and A. W. Bowman. A look at some data on the old faithful geyser. *Applied Statistics*, 39(3): 357–365, 1990. doi: 10.2307/2347385. [p265, 266]
- A. W. Bowman. An alternative method of cross-validation for the smoothing of density estimates. *Biometrika*, 71(2):353–360, 1984. doi: 10.1093/biomet/71.2.353. [p265]
- L. Breiman, W. Meisel, and E. Purcell. Variable kernel estimates of multivariate densities. *Technometrics*, 19(2):135–144, 1977. doi: 10.2307/1268623. [p267]
- L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. CRC Press, 1984. [p271]
- M. J. Brewer. A Bayesian model for local smoothing in kernel density estimation. *Statistics and Computing*, 10(4):299–309, 2000. doi: 10.1023/a:1008925425102. [p267]
- S. X. Chen. Beta kernel estimators for density functions. *Computational Statistics & Data Analysis*, 31(2): 131–145, 1999. doi: 10.1016/s0167-9473(99)00010-9. [p260]
- S. X. Chen. Probability density function estimation using gamma kernels. *Annals of the Institute of Statistical Mathematics*, 52(3):471–480, 2000. doi: 10.1023/a:1004165218295. [p260, 269]
- T. Duong. ks: Kernel density estimation and kernel discriminant analysis for multivariate data in R. *Journal of Statistical Software*, 21(7):1–16, 2007. doi: 10.18637/jss.v021.i07. [p258]
- V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969. doi: 10.1137/1114019. [p261]
- G. L. Geoffroy. Synthesis, molecular dynamics, and reactivity of mixed-metal clusters. *Accounts of Chemical Research*, 13(12):469–476, 1980. doi: 10.1021/ar50156a006. [p270]
- W. Härdle. *Smoothing Techniques: With Implementation in S*. Springer-Verlag, 2012. doi: 10.1007/978-1-4612-4432-5. [p265]
- T. Hayfield and J. S. Racine. Nonparametric econometrics: The np package. *Journal of Statistical Software*, 27(5):1–32, 2008. doi: 10.18637/jss.v027.i05. [p258, 271]
- T. Hayfield and J. S. Racine. *np: Nonparametric Kernel Smoothing Methods for Mixed Datatypes*, 2014. URL <https://CRAN.R-project.org/package=np>. R package version 0.60-2. [p271]
- G. Igarashi and Y. Kakizawa. Bias corrections for some asymmetric kernel estimators. *Journal of Statistical Planning and Inference*, 159:37–63, 2015. doi: 10.1016/j.jspi.2014.11.003. [p260, 261]

- X. Jin and J. Kawczak. Birnbaum-Saunders and lognormal kernel estimators for modelling durations in high frequency financial data. *Annals of Economics and Finance*, 4:103–124, 2003. [p260]
- C. C. Kokonendji and T. Senga Kiessé. Discrete associated kernels method and extensions. *Statistical Methodology*, 8(6):497–516, 2011. doi: 10.1016/j.stamet.2011.07.002. [p258, 259, 260, 262, 263]
- C. C. Kokonendji and S. M. Somé. On multivariate associated kernels for smoothing some density function. *arXiv:1502.01173*, 2015. URL <https://arxiv.org/abs/1502.01173>. [p274]
- C. C. Kokonendji and S. S. Zocchi. Extensions of discrete triangular distributions and boundary bias in kernel estimation for discrete functions. *Statistics & Probability Letters*, 80(21):1655–1662, 2010. doi: 10.1016/j.spl.2010.07.008. [p260]
- C. C. Kokonendji, T. Senga Kiessé, and S. S. Zocchi. Discrete triangular distributions and non-parametric estimation for probability mass function. *Journal of Nonparametric Statistics*, 19(6-8): 241–254, 2007. doi: 10.1080/10485250701733747. [p260]
- C. C. Kokonendji, T. Senga Kiessé, and C. G. B. Demétrio. Appropriate kernel regression on a count explanatory variable and applications. *Advances and Applications in Statistics*, 12(1):99–125, 2009. doi: 10.1007/s00180-015-0627-1. [p271, 272]
- J.-P. Lecoutre. Uniform consistency of a class of regression function estimators for Banach-space valued random variable. *Statistics & Probability Letters*, 10(2):145–149, 1990. doi: 10.1016/0167-7152(90)90010-5. [p270]
- F. G. Libengué. *Méthode Non-Paramétrique des Noyaux Associés Mixtes et Applications*. PhD thesis, Besançon, 2013. [p258, 259, 260, 261, 262, 263, 264]
- J. Marron. A comparison of cross-validation techniques in density estimation. *The Annals of Statistics*, pages 152–162, 1987. doi: 10.1214/aos/1176350258. [p265]
- E. A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964. doi: 10.1137/1109020. [p271]
- E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962. doi: 10.1214/aoms/1177704472. [p271]
- J. Racine and Q. Li. Nonparametric estimation of regression functions with both categorical and continuous data. *Journal of Econometrics*, 119(1):99–130, 2004. doi: 10.1016/s0304-4076(03)00157-x. [p260]
- C. H. Reinsch. Smoothing by spline functions. *Numerische Mathematik*, 10(3):177–183, 1967. doi: 10.1007/bf02162161. [p270]
- M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956. doi: 10.1214/aoms/1177728190. [p271]
- S. R. Sain. Multivariate locally adaptive density estimation. *Computational Statistics & Data Analysis*, 39 (2):165–186, 2002. doi: 10.1016/s0167-9473(01)00053-6. [p263]
- S. R. Sain and D. W. Scott. On locally adaptive density estimation. *Journal of the American Statistical Association*, 91(436):1525–1534, 1996. doi: 10.2307/2291578. [p267]
- O. Scaillet. Density estimation using inverse and reciprocal inverse Gaussian kernels. *Nonparametric Statistics*, 16(1-2):217–226, 2004. doi: 10.1080/10485250310001624819. [p261]
- B. W. Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 47(1):1–52, 1985. doi: 10.2307/2982831. [p270, 272, 273]
- S. M. Somé and C. C. Kokonendji. Effects of associated kernels in nonparametric multiple regressions. *Journal of Statistical Theory and Practice*, 10(2):456–471, 2016. doi: 10.1080/15598608.2016.1160010. [p274]
- J. W. Tukey. Discussion, emphasizing the connection between analysis of variance and spectrum analysis. *Technometrics*, 3(2):191–219, 1961. doi: 10.1080/00401706.1961.10489940. [p270]
- G. Wahba. *Spline Models for Observational Data*, volume 59. SIAM, 1990. doi: 10.1137/1.9781611970128. [p270]

- G. S. Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372, 1964. [p271]
- S. Zhang. A note on the performance of the gamma kernel estimators at the boundary. *Statistics & Probability Letters*, 80(7):548–557, 2010. doi: 10.1016/j.spl.2009.12.009. [p264]
- S. Zhang and R. J. Karunamuni. Boundary performance of the beta kernel estimators. *Journal of Nonparametric Statistics*, 22(1):81–104, 2010. doi: 10.1080/10485250903124984. [p264]
- X. Zhang, M. L. King, and R. J. Hyndman. A Bayesian approach to bandwidth selection for multivariate kernel density estimation. *Computational Statistics & Data Analysis*, 50(11):3009–3031, 2006. doi: 10.1016/j.csda.2005.06.019. [p267]
- X. Zhang, M. L. King, and H. L. Shang. Bayesian bandwidth selection for a nonparametric regression model with mixed types of regressors. *Econometrics*, 4(2):24, 2016. doi: 10.3390/econometrics4020024. [p267]
- Y. Ziane, S. Adjabi, and N. Zougab. Adaptive Bayesian bandwidth selection in asymmetric kernel density estimation for nonnegative heavy-tailed data. *Journal of Applied Statistics*, 42(8):1645–1658, 2015. doi: 10.1080/02664763.2015.1004626. [p258, 274]
- N. Zougab, S. Adjabi, and C. Kokonendji. Binomial kernel and Bayes local bandwidth in discrete function estimation. *Journal of Nonparametric Statistics*, 24(3):783–795, 2012. doi: 10.1080/10485252.2012.678847. [p258, 259, 267, 268]
- N. Zougab, S. Adjabi, and C. C. Kokonendji. A Bayesian approach to bandwidth selection in univariate associate kernel estimation. *Journal of Statistical Theory and Practice*, 7(1):8–23, 2013. doi: 10.1080/15598608.2013.756286. [p267, 268]
- N. Zougab, S. Adjabi, and C. C. Kokonendji. Bayesian estimation of adaptive bandwidth matrices in multivariate kernel density estimation. *Computational Statistics & Data Analysis*, 75:28–38, 2014a. doi: 10.1016/j.csda.2014.02.002. [p267, 268]
- N. Zougab, S. Adjabi, and C. C. Kokonendji. Bayesian approach in nonparametric count regression with binomial kernel. *Communications in Statistics – Simulation and Computation*, 43(5):1052–1063, 2014b. doi: 10.1080/03610918.2012.725145. [p268, 273]

Wanbitching E. Wansouwé  
The University of Maroua  
Higher Teachers' Training College  
Department of Computer Science  
P.O. Box 55 Maroua, Cameroon  
[ericwansouwe@gmail.com](mailto:ericwansouwe@gmail.com)

Sobom M. Somé  
Université Bourgogne Franche-Comté  
Laboratoire de Mathématiques de Besançon  
16 route de Gray, 25030 Besançon cedex, France  
[sobom.some@univ-fcomte.fr](mailto:sobom.some@univ-fcomte.fr)

Célestin C. Kokonendji  
Université Bourgogne Franche-Comté  
Laboratoire de Mathématiques de Besançon  
16 route de Gray, 25030 Besançon cedex, France  
[celestin.kokonendji@univ-fcomte.fr](mailto:celestin.kokonendji@univ-fcomte.fr)

# An Introduction to Principal Surrogate Evaluation with the **pseval** Package

by Michael C. Sachs, Erin E. Gabriel

**Abstract** We describe a new package called **pseval** that implements the core methods for the evaluation of principal surrogates in a single clinical trial. It provides a flexible interface for defining models for the risk given treatment and the surrogate, the models for integration over the missing counterfactual surrogate responses, and the estimation methods. Estimated maximum likelihood and pseudo-score can be used for estimation, and the bootstrap for inference. A variety of post-estimation methods are provided, including print, summary, plot, and testing. We summarize the main statistical methods that are implemented in the package and illustrate its use from the perspective of a novice R user.

## Introduction

A valid principal surrogate endpoint, also called a specific nonmechanistic correlate of protection (Plotkin and Gilbert, 2012), can be used as a target for treatment improvement in early phase trials and, in the specific setting of evaluation, for predicting individual treatment effects post-licensure. A surrogate is considered to be valid if it provides reliable predictions of treatment effects on the clinical endpoint of interest. Frangakis and Rubin (2002) introduced the concept of principal stratification and the definition of a principal surrogate (PS). Informally, a post-treatment intermediate response variable is a principal surrogate if causal effects of the treatment on the clinical outcome only exist when causal effects of the treatment on the intermediate variable exist. The criteria for a PS have been modified and extended in more recent works, with most current literature focusing on wide effect modification as the primary criterion of interest.

The goal of PS evaluation is estimation and testing of how treatment efficacy on the clinical outcome of interest varies over subgroups defined by possible treatment and surrogate combinations of interest; this is an effect modification objective. The combinations of interest are called the principal strata and they include a set of unobservable counterfactual responses: responses that would have occurred under a set of conditions counter to the observed conditions. To finesse this problem of unobservable responses, a variety of clever trial designs and estimation approaches have been proposed. Several of these have been implemented in the **pseval** package (Sachs and Gabriel, 2016).

## Methods

### Notation

Let  $Z_i$  be the treatment indicator for subject  $i$ , where 0 indicates the control or standard treatment, and 1 indicates the experimental treatment. We currently only allow for two levels of treatment and assume that the treatment assignments are randomized. Let  $S_i$  be the observed value of the intermediate response for subject  $i$ . Since  $S_i$  can be affected by treatment, there are two naturally occurring counterfactual values of  $S_i$ :  $S_i(1)$  under treatment, and  $S_i(0)$  under control. Let  $s_z$  be the realization of the random variable  $S(z)$ , for  $z \in \{0, 1\}$ . The outcome of interest is denoted  $Y_i$ . We consider the counterfactual values of  $Y_i(0)$  and  $Y_i(1)$ . We allow for continuous, binary, count, and time-to-event outcomes, thus  $Y_i$  may be a vector containing a time variable and an event/censoring indicator, i.e.  $Y_i = (T_i, \Delta_i)$  where  $\Delta_i = 1$  if  $T_i$  is an event time, and  $\Delta_i = 0$  if  $T_i$  is a censoring time. In event driven settings,  $S_i(z)$  is only defined if the event,  $Y_i(z)$ , does not occur before the potential surrogate  $S_i(z)$  is measured at a fixed time  $\tau$  after entry into the study. The data analyses only include participants who have not experienced the event outcome by time  $\tau$ .

### Estimands

Criteria for  $S$  to be a good surrogate are based on risk estimands that condition on the potential intermediate responses. The risk is defined as a mapping  $g$  of the cumulative distribution function of  $Y(z)$  conditional on the intermediate responses. The joint risk estimands conditions on the candidate surrogate under both level of treatment,  $(S(1), S(0))$ .

$$\text{risk}_1(s_1, s_0) = g \{ F_{s_1} [Y(1)|S(0) = s_0, S(1) = s_1] \},$$

$$\text{risk}_0(s_1, s_0) = g \{ F_{s_1} [Y(0)|S(0) = s_0, S(1) = s_1] \}.$$

For instance, for a binary outcome, the risk function may simply be the probability  $risk_z(s_1, s_0) = P(Y(z) = 1|S(0) = s_0, S(1) = s_1)$ , or for a time-to-event outcome the risk function may be the cumulative distribution function  $risk_z(s_1, s_0) = P(Y(z) \leq t|S(0) = s_0, S(1) = s_1)$ .

Currently we focus only on marginal risk estimands which condition only on  $S(1)$ , the intermediate response or biomarker under active treatment:

$$\begin{aligned} risk_1(s_1) &= g \{F_{s_1}[Y(1)|S(1) = s_1]\}, \\ risk_0(s_1) &= g \{F_{s_1}[Y(0)|S(1) = s_1]\}. \end{aligned}$$

Neither of the joint risk estimands are identifiable in a standard randomized trial, as either  $S(0)$  or  $S(1)$  or both will be missing for each subject. In the special case where  $S(0)$  is constant, such as the immune response to HIV antigens or Hep B in the placebo arm of a vaccine trial, the joint and marginal risk estimands are equivalent. This special case is referred to as case constant biomarker (CB) in much of the literature ([Gilbert and Hudgens, 2008](#)); i.e.,  $S_i(0) = c$  for subjects  $i$ . This may occur outside the vaccine setting when one considers the AUC of a treatment drug as a surrogate; those receiving placebo will have no drug and therefore all placebo AUC will be 0 or undefined. Under assumptions given below, and in the case CB setting, the marginal risk estimand is identifiable in the treatment arm; it is not identifiable in the control arm without further assumptions or trial augmentation ([Wolfson and Gilbert, 2010](#)).

There are specific trial augmentations that allow for the measurement or imputation of the missing counterfactual  $S_s$ , in the control and treatment arms. As well, Under one of these augmentations case CB can sometimes be induced by considering a function of the a candidate surrogate for evaluation. Greater detail on this point given below.

Specification of the distributions of  $Y(z)|S(1)$  determines the likelihood, we will denote this as  $f(y|\beta, s_1, z)$ . If  $S(1)$  were fully observed, simple maximum likelihood estimation could be used. The key challenge in estimating these risk estimands is solving the problem of conditioning on counterfactual values that are not observable for at least a subset of subjects in a randomized trial. This involves integrating out missing values based on some model, and under some set of assumptions and/or trial augmentations.

### Prinicipal surrogate criteria

[Frangakis and Rubin \(2002\)](#) gave a single criterion for a biomarker  $S$  to be a PS: causal effects of the treatment on the clinical outcome only exist when causal effects of the treatment on the intermediate variable exist. In general this can only be evaluated using the joint risk estimands, which consider not only the counterfactual values of the biomarker under treatment, but also under control  $S(0)$ . However, in the special case where all  $S(0)$  values are constant, say at level  $C$ , such as an immune response to HIV in a HIV negative population pre-vaccination this criteria, often referred to as average causal necessity (ACN), can be written in terms of the marginal risk estimands as:

$$risk_1(C) = risk_0(C).$$

More recently, other works [Gilbert and Hudgens \(2008\)](#), [Wolfson and Gilbert \(2010\)](#), [Huang and Gilbert \(2011\)](#), [Huang et al. \(2013\)](#), [Gabriel and Gilbert \(2014\)](#), and [Gabriel and Follmann \(2016\)](#) have suggested that this criterion is both too restrictive and in some cases can be vacuously true. Instead most current works suggest that the wide effect modification (WEM) criterion is of primary importance, ACN being of secondary importance. WEM is given formally in terms of the risk estimands and a known contrast function  $h$  satisfying  $h(x, y) = 0$  if and only if  $x = y$  by:

$$|h(risk_1(s_1), risk_0(s_1)) - h(risk_1(s_1^*), risk_0(s_1^*))| > \delta,$$

for at least some  $s_1 \neq s_1^*$  and  $\delta > 0$ , with the larger the  $\delta$  the better the surrogate. Examples of contrast functions are the treatment efficacy,  $h(x, y) = 1 - x/y$ , and the risk difference  $h(x, y) = x - y$ . To evaluate WEM and ACN we need to identify the risk estimands, which condition on data that is missing for at least half of the subjects in a standard randomized trial.

### Augmentation and assumptions

We first make three standard assumptions used in much of the literature for absorbing events outcomes:

- Stable Unit Treatment Value Assumption (SUTVA): Observations on the independent units in the trial should be unaffected by the treatment assignment of other units.

- Ignorable Treatment Assignment: The observed treatment assignment does not change the counterfactual clinical outcome.
- Equal individual risk up to the time of candidate surrogate measurement  $\tau$ .

In time-to-event settings one more assumption is needed:

- Non-informative censoring.

It should be noted that the equal individual risk assumption requires that time-to-event analysis start at time  $\tau$ , rather than at randomization.

[Wolfson and Gilbert \(2010\)](#) outlines how these assumptions are needed for identification of the risk estimands. Now to deal with the missing  $S(1)$  values among those with  $Z = 0$ , we next focus on three trial augmentations: Baseline immunogenicity predictor (BIP), closeout placebo vaccination (CPV), a concept that was extend to the setting of general treatment trials under the name of closeout control treatment (CCT) in [Gabriel and Follmann \(2016\)](#), and baseline surrogate measurement (BSM). For further details on these augmentations, we refer you to [Follmann \(2006\)](#), [Gilbert and Hudgens \(2008\)](#), [Gabriel and Gilbert \(2014\)](#), and for further augmenations not yet implemented to [Gabriel and Follmann \(2016\)](#).

## BIP

Briefly, a BIP  $W$  is any baseline measurement or set of measurements that is highly correlated with  $S$ . It is particularly useful if  $W$  is unlikely to be associated with the clinical outcome after conditioning on  $S$ , i.e.  $Y \perp W|S(1)$ ; some of the methods leverage this assumption. The BIP  $W$  is used to integrate out the missing  $S(1)$  among those with  $Z = 0$  based on a model for  $S(1)|W$  that is estimated among those with  $Z = 1$ . We describe how this model is used in the next section.

The assumptions needed for a BIP to be useful depend on the risk model used. If the BIP is included in the risk model, only the assumption of no interaction with treatment and the candidate surrogate are needed. However, if the BIP is not included in the risk model, the assumption that that clinical outcome is independent of the BIP given the candidate surrogate is needed. Although not a requirement for identification of the risk estimands, it has been found in most simulations studies that a correlation between the BIP and  $S(1)$  of greater than 0.7 is needed for unbiased estimation in finite samples.

## CPV or CCT

Under a CPV or CCT augmented design, control recipients that do not have events, or all willing control subjects for a non-event driven clinical outcome, are given the experimental treatment at the end of the follow-up period. Then, their intermediate response is measured at some time after that treatment. This measurement is then used as a direct imputation for the missing  $S(1)$ . The CPV augmentation was developed in the setting of vaccine trials, where the surrogate is an immune response and the outcome is infection. One set of conservative assumptions to use CPV as a direct imputation for  $S(1)$  in a vaccine trial are given in [Wolfson and Gilbert \(2010\)](#) are:

- Individual time constancy of the true intermediate response under active treatment,  $S(1) = S_{CPV}$  almost surely, for placebo recipients that are crossed over at the end of the trial, where  $S_{CPV}$  is the measurement of the candidate surrogate after crossover treatment of the placebo subjects.
- No events (infections) during the close-out period.

In the general treatment trial setting, the CCT augmentation can be used under the same Individual time constancy assumption, and the assumption that drop-out or unwillingness to receive close-out treatment is completely at random.

## BSM

[Gabriel and Gilbert \(2014\)](#) suggested the baseline augmentation BSM, which is a pre-treatment measurement of the candidate PS, denoted  $S_B$ . The BSM may be a good predictor of  $S(1)$  without any further assumptions. It can be used in the same way as a BIP. Alternatively you can transform  $S(1) - S_B$  and use this as the candidate surrogate, further increasing the association with the BSM/BIP. Under the BSM assumption outlined in [Gabriel and Gilbert \(2014\)](#);

- Time constancy of the true intermediate response under control,

then  $S(0) = S_{BSM}$  almost surely. You do not need this assumption to use a BSM, but if it holds then it induces the CB case, thus the joint and marginal risk estimands are equivalent.

## Risk estimation

### Estimated maximum likelihood

Let  $f(y|\beta, s_1, z)$  denote the density of  $Y|S(1), Z$  with parameters  $\beta$ . Further let  $R_i$  denote the indicator for missingness in  $S_i(1)$ . We proceed to estimate  $\beta$  by maximizing

$$\prod_{i=1}^n \{f(Y_i|\beta, S_i(1), Z_i)\}_i^R \left\{ \int f(Y_i|\beta, s, Z_i) d\hat{F}_{S(1)|W}(s|W_i) \right\}^{1-R_i}$$

with respect to  $\beta$ .

This procedure is called estimated maximum likelihood (EML) and was developed in [Pepe and Fleming \(1991\)](#). The key idea is that we are averaging the likelihood contributions for subjects missing  $S(1)$  with respect to the estimated distribution of  $S(1)|W$ , denoted by  $\hat{F}_{S(1)|W}(s|W_i)$ . The model for this distribution is referred to as the integration model. Recall that a BIP  $W$  that is strongly associated with  $S(1)$  is needed for adequate performance.

Closed-form inference is not available for EML estimates, thus we recommend use of the bootstrap for estimation of standard errors. It was suggested as an approach to principal surrogate evaluation by [Gilbert and Hudgens \(2008\)](#) and [Huang and Gilbert \(2011\)](#).

### Pseudoscore

[Huang et al. \(2013\)](#) suggest a different estimation procedure that does have a closed form variance estimator. Instead of numerically optimizing the estimated likelihood, the pseudoscore approach iteratively finds the solution to weighted versions of the score equations. Pseudoscore estimates were also suggested in [Wolfson \(2009\)](#) and implemented for several special cases in [Huang et al. \(2013\)](#). We have implemented here only one of the special cases: categorical BIP and binary  $Y$  ( $S$  may be continuous or categorical). In addition to having closed form variance estimators, it has been argued that the pseudo-score estimators are more efficient than the EML estimators. The closed form variance estimates are not yet implemented.

### Package features

Typically, users would have to code up the likelihood, integration model, and perform the optimization themselves. This is beyond the reach of many researchers who desire to use these methods. The goal of **pseval** is to correctly implement these methods with a flexible and user-friendly interface, enabling researchers to implement and interpret a wide variety of models.

The **pseval** package allows users to specify the type of augmented design that is used in their study, specify the form of the risk model along with the distribution of  $Y|S(1)$ , and specify different integration models to estimate the distribution of  $S(1)|W$ . Then the likelihood can be maximized and bootstraps run. Post-estimation summaries are available to display and analyze the treatment efficacy as a function of  $S(1)$ . All of this is implemented with a flexible and familiar interface.

## Package information

### Usage

Here we will walk through some basic analyses from the point of view of a new R user. Along the way we will highlight the main features of **pseval**. We support binary, continuous, count, and time-to-event outcomes, thus we will also need to load the **survival** package ([Therneau, 2015](#); [Therneau and Grambsch, 2000](#)).

### Example dataset

First let's create an example dataset. The **pseval** package provides the function `generate_example_data` which takes a single argument: the sample size.

```

set.seed(1492)
fakedata <- generate_example_data(n = 800)
head(fakedata)

##   Z      BIP      CPV      BSM      S.obs  time.obs
## 1 0  0.3353179 1.4851399 0.45961614 0.35268095 0.3301972
## 2 0  1.4536863 2.6379400 1.39591042 1.46688905 0.1195136
## 3 0 -0.7243934    NA -0.62723499 -0.73190763 0.2631222
## 4 0 -0.1183592 0.9421504 0.07738308 -0.01833409 0.1373458
## 5 0 -0.2352566    NA -0.14971448 -0.18470242 0.8543703
## 6 0 -0.7782851 0.1159434 -0.65721609 -0.66313714 0.2200481
##   event.obs Y.obs      S.obs.cat      BIP.cat
## 1          0    0 (-0.198,0.503] (0.0574,0.766]
## 2          1    0 (1.36, Inf] (0.766, Inf]
## 3          1    1 (-Inf,-0.198] (-Inf,-0.678]
## 4          1    0 (-0.198,0.503] (-0.678,0.0574]
## 5          1    1 (-0.198,0.503] (-0.678,0.0574]
## 6          1    0 (-Inf,-0.198] (-Inf,-0.678]

```

The example data includes both a time-to-event outcome, a binary outcome, a surrogate, a BIP, CPV, and BSM, and a categorical version of the surrogate. The true model for the time is exponential, with parameters (intercept) = -1, S(1) = 0.0, Z = 0.0, S(1):Z = -0.75. The true model for binary is logistic, with the same parameter values.

In the above table S.obs.cat and BIP.cat are formed as S.obs.cat <- factor(S.obs, levels=c(-Inf, quantile(c(S.0, S.1), c(.25, .5, .75), na.rm = TRUE), Inf)) and similarly for BIP.cat. Alternatively a user could input arbitrary numeric values to represent different discrete subgroups (e.g., 0s and 1s to denote 2 subgroups).

### The "psdesign" object

We begin by creating a "psdesign" object with the synonymous function. This is the object that combines the raw dataset with information about the study design and the structure of the data. Subsequent analysis will operate on this psdesign object. It is designed to be analogous to the svydesign function in the [survey](#) package (Lumley, 2014, 2004). The first argument is the data frame where the data are stored. All subsequent arguments describe the mappings from the variable names in the data frame to important variables in the PS analysis, using the same notation as above. Other covariates or variables can be mapped to arbitrary variable names using the same syntax. An optional weights argument describes the sampling weights, if present. Our first analysis will use the binary version of the outcome, with continuous S.1 and the BIP labeled *BIP*. The object has a print method, so we can inspect the result.

```

binary.ps <- psdesign(data = fakedata, Z = Z, Y = Y.obs, S = S.obs, BIP = BIP)
binary.ps

## Augmented data frame: 800 obs. by 6 variables.
##   Z Y S.1      S.0 cdfweights      BIP
## 1 0 0  NA  0.3527          1  0.335
## 2 0 0  NA  1.4669          1  1.454
## 3 0 1  NA -0.7319          1 -0.724
## 4 0 0  NA -0.0183          1 -0.118
## 5 0 1  NA -0.1847          1 -0.235
## 6 0 0  NA -0.6631          1 -0.778
##
## Empirical TE: 0.526
##
## Mapped variables:
##   Z  ->  Z
##   Y  ->  Y.obs
##   S  ->  S.obs
##   BIP ->  BIP
##
## Integration models:
##   None present, see ?add_integration for information on integration models.
##
```

```
## Risk models:
## None present, see ?add_riskmodel for information on risk models.
## No estimates present, see ?ps_estimate.
## No bootstraps present, see ?ps_bootstrap.
```

The printout displays a brief description of the data, including the empirical treatment efficacy estimate, the variables used in the analysis and their corresponding variables in the original dataset. Finally the printout invites the user to see the help page for `add_integration`, in order to add an integration model to the `psdesign` object, the next step in the analysis.

Missing values in the  $S$  variable are allowed. Note that any cases where  $S(1)$  is missing will be integrated over in the likelihood or score equations. Thus any cases that experienced an event prior to the time  $\tau$  when the surrogate was measured should be excluded from the dataset. The equal individual risk assumption allows us to make causal inferences even after excluding such cases.

`psdesign` easily accommodates case-control or case-cohort sampling. In this case, the surrogate  $S$  is only measured on a subset of the data, inducing missingness in  $S$  by design. Let's modify the fake dataset to see how it works. We're going to sample all of the cases, and 20% of the controls for measurement of  $S$ .

```
fakedata.cc <- fakedata
missdex <- sample((1:nrow(fakedata.cc))[fakedata.cc$Y.obs == 0],
  size = floor(sum(fakedata.cc$Y.obs == 0) * .8))
fakedata.cc[missdex, ]$S.obs <- NA
fakedata.cc$weights <- ifelse(fakedata.cc$Y.obs == 1, 1, .2)
```

Now we can create the "psdesign" object, using the entire dataset (including those missing  $S$ .obs) and passing the weights to the `weights` field.

```
binary.cc <- psdesign(data = fakedata.cc, Z = Z, Y = Y.obs, S = S.obs,
  BIP = BIP, weights = weights)
```

The other augmentation types can be defined by mapping variables to the names `BIP`, `CPV`, and/or `BSM`. The augmentations are handled as described in the previous section: `CPV` is used as a direct imputation for  $S(1)$ , and `BSM` is used as a direct imputation for  $S(0)$ . `BIPs` and `BSMs` are made available in the augmented dataset for use in the integration models which we describe in the next subsection.

For survival outcomes, a key assumption is that the potential surrogate is measured at a fixed time  $\tau$  after entry into the study. Any subjects who have a clinical outcome prior to  $\tau$  will be removed from the analysis, with a warning. If `tau` is not specified in the `psdesign` object, then it is assumed to be 0. Survival outcomes are specified by mapping `Y` to a `Surv` object, which requires the `survival` package:

```
surv.ps <- psdesign(data = fakedata, Z = Z, Y = Surv(time.obs, event.obs), S = S.obs,
  BIP = BIP, CPV = CPV, BSM = BSM)

## Warning in psdesign(data = fakedata, Z = Z, Y = Surv(time.obs,
## event.obs), : tau missing in psdesign: assuming that the
## surrogate S was measured at time 0.
```

## Integration models

The EML procedure requires an estimate of  $F_{S(1)|W}$ , and we refer to this as the integration model. Details are available in the help page for `add_integration`. Several integration models are implemented, including a parametric model that uses a formula interface to define a regression model, a semiparametric model that specifies a location and a scale model is robust to the specification of the distribution, and a non-parametric model that uses empirical conditional probability estimates for discrete  $W$  and  $S(1)$ .

For this first example, let's use the parametric integration model. We specify the mean model for  $S(1)|W$  as a formula. The predictor is generally a function of the `BIP` and the `BSM`, if available. We can add the integration model directly to the `psdesign` object and inspect the results. Note that in the formula, we refer to the variable names in the augmented dataset.

```
binary.ps <- binary.ps + integrate_parametric(S.1 ~ BIP)
binary.ps

## Augmented data frame: 800 obs. by 6 variables.
##   Z   Y   S.1     S.0 cdfweights    BIP
```

```

## 1 0 0 NA 0.3527      1 0.335
## 2 0 0 NA 1.4669      1 1.454
## 3 0 1 NA -0.7319     1 -0.724
## 4 0 0 NA -0.0183     1 -0.118
## 5 0 1 NA -0.1847     1 -0.235
## 6 0 0 NA -0.6631     1 -0.778
##
## Empirical TE: 0.526
##
## Mapped variables:
## Z -> Z
## Y -> Y.obs
## S -> S.obs
## BIP -> BIP
##
## Integration models:
## integration model for S.1 :
## integrate_parametric(formula = S.1 ~ BIP )
##
## Risk models:
## None present, see ?add_riskmodel for information on risk models.
## No estimates present, see ?ps_estimate.
## No bootstraps present, see ?ps_bootstrap.

```

We can add multiple integration models to a psdesign object, say we want a model for  $S(0)|W$ :

```

binary.ps + integrate_parametric(S.0 ~ BIP)

## Augmented data frame: 800 obs. by 6 variables.
## Z Y S.1 S.0 cdfweights BIP
## 1 0 0 NA 0.3527      1 0.335
## 2 0 0 NA 1.4669      1 1.454
## 3 0 1 NA -0.7319     1 -0.724
## 4 0 0 NA -0.0183     1 -0.118
## 5 0 1 NA -0.1847     1 -0.235
## 6 0 0 NA -0.6631     1 -0.778
##
## Empirical TE: 0.526
##
## Mapped variables:
## Z -> Z
## Y -> Y.obs
## S -> S.obs
## BIP -> BIP
##
## Integration models:
## integration model for S.1 :
## integrate_parametric(formula = S.1 ~ BIP )
## integration model for S.0 :
## integrate_parametric(formula = S.0 ~ BIP )
##
## Risk models:
## None present, see ?add_riskmodel for information on risk models.
## No estimates present, see ?ps_estimate.
## No bootstraps present, see ?ps_bootstrap.

```

In a future version of the package, we will allow for estimation of the joint risk estimands that depend on both  $S(0)$  and  $S(1)$ . We can also use splines, other transformations, and other variables in the formula:

```

library(splines)
binary.ps + integrate_parametric(S.1 ~ BIP^2)
binary.ps + integrate_parametric(S.1 ~ bs(BIP, df = 3))
binary.ps + integrate_parametric(S.1 ~ BIP + BSM + BSM^2)

```

To include additional baseline covariates in the model for  $S(1)$ , such as age or gender, these variables that are present in the data frame must be mapped in the `psdesign` function call so that they are visible in the subsequent functions:

```
binary.ps <- psdesign(data = fakedata, Z = Z, Y = Y.obs, S = S.obs, BIP = BIP,
                      BSM = BSM, age = age)
binary.ps + integrate_parametric(S.1 ~ BIP + age)
```

These are shown as examples, we will proceed with the simple linear model for integration. The other integration models are called `integrate_bivnorm`, `integrate_nonparametric`, and `integrate_semi parametric`. See their help files for details on the models and their specification.

The next step is to define the risk model.

### Risk models and likelihoods

The risk model is the specification of the distribution for the outcome  $Y$  given  $S(1)$  and  $Z$ . We accommodate a variety of flexible specifications for this model, for continuous, binary, time-to-event, and count outcomes. We have implemented exponential and weibull survival models, and a flexible specification for binary models, allowing for standard or custom link functions. See the help file for `add_riskmodel` for more details.

Let's add a simple binary risk model using the logit link. The argument `D` specifies the number of samples to use for the simulated annealing, also known as empirical integration, in the EML procedure. In general, `D` should be set to something reasonably large, like 2 or 3 times the sample size.

```
binary.ps <- binary.ps + risk_binary(model = Y ~ S.1 * Z, D = 50, risk = risk.logit)
binary.ps

## Augmented data frame: 800 obs. by 6 variables.
##   Z Y S.1   S.0 cdfweights     BIP
## 1 0 0  NA  0.3527      1  0.335
## 2 0 0  NA  1.4669      1  1.454
## 3 0 1  NA -0.7319      1 -0.724
## 4 0 0  NA -0.0183      1 -0.118
## 5 0 1  NA -0.1847      1 -0.235
## 6 0 0  NA -0.6631      1 -0.778
##
## Empirical TE: 0.526
##
## Mapped variables:
##   Z -> Z
##   Y -> Y.obs
##   S -> S.obs
##   BIP -> BIP
##
## Integration models:
##   integration model for S.1 :
##     integrate_parametric(formula = S.1 ~ BIP )
##
## Risk models:
##   risk_binary(model = Y ~ S.1 * Z, D = 50, risk = risk.logit )
##
## No estimates present, see ?ps_estimate.
## No bootstraps present, see ?ps_bootstrap.
```

### Estimation and bootstrap

We estimate the parameters and bootstrap using the same type of syntax. We can add a "ps\_estimate" object, which takes optional arguments `start` for starting values, and other arguments that are passed to the `optim` function in base R. The `method` argument determines the optimization method, we have found that "BFGS" works well in these types of problems and it is the default. Use "pseudo-score" as the `method` argument for pseudo-score estimation for binary risk models with categorical BIPs.

The `ps_bootstrap` function takes the additional arguments `n.boots` for the number of bootstrap replicates, and `progress.bar` which is a logical that displays a progress bar in the R console if true.

It is helpful to pass the estimates as starting values in the bootstrap resampling. With estimates and bootstrap replicates present, printing the psdesign object displays additional information.

```
binary.est <- binary.ps + ps_estimate(method = "BFGS")
binary.boot <- binary.est + ps_bootstrap(n.boots = 500, progress.bar = FALSE,
                                         start = binary.est$estimates$par, method = "BFGS")
binary.boot

## Augmented data frame: 800 obs. by 6 variables.
## Z Y S.1      S.0 cdfweights     BIP
## 1 0 0    NA  0.3527        1  0.335
## 2 0 0    NA  1.4669        1  1.454
## 3 0 1    NA -0.7319        1 -0.724
## 4 0 0    NA -0.0183        1 -0.0183
## 5 0 1    NA -0.1847        1 -0.235
## 6 0 0    NA -0.6631        1 -0.778
##
## Empirical TE: 0.526
##
## Mapped variables:
##   Z -> Z
##   Y -> Y.obs
##   S -> S.obs
##   BIP -> BIP
##
## Integration models:
##   integration model for S.1 :
##       integrate_parametric(formula = S.1 ~ BIP )
##
## Risk models:
##   risk_binary(model = Y ~ S.1 * Z, D = 50, risk = risk.logit )
##
## Estimated parameters:
## (Intercept)      S.1          Z      S.1:Z
## -0.920        -0.028       -0.220      -1.133
## Convergence: TRUE
##
## Bootstrap replicates:
##             Estimate boot.se lower.CL.2.5. upper.CL.97.5.
## (Intercept) -0.920    0.182     -1.286     -0.580
## S.1         -0.028    0.128     -0.276      0.220
## Z           -0.220    0.250     -0.697      0.277
## S.1:Z       -1.133    0.214     -1.581     -0.780
##             p.value
## (Intercept) 4.02e-07
## S.1         8.27e-01
## Z           3.80e-01
## S.1:Z       1.29e-07
##
## Out of 500 bootstraps, 500 converged ( 100 %)
##
## Test for wide effect modification on 1 degree of freedom. 2-sided p value < .0001
```

## Do it all at once

The next code chunk shows how the model can be defined and estimated all at once.

```
binary.est <- psdesign(data = fakedata, Z = Z, Y = Y.obs, S = S.obs, BIP = BIP) +
  integrate_parametric(S.1 ~ BIP) +
  risk_binary(model = Y ~ S.1 * Z, D = 50, risk = risk.logit) +
  ps_estimate(method = "BFGS")
```

## Plots and summaries

We provide summary and plotting methods for the psdesign object. If bootstrap replicates are present, the summary method does a test for wide effect modification. Under the parametric risk models implemented in this package, the test for wide effect modification is equivalent to a test of the null hypothesis that the  $S(1) : Z$  coefficient is equal to 0. This is implemented using a Wald test using the bootstrap estimate of the variance.

Another way to assess wide effect modification is to compute the standardized total gain (STG) ([Huang and Gilbert, 2011](#)) and ([Gabriel et al., 2015](#)). This is implemented in the calc\_STG function. The standardized total gain can be interpreted as the area sandwiched between the risk difference curve and the horizontal line at the marginal risk difference. It is a measure of the spread of the distribution of the risk difference, and is a less parametric way to test for wide effect modification. The calc\_STG function computes the STG at the estimated parameters and at the bootstrap samples, if present. The function prints the results and invisibly returns a list containing the observed STG, and the bootstrapped STGS.

```
calc_STG(binary.boot, progress.bar = FALSE)

## $obsSTG
## [1] 0.3397774
##
## $bootstraps
##   STG.boot.se STG.lower.CL.2.5 STG.upper.CL.97.5
## V1    0.1243311      0.1573031      0.6382418
```

The summary method also computes the marginal treatment efficacy marginalized over  $S(1)$  and compares it to the average treatment efficacy conditional on  $S(1)$ . This is an assessment of model fit. A warning will be given if the two estimates are dramatically different. These estimates are presented in the summary along with the empirical TE and the model-based marginal treatment efficacy that does not condition on  $S(1)$ .

```
smary <- summary(binary.boot)

## Augmented data frame: 800 obs. by 6 variables.
##   Z Y S.1     S.0 cdfweights     BIP
## 1 0 0 NA 0.3527 1 0.335
## 2 0 0 NA 1.4669 1 1.454
## 3 0 1 NA -0.7319 1 -0.724
## 4 0 0 NA -0.0183 1 -0.118
## 5 0 1 NA -0.1847 1 -0.235
## 6 0 0 NA -0.6631 1 -0.778
##
## Empirical TE: 0.526
##
## Mapped variables:
##   Z -> Z
##   Y -> Y.obs
##   S -> S.obs
##   BIP -> BIP
##
## Integration models:
##   integration model for S.1 :
##     integrate_parametric(formula = S.1 ~ BIP )
##
## Risk models:
##   risk_binary(model = Y ~ S.1 * Z, D = 50, risk = risk.logit )
##
## Estimated parameters:
## (Intercept)      S.1          Z      S.1:Z
##       -0.920     -0.028     -0.220     -1.133
## Convergence: TRUE
##
## Bootstrap replicates:
##   Estimate boot.se lower.CL.2.5. upper.CL.97.5.
## (Intercept)   -0.920     0.182      -1.286      -0.580
```

```

## S.1      -0.028   0.128     -0.276    0.220
## Z       -0.220   0.250     -0.697    0.277
## S.1:Z    -1.133   0.214     -1.581    -0.780
##          p.value
## (Intercept) 4.02e-07
## S.1        8.27e-01
## Z         3.80e-01
## S.1:Z     1.29e-07
##
## Out of 500 bootstraps, 500 converged ( 100 %)
##
## Test for wide effect modification on 1 degree of freedom. 2-sided p value < .0001
##
## Treatment Efficacy:
## empirical marginal model
## 0.526   0.526   0.539
## Model-based average TE is 2.3 % different from the empirical and 2.3 % different
##      from the marginal.

```

The `calc_risk` function computes the risk in each treatment arm, and contrasts of the risks. By default it computes the treatment efficacy, but there are other contrast functions available. The contrast function is a function that takes 2 inputs, the  $risk_0$  and  $risk_1$ , and returns some one dimensional function of those two inputs. It must be vectorized. Some built-in functions are "TE" for treatment efficacy =  $1 - risk_1(s)/risk_0(s)$ , "RR" for relative risk =  $risk_1(s)/risk_0(s)$ , "logRR" for log of the relative risk, and "RD" for the risk difference =  $risk_1(s) - risk_0(s)$ . You can pass the name of the function, or the function itself to `calc_risk`. See `?calc_risk` for more information about contrast functions.

Other arguments of the `calc_risk` function include `t`, the time at which to calculate the risk for time-to-event outcomes, `n.samps` which is the number of samples over the range of `S.1` at which the risk will be calculated, and `CI.type`, which can be set to "pointwise" for pointwise confidence intervals or "band" for a simultaneous confidence band. `sig.level` is the significance level for the bootstrap confidence intervals. If the outcome is time-to-event and `t` is not present, then it will use the restricted mean survival time.

```

head(calc_risk(binary.boot, contrast = "TE", n.samps = 20), 3)

##           S.1      Y      R0      R1 Y.boot.se
## V1 -2.2756987 -1.7437221 0.2980453 0.8177536 1.1622104
## V2 -1.4262708 -1.1360482 0.2930970 0.6260692 0.6957994
## V3 -0.5973759 -0.3532827 0.2883149 0.3901715 0.3328793
##          Y.upper.CL.0.95 Y.lower.CL.0.95 R0.boot.se R0.upper.CL.0.95
## V1      -0.30455106     -3.780389 0.08994238     0.4766278
## V2      -0.05541741     -2.556452 0.06901664     0.4331237
## V3      0.29675970     -1.275280 0.04941685     0.4007098
##          R0.lower.CL.0.95 R1.boot.se R1.upper.CL.0.95 R1.lower.CL.0.95
## V1      0.1188411 0.06911306     0.9368827     0.6720409
## V2      0.1468385 0.07592515     0.7762517     0.4768403
## V3      0.1734875 0.05079824     0.5248493     0.2834909

head(calc_risk(binary.boot, contrast = function(R0, R1) 1 - R1/R0, n.samps = 20), 3)

##           S.1      Y      R0      R1 Y.boot.se
## V1 -0.97417991 -0.71327775 0.2904830 0.4976780 0.4840781
## V2 -0.11875337  0.05966359 0.2855748 0.2685364 0.1882398
## V3 -0.09236484  0.08009338 0.2854242 0.2625636 0.1820450
##          Y.upper.CL.0.95 Y.lower.CL.0.95 R0.boot.se R0.upper.CL.0.95
## V1      0.1395560     -1.6775172 0.05815888     0.4084405
## V2      0.4746753     -0.4357974 0.03903555     0.3909614
## V3      0.4835707     -0.4036444 0.03849822     0.3904263
##          R0.lower.CL.0.95 R1.boot.se R1.upper.CL.0.95 R1.lower.CL.0.95
## V1      0.1763127 0.06531273     0.6258814     0.3695242
## V2      0.2043944 0.03260224     0.3454742     0.1923057
## V3      0.2053113 0.03176977     0.3379838     0.1879937

```

It is easy to plot the risk estimates. By default, the `plot` method displays the TE contrast, but this can be changed using the same syntax as in `calc_risk`.

```

plot(binary.boot, contrast = "TE", lwd = 2)
abline(h = smary$TE.estimates[2], lty = 3)

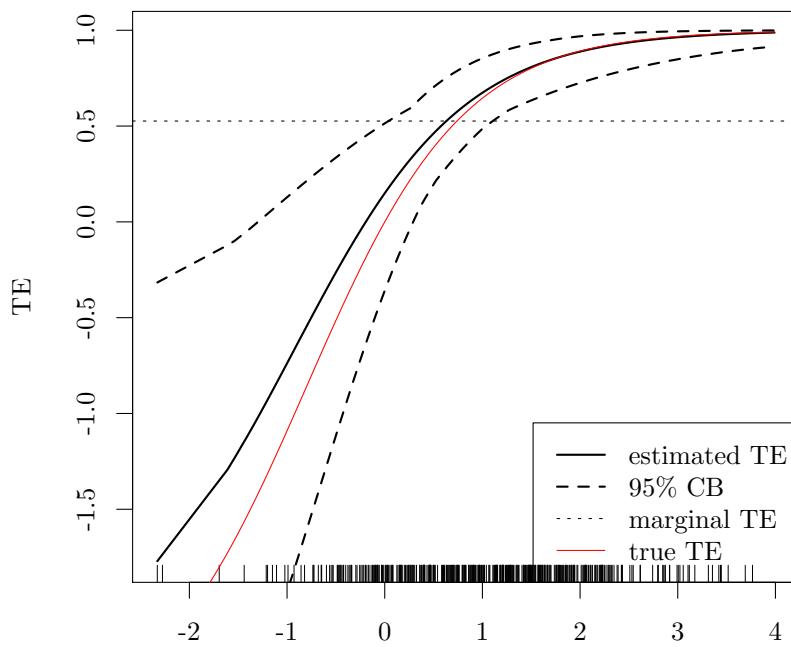
expit <- function(x) exp(x)/(1 + exp(x))
trueTE <- function(s){

  r0 <- expit(-1 - 0 * s)
  r1 <- expit(-1 - 1.25 * s)
  1 - r1/r0

}

rug(binary.boot$augdata$S.1)
curve(trueTE(x), add = TRUE, col = "red")
legend("bottomright", legend = c("estimated TE", "95%\% CB",
                                 "marginal TE", "true TE"),
       col = c("black", "black", "black", "red"),
       lty = c(1, 2, 3, 1), lwd = c(2, 2, 1, 1))

```



S.1

**Figure 1:** Plot showing the estimates using the example data, along with confidence bands (CB), and the true treatment efficacy (TE) curve.

By default, plots of psdesign objects with bootstrap samples will display simultaneous confidence bands for the curve. These bands  $L_\alpha$  satisfy

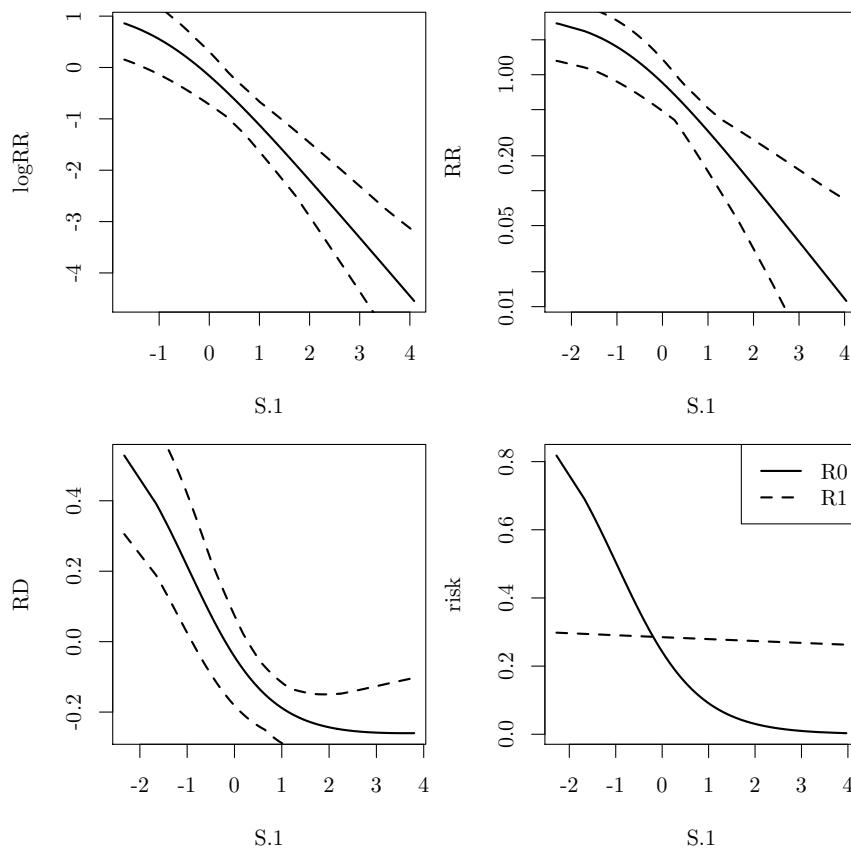
$$P \left\{ \sup_{s \in B} |\hat{TE}(s) - TE(s)| \leq L_\alpha \right\} \leq 1 - \alpha,$$

for confidence level  $\alpha$ . The alternative is to use pointwise confidence intervals, with the option `CI.type = "pointwise"`. These intervals satisfy

$$P \{ \hat{L}_\alpha \leq TE(s) \leq \hat{U}_\alpha \} \leq 1 - \alpha, \text{ for all } s.$$

Different summary measures are available for plotting. The options are "TE" for treatment efficacy  $= 1 - risk_1(s)/risk_0(s)$ , "RR" for relative risk  $= risk_1(s)/risk_0(s)$ , "logRR" for log of the relative risk, "risk" for the risk in each treatment arm, and "RD" for the risk difference  $= risk_1(s) - risk_0(s)$ . We can also transform using the log option of plot.

```
plot(binary.boot, contrast = "logRR", lwd = 2,
     col = c("black", "grey75", "grey75"))
plot(binary.boot, contrast = "RR", log = "y", lwd = 2,
     col = c("black", "grey75", "grey75"))
plot(binary.boot, contrast = "RD", lwd = 2,
     col = c("black", "grey75", "grey75"))
plot(binary.boot, contrast = "risk", lwd = 2, lty = c(1, 0, 0, 2, 0, 0))
legend("topright", legend = c("R0", "R1"), lty = c(1, 2), lwd = 2)
```



**Figure 2:** Plot illustrating ways that different risk contrast functions can be plotted.

The calc\_risk function is the workhorse that creates the plots. You can call this function directly to obtain estimates, standard errors, and confidence intervals for the estimated risk in each treatment arm and transformations of the risk like TE. The parameter n.samps determines the number of points at which to calculate the risk. The points are evenly spaced over the range of S.1. Use this function to compute other summaries, make plots using [ggplot2](#) (Wickham, 2009) or [lattice](#) (Sarkar, 2008) and more.

```
te.est <- calc_risk(binary.boot, CI.type = "pointwise", n.samps = 200)
head(te.est, 3)

##           S.1          Y        R0        R1 Y.boot.se
## V1 -2.328509 -1.770899 0.2983546 0.8267105 1.1943792
## V2 -2.275699 -1.743722 0.2980453 0.8177536 1.1622104
## V3 -1.694556 -1.360959 0.2946547 0.6956675 0.8334835
##      Y.lower.CL.2.5 Y.upper.CL.97.5 R0.boot.se R0.lower.CL.2.5
## V1      -4.768551     -0.6276106 0.09125321    0.1460172
## V2      -4.671015     -0.6184582 0.08994238    0.1475266
## V3      -3.456249     -0.4524787 0.07557692    0.1648504
```

```
##      R0.upper.CL.97.5 R1.boot.se R1.lower.CL.2.5 R1.upper.CL.97.5
## V1      0.4890998 0.06786962    0.6663453     0.9237321
## V2      0.4856823 0.06911306    0.6556765     0.9183559
## V3      0.4513863 0.07721039    0.5287710     0.8307616
```

## Summary and conclusion

We have implemented the core methods for principal surrogate evaluation in our **pseval** package. Our aim was to create a flexible and consistent user interface that allows for the estimation of a wide variety of statistical models in this framework. There has been some other work in this area. The **Surrogate** package implements the core methods for the evaluation of trial-level surrogates using a meta-analytic framework. It also has a wide variety of models, each implemented in a different function each with a long list of parameters (der Elst et al., 2016).

Our package uses the `+` sign to combine function calls into a single object. This is called “over-loading the `+` operator” and is most famously known from the **ggplot2** package. Conceptually, this was appealing to us because it allows users to build up analysis objects starting from the design, and ending with the estimation. The distinct analysis concepts of the design, risk model specification, integration model, and estimation/bootstrap approaches are separated into distinct function calls, each with a limited number of parameters. This makes it easier for users to keep track of their models, makes it easier to understand the methods involved, and allows for the specification of a wide variety of models by mixing and matching the function calls. This framework will also make it easier to maintain the codebase, and to extend it in the future as the methods evolve. Our package is useful for novice and expert R users alike, and implements an important set of statistical methods for the first time.

## Appendix

### Additional examples

#### Plot both types of CI

```
plot(binary.boot, contrast = "TE", lwd = 2, CI.type = "band")
sbs <- calc_risk(binary.boot, CI.type = "pointwise", n.samps = 200)
lines(Y.lower.CL.2.5 ~ S.1, data = sbs, lty = 3, lwd = 2)
lines(Y.upper.CL.97.5 ~ S.1, data = sbs, lty = 3, lwd = 2)
legend("bottomright", lwd = 2, lty = 1:3,
       legend = c("estimate", "simultaneous CI", "pointwise CI"))
```

#### Plot with ggplot2

```
library(ggplot2)
TE.est <- calc_risk(binary.boot, n.samps = 200)
ggplot(TE.est,
       aes(x = S.1, y = Y, ymin = Y.lower.CL.0.95, ymax = Y.upper.CL.0.95)) +
  geom_line() + geom_ribbon(alpha = .2) + ylab(attr(TE.est, "Y.function"))
```

#### Case-control design

```
cc.fit <- binary.cc + integrate_parametric(S.1 ~ BIP) +
  risk_binary(D = 10) + ps_estimate()
cc.fit
```

#### Survival outcome

```
surv.fit <- psdesign(fakedata, Z = Z, Y = Surv(time.obs, event.obs),
                      S = S.obs, BIP = BIP, CPV = CPV) +
  integrate_semi parametric(formula.location = S.1 ~ BIP, formula.scale = S.1 ~ 1) +
  risk_exponential(D = 10) + ps_estimate(method = "BFGS") + ps_bootstrap(n.boots = 20)
surv.fit
plot(surv.fit)
```

## Continuous outcome

```
fakedata$Y.cont <- log(fakedata$time.obs + 0.01)
cont.fit <- psdesign(fakedata, Z = Z, Y = Y.cont,
                      S = S.obs, BIP = BIP, CPV = CPV) +
  integrate_semiparametric(formula.location = S.1 ~ BIP, formula.scale = S.1 ~ 1) +
  risk_continuous(D = 10) + ps_estimate(method = "BFGS") + ps_bootstrap(n.boots = 20)
cont.fit
plot(cont.fit, contrast = "risk")
```

## Categorical S

`S.obs.cat` and `BIP.cat` are factors:

```
with(fakedata, table(S.obs.cat, BIP.cat))

cat.fit <- psdesign(fakedata, Z = Z, Y = Y.obs,
                      S = S.obs.cat, BIP = BIP.cat) +
  integrate_nonparametric(formula = S.1 ~ BIP) +
  risk_binary(Y ~ S.1 * Z, D = 10, risk = risk.probit) + ps_estimate(method = "BFGS")
cat.fit
plot(cat.fit)
```

## Pseudo-score

Categorical W allows for estimation of the model using the pseudo-score method for binary outcomes. `S` may be continuous or categorical:

```
cat.fit.ps <- psdesign(fakedata, Z = Z, Y = Y.obs,
                        S = S.obs, BIP = BIP.cat) +
  integrate_nonparametric(formula = S.1 ~ BIP) +
  risk_binary(Y ~ S.1 * Z, D = 10, risk = risk.logit) +
  ps_estimate(method = "pseudo-score") +
  ps_bootstrap(n.boots = 20, method = "pseudo-score")
summary(cat.fit.ps)
plot(cat.fit.ps)
```

## Bug reports

- Please file bugs and suggestions here as a github issue: <https://github.com/sachsmc/pseval/issues>.

## Bibliography

- W. V. der Elst, P. Meyvisch, A. Alonso, H. M. Ensor, and C. J. W. . G. Molenberghs. *Surrogate: Evaluation of Surrogate Endpoints in Clinical Trials*, 2016. URL <https://CRAN.R-project.org/package=Surrogate>. R package version 0.1-79. [p<sup>290</sup>]
- D. Follmann. Augmented designs to assess immune response in vaccine trials. *Biometrics*, 62(4):1161–1169, 2006. [p<sup>279</sup>]
- C. Frangakis and D. Rubin. Principal stratification in causal inference. *Biometrics*, 58(1):21–29, 2002. [p<sup>277, 278</sup>]
- E. E. Gabriel and D. Follmann. Augmented trial designs for evaluation of principal surrogates. *Biostatistics*, 17(3):453–467, 2016. [p<sup>278, 279</sup>]
- E. E. Gabriel and P. B. Gilbert. Evaluating principal surrogate endpoints with time-to-event data accounting for time-varying treatment efficacy. *Biostatistics*, 15(2):251–265, 2014. [p<sup>278, 279</sup>]
- E. E. Gabriel, M. C. Sachs, and P. B. Gilbert. Comparing and combining biomarkers as principle surrogates for time-to-event clinical endpoints. *Statistics in Medicine*, 34(3):381–395, 2015. ISSN 1097-0258. doi: 10.1002/sim.6349. [p<sup>286</sup>]

- P. Gilbert and M. Hudgens. Evaluating candidate principal surrogate endpoints. *Biometrics*, 64(4):1146–1154, 2008. [p<sup>278, 279, 280</sup>]
- Y. Huang and P. B. Gilbert. Comparing biomarkers as principal surrogate endpoints. *Biometrics*, 67(4):1442–1451, 2011. [p<sup>278, 280, 286</sup>]
- Y. Huang, P. B. Gilbert, and J. Wolfson. Design and estimation for evaluating principal surrogate markers in vaccine trials. *Biometrics*, 69(2):301–309, 2013. [p<sup>278, 280</sup>]
- T. Lumley. Analysis of complex survey samples. *Journal of Statistical Software*, 9(1):1–19, 2004. [p<sup>281</sup>]
- T. Lumley. *Survey: analysis of complex survey samples*, 2014. R package version 3.30. [p<sup>281</sup>]
- M. Pepe and T. Fleming. A nonparametric method for dealing with mismeasured covariate data. *Journal of the American Statistical Association*, 86(413):108–113, 1991. [p<sup>280</sup>]
- S. A. Plotkin and P. B. Gilbert. Nomenclature for immune correlates of protection after vaccination. *Clinical Infectious Diseases*, 54(11):1615–1617, 2012. [p<sup>277</sup>]
- M. C. Sachs and E. E. Gabriel. *pseval: Methods for Evaluating Principal Surrogates of Treatment Response*, 2016. R package version 1.3.0. [p<sup>277</sup>]
- 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. [p<sup>289</sup>]
- T. M. Therneau. *A Package for Survival Analysis in S*, 2015. R package version 2.38. [p<sup>280</sup>]
- T. M. Therneau and P. M. Grambsch. *ModModel Survival Data: Extending the Cox Model*. Springer, New York, 2000. [p<sup>280</sup>]
- H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. ISBN 978-0-387-98140-6. URL <http://had.co.nz/ggplot2/book>. [p<sup>289</sup>]
- J. Wolfson. *Statistical Methods for Identifying Surrogate Endpoints in Vaccine Trials*. Doctor of philosophy dissertation, University of Washington; Department of Biostatistics, Chair: Gilbert, Peter, 2009. [p<sup>280</sup>]
- J. Wolfson and P. Gilbert. Statistical identifiability and the surrogate endpoint problem, with application to vaccine trials. *Biometrics*, 66(4):1153–1161, 2010. [p<sup>278, 279</sup>]

*Michael C Sachs*

*Unit of Biostatistics, Institute of Environmental Medicine, Karolinska Institute  
Nobel väg 13  
17165 Stockholm, Sweden  
michael.sachs@ki.se*

*Erin E Gabriel*

*Biostatistics Research Branch, National Institute of Allergy and Infectious Disease  
5601 Fishers Lane  
Rockville, MD 20892  
USA  
erin.gabriel@nih.gov*

# Calculating Biological Module Enrichment or Depletion and Visualizing Data on Large-scale Molecular Maps with ACSNMineR and RNaviCell Packages

by Paul Deveau, Emmanuel Barillot, Valentina Boeva, Andrei Zinovyev and Eric Bonnet

**Abstract** Biological pathways or modules represent sets of interactions or functional relationships occurring at the molecular level in living cells. A large body of knowledge on pathways is organized in public databases such as the KEGG, Reactome, or in more specialized repositories, the Atlas of Cancer Signaling Network (ACSN) being an example. All these open biological databases facilitate analyses, improving our understanding of cellular systems. We hereby describe **ACSNMineR** for calculation of enrichment or depletion of lists of genes of interest in biological pathways. **ACSNMineR** integrates ACSN molecular pathways gene sets, but can use any gene set encoded as a GMT file, for instance sets of genes available in the Molecular Signatures Database (MSigDB). We also present **RNaviCell**, that can be used in conjunction with **ACSNMineR** to visualize different data types on web-based, interactive ACSN maps. We illustrate the functionalities of the two packages with biological data taken from large-scale cancer datasets.

## Introduction

Biological pathways and networks comprise sets of interactions or functional relationships, occurring at the molecular level in living cells (Adriaens et al., 2008; Barillot et al., 2012). A large body of knowledge on cellular biochemistry is organized in publicly available repositories such as the KEGG database (Kanehisa et al., 2011), Reactome (Croft et al., 2014) and MINT (Zanzoni et al., 2002). All these biological databases facilitate a large spectrum of analyses, improving our understanding of cellular systems. For instance, it is a very common practice to cross the output of high-throughput experiments, such as mRNA or protein expression levels, with curated biological pathways in order to visualize the changes, analyze their impact on a network and formulate new hypotheses about biological processes. Many biologists and computational biologists establish list of genes of interest (e.g. a list of genes that are differentially expressed between two conditions, such as normal vs disease) and then evaluate if known biological pathways have significant overlap with this list of genes.

We have recently released the Atlas of Cancer Signaling Network (ACSN), a web-based database which describes signaling and regulatory molecular processes that occur in a healthy mammalian cell but that are frequently deregulated during cancerogenesis (Kuperstein et al., 2015). The ACSN atlas aims to be a comprehensive description of cancer-related mechanisms retrieved from the most recent literature. The web interface for ACSN is using the NaviCell technology, a software framework dedicated to web-based visualization and navigation for biological pathway maps (Kuperstein et al., 2013). This environment is providing an easy navigation of maps through the use of the Google Maps JavaScript library, a community interface with a web blog system, and a comprehensive module for visualization and analysis of high-throughput data (Bonnet et al., 2015).

In this article, we describe two packages related to ACSN analysis and data visualization. The package **ACSNMineR** is designed for the calculation of gene enrichment and depletion in ACSN maps (or any user-defined gene set via the import function), while **RNaviCell** is dedicated to data visualization on ACSN maps. Both packages are available on the Comprehensive R Archive Network (<https://cran.r-project.org/web/packages/ACSNMineR/> and <https://cran.r-project.org/web/packages/RNaviCell/>), and on the GitHub repository (<https://github.com/sysbio-curie/ACSNMineR> and <https://github.com/sysbio-curie/RNaviCell>). For the remainder of this article, we describe the organization of each package and illustrate their capacities with several concrete examples demonstrating their capabilities.

## Packages organization

### ACSNMineR

Currently, ACSN maps cover signaling pathways involved in DNA repair, cell cycle, cell survival, cell death, epithelial-to-mesenchymal transition (EMT) and cell motility. Each of these large-scale

molecular maps is decomposed in a number of functional modules. The maps themselves are merged into a global ACSN map. Thus the information included in ACSN is organized in three hierarchical levels: a global map, five individual maps, and several functional modules. Each ACSN map covers hundreds of molecular players, biochemical reactions and causal relationships between the molecular players and cellular phenotypes. ACSN represents a large-scale biochemical reaction network of 4,826 reactions involving 2,371 proteins (as of today), and is continuously updated and expanded. We have included the three hierarchical levels in the **ACSNMineR** package, in order to be able to calculate enrichments at all three levels. The calculations are made by counting the number of occurrences of gene symbols (HUGO gene names) from a given list of genes of interest in all ACSN maps and modules. Table 1 is detailing the number of gene symbols contained in all the ACSN maps.

**Table 1:** ACSN maps included in the **ACSNMineR** package. Map: map name, Total: total number of gene symbols (HUGO) used to construct the map, Nb mod.: number of modules, Min: minimum number of gene symbols in the modules, Max: maximum number of gene symbols in the modules, Mean: average number of gene symbols per module. N.B.: one gene symbol may be present in several modules of the map.

Map	Total	Nb mod.	Min	Max	Mean
ACSN global	2239	67	2	629	79
Survival	1053	5	208	431	328
Apoptosis	667	7	19	382	136
EMT & Cell motility	634	9	18	629	137
DNA repair	345	21	3	171	45
Cell cycle	250	25	2	130	20

The statistical significance of the counts in the modules is assessed by using either the Fisher exact test (Fisher, 1922, 1934) or the hypergeometric test, which are equivalent for this purpose (Rivals et al., 2007).

The current ACSN maps are included in the **ACSNMineR** package, as a list of character matrices.

```
> length(ACSN_maps)
[1] 6
> names(ACSN_maps)
[1] "Apoptosis"      "CellCycle"       "DNA_repair"     "EMT_motility"   "Master"
[6] "Survival"
```

For each matrix, rows represent a module, with the name of the module in the first column, followed by a description of the module (optional), and then followed by all the gene symbols of the module. The maps will be updated according to every ACSN major release.

The main function of the **ACSNMineR** package is the enrichment function, which is calculating over-representation or depletion of genes in the ACSN maps and modules. We have included a small list of 12 Cell Cycle related genes in the package, named `genes_test` that can be used to test the main enrichment function and to get familiar with its different options.

```
> genes_test
[1] "ATM"        "ATR"        "CHEK2"      "CREBBP"     "TFDP1"      "E2F1"       "EP300"
[8] "HDAC1"      "KAT2B"      "GTF2H1"     "GTF2H2"     "GTF2H2B"
```

The example shown below is the simplest command that can be done to test a gene list for over-representation on the six included ACSN maps. With the list of 12 genes mentioned above and a default p-value cutoff of 0.05, we have a set of 8 maps or modules that are significantly enriched. The results are structured as a data frame with nine columns displaying the module name, the module size, the number of genes from the list in the module, the names of the genes that are present in the module, the size of the reference universe, the number of genes from the list that are present in the universe, the raw p-value, the p-value corrected for multiple testing and the type of test performed. The module field in the results data frame indicate the map name and the module name separated by a column character. If a complete map is significantly enriched or depleted, then only the map name is shown, without any module or column character. For instance, the third line of the results object below concern the E2F1 module of the CellCycle map.

```
> library(ACSNMineR)
> results <- enrichment(genes_test)
> dim(results)
```

```
[1] 8 9
> results[3,]
  module module_size nb_genes_in_module
V161 CellCycle:E2F1           19           12
                                         genes_in_module
V161 ATM ATR CHEK2 CREBPP TFDP1 E2F1 EP300 HDAC1 KAT2B GTF2H1 GTF2H2 GTF2H2B
  universe_size nb_genes_in_universe      p.value p.value.corrected    test
V161           2237           12 3.735018e-21     2.353061e-19 greater
```

The enrichment function can take up to nine arguments: the gene list (as a character vector), the list of maps that will be used to calculate enrichment or depletion, the type of statistical test (either the Fisher exact test or the hypergeometric test), the module minimal size for which the calculations will be done, the universe, the p-value threshold, the alternative hypothesis ("greater" for calculating over-representation, "less" for depletion and "both" for both tests) and a list of genes that should be removed from the universe (option "Remove\_from\_universe"). This option may be useful for instance if we know beforehand that a number of genes are not expressed in the samples considered.

Only the gene list is mandatory to call the enrichment function, all the other arguments have default values. The `maps` argument can either be a dataframe imported from a GMT file with the `format_from_gmt` function or a list of dataframes generated by the same procedure. The GMT format corresponds to the Broad Institute's Gene Matrix Transposed file format, a convenient and easy way to encode named sets of genes of interest in tab-delimited text files (it is not a graph or network format). By default, the function `enrichment` uses the ACSN maps previously described. The correction for multiple testing is set by default to use the method of Benjamini & Hochberg, but can be changed to any of the usual correction methods (Bonferroni, Holm, Hochberg, Holm, or Benjamini & Yekutieli ([Reiner et al., 2003](#))), or even disabled. The minimal module size represents the smallest size value of a module that will be used to compute enrichment or depletion. This is meant to remove results of low significance for module of small size. The universe in which the computation is made by default is defined by all the gene symbols contained in the maps. All the genes that were given as input and that are not present on the maps will be discarded. To keep all genes, the user can change the universe to HUGO, and in that case, the complete list of HUGO gene symbols will be used as the reference (> 39,000 genes). The threshold corresponds to the maximal value of the corrected p-value (unless the user chose not to correct for multiple testing) that will be displayed in the result table.

It may be of interest to compare enrichment of pathways in different cohorts or experiments. For example, enrichment of highly expressed pathways can reveal differences between two cancer types or two cell lines. To facilitate such comparisons, **ACSNMineR** provides a `multisample_enrichment` function. It relies on the `enrichment` function but takes a list of character vector `genes`. The name of each element of the list will be assumed to be the name of the sample for further analysis. Most of the arguments given to `multisample_enrichment` are the same as the ones passed to `enrichment`. However, the `cohort_threshold` is designed to filter out modules which would not pass the significance threshold in all samples.

Finally, to facilitate visualization of results, **ACSNMineR** integrates a representation function based on `ggplot2` syntax ([Wickham, 2009](#)). It allows representation of results from `enrichment` or `multisample_enrichment` with a limited number of parameters. Two types of display are available: heat-map tiles or bars. For multiple samples using a barplot representation, the number of rows used can be provided, otherwise all plots will be on the same row. For the heatmap, the color of the non-significant modules, and boundaries of the gradient for significant values can also be tuned.

We previously computed the p-value of the `genes_test` list with default parameters. The number of modules which have a p-value below 0.05 was 8, that can be compared to the 16 obtained without correction with the simple command shown below (some of the results are displayed in table 2).

```
enrichment(genes_test, correction_multitest = FALSE)
```

We can now plot the first six rows of the results obtained for corrected and uncorrected fisher test with heatmap format (Figure 1) or barplot (Figure 2) with the following commands:

```
# heatmap

represent_enrichment(enrichment = list(Corrected = results[1:6,],
Uncorrected = results_uncorrected[1:6,]),
                     plot = "heatmap", scale = "reverselog",
                     low = "steelblue" , high ="white", na.value = "grey")

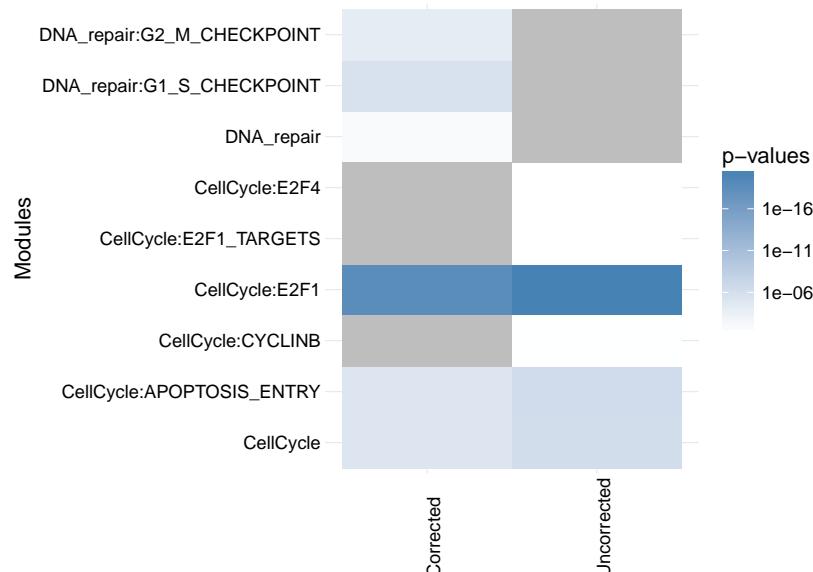
# barplot
```

**Table 2:** First rows of the results from enrichment analysis without correction. Module : name of the module. Mod. size: size of the module. Genes in module: genes from input which are found in the module. p-value: uncorrected p-value. Test : null hypothesis used, greater is synonym of enrichment.

Module	Mod. size	Genes in module	p-value	Test
CellCycle	242	ATM ATR CHEK2 CREBBP TFDP1 E2F1 EP300 HDAC1 KAT2B GTF2H1 GTF2H2 GTF2H2B	$5.4 \times 10^{-7}$	greater
CellCycle:APOPTOSIS_ENTRY	10	ATM ATR CHEK2 E2F1	$3.5 \times 10^{-7}$	greater
CellCycle:CYCLINB	7	ATM	0.04	greater

```
represent_enrichment(enrichment = list(Corrected = results[1:6, ],
                                      Uncorrected = results_uncorrected[1:6, ]),
                      plot = "bar", scale = "reverselog",
                      nrow = 1)
```

**Figure 1:** Representation of the enriched modules (first six rows for each setting), with either Bonferroni correction or no correction. Grey tiles means that the data is not available for this module in this sample. P-values of low significance are in white, whereas p-values of high significance are represented in blue.

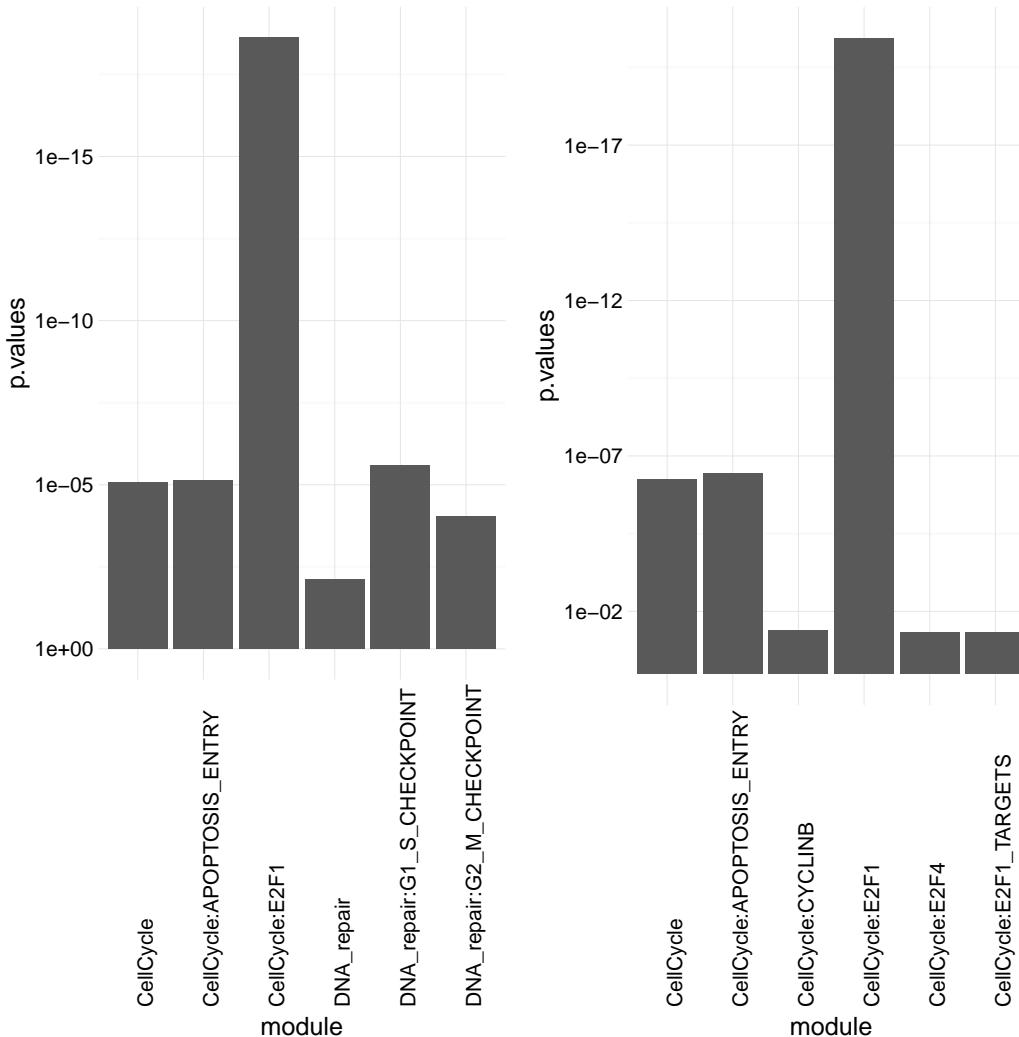


## RNaviCell

The NaviCell Web Service provides a server mode, which allows automating visualization tasks and retrieving data from molecular maps via RESTful (standard http/https) calls. Bindings to different programming languages are provided in order to facilitate the development of data visualization workflows and third-party applications (Bonnet et al., 2015). RNaviCell is the R binding to the NaviCell Web Service. It is implemented as a standard R package, using the R object-oriented framework known as Reference Classes (Wickham, 2015). Most of the work done by the user using graphical point-and-click operations on the NaviCell web interface are encoded as functions in the library encapsulating http calls to the server with appropriate parameters and data. Calls to the NaviCell server are performed using the library RCurl (Lang and the CRAN team, 2015), while data encoding/decoding in JSON format is performed with the RJSONIO library (Lang, 2014).

Once the RNaviCell library is installed and loaded, the first step is to create a NaviCell object and launch the browser session. This will automatically create a unique session ID with the NaviCell server. Once the session is established, various functions can be called to send data to the web session, set graphical options, visualize data on a map or get data from the map. There are 125

**Figure 2:** Representation of the enriched modules (first six rows for each setting), with either Bonferroni correction (left) or no correction (right). The modules are on the X axis and the p-values are on the Y axis.



functions available in the current version of RNaviCell. All of them are described with their different options in the RNaviCell documentation, and we provide a tutorial on the GitHub repository wiki (<https://github.com/sysbio-curie/RNaviCell/wiki/Tutorial>).

In the simple example detailed below, we create a NaviCell session, then load an expression data set from a local (tab-delimited) file. The data represent gene expression measured in a prostate cancer cell line resistant to hormonal treatment (agressive), and is taken from the Cell Line Encyclopedia project (Barretina et al., 2012). We visualize the data values on the Cell Cycle map (the default map), using heat maps. With this visualization mode, gene expression values are represented as a color gradient (green to red) in squares positioned next to the entities where the gene has been mapped (Figure 3). Note that the map is displayed in a browser and is *interactive*, i.e. users can zoom in to display more information and for example look in what reactions are involved the genes selected to be displayed, and lots of other informations (see Bonnet et al. (2015) and Kuperstein et al. (2015) for more details).

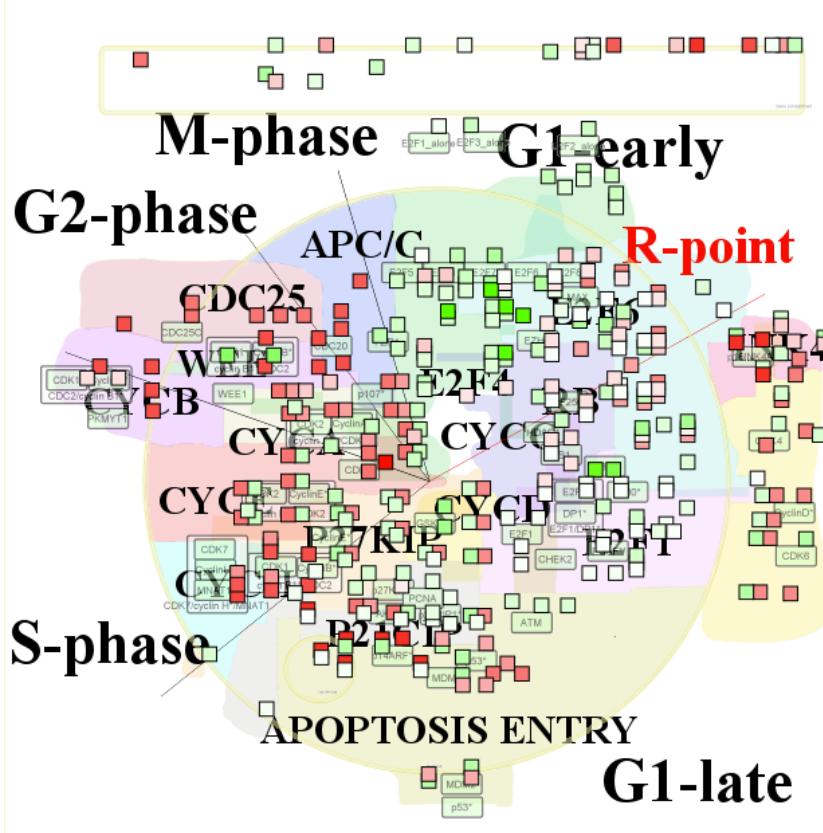
```
# a short RNaviCell script example

# load RNaviCell library

library(RNaviCell)

# create a NaviCell object and launch a server session
```

**Figure 3:** Gene expression values from a prostate cancer cell line visualized on the cell cycle map as heat map plots. The figure is a screenshot of the NaviCell map browser, with the map set at the top (the less detailed) zoom level. The essential phases of the cell cycle are indicated on the map (G1/S/G2/M). Note that on the web browser the map is interactive and the user can zoom in and out, change the graphical parameters, import additional data and perform functional analysis.



```
# this will automatically open a browser on the client
navicell <- NaviCell()
navicell$launchBrowser()

# import a gene expression matrix and
# send the data to the NaviCell server
# NB: the data_matrix object is a regular R matrix

data_matrix <- navicell$readDatatable('DU145_data.txt')
navicell$importDatatable("mRNA expression data", "DU145", data_matrix)

# set data set and sample for heat map representation

navicell$heatmapEditorSelectSample('0', 'data')
navicell$heatmapEditorSelectDatatable('0', 'DU145')
navicell$heatmapEditorApply()
```

## Case studies

### Analysis of breast cancer expression data

In a study published in 2008, Schmidt and colleagues analyzed gene expression patterns of 200 breast cancer patients not treated by systemic therapy after surgery using discovery approach to reveal additional prognostic motifs (Schmidt et al., 2008). Estrogen receptor (ER) expression and proliferative

activity of breast carcinomas are well-known and described prognostic markers. Patients with ER-positive carcinomas have a better prognosis than those with ER-negative carcinomas, and rapidly proliferating carcinomas have an adverse prognosis. Knowledge about the molecular mechanisms involved in the processes of estrogen-dependent tumor growth and proliferative activity has led to the successful development of therapeutic concepts, such as antiendocrine and cytotoxic chemotherapy.

The dataset corresponding to this study is available as a Bioconductor package. The code shown below is creating a list of differentially expressed genes between ER positive and ER negative samples, and calculates the enrichment in ACSN maps from this list of genes. As seen in Table 3, there is one map (DNA repair) and seven modules (belonging to the Cell Cycle, DNA repair and Apoptosis maps) enriched.

```
# load all necessary packages
library(breastCancerMAINZ)
library(Biobase)
library(limma)
library(ACSNMineR)
library(hgu133a.db)
library(RNaviCell)

# load data and extract expression and phenotype data
data(mainz)
eset <- exprs(mainz)
pdat <- pData(mainz)

# Create list of genes differentially expressed between ER positive and
# ER negative samples using moderated t-test statistics
design <- model.matrix(~factor(pdat$er == '1'))
lmFit(eset, design) -> fit
eBayes(fit) -> ebayes
toptable(ebayes, coef=2,n=25000) -> tt
which(tt$adj < 0.05) -> selection
rownames(tt[selection,]) -> probe_list
mget(probe_list, env = hgu133aSYMBOL) -> symbol_list
symbol_list <- as.character(symbol_list)

# calculate enrichment in ACSN maps

enrichment(symbol_list) -> results

dim(results)
[1] 8 9
```

**Table 3:** ACSN maps enrichment for genes differentially expressed between ER positive and ER negative samples in breast cancer. Module : name of the map/module. Mod. size: size of the module. Nb genes: number of genes from input which are found in the module. pval: raw p-value. Cor. pval: corrected p-value.

Module	Mod. size	Nb genes	pval	Cor. pval
Apoptosis:AKT_MTOR	79	47	0.00043	0.0068
CellCycle:E2F2_TARGETS	35	22	0.0055	0.043
CellCycle:E2F3_TARGETS	51	31	0.0023	0.025
CellCycle:E2F4_TARGETS	100	60	$5.8 \times 10^{-5}$	0.0037
DNA_repair	346	172	0.00038	0.0068
DNA_repair:CELL_CYCLE	82	49	0.00029	0.0068
DNA_repair:G1_CC_PHASE	25	18	0.0013	0.016
DNA_repair:S_CC_PHASE	46	28	0.0036	0.033

The Molecular Signatures Database (MSigDB) is one of the most widely used repository of well-annotated gene sets representing the universe of biological processes (Liberzon et al., 2011). We downloaded the canonical pathways set, counting more than 1,300 gene sets representing canonical pathways compiled by domain experts. The dataset is encoded with the GMT format, and can be imported within ACSNMineR with the `format_from_gmt` function. We calculate the enrichment for the

breast cancer differentially expressed gene list, simply specifying the MSigDB data we just imported as the maps option. Table 4 is displaying the pathways having a corrected p-value < 0.05. The prefix is indicating the database source, so we see that we have pathways from the KEGG, Reactome and PID databases. Consistent with our previous results, most of the enriched pathways are related to the cell cycle regulation.

```
# Import MSigDB canonical pathways and calculate enrichment on this database

mtsig <- format_from_gmt('c2.cp.v5.0.symbols.gmt')
enrichment(symbol_list, maps = mtsig)
```

**Table 4:** MSigDB canonical pathway database enrichment for genes differentially expressed between ER positive and ER negative samples in breast cancer. This table presents the 10 modules with lowest p-value out of 125 with corrected p-value lower than 0.05. Module : name of the module. Mod. size: size of the module. Nb genes: number of genes from input which are found in the module. Cor. pval: corrected p-value.

Pathway	Mod. size	Nb genes	Cor. pval
KEGG_CELL_CYCLE	128	76	$3.9 \times 10^{-8}$
REACTOME_CELL_CYCLE_MITOTIC	325	159	$3.9 \times 10^{-8}$
REACTOME_DNA_REPLICATION	192	98	$4.9 \times 10^{-6}$
PID_FOXM1PATHWAY	40	29	$3.1 \times 10^{-5}$
REACTOME_MITOTIC_M_M_G1_PHASES	172	87	$3.1 \times 10^{-5}$
REACTOME_CELL_CYCLE	421	182	$5 \times 10^{-5}$
REACTOME_MITOTIC_G1_G1_S_PHASES	137	71	$9 \times 10^{-5}$
PID_AURORA_B_PATHWAY	39	27	0.0002
REACTOME_S_PHASE	109	58	0.00024
PID_SYNDECAN_1_PATHWAY	46	30	0.00026

At last, we visualize the mean expression values for ER negative samples for all genes differentially expressed on the ACSN master (global) map using RNaviCell commands to create heatmaps.

```
# Select ER negative samples and calculate mean expression values

apply(eset[probe_list,pdat$er == 0],1,mean) -> er_minus_mean
names(er_minus_mean) <- symbol_list
er_minus_mean <- as.matrix(er_minus_mean)
colnames(er_minus_mean) <- c('exp')

# create a NaviCell session, import the expression matrix on the map and create
# heatmaps to represent the data points.

navicell <- NaviCell()
navicell$proxy_url <- "https://acsn.curie.fr/cgi-bin/nv_proxy.php"
navicell$map_url <- "https://acsn.curie.fr/navicell/maps/acsn/master/index.php"

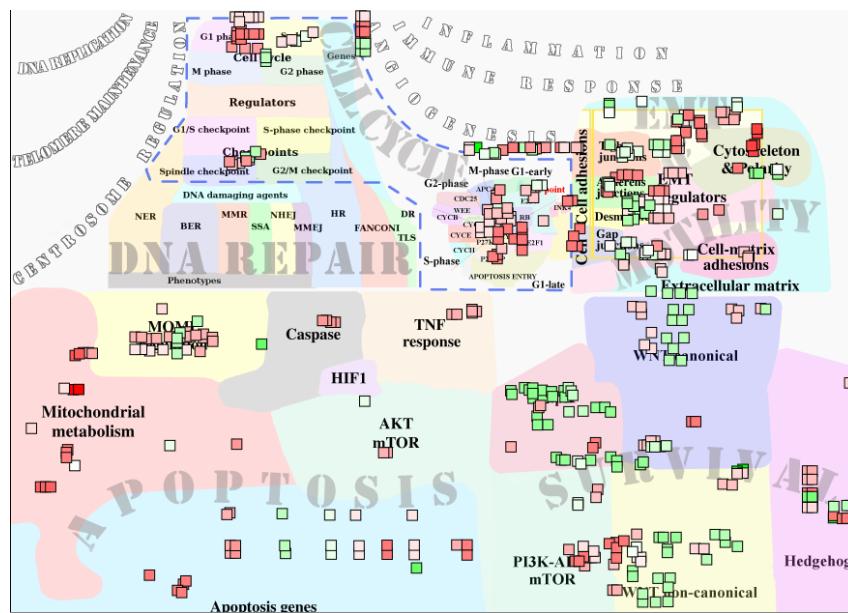
navicell$launchBrowser()
navicell$importDatatable("mRNA expression data", "GBM_exp", er_minus_mean)
navicell$heatmapEditorSelectSample('0','exp')
navicell$heatmapEditorSelectDatatable('0','GBM_exp')
navicell$heatmapEditorApply()
```

The Figure 4 is displaying the map for genes having a corrected p-value < 0.05.

### Analysis of glioblastoma mutation frequencies

Recent years have witnessed a dramatic increase in new technologies for interrogating the activity levels of various cellular components on a genome-wide scale, including genomic, epigenomic, transcriptomic, and proteomic information (Hawkins et al., 2010). Integrating these heterogeneous datasets provides more biological insights than performing separate analyses. For instance, international consortia such as The Cancer Genome Atlas (TCGA) have launched large-scale initiatives to characterize

**Figure 4:** Mean expression values for ER negative differentially expressed genes in breast cancer visualized as heatmaps on the ACSN master map.



multiple types of cancer at different levels on hundreds of samples. These integrative studies have already led to the identification of novel cancer genes (McLendon et al., 2008).

Malignant gliomas, the most common subtype of primary brain tumors, are aggressive, highly invasive, and neurologically destructive tumors considered to be among the deadliest of human cancers. In its most aggressive manifestation, glioblastoma (GBM), median survival ranges from 9 to 12 months, despite maximum treatment efforts (Maher et al., 2001). In this study we have analyzed whole-genome mutation data generated by the TCGA project on hundreds of patients. More specifically, we parsed the MAF (Mutation Annotation Format) GBM files produced by different sequencing centers to count and calculate gene mutation frequencies. We kept the mutations having a status likely to disturb the target protein's function (i.e. Frame\_Shift\_Del, Nonstop\_Mutation, In\_Frame\_Del, In\_Frame\_Ins, Missense\_Mutation, Nonsense\_Mutation, Splice\_Site, Translation\_Start\_Site). In total, we collected mutations for more than 13,000 genes in a total of 379 mutated samples. In order to retain the most frequently mutated genes, we calculated frequencies across all mutated samples, and kept genes having a frequency greater than 1% (3,293 genes). We further labelled genes having a frequency greater than 1% and less than 5% as "1" and genes highly mutated (frequency higher than 5%) as "2".

We loaded the data as a matrix in R and calculated the enrichment in ACSN maps with the ACSN-MineR function enrichment. The results are displayed in table 5. There are 6 modules significantly enriched in the DNA repair and EMT motility maps. Cell matrix adhesions and ECM (extra cellular matrix), part of the EMT motility map, are the modules with highest significance. The EMT motility map is significantly enriched at the global map level (second line in the table).

**Table 5:** ACSN maps enrichment for frequently mutated glioblastoma genes. Module : name of the module. Mod. size: size of the module. Nb genes: number of genes from input which are found in the module. Cor. pval: corrected p-value.

module	Mod. size	Nb genes	Cor. pval
DNA_repair:S_PHASE_CHECKPOINT	45	19	0.008
EMT_motility	635	181	0.0002
EMT_motility:CELL_MATRIX_ADHESIONS	73	45	3.73e-12
EMT_motility:CYTOSKELETON_POLARITY	154	47	0.022
EMT_motility:DESMOSOMES	29	15	0.002
EMT_motility:ECM	147	69	9.77e-11
EMT_motility:EMT_REGULATORS	629	178	0.0002

Visualization of the list of glioblastoma mutated genes is shown on figure 5. This figure was generated with the ACSNMineR commands detailed below. Results of the enrichment test correlate well with the visualization on the map, with a high density of low and high frequency mutated genes

in the EMT motility and DNA repair regions (maps) of the global ACSN map. Although they are not statistically significant, quite high densities can also be seen in other regions of the map.

```
library(RNaviCell)

# Create a NaviCell object, point it to the ACSN master map and launch
# a session.

navicell <- NaviCell()
navicell$proxy_url <- "https://acsn.curie.fr/cgi-bin/nv_proxy.php"
navicell$map_url <- "https://acsn.curie.fr/navicell/maps/acsn/master/index.php"
navicell$launchBrowser()

# Read the GBM data file and import it into the session.

mat <- navicell$readDatatable('gbm.txt')
navicell$importDatatable("Mutation data", "GBM", mat)

# set datatable and sample names for the glyph editor

navicell$drawingConfigSelectGlyph(1, TRUE)
navicell$glyphEditorSelectSample(1, "categ")
navicell$glyphEditorSelectShapeDatatable(1, "GBM")
navicell$glyphEditorSelectColorDatatable(1, "GBM")
navicell$glyphEditorSelectSizeDatatable(1, "GBM")
navicell$glyphEditorApply(1)

# set color, shape and size parameters for glyphs

navicell$unorderedConfigSetDiscreteShape("GBM", "sample", 0, 1)
navicell$unorderedConfigSetDiscreteShape("GBM", "sample", 1, 5)
navicell$unorderedConfigApply("GBM", "shape")

navicell$unorderedConfigSetDiscreteColor("GBM", "sample", 0, "398BC3")
navicell$unorderedConfigSetDiscreteColor("GBM", "sample", 1, "CC5746")
navicell$unorderedConfigApply("GBM", "color")

navicell$unorderedConfigSetDiscreteSize("GBM", "sample", 0, 4)
navicell$unorderedConfigSetDiscreteSize("GBM", "sample", 1, 14)

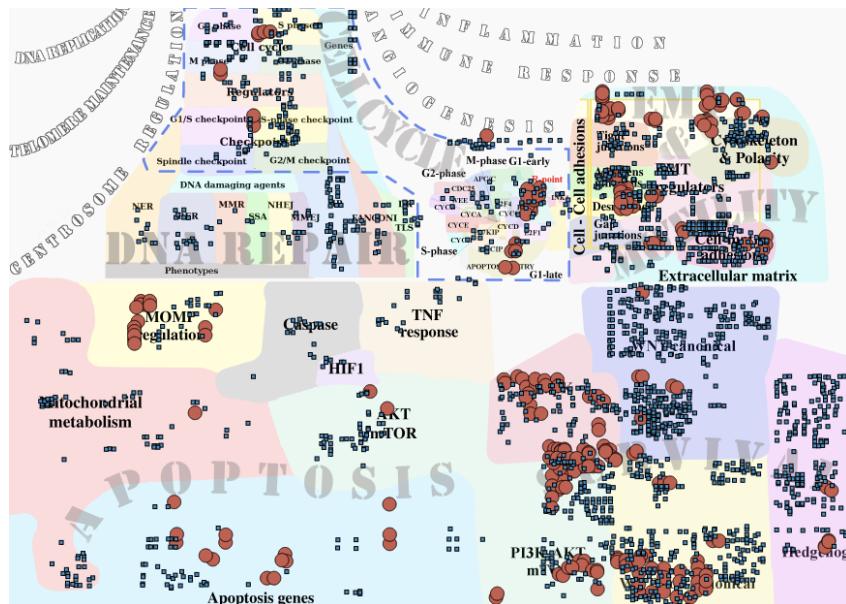
navicell$unorderedConfigApply("GBM", "size")
```

## Summary and perspectives

In this work, we presented the R package **ACSNMineR**, a novel package for the calculation of p-values for enrichment or depletion of genes in biological pathways. The package includes the six large-scale molecular maps and 67 functional modules of the Atlas of Cancer Signaling Network (ACSN). Enrichment can be calculated for those maps and modules with several options to play with, but can also be calculated for other databases of molecular pathways, that can be imported from GMT formated files.

We also describe in this work the **RNaviCell** package, a R package convenient to use with **ACSN-MineR**. This package is dedicated to create web-based and interactive data visualization on ACSN maps. Users can use this tools to represent genes of interest that have been shown to be related to the maps by calculating enrichment with **ACSNMineR**. Creating maps with the graphical user interface of the ACSN website can be a tedious task if the user has multiple samples or gene lists, and wants to compare their representations on ACSN maps. The **RNaviCell** package can be used to automate the process of creating the graphical representations automatically. The maps are displayed in a browser and are interactive, with the possibility for the user to zoom in and out, search for genes or molecular species, and see the details of the molecular reactions (what partners are involved, what is the state of a given species, etc.). For more details on how to use the interface and the different possibilities, see [Kuperstein et al. \(2013\)](#), [Bonnet et al. \(2015\)](#) and [Kuperstein et al. \(2015\)](#). We have shown how the packages **ACSNMineR** and **RNaviCell** can be combined to analyze expression data from breast cancer samples, and also to analyze the frequency of mutated genes in glioblastoma cancer samples.

**Figure 5:** Glioblastoma gene mutation frequency categories represented as glyphs on the ACSN global cancer map. High frequency mutated genes are pictured as large red circles, while low frequency mutated genes are depicted as small blue squares.



Of course, **ACSNMineR** is not the only R package for enrichment calculations. For instance, **GOstats** (Falcon and Gentleman, 2007) is probably one of the first packages that was created to calculate enrichment for Gene Ontology categories. **GOstats** can also be used to calculate enrichment for other biological pathways categories, such as KEGG pathways (by using an instance of the class `KEGGHyperGParams`) or PFAM protein families (using `PFAMHyperGParams`). However, its usage might not be as straightforward as **ACSNMineR**, and it does not seem possible to test user-defined biological pathways. Furthermore, other authors have pointed out that the KEGG database used by this package has not been updated since 2012. **clusterProfiler** is a recent R package released for enrichment analysis of Gene Ontology and KEGG with either hypergeometric test or Gene Set Enrichment Analysis (GSEA) (Yu et al., 2012). Via other packages, support for analysis of Disease Ontology and Reactome Pathways is possible. Interestingly, this package also offers the possibility to import user-defined gene set, through tab-delimited pairwise definition files. Other notable packages for enrichment calculations are **ReactomePA** for Reactome molecular pathways (Yu and He, 2016), **miRNAPath** for microRNA pathways (Cogswell et al., 2008) and **gage** (Luo et al., 2009). We believe that the main advantages of **ACSNMineR** compared to other packages are a direct access to the full set of ACSN maps (updated on a regular basis) and an easy way to test MSigDB gene sets or any user-defined gene set formatted appropriately.

In order to improve **ACSNMineR**, we may in the near future try to improve the speed of calculation, which might be a problem if a very large number of samples or experiments have to be analyzed rapidly. For instance, we could use the `foreach` and `%dopar%` operator to parallelize the most computationally demanding operations. It could also be useful to implement more sensitive methods of gene set enrichment measures, such as the Gene Set Enrichment Analysis (GSEA) method.

**RNaviCell** relies on standard HTTP calls to provide informations and calculations, and we have developed a number of bindings for popular programming languages such as R, Java and Python (Bonnet et al., 2015). This open architecture is designed to facilitate the development of utilities by other programmers and to facilitate the integration of ACSN maps in existing frameworks. The development of such services, sometimes called "microservices" (Fowler, 2014) is in expansion. Furthermore, this kind of open architecture could clear the way for a more unified and general access to reaction networks database, including for example WikiPathways (Kelder et al., 2012), Reactome (Croft et al., 2014) and other databases. The PSICQUIC project is a successfull example of such an architecture (Aranda et al., 2011). It is an effort of the HUPO proteomics standard initiative to standardize the access to molecular interaction databases programmatically, based on the specification of web services (using REST and SOAP calls) and a common query language (MIQL).

## Acknowledgements

This work was supported by a grant "Projet Incitatif et Collaboratif Computational Systems Biology Approach for Cancer" from Institut Curie. The authors would like to thank Pierre Gestraud for his comments on early versions of the **ACSNMineR** package and Eric Viara for guidance and assistance on the development of the **RNaviCell** package.

## Bibliography

- M. E. Adriaens, M. Jaillard, A. Waagmeester, S. L. Coort, A. R. Pico, and C. T. Evelo. The public road to high-quality curated biological pathways. *Drug Discovery Today*, 13(19):856–862, 2008. [p293]
- B. Aranda, H. Blankenburg, S. Kerrien, F. S. Brinkman, A. Ceol, E. Chautard, J. M. Dana, J. De Las Rivas, M. Dumousseau, E. Galeota, et al. Psicquic and psiscore: accessing and scoring molecular interactions. *Nature Methods*, 8(7):528–529, 2011. [p303]
- E. Barillot, L. Calzone, P. Hupe, J.-P. Vert, and A. Zinovyev. *Computational systems biology of cancer*. CRC Press, 2012. [p293]
- J. Barretina, G. Caponigro, N. Stransky, K. Venkatesan, A. A. Margolin, S. Kim, C. J. Wilson, J. Lehár, G. V. Kryukov, D. Sonkin, et al. The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603–607, 2012. [p297]
- E. Bonnet, E. Viara, I. Kuperstein, L. Calzone, D. P. Cohen, E. Barillot, and A. Zinovyev. Navicell web service for network-based data visualization. *Nucleic Acids Research*, page gkv450, 2015. [p293, 296, 297, 302, 303]
- J. P. Cogswell, J. M. Ward, I. A. Taylor, M. Waters, Y. Shi, B. Cannon, K. Kelnar, J. Kemppainen, D. Brown, C. Chen, R. K. Prinjha, J. C. Richardson, A. M. Saunders, A. D. Roses, and C. A. Richards. Identification of mirna changes in alzheimer's disease brain and csf yields putative biomarkers and insights into disease pathways. *Journal of Alzheimer's Disease*, 14:27–41, 2008. [p303]
- D. Croft, A. F. Mundo, R. Haw, M. Milacic, J. Weiser, G. Wu, M. Caudy, P. Garapati, M. Gillespie, M. R. Kamdar, et al. The reactome pathway knowledgebase. *Nucleic Acids Research*, 42(D1):D472–D477, 2014. [p293, 303]
- S. Falcon and R. Gentleman. Using gostats to test gene lists for go term association. *Bioinformatics*, 23(2):257–258, 2007. [p303]
- R. A. Fisher. On the interpretation of  $\chi^2$  from contingency tables, and the calculation of p. *Journal of the Royal Statistical Society*, pages 87–94, 1922. [p294]
- R. A. Fisher. *Statistical methods for research workers*. Oliver and Boyd, 1934. ISBN 0-05-002170-2. [p294]
- M. Fowler. *Microservices*, 2014. URL <http://martinfowler.com/articles/microservices.html>. Microservices. [p303]
- R. D. Hawkins, G. C. Hon, and B. Ren. Next-generation genomics: an integrative approach. *Nature Reviews Genetics*, 11(7):476–486, 2010. [p300]
- M. Kanehisa, S. Goto, Y. Sato, M. Furumichi, and M. Tanabe. Kegg for integration and interpretation of large-scale molecular data sets. *Nucleic Acids Research*, page gkr988, 2011. [p293]
- T. Kelder, M. P. van Iersel, K. Hanspers, M. Kutmon, B. R. Conklin, C. T. Evelo, and A. R. Pico. Wikipathways: building research communities on biological pathways. *Nucleic Acids Research*, 40(D1):D1301–D1307, 2012. [p303]
- I. Kuperstein, D. P. Cohen, S. Pook, E. Viara, L. Calzone, E. Barillot, and A. Zinovyev. Navicell: a web-based environment for navigation, curation and maintenance of large molecular interaction maps. *BMC systems biology*, 7(1):100, 2013. [p293, 302]
- I. Kuperstein, E. Bonnet, H. Nguyen, D. Cohen, E. Viara, L. Grieco, S. Fourquet, L. Calzone, C. Russo, M. Kondratova, et al. Atlas of cancer signalling network: a systems biology resource for integrative analysis of cancer data with google maps. *Oncogenesis*, 4(7):e160, 2015. [p293, 297, 302]
- D. T. Lang. *RJSONIO: Serialize R objects to JSON, JavaScript Object Notation*, 2014. URL <http://CRAN.R-project.org/package=RJSONIO>. R package version 1.3-0. [p296]

- D. T. Lang and the CRAN team. *RCurl: General Network (HTTP/FTP/...) Client Interface for R*, 2015. URL <http://CRAN.R-project.org/package=RCurl>. R package version 1.95-4.7. [p296]
- A. Liberzon, A. Subramanian, R. Pinchback, H. Thorvaldsdóttir, P. Tamayo, and J. P. Mesirov. Molecular signatures database (msigdb) 3.0. *Bioinformatics*, 27(12):1739–1740, 2011. [p299]
- Luo, Weijun, Friedman, Michael, Shedden, Kerby, Hankenson, Kurt, Woolf, and Peter. Gage: generally applicable gene set enrichment for pathway analysis. *BMC Bioinformatics*, 10:161, 2009. [p303]
- E. A. Maher, F. B. Furnari, R. M. Bachoo, D. H. Rowitch, D. N. Louis, W. K. Cavenee, and R. A. DePinho. Malignant glioma: genetics and biology of a grave matter. *Genes & development*, 15(11):1311–1333, 2001. [p301]
- R. McLendon, A. Friedman, D. Bigner, E. G. Van Meir, D. J. Brat, G. M. Mastrogianakis, J. J. Olson, T. Mikkelsen, N. Lehman, K. Aldape, et al. Comprehensive genomic characterization defines human glioblastoma genes and core pathways. *Nature*, 455(7216):1061–1068, 2008. [p301]
- A. Reiner, D. Yekutieli, and Y. Benjamini. Identifying differentially expressed genes using false discovery rate controlling procedures. *Bioinformatics*, 19(3):368–375, 2003. [p295]
- I. Rivals, L. Personnaz, L. Taing, and M.-C. Potier. Enrichment or depletion of a go category within a class of genes: which test? *Bioinformatics*, 23(4):401–407, 2007. [p294]
- M. Schmidt, D. Böhm, C. von Törne, E. Steiner, A. Puhl, H. Pilch, H.-A. Lehr, J. G. Hengstler, H. Kölbl, and M. Gehrmann. The humoral immune system has a key prognostic impact in node-negative breast cancer. *Cancer research*, 68(13):5405–5413, 2008. [p298]
- 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>. [p295]
- H. Wickham. *Reference classes*, 2015. URL <http://adv-r.had.co.nz/R5.html>. Advanced R. [p296]
- G. Yu and Q.-Y. He. Reactomepa: an r/bioconductor package for reactome pathway analysis and visualization. *Molecular BioSystems*, page accepted, 2016. doi: 10.1039/C5MB00663E. URL <http://pubs.rsc.org/en/Content/ArticleLanding/2015/MB/C5MB00663E>. [p303]
- G. Yu, L.-G. Wang, Y. Han, and Q.-Y. He. clusterprofiler: an r package for comparing biological themes among gene clusters. *Omics: a journal of integrative biology*, 16(5):284–287, 2012. [p303]
- A. Zanzoni, L. Montecchi-Palazzi, M. Quondam, G. Ausiello, M. Helmer-Citterich, and G. Cesareni. Mint: a molecular interaction database. *FEBS Letters*, 513(1):135–140, 2002. [p293]

*Paul Deveau*  
Computational Systems Biology of Cancer, Institut Curie  
Genetics and Biology of Cancers, Institut Curie  
26, rue d'Ulm 75248 Paris  
France  
[paul.deveau@curie.fr](mailto:paul.deveau@curie.fr)

*Emmanuel Barillot*  
Computational Systems Biology of Cancer, Institut Curie  
26, rue d'Ulm 75248 Paris  
France  
[emmanuel.barillot@curie.fr](mailto:emmanuel.barillot@curie.fr)

*Valentina Boeva*  
Institut Cochin, Inserm U1016, CNRS UMR 8104, Université Paris Descartes UMR-S1016  
22, Rue Mechanin 75014 Paris  
France  
[valentina.boeva@curie.fr](mailto:valentina.boeva@curie.fr)

*Andrei Zinovyev*  
Computational Systems Biology of Cancer, Institut Curie  
26, rue d'Ulm 75248 Paris  
France  
[andrei.zinovyev@curie.fr](mailto:andrei.zinovyev@curie.fr)

*Eric Bonnet*  
Centre National de Génotypage, Institut de Génomique, CEA  
2, rue Gaston Crémieux, 91057 Evry  
France  
[eric.bonnet@cng.fr](mailto:eric.bonnet@cng.fr)

# Subgroup Discovery with Evolutionary Fuzzy Systems in R: the SDEFSR Package

by Ángel M. García, Francisco Charte, Pedro González, Cristóbal J. Carmona and María J. del Jesus

**Abstract** Subgroup discovery is a data mining task halfway between descriptive and predictive data mining. Nowadays it is very relevant for researchers due to the fact that the knowledge extracted is simple and interesting. For this task, evolutionary fuzzy systems are well suited algorithms because they can find a good trade-off between multiple objectives in large search spaces. In fact, this paper presents the SDEFSR package, which contains all the evolutionary fuzzy systems for subgroup discovery presented throughout the literature. It is a package without dependencies on other software, providing functions with recommended default parameters. In addition, it brings a graphical user interface to avoid the user having to know all the parameters of the algorithms.

## Introduction

Subgroup discovery (SD) is a data mining field that aims to describe data using supervised learning techniques. The goal is to find simple, general and interesting patterns with respect to a given variable of interest. Throughout the literature, SD has been applied with success to different real-world problems in areas such as marketing (del Jesus et al., 2007; Berlanga et al., 2006), medicine (Carmona et al., 2015, 2013a; Stiglic and Kokol, 2012; Carmona et al., 2011; Gamberger et al., 2003) and e-learning (Poitras et al., 2016; Lemmerich et al., 2011; Carmona et al., 2010b), amongst others (Atzmueller et al., 2016; Jin et al., 2014; Carmona et al., 2013b; Rodriguez et al., 2013; Carmona et al., 2012).

SD is an useful rule learning process for complex search spaces. Therefore, the search strategy used becomes a key factor in the efficiency of the method. Different strategies can be found in the literature such as beam search in the algorithm CN2-SD (Lavrač et al., 2004b) and Apriori-SD (Kavšek and Lavrač, 2006), exhaustive algorithms such as SDMap (Atzmueller and Puppe, 2006) or Evolutionary Algorithms (EAs), for example.

EAs are stochastic algorithms for optimising and searching tasks, based on the natural evolution process (John, 1992). There are different paradigms within EAs: genetic algorithms (John, 1992; Goldberg, 1989), evolution strategies (Schwefel, 1995), evolutionary programming (Fogel, 2006) and genetic programming (Koza, 1992). With these methods the use of rules to represent the knowledge is known as evolutionary rule-based systems (Freitas, 2003) and has the advantage of allowing the inclusion of domain knowledge, also returning better rules. The use of EAs is very well suited for SD because these algorithms perform a global search in the space in a convenient way, stimulating the obtaining of simple, interesting and precise subgroups.

Fuzzy logic is an extension of traditional set theory whose main aim is to model imprecise knowledge (Zadeh, 1975). The main element is the fuzzy set, which allows belonging degrees in the range [0,1] where zero means not belonging at all and one means absolute belonging. A linguistic variable is a set of overlapped fuzzy sets which define linguistic labels that cover all the range of a numeric variable. The main advantage of using fuzzy logic on SD is obtaining a knowledge representation for numeric variables more understandable for experts. It improves the interpretability of rules and the knowledge representation is very close to human reasoning (Hüllermeier, 2011). In addition, it avoids the possible loss of information in variables with continuous domains due to a previous discretisation stage.

Nowadays, there are several frameworks that allow one to perform data mining tasks, but only a few of them have implementations of SD algorithms. The best known frameworks with SD algorithms are KEEL (Alcalá-Fdez et al., 2011) and VIKAMINE (Atzmueller and Lemmerich, 2012), but ORANGE (Demšar et al., 2013), and CORTANA (Meeng and Knobbe, 2011) also provide some implementations. In fact, VIKAMINE also provides an R package called `rsubgroup` (Atzmueller, 2014) which is an interface for R to VIKAMINE Java algorithms.

In this contribution the **SDEFSR** package is introduced. It provides the user with the most important evolutionary fuzzy rule-based methods for SD documented in the literature, being a major contribution to the R community. In addition, it also brings the capability of reading datasets in the KEEL data format (Alcalá-Fdez et al., 2011). This file format is not natively supported by R. Similarly, this package provides a Graphical User Interface (GUI) to make this task easier for the user, especially the unexperienced one.

This contribution is organized according to the following structure: The first section presents SD concepts, main properties and features. In the second section, the structure of the SDEFSR package

and its operations are described. In the third section, a complete example of use of the package is presented. Finally, the GUI of SDEFSR is shown in the last section.

## Subgroup discovery

SD was defined by ([Wrobel, 2001](#)) as:

*"In subgroup discovery, we assume we are given a so-called population of individuals (objects, customers, ...) and a property of those individuals we are interested in. The task of subgroup discovery is then to discover the subgroups of the population that are statistically "most interesting", i.e. are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest."*

SD tries to find relationships between different properties of a set with respect to one interesting or target variable. Such relations must be statistically interesting, so it is not necessary to find complete relations, partial relations could be interesting too.

Usually these relations are represented by means of rules, one per relation. A rule is defined as ([Lavrač et al., 2004a](#); [Gamberger and Lavrac, 2002](#)):

$$R : Cond \rightarrow Target_{value} \quad (1)$$

where  $Target_{value}$  is the value for the variable of interest (target variable) for the SD task and  $Cond$  is normally a conjunction of attribute-value pairs which describe the characteristics of the induced subgroup.

SD is a data mining task halfway between description and classification, because it has a target variable but its objective is not to predict but rather to describe. The use of a target variable is not possible in description, because description simply finds relationships between unlabeled objects.

A key point to fully understanding the goal of SD is how it differentiates from the classification objective. Classification is a predictive task that tries to split the entire search space, usually in a complex way, aiming to find the correct value for the variable in new incoming instances. On the other hand, SD aims to find interesting relationships among these instances regarding the variable of interest.

For instance, assuming there is a group of patients, the variable of interest is whether they have heart disease or not. The predictive data mining objective is to predict if new patients will have heart disease. However, SD tries to find which subgroup of patients are more likely to have heart disease according to certain characteristics. This could be relevant to develop a treatment against this disease.

## Main elements of subgroup discovery algorithms

Below, the most relevant aspects of SD algorithms are presented ([Atzmueller et al., 2004](#)):

- *Type of the target variable:* This is the kind of information the interest variable can hold. The target variable could be binary (two possible values), categorical ( $n$  possible values) or numerical (a real value within a range). Nevertheless, the majority of SD algorithms can only deal with binary or categorical target variables.
- *Description language:* Knowledge representation is a key factor in SD due to its descriptive nature. In this way, rules must be as simple as possible. Rules are usually represented by conjunctions of attribute-value pairs or in disjunctive normal form. Fuzzy logic could also be included in the rules in order to improve the interpretability of the knowledge ([Zadeh, 1975](#); [Hüllermeier, 2005](#)).
- *Quality measures:* This is the most important aspect in the design of SD algorithms. The quality measures must guide the learning process and must show the quality of the extracted knowledge. They are briefly described below.
- *Search strategy:* The search space grows exponentially with the number of variables. The use of a search strategy able to find a good solution, or the optimal one, by searching efficiently through the whole search space is very important.

## Quality measures for subgroup discovery

A quality measure tries to measure the interestingness of a given rule or subgroup, but there is not a formal definition of what interestingness is. However, the interestingness could be defined as a concept that emphasises conciseness, coverage, reliability, peculiarity, diversity, novelty, surprisingness, utility, and actionability ([Geng and Hamilton, 2006](#)). For SD, the most used criteria to measure the

interestingness of a rule are conciseness, generality or coverage, reliability, and novelty (Carmona et al., 2014).

Quality measures that accomplish this criteria available in the **SDEFSR** package are:

- Measures for conciseness (or complexity). It measures the complexity of the induced rules. Rules with a few number of attribute-value pairs are easy to remember and add to the expert's knowledge. The quality measures associated to this criterion are:
  - $N_r$ : The number of rules generated. A rule set with a large number of rules is much more difficult to remember than other that has less rules. Additionally, the lower the number of rules, the easier for the expert to filter those rules that are interesting.
  - $N_v$ : The number of variables of the rules generated. Rules with less variables are easier to understand and to remember; also they tend to be more general. Thus, rules with low number of variables are interesting.
- Measures for generality (or coverage). It measures the capacity of a rule to match with a great number of examples in the dataset. Also, the capacity to generalise the rule to other instances that are not in the training dataset is greater. The quality measures associated to this criterion are:
  - *Support*: It measures the frequency of correctly classified examples covered by the rule:

$$Sup(R) = \frac{n(Cond \wedge Target_{value})}{n_s} \quad (2)$$

where  $n(Cond \wedge Target_{value})$  means the number of examples that satisfy the antecedent and consequent part of the rule and  $n_s$  is the number of examples in the dataset.

- *Coverage*: It measures the percentage of examples covered by the rule related to the total number of examples:

$$Cov(R) = \frac{n(Cond)}{n_s} \quad (3)$$

where  $n(Cond)$  means the number of examples that satisfy the antecedent part of the rule.

- Measures for reliability. A rule is reliable when the relation described in the rule occurs in a high percentage of cases where the rule can be applied. The quality measures associated to this criterion are:
  - *Confidence*: It measure the percentage of examples correctly covered of the total of covered examples:

$$Conf(R) = \frac{n(Cond \wedge Target_{value})}{n(Cond)} \quad (4)$$

- Measures for novelty. A rule is novel if the knowledge obtained from this one is unknown by the user or it is unable to infer such knowledge from other rules. For this kind of criterion, the quality measures availables in the package are:

- *Significance*: It reflects the novelty in the distribution of the examples covered by the rule regarding the whole dataset:

$$Sign(R) = 2 \cdot \sum_{k=1}^{n_c} n(Cond \wedge Target_{value_k}) \cdot \log \left( \frac{n(Cond \wedge Target_{value_k})}{n(Cond \wedge Target_{value}) \cdot p(Cond)} \right) \quad (5)$$

where  $p(Cond) = \frac{n(Cond)}{n_s}$ ,  $n_c$  is the number of possible values of the target variable and  $Target_{value_k}$  is the  $k$ -th value of the target variable.

- Hybrid measures. These measures try to maximise more than one criterion with a single expression that finds a good trade-off between the criteria used. The hybrid quality measures implemented are:

- *Unusualness*: It is defined as the weighted relative accuracy of a rule and tries to maximise generality and reliability:

$$WRAcc(R) = \frac{n(Cond)}{n_s} \left( \frac{n(Cond \wedge Target_{value})}{n(Cond)} - \frac{n(Target_{value})}{n_s} \right) \quad (6)$$

- *True Positive Rate (TPR) or Sensitivity:* It measures the proportion of covered examples that has been correctly classified.

$$TPR(R) = \frac{n(Cond \wedge Target_{value})}{n(Target_{value})} \quad (7)$$

- *False Positive Rate (FPR):* It measures the proportion of examples that are covered that do not belong to the target variable.

$$FPR(R) = \frac{n(Cond \wedge \overline{Target_{value}})}{n(\overline{Target_{value}})} \quad (8)$$

where  $\overline{Target_{value}}$  means the negation of  $Target_{value}$ , i.e., the examples that not belong to the target class.

A more complete classification and definition of quality measures for SD is available in (Herrera et al., 2011) and (Atzmueller, 2015).

## Evolutionary fuzzy systems

Evolutionary fuzzy systems (EFSs) are the union of two powerful tools for approximate reasoning: EAs and fuzzy logic.

In one side, EAs (Eiben and Smith, 2003) are stochastic algorithms based on the natural evolution to solve complex optimisation problems. They are based on a population of representations of possible solutions, called chromosomes. A basic EA scheme is:

1. Generate the initial population
2. Evaluate the chromosomes of the population. This is the most important and expensive part of the EA. In the algorithms of this package, quality measures described above are used as evaluation function.
3. Select the chromosomes which the genetic operators will be applied.
4. Apply the genetic operators. The most used are:
  - Crossover operator. Which takes two chromosomes and generates two descendants as a combination of the elements of the parents.
  - Mutation operator. Which takes a chromosome and changes randomly a gene (a value of the possible solution).
5. Replace the population with the new generated chromosomes.
6. Go to step 2 until a stopping criterion is reached. Normally this criterion is a number of evaluations or generations.

These algorithms perform efficiently a global stochastic search through a huge search space. However, it is possible that these algorithms can not find an optimal solution (a global optimum), but a good one (a local optimum) that can solve the problem too. They are well suited for SD because the problem of finding subgroups can be formulated from the optimisation point of view as coding rules as a parameter structure that optimise some measures. Additionally, different kinds of rules exist in SD (with inequality, with intervals, fuzzy rules...). This can change dramatically the size of the search space and EAs can adapt these structures easily without performance degradation. Likewise, the selection of the genetic operators can make EAs great candidates to introduce expert knowledge in the search process (Carmona et al., 2014).

On the other side, fuzzy logic (Zadeh, 1975) is an extension of the traditional set theory. Its main objective is to model and deal with imprecise knowledge. The main difference with traditional set theory is that belonging degree is not zero or one, but a real value in [0,1]. This possibility allow one to define fuzzy limits and the chance of overlapping between fuzzy sets.

A fuzzy variable is a set of linguistic labels, e.g. low, medium and high, which are defined by overlapped fuzzy sets (Hüllermeier, 2011). This information is closer to human reasoning and it is possible to calculate with precision the value of each belonging degree. This expressivity allows one to obtain simpler rules because continuous variables are more understandable for humans. A rule can be represented by means of a set of fuzzy variables. To determine if the rule covers an example it is necessary to calculate the belonging degree of each variable in the rule with respect to the example. If all the variables have a belonging degree grater than zero, the example is covered.

EFSs are the union of both techniques, and work three ways(Herrera, 2008):

- EAs that evolve the fuzzy rules (changing the number of variables and their values) and use fuzzy set definitions defined by the user. This way of work is used by all the algorithms implemented in the **SDEFSR** package.
- EAs that evolve the fuzzy sets, changing the number of fuzzy sets for each variable, its shapes, etc.
- EAs that evolve both rules and fuzzy sets.

## The **SDEFSR** package

**SDEFSR** is a package entirely written on R from scratch. To the best of our knowledge, this package includes all the EFSs for SD presented throughout the literature. In addition, **SDEFSR** has the capacity to read data in different standard and well-known formats such as ARFF (Weka), KEEL, CSV and `data.frame` objects. Similarly, all functions included in the **SDEFSR** package have default parameters with values recommended by the authors. This allows the algorithms to be executed in an easy way without the necessity of knowing the parameters for final users.

### Algorithms included in the package

**SDEFSR** implements the following SD algorithms ([Ventura and Luna, 2016](#)):

- SDIGA ([del Jesus et al., 2007](#)). This is a mono-objective EA ([Back et al., 1997](#)) based on an Iterative Rule Learning (IRL) approach in which only the best rule is extracted from an execution of the EA. This EA is executed iteratively until a stopping criterion is reached.
- MESDIF ([Berlanga et al., 2006](#)). A multi-objective EA ([Deb, 2001](#)) based on the SPEA2 ([Zitzler et al., 2002](#)) algorithm. It returns the best  $n$  rules (where  $n$  is an user parameter) in the pareto front.
- NMEEF-SD ([Carmona et al., 2010a](#)). Another multi-objective EA, based on the NSGA-II ([Deb et al., 2000](#)) algorithm which returns rules in the pareto front with a confidence greater than a given threshold. It has a reinitialisation operator to promote diversity and maximise the covering of all examples for target variable.
- FuGePSD ([Carmona et al., 2015](#)). An algorithm that uses genetic programming in which a competitive-cooperative scheme is implemented in order to obtain the best global rules. The key operation of this algorithm is the Token Competition ([Leung et al., 1992](#)), which promotes the extraction of precise, general and also diverse rules from the evolutionary process.

All these methods share the following characteristics:

- They use fuzzy logic to improve the interpretability of results, making them robust when working with noisy data ([Luengo et al., \(In Press\)](#) and also allowing one to include expert knowledge on the evolutionary learning process.
- Rules can be represented by a conjunction of attribute-value pairs (canonical form) or in disjunctive normal form (DNF). In ([Carmona et al., 2009](#)) there is an analysis justifying not using the DNF rule representation on some SD algorithms.
- It is possible to specify the quality measures used to guide the evolutionary process.
- All of the algorithms can deal with categorical (or multi-class) target variables.

### Package architecture, extensibility, limitations and comparison with similar packages

The main advantage of the **SDEFSR** package is that it provides all EFSs for SD that exist in the literature. These algorithms are not included in R at the moment. Therefore, this package provides to the R community a brand new possibility for data mining and data analysis.

The base of the package is defined by two S3 classes. These classes are:

- "SDEFSR\_Dataset". This object defines a dataset and contains information about it. Such information are stored in the following fields:
  - `relation`. Defines the name of the relation that this dataset belongs to, e.g. "german credit".
  - `attributeNames`. Stores the names of the attributes.

- `attributeTypes`. A character that defines the data type of the attribute, i.e. 'r' for real values, 'e' for integers and 'c' for categoricals variables.
- `min`. A vector with the minimum value for numerical variables, for categorical variables, this value is zero.
- `max`. A vector with the maximum value for numerical variables or the number of different categorical values that has a categorical variable.
- `nVars`. The number of variables in the dataset (excluding the target class).
- `data`. A list that contains each example of the dataset. The categorical variables in data are codified. This means that a categorical value is represented as a value in [0, max – 1] that represents the position of the value. For example, in the german-credit dataset, the variable `ForeignWorker` has two values: "A201" and "A202", these values are represented in the data field of a "SDEFSR\_Dataset" class as 0 or 1, respectively.
- `class_names`. A vector with the categorical values of the target class. By default, the target variable is the last one. If it is not categorical, the method that reads the dataset fails. The user can select other target class when executing the algorithms.
- `examplesPerClass`. A list with the number of examples belonging to each class.
- `lostData`. A logical that indicates the presence of lost data.
- `covered`. A logical vector with length the number of examples that indicates which examples are covered by the generated rules. This value by default is NA, but it is used in some algorithms like SDIGA.
- `fuzzySets`. A list that indicates the fuzzy sets definitions for each variable. This value is NA by default.
- `crispSets`. A list that indicates the crisp sets obtained from the fuzzy sets. As this value is inferred from `fuzzySets`, this is NA by default.
- `sets`. A vector that defines the number of fuzzy sets that each variable has or the number of categorical values.
- `categoricalValues`. A list that contains a vector of names of each categorical variable or NA if the variable is numerical.
- `Ns`. The number of examples in the dataset.

This class also exports the well-known S3 methods `print()` and `summary()` that show the data structure without codification and a summary with basic information about the dataset respectively.

- "SDEFSR\_Rules". This class is a list that contains the rules generated by an algorithm. To know the number of rules generated, it is possible to use `length(SDEFSR_RulesObject)`. Each rule has the following fields:
  - `rule`. The string that represents the rule description.
  - `qualityMeasures`. A list that contains the quality measures of the rule. Such measures are the same as described in the quality measures section.

This object must be returned for all the SD algorithms of this package, and it is necessary to make an analysis of the generated rules. This object is necessary for the exported functions `plotRules()` that plots an FPR vs TPR graph that allows the visualisation of rules, and the well-known method `sort()` that return other "SDEFSR\_Rules" object with the rules sorted by a given quality measure in descendant order. Likewise, this object overloads the subset operator ('[*l*]) to allow filtering operations easily.

Additionally, the package has a general function that reads datasets in ARFF, KEEL or CSV format called `read.dataset()` and `SDEFSR_DatasetFromDataFrame()` to transform a `data.frame` into a "SDEFSR\_Dataset". In summary, exported functions and S3 objects are presented in Table 1.

A potential future extension of the package will be the inclusion of the confusion matrix for each rule. With this matrix it would be possible to infer the rest of the quality measures. Also, additional quality measures could be easily added. Statistical validation of the results are now implemented in package `rsubgroup`, which is already available on CRAN. Therefore, **SDEFSR** delegates this task to that package.

The `rsubgroup` package contains SD algorithms that can complement the algorithms available in the **SDEFSR** package. `rsubgroup` includes more established algorithms for SD like beam search (Lowerre, 1976) or SD-Map (Atzmueller and Puppe, 2006). This opens a wide range of SD algorithms that a user can execute in R. Nevertheless, both packages have great differences when calling SD

S3 Objects and methods	SD Algorithms	Other methods
"SDEFSR_Dataset"	SDIGA()	read.dataset()
print.SDEFSR_Dataset()	MESDIF()	SDEFSR_DatasetFromDataFrame()
summary.SDEFSR_Dataset()	NMEEF_SD()	plotRules()
"SDEFSR_Rules"	FUGEPSD()	
"[.SDEFSR_Rules"		
"sort.SDEFSR_Rules()"		

**Table 1:** Methods and rules exported by the **SDEFSR** package.

methods and the results. This means that a future package, which joins both into one standard framework would be interesting.

Below, the neccesary methodology to perform the inclusion of new algorithms in the SDEFSR package is shown:

1. The SD algorithm must use an "SDEFSR\_Dataset" object as input and return an "SDEFSR\_Rules" object with the results. Optionally, these results can be displayed on the console in a human-readable way.
2. The method can be executed with a parameter that contains the path of a parameter file. This file must contain the same parameters as the algorithm. This means that the algorithm must be executed either by a parameter file or by writting all neccesary parameters.
3. The majority of the parameters must have default values to ease the use of the algorithm.
4. The SD algorithm must not depend on other packages that are not in the "base" set of packages.
5. The source code of the algorithm must be added in a separate file with the name of such method.

As many others packages in R, this package does not have any automated test that control the quality of the code or results obtained. Instead, as the algorithms implemented exists in other platforms (KEEL), we check the quality of the methods comparing the results with the original implementation against a significant number of datasets with a 5-fold cross validation schema. This test showed that the results of the methods in **SDEFSR** are very close or equal to the results obtained from the reference implementations. Developers who want to add a new method to the package, must demonstrate the validity of the results obtained.

## An example of use

This section describes an example of use of the package, covering the installation and loading of the package to the execution of a SD algorithm and the analysis of the rules generated.

### Installing and loading the SDEFSR package

The **SDEFSR** package is now available at CRAN, so it can be installed like any other package by simply typing:

```
> install.packages("SDEFSR")
```

The development version is available on GitHub at <https://github.com/SIMIDAT/SDEFSR>. To install and use the development version you need to install the **devtools** (Wickham and Chang, 2015) package and then use the command:

```
> devtools::install_github("SIMIDAT/SDEFSR")
```

The package can be loaded using either the `library()` or `require()` functions. Once the package has been loaded, there are six sample datasets stored as "SDEFSR\_Dataset" objects available: 'carTra', 'carTst', 'germanTra', 'germanTst', 'habermanTra' and 'habermanTst' that correspond to the 'car', 'german' and 'haberman' (Alcalá-Fdez et al., 2011) training and test datasets respectively. These are contained in the 'carTra.rda', 'carTst.rda', 'germanTra.rda', 'germanTst.rda', 'habermanTra.rda' and 'habermanTst.rda' files respectively, which are lazily loaded with the package. Also, rules generated by the SDIGA algorithm with default parameters over the 'haberman' dataset are loaded as 'habermanRules' as a "SDEFSR\_Rules" object. These rules are stored in the 'habermanRules.rda' file.

## Loading a dataset

In order to use SD algorithms available in the **SDEFSR** package, a "SDEFSR\_Dataset" object is necessary. This object can be generated using the `read.dataset()` function. This function reads ".dat" files with the KEEL data mining tool format, ARFF files ("arff") from WEKA or even CSV files ("csv"). The source code for reading ARFF files has been taken from the **mldr** package(Charte and Charte, 2015). Assuming the files 'iris.dat', 'iris.arff' and 'iris.csv' corresponding to the classical iris dataset in KEEL, ARFF and CSV formats respectively in the working directory, the loading of these files will be as follows:

```
> irisFromKEEL <- read.dataset("iris.dat")
> irisFromARFF <- read.dataset("iris.arff")
> irisFromCSV <- read.dataset("iris.csv")
```

Note that the function detects the type of data by the extension. To read csv files, the function has optional parameters that defines the separator used between fields, the decimal separator, the quote indicator and the NA identifier as parameters. By default, these options and values are `sep = ", "`, `quote = "\"",` `dec = ". "` and `na.strings = "?"` respectively. It is important to remark that sparse ARFF data is not supported.

If the dataset is not available in these formats, it is possible to obtain a "SDEFSR\_Dataset" object from a `data.frame`. This `data.frame` could be loaded by `read.table()` or similar functions. Eventually, the resulting `data.frame` has to be given to the `SDEFSR_DatasetFromDataFrame()` function. As we can see, this function allows the creation of datasets on the fly, as in this example:

```
> df <- data.frame(matrix(data = runif(1000), ncol = 10))
#Add class attribute (It must be the last attribute and it must be categorical)
> df[,11] <- c("Class 0", "Class 1", "Class 2", "Class 3")
> SDEFSR_DatasetObject <- SDEFSR_DatasetFromDataFrame(df, relation = "random")
```

This will assign to `SDEFSR_DatasetObject` a new dataset created randomly with 100 examples and 11 attributes. Note that the target variable must be categorical, because the SD algorithms can only deal with categorical target variables.

The `SDEFSR_DatasetFromDataFrame()` function has three additional parameters: `names`, `types`, and `classNames`. These allow the manual assignment of attribute names, their types, and a vector with values of target variable, respectively. Leaving the default values (NA), the function automatically retrieves these values through the information found on the dataset. However, if the information in the dataset is not accurate, it could cause unexpected results for the SD algorithms.

## Obtaining information from a loaded dataset

Once the dataset is loaded, it is possible to view a simple summary of its content by using the usual `summary()` function:

```
> summary(irisFromKEEL)
Summary of the SDEFSR_Dataset object: 'irisFromKEEL'
  - relation: iris
  - nVars: 4
  - Ns: 150
  - attributeNames: SepalLength, SepalWidth, PetalLength, PetalWidth, Class
  - class_names: Iris-setosa, Iris-versicolor, Iris-virginica
  - examplesPerClass: 50, 50, 50
```

Any of these values can be obtained individually using the '\$' operator on the "SDEFSR\_Dataset" object:

```
> irisFromKEEL$nVars
[1] 4
> irisFromKEEL$attributeNames
[1] "SepalLength" "SepalWidth" "PetalLength" "PetalWidth" "Class"
```

Also, it is possible to print the dataset as a `data.frame` with the `print()` function.

## Executing subgroup discovery algorithms

It is possible to execute a SD algorithm in two ways: through a parameter file, specifying as argument the path to such file, or by entering all the parameter names and values at the command line. You can

find the structure of the parameter file, among other useful information, on the help pages of each function.

Assuming the "SDEFSR\_Dataset" object 'irisFromKEEL' that has been loaded, and that the 'params.txt' parameter file is stored in the working directory, the easiest way to run the MESDIF() algorithm, for example, will be:

```
> ruleSet <- MESDIF(paramFile = "param.txt")
#or
> ruleSet <- MESDIF(training = irisFromKEEL)
```

The first way will execute the algorithm with the parameters and datasets defined in the parameter file. The second one will execute the algorithm with the specified "SDEFSR\_Dataset" object, and default values for remainder parameters. By default, the target variable used is the last defined in the dataset and the algorithm searches rules for all the values of the target variable.

An example of an execution with all parameters and the result obtained could be:

```
> ruleSet <- MESDIF(paramFile = NULL, training = irisFromKEEL, test = NULL,
+                     output = c("optionsFile.txt", "rulesFile.txt", "testQM.txt"),
+                     seed = 0, nLabels = 3, nEval = 300, popLength = 100,
+                     eliteLength = 2, crossProb = 0.6, mutProb = 0.01,
+                     RulesRep = "can", Obj1 = "CSUP", Obj2 = "CCNF", Obj3 = "null",
+                     Obj4 = "null", targetVariable = "Class",
+                     targetClass = "Iris-virginica")
-----
Algorithm: MESDIF
Relation: iris
Training dataset: training
Test dataset: test
Rules Representation: CAN
Number of evaluations: 300
Number of fuzzy partitions: 3
Population Length: 100
Elite Population Length: 2
Crossover Probability: 0.6
Mutation Probability: 0.01
Obj1: CSUP (Weighth: )
Obj2: CCNF (Weighth: )
Obj3: null (Weighth: )
Obj4: null
Number of examples in training: 150
Number of examples in test: 150
Target Variable: Class
Target Class: Iris-virginica
-----
Searching rules for only one value of the target class...

GENERATED RULE 1
Variable SepalWidth = Label 1 ( 2 , 3.2 , 4.4 )
THEN Iris-virginica

GENERATED RULE 2
Variable PetalWidth = Label 2 ( 1.3 , 2.5 , 3.7 )
THEN Iris-virginica

Testing rules...

Rule 1 :
- N_vars: 2
- Coverage: 0.8
- Significance: 0.602743
- Unusualness: 0.02
- Accuracy: 0.357724
- CSupport: 0.286667
- FSupport: 0.245
```

```
- CConfidence: 0.358333
- FConfidence: 0.351955
- True Positive Rate: 0.86
- False Positive Rate: 0.77
```

Rule 2 :

```
- N_vars: 2
- Coverage: 0.193333
- Significance: 27.673033
- Unusualness: 0.128889
- Accuracy: 0.9375
- CSupport: 0.193333
- FSupport: 0.201667
- CConfidence: 1
- FConfidence: 0.889706
- True Positive Rate: 0.58
- False Positive Rate: 0
```

Global:

```
- N_rules: 2
- N_vars: 2
- Coverage: 0.496666
- Significance: 14.137888
- Unusualness: 0.074444
- Accuracy: 0.647612
- CSupport: 0.3
- FSupport: 0.223334
- FConfidence: 0.620831
- CConfidence: 0.679166
- True Positive Rate: 0.72
- False Positive Rate: 0.385
```

The output has three defined sections:

- The first one provides to the user information about the current execution, i.e., the values given to the parameters.
- After that, the rules obtained are shown one by one. These rules are numbered, starting at 1.
- Finally, the quality measures applied over the test (or training if `test = NULL`) dataset for each rule are shown. At the end of results, the "Global" section shows the average results for the quality measures analysed,

As this output could be extremely large, the function also saves it to three files, one for each of the above sections. The name of these files by default are '`optionsFile.txt`', '`rulesFile.txt`' and '`testQM.txt`' and being saved into the working directory, overwriting existing files. The format of these files is identical to the output generated by the algorithm, but divided into the sections described above.

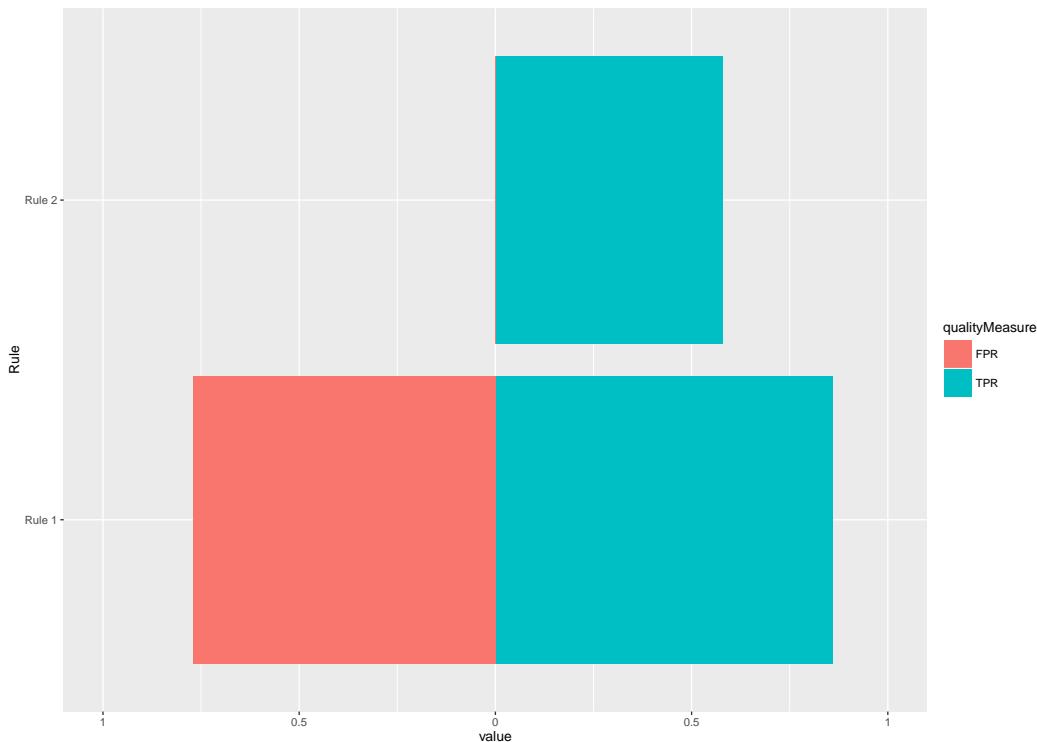
The `output` parameter must be included if the authors desire the modification of the names of paths in the stored files. It accepts a vector with the names or paths to be used instead of the default ones. Additionally to this output, the function returns the "`SDEFSR_Rules`" object which contains the rules generated and the quality measures associated to each rule.

### Analysing the rules obtained

After the execution of a SD algorithm, it returns a "`SDEFSR_Rules`" object that contains the rules obtained. Following the example, with the `ruleSet` object obtained we can plot a TPR vs FPR plot to view the reliability and generality of the rule (Kralj et al., 2005). Reliable rules have low values of FPR and high TPR values, and too general variables have high values for both TPR and FPR. To plot the rules, we can use the function `plotRules()`. (This function depends on the package `ggplot2`. If this is not installed, the user will be queried to install it.) The resulting plot is shown in Figure 1.

Additionally, we can directly order the rule set by a quality measure with the `sort()` function which returns another "`SDEFSR_Rules`" object with the rules sorted. By default, the ordering is done by confidence.

```
rulesOrderedBySignificance <- sort(x = ruleSet, decreasing = TRUE, by = "Significance")
```



**Figure 1:** Plot generated after executing `plotRules()` on the "SDEFSR\_Rules" object obtained in the example.

Filtering rules by number of attribute-value pairs or keeping those rules with a quality measure greater than a given threshold are interesting functionalities to extract only high-quality rules. Such filtering operations are quite simple to apply in **SDEFSR**. Using the subset operator ('[]') and introducing the filtering conditions will generate a new "SDEFSR\_Rules" object with the result:

Apply filter by unusualness:

```
> filteredRules <- ruleSet[Unusualness > 0.05]
```

We check only if the number of rules decrease. In this case, this value must be 1.

```
> length(filteredRules)
[1] 1
```

Also, you can make the filter as complex as you can Filter by Unusualness, TPr and number of variables:

```
> filteredRules <- ruleSet[Unusualness > 0.05 & TPr > 0.9 & nVars == 3]
```

In this case, there are not rules that match the conditions. Therefore, the number of rules must be 0.

```
> length(filteredRules)
[1] 0
```

## The user interface

The **SDEFSR** package provides the user with a GUI to ease the use of SD algorithms. It also allows basic exploratory analysis of the data to be performed. This GUI is accessible by calling the function:

```
> SDEFSR_GUI()
```

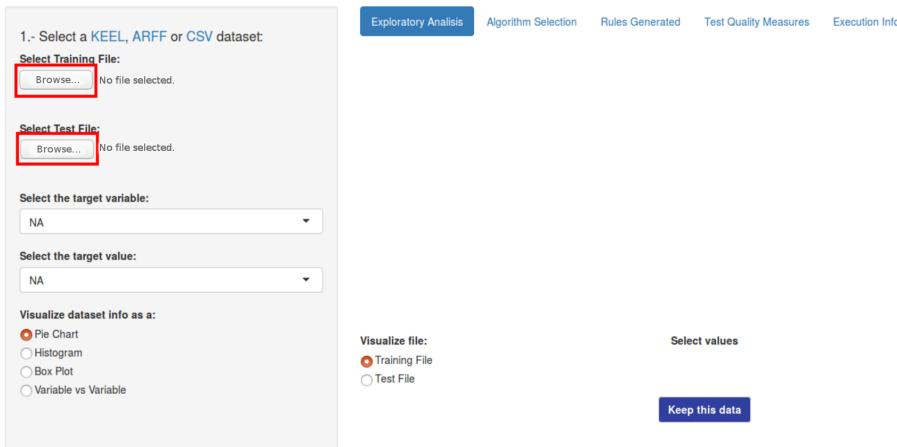
The GUI was generated using the **shiny** package, therefore this function depends on this package. It depends on package **ggplot2** too. If **shiny** or **ggplot2** are not installed in the system, the user is given the option to install them.

```
> SDEFSR_GUI()
Package 'shiny' is not installed and must be installed to run this GUI.
Do you want to install it? (Y/n): Y
```

:

Once the package has been installed, the GUI is launched. In Figure 2 the initial state of the GUI is shown. It is structured into two areas. On the left the user can select the training and test files to be used, the target variable, and the value to be used by the SD algorithm as target variable. Finally, the group of radio buttons allows the user to change between different graphics to perform a basic exploratory analysis. On the right there is a tab panel where tabs are organised by the logical process of execution and visualisation of results of a SD algorithm.

### Subgroup Discovery with Evolutionary Fuzzy Systems in R

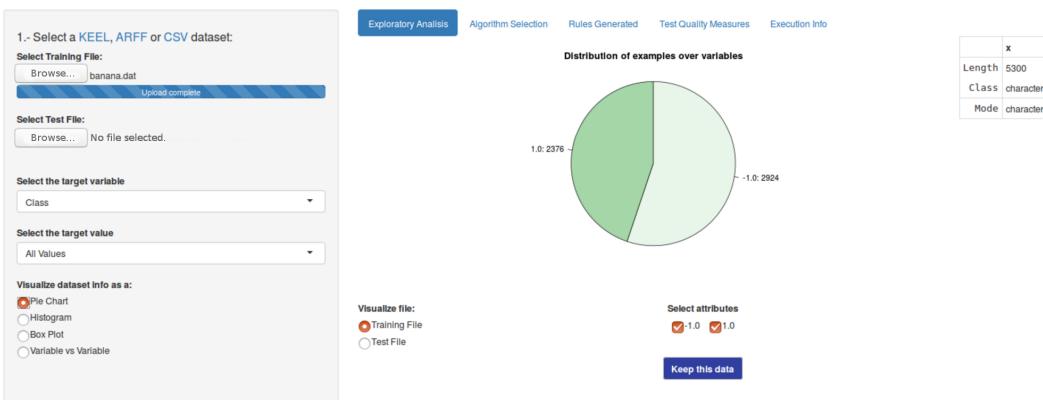


**Figure 2:** Initial screen of the SDEFSR user interface.

At this moment, the user only can perform the loading of datasets as a training or test file. The in Figure 2 within the red box initiate the reading of files with the the same file formats as the `read.dataset()` function. Once a dataset has been loaded, an initial plot appears. This will be a pie chart if the last variable is categorical or a histogram otherwise. Figure 3 shows an example of such a graph. This graph could show information about all the variables in the dataset. For example, the pie chart shows the value and the number of examples that belongs to this value. To the right of this plot, a table provides some information about the distribution of the data samples. This becomes interesting with numerical variables where basic statistical information is displayed. To change the variable being visualised, use the "Select the target variable" dropdown menu.

The 'Keep this data' button brings an interesting function. This button allows to filter examples whose values contain unselected values from the 'Select attributes' field for categorical variables or values from numerical variables that are not within the range specified on the 'Show range' slider.

### Subgroup Discovery with Evolutionary Fuzzy Systems in R

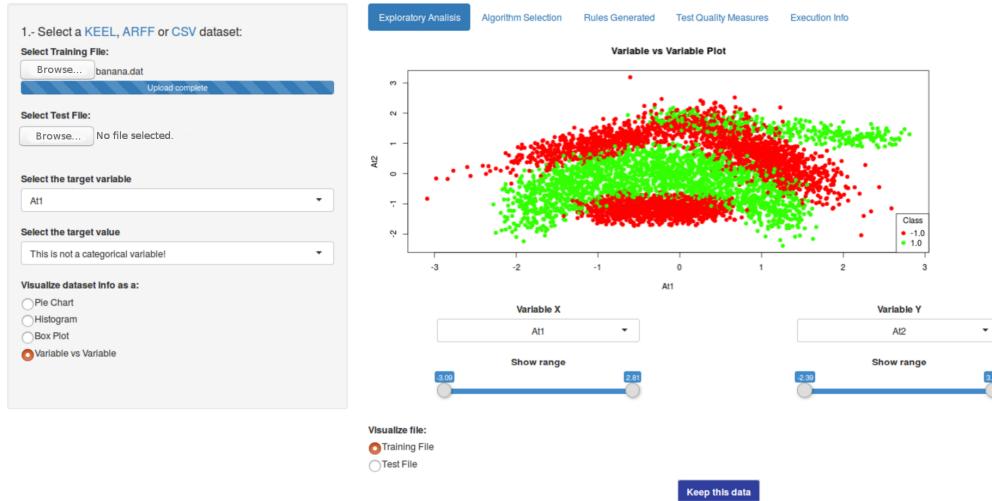


**Figure 3:** Screenshot of the GUI once a dataset has been loaded.

Another functionality is the 'Variable vs Variable' dataset visualisation. As shown in Figure 4 , it is possible to select two variables of the dataset and visualise their behaviour with respect to the

target class. In this case, the target class is the last variable of the dataset. The plot shown depends on the types of variables chosen. If both are numerical, a scatter plot like in Figure 4 is shown and if both are categorical, a bar plot is shown. A numerical variable versus a categorical variable is an undefined functionality, thus, no plot is shown.

### Subgroup Discovery with Evolutionary Fuzzy Systems in R

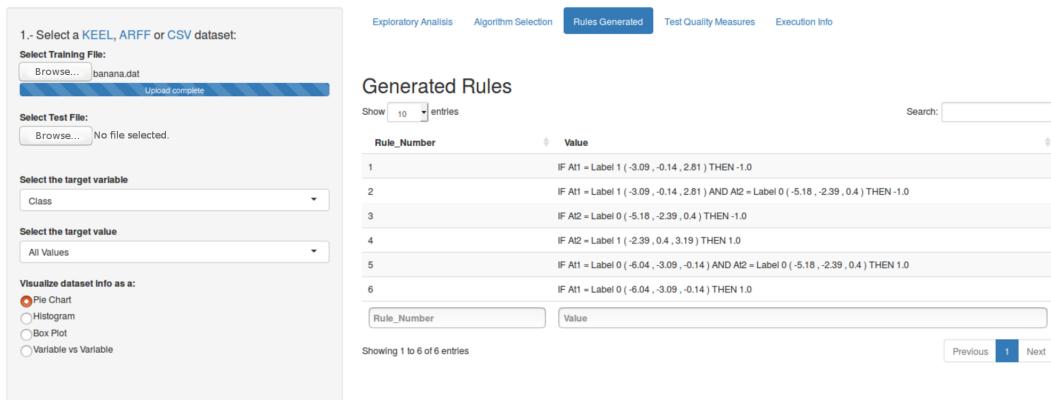


**Figure 4:** Screenshot of the 'Variable vs Variable' functionality.

On the "Algorithm Selection" tab, the user can choose a SD algorithm to execute, and easily modify all the available parameters.

After the execution of a SD algorithm the "Rules Generated" tab is automatically selected as shown in Figure 5. Here the user can see the rules over a DataTable. The most important function is the "Search" field where the user can find rules with a specific variable. For example, with the results obtained executing MESDIF with the 'banana' (Alcalá-Fdez et al., 2011) dataset, which is an artificial dataset whose classes form a banana shape, and leaving the default parameters of the GUI. Typing "At1" on the search box filter rules with the variable At1 on the antecedent. Similarly, typing "THEN 1.0" filter rules with the value 1.0 on the consequent.

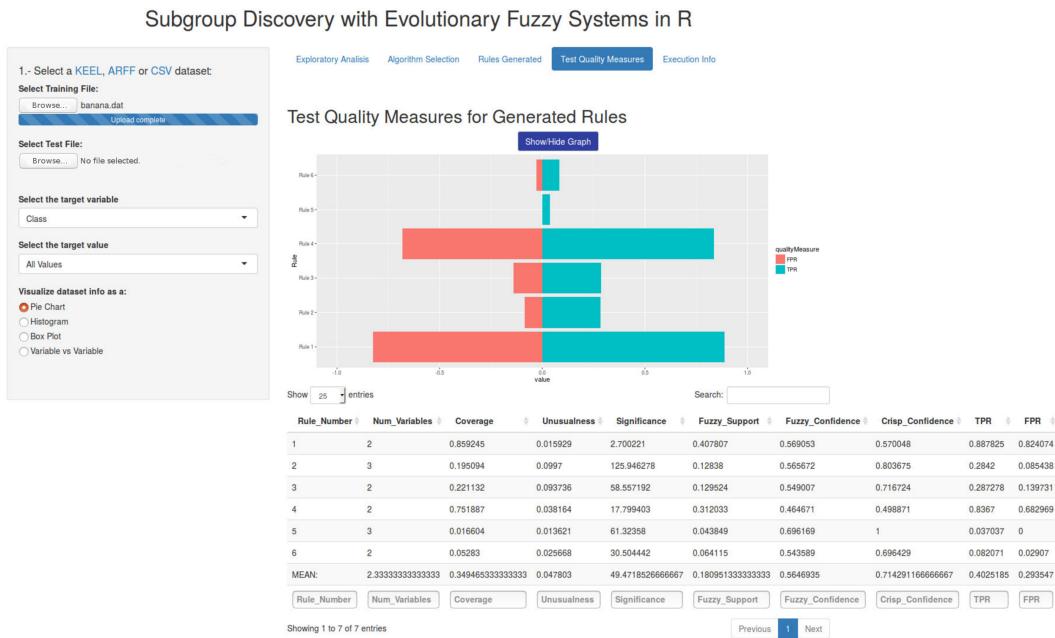
### Subgroup Discovery with Evolutionary Fuzzy Systems in R



**Figure 5:** Screen of the "Rules Generated" tab with generated rules.

Tab "Test Quality Measures" shows another DataTable with the quality measures for each rule. The filter options are similar to the DataTable of "Rules Generated". As shown in Figure 6, the button 'Show/Hide Graph' shows a FPR vs TPR plot of the generated rules, similar to the `plotRules()` function.

Finally, "Execution Info" shows information about the current execution, i.e., the parameters used in this execution. This information is like the execution information of a console execution (See Sec. 40.4.4).



**Figure 6:** Screen of the "Quality Measures" tab with the FPR vs TPR graph displayed.

## Summary

In this paper the **SDEFSR** package has been introduced. This package implements all the EFS-based algorithms for SD that exist in the specialised literature. The package can use datasets in ARFF, ".dat" of KEEL and CSV formats or a `data.frame` object. The main contribution of this package is the ease of use of the algorithms by means of functions with recommended parameters by default and different ways of execution, saving the user from the need to know the names of all the parameters of each algorithm. Also, a GUI is presented in order to make this task even easier.

The development of the package will continue in the future, including more functionality to work with datasets in more different formats, adding new SD algorithms, improving the performance of existing ones, and also bringing all this functionality to the GUI, which will be extended with more advanced tools for exploratory analysis.

## Acknowledgments

This paper has been partially supported by the project TIN2015-68854-R (FEDER Funds) of the Spanish Ministry of Economy and Competitiveness.

The code of this package, available at <https://github.com/SIMIDAT/SDEFSR> is MIT-licensed.

## Bibliography

- J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 2-3(17):255–287, 2011. [p307, 313, 319]
- M. Atzmueller. *rsubgroup: Subgroup Discovery and Analytics*, 2014. URL <http://CRAN.R-project.org/package=rsubgroup>. R package version 0.6. [p307]
- M. Atzmueller. Subgroup discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(1):35–49, 2015. [p310]
- M. Atzmueller and F. Lemmerich. Vikamine—open-source subgroup discovery, pattern mining, and analytics. In *Machine Learning and Knowledge Discovery in Databases*, pages 842–845. Springer, 2012. [p307]

- M. Atzmueller and F. Puppe. Sd-map—a fast algorithm for exhaustive subgroup discovery. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 6–17. Springer, 2006. [p307, 312]
- M. Atzmueller, F. Puppe, and H.-P. Buscher. Towards knowledge-intensive subgroup discovery. In *Proceedings of the Lernen - Wissensentdeckung - Adaptivität - Fachgruppe Maschinelles Lernen*, pages 111–117, 2004. [p308]
- M. Atzmueller, S. Doerfel, and F. Mitzlaff. Description-oriented community detection using exhaustive subgroup discovery. *Information Sciences*, 329:965–984, 2016. [p307]
- T. Back, D. B. Fogel, and Z. Michalewicz. *Handbook of evolutionary computation*. IOP Publishing Ltd., 1997. [p311]
- F. Berlanga, M. J. Del Jesus, P. González, F. Herrera, and M. Mesonero. Multiobjective evolutionary induction of subgroup discovery fuzzy rules: a case study in marketing. *Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining*, pages 337–349, 2006. [p307, 311]
- C. J. Carmona, P. González, M. J. del Jesus, and F. Herrera. An analysis of evolutionary algorithms with different types of fuzzy rules in subgroup discovery. In *Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on*, pages 1706–1711. IEEE, 2009. [p311]
- C. J. Carmona, P. González, M. J. del Jesus, and F. Herrera. Nmeesf-sd: non-dominated multiobjective evolutionary algorithm for extracting fuzzy rules in subgroup discovery. *IEEE Transactions on Fuzzy Systems*, 18(5):958–970, 2010a. [p311]
- C. J. Carmona, P. González, M. J. del Jesus, C. Romero, and S. Ventura. Evolutionary algorithms for subgroup discovery applied to e-learning data. In *Proceedings of Education Engineering Conference (EDUCON)*, pages 983–990. IEEE, 2010b. [p307]
- C. J. Carmona, P. González, M. Del Jesus, M. Navío-Acosta, and L. Jiménez-Trevino. Evolutionary fuzzy rule extraction for subgroup discovery in a psychiatric emergency department. *Soft Computing*, 15(12):2435–2448, 2011. [p307]
- C. J. Carmona, S. Ramirez-Gallego, F. Torres, E. Bernal, M. J. del Jesus, and S. Garcia. Web usage mining to improve the design of an e-commerce website: Orolivesur.com. *Expert Systems with Applications*, pages 11243–11249, 2012. [p307]
- C. J. Carmona, C. Chrysostomou, H. Seker, and M. J. del Jesus. Fuzzy rules for describing subgroups from influenza a virus using a multi-objective evolutionary algorithm. *Applied Soft Computing*, 13(8):3439–3448, 2013a. [p307]
- C. J. Carmona, P. González, B. García-Domingo, M. Del Jesus, and J. Aguilera. Mefes: an evolutionary proposal for the detection of exceptions in subgroup discovery. an application to concentrating photovoltaic technology. *Knowledge-Based Systems*, 54:73–85, 2013b. [p307]
- C. J. Carmona, P. González, M. J. del Jesus, and F. Herrera. Overview on evolutionary subgroup discovery: analysis of the suitability and potential of the search performed by evolutionary algorithms. *WIREs Data Mining and Knowledge Discovery*, 4(2):87–103, 2014. [p309, 310]
- C. J. Carmona, V. Ruiz-Rodado, M. J. del Jesus, A. Weber, M. Grootveld, P. González, and D. Elizondo. A fuzzy genetic programming-based algorithm for subgroup discovery and the application to one problem of pathogenesis of acute sore throat conditions in humans. *Information Sciences*, 298:180–197, 2015. [p307, 311]
- F. Charte and D. Charte. Working with multilabel datasets in R: The mlrd package. *The R Journal*, 7(2):149–162, dec 2015. URL <http://journal.r-project.org/archive/2015-2/charte-charte.pdf>. [p314]
- K. Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001. [p311]
- K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. *Lecture notes in computer science*, 1917:849–858, 2000. [p311]
- M. J. del Jesus, P. González, F. Herrera, and M. Mesonero. Evolutionary fuzzy rule induction process for subgroup discovery: a case study in marketing. *IEEE Transactions on Fuzzy Systems*, 15(4):578–592, 2007. [p307, 311]

- J. Demšar, T. Curk, A. Erjavec, Č. Gorup, T. Hočevar, M. Milutinovič, M. Možina, M. Polajnar, M. Toplak, A. Starić, et al. Orange: data mining toolbox in python. *The Journal of Machine Learning Research*, 14(1):2349–2353, 2013. [p307]
- A. Eiben and J. Smith. Introduction to evolutionary algorithms, 2003. [p310]
- D. B. Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*, volume 1. John Wiley & Sons, 2006. [p307]
- A. A. Freitas. A survey of evolutionary algorithms for data mining and knowledge discovery. In *Advances in evolutionary computing*, pages 819–845. Springer, 2003. [p307]
- D. Gamberger and N. Lavrac. Expert-guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, pages 501–527, 2002. [p308]
- D. Gamberger, N. Lavra??, and G. Krsta??i?? Active subgroup mining: A case study in coronary heart disease risk group detection. *Artificial Intelligence in Medicine*, 28(1):27–57, 2003. [p307]
- L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys (CSUR)*, 38(3):9, 2006. [p308]
- D. Goldberg. Genetic algorithm in search, optimization and machine learning. MA: Addison-Wesley Longman Publishing Co., Inc, 1989. [p307]
- F. Herrera. Genetic fuzzy systems: taxonomy, current research trends and prospects. *Evolutionary Intelligence*, 1(1):27–46, 2008. [p310]
- F. Herrera, C. J. Carmona, P. González, and M. J. del Jesus. An overview on subgroup discovery: foundations and applications. *Knowledge and information systems*, 29(3):495–525, 2011. [p310]
- E. Hüllermeier. Fuzzy methods in machine learning and data mining: Status and prospects. *Fuzzy sets and Systems*, 156(3):387–406, 2005. [p308]
- E. Hüllermeier. Fuzzy sets in machine learning and data mining. *Applied Soft Computing*, 11(2):1493–1505, 2011. [p307, 310]
- N. Jin, P. Flach, T. Wilcox, R. Sellman, J. Thumim, and A. Knobbe. Subgroup discovery in smart electricity meter data. *IEEE Transactions on Industrial Informatics*, 10(2):1327–1336, 2014. [p307]
- H. John. Adaptation in natural and artificial systems, 1992. [p307]
- B. Kavšek and N. Lavrač. Apriori-sd: Adapting association rule learning to subgroup discovery. *Applied Artificial Intelligence*, 20(7):543–583, 2006. [p307]
- J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992. [p307]
- P. Kralj, N. Lavrac, and B. Zupan. Subgroup visualization. In *8th International Multiconference Information Society (IS-05)*, pages 228–231, 2005. [p316]
- N. Lavrač, B. Cestnik, D. Gamberger, and P. Flach. Decision support through subgroup discovery: three case studies and the lessons learned. *Machine Learning*, 57(1-2):115–143, 2004a. [p308]
- N. Lavrač, B. Kavšek, P. Flach, and L. Todorovski. Subgroup discovery with cn2-sd. *The Journal of Machine Learning Research*, 5:153–188, 2004b. [p307]
- F. Lemmerich, M. Ifland, and F. Puppe. Identifying and presenting influence factors on student drop-outs by subgroup discovery. In *LWA 2011 - Technical Report of the Symposium "Lernen, Wissen, Adaptivität - Learning, Knowledge, and Adaptivity 2011" of the GI Special Interest Groups KDML, IR and WM*, pages 129–132, 2011. [p307]
- K. S. Leung, Y. Leung, L. So, and K. F. Yam. Rule learning in expert systems using genetic algorithm: 1, concepts. In *Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks (IIZUKA??92)*, volume 1, pages 201–204, 1992. [p311]
- B. T. Lowerre. *The Harpy speech recognition system*. PhD thesis, Carnegie-Mellon Univ., Pittsburgh, PA., 1976. [p312]
- J. Luengo, A. Garc??a-Vico, M. D. P??rez-Godoy, and C. Carmona. The influence of noise on the evolutionary fuzzy systems for subgroup discovery. *Soft Computing*, pages 1–25, (In Press). [p311]

- M. Meeng and A. Knobbe. Flexible enrichment with cortana–software demo. In *Proceedings of BeneLearn*, pages 117–119, 2011. [p307]
- E. Poitras, S. Lajoie, T. Doleck, and A. Jarrell. Subgroup discovery with user interaction data: An empirically guided approach to improving intelligent tutoring systems. *Educational Technology and Society*, 19(2):204–214, 2016. [p307]
- D. Rodriguez, R. Ruiz, J. C. Riquelme, and R. Harrison. A study of subgroup discovery approaches for defect prediction. *Information and Software Technology*, 55(10):1810 – 1822, 2013. [p307]
- H.-P. Schwefel. Sixth-generation computer technology series, 1995. [p307]
- G. Stiglic and P. Kokol. Discovering subgroups using descriptive models of adverse outcomes in medical care. *Methods of Information in Medicine*, 51(4):348–352, 2012. [p307]
- S. Ventura and J. M. Luna. Pattern mining with evolutionary algorithms, 2016. [p311]
- H. Wickham and W. Chang. *devtools: Tools to Make Developing R Packages Easier*, 2015. URL <http://CRAN.R-project.org/package=devtools>. R package version 1.8.0. [p313]
- S. Wrobel. Inductive logic programming for knowledge discovery in databases. In *Relational data mining*, pages 74–101. Springer, 2001. [p308]
- L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning???.*Information sciences*, 8(3):199–249, 1975. [p307, 308, 310]
- E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *International Congress on Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, 2002. [p311]

Ángel M. García

Department of Computer Science. University of Jaén

Jaén

Spain

[agvico@ujaen.es](mailto:agvico@ujaen.es)

Francisco Charte

Department of Computer Science and Artificial Intelligence. University of Granada

Granada

Spain

[fcharte@ugr.es](mailto:fcharte@ugr.es)

Pedro González

Department of Computer Science. University of Jaén

Jaén

Spain

[pglez@ujaen.es](mailto:pglez@ujaen.es)

Cristóbal J. Carmona

Department of Civil Engineering, Languages and Systems Area, University of Burgos.

Burgos

Spain

[cjcarmona@ubu.es](mailto:cjcarmona@ubu.es)

María J. del Jesus

Department of Computer Science. University of Jaén

Jaén

Spain

[mjjesus@ujaen.es](mailto:mjjesus@ujaen.es)

# dCovTS: Distance Covariance/Correlation for Time Series

by Maria Pitsillou and Konstantinos Fokianos

**Abstract** The distance covariance function is a new measure of dependence between random vectors. We drop the assumption of iid data to introduce distance covariance for time series. The R package **dCovTS** provides functions that compute and plot distance covariance and correlation functions for both univariate and multivariate time series. Additionally it includes functions for testing serial independence based on distance covariance. This paper describes the theoretical background of distance covariance methodology in time series and discusses in detail the implementation of these methods with the R package **dCovTS**.

## Introduction

There has been a considerable recent interest in measuring dependence by employing the concept of the *distance covariance function*. Székely et al. (2007) initially introduced the distance covariance as a new measure of dependence defined as the weighted  $L_2$ -norm between the joint characteristic function of two random vectors of arbitrary, but not necessarily of equal dimensions, and their marginal characteristic functions. However, the idea of using distance covariance for detecting independence can be also found in some early work by Feuerverger (1993). He considered measures of this form, with the main differences being the restriction to the univariate case and the choice of the weight function. Since Székely et al.'s (2007) work, there has been a wide range of studies extending the distance covariance definition and methodology in various topics; see Gretton et al. (2009) and Josse and Holmes (2014) and the references therein for a nice review.

Székely et al.'s (2007) distance covariance methodology is based on the assumption that the underlying data are iid. However, this assumption is often violated in many practical problems. Remillard (2009) proposed to extend the distance covariance methodology to a time series context in order to measure serial dependence. There have been few works on how to develop a distance covariance methodology in the context of time series (Zhou, 2012; Dueck et al., 2014; Davis et al., 2016). Motivated by the work of Székely et al. (2007), Zhou (2012) recently defined the so-called *auto-distance covariance function* (ADCV) - and its rescaled version, the so-called *auto-distance correlation function* (ADCF), for a strictly stationary multivariate time series. Compared to the classical Pearson autocorrelation function (ACF) which measures the strength of linear dependencies and can be equal to zero even when the variables are related, ADCF vanishes only in the case where the observations are independent. However, Zhou (2012) studied the asymptotic behavior of ADCV at a fixed lag order. Fokianos and Pitsillou (2016a) relaxed this assumption and constructed a univariate test of independence by considering an increasing number of lags following Hong's (1999) generalized spectral domain methodology. Although the proposed methodology is for univariate processes, it can be extended for multivariate processes.

Zhou (2012) developed a distance covariance methodology for multivariate time series, but he did not explore the interrelationships between the various time series components. Fokianos and Pitsillou (2016b) made this possible by defining the matrix version of pairwise auto-distance covariance and correlation functions. In particular, they construct multivariate tests of independence based on these new measures in order to identify whether there is some inherent nonlinear interdependence between the component series.

The **energy** (Rizzo and Székely, 2014) package for R is a package that involves a wide range of functions for the existing distance covariance methodology. However, there is no package for the aforementioned distance covariance methodology in time series. Thus, we aim at filling this gap by publishing an R-package named **dCovTS**. In this first version of the package, we provide functions that compute and plot ADCV and ADCF using the functions `dcov()` and `dcor()` respectively from **energy** package. The new testing methodology proposed by Fokianos and Pitsillou (2016a,b) is also included in the package.

The structure of the paper is as follows. In the first two sections we introduce the theoretical background of distance covariance function for both univariate and multivariate time series respectively. In the next section, we briefly state the main results about the asymptotic properties of distance covariance function. The proposed testing methodology for both univariate and multivariate time series are also described. Empirical  $p$ -values of the tests and empirical critical values for the distance correlation plots are computed via the wild bootstrap methodology (Dehling and Mikosch, 1994; Shao, 2010; Leucht and Neumann, 2013b) which is explained in the corresponding section. The implementation

section demonstrates the usage of the package with two real data examples. Lastly, we give some concluding remarks and some further points for future extensions of the **dCovTS** package.

## Distance covariance function

Denote a univariate strictly stationary time series by  $\{X_t, t \in \mathbb{Z}\}$ . Motivated by Székely et al. (2007) and Zhou (2012), we define the distance covariance function as a function of the joint and marginal characteristic functions of the pair  $(X_t, X_{t+j})$ . Denote by  $\phi_j(u, v)$  the joint characteristic function of  $X_t$  and  $X_{t+j}$ ; that is

$$\phi_j(u, v) = E\left[\exp\left(i\left(uX_t + vX_{t+j}\right)\right)\right], \quad j = 0, \pm 1, \pm 2, \dots,$$

and the marginal characteristic functions of  $X_t$  and  $X_{t+j}$  as  $\phi(u) := \phi_j(u, 0)$  and  $\phi(v) := \phi_j(0, v)$  respectively, where  $(u, v) \in \mathbb{R}^2$ , and  $i^2 = -1$ . For a strictly stationary  $\alpha$ -mixing univariate time series, Hong (1999) defined a new measure of dependence between the joint characteristic function of  $X_t$  and its lagged observation  $X_{t+j}$  and the product of their marginals, namely

$$\sigma_j(u, v) = \phi_j(u, v) - \phi(u)\phi(v), \quad j = 0, \pm 1, \pm 2, \dots, \quad (1)$$

where  $(u, v) \in \mathbb{R}^2$ . Considering the property that the joint characteristic function factorizes under independence of  $X_t$  and  $X_{t+j}$ ,  $\sigma_j(u, v)$  equals 0 if and only if  $X_t$  and  $X_{t+j}$  are independent. Thus, compared to the classical autocorrelation function (ACF),  $\sigma_j(\cdot, \cdot)$  can capture all pairwise dependencies including those with zero autocorrelation. The auto-distance covariance function (ADCV),  $V_X(j)$ , between  $X_t$  and  $X_{t+j}$  is then defined as the square root of

$$V_X^2(j) = \int_{\mathbb{R}^2} \left| \sigma_j(u, v) \right|^2 d\mathcal{W}(u, v), \quad j = 0, \pm 1, \pm 2, \dots \quad (2)$$

where  $\mathcal{W}(\cdot, \cdot)$  is a positive weight function for which the above integral exists.

Although Hong (1999) suggests the use of an arbitrary integrable weight function,  $\mathcal{W}(\cdot, \cdot)$ , we propose the use of a non-integrable weight function, i.e.

$$\mathcal{W}(u, v) = \mathcal{W}_0(u)\mathcal{W}_0(v) = \frac{1}{\pi |u|^2} \frac{1}{\pi |v|^2}, \quad (u, v) \in \mathbb{R}^2 \quad (3)$$

which avoids missing any potential dependence among observations (Székely et al., 2007, p. 2771). Rescaling (2), one can define the auto-distance correlation function (ADCF) as the positive square root of

$$R_X^2(j) = \frac{V_X^2(j)}{V_X^2(0)}, \quad j = 0, \pm 1, \pm 2, \dots \quad (4)$$

for  $V_X^2(0) \neq 0$  and zero otherwise. Székely et al. (2007) showed that by applying a non-integrable weight function, like (3), ADCF is scale invariant and is not zero under dependence.

The empirical ADCV,  $\hat{V}_X(\cdot)$ , is the non-negative square root of

$$\hat{V}_X^2(j) = \frac{1}{(n-j)^2} \sum_{r,l=1+j}^n A_{rl}B_{rl}, \quad 0 \leq j \leq (n-1) \quad (5)$$

and  $\hat{V}_X^2(-j) = \hat{V}_X^2(j)$ , for  $-(n-1) \leq j < 0$ , where  $A = A_{rl}$  and  $B = B_{rl}$  are Euclidean distance matrices given by

$$A_{rl} = a_{rl} - \bar{a}_{r.} - \bar{a}_{.l} + \bar{a}_{..},$$

with  $a_{rl} = |X_r - X_l|$ ,  $\bar{a}_{r.} = (\sum_{l=1+j}^n a_{rl}) / (n-j)$ ,  $\bar{a}_{.l} = (\sum_{r=1+j}^n a_{rl}) / (n-j)$ ,  $\bar{a}_{..} = (\sum_{r,l=1+j}^n a_{rl}) / (n-j)^2$ .  $B_{rl}$  is defined analogously in terms of  $b_{rl} = |Y_r - Y_l|$ , where  $Y_t \equiv X_{t+j}$ . Székely and Rizzo (2014) proposed an unbiased version of the sample distance covariance. In the context of time series data this is given by

$$\tilde{V}_X^2(j) = \frac{1}{(n-j)(n-j-3)} \sum_{r \neq l} \tilde{A}_{rl}\tilde{B}_{rl}, \quad (6)$$

for  $n > 3$ , where  $\tilde{A}_{rl}$  is the  $(r, l)$  element of the so-called  $\mathcal{U}$ -centered matrix  $\tilde{A}$ , defined by

$$\tilde{A}_{rl} = \begin{cases} a_{rl} - \frac{1}{n-j-2} \sum_{t=1+j}^n a_{rt} - \frac{1}{n-j-2} \sum_{s=1+j}^n a_{sl} + \frac{1}{(n-j-1)(n-j-2)} \sum_{t,s=1+j}^n a_{ts}, & r \neq l; \\ 0, & r = l. \end{cases}$$

The empirical ADCF,  $\widehat{R}_X(j)$  (or its unbiased version,  $\widetilde{R}_X(j)$ ), can be obtained by replacing (5) (or (6)) into (4). The functions `ADCV()` and `ADCF()` in **dCovTS** return the empirical quantities  $\widehat{V}_X(\cdot)$  and  $\widehat{R}_X(\cdot)$  respectively. Using the same functions with argument `unbiased=TRUE`, the results correspond to the unbiased squared quantities  $\widetilde{V}_X^2(\cdot)$  and  $\widetilde{R}_X^2(\cdot)$ . Note that the default option has been set to `unbiased=FALSE` (corresponding to (5)).

## Distance covariance matrix

We denote by  $\{\mathbf{X}_t : t = 0, \pm 1, \pm 2, \dots\}$  a  $d$ -dimensional time series process, with components  $\{X_{t,i}\}_{i=1}^d$ . The characteristic functions can be defined in analogous way as in the univariate case. In particular, the joint characteristic function of  $X_{t,r}$  and  $X_{t+j;m}$  is given by

$$\phi_j^{(r,m)}(u, v) = E\left[\exp\left(i(uX_{t,r} + vX_{t+j;m})\right)\right], \quad j = 0, \pm 1, \pm 2, \dots$$

and the marginal characteristic functions of  $X_{t,r}$  and  $X_{t+j;m}$  by  $\phi^{(r)}(u) := \phi_j^{(r,m)}(u, 0)$  and  $\phi^{(m)}(v) := \phi_j^{(r,m)}(0, v)$  respectively, with  $(u, v) \in \mathbb{R}^2$ ,  $r, m = 1, \dots, d$  and  $i^2 = -1$ . The pairwise ADCV between  $X_{t,r}$  and  $X_{t+j;m}$  is denoted by  $V_{rm}(j)$  and it is defined as the non-negative square root of

$$V_{rm}^2(j) = \int_{\mathbb{R}^2} \left| \sigma_j^{(r,m)}(u, v) \right|^2 \mathcal{W}(u, v) du dv, \quad j = 0, \pm 1, \pm 2, \dots$$

where  $\mathcal{W}(\cdot, \cdot)$  is given by (3) and  $\sigma_j^{(r,m)}(u, v)$  is similarly defined as in the univariate case, namely

$$\sigma_j^{(r,m)}(u, v) = \phi_j^{(r,m)}(u, v) - \phi^{(r)}(u)\phi^{(m)}(v).$$

Clearly,  $V_{rm}^2(j) \geq 0$ ,  $\forall j$  and  $X_{t,r}$  and  $X_{t+j;m}$  are independent if and only if  $V_{rm}^2(j) = 0$ . The ADCV matrix,  $V(j)$ , is then defined by

$$V(j) = \left[ V_{rm}(j) \right]_{r,m=1}^d, \quad j = 0, \pm 1, \pm 2, \dots \quad (7)$$

The pairwise ADCF between  $X_{t,r}$  and  $X_{t+j;m}$ ,  $R_{rm}(j)$ , is a coefficient that lies in the interval  $[0, 1]$  and also measures dependence and is defined as the positive square root of

$$R_{rm}^2(j) = \frac{V_{rm}^2(j)}{\sqrt{V_{rr}^2(0)} \sqrt{V_{mm}^2(0)}}, \quad (8)$$

for  $V_{rr}(0)V_{mm}(0) \neq 0$  and zero otherwise. The ADCF matrix of  $\mathbf{X}_t$ , is then defined as

$$R(j) = \left[ R_{rm}(j) \right]_{r,m=1}^d, \quad j = 0, \pm 1, \pm 2, \dots$$

$V_{rm}(j)$  measures the dependence of  $X_{t,r}$  on  $X_{t+j;m}$ . In general,  $V_{rm}(j) \neq V_{mr}(j)$  for  $r \neq m$ , since they measure different dependence structure between the series  $\{X_{t,r}\}$  and  $\{X_{t;m}\}$  for all  $r, m = 1, 2, \dots, d$ . Thus,  $V(j)$  and  $R(j)$  are non-symmetric matrices. Moreover, because of the assumed stationarity and relation  $\text{Cov}(x, y) = \text{Cov}(y, x)$ ,  $V(j) = V'(-j)$  and consequently  $R(j) = R'(-j)$ . More properties of these new defined functions can be found in [Fokianos and Pitsillou \(2016b\)](#).

Estimation of  $V_{rm}^2(\cdot)$  can be dealt in a similar way as in the univariate case. In particular, let first  $Y_{t;m} \equiv X_{t+j;m}$ . Based on the sample  $\{(X_{t,r}, Y_{t;m}) : t = 1 + j, \dots, n\}$ , we define the Euclidean distance matrices by  $(a_{ts}^r) = |X_{t,r} - X_{s,r}|$  and  $(b_{ts}^m) = |Y_{t;m} - Y_{s;m}|$  and the centered distance matrices by

$$\begin{aligned} A_{ts}^r &= a_{ts}^r - \bar{a}_{t.}^r - \bar{a}_{s.}^r + \bar{a}_{..}^r, \\ B_{ts}^m &= b_{ts}^m - \bar{b}_{t.}^m - \bar{b}_{s.}^m + \bar{b}_{..}^m, \end{aligned}$$

where the quantities in the right hand side are defined analogously as those defined in the univariate

case. The biased estimator of  $V_{rm}^2(\cdot)$  is then given by

$$\hat{V}_{rm}^2(j) = \begin{cases} \frac{1}{(n-j)^2} \sum_{t,s=1+j}^n A_{ts}^r B_{ts}^m, & 0 \leq j \leq (n-1); \\ \frac{1}{(n+j)^2} \sum_{t,s=1}^{n+j} A_{ts}^r B_{ts}^m, & -(n-1) \leq j < 0. \end{cases} \quad (9)$$

Analogously to (6), an unbiased estimator of  $\hat{V}_{rm}^2(\cdot)$  is given by

$$\tilde{V}_{rm}^2(j) = \begin{cases} \frac{1}{(n-j)(n-j-3)} \sum_{t,s=1+j}^n \tilde{A}_{ts}^r \tilde{B}_{ts}^m, & 0 \leq j \leq (n-1); \\ \frac{1}{(n+j)(n+j-3)} \sum_{t,s=1}^{n+j} \tilde{A}_{ts}^r \tilde{B}_{ts}^m, & -(n-1) \leq j < 0, \end{cases} \quad (10)$$

where  $\tilde{A}_{ts}^r$  are computed appropriately.

The sample ADCV matrix,  $\hat{V}(\cdot)$ , is then obtained by replacing its elements by the positive square root of (9) and can be calculated from **dCovTS** using the `mADCV()` function. The estimator based on (10),  $\tilde{V}(\cdot)$ , is obtained from **dCovTS** using the argument `unbiased = TRUE`. The package also gives the sample ADCF matrix  $\hat{R}(\cdot)$  (function `mADCF()`) which is obtained after replacing (9) (or  $\tilde{R}_X(j)$  which is based on (6)) into (8). The distance correlation plots for both univariate and multivariate time series are obtained by the `ADCFplot()` and `mADCFplot()` functions respectively, where the shown critical values (blue dotted horizontal line) are computed by employing bootstrap methodology described in the appropriate section. Recall that these are computed by using the biased definition of distance covariance and correlation.

## Consistency and asymptotic distribution of distance covariance

Consider first the univariate case. For a strictly stationary and  $\alpha$ -mixing process  $X_t$ , with  $E|X_t| < \infty$ , then for all  $j = 0, \pm 1, \pm 2, \dots$

$$\hat{V}_X^2(j) \rightarrow V_X^2(j)$$

almost surely, as  $n \rightarrow \infty$ . A detailed proof of this result can be found in [Fokianos and Pitsillou \(2016a\)](#). Under mild conditions, [Zhou \(2012\)](#) obtained the weak consistency of  $\hat{V}_X^2(\cdot)$  and its asymptotic distribution at a fixed lag, but under alternative mixing conditions.

In addition, [Fokianos and Pitsillou \(2016b\)](#) showed that for a  $d$ -dimensional strictly stationary and ergodic time series process  $\{\mathbf{X}_t\}$  with  $E|X_{t,r}| < \infty$  for  $r = 1, \dots, d$ , then for all  $j = 0, \pm 1, \pm 2, \dots$

$$\hat{V}(j) \rightarrow V(j)$$

almost surely as  $n \rightarrow \infty$ . Under pairwise independence, the empirical pairwise ADCV is a degenerate  $V$ -statistic of order two with a measurable kernel function that is symmetric, continuous and positive semidefinite Then

$$(n-j)\hat{V}_X^2(j) \rightarrow Z := \sum_k \lambda_k Z_k^2 \quad (11)$$

in distribution, as  $n \rightarrow \infty$ , where  $\{Z_k\}$  is an iid sequence of  $N(0, 1)$  random variables, and  $(\lambda_k)$  is a sequence of nonzero eigenvalues. A similar result showing the limiting distribution of  $\hat{V}_{rm}(\cdot)$  can be obtained by replacing  $\hat{V}_X(\cdot)$  by  $\hat{V}_{rm}(\cdot)$  in (11).

## Testing for pairwise dependence in univariate time series

As shown in the previous section, the asymptotic distribution of distance covariance is derived at a fixed lag, for both univariate and multivariate time series. [Fokianos and Pitsillou \(2016a,b\)](#) constructed the asymptotic behavior of distance covariance considering an increasing number of lags by employing [Hong's \(1999\)](#) generalized spectral domain methodology. [Hong \(1999\)](#) highlighted that standard spectral density approaches become inappropriate for non-Gaussian and nonlinear processes with zero autocorrelation. Considering a univariate strictly stationary  $\alpha$ -mixing process, he proposed

the generalized spectral density, which is the Fourier transform of  $\sigma_j(u, v)$  defined in (1), given by

$$f(\omega, u, v) = \frac{1}{2\pi} \sum_{j=-\infty}^{\infty} \sigma_j(u, v) e^{-ij\omega}.$$

Under the null hypothesis of independence, the corresponding null density is given by

$$f_0(\omega, u, v) = \frac{1}{2\pi} \sigma_0(u, v), \quad \omega \in [-\pi, \pi].$$

Any deviation of  $f$  from  $f_0$  is a strong evidence of pairwise dependence. Thus, Hong (1999) compares the Parzen's (1957) kernel-type estimators  $\hat{f}(\omega, u, v)$  and  $\hat{f}_0(\omega, u, v)$  via an  $L_2$ -norm resulting in a test statistic of the form

$$T_n^{(2)} = \int_{\mathbb{R}^2} \sum_{j=1}^{n-1} (n-j) k^2(j/p) |\hat{\sigma}_j(u, v)| d\mathcal{W}(u, v), \quad (12)$$

where  $\mathcal{W}(\cdot, \cdot) : \mathbb{R}^2 \rightarrow \mathbb{R}$  is an arbitrary nondecreasing function with bounded total variation,  $p$  is a bandwidth of the form  $p = cn^\lambda$  for  $c > 0$ ,  $\lambda \in (0, 1)$  and  $k(\cdot)$  is a Lipschitz-continuous kernel function satisfying the following assumption:

**Assumption 1.**  $k : \mathbb{R} \rightarrow [-1, 1]$  is symmetric and is continuous at 0 and at all but a finite number of points, with  $k(0) = 1$ ,  $\int_{-\infty}^{\infty} k^2(z) dz < \infty$  and  $|k(z)| \leq C |z|^{-b}$  for large  $z$  and  $b > 1/2$ .

The function `kernelFun()` in **dCovTS** computes a number of such kernel functions including the truncated (default option), Bartlett, Daniell, QS and Parzen kernels.

Fokianos and Pitsillou (2016a) proposed a portmanteau type statistic based on ADCV

$$T_n = \sum_{j=1}^{n-1} (n-j) k^2(j/p) \hat{V}_X^2(j). \quad (13)$$

Under the null hypothesis that the data are iid and some further assumptions about the kernel function  $k(\cdot)$ , the standardized version of  $T_n$  follows a  $N(0, 1)$  asymptotically, and it is consistent. The authors also considered a similar test statistic based on ADCF

$$\sum_{j=1}^{n-1} (n-j) k^2(j/p) \hat{R}_X^2(j). \quad (14)$$

The function `UnivTest` from **dCovTS** package performs univariate tests of independence based on (13) and its rescaled version (14), using the arguments `testType = "covariance"` and `testType = "correlation"` respectively.

## Testing for pairwise dependence in multivariate time series

Following a similar methodology described in the previous section, Fokianos and Pitsillou (2016b) suggested a test statistic suitable for testing pairwise independence in a multivariate time series framework. The proposed test statistic is based on the ADCV matrix (7), and it is given by

$$\tilde{T}_n = \sum_{j=1}^{n-1} (n-j) k^2(j/p) \text{tr} \hat{V}^*(j) \hat{V}(j). \quad (15)$$

where  $k(\cdot)$  is a univariate kernel function satisfying Assumption 1,  $p$  is a bandwidth as described before. Moreover,  $\hat{V}^*(\cdot)$  denotes the complex conjugate matrix of  $\hat{V}(\cdot)$  and  $\text{tr}(A)$  denotes the trace of the matrix  $A$ . The authors formed the statistic (15) in terms of the ADCF matrix as follows

$$\bar{T}_n = \sum_{j=1}^{n-1} (n-j) k^2(j/p) \text{tr} \hat{V}^*(j) \hat{D}^{-1} \hat{V}(j) \hat{D}^{-1}, \quad (16)$$

where  $D = \text{diag}\{V_{rr}(0), r = 1, 2, \dots, d\}$ . Under the null hypothesis of independence and some further assumptions about the kernel function  $k(\cdot)$ , the standardized version of the test statistics  $\tilde{T}_n$  and  $\bar{T}_n$  given in (15) and (16) were proved to follow  $N(0, 1)$  asymptotically and they are consistent. The multivariate tests of independence based on  $\tilde{T}_n$  and  $\bar{T}_n$  are performed via `mADCVtest()` and `mADCFtest()` respectively in **dCovTS** package.

**Table 1:** Functions in **dCovTS**

Function	Description
ADCF, mADCF	Estimates distance correlation for a univariate and multivariate time series respectively
ADCV, mADCV	Estimates distance covariance for a univariate and multivariate time series respectively
ADCFplot, mADCFplot	Plots sample distance correlation in a univariate and multivariate time series framework respectively
kernelFun	Gives a range of univariate kernel function, $k(\cdot)$ , that satisfy Assumption 1
UnivTest	Performs a univariate test of independence based on $T_n$
mADCFtest, mADCVtest	Perform multivariate tests of independence based on $\bar{T}_n$ and $\tilde{T}_n$ respectively

## Bootstrap methodology

To examine the asymptotic behavior of the proposed test statistics, a resampling method is proposed. First, recall that all test statistics  $T_n$ ,  $\bar{T}_n$  and  $\tilde{T}_n$  of equations (13), (15) and (16) respectively, are functions of degenerate  $V$ -statistics of order two. Dehling and Mikosch (1994) proposed wild bootstrap techniques to approximate the distribution of degenerate  $U$ -statistics for the case of iid data. Recently, Leucht and Neumann (2013a,b) suggested the use of a new variant of dependent wild bootstrap (Shao, 2010) to approximate the limit distribution of degenerate  $U$ - and  $V$ -statistics for dependent data. The method relies on generating auxiliary random variables  $(W_{tn}^*)_{t=1}^{n-j}$ . Shao (2010) highlighted that the methodology of wild bootstrap for time series extends that of Wu (1986) by allowing the auxiliary random variables  $W_{tn}^*$  to be dependent. In particular, Leucht and Neumann (2013b) proposed to generate the sequence  $W_{tn}^*$  by a first order autoregressive model. In the case of independent data, Dehling and Mikosch (1994) studied the wild bootstrap methodology by employing independent auxiliary variables  $W_{tn}^*$ . Because our focus is on testing independence we implement the calculation of the test statistics by using  $W_{tn}^*$  iid standard normal random variables. Thus, the empirical  $p$ -values of the tests are derived based on this methodology.

We also suggest the use of independent wild bootstrap for obtaining simultaneous 95% empirical critical values for the distance correlation plots. In the case of a univariate time series, we additionally propose the subsampling approach suggested by Zhou (2012, Section 5.1) for computing the pairwise 95% critical values (argument `method = "Subsampling"`). The choice of the subsampling block size is based on the minimum volatility method proposed by Politis et al. (1999, Section 9.4.2). In addition, the package provides the ordinary independent bootstrap methodology to derive empirical  $p$ -values of the tests and simultaneous 95% critical values for the ADCF plots (argument `method = "Independent Bootstrap"`). The default bootstrap method provided to the user is the independent wild bootstrap technique.

The computation of the bootstrap replications, and thus the empirical  $p$ -values and the critical values, can be distributed to multiple cores simultaneously (argument `parallel = TRUE`). To do this, the `doParallel` (Analytics and Weston, 2015) package needs to be installed first, in order to register a computing cluster.

## Implementation of **dCovTS** package

The current version of **dCovTS** package (version number 1.1) is available from CRAN and can be downloaded via <https://cran.r-project.org/web/packages/dCovTS/>. The aim of the **dCovTS** package is to provide a set of functions that compute and plot distance covariance and correlation functions in both univariate and multivariate time series. As we mentioned, the package supports both versions of biased and unbiased estimators of distance covariance and correlation functions. Moreover, it offers functions that perform univariate and multivariate tests of independence based on distance covariance function using the biased estimator (corresponding to (5) and (9)). All these functions are provided in Table 1. Apart from these functions, the package also provides two real datasets listed in Table 2. A more detailed description of the functions and datasets can be found in the help files. We apply **dCovTS** to two real data examples.

**Table 2:** Datasets in **dCovTS**

Data	Description
ibmSp500	Monthly returns of IBM and S&P 500 composite index from January 1926 to December 2011
MortTempPart	Mortality, temperature and pollution data measured daily in Los Angeles County over the period 1970-1979

### Regression with autocorrelated errors

We first consider the pollution, temperature and mortality data measured daily in Los Angeles County over the 10 year period 1970-1979 ([Shumway et al., 1988](#)). The data are available in our package by the argument `MortTempPart` and contain 508 observations and 3 variables representing the mortality ("cmort"), temperature ("tempr") and pollutant particulates ("part") data.

```
library(dCovTS)
data(MortTempPart)
MortTempPart[1:10,] # the first ten observations
##      cmort tempr part
## 1    97.85 72.38 72.72
## 2   104.64 67.19 49.60
## 3    94.36 62.94 55.68
## 4    98.05 72.49 55.16
## 5    95.85 74.25 66.02
## 6    95.98 67.88 44.01
## 7    88.63 74.20 47.83
## 8    90.85 74.88 43.60
## 9    92.06 64.17 24.99
## 10   88.75 67.09 40.41
attach(MortTempPart)
```

Following the analysis of [Shumway and Stoffer \(2011\)](#), the possible effects of temperature ( $T_t$ ) and pollutant particulates ( $P_t$ ) on daily cardiovascular mortality ( $M_t$ ) are examined via regression. In particular, once the temperature is adjusted for its mean ( $T_ = 74.3$ ), we fit the following regression model using the function `lm()`

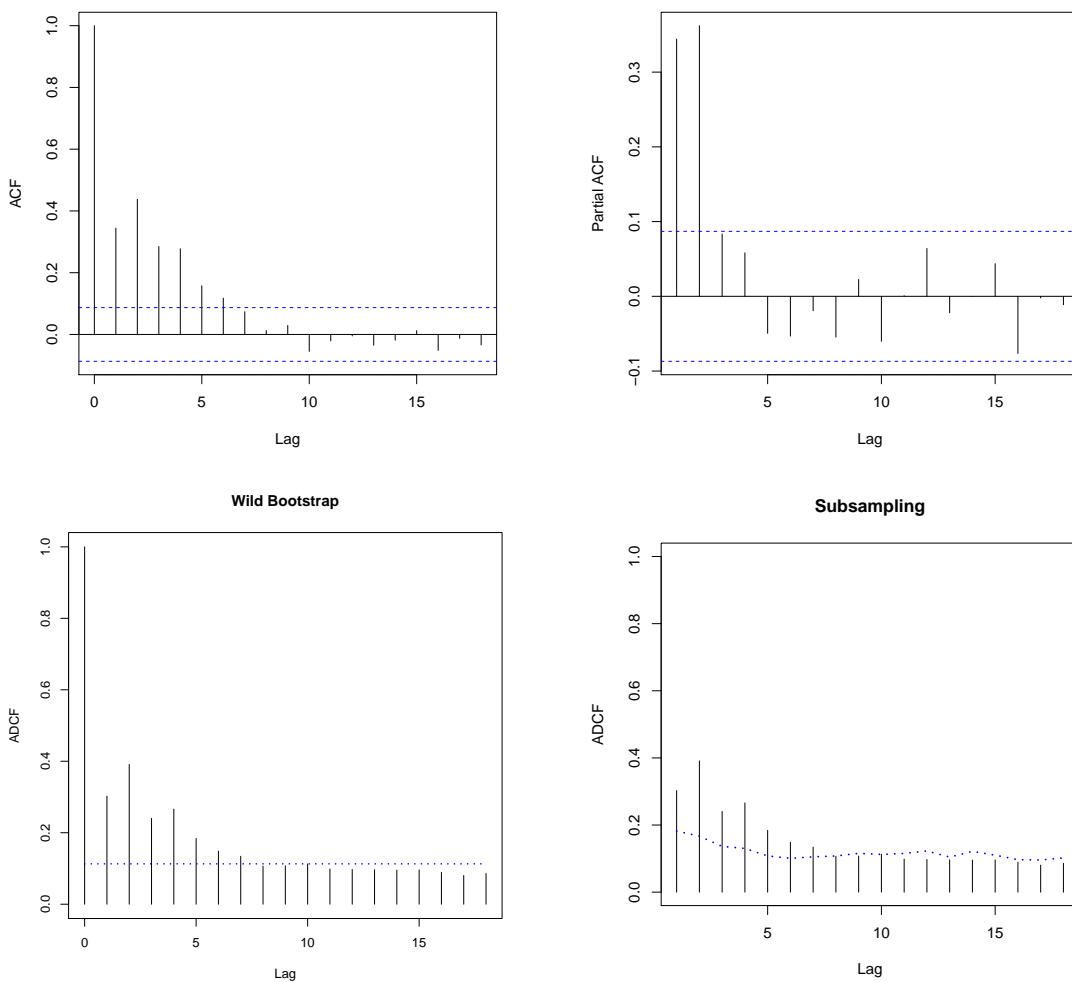
$$\begin{aligned}\hat{M}_t = & 2831.49 - 1.396_{(0.101)} t - 0.472_{(0.032)} (T_t - T.) \\ & + 0.023_{(0.003)} (T_t - T.)^2 + 0.255_{(0.019)} P_t,\end{aligned}\quad (17)$$

where the standard errors of the estimators are given in parentheses. Figure 1 provides the sample autocorrelation (ACF), partial correlation (PACF) and ADCF plots of the residuals of model (17). The plots shown in Figure 1 suggest an AR(2) process for the residuals. The new fit is

$$\begin{aligned}\hat{M}_t = & 3075.15 - 1.517_{(0.423)} t - 0.019_{(0.050)} (T_t - T.) \\ & + 0.015_{(0.002)} (T_t - T.)^2 + 0.155_{(0.027)} P_t,\end{aligned}\quad (18)$$

where the standard errors of the estimators are given in parentheses. The above model fit was derived by using the `arima()` function of R. The correlation plots for the residuals from the new model (18) are shown in Figure 2 indicating that there is no serial dependence. The calls for both model fits and their diagnostic plots are given below. ADCF plots (lower plots of Figures 1 and 2) are constructed using both resampling schemes explained in the previous section: independent wild bootstrap (with  $b = 499$  replications) and subsampling.

```
temp <- tempr - mean(tempr) # center temperature
temp2 <- temp^2
trend <- time(cmort)
fit <- lm(cmort ~ trend + temp + temp2 + part, na.action = NULL)
Residuals <- as.numeric(resid(fit))
##Correlation plots
acf(Residuals, lag.max = 18, main = "")
pacf(Residuals, lag.max = 18, main = "")
ADCFplot(Residuals, MaxLag = 18, main = "Wild Bootstrap", method = "Wild")
ADCFplot(Residuals, MaxLag = 18, main = "Subsampling", method = "Subsampling")
```

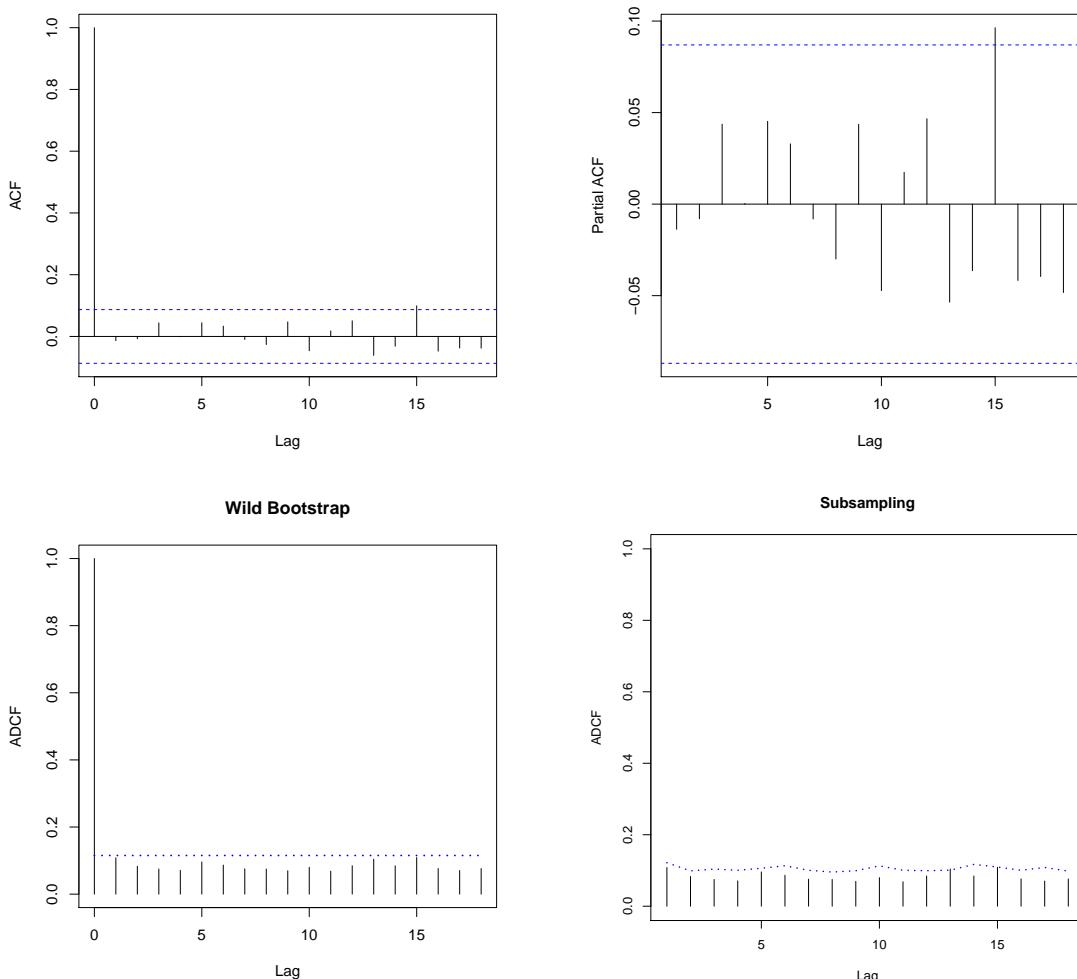


**Figure 1:** Sample ACF, PACF and ADCF plots of the mortality residuals of model (17).

```
fit2 <- arima(cmort, order =c(2, 0, 0), xreg = cbind(trend, temp, temp2, part))
Residuals2 <- as.numeric(residuals(fit2))
##Correlation plots
acf(Residuals2, lag.max = 18, main = "")
pacf(Residuals2, lag.max = 18, main = "")
ADCFplot(Residuals2, MaxLag = 18, main = "Wild Bootstrap", method = "Wild")
ADCFplot(Residuals2, MaxLag = 18, main = "Subsampling", method = "Subsampling")
```

To formally confirm the absence of any serial dependence among the new residuals of model (18), as shown in Figure 2, we perform univariate tests of independence based on the test statistic  $T_n$  given in (13). We use the `UnivTest()` function from our package with argument `testType = "covariance"` (default option). In order to examine the effect of using different bandwidths, we choose  $p = [3n^\lambda]$  for  $\lambda = 0.1, 0.2$  and  $0.3$ , that is  $p = 6, 11$ , and  $20$ , and we apply Bartlett kernel. The resulting  $p$ -values are  $0.118, 0.170$  and  $0.208$  respectively suggesting acceptance of independence. We calculated  $p$ -values for  $b = 499$  independent wild bootstrap replications. The bootstrap procedure can be computed on multiple cores simultaneously by using the argument `parallel = TRUE` (they take about 10, 14 and 23 seconds respectively on a standard laptop with Intel Core i5 system and CPU 2.30 GHz):

```
UnivTest(Residuals2, type = "bartlett", p = 6, b = 499, parallel = TRUE)
##           Univariate test of independence based on distance covariance
##           data: Residuals2, kernel type: bartlett, bandwidth=6, boot replicates 499
```



**Figure 2:** Sample ACF, PACF and ADCF plots of the mortality residuals of model (18) indicating that the new residuals can be taken as white noise.

```

## Tn = 67.7344, p-value = 0.118

UnivTest(Residuals2, type = "bartlett", p = 11, b = 499, parallel = TRUE)

##      Univariate test of independence based on distance covariance
##
## data: Residuals2, kernel type: bartlett, bandwidth=11, boot replicates 499
## Tn = 125.6674, p-value = 0.170

UnivTest(Residuals2, type = "bartlett", p = 20, b = 499, parallel = TRUE)

##      Univariate test of independence based on distance covariance
##
## data: Residuals2, kernel type: bartlett, bandwidth=20, boot replicates 499
## Tn = 225.9266, p-value = 0.208

```

We compare the proposed test statistic with other test statistics to check its performance. In particular, we consider the Box-Pierce (Box and Pierce, 1970) test statistic

$$BP = n \sum_{j=1}^p \hat{\rho}^2(j),$$

the Ljung-Box (Ljung and Box, 1978) test statistic

$$LB = n(n+2) \sum_{j=1}^p (n-j)^{-1} \hat{\rho}^2(j),$$

the test statistic proposed by Hong (1996)

$$T_n^{(1)} = n \sum_{j=1}^{n-1} k^2(j/p) \hat{\rho}^2(j)$$

and the test statistic  $T_n^{(2)}$  proposed by Hong (1999) defined in (12) with  $\mathcal{W}(u, v) = \Phi(u)\Phi(v)$ , and  $\Phi(\cdot)$  being the cumulative distribution function of standard normal. For the aforesated bandwidth values, all these alternative test statistic give large  $p$ -values indicating the absence of any serial dependence among the new residuals. More precisely, BP and LB give 0.848, 0.906, 0.170 and 0.844, 0.901, 0.142 respectively. BP and LB based tests are performed in R by the function `Box.test()` as follows:

```
box1 <- Box.test(Residuals2, lag = 6)
box2 <- Box.test(Residuals2, lag = 11)
box3 <- Box.test(Residuals2, lag = 20)
ljung1 <- Box.test(Residuals2, lag = 6, type = "Ljung")
ljung2 <- Box.test(Residuals2, lag = 11, type = "Ljung")
ljung3 <- Box.test(Residuals2, lag = 20, type = "Ljung")
```

The  $p$ -values obtained by  $T_n^{(1)}$  are 0.896, 0.930 and 0.870 respectively.  $T_n^{(2)}$  gives the following  $p$ -values: 0.854, 0.752 and 0.504 respectively.  $T_n^{(1)}$  and  $T_n^{(2)}$  are calculated by employing the Bartlett kernel. These  $p$ -values are calculated for  $b = 499$  ordinary bootstrap replications. The R functions for constructing these test statistics are beyond the scope of this paper and are available from the authors upon request.

## Bivariate financial time series

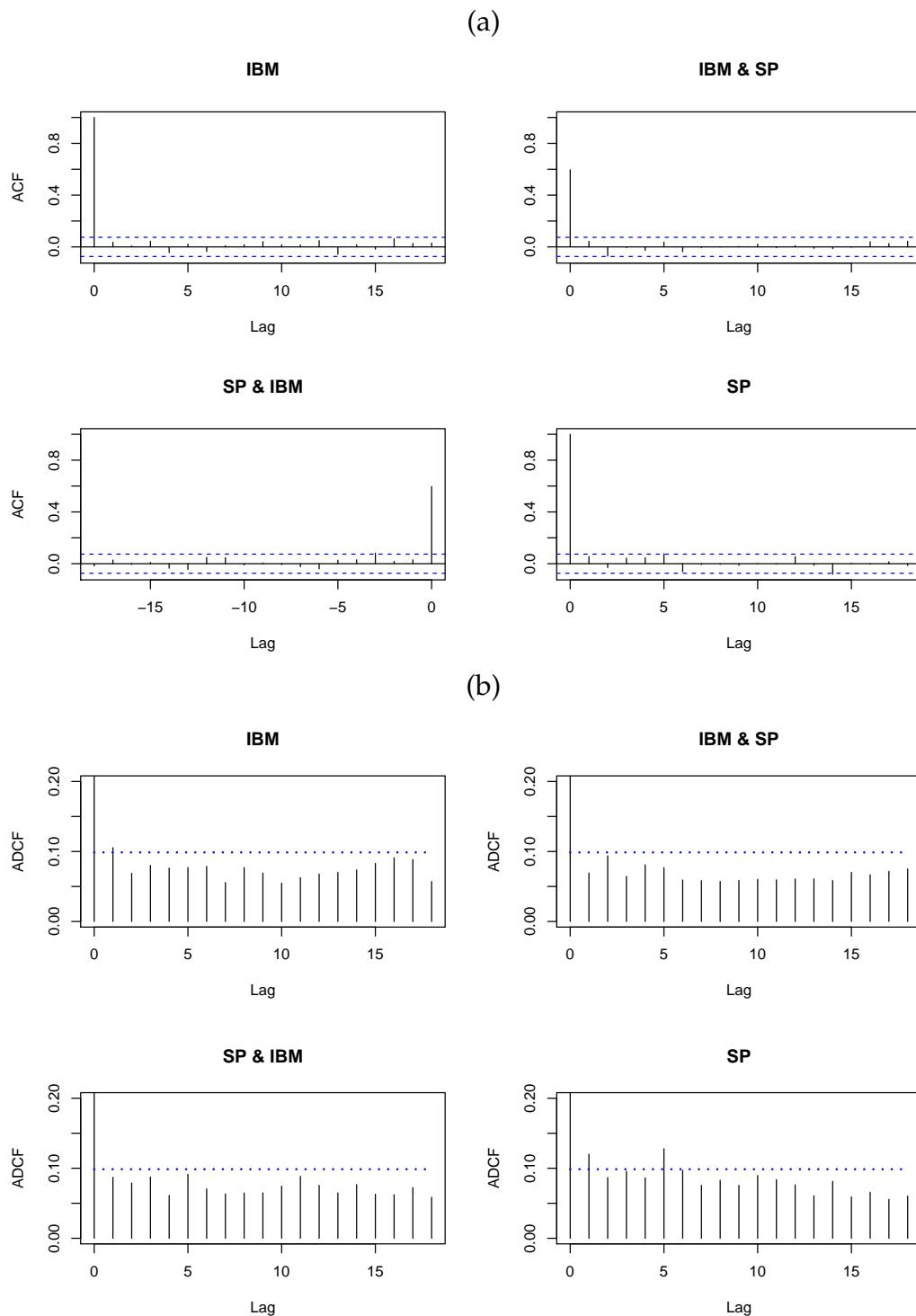
We now analyze the monthly log returns of the stocks of International Business Machines (IBM) and the S&P 500 composite index starting from 30 September 1953 to 30 December 2011 for 700 observations. A larger dataset is available in our package by the object `ibmSp500` starting from January 1926 for 1032 observations. It is actually a combination of two smaller datasets: the first one was first reported by Tsay (2010) and the second one was first reported by Tsay (2014). ACF and ADCF plots of the original series are provided in Figure 3, whereas Figure 4 shows the ACF and ADCF plots of the squared series.

The R commands for constructing these plots are as follows:

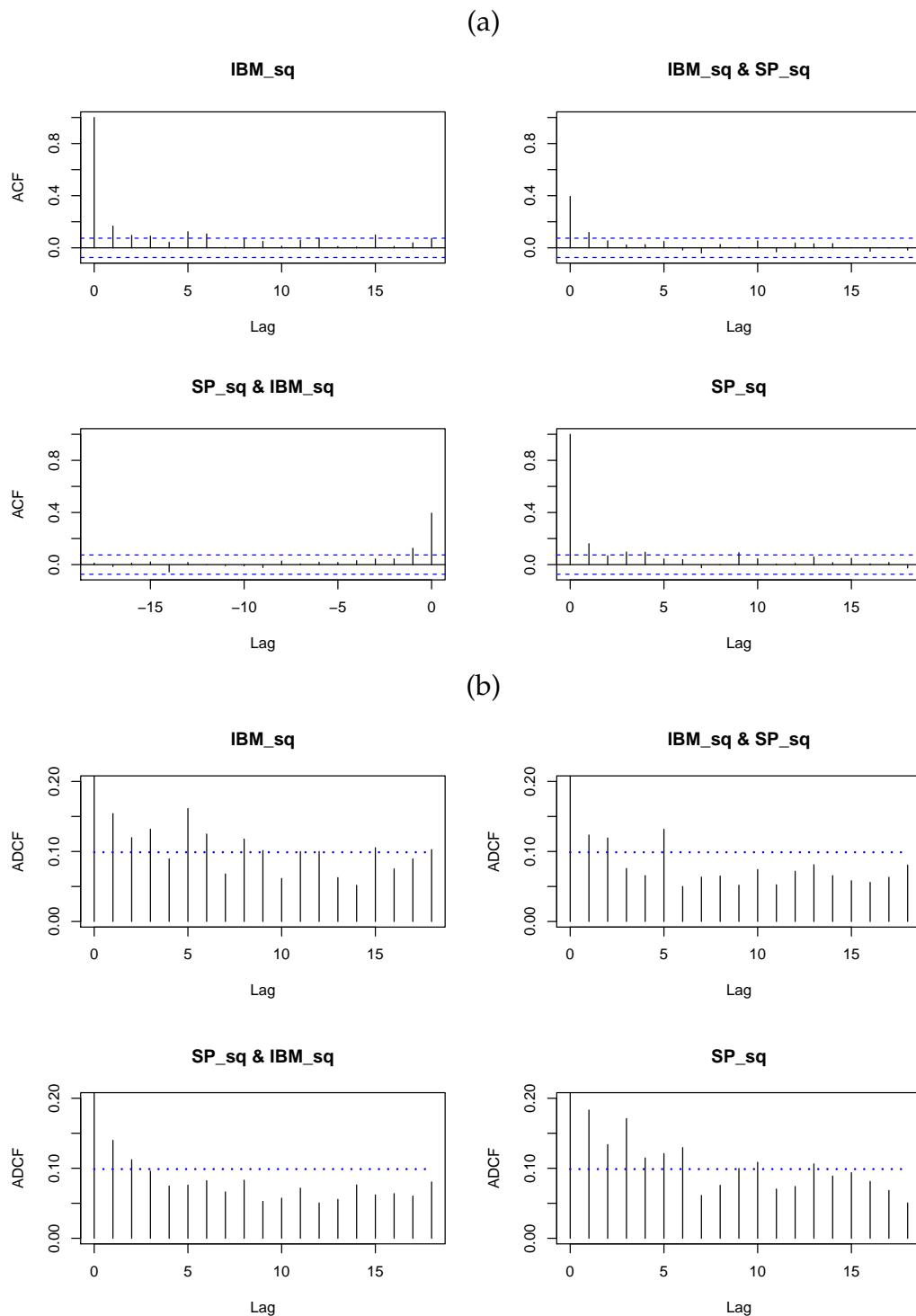
```
data(ibmSp500)
new_data <- tail(ibmSp500[,2:3], 700)
series <- log(new_data + 1)
t=scale(lseries, center = TRUE, scale = FALSE)
t2 <- at^2
olnames(at) <- c("IBM", "SP")
olnames(at2) <- c("IBM_sq", "SP_sq")
cf(at, lag.max = 18)
cf(at2, lag.max = 18)
ADCFplot(at, MaxLag = 18, ylim = c(0, 0.2))
ADCFplot(at2, MaxLag = 18, ylim = c(0, 0.2))
```

The ACF plots of the original series (upper panel of Figure 3) suggest no serial correlation among observations, while the ACF plots of the squared series (upper panel of Figure 4) imply strong dependence. This confirms the conditional heteroscedasticity in the monthly log returns. However, the ADCF plots for both original and squared series (lower panels of Figures 3 and 4) suggest dependence. Indeed, choosing  $p = [3n^\lambda]$  for  $\lambda = 0.1, 0.2$  and  $0.3$ , that is  $p = 6, 12$  and  $22$ , and employing Bartlett kernel,  $\bar{T}_n$  gives low  $p$ -values (0.022, 0.014 and 0.020 respectively). The calls for these three multivariate tests of independence can be found below (they take about 2, 3 and 6 minutes respectively for  $b = 499$  bootstrap replications on a standard laptop with Intel Core i5 system and CPU 2.30 GHz):

```
mADCFtest(at, "bartlett", p = 6, b = 499, parallel = TRUE)
```



**Figure 3:** (a) The sample ACF of the original series. (b) The sample ADCF of the original series.



**Figure 4:** (a) The sample ACF of the squared series. (b) The sample ADCF of the squared series.

```

##      Multivariate test of independence based on distance correlation
##
## data: at, kernel type: bartlett, bandwidth=6, boot replicates 499
## Tnbar = 34.1743, p-value = 0.022

mADCFtest(at, "bartlett", p = 12, b = 499, parallel = TRUE)

##      Multivariate test of independence based on distance correlation
##
## data: at, kernel type: bartlett, bandwidth=12, boot replicates 499
## Tnbar = 71.1713, p-value = 0.014

mADCFtest(at, "bartlett", p = 22, b = 499, parallel = TRUE)

##      Multivariate test of independence based on distance correlation
##
## data: at, kernel type: bartlett, bandwidth=22, boot replicates 499
## Tnbar = 122.9424, p-value = 0.02

```

To compare the performance of the proposed test statistic  $\bar{T}_n$ , we consider the multivariate Ljung-Box statistic (Hosking, 1980) defined by:

$$mLB = n^2 \sum_{j=1}^p (n-j)^{-1} \text{tr}\{\hat{\Gamma}'(j)\hat{\Gamma}^{-1}(0)\hat{\Gamma}(j)\hat{\Gamma}^{-1}(0)\}$$

where  $\hat{\Gamma}(\cdot)$  is the ordinary covariance matrix. In contrast to the  $\bar{M}_n$ 's results,  $mLB$  gives large  $p$ -values (0.218, 0.731 and 0.525) respectively. The [portes](#) (Mahdi and McLeod, 2012) package needs to be installed in order to perform tests of independence based on  $mLB$  statistic:

```

> library(portes)
> LjungBox(at, c(6, 12, 22))

```

Assuming that the bivariate log returns follows a VAR model and employing the AIC to choose its best order, we obtain that a VAR(2) model fits the data well. Figure 5 shows the ACF plots (upper panel) and ADCF plots (lower panel) of the residuals after fitting a VAR(2) model to the original bivariate log return series using the function `VAR()` from the [MTS](#) (Tsay, 2015) package. In contrast to the ACF plot, the ADCF plot still indicates some dependence among the residuals. Constructing tests of independence based on  $\bar{T}_n$  and  $mLB$  for the same choices of bandwidth,  $p = 6, 12, 22$ , we confirm this visual result. In particular, employing a Bartlett kernel,  $\bar{T}_n$  statistic gives low  $p$ -values (0.036, 0.018 and 0.034 respectively) whereas the  $mLB$  statistic yields large  $p$ -values (0.669, 0.958 and 0.806 respectively). The calls for the plots of Figure 5 and the corresponding tests of independence are as follows:

```

library(MTS)
model <- VAR(at, 2)
resids <- residuals(model)
colnames(resids) <- c("IBM_res", "SP_res")
windows(9, 6)
acf(resids, lag.max = 18)
mADCFplot(resids, MaxLag = 18, ylim = c(0, 0.13))

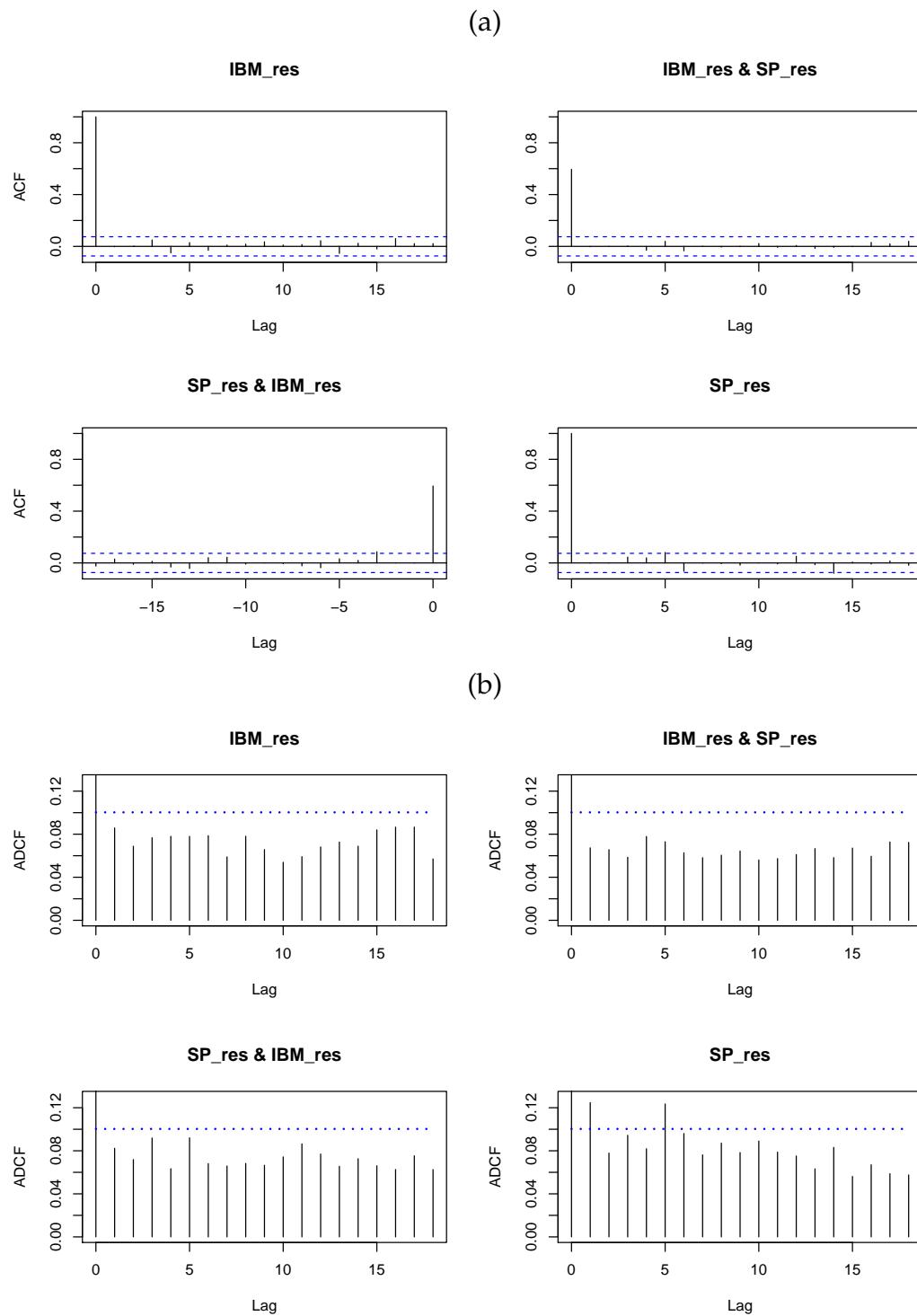
## Tests of independence based on \overline{T}_n
mADCFtest(resids, "bartlett", p = 6, b = 499, parallel = TRUE)

##      Multivariate test of independence based on distance correlation
##
## data: resids, kernel type: bartlett, bandwidth=6, boot replicates 499
## Tnbar = 29.9114, p-value = 0.036

mADCFtest(resids, "bartlett", p = 12, b = 499, parallel = TRUE)

##      Multivariate test of independence based on distance correlation
##
## data: resids, kernel type: bartlett, bandwidth=12, boot replicates 499

```



**Figure 5:** (a) The sample ACF and (b) sample ADCF of the residuals after fitting VAR(2) model to the original series.

```

## Tnbar = 64.7754, p-value = 0.018

mADCFtest(resids, "bartlett", p = 22, b = 499, parallel = TRUE)

##           Multivariate test of independence based on distance correlation
##
## data: resids, kernel type: bartlett, bandwidth=22, boot replicates 499
## Tnbar = 115.3462, p-value = 0.034

## Tests of independence based on mLB
LjungBox(resids, c(6, 12, 22))

```

## Summary and further research

There have been many works in the literature based on Székely et al.'s (2007) distance covariance methodology. The R package **energy** (Rizzo and Szekely, 2014), provides functions that cover this methodology. However, there is no published package that includes functions about distance covariance for time series data. **dCovTS** contributes to filling this gap by providing functions that compute distance covariance and correlation functions for both univariate and multivariate time series. We also include functions that develop univariate and multivariate tests of serial dependence based on distance covariance and correlation functions.

There is a number of possible extensions of this package, and some of them are not covered by existing theory and can be seen as further research. One possible direction is to develop a theory based on partial ADCV or conditional ADCV and a related testing methodology to identify possible dependencies among time series (see Székely and Rizzo (2014) for partial distance covariance methodology and Poczos and Schneider (2012), Wang et al. (2015) for conditional distance covariance methodology; all three works deal with independent random variables). Among the many applications of partial correlation are graphical models. Thus, a graphical modeling theory based on partial ADCV could be carried out and this methodology can be included for a future version of this package.

## Acknowledgments

The authors thank Tobias Liboschik for his considerable help on the development of this package. The authors would also like to thank Dominic Edelmann for carefully checking the package and making helpful comments and suggestions for its improvement. In addition, we would like to extend our gratitude to R. Bivand and to an anonymous reviewer whose comments improved our original submission.

## Bibliography

- R. Analytics and S. Weston. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*, 2015. URL <http://CRAN.R-project.org/package=doParallel>. R package version 1.0.10. [p329]
- G. E. P. Box and D. A. Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American Statistical Association*, 65:1509–1526, 1970. [p332]
- R. A. Davis, M. Matsui, T. Mikosch, and P. Wan. Applications of distance correlation to time series. <http://arxiv.org/abs/1606.05481>, 2016. [p324]
- H. Dehling and T. Mikosch. Random quadratic forms and the bootstrap for U-statistics. *Journal of Multivariate Analysis*, 51:392–413, 1994. [p324, 329]
- J. Dueck, D. Edelmann, T. Gneiting, and D. Richards. The affinely invariant distance correlation. *Bernoulli*, 20:2305–2330, 2014. [p324]
- A. Feuerverger. A consistent test for bivariate dependence. *International Statistical Review*, 61:419–433, 1993. [p324]
- K. Fokianos and M. Pitsillou. Consistent testing for pairwise dependence in time series. *Technometrics*, 2016a. <http://dx.doi.org/10.1080/00401706.2016.1156024>. [p324, 327, 328]

- K. Fokianos and M. Pitsillou. Testing pairwise independence for multivariate time series by the auto-distance correlation matrix. Submitted for publication, 2016b. [p324, 326, 327, 328]
- A. Gretton, K. Fukumizu, and B. K. Sriperumbudur. Discussion of: Brownian distance covariance. *The Annals of Applied Statistics*, 3:1285–1294, 2009. [p324]
- Y. Hong. Consistent testing for serial correlation of unknown form. *Econometrica*, 64:837–864, 1996. [p333]
- Y. Hong. Hypothesis testing in time series via the empirical characteristic function: A generalized spectral density approach. *Journal of the American Statistical Association*, 94:1201–1220, 1999. [p324, 325, 327, 328, 333]
- J. R. M. Hosking. Multivariate Portmanteau statistic. *Journal of the American Statistical Association*, 75: 349–386, 1980. [p336]
- J. Josse and S. Holmes. Tests of independence and beyond. <http://arxiv.org/pdf/1307.7383.pdf>, 2014. [p324]
- A. Leucht and M. H. Neumann. Degenerate U- and V-statistics under ergodicity: asymptotics, bootstrap and applications in statistics. *Annals of the Institute of Statistical Mathematics*, 65:349–386, 2013a. [p329]
- A. Leucht and M. H. Neumann. Dependent wild bootstrap for degenerate U- and V- statistics. *Journal of Multivariate Analysis*, 117:257–280, 2013b. [p324, 329]
- G. M. Ljung and G. E. P. Box. On a measure of lack of fit in time series models. *Biometrika*, 62:297–303, 1978. [p333]
- E. Mahdi and A. I. McLeod. Improved multivariate Portmanteau diagnostic test. *Journal of Time Series Analysis*, 33:211–222, 2012. URL <http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9892.2011.00752.x/abstract>. [p336]
- E. Parzen. On consistent estimates of the spectrum of a stationary time series. *Annals of Mathematical Statistics*, 28:329–348, 1957. [p328]
- B. Poczos and J. Schneider. Conditional distance variance and correlation. <http://www.cs.cmu.edu/~bpoczos/articles/poczos12distancecorr.pdf>, 2012. [p338]
- N. P. Politis, J. P. Romano, and M. Wolf. *Subsampling*. Springer-Verlag, New York, 1999. [p329]
- B. Remillard. Discussion of: Brownian distance covariance. *The Annals of Applied Statistics*, 3:1295–1298, 2009. [p324]
- M. L. Rizzo and G. J. Szekely. *energy: E-statistics (energy statistics)*, 2014. URL <https://CRAN.R-project.org/package=energy>. R package version 1.6.2. [p324, 338]
- X. Shao. The dependent wild bootstrap. *Journal of the American Statistical Association*, 105:218–235, 2010. [p324, 329]
- R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications With R Examples*. Springer-Verlag, New York, 2011. Third Edition. [p330]
- R. H. Shumway, R. S. Azari, and Y. Pawitan. Modeling mortality fluctuations in Los Angeles as functions of pollution and weather effects. *Environmetnal research*, 45:224–241, 1988. [p330]
- G. J. Székely and M. L. Rizzo. Partial distance correlation with methods for dissimilarities. *The Annals of Statistics*, 42:2382–2412, 2014. [p325, 338]
- G. J. Székely, M. L. Rizzo, and N. K. Bakirov. Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35:2769–2794, 2007. [p324, 325, 338]
- R. S. Tsay. *Analysis of Financial Time Series*. Wiley Series in Probability and Statistics, Hoboken, NJ, 2010. Third Edition. [p333]
- R. S. Tsay. *Multivariate Time Series Analysis With R and Financial Applications*. Wiley Series in Probability and Statistics, Hoboken, NJ, 2014. [p333]
- R. S. Tsay. *MTS: All-Purpose Toolkit for Analyzing Multivariate Time Series (MTS) and Estimating Multivariate Volatility Models*, 2015. URL <http://CRAN.R-project.org/package=MTS>. R package version 0.33. [p336]

- X. Wang, W. Pan, W. Hu, Y. Tian, and H. Zhang. Conditional distance correlation. *Journal of the American Statistical Association*, 2015. DOI: 10.1080/01621459.2014.993081. [p338]
- C. F. J. Wu. Jackknife, bootstrap and other resampling methods in regression analysis. *The Annals of Statistics*, 14:1261–1295, 1986. [p329]
- Z. Zhou. Measuring nonlinear dependence in time-series, a distance correlation approach. *Journal of Time Series Analysis*, 33:438–457, 2012. [p324, 325, 327, 329]

*Maria Pitsillou*

*Department of Mathematics & Statistics  
University of Cyprus  
Cyprus  
pitsillou.maria@ucy.ac.cy*

*Konstantinos Fokianos*

*Department of Mathematics & Statistics  
University of Cyprus  
Cyprus  
fokianos@ucy.ac.cy*

# comf: An R Package for Thermal Comfort Studies

by Marcel Schweiker

**Abstract** The field of thermal comfort generated a number of thermal comfort indices. Their code implementation needs to be done by individual researchers. This paper presents the R package, **comf**, which includes functions for common and new thermal comfort indices. Additional functions allow comparisons between the predictive performance of these indices. This paper reviews existing thermal comfort indices and available code implementations. This is followed by the description of the R package and an example how to use the R package for the comparison of different thermal comfort indices on data from a thermal comfort study.

## Introduction

Since the 1960's, researchers in the field of thermal comfort generated a number of thermal comfort indices (see 44.1.1 for details). The three most common indices are the predicted mean vote (PMV) introduced by Fanger (1970), the standard effective temperature (SET) by Gagge et al. (1986), and the adaptive comfort equation as presented e.g. in DIN EN 15251 (2012) and ASHRAE (2013). The latter is based on the work of Auliciems (1981a), de Dear et al. (1997), Nicol and Humphreys (2002) and others.

The purpose of these indices is the prediction of a) thermally acceptable indoor conditions or b) the evaluation of indoor conditions by a group of persons. The calculation procedures and/or equations for the indices are described in the literature they are introduced. However, in most cases the code implementation needs to be done by each researcher individually. Such process is a source for errors; chances are high that the codes of two researchers do not lead to the same outcome given identical input parameters. Therefore, I introduce the R package, **comf**, that enables the calculation of the most common thermal comfort indices and several new indices. The objective is to create a publicly available reference for comparisons and benchmarking.

Additional functions of this package allow a comparison between the outcome of those indices compared to subjective evaluations obtained by a given sample of occupants in a building. Such evaluation of the indices performance is seldom found in thermal comfort research. Given the increasing number of indices, such comparison needs to be done more often in order to judge under which circumstances which index performs best (Schweiker and Wagner, 2015, 2016).

In this paper, existing thermal comfort indices are reviewed together with available tools and code implementations for their calculation. This is followed by the introduction of the R package, **comf**, and an example application to an existing dataset.

## Review of thermal comfort indices

Table 1 gives an overview of the indices dealt with in this article or included in the R package, **comf**. Thereby, this list is not exclusive given the high number of additional comfort indices.

The indices can be grouped according to their outcome in those predicting

- a mean vote on the 7-point thermal sensation scale,
- a neutral or comfortable temperature, or
- other values related to the perception of the thermal indoor environment.

The 7-point thermal sensation scale is the standard scale for the assessment of thermal perception given to subjects and is coded -3 cold, -2 cool, -1 slightly cool, 0 neither cold nor warm, +1 slightly warm, +2 warm, +3 hot (ISO 7726, 1998). Traditionally, this scale was used as categorical scale, but recent studies are using either a categorical or a continuous version (Schweiker et al., 2016a).

The neutral temperature is a set of operative temperatures evaluated in average as neutral (neither cold nor warm) on the 7-point thermal sensation scale (Auliciems, 1981b; Humphreys, 1978). Sometimes, this is also referred to as comfortable temperature (Schweiker et al., 2016a). Additional members of this group of indices are the adaptive comfort temperatures (Brager and de Dear, 2001; Nicol and Humphreys, 2010)

The other values are e.g. related to the thermal strain of an individual due to the indoor thermal environment (Gagge et al., 1986), the exergy consumption rate of the human body (Shukuya, 2009), or

Index	Input variables <sup>1</sup>	Output	Reference
PMV	ta, tr, rh, vel, clo, met	Predicted mean vote (−3 to +3)	(Fanger, 1970)
PMV <sub>adj</sub>	ta, tr, rh, vel, clo, met	Predicted mean vote (−3 to +3)	(ASHRAE, 2013; Schiavon et al., 2014)
ePMV	ta, tr, rh, vel, clo, met, e or asv	Predicted mean vote (−3 to +3)	(Fanger and Tof-tum, 2002)
aPMV	ta, tr, rh, vel, clo, met, λ or asv	Predicted mean vote (−3 to +3)	(Yao et al., 2009)
ATHB <sub>pmv</sub>	ta, tr, rh, vel, met, trm, psych	Predicted mean vote (−3 to +3)	(Schweiker and Wagner, 2015)
PTS	ta, tr, rh, vel, clo, met	Predicted thermal sensation (−3 to +3)	(McIntyre, 1980)
PTSe	ta, tr, rh, vel, clo, met, e or asv	Predicted thermal sensation (−3 to +3)	(Gao et al., 2015)
PTSa	ta, tr, rh, vel, clo, met, λ or asv	Predicted thermal sensation (−3 to +3)	(Gao et al., 2015)
ATHB <sub>pts</sub>	ta, tr, rh, vel, met, trm, psych	Predicted thermal sensation (−3 to +3)	(Schweiker and Wagner, 2016)
tAdapt15251	trm	Adaptive comfort temperature	(DIN EN 15251, 2012; Nicol and Humphreys, 2010)
tAdaptASHRAE	tmmo	Adaptive comfort temperature	(Brager and de Dear, 2001)
tnAuliciems	ta, tmmo	Neutral temperature	(Auliciems, 1981b)
tnHumphreysNV	tmmo	Neutral temperature in natural-ventilated buildings	(Humphreys, 1978)
tnHumphreysAC	tmmo	Neutral temperature in climate-controlled buildings	(Humphreys, 1978)
PPD	ta, tr, rh, vel, clo, met	Predicted percentage dissatisfied (0 to 100)	(Fanger, 1970)
SET	ta, tr, rh, vel, clo, met	Standard effective temperature	(Gagge et al., 1986)
dTNZ	ta, vel, clo, met	Distance to thermoneutral zone	(Kingma et al., 2016)
Ex	ta, tr, rh, vel, clo, met, tao, rho	Human body exergy consumption rate	(Shukuya, 2009)

<sup>1</sup>ta = air temperature; tr = radiant temperature; rh = relative humidity; vel = air velocity; clo = clothing insulation level; met = metabolic rate; tao = outdoor air temperature; rho = outdoor relative humidity; trm = running mean outdoor temperature; tmmo = monthly mean outdoor temperature; e = expectancy factor; λ = adaptive coefficient; psych = factor related to psychological adaptation; asv = actual sensation vote

**Table 1:** Thermal comfort indices included in the R package, **comf**, their input variables, output description, values, and references

the distance of observed operative temperature or mean skin temperature to the thermoneutral zone (Kingma et al., 2016) and will be explained below.

The first group of indices predicting a mean vote on the thermal sensation scale consists of the PMV-index and its alterations (s. below) together with the predicted thermal sensation (PTS) based on the SET-index and corresponding adjusted versions.

The PMV index is based on the assumption that comfortable conditions are perceived when there is a balance between the heat generated by the metabolism and the heat lost or gained through convection, radiation, and evaporation (Fanger, 1970).

Alterations to the PMV-index are

- the adjusted PMV ( $PMV_{adj}$ ), which modifies the PMV model for elevated air velocities (ASHRAE, 2013; Schiavon et al., 2014),
- the ePMV, which uses the expectancy factor,  $e$ , to account for variations in the expectation of people (Fanger and Toftum, 2002),
- the aPMV, which alters the PMV based on an adaptive coefficient,  $\lambda$ , which represents the sum of behavioural, physiological, and psychological adaptation (Yao et al., 2009), and
- the ATHB<sub>pmv</sub>, which adjusts the input values for clothing level and metabolic rate based on individual equations for the three just mentioned adaptive processes (Schweiker and Wagner, 2015).

In order to calculate the PTS it is necessary to calculate the SET first (s. below). Then, PTS can be calculated through the equation (McIntyre, 1980):

$$PTS = .25 \cdot SET - 6.03. \quad (1)$$

Adjusted versions of the PTS are parallel to the alterations to PMV,

- the PTSe using the expectancy factor (Gao et al., 2015),
- the PTSa using the adaptive coefficient (Gao et al., 2015), and
- the ATHB<sub>pts</sub> changing the input values of clothing level and metabolic rate for the calculation of SET (Schweiker and Wagner, 2016).

The second group of indices consists of the adaptive comfort equations given e.g. in DIN EN 15251 (2012) and Brager and de Dear (2001) as well as the equations for the neutral temperatures by Auliciems (1981b) and Humphreys (1978). Both types of equations calculate the indoor environmental temperature to be evaluated as neutral on the 7-point thermal sensation scale or as comfortable.

The third group consists of the predicted percentage of dissatisfied (PPD), the SET, the distance to the thermoneutral zone (dTNZ), and the exergy consumption rate (Ex).

The predicted percentage of dissatisfied (PPD) is calculated based on the PMV value as described in Fanger (1970) by

$$PPD = 100 - 95e^{[-(.3353 \cdot PMV^4 + .2179 \cdot PMV^2)]}. \quad (2)$$

The SET is "the temperature of an imaginary environment at 50% relative humidity, <0.1 m/s average air speed, and mean radiant temperature equal to average air temperature, in which total heat loss from the skin of an imaginary occupant with an activity level of 1.0 met and a clothing level of 0.6 clo is the same as that from a person in the actual environment, with actual clothing and activity level" (ASHRAE, 2013) and is based on the work by Gagge and his group (Gagge et al., 1986).

The dTNZ was introduced by Kingma et al. (2016) and presents a biophysical approach to predict thermal sensation. Similar to the ATHB, the dTNZ is a new concept and still needs to be further evaluated. The same is true for the concept of Ex. A lower Ex was shown to be related to conditions regarded as thermally comfortable. Schweiker et al. (2016b); Simone et al. (2011) demonstrated that there is a relationship between Ex, thermal sensation, and thermal acceptance.

## Existing software and tools

Only few of the thermal comfort indices can be calculated with existing tools. Table 2 gives an overview of existing software, applications, and code implementations. In addition, several building energy performance simulation programs, e.g. Energy+, do offer the option to calculate the PMV value or other value.

Name	Type	Comfort indices	Link/Source
ASHRAE Thermal Comfort Tool	Software	PMV, PMV <sub>adj</sub> , PPD, SET, Tadapt	<a href="https://www.ashrae.org/resources--publications/bookstore/thermal-comfort-tool">https://www.ashrae.org/resources--publications/bookstore/thermal-comfort-tool</a>
CBE comfort tool	Web application	PMV/PMV <sub>adj</sub> , PPD, SET, Tadapt	<a href="http://comfort.cbe.berkeley.edu/">http://comfort.cbe.berkeley.edu/</a> <sup>1</sup> , Fountain and Huizenga (1995); Schiavon et al. (2014)
USYD homepage	Web application	PMV, PPD, SET, ET, + <sup>2</sup>	<a href="http://web.arch.usyd.edu.au/~rdedear/">http://web.arch.usyd.edu.au/~rdedear/</a>
ISO 7730	Code snippets in BASIC	PMV, PPD	ISO 7730 (2005)
Gagge et al.	Code snippets in FORTRAN	SET, ET, + <sup>2</sup>	Gagge et al. (1986)
ASHRAE PMV	Code snippets in BASIC	PMV, PPD	ASHRAE (2013)
ASHRAE SET <sup>3</sup>	Code snippets in Java	SET	ASHRAE (2013)
Schweiker et al.	Code snippets in R	HbExUnSt	Schweiker et al. (2016b)
Shukuya	Excel sheet	HbExUnSt	Shukuya

<sup>1</sup>The source code is available at [https://github.com/CenterForTheBuiltEnvironment/comfort\\_tool](https://github.com/CenterForTheBuiltEnvironment/comfort_tool)

<sup>2</sup>+ = and other indices

<sup>3</sup>A version of SET fit for adjusting PMV to higher air velocities

**Table 2:** Existing applications, software, and code snippets for the calculation of thermal comfort indices

Notable exceptions are the calculations of the most common indices: a BASIC code is given in ISO 7730 (2005) for the calculation of PMV and PPD and a FORTRAN code for the calculation of SET was presented in Gagge et al. (1986). Recently a JavaScript-version for SET calculation was included in ASHRAE (2013). However, this version does not use the full code for calculation of SET by Gagge et al. (1986) or Fountain and Huizenga (1995), but a modified version. The difference is that for the SET-code used in ASHRAE (2013), the part related to convection from metabolically-generated air movement has been removed. This was done in order to have a smooth transition from original PMV values up to .15 m/s of air velocity to the adjusted PMV values starting above this air velocity.

Another source for code implementations is the source code of the CBE comfort tool, which is available at [https://github.com/CenterForTheBuiltEnvironment/comfort\\_tool](https://github.com/CenterForTheBuiltEnvironment/comfort_tool). This includes code in JavaScript and Python for the calculation of PMV, PMV<sub>adj</sub>, the adaptive comfort temperature and range, and the modified SET calculation as described in ASHRAE (2013).

## Introduction to the package comf

The idea behind the R package, **comf** is to support researchers in the field of thermal comfort not only through publicly available code implementations for the calculation of comfort indices in R, but also through additional functions. Therefore, the main functions of this package can be grouped into those related to

- the preparation of a dataset and transformation of physical variables,
- the calculation of one or more comfort indices (see Table 1), and
- the evaluation of the performance of a comfort index.

## Preparation of a dataset and conversion of physical variables

Each thermal comfort index requires different input parameters. Therefore, the R package, **comf**, offers two procedures in order to prepare a dataset to be used as input to the calculation of one or more thermal comfort indices.

The first procedure starts with calling the function `createCond`. This function creates a list with standard values for the variables required for all comfort indices included in this package. Own data or further adjustments to these values could be done as follows:

```
install.packages("comf_0.1.6.tar.gz", repos=NULL, type="source")
library(comf)

lsCond <- createCond()
lsCond$ta <- 21:30
lsCond$rh <- 51:60
lsCond$met <- 1.0
```

It is important that the length of vectors assigned to the elements of this list are either 1 or do have the same length. In above example, it is not possible to assign a vector with 11 items to `ta`, the indoor air temperature, and a vector with 10 items to `rh`, the relative humidity indoors.

The second procedure starts with a dataframe containing all variables to be used for the calculation. This procedure requires the user to know the required variables. This dataframe can then be transferred into a list or used directly.

```
ta <- 21:30
rh <- 51:60
met <- 1.0
dfCond <- data.frame(ta, rh, met)
lsCond2 <- as.list(dfCond)
```

In addition, **comf** offers a variety of small functions to convert variables from one type to another. This includes among others

- `calcDewp`, which calculates the dew point temperature, given air temperature and relative humidity,
- `calcEnth`, which calculates the enthalpy of the air, given air temperature, relative humidity, and barometric pressure,
- `calcRH`, which calculates the relative humidity of air, given air temperature, mixing ratio, and barometric pressure,
- `calcTroin`, which calculates the operative and radiant temperature for standard globe measurements according to ISO 7726 (1998), given air temperature, globe temperature, air velocity, and metabolic rate.

## Calculating one or more comfort indices

Before the preparation of a dataset, it is important to know that the structure of the input to functions for the calculation of one specific index such as `calcATHB` differs to that of the main function, `calcComfInd`. The latter requires a list or data frame with variables as described below, while the former works with vectors or data frames.

There are again two possibilities to calculate one or more comfort indices.

The first one uses the main function of this package, `calcComfInd`. This function requires a list or data frame of variables together with a vector of comfort indices to be calculated, e.g. `request="all"` to calculate all indices or `request=c("ATHBpmv", "pmv")` to calculate these two. The list of variables can consist of one item per variable or several items per variable, i.e. one value for each input parameter, or for some parameters 234 values and for the others one parameter. The rationale behind this is that very often, variables such as age, gender, or metabolic rate do not differ in a given dataset, while others like the indoor air temperature are different for each case. A complete list of indices to be calculated can be found in the help file of `calcComfInd` or obtained calling `listOfRequests()`.

The function `calcComfInd` checks whether there is only one or more values for each variable and whether all variables required for the thermal comfort index to be calculated exist in the list. In case one or more required variables do not exist, the index is calculated using pre-defined standard values for these variables. In such case a warning is given at the end of the calculation in order to inform the user about the missing variable(s) and the value(s) used for the calculation.

```
# using lsCond from above does not produce a warning
calcComfInd(lsCond, request="all")

# using lsCond2 from above displays 31 warnings which report
# the corresponding standard values used
calcComfInd(lsCond2, request="all")
warnings()

# the results however are identical
```

Individual functions, e.g. `calcSET` to calculate SET, can be used for the second possibility to calculate one or more comfort indices. With this procedure, one can use a data frame or a list of vectors to calculate a specific thermal comfort index. The list of required variables as well as information about the standard values used when a variable is missing is included in each helpfile, e.g. `?calcSET`.

The following example illustrates the usage for multiple input lines:

```
ta <- c(20,22,24)
tr <- ta
vel <- rep(0.15,3)
rh <- rep(50,3)

maxLength <- max(sapply(list(ta, tr, vel, rh), length))
SET <- sapply(seq(maxLength),
  function(x) { calcSET(ta[x], tr[x], vel[x], rh[x]) } )
```

### Evaluating the performance of one or more comfort indices

Due to the number of new or adjusted indices being presented in the scientific literature, the comparison between the performance of them will be an important aspect in future studies. The R package, `comf`, includes functions for different performance criteria.

The function `calcBias` calculates the mean bias, its standard deviation, and standard error between the actual (observed) thermal sensation vote (ASV) and the predicted thermal sensation vote (PSV) ([Humphreys and Nicol, 2002](#)). This is calculated according to

$$\text{mean bias} = \text{mean}(PSV_i - ASV_i), \quad (3)$$

where  $i$  denotes the individual vote.

The true positive rate (TPR) is the proportion of true predicted cases, where the categorical ASV is equal to the categorical PSV ([Schweiker and Wagner, 2015](#)). This can be calculated using the function `calcTPRTSV`, which calculates

$$TPR = \frac{1}{n} \sum_{i=1}^k tp_k, \quad (4)$$

where  $k$  denotes the category of the sensation scale (e.g. *cold*),  $n$  the total number of votes, and  $tp$  the true positive cases, where the categorical PSV is equal to the categorical ASV.

The function `calcAvgAcc` calculates the average accuracy between PSV and ASV according to [Sokolova and Lapalme \(2009\)](#) by

$$\text{average accuracy} = \frac{1}{l} \sum_{i=1}^l \frac{tp_i + tn_i}{tp_i + fn_i + fp_i + tn_i}, \quad (5)$$

where  $l$  denotes the number of categories of the sensation scale,  $tp$ ,  $tn$ ,  $fn$ , and  $fp$  the number of true positives, true negatives, false positives, and false negatives for the corresponding class. Note that the value of the average accuracy depends strongly on the distribution of ASV, i.e. in case most of the ASV's are in the same category, e.g. *neutral*, the average accuracy is very high due to the fact that for all other categories the number of true negative predicted votes is high as well.

Column	Variable name	Unit	Derivation
ta	Air temperature	°C	Measured
tr	Radiant temperature	°C	Assumed to be equal to ta
rh	Relative humidity	%	Measured
trm	Running mean outdoor temperature	°C	Calculated from tout using the equation from <a href="#">DIN EN 15251 (2012)</a>
clo	Clothing insulation level	CLO	Assessed during visit
tout	Outdoor air temperature	°C	Measured
vel	Air velocity	m/s	Assumed based on state of window(s) and door
met	Metabolic rate	MET	Assumed based on <a href="#">ISO 7730 (2005)</a>
asv	Actual sensation vote	—	Obtained through questionnaire

**Table 3:** Variables included in the dataset together with their derivation

## An example using data from a field experiment

The R package, **comf**, includes a data set deriving from a field experiment. This field measurement is described in detail in [Hawighorst et al. \(2016\)](#) and [Schweiker and Wagner \(2016\)](#). The data set included in this package contains 156 samples, which is a subset of the original data set with 620 samples, and was drawn with the R function `sample`.

The original data set was obtained by two field experiments in six office buildings in southern Germany. They were conducted during the summer periods of 2011 and 2012. Data loggers for air temperature and relative humidity were placed in the offices. In addition outdoor air temperature and relative humidity were measured with another data logger on the roof of each building.

Subjects were visited up to 4 times during a two week period. The number of votes obtained by each subject differs due to absence periods of subjects. During each visit, subjects were asked about their thermal sensation (7-point categorical scale with the categories -3 cold, -2 cool, -1 slightly cool, 0 neutral, +1 slightly warm, +2 warm, +3 hot) together with a set of additional questions not relevant for this paper. While the subjects answered the paper-pencil based questionnaire the investigator noted down the clothing level of each subject. Written informed consent was obtained from the subjects prior to the installment of the data loggers. The air velocity included in the data set was estimated based on the state of window(s), door, and table fan and detailed measurements in a single room.

The variables included in the dataset are presented in Table 3. Descriptive statistics of the data set included in the package can be explored using:

```
library(comf)
library(psych)
data(dfField)
describe(dfField)
```

In order to calculate a number of comfort indices for the conditions present in the data, it is recommended to start with the list of standard values and assign the values of the data set to the corresponding items of the list by:

```
# creating a list with standard values
lsField <- createCond()

# assigning the variables included in the data set to the list
variables <- c("ta", "tr", "vel", "rh", "clo", "met", "trm", "asv", "tao")
for(i in 1:length(variables)) {
  lsField[[variables[i]]] <- dfField[[variables[i]]]
}
```

For this example, the following 8 thermal comfort indices will be calculated and compared: PMV, PMVadj, ATHBpmv, aPMV, ePMV, PTS, PTSa, PTSe, and ATHBpts. In order to be able to calculate aPMV, ePMV, PTSa, and PTSe, one needs to get an estimate for the adaptive coefficient and expectancy factor. This is done using the corresponding functions of the package by:

```
lsField$epCoeff <- calcepCoeff(lsField)$epCoeff
lsField$apCoeff <- calcapCoeff(lsField)$apCoeff
```

```
lsField$esCoeff <- calcesCoeff(lsField)$esCoeff
lsField$asCoeff <- calcasCoeff(lsField)$asCoeff
```

Then, the thermal comfort indices are calculated at once using the function `calcComfInd`. Note that it would be also possible to calculate all eight indices individually by calling their function as described above.

```
indices <- c('pmv', 'pmvadj', 'apmv', 'epmv', 'ATHBpmv',
'pts', 'ptsa', 'ptse', 'ATHBpts')
results <- calcComfInd(lsField, request = indices)
```

For the comparison between predicted thermal sensation votes and actual thermal sensation votes, the predicted continuous sensation votes need to be converted into categorical ones. This is necessary, because the actual sensation vote included in the dataset was obtained using a categorical scale. This can be done using the function `cutTSV`, which converts continuous thermal sensation votes to categorical ones. The conversion is done using intervals closed on the right, e.g. setting all values higher than  $-2.5$  and lower or equal  $-1.5$  to the value of  $-2$ .

```
asv.cat <- cutTSV(dfField$asv)
results.cat <- lapply(seq(length(indices)), function(i) {cutTSV(results[,i])})
names(results.cat) <- indices
```

With the binned predicted values of thermal sensation votes, the mean bias, its standard error, and the true positive rate (TPR) can be calculated for each thermal comfort index individually:

```
# calculating mean value of bias between predicted and actual sensation vote
# for each comfort index
meanBias <- sapply(indices, function(i) {
  calcBias(asv.cat, results.cat[[i]])$meanBias
})

# calculating standard error of bias between predicted and actual sensation vote
# for each comfort index
seBias <- sapply(indices, function(i) {
  calcBias(asv.cat, results.cat[[i]])$seBias
})

# calculating the true positive rate for each comfort index
TPR <- sapply(indices, function(i) {
  calcTPRTSV(asv.cat, results.cat[[i]])
})
```

The comparison of bias and true positive rate can be done e.g. graphically using the R package, `ggplot2`:

```
library(ggplot2)
library(plyr)
library(reshape2)

group <- c("PMV", "PMVadj", "aPMV", "ePMV", "ATHB pmv", "PTS", "PTSa",
"PTSe", "ATHB pts")
lower <- meanBias - seBias
upper <- meanBias + seBias

fig4Win <- data.frame(meanBias, TPR, group, lower, upper)
fig4Win$variable <- rep(2,9)
fig4Win$group <- factor(fig4Win$group, levels = fig4Win$group)

addline_format <- function(x,...){
```

```

    gsub('\\s','\n',x)
}

means.barplot <- qplot(x=group, y=meanBias, data=fig4Win, geom="point",
  stat="identity", position="dodge", ymax=.5) +
  scale_x_discrete(breaks=unique(fig4Win$group),
  labels=addline_format(c("PMV", "PMVadj", "aPMV", "ePMV",
  "ATHB pmv", "PTS", "PTSa", "PTSe", "ATHB pts")))

means.barplot + geom_errorbar(aes(ymax=upper,ymin=lower),
  position=position_dodge(0.9), data=fig4Win) +
  theme_bw() +
  xlab("Comfort indices") +
  ylab("bias PSV - ASV") +
  ylim(c(-1,.5))

## uncomment next line to save file to current working directory
#ggsave("Fig1_MeanBias.png")

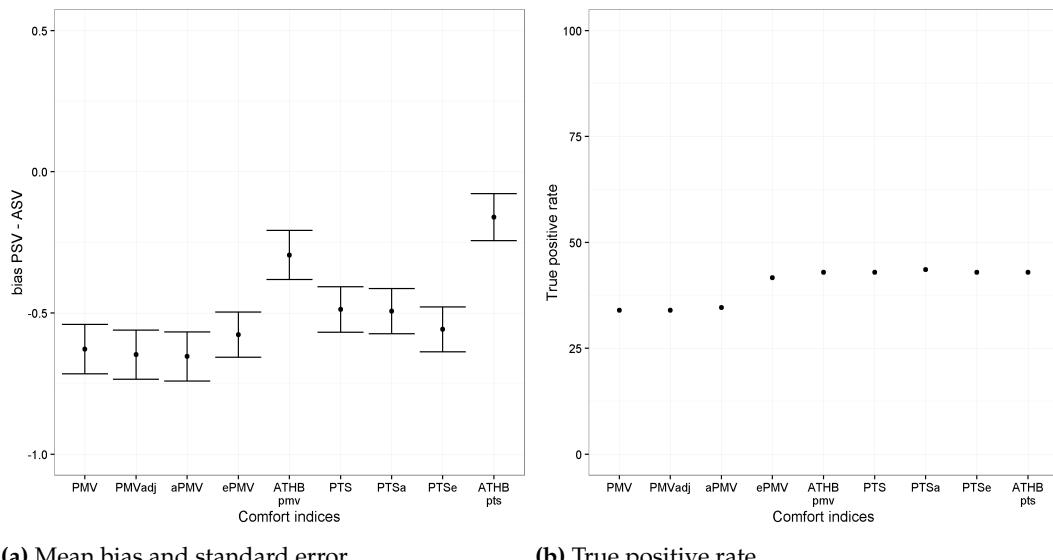
means.barplot <- qplot(x=group, y=TPR*100, data=fig4Win, geom="point",
  stat="identity", position="dodge", ymax=100) +
  scale_x_discrete(breaks=unique(fig4Win$group),
  labels=addline_format(c("PMV", "PMVadj", "aPMV", "ePMV", "ATHB pmv",
  "PTS", "PTSa", "PTSe", "ATHB pts")))

means.barplot +
  theme_bw() +
  xlab("Comfort indices") +
  ylab("True positive rate") +
  ylim(c(0,100))

## uncomment next line to save file to current working directory
#ggsave("Fig1_TPR.png")

```

The result can be seen in Figure 1. This shows, that for this particular data set, the indices ATHBpmv and ATHBpts have the lowest mean bias between predicted and actual sensation votes. Related to the true positive rate, there are five indices with a similar performance of around 42% of truly predicted sensation votes, while the true positive rate of the other three indices is around 34%.



(a) Mean bias and standard error.

(b) True positive rate.

**Figure 1:** True positive rate for eight comfort indices.

## Summary

This article has described the R package `comf`. This package implements several functions to assist researchers in the field of thermal comfort. The main functions calculate various common and less common thermal comfort indices. Additional functions are related to the preparation of a suitable data set and to the comparison of observed and predicted assessment.

## Acknowledgements

The author would like to thank Sophia Mueller for her outstanding work during the preparation of this package. The author would like to thank Masanori Shukuya for the permission to include the code implementation based on the VBA code for the calculation of human body exergy consumption rate. In addition, the author would like to thank those who contributed to parts of the code or checked one or more of the functions, namely Michael Kleber and Boris Kingma.

## Bibliography

- ASHRAE. Standard 55-2013. Thermal environmental conditions for human occupancy. *American Society of Heating, Refrigerating and Air-Conditioning Engineering*, Atlanta, USA, 2013. [p<sup>341, 342, 343, 344</sup>]
- A. Auliciems. Towards a psychophysiological model of thermal perception. *International Journal of Biometeorology*, 25:109–122, 1981a. [p<sup>341</sup>]
- A. Auliciems. Psycho-physiological criteria for global thermal zones of building design. *Int J Biometeorol*, 26:69–86, 1981b. [p<sup>341, 342, 343</sup>]
- G. S. Brager and R. de Dear. Climate, comfort, & natural ventilation: a new adaptive comfort standard for ASHRAE standard 55. *Center for the Built Environment*. UC Berkeley, 2001. [p<sup>341, 342, 343</sup>]
- R. de Dear, G. S. Brager, and D. Cooper. Developing an adaptive model of thermal comfort and preference. In *Final Report on ASHRAE Research Project 884*. Macquarie University Sydney, 1997. [p<sup>341</sup>]
- DIN EN 15251. Indoor environmental input parameters for design and assessment of energy performance of buildings addressing indoor air quality, thermal environment, lighting and acoustics; German version EN 15251:2012, August 2012. DIN EN 15251. [p<sup>341, 342, 343, 347</sup>]
- P. O. Fanger. *Thermal Comfort Analysis and Applications in Environmental Engineering*. McGraw-Hill, New York, 1970. [p<sup>341, 342, 343</sup>]
- P. O. Fanger and J. Toftum. Extension of the PMV model to non-air-conditioned buildings in warm climates. *Energy and Buildings*, 34:533–536, 2002. [p<sup>342, 343</sup>]
- M. Fountain and C. Huijzena. A thermal sensation model for use by the engineering profession. Technical report, ASHRAE RP-781 Final report, 1995. [p<sup>344</sup>]
- A. P. Gagge, A. P. Fobelets, and L. G. Berglund. A standard predictive index of human response to the thermal environment. *ASHRAE Transactions*, 92 (2B):709–731, 1986. [p<sup>341, 342, 343, 344</sup>]
- J. Gao, Y. Wang, and P. Wargocki. Comparative analysis of modified pmv models and set models to predict human thermal sensation in naturally ventilated buildings. *Building and Environment*, 92: 200–208, 2015. [p<sup>342, 343</sup>]
- M. Hawighorst, M. Schweiker, and A. Wagner. Thermo-specific self-efficacy (specse) in relation to perceived comfort and control. *Building and Environment*, 102:193 – 206, 2016. ISSN 0360-1323. doi: <http://dx.doi.org/10.1016/j.buildenv.2016.03.014>. URL <http://www.sciencedirect.com/science/article/pii/S0360132316300919>. [p<sup>347</sup>]
- M. A. Humphreys. Outdoor temperatures and comfort indoors. *Batiment International, Building Research and Practice*, 6(2):92–92, 1978. [p<sup>341, 342, 343</sup>]
- M. A. Humphreys and J. F. Nicol. The validity of iso-pmv for predicting comfort votes in every-day thermal environments. *Energy and buildings*, 34(6):667–684, 2002. [p<sup>346</sup>]
- ISO 7726. Ergonomics of the Thermal Environment, Instruments for Measuring Physical Quantities. Geneva: International Standard Organization, 1998. [p<sup>341, 345</sup>]

- ISO 7730. Ergonomics of the thermal environment. Analytical determination and interpretation of thermal comfort using calculation of the PMV and PPD indices and local thermal comfort criteria. Geneva: International Standard Organization, 2005. [p344, 347]
- B. R. Kingma, M. Schweiker, A. Wagner, and W. D. van Marken Lichtenbelt. Exploring the potential of a biophysical model to understand thermal sensation. In *Proceedings of 9th Windsor Conference: Making comfort relevant Cumberland Lodge, Windsor, UK*, 2016. [p342, 343]
- D. A. McIntyre. *Indoor climate*. Applied Science Publisher, London, 1980. [p342, 343]
- J. F. Nicol and M. A. Humphreys. Adaptive thermal comfort and sustainable thermal standards for buildings. *Energy and buildings*, 34(6):563–572, 2002. [p341]
- J. F. Nicol and M. A. Humphreys. Derivation of the adaptive equations for thermal comfort in free-running buildings in European standard {EN15251}. *Building and Environment*, 45(1):11 – 17, 2010. ISSN 0360-1323. doi: <http://dx.doi.org/10.1016/j.buildenv.2008.12.013>. URL <http://www.sciencedirect.com/science/article/pii/S036013230800303X>. International Symposium on the Interaction between Human and Building Environment Special Issue Section. [p341, 342]
- S. Schiavon, T. Hoyt, and A. Piccioli. Web application for thermal comfort visualization and calculation according to ASHRAE standard 55. *Building Simulation*, 7(4):312–334, 2014. doi: <http://dx.doi.org/10.1007/s12273-013-0162-3>. URL <http://escholarship.org/uc/item/4db4q37h>. [p342, 343, 344]
- M. Schweiker and A. Wagner. A framework for an adaptive thermal heat balance model (athb). *Building and Environment*, 94:252 – 262, 2015. ISSN 0360-1323. doi: <http://dx.doi.org/10.1016/j.buildenv.2015.08.018>. URL <http://www.sciencedirect.com/science/article/pii/S0360132315300998>. [p341, 342, 343, 346]
- M. Schweiker and A. Wagner. Exploring potentials and limitations of the adaptive thermal heat balance framework. In *Proceedings of 9th Windsor Conference: Making comfort relevant Cumberland Lodge, Windsor, UK*, 2016. [p341, 342, 343, 347]
- M. Schweiker, X. Fuchs, S. Becker, M. Shukuya, M. Dovjak, M. Hawighorst, and J. Kolarik. Challenging the assumptions for thermal sensation scales. *Building Research & Information*, 2016a. doi: 10.1080/09613218.2016.1183185. [p341]
- M. Schweiker, J. Kolarik, M. Dovjak, and M. Shukuya. Unsteady-state human-body exergy consumption rate and its relation to subjective assessment of dynamic thermal environments. *Energy and Buildings*, 116:164 – 180, 2016b. ISSN 0378-7788. doi: <http://dx.doi.org/10.1016/j.enbuild.2016.01.002>. URL <http://www.sciencedirect.com/science/article/pii/S0378778816300020>. [p343, 344]
- M. Shukuya. Exergy concept and its application to the built environment. *Building and Environment*, 44(7):1545 – 1550, 2009. ISSN 0360-1323. doi: <http://dx.doi.org/10.1016/j.buildenv.2008.06.019>. URL <http://www.sciencedirect.com/science/article/pii/S0360132308001637>. The 6th International Conference on Indoor Air Quality, Ventilation & Energy Conservation in Buildings (IAQVEC 2007), Sendai, Japan, 28-31 October, 2007. [p341, 342]
- A. Simone, J. Kolarik, T. Iwamatsu, H. Asada, M. Dovjak, L. Schellen, M. Shukuya, and B. W. Olesen. A relation between calculated human body exergy consumption rate and subjectively assessed thermal sensation. *Energy and Buildings*, 43(1):1–9, 2011. ISSN 0378-7788. doi: DOI:10.1016/j.enbuild.2010.08.007. [p343]
- M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009. [p346]
- R. Yao, B. Li, and J. Liu. A theoretical adaptive model of thermal comfort - Adaptive Predicted Mean Vote (aPMV). *Building and Environment*, 44:2089–2096, 2009. [p342, 343]

Marcel Schweiker  
Karlsruhe Institute of Technology, Building Science Group  
Englerstr. 7, 76131 Karlsruhe  
Germany  
[marcel.schweiker@kit.edu](mailto:marcel.schweiker@kit.edu)

# water: Tools and Functions to Estimate Actual Evapotranspiration Using Land Surface Energy Balance Models in R

by Guillermo Federico Olmedo, Samuel Ortega-Farías, Daniel de la Fuente-Sáiz, David Fonseca-Luego and Fernando Fuentes-Peñailllo

**Abstract** The crop water requirement is a key factor in the agricultural process. It is usually estimated throughout actual evapotranspiration ( $ET_a$ ). This parameter is the key to develop irrigation strategies, to improve water use efficiency and to understand hydrological, climatic, and ecosystem processes. Currently, it is calculated with classical methods, which are difficult to extrapolate, or with land surface energy balance models (LSEB), such as METRIC and SEBAL, which are based on remote sensing data. This paper describes **water**, an open implementation of LSEB. The package provides several functions to estimate the parameters of the LSEB equation from satellite data and proposes a new object class to handle weather station data. One of the critical steps in METRIC is the selection of “cold” and “hot” pixels, which **water** solves with an automatic method. The **water** package can process a batch of satellite images and integrates most of the already published sub-models for METRIC. Although **water** implements METRIC, it will be expandable to SEBAL and others in the near future. Finally, two different procedures are demonstrated using data that is included in **water** package.

## Introduction and motivation

The crop water requirement is a key factor in the agricultural process. It is usually estimated throughout actual evapotranspiration ( $ET_a$ ). An accurate quantification of  $ET_a$  helps to develop irrigation strategies, improve the efficiency of water use and increase the irrigated area and the production (Millar, 1993; Baruch and Fisher, 1991; Ferreyra et al., 1985).

Traditional methods to estimate  $ET_a$  are based on (a) direct measurements by sophisticated instruments, such as lysimeters (Payero and Irmak, 2008; López-Urrea et al., 2009), Eddy covariance systems (Paço et al., 2006; Parent and Anctil, 2012; Poblete-Echeverría and Ortega-Farias, 2013) or Bowen ratios (Cragoa and Brutsaert, 1996; Ortega-Farías et al., 1995; Twine et al., 2000), or on (b) empirical methods, such as the FAO-56 approach (Allen et al., 1998). This method uses a reference evapotranspiration ( $ET_r$ ) from an automatic weather station multiplied by crop coefficients ( $K_c$ ) from literature (Allen et al., 2005). Although all these methods can be accurate enough, they are restrictive to be extrapolated to a farm or a regional level since they do not take into account the effect that the spatial and temporal variation of the soil, the climate and the crop have over the  $ET_a$  (Allen et al., 2011).

However, new physical methods to estimate  $ET_a$  have been developed using remote sensing data, considering the land spatial and temporal patterns. A major restriction for the estimation of  $ET_a$  using remote sensing is the need of an absolute surface temperature for calibration. One of the first methods to be developed and applied worldwide was the Surface Energy Balance Algorithms for Land (SEBAL) model (Bastiaanssen et al., 1998a,b). This model calculates  $ET_a$  from satellite-based land surface energy balance (LSEB) equation and uses a near-surface temperature gradient ( $dT$ ) for calibration.  $dT$  is computed by taking two pixels with extreme water condition (anchor pixels) selected in the scene to generate a linear relationship between surface temperature and the difference between surface and air temperatures. Based on this, Allen et al. (2007b) developed the *Mapping EvapoTranspiration at High Resolution with Internalized Calibration* (METRIC) model. The main difference between SEBAL and METRIC is that the latter uses the  $ET_r$  from a weather station, incorporating climatic conditions, while SEBAL uses the potential evaporation from a water body in the scene considering that sensible heat and soil heat fluxes are zero.

METRIC has been widely applied to estimate  $ET_a$  at field and regional scale over different crops such as wheat, corn, soybean and alfalfa, with errors ranging between 3 and 20% (Allen et al., 2007a; Choi et al., 2009; Mkhwanazi and Chávez, 2012). In recent years METRIC has been used to compute  $ET_a$  over sparse woody canopies such as vineyards and olive orchards (Carrasco-Benavides et al., 2012, 2014; Santos et al., 2012; Pôças et al., 2014) in both flat and mountainous terrains (Allen et al., 2013b).

The current implementations of SEBAL and METRIC imply the need to use more than one software to run the model (Allen et al., 2010) and they involve multiple steps. There are many different sub-models published for the estimation of some parameters (e.g. leaf area index, momentum roughness length, land surface temperature) that are not integrated into the current implementation of METRIC. Allen et al. (2013a) proposed a methodology for an automation procedure by using statistical conditions

and expert knowledge. This technique reduced the effect of human criteria helping to increase the model robustness. However, a software tool for the automatic selection of anchor pixels has not been published yet.

### About land surface energy balance models and crop evapotranspiration

As mentioned above, METRIC estimates  $ET_a$  as the residual from the surface energy balance equation considering information from satellite images and weather stations located near to the study site. Below, the key equations are detailed, beginning with the estimation of  $ET_a$  as the residual from the surface energy balance equation:

$$LE = R_n - G - H \quad (1)$$

where  $LE$  is latent heat flux consumed by  $ET_a$  ( $W \cdot m^{-2}$ );  $R_n$  is net radiation ( $W \cdot m^{-2}$ );  $G$  is soil heat flux ( $W \cdot m^{-2}$ ); and  $H$  is the sensible heat flux convected to the air ( $W \cdot m^{-2}$ ).

$R_n$  is calculated considering information obtained at the time of satellite overpass. Some correction processes are necessary, such as radiometric and atmospheric corrections.  $G$  is estimated using an empirical equation that considers mainly  $R_n$ , surface temperature, normalized difference vegetation index (NDVI), soil-adjusted vegetation index (SAVI) and albedo. More detailed information concerning the equations and models used in METRIC can be found in Allen et al. (2007b).

$H$  is the general equation of heat transport and is estimated using an approach called “calibration using inverse modeling at extreme conditions” (CIMEC) (Allen et al., 2013a). This method involves the selection of pixels with near extreme conditions (hot and cold anchor pixels) from which the  $ET_a$  can be estimated and assigned.  $H$  is computed as follows:

$$H = \frac{\rho \cdot c_p \cdot dT}{r_{ah}} \quad (2)$$

where  $dT$  is the difference between land surface and near-surface air temperatures,  $r_{ah}$  is the aerodynamic resistance to heat transport ( $s \cdot m^{-1}$ ),  $\rho$  is the air density ( $kg \cdot m^{-3}$ ) and  $c_p$  is the specific heat of air ( $1004 \cdot J \cdot kg^{-1} \cdot ^\circ K^{-1}$ ).  $dT$  is solved by using a linear relationship between air temperature and the estimated surface temperature of the the anchor pixels (Bastiaanssen et al., 1998a). To calculate  $r_{ah}$ , wind speed is extrapolated to a height at which forces of buoyancy and mechanical mix are equal (about 200 meters), using an iterative correction process based on the Monin-Obhukov equations (Allen, 1996; Bastiaanssen et al., 1998a).

After  $LE$  from Equation 1, it is possible to compute the instantaneous evapotranspiration values:

$$ET_{inst} = 3600 \cdot \frac{LE}{\lambda \rho_w} \quad (3)$$

where  $ET_{inst}$  is the instantaneous  $ET_a$  at the satellite overpass ( $mm \cdot h^{-1}$ ); 3600 is the conversion factor from seconds to hours;  $\rho_w$  is the density of water ( $1000 kg \cdot m^{-3}$ ); and  $\lambda$  is the water latent heat of vaporization ( $J \cdot kg^{-1}$ ).

Finally, the daily ET is computed pixel by pixel as:

$$ET_{24} = \frac{ET_{inst}}{ET_r} ET_{r\_24} \quad (4)$$

where  $ET_{inst}$  is the instantaneous  $ET_a$  estimated on equation 3;  $ET_r$  is the standardized 0.5 m tall alfalfa reference evapotranspiration at the image time and  $ET_{r\_24}$  is the cumulative 24 h  $ET_r$  for the image day. The relationship between  $ET_{inst}$  and  $ET_r$  is the reference ET fraction and is the same as the alfalfa based coefficient,  $K_c$ , and is used to extrapolate  $ET_a$  from the image time to periods of 24 hours or longer (Allen et al., 2007b).

### Sensible heat flux

As Allen et al. (2007b) mentioned, the computation of latent heat flux ( $LE$ ) is only as accurate as the summed estimates for  $R_n$ ,  $G$ , and  $H$ . Table 1 shows some errors reported by METRIC models for different crops. It can be seen that net radiation ( $R_n$ ) and soil heat flux ( $G$ ) present the lowest

**Table 1:** Root mean square error (RMSE) of energy balance components estimated using METRIC for different crops

Crop	Validation tool	$R_n$ ( $Wm^{-2}(\%)$ )	$G$ ( $Wm^{-2}(\%)$ )	$H$ ( $Wm^{-2}(\%)$ )	$LE$ ( $Wm^{-2}(\%)$ )	$ET$ ( $mmh^{-1}(\%)$ )	Source
grass	lysimeter	nr	nr	nr	nr	(4-22%) (mean = 4%)	1
sugar beet	lysimeter	nr	nr	nr	nr	(6-137%) (mean = 1%)	1
soybean	lysimeter	22.1 (4.1%)	14.2 (27.6%)	nr	nr	0.14 (17.6%)	2
corn, soybean	EC	nr	nr	39-48	34-44	0.58-0.89	3
olive	EC	nr	nr	nr	nr	0.14-1.2	4
vineyard	EC	24 (3.8%)	16 (9.4%)	39-59 (10-26.0%)	33-54 (14-27.2%)	0.9 (9%)	5

nr: not reported; EC: Eddy covariance

Sources: 1: Allen et al. (2007a); 2: Mkhwanazi and Chávez (2012, 2013); 3: González-Dugo et al. (2009); 4: Santos et al. (2012); Póças et al. (2014); 5: Poblete-Echeverría and Ortega-Farias (2012); Carrasco-Benavides et al. (2012, 2014)

estimation errors, while sensible heat flux ( $H$ ), is the hardest component of the surface energy balance to estimate.

One of the weaknesses of METRIC model reported in the literature is the selection of the anchor pixels. Long and Singh (2013) and Morton (2013) indicated that the selection of anchor pixels is subjective and depends on the ability of the operator to search and isolate the most appropriate hot and cold pixels. This process produces important biases in the estimation of  $H$ . Also, Choragudi (2011); Wang et al. (2009) mentioned that METRIC was very sensitive to the selection of the hot pixel. A group of possible candidates could have minimal differences in some attributes, but these can generate a big bias in the estimations. It means that the estimation of  $H$  in METRIC is very sensitive to the selection of anchor pixels.

The objective of this article is to propose an open implementation of a land surface energy balance model (LSEB) as an R package that integrates most of the METRIC sub-models and allows automatic selection of the anchor pixels. In this version of the package, specific functions for METRIC model (Allen et al., 2007b) are provided. Apart from the previous features, this package: i) is written in R, one of the most used scientific programming languages; ii) can be automated for batch processing of many satellite images; iii) provides functions for loading and processing satellite images; iv) provides functions and a new class object to manage weather station data; and v) is a free and open software.

## About the water package

### Package organization

The **water** package is developed in R to estimate actual evapotranspiration ( $ET_a$ ) from *Landsat satellite scenes* using Land Surface Energy Balance (LSEB) models, such as METRIC (Allen et al., 2007b).

Functions in **water** package are arranged in three groups: i) general functions to estimate sub-components of LSEB (e.g. leaf area index, albedo, land surface temperature, momentum roughness length); ii) specific functions to estimate the components of LSEB; iii) internal functions and methods to handle data from a weather station using a new proposed S3 class and functions to control global options such as saving results to disk, overwrite files, etc.

The first group of functions consist of models and equations to estimate the sub-components, which generally are controlled by the argument `method`. Most of the models available here are presented in the Appendix. In the second group, there are three functions: i) `METRIC.Rn()`; ii) `METRIC.G()` and iii) `METRIC.EB()`. The first one estimates net radiation using METRIC model. The second one estimates soil heat flux, and the third one estimate all the components of energy balance:  $R_n$ ,  $G$ ,  $H$  and  $LE$  according to METRIC model.

Three example datasets are provided with the **water** package: i) a subset of a Landsat 7 scene path 223, row 85 from 15th February 2013, bands 1 to 7; ii) data from a weather station from the same Landsat subset in CSV file format (comma separated values); iii) a subset of NASA SRTM digital elevation model, with the same spatial extent as the example image. These datasets are used in examples in Sections 46.3.2 and 46.3.3, and also in the vignettes included with the package.

The **water** package is available on The Comprehensive R Archive Network at <https://CRAN.R-project.org/package=water>. This software is made freely available under the terms and conditions of the GNU General Public License.

### Key functions in the water package

The key functions included in the package are:

`read.WSdata()` This function allows to import weather station data from a table in comma-separated values (CSV) format. The result is a new object of class "waterWeatherStation". The main input arguments are the CSV file and a vector with the order of the needed variables (radiation, temperature, wind speed and relative humidity) called `columns`. An optional argument is a Landsat metadata file (MTL). When this function is used with the CSV and the MTL files, it will interpolate the weather conditions to the moment of the satellite overpass.

`METRIC.EB()` This is the main function of `water` package. It runs each of the sub-models needed to get all the components of the LSEB (equation 1), from satellite and weather station data. The input arguments are: a satellite scene and a "waterWeatherStation" object. The arguments `alb.coeff`, `LAI.method`, `Zom.method` and `anchors.method` allow choosing between the different sub-models or coefficients used. More information about the sub-models available in `water` is presented in the Appendix. An optional logical argument is `plain`, which allows to use a digital elevation model or to consider that the surface is flat. When using a digital elevation model, the net radiation estimation and the land surface temperature are corrected using the elevation, slope and aspect of the surface. The output of this function is a raster layer object (from `raster` package) with 4 different layers: net radition, soil heat flux, sensible heat flux and surface temperature.

`ET24h()` This function estimates the  $ET_r$  for a 24-hour period. The input arguments are the components of the LSEB ( $R_n$ ,  $G$ ,  $H$ ) and a "waterWeatherStation" object. The argument `ET` allows to select between two  $ET_r$  methods. `ET="ETo"`, is the method for short crops, similar to clipped, cool-season grass and `ET="ETr"`, is the method for tall crops, similar to 0.5 m tall full-cover alfalfa. By default, it will use `ET="ETr"`, but the user should choose according to the conditions of the weather station. The output of this function is a raster layer object (from `raster` package) with the 24-hour  $ET_a$  in  $mm \cdot day^{-1}$ .

`METRIC.EB()` uses many different steps to estimate the parameters needed to calculate the components of the LSEB. These steps are available as individual functions like:

`albedo()` This function is used to calculate the broadband albedo from narrowband satellite data. This process involves applying a weighting function with empirical coefficients. `water` includes models and coefficients described by [Tasumi et al. \(2008\)](#) and [Liang \(2001\)](#), as well as new coefficients for Landsat 8 estimated by The Simple Model of the Atmospheric Radiative Transfer of Sunshine (SMARTS2) version 2.9.5 ([Gueymard, 1995](#)). `coeff="01medo"` computes clear sky spectral irradiances for the spectral range of each Landsat 8 OLI band (see Appendix, Section 46.6.1). The output of this function is a raster layer object (from `raster` package) with the albedo.

`LAI()` This function estimates the leaf area index (LAI) from Landsat data. `water` includes many different models to estimate LAI (see Appendix, Section 46.6.2). In the Examples Section of this article, the METRIC 2010 method ([Allen et al., 2010](#)) will be used. This method estimates LAI from SAVI ([Huete, 1988](#)), as follows:

$$SAVI = (1 + L)(\rho_{NIR} - \rho_R) / (L + \rho_{NIR} + \rho_R) \quad (5)$$

where  $\rho$  is the reflectance at the top-of-atmosphere from  $R$ , the red band, and from  $NIR$ , the near infrared band;  $L$  is a soil correction factor. By default `water` uses  $L=0.5$ , although [Allen et al. \(2007b\)](#) suggested a value of  $L=0.1$  in METRIC applications for western USA. This value varies by the amount or coverage of green vegetation: in very high coverage vegetation regions,  $L=0$  and  $SAVI = NDVI$ ; in areas with no green vegetation,  $L=1$ . Then SAVI is used to estimate LAI as follows:

$$LAI = 11 \cdot SAVI^3 \quad (6)$$

The output of this function is a raster layer object (from `raster` package) with the estimated  $LAI$ .

As it was mentioned before, the critical part in the estimation of the LSEB is the sensible heat flux estimation. The key functions used to estimate this are: `momentumRoughnessLength()`; `calcAnchors()` and `calcH()`.

`momentumRoughnessLength()` This function estimates the Momentum Roughness Length ( $Z_{om}$ ) from the average vegetation height around the weather station. `water` includes several methods to estimate  $Z_{om}$  from Landsat data (see Appendix, Section 46.6.4). For example, when `method="short-crops"` ([Allen et al., 2007b](#))  $Z_{om}$  is estimated as:

**Table 2:** Ranges of variables used for selection of anchor (cold and hot) pixels in `calcAnchors(method="CITRA-MCB")`

Variable	Cold pixel	Hot pixel
albedo*	0.18 - 0.25	0.13 - 0.15
NDVI*	0.76 - 0.84	0.10 - 0.28
LAI ( $m^2 \cdot m^{-2}$ )	3 - 6	-
$Z_{om}$ (m)	0.03 - 0.08	$\leq 0.005$

\* dimensionless

$$Z_{om} = 0.0018 \text{LAI} \quad (7)$$

And if `mountainous=TRUE`, this value of  $Z_{om}$  is corrected as:

$$Z_{om\_mtn} = Z_{om} \cdot \left(1 + \frac{(180/\pi) \cdot \text{slope} - 5}{20}\right) \quad (8)$$

The output of this function is a raster layer object with the estimated  $Z_{om}$

`calcAnchors()` This function automatically select anchor pixels that represent the dry and wet ends of the ET spectrum within the satellite scene using information of land surface characteristics (LAI, albedo,  $Z_{om}$ ,  $T_s$ ). When the anchor pixels are found, it assigns the  $ET_a$  estimates.

The criteria to select “hot” and “cold” pixels were adapted from [Allen et al. \(2011\)](#), [Tasumi \(2003\)](#) and shown in Table 2. The methodology searches for image-specific pixels with the lowest and highest temperature values that match with these criteria. The output of this function is a data frame with the coordinates of the selected anchor pixels.

`calcH()` This function applies the CIMEC self-calibration method in order to generate an iterative process for the “hot” and “cold” pixels and absorb all biases in the computation of  $H$ . A near surface temperature difference ( $dT$ ) is used in place of a surface air temperature difference to drive the determination of sensible heat flux, in an iterative correction process. The convergence of the function is reached when the change in the estimated aerodynamic resistance is less than 1% for the cold and hot condition. There is an argument called `verbose` to control how much information about the iterative process is shown in the output. The output of this function is a raster layer object with the estimated soil heat flux.

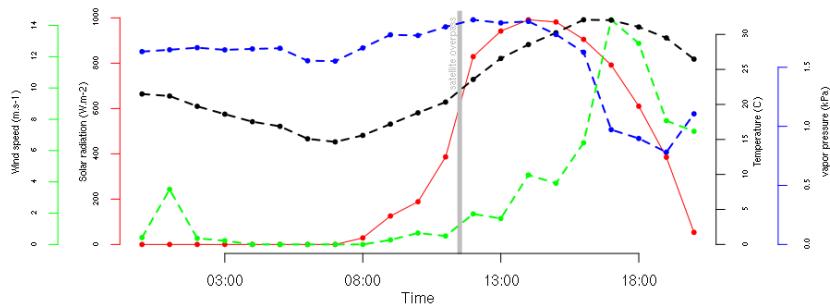
## Input data requirements

METRIC uses two sources to estimate the LSEB: a satellite image and a weather station with hourly data.

METRIC can be run using different satellite sensors. Currently, the coefficients needed for Landsat 7 and 5 are available in [Allen et al. \(2007b\)](#). Those coefficients can also be applied to Landsat 8 data. In the Appendix Section 46.6.1, we propose specific coefficients for the estimation of albedo using this satellite sensor. Other coefficients to run METRIC using MODIS images are available in [Allen et al. \(2007b\)](#), and will be included in future versions of `water` package.

The weather station data should include near-surface air temperature, wind speed, relative humidity and solar radiation. The `water` package uses the function `read.WSdata()` to convert the weather station data from a comma-separate values table to a special R object. The input data should be on an hourly or shorter time basis. The expected units are  $^{\circ}\text{C}$  for temperature;  $m \cdot s^{-1}$  for wind speed; % for relative humidity and  $W \cdot m^{-2}$  for solar radiation. However, this function uses a parameter `cf` when conversion factors are needed to convert the variables from different units (e.g. if wind speed is in  $km \cdot h^{-1}$  the conversion factor for this variable should be 0.278). To estimate  $ET_r$  from the weather station data using [Allen et al. \(2005\)](#) equation, more information is needed: position in latitude and longitude, and the wind sensor height in meters. When the weather station data is being imported, a satellite metadata file can be included as a function argument. This allows interpolating the weather conditions at the exact moment of the satellite overpass.

The `water` package only uses one weather station to estimate  $ET_r$  for an entire Landsat scene of  $180 \times 180 \text{ km}$ . The model uses  $ET_r$  to derive the ET reference fraction ( $ET_rF$ ) at image time (using equation 4). This assumes that  $ET_a$  in the entire area changes in proportion to the change in  $ET_r$  at the weather station ([Allen et al., 2007b](#)). This means that  $ET_r$  is only used as an index of the relative



**Figure 1:** Plot of the hourly weather station data. The conditions at the time of the sattelite overpass are marked by a gray bar. This is the default plot for `waterWeatherStation` objects.

change and this is retained through the  $ET_rF$ . Any biases caused by variation in weather conditions should be canceled by using the same  $ET_rF$  for both instantaneous and 24 h period. Nevertheless, it is recommended to use a spatial mask when the weather conditions are heterogeneous, for example in irrigated areas surrounded by deserts.

### Performance and memory use

The **water** package uses large temporal memory in order to obtain the results and sub products. Most of the results are "RasterLayer" or "RasterStack" objects from **raster** package (Hijmans, 2015). Processing an entire Landsat scene could need more than 2 gigabytes of memory to store the temporal data. One approach to solve this is to write products and sub-products to the disk. There is an option `writeResults=TRUE` in function `waterOptions()` to force **water** to store the results on the disk, instead of on temporal memory.

If **water** runs out of memory while processing data, it will usually stop working without a warning message. We suggest processing only a portion of a Landsat scene using an area-of-interest (aoi) polygon or storing results to disk.

### Example code and datasets

Two different approaches to estimate land surface energy balance demonstrate the features and procedures in **water**. The first example in Section 46.3.2 is a simple procedure, and the second one in Section 46.3.3 refers to an advanced procedure. Finally, the estimation of  $ET_a$  from the output of any of the previous procedures is demonstrated in Section 46.3.4.

In section 46.3 functions `createAoi()` and `read.WSdata()` are summarized. Then, in section 46.3.2 function `METRIC.EB()` is shown. In section 46.3.3 the LSEB is estimated step by step. And later, in section 46.3.4 daily  $ET_r$  is estimated using functions `dailyET()` and `ET24h()`.

## Estimating ET<sub>a</sub> using METRIC model and water package

### Base data preparation

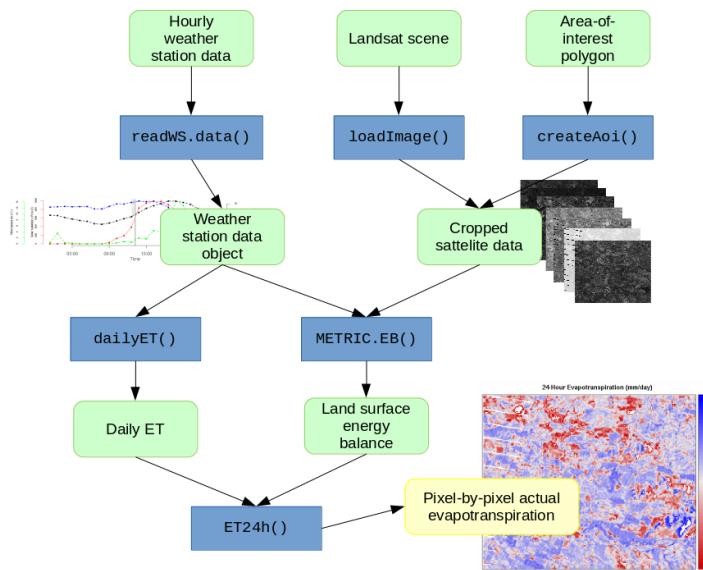
To calculate METRIC Actual Evapotranspiration using **water** package, three sources are needed:

1. A raw Landsat 7-8 satellite image.
2. A Weather Station data (.csv file).
3. A polygon with our Area-of-interest (AOI) Spatial-Polygon object, to run the model using only a portion of the satellite scene.

First, AOI is created as a polygon using bottomright and topleft coordinates:

```
aoi <- createAoi(topleft = c(272955, 6085705),
                  bottomright = c(288195, 6073195), EPSG = 32719)
```

Then, the weather station data is loaded using the function `read.WSdata()`. This function converts the CSV file into a "waterWeatherStation" object. Then, if a Landsat metadata file (MTL file) is provided, the time-specific weather conditions at the time of satellite overpass will be calculated. This is shown on Figure 1 as a gray bar. Files 'apples.csv' and 'L7.MTL.txt' are included in the package as raw data. In R, `system.file()` is used to call this files.



**Figure 2:** Schematic diagram with the functions, data inputs and outputs used in the simple procedure when running the METRIC model with **water** package. The green rounded boxes represent data, the blue boxes represent the functions and the yellow box is the final result.

```

 csvfile <- system.file("exdata", "apples.csv", package = "water")
 MTLfile <- system.file("exdata", "L7.MTL.txt", package = "water")

 WeatherStation <- read.WSdata(WSdata=csvfile,
                                date.format = "%d/%m/%Y",
                                lat = -35.42222, long = -71.38639,
                                elev = 201, height = 2.2,
                                MTL = MTLfile)
  
```

Next, the Landsat satellite image is loaded. **water** provides a function to load a Landsat image (`loadImage()`) from TIFF files. Landsat images can be downloaded directly from USGS archives in *Earth Explorer* (<http://earthexplorer.usgs.gov/>). In this article, an example dataset will be used which comes with **water** package as demonstration data.

```
image.DN <- L7_Talca
```

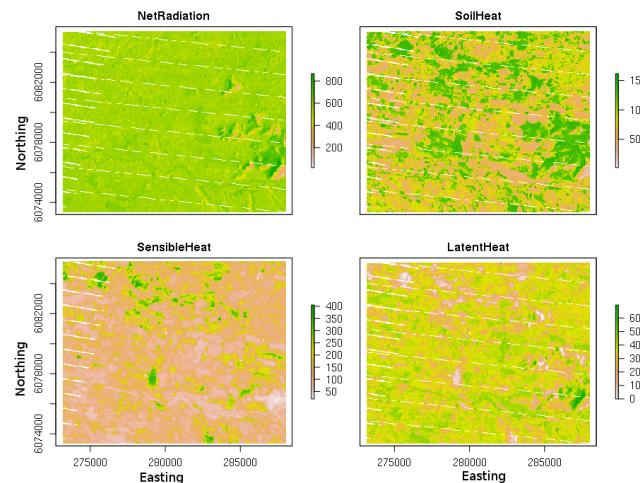
Finally, Digital Elevation Model (DEM) will be created for the area being processed. **water** provides two functions to do this: `checkSRTMgrids()` will search for the downloadable grid files in <http://earthexplorer.usgs.gov/>. However, this function will only print the links to the files. The downloading process has to be done manually. After this, `prepareSRTMdata()` can be used to mosaic and clip those files using the same extent of the image. In this article, the example data, provided with **water** package, will be loaded.

```
DEM <- DEM_Talca
```

### Simple procedure

The simple procedure is summarized on Figure 2. The function `METRIC.EB()` will be used to estimate the land surface energy balance. This function has many parameters to choose from the different METRIC model equations. e.g. changes can be made in:

- Coefficients used to estimate broadband albedo from narrowband data.
- Model to estimate Leaf Area Index (LAI) from satellite data.
- Model to estimate momentum roughness length ( $Z_{om}$ )
- Automatic method for the selection of anchor pixels
- Reference ET coefficient and momentum roughness length estimated for the weather station



**Figure 3:** Land surface energy balance ( $\text{W} \cdot \text{m}^{-2}$ ) estimated using METRIC with **water** package

When this function is run, the energy balance and the surface temperature ( $T_s$ ) used are assigned to the Energy.Balance object. This function prints the position and characteristics of the anchor pixels to the console. Also, a plot with the values of the aerodynamic resistance during the iterative process is generated after every iteration. Here, the logical argument verbose controls how much information is shown in the output, and the plotting of the diagnostic graph.

```
Energy.Balance <- METRIC.EB(image.DN = image.DN,
                             plain = FALSE, DEM = DEM,
                             WeatherStation = WeatherStation, ETp.coef = 1.2,
                             MTL = MTLfile, sat = "L7",
                             thermalband = image.DN$thermal.low)
```

The results of the energy balance estimated using this function are shown in Figure 3. The console output with information related to the anchor pixels goes like this:

pixel	X	Y	$T_s$	LAI	type	
1	139253	282420	-3922830	323.1587	0.13	hot
2	121566	274710	-3921780	310.0151	4.40	cold

### Advanced procedure

The advanced procedure involves running many different functions one-by-one, this is summarized in figures 4 and 5. These functions were run inside the code of METRIC.EB() in the previous example. Running **water** with this procedure allows to have more control in the different arguments used.

### Net Radiation estimation

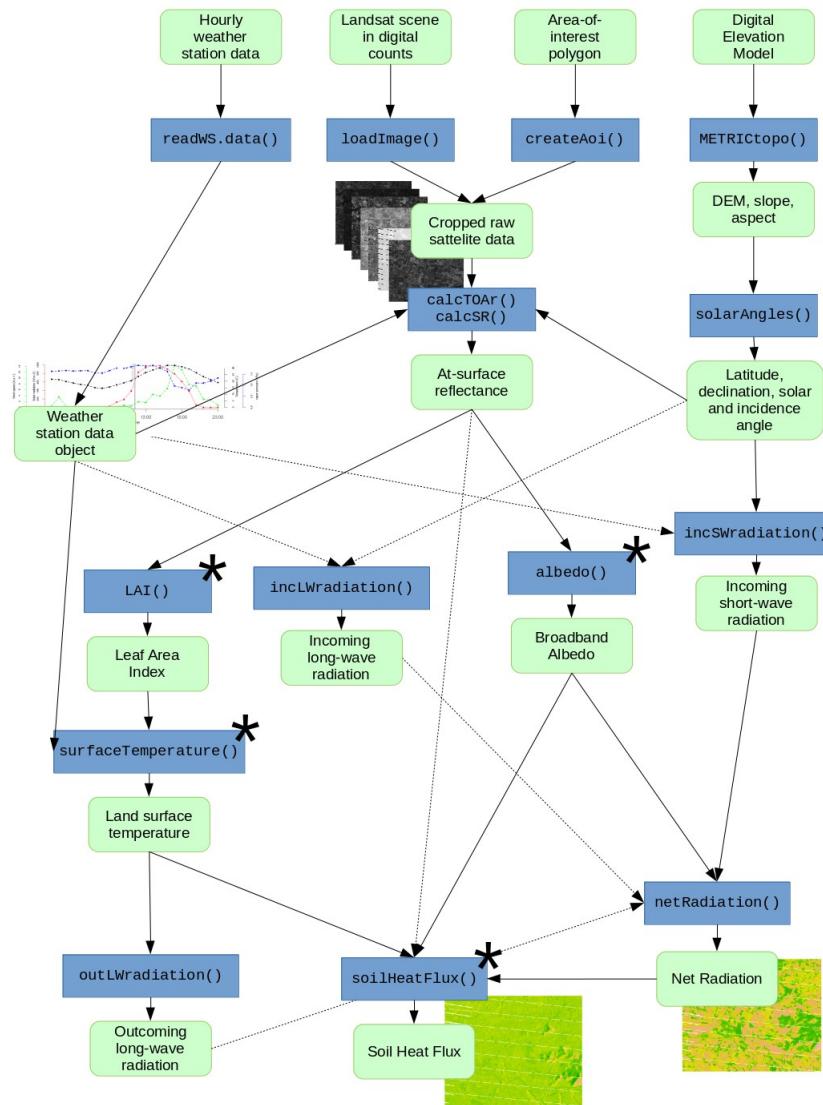
In order to calculate the  $R_n$  for the loaded Landsat satellite data, a surface model (slope + aspect) from the DEM is calculated, then the solar angles (latitude, declination, hour angle and solar incidence angle) are calculated. Then incSWradiation() is used to calculate incoming solar radiation.

```
surface.model <- METRICtopo(DEM)

solar.angles.r <- solarAngles(surface.model = surface.model,
                                WeatherStation = WeatherStation, MTL = MTLfile)

Rs.inc <- incSWradiation(surface.model = surface.model,
                           solar.angles = solar.angles.r,
                           WeatherStation = WeatherStation)
```

After this, reflectances are calculated at the top-of-atmosphere (TOA), and surface reflectance derived from the Landsat image as:



**Figure 4:** Schematic diagram with the functions, data inputs and outputs used in the advanced procedure when estimating Net Radiation and Soil Heat Flux running the METRIC model with **water** package. The green rounded boxes represent data and the blue boxes represent the functions. The functions marked with “\*” have multiple methods available.

```
image.TOAr <- calcTOAr(image.DN = image.DN, sat = "L7", MTL = MTLfile,
                        incidence.rel = solar.angles.r$incidence.rel)

image.SR <- calcSR(image.TOAr = image.TOAr, sat = "L7",
                     surface.model = surface.model,
                     incidence.hor = solar.angles.r$incidence.hor,
                     WeatherStation = WeatherStation, ESPA = FALSE)
```

Following this, *broadband albedo* is calculated as the sum of visible to near infrared narrowband satellite bands and coefficients related to atmospheric transmittance of global solar beam radiation. In this example `coeff="Tasumi"` was used.

```
albedo <- albedo(image.SR=image.SR, coeff="Tasumi")
```

Later on, *Leaf Area Index (LAI)* is calculated using the satellite data. In this example `method=metric2010` is used:

```
LAI <- LAI(method="metric2010", image=image.TOAr, L=0.1)
```

Land surface temperature ( $T_s$ ) is estimated using computed LAI values in order to estimate consequently the surface emissivity and brightness temperature from Landsat's thermal band (TIR). Then this information is used to compute the incoming and outgoing long-wave radiation as:

```
Ts <- surfaceTemperature(LAI = LAI, sat = "L7",
                           thermalband = image.DN$thermal.low,
                           WeatherStation = WeatherStation)

Rl.out <- outLWradiation(LAI = LAI, Ts = Ts)

Rl.inc <- incLWradiation(WeatherStation, DEM = surface.model$DEM,
                           solar.angles = solar.angles.r, Ts = Ts)
```

Finally, Net Radiation ( $R_n$ ) can be estimated pixel by pixel as follows:

```
Rn <- netRadiation(LAI, albedo, Rs.inc, Rl.inc, Rl.out)
```

## Soil Heat Flux estimation

Soil heat flux is estimated  $G$ , using as input data the  $R_n$ , surface reflectance,  $T_s$ , LAI and albedo. In this example the original METRIC (2007) based-method will be used, which is:

```
G <- soilHeatFlux(image = image.SR, Ts = Ts, albedo = albedo,
                    Rn = Rn, LAI = LAI)
```

## Sensible Heat Flux estimation

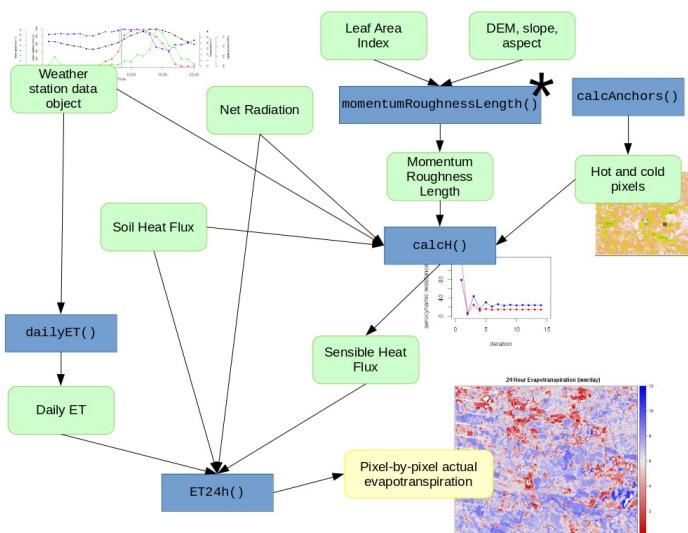
To estimate the sensible heat fluxes derived from the Landsat satellite data, first, the calculation of the momentum roughness length ( $Z_{om}$ ) is needed.

```
Z.om <- momentumRoughnessLength(LAI = LAI, mountainous = TRUE,
                                    method = "short.crops",
                                    surface.model = surface.model)
```

Then, `calcAnchors()` is used to search for the anchor pixels within the Landsat scene. And finally, `calcH()` is used to run the CIMEC process and estimate the sensible heat flux:

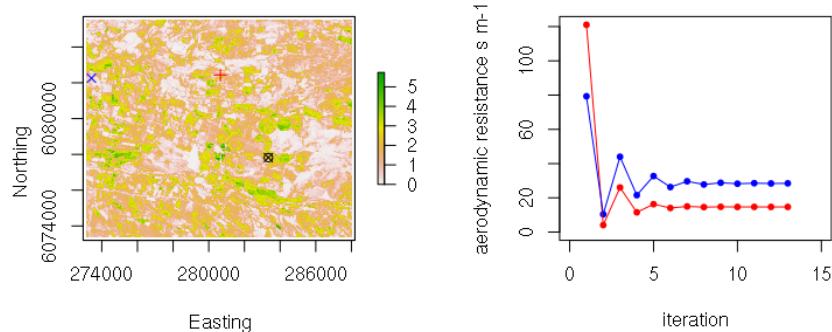
```
hot.and.cold <- calcAnchors(image = image.TOAr, Ts = Ts,
                               LAI = LAI, plots = FALSE, albedo = albedo,
                               Z.om = Z.om, n = 1,
                               anchors.method = "CITRA-MCB",
                               deltaTemp = 5, verbose = FALSE)

H <- calcH(anchors = hot.and.cold, Ts = Ts, Z.om = Z.om,
            WeatherStation = WeatherStation, ETp.coef = 1.05,
            Z.om.ws = 0.0018, DEM = DEM, Rn = Rn, G = G, verbose = TRUE)
```



**Figure 5:** Schematic diagram with the functions, data inputs and outputs used in the advanced procedure when estimating the sensible heat flux running the METRIC model with **water** package. The green rounded boxes represent data, the blue boxes represent the functions and the yellow box is the final result. The functions marked with \* have multiple methods available.

When the function `calcH()` is running, and `verbose=TRUE`, the output shows the intermediate values of the CIMEC process parameters. Also, the value for the aerodynamic resistance and its change for every iteration is plotted iteratively by this function. This plot is shown on Figure 6. In this example, the change in the value of the aerodynamic resistance goes down after iteration #9, and in iteration #14 is less than 1%.



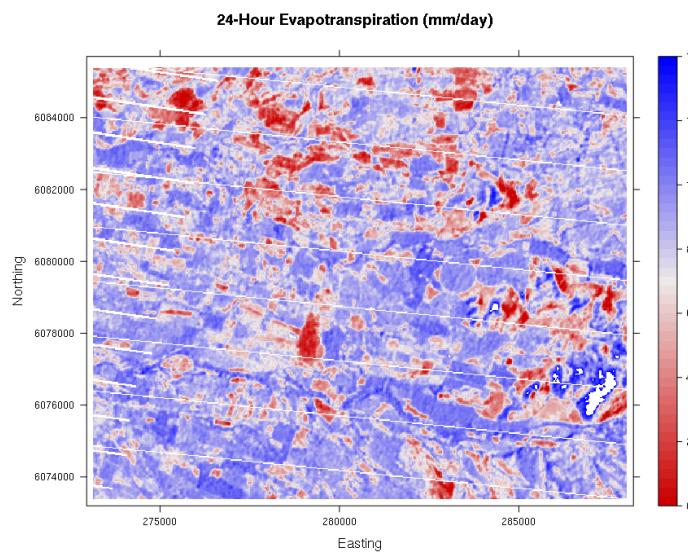
**Figure 6:** Leaf area index  $m^2 m^{-2}$  and the position of the hot pixel (red cross), cold pixel (blue X) and the weather station (circle with a X) (left). Change on aerodynamic resistance convergence in the iterative process for hot (red) and cold (blue) conditions (right).

## Daily crop evapotranspiration estimation

To estimate the daily actual evapotranspiration from the Landsat scene, the daily reference ET ( $ET_r$ ) is needed. The daily  $ET_r$  can be calculated with `dailyET()`. This function calculates the cumulative 24h standardized reference evapotranspiration for the day of the image using the ASCE standardized Penman-Monteith method (Allen et al., 2005). Finally, 24h crop ET can be estimated for every pixel of the Landsat scene using the function `ET24h()` (Figure 7):

```
ET_WS <- dailyET(WeatherStation = WeatherStation, ET = "ETr")
```

```
ET.24 <- ET24h(Rn = Rn, G = G, H = H\$H,  
                 Ts = Ts, WeatherStation = WeatherStation,  
                 ETr.daily = ET_WS)
```



**Figure 7:** Crop evapotranspiration in mm/day, estimated using METRIC and **water** package

## Conclusions

The **water** package offers a fast and reliable platform for estimating actual evapotranspiration using the land surface energy balance. It includes different methods for many sub-models. The simple procedure showed in this article allows to estimate the  $ET_a$  in a simple and fast way. The advanced procedure allows to have more control in the different available methods. Further versions of this package will implement other LSEB models such as SEBAL. Also, other satellite sensors such as MODIS will be included. Because **water** is written in R, a language very popular in the scientific community and published as free software, further developments could come from a wide community of users.

## Acknowledgments

This work was supported by the Argentinian government through the projects INTA 1133043 and 1133042, and Universidad de Talca, Chile through the research program “Adaptation of Agriculture to Climate Change (A2C2)”. The authors wish to thank Danlu Guo and an anonymous reviewer for their careful reading of the manuscript and their enriching comments. Finally, we thank Marcos Angelini, Rosana Vallone and Marcos Carrasco-Benavides for their valuable support and suggestions to improve this work.

## Bibliography

- R. Allen, T. M., R. Trezza, and J. Kjaersgaard. *METRIC. Mapping Evapotranspiration at High Resolution. Applications Manual for Landsat Satellite Imagery*. University of Idaho, Kimberly, Idaho, version 2.0.5 edition, January 2010. [p352, 355]
- R. Allen, A. Irmak, R. Trezza, J. M. H. Hendrickx, W. Bastiaanssen, and J. Kjaersgaard. Satellite-based ET estimation in agriculture using SEBAL and METRIC. *Hydrological Processes*, 25(26):4011–4027, Dec. 2011. ISSN 08856087. doi: 10.1002/hyp.8408. [p352, 356]
- R. G. Allen. Assessing integrity of weather data for reference evapotranspiration estimation. *Journal of Irrigation and Drainage Engineering*, 122(2):97–106, 1996. ISSN 0733-9437. doi: 10.1061/(ASCE)0733-9437(1996)122:2(97). [p353]
- R. G. Allen, L. S. Pereira, D. Raes, and M. Smith. *Crop evapotranspiration - Guidelines for computing crop water requirements*, volume 56. Food & Agriculture Org, 1998. ISBN 92-5-104219-5. [p352]
- R. G. Allen, I. A. Walter, E. R., T. Howell, D. Itenfisu, and M. Jensen. The ASCE standardized reference evapotranspiration equation. Technical report, 2005. [p352, 356, 363]

- R. G. Allen, M. Tasumi, A. Morse, R. Trezza, J. L. Wright, W. Bastiaanssen, W. Kramber, I. Lorite, and C. W. Robison. Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC)—applications. *Journal of Irrigation and Drainage Engineering*, 133(4):395–406, Aug. 2007a. ISSN 0733-9437. doi: 10.1061/(ASCE)0733-9437(2007)133:4(395). [p352, 354]
- R. G. Allen, M. Tasumi, and R. Trezza. Satellite-based energy balance for mapping evapotranspiration with internalized calibration (METRIC)—model. *Journal of Irrigation and Drainage Engineering*, 133: 380, 2007b. [p352, 353, 354, 355, 356, 368, 369, 370]
- R. G. Allen, B. Burnett, W. Kramber, J. Huntington, J. Kjaersgaard, A. Kilic, C. Kelly, and R. Trezza. Automated calibration of the METRIC-Landsat evapotranspiration process. *JAWRA Journal of the American Water Resources Association*, 49(3):563–576, June 2013a. ISSN 1093474X. doi: 10.1111/jawr.12056. [p352, 353]
- R. G. Allen, R. Trezza, A. Kilic, M. Tasumi, and H. Li. Sensitivity of landsat-scale energy balance to aerodynamic variability in mountains and complex terrain. *JAWRA Journal of the American Water Resources Association*, 49(3):592–604, June 2013b. ISSN 1093474X. doi: 10.1111/jawr.12055. [p352]
- Z. Baruch and M. Fisher. Factores climáticos y de competencia que afectan el desarrollo de la planta en el establecimiento de una pastura. *Establecimiento y renovación de pasturas. CIAT. Cali, Colombia*, pages 103–142, 1991. [p352]
- W. Bastiaanssen, M. Menenti, R. Feddes, and A. Holtslag. A remote sensing surface energy balance algorithm for land (SEBAL). 1. Formulation. *Journal of Hydrology*, 212-213:198–212, Dec. 1998a. ISSN 00221694. doi: 10.1016/S0022-1694(98)00253-4. [p352, 353]
- W. G. M. Bastiaanssen, M. Menenti, R. a. Feddes, and a. a. M. Holtslag. A remote sensing surface energy balance algorithm for land (SEBAL): 2. Validation. *Journal of Hydrology*, 212-213(1-4):198–212, 1998b. ISSN 00221694. doi: 10.1016/S0022-1694(98)00253-4. [p352]
- M. Carrasco-Benavides, S. Ortega-Farías, L. O. Lagos, J. Kleissl, L. Morales, C. Poblete-Echeverría, and R. G. Allen. Crop coefficients and actual evapotranspiration of a drip-irrigated Merlot vineyard using multispectral satellite images. *Irrigation Science*, 30(6):485–497, aug 2012. ISSN 0342-7188. doi: 10.1007/s00271-012-0379-4. [p352, 354]
- M. Carrasco-Benavides, S. Ortega-Farías, L. Lagos, J. Kleissl, L. Morales-Salinas, and A. Kilic. Parameterization of the satellite-based model METRIC for the estimation of instantaneous surface energy balance components over a drip-irrigated vineyard. *Remote Sensing*, 6(11):11342–11371, Nov. 2014. ISSN 2072-4292. doi: 10.3390/rs61111342. [p352, 354, 368]
- M. Choi, W. P. Kustas, M. C. Anderson, R. G. Allen, F. Li, and J. H. Kjaersgaard. An intercomparison of three remote sensing-based surface energy balance algorithms over a corn and soybean production region (Iowa, U.S.) during SMACEX. *Agricultural and Forest Meteorology*, 149(12):2082–2097, Dec. 2009. ISSN 01681923. doi: 10.1016/j.agrformet.2009.07.002. [p352]
- V. N. R. K. Choragudi. Sensitivity analysis on mapping evapotranspiration at high resolution using internal calibration (METRIC). *Civil Engineering Theses, Dissertations, and Student Research. Paper 35.* <http://digitalcommons.unl.edu/civilengdiss/35>, 2011. [p354]
- R. Cragoa and W. Brutsaert. Daytime evaporation and the self-preservation of the evaporative fraction and the Bowen ratio. *Journal of Hydrology*, 178(1–4):241 – 255, 1996. ISSN 0022-1694. doi: [http://dx.doi.org/10.1016/0022-1694\(95\)02803-X](http://dx.doi.org/10.1016/0022-1694(95)02803-X). [p352]
- R. Ferreyra, G. Sellés, and J. Tosso. Efecto de diferentes alturas de agua sobre el cultivo del pimiento. I. Influencia de los excesos de humedad. *Agricultura Técnica*, 45(1):47–51, 1985. [p352]
- M. Gonzalez-Dugo, C. Neale, L. Mateos, W. Kustas, J. Prueger, M. Anderson, and F. Li. A comparison of operational remote sensing-based models for estimating crop evapotranspiration. *Agricultural and Forest Meteorology*, 149(11):1843–1853, nov 2009. ISSN 01681923. doi: 10.1016/j.agrformet.2009.06.012. [p354]
- C. Gueymard. *SMARTS2: a simple model of the atmospheric radiative transfer of sunshine: algorithms and performance assessment*. Florida Solar Energy Center Cocoa, FL, 1995. [p355]
- R. J. Hijmans. *raster: Geographic Data Analysis and Modeling*, 2015. URL <http://CRAN.R-project.org/package=raster>. R package version 2.4-20. [p357]
- A. Huete. A soil-adjusted vegetation index (SAVI). *Remote Sensing of Environment*, 25(3):295 – 309, 1988. ISSN 0034-4257. doi: [http://dx.doi.org/10.1016/0034-4257\(88\)90106-X](http://dx.doi.org/10.1016/0034-4257(88)90106-X). [p355]

- J. C. Jimenez-Munoz, J. Cristobal, J. A. Sobrino, G. S??ria, M. Ninyerola, and X. Pons. Revision of the single-channel algorithm for land surface temperature retrieval from Landsat thermal-infrared data. *IEEE Transactions on Geoscience and Remote Sensing*, 47(1):339–349, 2009. ISSN 01962892. doi: 10.1109/TGRS.2008.2007125. [p369]
- J. C. Jimenez-Munoz, J. A. Sobrino, D. Skokovic, C. Mattar, and J. Cristobal. Land surface temperature retrieval methods from Landsat-8 thermal infrared sensor data. *IEEE Geoscience and Remote Sensing Letters*, 11(10):1840–1843, 2014. ISSN 1545598X. doi: 10.1109/LGRS.2014.2312032. [p369]
- L. F. Johnson, D. E. Roczen, S. K. Youkhana, R. R. Nemani, and D. F. Bosch. Mapping vineyard leaf area with multispectral satellite imagery. *Computers and Electronics in Agriculture*, 38(1):33–44, 2003. ISSN 01681699. doi: 10.1016/S0168-1699(02)00106-0. [p368]
- S. Liang. Narrowband to broadband conversions of land surface albedo I Algorithms. *Remote Sens. Environ.*, 76:213–238, 2001. ISSN 00344257. doi: 10.1016/S0034-4257(00)00205-4. [p355, 368]
- D. Long and V. P. Singh. Assessing the impact of end-member selection on the accuracy of satellite-based spatial variability models for actual evapotranspiration estimation. *Water Resources Research*, 49(5):2601–2618, 2013. [p354]
- R. López-Urrea, a. Montoro, P. López-Fuster, and E. Fereres. Evapotranspiration and responses to irrigation of broccoli. *Agricultural Water Management*, 96(7):1155–1161, 2009. ISSN 03783774. doi: 10.1016/j.agwat.2009.03.011. [p352]
- A. Millar. *Manejo de agua y producción agrícola*. IICA, 1993. [p352]
- M. M. Mkhwanazi and J. L. Chávez. Using METRIC to estimate surface energy fluxes over an alfalfa field in Eastern Colorado. In *32nd Annual AGU Hydrology Days*, volume 1, pages 90–98. Colorado State University, 2012. [p352, 354]
- M. M. Mkhwanazi and J. L. Chávez. Mapping evapotranspiration with the remote sensing ET algorithms METRIC and SEBAL under advective and non-advective conditions : accuracy determination with weighing lysimeters. (970), 2013. [p354]
- C. G. Morton. *Assessing Calibration Uncertainty and Automation for Estimating Evapotranspiration from Agricultural Areas Using METRIC*. PhD thesis, Department of Geography, University of Nevada, 2013. [p354]
- S. Ortega-Farías, R. Cuenca, and M. English. Hourly grass evapotranspiration in modified maritime environment. *Journal of Irrigation and Drainage Engineering*, 121(6):369–373, nov 1995. ISSN 0733-9437. doi: 10.1061/(ASCE)0733-9437(1995)121:6(369). [p352]
- A.-C. Parent and F. Anctil. Quantifying evapotranspiration of a rainfed potato crop in South-eastern Canada using eddy covariance techniques. *Agricultural Water Management*, 113:45 – 56, 2012. ISSN 0378-3774. doi: http://dx.doi.org/10.1016/j.agwat.2012.06.014. [p352]
- J. O. Payero and S. Irmak. Construction, installation, and performance of two repacked weighing lysimeters. *Irrigation Science*, 26(2):191–202, 2008. ISSN 03427188. doi: 10.1007/s00271-007-0085-9. [p352]
- T. Paço, M. Ferreira, and N. Conceição. Peach orchard evapotranspiration in a sandy soil: Comparison between eddy covariance measurements and estimates by the FAO 56 approach. *Agricultural Water Management*, 85(3):305 – 313, 2006. ISSN 0378-3774. doi: http://dx.doi.org/10.1016/j.agwat.2006.05.014. [p352]
- C. Poblete-Echeverría and S. Ortega-Farias. Calibration and validation of a remote sensing algorithm to estimate energy balance components and daily actual evapotranspiration over a drip-irrigated Merlot vineyard. *Irrigation Science*, 30(6):537–553, 2012. ISSN 03427188. doi: 10.1007/s00271-012-0381-x. [p354]
- C. Poblete-Echeverría and S. Ortega-Farias. Evaluation of single and dual crop coefficients over a drip-irrigated Merlot vineyard (*Vitis vinifera L.*) using combined measurements of sap flow sensors and an eddy covariance system. *Australian Journal of Grape and Wine Research*, 19(2):249–260, 2013. ISSN 1755-0238. doi: 10.1111/ajgw.12019. URL <http://dx.doi.org/10.1111/ajgw.12019>. [p352]
- I. Pôças, T. a. Paço, M. Cunha, J. a. Andrade, J. Silvestre, A. Sousa, F. L. Santos, L. S. Pereira, and R. G. Allen. Satellite-based evapotranspiration of a super-intensive olive orchard: Application of METRIC algorithms. *Biosystems Engineering*, 8, 2014. ISSN 15375110. doi: 10.1016/j.biosystemseng.2014.06.019. [p352, 354, 368, 370]

- C. Santos, I. J. Lorite, R. G. Allen, and M. Tasumi. Aerodynamic parameterization of the satellite-based energy balance (METRIC) model for ET estimation in rainfed olive orchards of Andalusia, Spain. *Water Resources Management*, 26(11):3267–3283, 2012. ISSN 09204741. doi: 10.1007/s11269-012-0071-8. [p352, 354, 370]
- M. Tasumi. *Progress in operacional estimation of regional evapotranspiration using satellite imagery*. PhD thesis, University of Idaho (USA), 2003. [p356]
- M. Tasumi, R. G. Allen, and R. Trezza. At-surface reflectance and albedo from satellite for operational calculation of land surface energy balance. *Journal of Hydrologic Engineering*, 13(2):51–63, 2008. doi: 10.1061/(ASCE)1084-0699(2008)13:2(51). [p355, 368]
- D. P. Turner, W. B. Cohen, R. E. Kennedy, K. S. Fassnacht, and J. M. Briggs. Relationships between leaf area index and Landsat TM spectral vegetation indices across three temperate zone sites. *Remote Sensing of Environment*, 70(1):52–68, 1999. ISSN 00344257. doi: 10.1016/S0034-4257(99)00057-7. [p368]
- T. Twine, W. Kustas, J. Norman, D. Cook, P. Houser, T. Meyers, J. Prueger, P. Starks, and M. Wesely. Correcting eddy-covariance flux underestimates over a grassland. *Agricultural and Forest Meteorology*, 103(3):279 – 300, 2000. ISSN 0168-1923. doi: [http://dx.doi.org/10.1016/S0168-1923\(00\)00123-4](http://dx.doi.org/10.1016/S0168-1923(00)00123-4). [p352]
- J. Wang, T. Sammis, V. Gutschick, M. Gebremichael, and D. Miller. Sensitivity analysis of the surface energy balance algorithm for land (SEBAL). *Transactions of the ASABE*, 52(3):801–811, 2009. [p354]
- G. E. Wukelic, D. E. Gibbons, L. M. Martucci, and H. P. Foote. Radiometric calibration of Landsat Thematic Mapper Thermal Band. *Remote Sensing of Environment*, 28:339–347, June 1989. [p369]

## Appendix

### Included models for albedo

1. `coeff="Tasumi"` ([Tasumi et al., 2008](#))

$$\text{albedo} = \rho_{s,B} \cdot 0.254 + \rho_{s,G} \cdot 0.149 + \rho_{s,R} \cdot 0.147 + \rho_{s,NIR} \cdot 0.311 + \rho_{s,SWIR1} \cdot 0.103 + \rho_{s,SWIR2} \cdot 0.036 \quad (9)$$

2. `coeff="Liang"` ([Liang, 2001](#))

$$\text{albedo} = \rho_{s,B} \cdot 0.356 + \rho_{s,R} \cdot 0.130 + \rho_{s,NIR} \cdot 0.373 + \rho_{s,SWIR1} \cdot 0.085 + \rho_{s,SWIR2} \cdot 0.072 - 0.0018 \quad (10)$$

3. `coeff="Olmedo"`

$$\text{albedo} = \rho_{s,B} \cdot 0.246 + \rho_{s,G} \cdot 0.146 + \rho_{s,R} \cdot 0.191 + \rho_{s,NIR} \cdot 0.304 + \rho_{s,SWIR1} \cdot 0.105 + \rho_{s,SWIR2} \cdot 0.008 \quad (11)$$

where  $\rho_{s,b}$  is the surface reflectance for band  $b$ .

### Included models for Leaf Area Index

1. `method="metric"` ([Allen et al., 2007b](#))

$$\text{SAVI}_{ID} = (1 + L)(\rho_{t,NIR} - \rho_{t,R}) / (L + \rho_{t,NIR} + \rho_{t,R}) \quad (12)$$

where  $\rho$  is the reflectance at top-of-atmosphere, and the subindex refers to bands  $R$ :red or  $NIR$ :near infrared; and  $L$  is a soil correction factor. The default value used for  $L$  is 0.5, and in METRIC applications in western us, [Allen et al. \(2007b\)](#) suggested a value of  $L=0.1$ . And Leaf Area Index is:

$$\text{LAI} = -\frac{\ln((0.69 - \text{SAVI}_{ID}) / 0.59)}{0.91} \quad (13)$$

2. `method="metric2010"` ([Pôças et al., 2014](#))

$$\text{LAI} = 11 \cdot \text{SAVI}_{ID}^3 \quad (14)$$

3. `method="vineyard"` ([Johnson et al., 2003](#))

$$\text{NDVI} = (\rho_{t,NIR} - \rho_{t,R}) / (\rho_{t,NIR} + \rho_{t,R}) \quad (15)$$

where  $\rho$  is the reflectance at top-of-atmosphere, and the subindex refers to bands  $R$ :red or  $NIR$ :near infrared. And Leaf Area Index is:

$$\text{LAI} = 4.9 \cdot \text{NDVI} - 0.46 \quad (16)$$

4. `method="MCB"` ([Carrasco-Benavides et al., 2014](#))

$$\text{LAI} = 1.2 - 3.08 \cdot \exp(-2013.35 \cdot \text{NDVI}^{6.41}) \quad (17)$$

5. `method="turner"` ([Turner et al., 1999](#))

$$\text{NDVI} = (\rho_{s,NIR} - \rho_{s,R}) / (\rho_{s,NIR} + \rho_{s,R}) \quad (18)$$

where  $\rho$  is the reflectance at surface level, and the subindex refer to bands *R*:red or *NIR*:near infrared. And Leaf Area Index is:

$$\text{LAI} = 0.5724 + 0.0989 \cdot \text{NDVI} - 0.0114 \cdot \text{NDVI}^2 + 0.0004 \cdot \text{NDVI}^3 \quad (19)$$

### Included models for Land Surface Temperature

1. method="metric" (Allen et al., 2007b) (Landsat 7)

$$\epsilon_{NB} = 0.97 + 0.0033 \cdot \text{LAI} \quad (20)$$

and  $\epsilon_{NB} = 0.98$  when  $\text{LAI} > 3$ ; where  $\epsilon_0$  is the broadband surface emissivity (dimesionless); and  $\text{LAI}$  is the leaf area index ( $m^2 \cdot m^{-2}$ ).

$$T_s = \frac{K_2}{\ln[(\epsilon_{NB} K_1 / R_c) + 1]} \quad (21)$$

where  $T_s$  is the land surface temperature (K);  $K_1$  and  $K_2$  are specific constants for Landsat 7 ( $K_1 = 666.1$  and  $K_2 = 1283 W \cdot m^{-2} \cdot sr^{-1} \cdot \mu m$ ); and  $R_c$  is the corrected thermal radiance from the surface using the spectral radiance from band 6 of Landsat following Wukelic et al. (1989).

2. method="SC" (Jimenez-Munoz et al., 2009, 2014) (Landsat 8)

$$T_s = \gamma \left[ \frac{1}{\epsilon} (v_1 L + v_2) + v_3 \right] + \delta \quad (22)$$

where  $\epsilon$  is the broadband surface emissivity; and  $\gamma$  and  $\delta$  are two parameters given by

$$\gamma = \frac{T^2}{b_\gamma \cdot L} \quad (23)$$

$$\delta = T - \frac{T^2}{b_\gamma} \quad (24)$$

where  $T$  is the at-sensor brightness temperature;  $b_\gamma$  is a coefficient equal to 1324K for L8 band 10, or 1199K for band 11. And  $v_1$ ,  $v_2$  and  $v_3$  are the atmospheric functions, given by

$$v_1 = \frac{1}{\tau_{NB}}; v_2 = -R_{sky} - \frac{R_p}{\tau_{NB}}; v_3 = R_{sky} \quad (25)$$

where  $\tau_{NB}$  is the narrow band transmissivity of air;  $R_{sky}$  is the narrow band downward thermal radiation from a clear sky ( $Wm^{-2}sr^{-1}\mu m^{-1}$ ); and  $R_p$  is path radiance in the 10.4–12.54  $\mu m$  band ( $Wm^{-2}sr^{-1}\mu m^{-1}$ ).

3. method="SW" (Jimenez-Munoz et al., 2014) (Landsat 8)

$$\begin{aligned} T_s = & T_i + 1.378(T_{10} - T_{11}) + 0.183(T_{10} - T_{11})^2 - 0.268 + (54.30 - 2.238w)(1 - \epsilon) \\ & + (-129.20 + 16.40w)\Delta\epsilon \end{aligned} \quad (26)$$

where  $T_s$  is the land surface temperature (K);  $T_{10}$  and  $T_{11}$  are the at-sensor brightness temperatures for bands 10 and 11 of Landsat 8 (K);  $\epsilon$  is the mean emissivity;  $w$  is the total atmospheric water vapor content (in  $g \cdot cm^{-2}$ ) and  $\Delta\epsilon$  is the emissivity difference.

### Included models for Momentum Roughness Length

1. method="short-crops" (Allen et al., 2007b)

$$Z_{om} = 0.018 * \text{LAI} \quad (27)$$

2. `method="custom"` ([Allen et al., 2007b](#))

$$Z_{om} = \exp((a * \text{NDVI}/\text{albedo}) + b) \quad (28)$$

where  $a$  and  $b$  are the regression coefficients derived by adjusting a lineal model between  $\log Z_{om} \sim \text{NDVI}/\text{albedo}$  for points inside the Landsat scene representing specific vegetation types.

3. `method="Perrier"` ([Santos et al., 2012](#); [Pôças et al., 2014](#))

$$Z_{om} = ((1 - \exp(-a * \text{LAI}/2)) * \exp(-a * \text{LAI}/2))^h \quad (29)$$

where  $h$  is the crop height in meters, and  $a$  is:

$$a < -(2 * (1 - fLAI))^{-1}$$

when  $fLAI > 0.5$ , or

$$a < -2 * fLAI$$

when  $fLAI < 0.5$ . And  $fLAI$  is the proportion of LAI lying above  $h/2$ .

*Guillermo Federico Olmedo  
EEA INTA Mendoza  
San Martín 3853, Luján de Cuyo, Mendoza  
Argentina  
[olmedo.guillermo@inta.gob.ar](mailto:olmedo.guillermo@inta.gob.ar)*

*Samuel Ortega-Farías  
Adaptation of Agriculture to Climate Change (A2C2)  
Centro de Investigación y Transferencia en Riego y Agroclimatología  
Universidad de Talca  
2 Norte 685, Talca  
Chile  
[sortega@utalca.cl](mailto:sortega@utalca.cl)*

*Daniel de la Fuente-Sáiz  
Centro de Investigación y Transferencia en Riego y Agroclimatología  
Universidad de Talca  
2 Norte 685, Talca  
Chile  
[ddelafuente@utalca.cl](mailto:ddelafuente@utalca.cl)*

*David Fonseca-Luengo  
Centro de Investigación y Transferencia en Riego y Agroclimatología  
Universidad de Talca  
2 Norte 685, Talca  
Chile  
[daviddtoto.fonseca@gmail.com](mailto:daviddtoto.fonseca@gmail.com)*

*Fernando Fuentes-Peñaillillo  
Centro de Investigación y Transferencia en Riego y Agroclimatología  
Universidad de Talca  
2 Norte 685, Talca  
Chile  
[ffuentesp@utalca.cl](mailto:ffuentesp@utalca.cl)*

# quantreg.nonpar: An R Package for Performing Nonparametric Series Quantile Regression

by Michael Lipsitz, Alexandre Belloni, Victor Chernozhukov, and Iván Fernández-Val

**Abstract** The R package **quantreg.nonpar** implements nonparametric quantile regression methods to estimate and make inference on partially linear quantile models. **quantreg.nonpar** obtains point estimates of the conditional quantile function and its derivatives based on series approximations to the nonparametric part of the model. It also provides pointwise and uniform confidence intervals over a region of covariate values and/or quantile indices for the same functions using analytical and resampling methods. This paper serves as an introduction to the package and displays basic functionality of the functions contained within.

## Introduction: Nonparametric series quantile regression

Let  $Y$  be an outcome variable of interest, and  $X$  a vector of observable covariates. The covariate vector is partitioned as  $X = (W, V)$ , where  $W$  is the key covariate or treatment, and  $V$  is a possibly high dimensional vector with the rest of the covariates that usually play the role of control variables. We can model the  $\tau$ -quantile of  $Y$  conditional on  $X = x$  using the partially linear quantile model

$$Q_{Y|X}(\tau | x) = g(\tau, w) + v'\gamma(\tau), \quad \tau \in [0, 1].$$

Belloni et al. (2011) developed the nonparametric series quantile regression (QR) approximation

$$Q_{Y|X}(\tau | x) \approx Z(x)' \beta(\tau), \quad \beta(\tau) = (\alpha(\tau)', \gamma(\tau)')', \quad Z(x) = (Z(w)', v')',$$

where the unknown function  $g(\tau, w)$  is approximated by a linear combination of series terms  $Z(w)'\alpha(\tau)$ . The vector  $Z(w)$  includes transformations of  $w$  that have good approximation properties such as powers, indicators, trigonometric terms or  $B$ -splines. The function  $\tau \mapsto \alpha(\tau)$  contains quantile-specific coefficients. The **quantreg.nonpar** package implements estimation and inference method for linear functionals of the conditional quantile function based on the series QR approximation. These functionals include:

1. Conditional quantile function itself:  $(\tau, x) \mapsto Q_{Y|X}(\tau | x) \approx Z(x)'\beta(\tau)$ .
2. Partial first and second derivative functions with respect to  $w$ :  
 $(\tau, x) \mapsto \partial^k Q_{Y|X}(\tau | x) / \partial w^k = \partial^k g(\tau, w) / \partial w^k \approx \partial^k Z(w)'\beta(\tau) / \partial w^k, k \in \{1, 2\}$ .
3. Average partial first and second derivative functions with respect to  $w$ :  
 $(\tau) \mapsto \int \partial^k g(\tau, w) / \partial w^k d\mu \approx \int \partial^k Z(w)'\beta(\tau) / \partial w^k d\mu, k \in \{1, 2\}$ , where  $\mu$  is a measure for  $W$ .

Both pointwise or uniform inference over a region of quantile indices and/or covariate values are implemented.

The coefficient vector  $\beta(\tau)$  is estimated using the QR estimator of Koenker and Bassett (1978). Let  $\{(Y_i, X_i) : 1 \leq i \leq n\}$  be a random sample from  $(Y, X)$  and let  $\hat{\beta}(\tau)$  be the QR estimator of  $\beta(\tau)$ , i.e.,

$$\hat{\beta}(\tau) \in \arg \min_{\beta \in \mathbb{R}^m} \sum_{i=1}^n \rho_\tau(Y_i - Z(X_i)'\beta), \quad \tau \in \mathcal{T} \subseteq (0, 1),$$

where  $\rho_\tau(z) = (\tau - 1\{z < 0\})z$  is the check function,  $\mathcal{T}$  is a compact set, and  $m = \dim \beta(\tau)$ . We then construct estimators of the linear functionals of the conditional quantile function by applying the plug-in principle to the series approximations. For example, the series QR quantile estimator of  $Q_{Y|X}(\tau | x)$  is

$$\hat{Q}_{Y|X}(\tau | x) = Z(x)'\hat{\beta}(\tau).$$

A challenge to perform inference in this setting is that  $m$  should increase with the sample size in order to reduce approximation error. Accordingly, the empirical series QR coefficient process  $\tau \mapsto \sqrt{n}(\hat{\beta}(\tau) - \beta(\tau))$  has increasing dimension with  $n$  and therefore does not have a limit distribution. Belloni et al. (2011) dealt with this problem by deriving two couplings or strong approximations to  $\tau \mapsto \sqrt{n}(\hat{\beta}(\tau) - \beta(\tau))$ . A coupling is a construction of two processes on the same probability space that are uniformly close to each other with high probability. In this case, Belloni et al. (2011)

constructed a pivotal process and a Gaussian process of dimension  $m$  that are uniformly close to  $\tau \mapsto \sqrt{n}(\hat{\beta}(\tau) - \beta(\tau))$ . They also provided four methods to estimate the distribution of these coupling processes that can be used to make inference on linear functionals of the conditional quantile function:

1. Pivotal: analytical method based on the pivotal coupling.
2. Gradient bootstrap: resampling method based on the pivotal coupling.
3. Gaussian: analytical method based on the Gaussian coupling.
4. Weighted bootstrap: resampling method based on the Gaussian coupling.

The **quantreg.nonpar** package implements all these methods.

Additionally, the linear functionals of interest might be naturally monotone in some of their arguments. For example, the conditional quantile function  $\tau \mapsto Q_{Y|X}(\tau | x)$  is increasing, and in the growth chart application of the next section the conditional quantile function of height,  $(\tau, x) \mapsto Q_{Y|X}(\tau | x)$ , is increasing with respect to both the quantile index,  $\tau$ , and the treatment age,  $w$ . The series QR estimates might not satisfy this logical monotonicity restriction giving rise to the so-called quantile crossing problem in the case of  $\tau \mapsto \hat{Q}_{Y|X}(\tau | x)$ . The **quantreg.nonpar** package deals with the quantile crossing and other non monotonicity problems in the estimates of the linear functionals by applying the rearrangement method of Chernozhukov et al. (2009) and Chernozhukov et al. (2010).

## Related R packages

Several existing R packages are available to estimate conditional quantile models. The package **quantreg** (Koenker, 2016) includes multiple commands for parametric and nonparametric quantile regression. The command `rqss` estimates univariate and bivariate local nonparametric smoothing splines, and the command `rearrange` implements the rearrangement method to tackle the quantile crossing problem. The package **QuantifQuantile** (Charlier et al., 2015) estimates univariate conditional quantile models using a local nonparametric method called optimal quantization or partitioning. The nonparametric methods implemented in the previous packages are local or kernel-type, whereas our methods are global or series-type. Finally, the command `gcrq` in the package **quantregGrowth** (Muggeo et al., 2013) implements a univariate B-spline global nonparametric method with a penalty to deal with the quantile crossing and to impose monotonicity with respect to the covariate. To our knowledge, no existing R package allows the user to perform uniform nonparametric inference on linear functionals of the conditional quantile function over a region of quantile indices and/or covariate values, making **quantreg.nonpar** the first package to do so.

## The package **quantreg.nonpar**

### Model specification

We illustrate the functionality of the package with an empirical application based on data from Koenker (2011) for childhood malnutrition in India, where we model the effect of a child's age and other covariates on the child's height. Here,  $Y$  is the child's height in centimeters;  $W$  is the child's age in months; and  $V$  is a vector of 22 controls. These controls include the mother's body mass index (BMI), the number of months the child was breastfed, and the mother's age (as well as the square of the previous three covariates); the mother's years of education and the father's years of education; dummy variables for the child's sex, whether the child was a single birth or multiple birth, whether or not the mother was unemployed, whether the mother's residence is urban or rural, and whether the mother has each of: electricity, a radio, a television, a refrigerator, a bicycle, a motorcycle, and a car; and factor variables for birth order of the child, the mother's religion and quintiles of wealth.

First, we load the data and construct the variables that will be used in the analysis. Note that the variable prefixes "c" and "m" refer to "child" and "mother". For each factor variable (`csex`, `ctwin`, `cbirthorder`, `munemployed`, `mreligion`, `mresidence`, `wealth`, `electricity`, `radio`, `television`, `refrigerator`, `bicycle`, `motorcycle`, and `car`), we generate a variable "facvar" which is the factor version of the variable "var". For each quadratic variable (`mbmi`, `breastfeeding`, and `mage`), we generate a variable "varsq" which is the variable squared. For example:

```
R> data <- india
R> faccsex <- factor(csex)
R> mbmisq <- mbmi^2
```

We also construct the formula to be used for the linear part of the model,  $v'\gamma(\tau)$ :

```
R> form.par <- cheight ~ mbmi + mbmisq + breastfeeding + breastfeedingsq + mage +
+   magesq + medu + edupartner + faccsex + facctwin + faccbirthorder +
+   facmunemployed + facmreligion + facmresidence + facwealth + facelectricity +
+   facradio + factelevision + facrefrigerator + facbicycle + facmotorcycle + faccar
```

Note that this formula does not contain a term for our variable of interest  $W$ ; namely, the child's age. Let us now construct the nonparametric bases that will be used to estimate the effect of  $W$ , i.e.,  $g(\tau, w) \approx Z(w)'\alpha(\tau)$ . For our base case, we construct a cubic  $B$ -spline basis with knots at the  $\{0, 0.1, 0.2, \dots, 0.9, 1\}$  quantiles of the observed values of child's age.

```
R> basis.bsp <- create.bspline.basis(breaks = quantile(cage, c(0:10)/10))
```

Finally, we set the values of some of the other parameters. For the purposes of this example, we use 500 simulations for the pivotal and Gaussian methods, and 100 repetitions for the weighted and gradient bootstrap methods. The set of analyzed quantile indices will be  $\{0.04, 0.08, \dots, 0.96\}$ , but we will have npqr print only results for quantile indices contained in the set  $\{0.2, 0.4, 0.6, 0.8\}$ . Finally, we will use  $\alpha = 0.05$  as the significance level for the confidence intervals (i.e., the confidence level is 0.95).

```
R> B <- 500
R> B.boot <- 100
R> taus <- c(1:24)/25
R> print.taus <- c(1:4)/5
R> alpha <- 0.05
```

### Comparison of the inference processes

Initially, we will focus on the average growth rate, i.e., the average first derivative of the conditional quantile function with respect to child's age

$$\tau \mapsto \int \partial_w g(\tau, w) d\mu(w), \quad \tau \in \mathcal{T},$$

where  $\mu$  is a measure for  $W$  and  $\mathcal{T}$  is the set of quantile indices of interest specified with taus. We specify the average first derivative with the options nnderivs = 1 and average = 1. Inference will be performed uniformly over  $\mathcal{T}$ , and the standard errors will be computed unconditionally for the pivotal and Gaussian processes; see Section [Confidence intervals and standard errors](#).

We first construct the four inference processes based on the  $B$ -spline basis. By default, npqr generates output similar to that seen below. In this example, output is suppressed in each call following the first. Instead of invoking a particular process, we may also set process = "none". In that case, inference will not be performed, and only point estimates will be reported.

```
R> piv.bsp <- npqr(formula = form.par, basis = basis.bsp, var = "cage", taus = taus,
+   nnderivs = 1, average = 1, print.taus = print.taus, B = B, uniform = TRUE)
R> gaus.bsp <- update(piv.bsp, process = "gaussian", printOutput = FALSE)
R> wboot.bsp <- update(gaus.bsp, process = "wbootstrap", B = B.boot)
R> gboot.bsp <- update(wboot.bsp, process = "gbootstrap")
```

The output for the pivotal method (which is generated whenever printOutput = TRUE) is given in Figure 1.

The point estimates represent the average derivative of the conditional quantile function with respect to the variable of interest: the child's age. In other words, each value represents the average rate of growth (in centimeters per month) at each quantile of the height distribution. They are reported, along with their standard errors and respective two-sided and one-sided confidence intervals, at each quantile for which output was requested using print.taus. The null hypotheses on which hypothesis testing is performed state that the average growth rate is negative, positive, and equal to zero, respectively, at all quantiles of the distribution. We reject, at the 5% level, the null hypotheses that the growth rate is negative and that the growth rate is equal to zero. We cannot reject, at the 5% level, the null hypothesis that the growth rate is positive.

Additionally, the following results are saved in piv.bsp:

- **piv.bsp\$CI:** a  $1 \times \text{length}(\text{taus}) \times 2$  array: each pair is the lower and upper bounds of the 95% confidence interval for the average derivative of the conditional quantile function at each quantile index in taus.
- **piv.bsp\$CI.oneSided:** a  $1 \times \text{length}(\text{taus}) \times 2$  array: each pair contains bounds for two separate one-sided 95% confidence intervals (a lower bound and an upper bound, respectively) for the average derivative of the conditional quantile function at each quantile index in taus.

```

Method: pivotal
Standard error estimation is unconditional.

No. of obs.: 37623
No. of simulations or bootstrap repetitions: 500

-----
Quantile Estimates & Inference
-----


| Quantile | Point Estimate | Standard Error | Uniform            |        | One-sided 95% CIs |  |
|----------|----------------|----------------|--------------------|--------|-------------------|--|
|          |                |                | 95% Conf. Interval | Lower  | Upper             |  |
| 0.2      | 0.7676         | 0.008542       | 0.7439 0.7913      | 0.7461 | 0.7891            |  |
| 0.4      | 0.7761         | 0.00751        | 0.7553 0.797       | 0.7572 | 0.795             |  |
| 0.6      | 0.7967         | 0.0077         | 0.7754 0.8181      | 0.7774 | 0.8161            |  |
| 0.8      | 0.8225         | 0.008492       | 0.799 0.8461       | 0.8011 | 0.8439            |  |

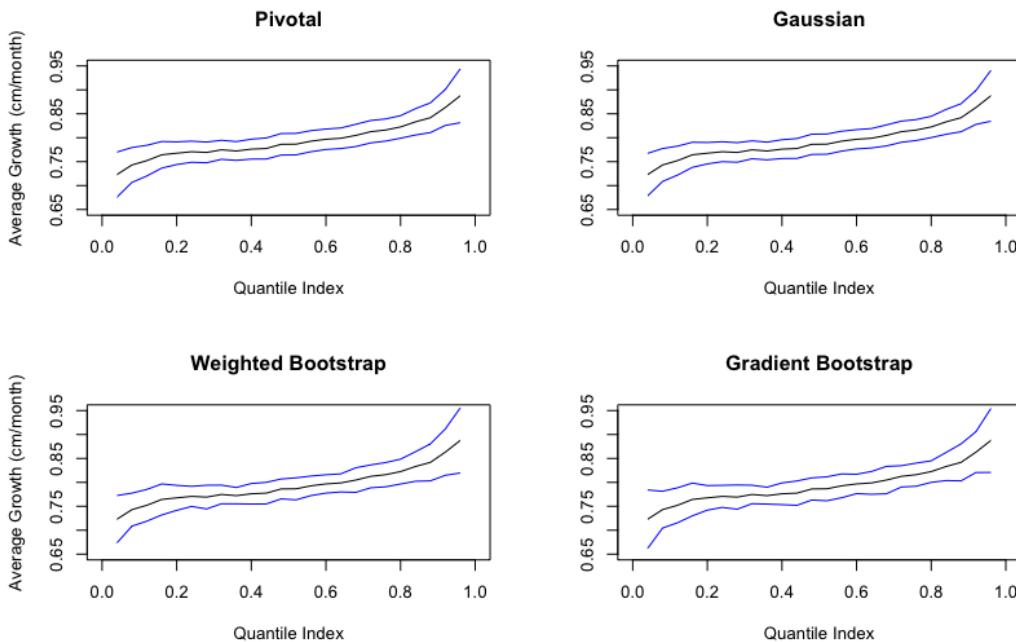

-----
Hypothesis Testing
-----
Null Hypothesis: p-value
=====
theta(tau, w) <= 0 for all tau, w 0
theta(tau, w) >= 0 for all tau, w 1
theta(tau, w) = 0 for all tau, w 0.02642
=====
```

Figure 1: Output for the pivotal method.

- `piv.bsp$point.est`: a  $1 \times \text{length}(\text{taus})$  matrix: each entry is the point estimate for the average derivative of the conditional quantile function at each quantile index in `taus`.
- `piv.bsp$std.error`: a  $1 \times \text{length}(\text{taus})$  matrix: each entry is the standard error of the estimator of the average derivative of the conditional quantile function at each quantile index in `taus` (here, unconditional on the sample).
- `piv.bsp$pvalues`: a three item vector containing the  $p$ -values reported above: the first tests the null hypothesis that the average derivative is less than zero everywhere (at each quantile index in `taus`); the second tests the null hypothesis that the average derivative is everywhere greater than zero; the third tests the null hypothesis that the average derivative is everywhere equal to zero.
- `piv.bsp$taus`: the input vector `taus`, i.e.,  $\{0.04, 0.08, \dots, 0.96\}$ .
- `piv.bsp$coefficients`: a list of length `length(taus)`: each element of the list contains the estimates of the QR coefficient vector  $\beta(\tau)$  at the corresponding quantile index.
- `piv.bsp$var.unique`: a vector containing all values of the covariate of interest, `W`, with no repeated values.
- `piv.bsp$load`: the input vector or matrix `load`. If `load` is not input (as in this case), the output `load` is generated based on `average` and `nnderivs`. Here, it is a vector containing the average value of the derivative of the regression equation with respect to the variable of interest, not including the estimated coefficients.

Using `piv.bsp$taus`, `piv.bsp$CI`, and `piv.bsp$point.est`, as well as the corresponding objects for the Gaussian, weighted bootstrap, and gradient bootstrap methods, we construct plots containing the estimated average quantile derivatives, as well as 95% uniform confidence bands over the quantile indices in `taus`:

```
R> par(mfrow = c(2, 2))
R> yrange <- c(.65, .95)
R> xrange <- c(0, 1)
R> plot(xrange, yrange, type = "n", xlab = "Quantile Index",
+       ylab = "Average Growth (cm/month)", ylim = yrange)
R> lines(piv.bsp$taus, piv.bsp$point.est)
R> lines(piv.bsp$taus, piv.bsp$CI[, , 1], col = "blue")
```



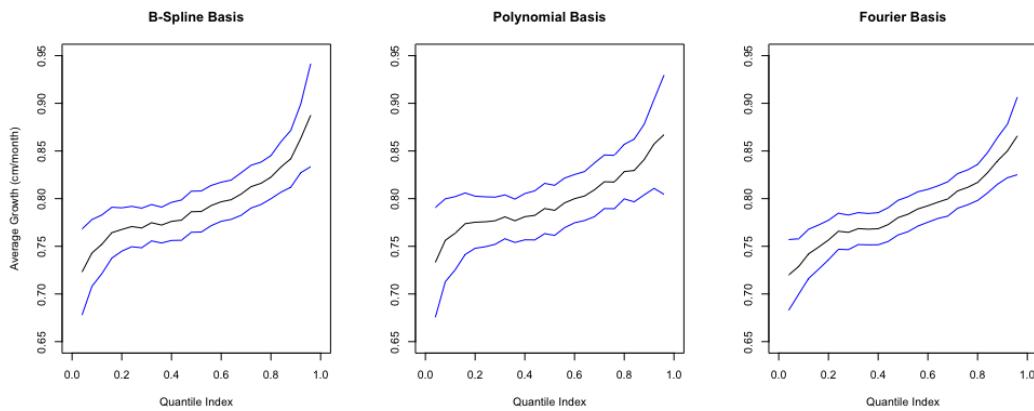
**Figure 2:** Comparison of inference methods for growth rate: Point estimates and 95% uniform confidence bands for the average derivative of the conditional quantile function of height with respect to age based on B-spline series approximation.

```
R> lines(piv.bsp$taus, piv.bsp$CI[, 2], col = "blue")
R> title("Pivotal")
R> plot(xrange, yrangle, type = "n", xlab = "Quantile Index", ylab = "", ylim = yrangle)
R> lines(gaus.bsp$taus, gaus.bsp$point.est)
R> lines(gaus.bsp$taus, gaus.bsp$CI[, 1], col="blue")
R> lines(gaus.bsp$taus, gaus.bsp$CI[, 2], col="blue")
R> title("Gaussian")
R> plot(xrange, yrangle, type = "n", xlab = "Quantile Index",
+       ylab = "Average Growth (cm/month)", ylim = yrangle)
R> lines(wboot.bsp$taus, wboot.bsp$point.est)
R> lines(wboot.bsp$taus, wboot.bsp$CI[, 1], col = "blue")
R> lines(wboot.bsp$taus, wboot.bsp$CI[, 2], col = "blue")
R> title("Weighted Bootstrap")
R> plot(xrange, yrangle, type = "n", xlab = "Quantile Index", ylab = "", ylim = yrangle)
R> lines(gboot.bsp$taus, gboot.bsp$point.est)
R> lines(gboot.bsp$taus, gboot.bsp$CI[, 1], col = "blue")
R> lines(gboot.bsp$taus, gboot.bsp$CI[, 2], col = "blue")
R> title("Gradient Bootstrap")
R> title("Average Growth Rate with 95% CI", outer = TRUE)
```

As we can see in Figure 2, the confidence bands generated are roughly similar. Note that the point estimates are the same for all the methods.

We can compare the computation times of each of the approximations using the command `system.time`. Additionally, we compare the *p*-values generated by each of the four inference methods. Note that computation times may vary widely depending on the machine in use. However, the relative computation times will be approximately constant across different machines. The computation times in the table below were obtained on a computer with two eight-core 2.6 GHz processors (note: `npqr` does not make use of parallel computing).

```
R> pval.dimnames <- vector("list", 2)
R> pval.dimnames[[1]] <- c("Pivotal", "Gaussian", "Weighted Bootstrap",
+   "Gradient Bootstrap")
R> pval.dimnames[[2]] <- c("H0: Growth Rate <= 0", "H0: Growth Rate >= 0",
+   "H0: Growth Rate = 0", "Computation Minutes")
R> pvals <- matrix(NA, nrow = 4, ncol = 4, dimnames = pval.dimnames)
R> pvals[1, ] <- c(round(piv.bsp$pvalues, digits = 4), round(piv.time, digits = 0))
R> pvals[2, ] <- c(round(gaus.bsp$pvalues, digits = 4), round(gaus.time, digits = 0))
```



**Figure 3:** Comparison of series bases for growth rate: Point estimates and 95% uniform confidence bands for the average derivative of the conditional quantile function of height with respect to age based on *B*-spline, polynomial, and Fourier series approximations.

```
R> pvals[3, ] <- c(round(wboot.bsp$pvalues, digits = 4), round(wboot.time, digits = 0))
R> pvals[4, ] <- c(round(gboot.bsp$pvalues, digits = 4), round(gboot.time, digits = 0))
R> pvals
```

	H0: Growth Rate <= 0	H0: Growth Rate >= 0	H0: Growth Rate = 0
Pivotal	0	1	0.0237
Gaussian	0	1	0.0234
Weighted Bootstrap	0	1	0.0221
Gradient Bootstrap	0	1	0.0221
Computation Minutes			
Pivotal	0.9		
Gaussian	0.6		
Weighted Bootstrap	30.0		
Gradient Bootstrap	346.0		

As expected, we reject at the 5% level the null hypothesis that the growth rate is negative and the null hypothesis that the growth rate is equal to zero in all cases, and we fail to reject the null hypothesis that the growth rate is positive in all cases. For the one-sided tests, the relevant null hypothesis is that the average growth rate is less than or equal to zero (greater than or equal to zero) at all the quantile indices in *taus*. For the two-sided test, the relevant null hypothesis is that the average growth rate is equal to zero at all the quantile indices in *taus*. Additionally, note that the pivotal and Gaussian methods are substantially faster than the two bootstrap methods.

### Comparison of series bases

Another option is to take advantage of the variety of bases available in the **quantreg.nonpar** package. Here, we consider three bases: the *B*-spline basis used in the analysis above, an orthogonal polynomial basis of degree 12, and a Fourier basis with 9 basis functions and a period of 200 months. We compare the estimates of the average quantile derivative function generated by using each of these bases. We construct the orthogonal polynomial basis and the Fourier basis with the commands:

```
R> basis.poly <- poly(cage, degree = 12)
R> basis.four <- create.fourier.basis(rangeval = range(data$cage), nbasis = 9,
+ period = 200)
```

In this section, we focus on the pivotal method for inference. We run *npqr* for the orthogonal polynomial basis and the Fourier basis, mimicking the analysis run above for the *B*-spline basis.

```
R> piv.poly <- update(piv.bsp, basis = basis.poly)
R> piv.four <- update(piv.bsp, basis = basis.four)
```

Similar to Section [Comparison of the inference processes](#), we plot the point estimates with their uniform 95% confidence bands for each basis. Figure 3 shows that, given the parameters of the chosen bases, the type of basis does not have an important impact on the estimation and inference on the growth rate charts. A table containing the *p*-values associated with the hypothesis tests for each basis are generated by the following code:

```
R> pval2.dimnames <- vector("list", 2)
R> pval2.dimnames[[1]] <- c("B-spline", "Polynomial", "Fourier")
R> pval2.dimnames[[2]] <- c("H0: Growth Rate <= 0", "H0: Growth Rate >= 0",
+   "H0: Growth Rate = 0")
R> pvals2 <- matrix(NA, nrow = 3, ncol = 3, dimnames = pval2.dimnames)
R> pvals2[1, ] <- round(piv.bsp$pvalues, digits = 4)
R> pvals2[2, ] <- round(piv.poly$pvalues, digits = 4)
R> pvals2[3, ] <- round(piv.four$pvalues, digits = 4)
R> pvals2

      H0: Growth Rate <= 0 H0: Growth Rate >= 0 H0: Growth Rate = 0
B-spline          0           1        0.0239
Polynomial        0           1        0.0334
Fourier          0           1        0.0386
```

For all bases, the tests' conclusions are identical: at the 5% level, we reject the null hypothesis that the average growth rate is negative, fail to reject the null hypothesis that the average growth rate is positive, and reject the null hypothesis that the average growth rate is equal to zero.

### Confidence intervals and standard errors

Now, we illustrate two additional options available to the user. First, to perform inference pointwise over a region of covariate values and/or quantile indices instead of uniformly, and second, to estimate the standard errors conditional on the values of the covariate  $W$  in the sample. When inference is uniform, the test statistic used in construction of the confidence interval is the maximal  $t$ -statistic across all covariate values and quantile indices in the region of interest, whereas pointwise inference uses the  $t$ -statistic at each covariate value and quantile index. When standard errors are estimated unconditionally, a correction term is used to account for the fact that the empirical distribution of  $W$  is an estimator of the distribution of  $W$ . The option to estimate standard errors conditionally or unconditionally is not available for the bootstrap methods. The inference based on these methods is always unconditional.

We will use only the pivotal method with a  $B$ -spline basis for this illustration. First, we run npqr for each combination of options mentioned above:

```
R> piv.bsp <- npqr(formula = form.par, basis = basis.bsp, var = "cage", taus = taus,
+   B = B, nderivs = 1, average = 1, alpha = alpha, process = "pivotal",
+   uniform = TRUE, se = "unconditional", printOutput = FALSE)
R> piv.bsp.cond <- update(piv.bsp, se = "conditional")
R> piv.bsp.point <- update(piv.bsp, uniform = FALSE, se = "unconditional")
R> piv.bsp.point.cond <- update(piv.bsp, uniform = FALSE, se = "conditional")
```

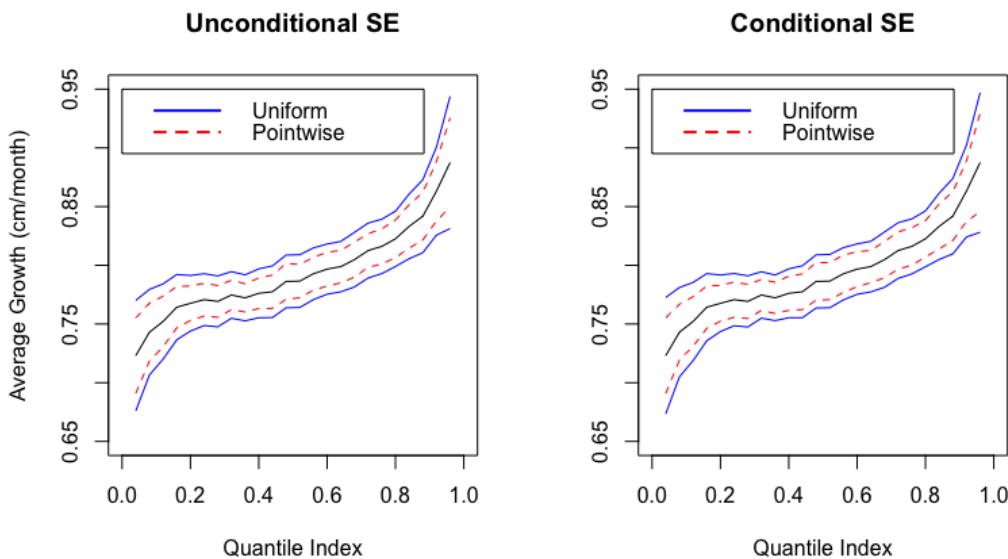
We obtain Figure 4 using the graphing techniques described in Sections [Comparison of the inference processes](#) and [Comparison of series bases](#). As is visible in this figure, usage of conditional standard errors changes the confidence bands only minimally in our example. As expected, the pointwise confidence bands are narrower than the uniform confidence bands.

We can also compare how much of the differences (or lack thereof) in the confidence bands are driven by differences in the standard errors versus the test statistics. Here, we compare the estimated standard errors at the median for conditional versus unconditional inference:

```
R> piv.bsp.med <- npqr(formula = form.par, basis = basis.bsp, var = "cage", taus = 0.5,
+   B = B, nderivs = 1, average = 1, alpha = alpha, process = "pivotal", uniform = TRUE,
+   se = "unconditional", printOutput = FALSE)
R> piv.bsp.cond.med <- update(piv.bsp.med, se = "conditional")
R> stderr.dimnames <- vector("list", 2)
R> stderr.dimnames[[1]] <- c("Unconditional", "Conditional")
R> stderr.dimnames[[2]] <- c("Standard Error")
R> stderr <- matrix(NA, nrow = 2, ncol = 1, dimnames = stderr.dimnames)
R> stderr[1, ] <- piv.bsp.med$std.error[1]
R> stderr[2, ] <- piv.bsp.cond.med$std.error[1]
R> stderr
```

	Standard Error
Unconditional	0.008104
Conditional	0.007663

Finally, we compare  $p$ -values generated by each of the option choices:



**Figure 4:** Comparison of pointwise vs. uniform and conditional vs. unconditional inference for growth rate: 95% uniform and pointwise confidence bands for the average derivative of the conditional quantile function of height with respect to age based on *B*-spline series approximation. The left panel uses unconditional standard errors in the construction of the bands. The right panel uses conditional standard errors.

```
R> pval3.dimnames <- vector("list", 2)
R> pval3.dimnames[[1]] <- c("Uniform", "Unconditional", "Uniform", "Conditional",
+   "Pointwise", "Unconditional", "Pointwise", "Conditional")
R> pval3.dimnames[[2]] <- c("H0: Growth Rate <= 0", "H0: Growth Rate >= 0",
+   "H0: Growth Rate = 0")
R> pvals3 <- matrix(NA, nrow = 4, ncol = 3, dimnames = pval3.dimnames)
R> pvals3[1, ] <- round(piv.bsp$pvalues, digits = 4)
R> pvals3[2, ] <- round(piv.bsp$cond$pvalues, digits = 4)
R> pvals3[3, ] <- round(piv.bsp$point$pvalues, digits = 4)
R> pvals3[4, ] <- round(piv.bsp$point$cond$pvalues, digits = 4)
R> pvals3

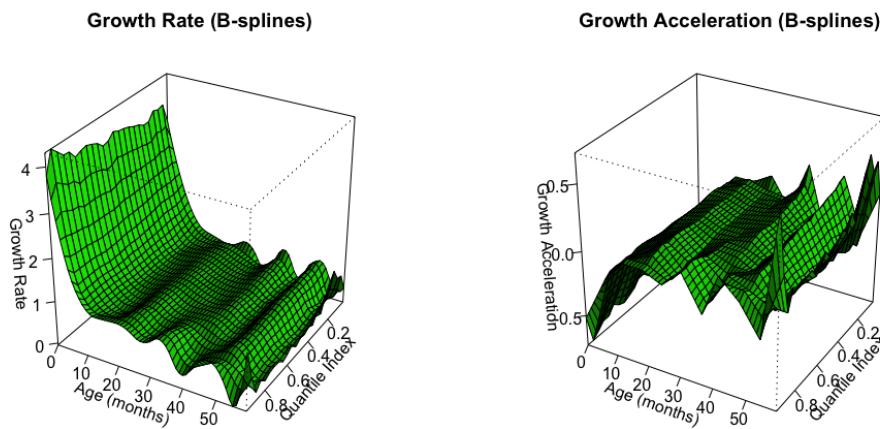
          H0: Growth Rate <= 0 H0: Growth Rate >= 0 H0: Growth Rate = 0
Uniform, Unconditional           0             1            0.0239
Uniform, Conditional            0             1            0.0243
Pointwise, Unconditional         0             1            0.0267
Pointwise, Conditional          0             1            0.0222
```

In this example, where the sample size is large, about 38,000 observations, conditional versus unconditional standard errors and uniform versus pointwise inference have little impact on the estimated *p*-values.

### Estimation and uniform inference on linear functionals

Finally, we illustrate how to estimate and make uniform inference on linear functionals of the conditional quantile function over a region of covariate values and quantile indices. These functionals include the function itself and derivatives with respect to the covariate of interest. The **quantreg.nonpar** package is able to perform estimation and inference on the conditional quantile function, its first derivative, and its second derivative over a region of covariate values and/or quantile indices. We also illustrate how to report the estimates using three dimensional plots.

First, we consider the first and second derivatives of the conditional quantile function. In the application they correspond to the growth rate and growth acceleration of height with respect to age as a function of age (from 0 to 59 months) and the quantile index. To do so, we use the output of npqr called var.unique, which contains a vector with all the distinct values of the covariate of interest (cage here). To generate this output, we estimate the first and second derivatives of the conditional quantile function using a *B*-spline series approximation over the covariate values in var.unique and the quantile indices in taus:



**Figure 5:** Growth rate and acceleration: Estimates of the first and second derivatives of the conditional quantile function of height with respect to age.

```
R> piv.bsp.firstderiv <- npqr(formula = form.par, basis = basis.bsp, var = "cage",
+   taus = taus, nderivs = 1, average = 0, print.taus = print.taus, B = B,
+   process = "none", printOutput = FALSE)
R> piv.bsp.secondderiv <- update(piv.bsp.firstderiv, nderivs = 2)
```

Next, we generate vectors containing the region of covariate values and quantile indices of interest:

```
R> xsurf1 <- as.vector(piv.bsp.firstderiv$taus)
R> ysurf1 <- as.vector(piv.bsp.firstderiv$var.unique)
R> zsurf1 <- t(piv.bsp.firstderiv$point.est)
R> xsurf2 <- as.vector(piv.bsp.secondderiv$taus)
R> ysurf2 <- as.vector(piv.bsp.secondderiv$var.unique)
R> zsurf2 <- t(piv.bsp.secondderiv$point.est)
```

Finally, we create the three dimensional plots for:

$$(\tau, w) \mapsto \partial^k g(\tau, w) / \partial w^k, (\tau, w) \in I,$$

where  $k \in \{1, 2\}$ , and  $I$  is the region of interest.

```
R> par(mfrow = c(1, 2))
R> persp(xsurf1, ysurf1, zsurf1, xlab = "Quantile Index", ylab = "Age (months)",
+   zlab = "Growth Rate", ticktype = "detailed", phi = 30, theta = 120, d = 5,
+   col = "green", shade = 0.75, main = "Growth Rate (B-splines)")
R> persp(xsurf2, ysurf2, zsurf2, xlab = "Quantile Index", ylab = "Age (months)",
+   zlab = "Growth Acceleration", ticktype = "detailed", phi = 30, theta = 120,
+   d = 5, col = "green", shade = 0.75, main = "Growth Acceleration (B-splines)")
```

These commands produce Figure 5. Here, we see that the growth rate is positive at all ages and quantile indices. The growth rate decreases in the first few months of life and stabilizes afterwards, which can also be seen in the graph of growth acceleration. Growth acceleration is negative at young ages but stabilizes around zero at about 15 months. Both growth rate and growth acceleration are relatively homogeneous across quantiles at all ages. Saved in `piv.bsp.firstderiv$pvalues` and `piv.bsp.secondderiv$pvalues` are the  $p$ -values from hypothesis tests to determine whether the first and second derivatives, respectively, are negative, positive, and equal to zero uniformly over the region of ages and quantile indices:

Order of Derivative	H0: Growth Rate $\leq 0$	H0: Growth Rate $\geq 0$	H0: Growth Rate $= 0$
First Derivative	0	1	0.042
Second Derivative	0	1	0.061

Thus, we reject at the 5% level the null hypotheses that growth rate is negative, that growth rate is equal to zero, and that growth acceleration is positive over all the first five years of the children's lives at all the quantiles of interest. We come close to rejecting at the 5% level the null hypothesis that growth acceleration is equal to zero over all the first five years of the children's lives at all the quantiles of interest.

Similarly, we estimate the conditional quantile function over a region of covariate values and quantile indices, which corresponds to a growth chart in our application. Here, we use a fully saturated indicator basis for the series approximation to the nonparametric part of the model. We also compare the original estimates of the resulting growth chart to rearranged estimates that impose that the conditional quantile function of height is monotone in age and the quantile index. In this example, the conditional quantile function estimated using all data is nearly monotone without rearrangement. To illustrate the power of rearrangement when estimates are not monotone, we use a subset of the data containing the first 1,000 observations:

```
R> data.subset <- data[1:1000, ]
R> detach(data)
R> attach(data.subset)
```

Now, we create the fully saturated indicator basis for cage:

```
R> faccage <- factor(cage)
```

To perform estimation using this basis, we input faccage for basis:

```
R> piv.fac.fun <- npqr(formula = form.par, basis = faccage, var = "cage", taus = taus,
+   print.taus = print.taus, B = B, nnderivs = 0, average = 0, alpha = alpha,
+   process = "none", rearrange = FALSE, rearrange.vars = "both", se = "conditional",
+   printOutput = FALSE, method = "fn")
```

We also obtain the rearranged estimates with respect to age and the quantile index using the options of the command `npqr`. Note that we input "both" for `rearrange.vars`. This option performs rearrangement over quantile indices and age. Other allowable options are "quantile" (for monotonization over quantile indices only) and "var" (for monotonization over the variable of interest only).

```
R> piv.fac.fun.re <- update(piv.fac.fun, rearrange.vars = "both")
```

Now, we construct three dimensional plots for the estimates of the conditional quantile function:

$$(\tau, w) \mapsto Q_{Y|X}(\tau | x) = g(\tau, w) + v'\gamma(\tau), \quad (\tau, w) \in I,$$

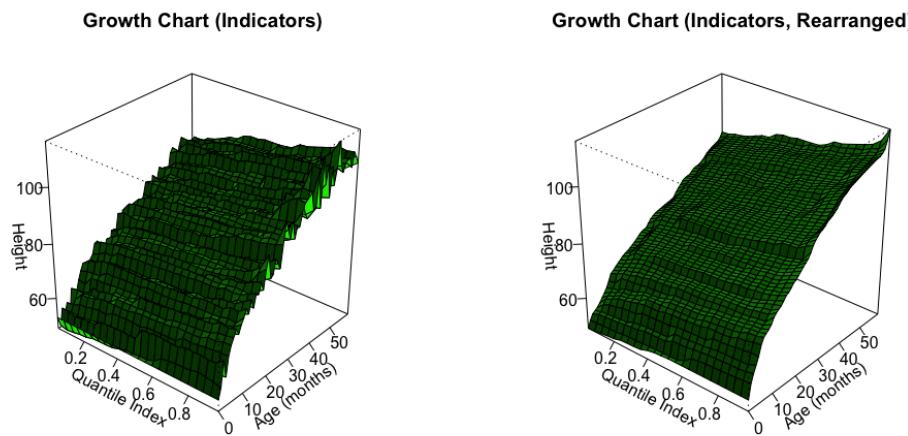
where  $v$  are evaluated at the sample mean for cardinal variables (`mbmi`, `breastfeeding`, `mage`, `medu`, and `edupartner`) and the sample mode for unordered factor variables (`faccsex`, `facctwin`, `faccbirthorder`, `facmunemployed`, `facmreligion`, `facmresidence`, `facwealth`, `facelectricity`, `farcradio`, `factetelevision`, `facrefrigerator`, `facbicycle`, `facmotorcycle`, and `faccar`).

```
R> xsurf <- as.vector(piv.fac.fun$taus)
R> ysurf <- as.vector(piv.fac.fun$var.unique)
R> zsurf.fac <- t(piv.fac.fun$point.est)
R> zsurf.fac.re <- t(piv.fac.fun.re$point.est)
R> par(mfrow = c(1, 2))
R> persp(xsurf, ysurf, zsurf.fac, xlab = "Quantile Index", ylab = "Age (months)",
+   zlab = "Height", ticktype = "detailed", phi = 30, theta = 40, d = 5,
+   col = "green", shade = 0.75, main = "Growth Chart (Indicators)")
R> persp(xsurf, ysurf, zsurf.fac.re, xlab = "Quantile Index", ylab = "Age (months)",
+   zlab = "Height", ticktype = "detailed", phi = 30, theta = 40, d = 5,
+   col = "green", shade = 0.75, main = "Growth Chart (Indicators, Rearranged)")
```

Figure 6 shows that the rearrangement fixes the non-monotonic areas of the original estimates.

## Conclusion

In this paper we introduced the R package `quantreg.nonpar`, which implements the methods of Belloni et al. (2011) to estimate and make inference on partially linear quantile models. The package allows the user to obtain point estimates of the conditional quantile function and its derivatives based on a nonparametric series QR approximation. Using pivotal, gradient bootstrap, Gaussian, and a weighted bootstrap methods, the user is also able to obtain pointwise and uniform confidence intervals. We apply the package to a dataset containing information on child malnutrition in India, illustrating the ability of `quantreg.nonpar` to generate point estimates and confidence intervals, as well as output that allows for easy visualization of the computed values. We also illustrate the ability of the package to monotonize estimates by the variable of interest and by quantile index.



**Figure 6:** Growth chart, with and without rearrangement: Estimates of the conditional quantile function of height based on a fully saturated indicator approximation with respect to age.

## Acknowledgments

We wish to thank Jim Ramsay for assistance with the `fda` package (Ramsay et al., 2014), Roger Koenker for sharing the data used in Koenker (2011), and an anonymous referee for insightful comments. We gratefully acknowledge research support from the NSF.

## Bibliography

- A. Belloni, V. Chernozhukov, D. Chetverikov, and I. Fernandez-Val. Conditional quantile processes based on series or many regressors. *ArXiv e-prints*, 2011. URL <https://arxiv.org/abs/1105.6154>. [p371, 380]
- I. Charlier, D. Paindaveine, and J. Saracco. *QuantifQuantile: Estimation of Conditional Quantiles Using Optimal Quantization*, 2015. URL <https://CRAN.R-project.org/package=QuantifQuantile>. R package version 2.2. [p372]
- V. Chernozhukov, I. Fernández-Val, and A. Galichon. Improving point and interval estimators of monotone functions by rearrangement. *Biometrika*, 96(3):559–575, 2009. doi: 10.1093/biomet/asp030. [p372]
- V. Chernozhukov, I. Fernández-Val, and A. Galichon. Quantile and probability curves without crossing. *Econometrica*, 78(3):1093–1125, 2010. doi: 10.3982/ecta7880. [p372]
- R. Koenker. Additive models for quantile regression: Model selection and confidence bandaids. *Brazilian Journal of Probability and Statistics*, 25(3):239–262, 2011. doi: 10.1214/10-bjps131. [p372, 381]
- R. Koenker. *quantreg: Quantile Regression*, 2016. URL <https://CRAN.R-project.org/package=quantreg>. R package version 5.21. [p372]
- R. Koenker and G. Bassett. Regression quantiles. *Econometrica*, 46(1):33–50, 1978. doi: 10.2307/1913643. [p371]
- V. M. R. Muggeo, M. Sciandra, A. Tomasello, and S. Calvo. Estimating growth charts via nonparametric quantile regression: A practical framework with application in ecology. *Environmental and Ecological Statistics*, 20:519–531, 2013. doi: 10.1007/s10651-012-0232-1. [p372]
- J. O. Ramsay, H. Wickham, S. Graves, and G. Hooker. *fda: Functional Data Analysis*, 2014. URL <https://CRAN.R-project.org/package=fda>. R package version 2.4.4. [p381]

Michael Lipsitz  
Boston University  
Department of Economics  
USA  
[lipsitzm@bu.edu](mailto:lipsitzm@bu.edu)

*Alexandre Belloni*  
*Duke University*  
*Fuqua School of Business*  
*USA*  
[abn5@duke.edu](mailto:abn5@duke.edu)

*Victor Chernozhukov*  
*MIT*  
*Department of Economics*  
*USA*  
[vchern@mit.edu](mailto:vchern@mit.edu)

*Iván Fernández-Val*  
*Boston University*  
*Department of Economics*  
*USA*  
[ivanf@bu.edu](mailto:ivanf@bu.edu)

# nmfgpu4R: GPU-Accelerated Computation of the Non-Negative Matrix Factorization (NMF) Using CUDA Capable Hardware

by Sven Koitka and Christoph M. Friedrich

**Abstract** In this work, a novel package called **nmfgpu4R** is presented, which offers the computation of *Non-negative Matrix Factorization (NMF)* on *Compute Unified Device Architecture (CUDA)* platforms within the R environment. Benchmarks show a remarkable speed-up in terms of time per iteration by utilizing the parallelization capabilities of modern graphics cards. Therefore the application of NMF gets more attractive for real-world sized problems because the time to compute a factorization is reduced by an order of magnitude.

## Introduction

Dimension reduction techniques are commonly used in machine learning and data mining tasks. For instance in text mining a corpora with thousands of words in the vocabulary could be too complex to be learned by *Support Vector Machines (SVM)* directly. Therefore the most important structure within the data must be extracted prior to the learning process. In the context of text mining new data axes at best represent topics in the corpora, which are used to approximate the original documents. Furthermore by reducing the feature space of the data it is less likely to be influenced by the *Curse of Dimensionality (CoD)* (Bellman, 1961).

There are several methods to reduce the dimension of a data matrix, for example *Principal Component Analysis (PCA)* (Pearson, 1901) and *Latent Dirichlet Allocation (LDA)* (Blei et al., 2003). Another powerful technique namely *Non-negative Matrix Factorization (NMF)* (Lee and Seung, 1999) will be discussed in the first section of this work. Currently available NMF implementations require a prohibitively long computation time, which make the usage for real-world applications impractical. Therefore we present an implementation using the *Compute Unified Device Architecture (CUDA)* platform with a binding to the R environment. Furthermore the package is developed platform independent and is compatible with all three major platforms for R: Windows, Linux and Mac OS X.

## Overview of non-negative matrix factorization

Let  $X \in \mathbb{R}_+^{n \times m}$  be a matrix with  $n$  attributes and  $m$  observations in the dataset, then the data matrix  $X$  is approximated by the product of two new matrices  $W$  and  $H$  (Lee and Seung, 2001):

$$X \approx WH \quad (1)$$

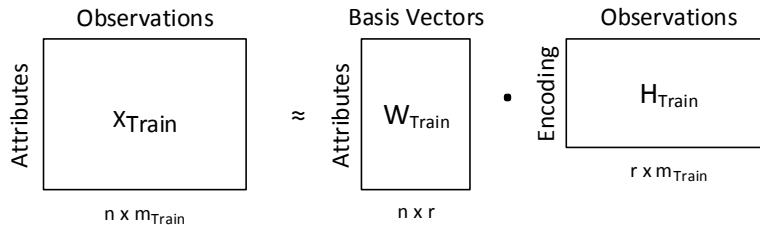
Each column of the matrix  $W \in \mathbb{R}_+^{n \times r}$  represents a single basis vector, whereas each column of the matrix  $H \in \mathbb{R}_+^{r \times m}$  represents an encoding vector. Therefore a column of the data matrix can be approximated by the linear combination of all basis vectors with one encoding vector (Lee and Seung, 2001). The importance of each basis vector can be seen by analysing the row sums of matrix H. Row sums with a low value identify basis vectors with very little influence on the dataset and vice versa (Skillcorn, 2007). It is also important to note that the data matrix as well as both matrices  $W$  and  $H$  contain only non-negative values.

Besides the general convention in the context of data mining, NMF expects columns to represent observations of the dataset instead of attributes (Skillcorn, 2007), as visualized in Figure 1. For that reason it is very important to read the data matrix definition in the literature carefully.

Contrary to PCA or *Singular Value Decomposition (SVD)*, the basis vectors are not linearly independent and thus the solution is not unique. However the reconstruction of the data matrix is purely additive and yields a more natural parts-based decomposition (Lee and Seung, 1999).

As the factorization should represent a compressed form of the original data matrix, one approach is to choose  $r$  depending on the number of rows and columns of the data matrix (Lee and Seung, 2001):

$$r < \frac{n \cdot m}{n + m} \quad (2)$$



**Figure 1:** NMF model which approximates the data matrix by a linear combination of basis vectors and an encoding matrix.

In general, one should choose  $r \ll m$  (Shahnaz et al., 2006). However, choosing the right parameter depends on the dataset and usage of the factorization.

### Pseudo-code

1. **Input:** Data matrix  $X \in \mathbb{R}_+^{n \times m}$  and number of features  $r$
2. Initialize  $W^{(0)} \in \mathbb{R}_+^{n \times r}$ ,  $H^{(0)} \in \mathbb{R}_+^{r \times m}$  with non-negative values and set  $k = 0$
3. **while**  $k < k_{\max}$  **and not** converged:
  - (a) Fix matrix  $W^{(k)}$  and compute matrix  $H^{(k+1)}$
  - (b) Fix matrix  $H^{(k+1)}$  and compute matrix  $W^{(k+1)}$
  - (c) Evaluate error function to check for convergence
  - (d)  $k = k + 1$
4. Set  $W = W^{(k)}$  and  $H = H^{(k)}$

### Initialization of factor matrices

Using a good initialization of the matrices can decrease the required number of iterations and further improve the factorization's quality. Depending on the chosen algorithm either only matrix  $W$  or both matrices need to be initialized.

Several different approaches were presented to execute step 2 of the pseudo-code, the most simple one by Lee and Seung (1999, 2001) namely initializing both matrices just with random values. A more complex initialization uses the SVD of the data matrix (Boutsidis and Galloopoulos, 2008), a very expensive approach which should be only used if the SVD is already available (Langville et al., 2014). However this initialization yields a unique factorization because SVD is also unique.

In general, the convergence theory of NMF is not researched enough. For example, Lee and Seung (2001) had shown that the multiplicative update rules converge to a local minimum. However Gonzalez and Zhang (2005) disproved that and clearly state the algorithm is only proven to converge at most to a saddle point. In fact most of the newer algorithms are only guaranteed to converge to a local minimum. This is mainly because NMF is a non-convex optimization problem (Lee and Seung, 2001). In each computation step only one of two matrices gets updated, independently from the other one. Hence finding a global minimum is unlikely, however multiple local minima do exist. If the execution time of an algorithm is short enough, then a Monte-Carlo like approach can be chosen (Berry et al., 2007). That implies executing the algorithm multiple times using different initializations each time and picking the factorization with the best quality.

### Error function

In the literature, different error or loss functions are proposed. The most common are *Kullback-Leibler Divergence* (Lee and Seung, 1999) and *Frobenius norm* (Paatero and Tapper, 1994; Lee and Seung, 2001). Since only the Frobenius norm is used in this work, Kullback-Leibler divergence won't be discussed.

In an abstract sense, the Frobenius norm of a matrix  $A \in \mathbb{R}_+^{n \times m}$  is equal to the Euclidean distance of a vector  $\vec{a} \in \mathbb{R}_+^{n \cdot m}$ . To be more precise the Frobenius norm is the square root of the sum of all

squared matrix elements (Reinhardt et al., 2013):

$$\|A\|_F := \sqrt{\sum_{i=1}^n \sum_{j=1}^m |a_{ij}|^2} \quad (3)$$

Besides this general definition there do exist alternative representations, among others the representation using the trace of a matrix (Reinhardt et al., 2013):

$$\|A\|_F := \sqrt{\text{trace}(A^T A)} \quad (4)$$

For optimized computation the widely used minimization problem is rearranged using this equivalence:

$$\min \frac{1}{2} \cdot \|X - WH\|_F^2 = \min \frac{1}{2} \left( \text{trace}(X^T X) - 2 \cdot \text{trace}(H^T W^T X) + \text{trace}(H H^T W^T W) \right) \quad (5)$$

Upon first sight the error function seems to be more expensive to compute but actually most terms get computed during the algorithm execution anyway (Berry et al., 2007; Langville et al., 2014). Furthermore, the trace ( $X^T X$ ) is constant and can be precomputed.

The following algorithms minimize the Frobenius norm, but can also easily be derived for other error functions.

### Updating with multiplicative update rules

Multiplicative update rules have been first described by Lee and Seung (1999, 2001) and are the fastest algorithms in terms of computational cost per iteration. In fact this type of algorithm is a special case of the gradient-descent algorithm with a specific step size (Lee and Seung, 2001). Both update rules for the matrices  $W$  and  $H$  are applied in an alternating fashion to solve step 3a) and 3b) of the NMF pseudo-code:

$$H^{(k+1)} = H^{(k)} \otimes \left( (W^{(k)})^T X \right) \oslash \left( (W^{(k)})^T W^{(k)} H^{(k)} \right) \quad (6)$$

$$W^{(k+1)} = W^{(k)} \otimes \left( X (H^{(k+1)})^T \right) \oslash \left( W^{(k)} H^{(k+1)} (H^{(k+1)})^T \right) \quad (7)$$

Where  $\otimes$  denotes the element-wise matrix multiplication and  $\oslash$  the element-wise matrix division. However it is advised to add an epsilon to the denominator, e.g.  $\epsilon \approx 10^{-9}$  for double precision floating point values, to avoid divisions by zero (Berry et al., 2007). Referring to table 3 in the implementation section, multiplicative update rules are used in `mu` and `nsNMF` for both matrices, in `gdcls` only for matrix  $W$ .

### Updating with alternating least squares

*Alternating Least Squares (ALS)* type algorithms are another approach to solve step 3a) and 3b) of the NMF pseudo-code. The central idea is that for one given matrix the other one can be computed using a least-squares projection (Paatero and Tapper, 1994).

$$(W^{(k)})^T W^{(k)} H^{(k+1)} = (W^{(k)})^T X \quad (8)$$

$$H^{(k+1)} (H^{(k+1)})^T (W^{(k+1)})^T = H^{(k+1)} X^T \quad (9)$$

In the first step, Equation 8 gets solved to  $H^{(k+1)}$  whereby the computation of matrix  $W$  becomes possible. Equation 9 gets solved for  $(W^{(k+1)})^T$ , followed by transposing the solution to acquire the matrix  $W^{(k+1)}$ .

Since solving a linear equation system possibly yields negative values, the non-negativity constraint for both matrices  $W$  and  $H$  must be ensured after each solving step. One possible solution for this problem is to set all negative values to zero (Langville et al., 2014).

Langville et al. (2014) describe ALS extensions like *Alternating Constraint Least Squares (ACLS)* and *Alternating Hoyer Constraint Least Squares (AHCLS)*, which use additional parameters to provide a more

sparse factorization. Therefore both diagonal and non-diagonal values of the covariance matrices  $W^T W$  and  $H H^T$  get manipulated. For example, the AHCLS uses the additional parameters  $\lambda_W, \lambda_H, \alpha_W$  and  $\alpha_H$  to solve the following equations:

$$\left( \left( W^{(k)} \right)^T W^{(k)} + \lambda_H \cdot \beta_H \cdot I - \lambda_H \cdot E \right) H^{(k+1)} = \left( W^{(k)} \right)^T X \quad (10)$$

$$\left( H^{(k+1)} \left( H^{(k+1)} \right)^T + \lambda_W \cdot \beta_W \cdot I - \lambda_W \cdot E \right) \left( W^{(k+1)} \right)^T = H^{(k+1)} X^T \quad (11)$$

Where  $I \in \mathbb{R}^{r \times r}$  denotes the identity matrix and  $E \in \mathbb{R}^{r \times r}$  a matrix of ones, furthermore  $\beta_W$  and  $\beta_H$  are defined as:

$$\beta_W := ((1 - \alpha_H) \cdot \sqrt{r} + \alpha_H)^2 \quad (12)$$

$$\beta_H := ((1 - \alpha_W) \cdot \sqrt{r} + \alpha_W)^2 \quad (13)$$

The authors have advised to use  $\lambda_W, \lambda_H \in [0, \infty)$  and  $\alpha_W, \alpha_H \in [0, 1]$ , where  $\alpha_W$  and  $\alpha_H$  should represent the requested percentage of sparsity. As a head start all four values should be set to 0.5. Once more referring to Table 3 in the implementation section, ALS update rules are used in `als`, `acls`, and `ahcls` for both matrices, in `gdccls` only for matrix  $H$ .

## The NMF algorithm for R using CUDA: `nmfgpu4R`

There already exist some approaches to compute NMF in R, for example the **NMF** (Gaujoux and Seoighe, 2010) and **NMFN** (Liu, 2012) packages on CRAN. However both packages use the CPU for the computational process and even with parallelization of multiple runs the usage for real-world datasets is limited.

CUDA-based implementations of NMF are already part of the **GPUMLib**<sup>1</sup> (Lopes and Ribeiro, 2010), which itself contains various algorithms of machine learning tasks for CUDA platforms. Currently, as of version 0.3.4, there are two algorithms available, one additive and one multiplicative, for both Frobenius norm and Kullback-Leibler divergence. Besides that no complex initialization strategies or algorithms incorporating constraints are available. Furthermore the computation of NMF is restricted to single precision format, which might not be suitable for every dataset.

In this work we propose a new package called **nmfgpu4R**<sup>2</sup>, which is a binding to a separate library called **nmfgpu**<sup>3</sup> written in C++11 using CUDA (version  $\geq 7.0$ ) for Nvidia GPUs with compute capability  $\geq 3.0$  (Kepler). When using CUDA, different build tools must be chosen depending on the platform. This limitation is induced by Nvidia's nvcc compiler, which only supports one compiler per platform (nvcc itself is built on top of one compiler). By splitting the package and C++ library in two separate modules, it is possible to provide both **nmfgpu4R** and **nmfgpu** for all three major platforms: Windows, Linux, and Mac OS X.

Modern *Graphics Processing Units (GPU)* can also be used as *High Performance Computing (HPC)* devices using either OpenCL or CUDA. Latter is restricted to only Nvidia hardware but is more common and can be integrated directly into C/C++ source code. One advantage of the GPU over CPU parallelization is that algorithms have to be developed scalable and data parallel. Synchronization and data transfer logic has to be handled by the developer and therefore these algorithms are able to profit more from new and more powerful hardware generations. For more information about the CUDA platform please visit the Nvidia CUDA website<sup>4</sup>.

## Supported data matrix formats

Internally the library computes the algorithms using dense matrices, so one option is to pass in a numeric matrix with proper dimensions. Furthermore the **nmfgpu4R** package currently supports S4 classes from the **Matrix** package, developed by Bates and Maechler (2014), and the **SparseM** package, developed by Koenker and Ng (2015). A complete reference about supported S4 classes is listed

<sup>1</sup><https://sourceforge.net/projects/gpumlib/> (last access: 18.04.2016)

<sup>2</sup><https://github.com/razorx89/nmfgpu4R> (last access: 18.04.2016)

<sup>3</sup><https://github.com/razorx89/nmfgpu> (last access: 18.04.2016)

<sup>4</sup>[http://www.nvidia.com/object/cuda\\_home\\_new.html](http://www.nvidia.com/object/cuda_home_new.html) (last access: 18.04.2016)

<https://developer.nvidia.com/cuda-zone> (last access: 18.04.2016)

in table 1. It is important to note that the sparse matrices get converted into dense matrices on the GPU-side. At the moment, a computation using sparse algorithms does not take place at any time.

Storage Format	Matrix	SparseM
Dense	"dgeMatrix"	-
Coordinate (COO)	"dgTMatrix"	"matrix.coo"
Compressed Sparse Column (CSC)	"dgCMatrix"	"matrix.csc"
Compressed Sparse Row (CSR)	"dgRMatrix"	"matrix.csr"

**Table 1:** Supported S4 classes as input data matrix to **nmfgpu4R**.

However this feature allows large sparse matrices to be converted much faster in GPU memory. For example this might be quite useful for *Bag-of-Words* (*BoW*) in text mining (Salton and Buckley, 1988) or *Bag-of-Visual-Words* (*BoVW*) in image classification / retrieval (Cula and Dana, 2001), where the vocabulary is commonly very large but the frequencies are mostly zero.

### Customizing the initialization

Algorithms of the Non-negative Matrix Factorizations solve a non-convex optimization problem. Thus choosing a good initialization can reduce the number of iterations and yield better results. In NMF four different initialization strategies are implemented. There are different approaches to choose an initialization for both matrices  $W$  and  $H$ . It is important to keep in mind that when an ALS type algorithm is chosen only matrix  $W$  has to be initialized. Matrix  $H$  will be computed in the first iteration from only matrix  $W$  and the data matrix.

Strategy	Matrix $W$	Matrix $H$
CopyExisting	Copy $W$	Copy $H$
AllRandomValues	Random	Random
MeanColumns	Mean of $k$ random columns	Random
k-means/Random	k-means	Random
k-means/NonNegativeWTH	k-means	$h_{ij} = \max(0, (W^T X)_{ij})$
EIn-NMF	k-means	$h_{ij} = 1 / \left( \sum_{k=1}^i \left( \frac{\ x_j - c_k\ _2}{\ x_j - c_i\ _2} \right)^{\frac{2}{1-m}} \right)$

**Table 2:** Supported initialization strategies for initializing matrix  $W$  and  $H$ .

All supported initializations by **nmfgpu4R** are listed in Table 2. Strategy *CopyExisting* can be used to provide user-defined initializations for both matrices  $W$  and  $H$  which get copied directly into GPU memory. When using *AllRandomValues* both matrices  $W$  and  $H$  get initialized by random values which is the most common but also the simplest strategy (Pauca et al., 2006). Langville et al. (2014) presented a method called *MeanColumns* to form initial basis vectors from data columns. The idea behind this initialization is that if the data columns are sparse then the initial basis vectors should be sparse as well. Furthermore, k-means clustering can be used to find initial basis vectors (Gong and Nandi, 2013). If matrix  $H$  has to be initialized in the context of k-means based initializations, then there are different approaches. Most complex is the *EIn-NMF* initialization which computes the membership degree of each data column (Gong and Nandi, 2013).

### Using different algorithms

There are currently six different algorithms implemented in **nmfgpu4R**, because NMF models can be computed in different ways and, furthermore, can be restricted by constraints. Those algorithms which do have extra constraints, can also be adjusted through parameters. In Table 3 all implemented algorithms and their corresponding publications are listed.

A few of these algorithms will be evaluated in the benchmark section, using two different image datasets. In general the right choice of algorithm depends on the data and noise within the data. For an overview of all required parameters for a specific algorithm, please have a look at the package documentation.

Method	Name	Publication
acls	Alternating Constrained Least Squares	Langville et al. (2014)
ahcls	Alternating Hoyer Constrained Least Squares	Langville et al. (2014)
als	Alternating Least Squares	Paatero and Tapper (1994)
gdcls	Gradient Descent Constrained Least Squares	Shahnaz et al. (2006)
mu	Multiplicative Update Rules (Frobenius Norm)	Lee and Seung (2001)
nsmf	non-smooth Non-negative Matrix Factorization	Pascual-Montano et al. (2006)

Table 3: Overview of implemented algorithms in **nmfgpu4R**.

### Adjusting convergence tests

Most NMF implementations only use the number of iterations as a convergence test, as this is a very cheap test. However, for a mathematically correct convergence test an error function has to be computed and observed during the algorithm execution. In NMF there are four different stopping criteria implemented, which can also be combined. The **nmfgpu4R** package implements both: the convergence test by observing an error function, as the primary and an upper limit of iterations, as the secondary convergence criterion.

Setting the threshold value can be done by passing in the parameter `threshold`. This value is actually interpreted differently depending on the configured error function. Currently the *Frobenius Norm* and *Root Mean Squared Deviation (RMSD)* are supported. One advantage of the RMSD error function is that it is normalized by the number of data matrix elements and therefore independent of the data matrix dimension. By passing in the parameter `maxiter` the maximum number of iterations can be overwritten, which is by default set to 2000. For example, execute the algorithm until the delta error is less than 0.1 regarding the RMSD error function but at most 500 iterations:

```
result <- nmf(data, r, threshold=0.1, thresholdType="rmsd", maxiter=500)
```

Depending on the datasets the ALS type algorithms are sometimes not stable and therefore not monotonically decreasing. In such a case the convergence test using the threshold value will not work properly.

### Encoding matrix for new unseen data

A simple but effective method to calculate an encoding matrix for unseen data was described by Lopes and Ribeiro (2011), which allows NMF to be used within learning algorithms. Using this method the training data gets factorized with a normal NMF step. However the factorization step of the testing data reuses the matrix  $W$  and only updates the matrix  $H$ . Thus the resulting matrix  $H$  is an encoding of learned basis vectors from the training data. A complete scheme of the process is visualized in figure 2.

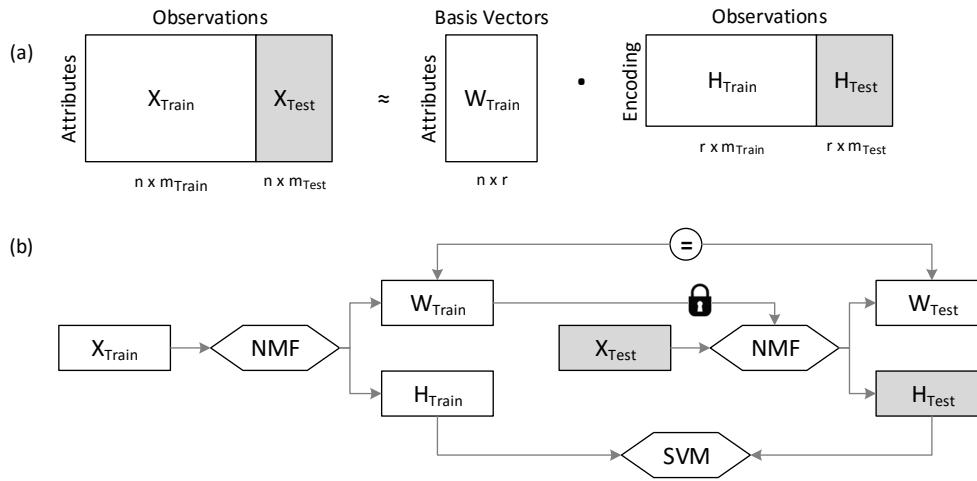
As a result, structures between both training and test data are being preserved, but the feature dimension in matrix  $H$  can be reduced to a fraction of the original dimension. Hence, learning, for example, a Support-Vector-Machine (SVM) can be speeded up and furthermore prediction accuracy can be improved.

In the following example code the `nmf` method is used to train the basis vectors for the training dataset. After that, the generic `predict` method can be used to either retrieve the encoding matrix  $H$  of the training data or to generate an encoding matrix for a new data matrix. The objective here is to reduce the 4 dimensions of the `iris` dataset (Fischer, 1936) to 2 dimensions.

```
# Set seed for reproducible results
set.seed(42)

# Split iris dataset into training and test data
idx <- sample(1:nrow(iris), 100, replace=F)
data.train <- iris[idx,-5]
data.test <- iris[-idx,-5]

# Compute model and retrieve encoding matrix H for both training and test data
library(nmfgpu4R)
nmfgpu4R.init()
model <- nmftt(data.train), 2)
encoding.train <- t(predict(model)) # Identical: encoding.train <- t(model$H)
```



**Figure 2:** (a) Prediction of an encoding matrix for unseen data. The data matrix of the existing NMF model is "extended" by new data, but the basis vectors are fixed. (b) Data flow visualization of the prediction process in the context of a SVM (derived from Lopes and Ribeiro (2011)).

```

encoding.test <- t(predict(model, t(data.test)))

# Use encoding matrices to predict "Species"
library(e1071)
model.svm <- best.svm(x=encoding.train, y=iris$Species[idx])
prediction <- predict(model.svm, encoding.test)
table(iris[-idx,5], prediction)

```

Using the *iris* dataset is just an example and should be replaced with a much larger dataset to fully utilize the GPU. Furthermore an improvement in speed and possibly in accuracy over non-reduced data is more likely to be observed when the dimension is reduced by a larger magnitude.

This example learns basis vectors from a training dataset and predicts the encoding matrix for the test dataset. To visualize the encoding matrices of both datasets and their relationships, a simple scatter plot can be made with the following code:

```

# Plot encoding matrices
library(ggplot2)
data.plot <- data.frame(rbind(encoding.train, encoding.test),
                        class=unlist(list(iris[idx,5], iris[-idx,5])),
                        type=c(rep("Train", nrow(data.train)),
                               rep("Test", nrow(data.test))))
ggplot(data.plot, aes(x=r1, y=r2, color=class, shape=type)) + geom_point()

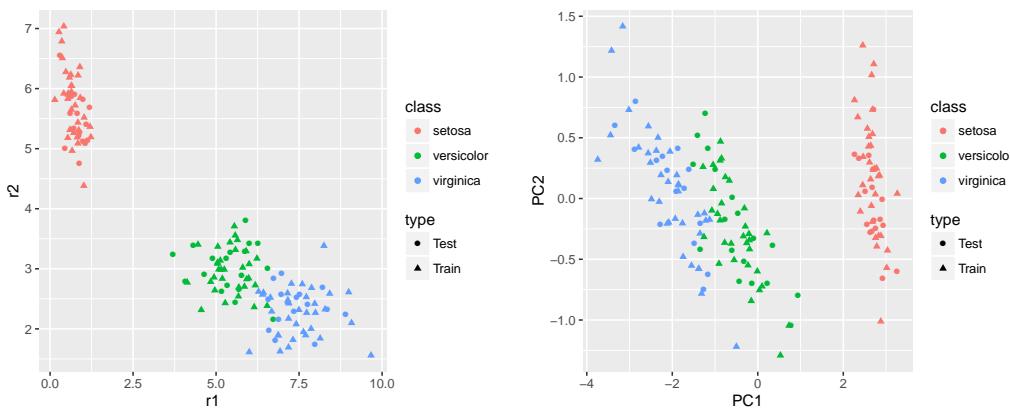
```

As shown in Figure 3, both datasets share the same structure. Observations from each of the three classes are predicted to belong to the same area as the training observations.

### Issues during development

The **nmfgpu4R** package provides a binding to an independent C++ library, which uses the latest C++ features from the C++11 standard. In order to support multiple platforms deploying an extra library is a necessary step since the Nvidia CUDA compiler *nvcc* only supports the Microsoft Visual C++ compiler on Windows platforms. But R uses its own compilation tool chain and therefore does not allow the Microsoft Visual C++ compiler.

The main problem is that C++ compilers emit an object code which is not compatible with the object code of another compiler. R uses g++ from the MinGW tool chain and therefore both compiled binaries are not link-compatible, virtual tables are only compatible in some situations and struct returns simply do not work. Furthermore since the object code is not link-compatible one must fall back to an extern "C" interface, which then can be loaded using native system calls like *GetProcAddress* on Windows or *dlsym* on Linux/Mac OS. Such issues do not come up on Linux or Mac OS because R uses on these platforms the default configured compiler which is also supported by the nvcc compiler.



**Figure 3:** Visualization of the encoding matrices for the iris dataset (Fischer, 1936), which is reduced by the nmf method (left) and by the pcromp method (right) to 2 dimensions.

## Benchmarks

In this section multiple benchmarks are described which were performed on the *Yale Face Database* (Belhumeur et al., 1997) and *Cropped Extended Yale Face Database B* (Lee et al., 2005). As a preprocessing step all images were scaled to a common height of 64 pixels while preserving the aspect ratio. The resulting matrix dimensions can be taken from table 4.

Dataset	Pixels (orig.)	Pixels (scaled)	Count	Matrix
Yale Face Database	320 × 243	64 × 64	165	4096 × 165
Extended Yale Face Database B	168 × 192	56 × 64	2414	3584 × 2414

**Table 4:** Dimensions of data matrices which where used to benchmark existing CPU implementations as well as GPU implementations by the **nmfgpu4R** package.

For testing, a system server with CentOS 7.2.1511, Intel Xeon E5-2687W v3 @3.10GHz (10 physical cores), 256GB RAM, Nvidia GeForce GTX Titan X and two Nvidia Tesla K80 was used. R is a custom build of version 3.3.1 using OpenBLAS<sup>5</sup> as BLAS back-end.

In this benchmark the **nmfgpu4R** (version 0.2.5.1) package is compared to the CRAN packages **NMF** (version 0.20.6) and **NMFN** (version 2.0), which both provide CPU implementations of common NMF algorithms. The **NMF** package does provide optimized C++ algorithms as well as pure R implementations. Regarding the package documentation parallelization is only performed using clusters for parallelizing multiple runs of the same algorithm with different initializations. In order to fully utilize the CPU cores, pure R algorithms were benchmarked using an OpenBLAS back-end with OPENBLAS\_NUM\_THREADS=10. Algorithms from the **NMFN** package were modified to accept preinitialized matrices to be able to compare the algorithms with identical starting points. Both the CPU and GPU algorithms were executed 10 times each.

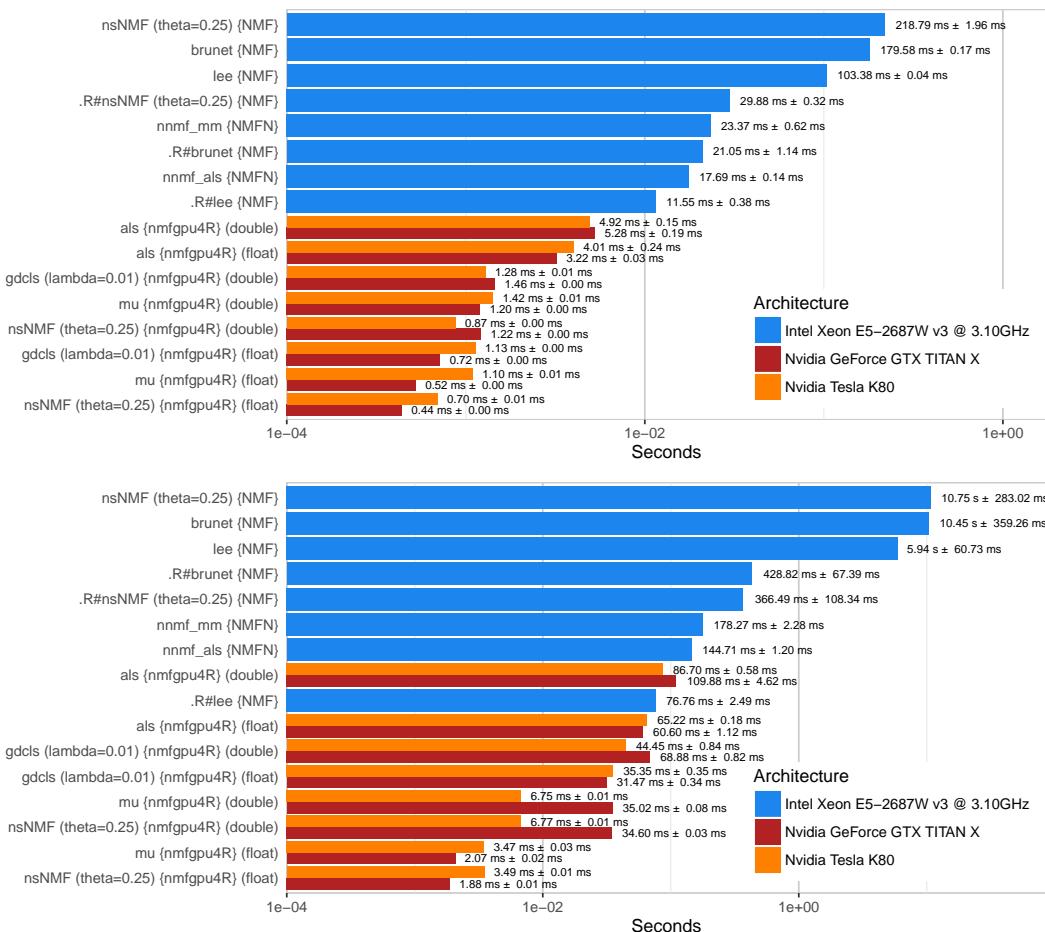
As already stated in the previous section Alternating Least Squares algorithms seem to perform poorly on very dense datasets, leading to a non-stable factorization or even no solution at all. However the execution times of the ALS algorithms in **nmfgpu4R** are the highest of all GPU algorithms, but they are still very low compared to the ALS implementation in **NMFN**, which is shown by Figure 4 (top). Furthermore, the optimized C++ algorithms in the **NMF** package are much slower when computed in sequential mode compared to the R implementations, which are accelerated by the multi-threaded OpenBLAS back-end.

Overall the multiplicative algorithm is the fastest algorithm for both GPU and CPU. Depending on the dataset it might be useful to compute the factorization in single precision format, because modern GPUs have still more single precision than double precision floating point units. As shown by Figure 4, GPUs of Nvidia's GeForce series are optimized for single precision calculations, which is sufficient for end-user gaming experience. However double precision computation is very limited on those cards, whereas the Tesla series also provides enough double precision units for fast calculations. As Table 5 indicates, there is no noticeable difference in terms of factorization quality but very much in execution time. Small variations between error functions can be caused due to computational ordering and on GPU-side due to dispatching of thread blocks.

<sup>5</sup><http://www.openblas.net> (last access: 31.10.2016)

Device	Package	Algorithm	Yale Face Database		Extended Yale Face Database B+		
			$\ X - WH\ _F$	Elapsed Time	$\ X - WH\ _F$	Elapsed Time	
Intel Xeon E5-2687W v3 @3.10GHz	NMF	.R#brunet	82.12 ± 0.68	42.11 ± 2.28s	218.39 ± 0.58	857.64 ± 134.77s	
		.R#lee	75.68 ± 0.56	23.10 ± 0.75s	208.15 ± 0.78	153.53 ± 4.98s	
		.R#nsNMF ( $\theta=0.25$ )	92.15 ± 0.69	59.75 ± 0.64s	243.59 ± 0.50	732.97 ± 216.69s	
		brunet	82.12 ± 0.68	359.15 ± 0.34s	218.39 ± 0.58	20891.94 ± 718.52s	
		lee	75.68 ± 0.56	206.76 ± 0.08s	208.15 ± 0.78	11877.49 ± 121.46s	
	NMFN	nsNMF ( $\theta=0.25$ )	92.15 ± 0.69	437.58 ± 3.91s	243.59 ± 0.50	21503.39 ± 566.04s	
		nmmf_als	227.86 ± 18.11	35.37 ± 0.28s	1188.68 ± 49.38	289.41 ± 2.41s	
		nmmf_mm	75.72 ± 0.52	46.74 ± 1.23s	208.02 ± 0.75	356.53 ± 4.56s	
	Nvidia GeForce GTX TITAN X (Maxwell)	nmfgpu4R (double)	als	218.97 ± 8.71	10.56 ± 0.38s	1182.94 ± 50.47	219.77 ± 9.24s
		gdcls ( $\lambda=0.01$ )	92.47 ± 0.91	2.91 ± 0.01s	217.94 ± 0.85	137.75 ± 1.63s	
		mu	75.86 ± 0.49	2.40 ± 0.01s	208.37 ± 0.72	70.03 ± 0.15s	
		nsNMF ( $\theta=0.25$ )	88.97 ± 0.47	2.43 ± 0.01s	254.38 ± 0.85	69.20 ± 0.05s	
		nmfgpu4R (float)	als	231.60 ± 19.44	6.43 ± 0.07s	1161.70 ± 34.22	121.20 ± 2.24s
		gdcls ( $\lambda=0.01$ )	92.75 ± 1.06	1.44 ± 0.01s	219.04 ± 0.90	62.95 ± 0.68s	
		mu	75.86 ± 0.49	1.05 ± 0.01s	208.38 ± 0.72	4.14 ± 0.04s	
		nsNMF ( $\theta=0.25$ )	86.74 ± 0.33	0.88 ± 0.01s	237.35 ± 0.77	3.76 ± 0.01s	
Nvidia Tesla K80 (Kepler)	nmfgpu4R (double)	als	223.00 ± 14.60	9.84 ± 0.30s	1160.46 ± 57.60	173.40 ± 1.17s	
		gdcls ( $\lambda=0.01$ )	93.26 ± 1.31	2.57 ± 0.01s	218.75 ± 1.37	88.89 ± 1.68s	
		mu	74.24 ± 0.32	2.84 ± 0.01s	217.71 ± 0.94	13.51 ± 0.01s	
		nsNMF ( $\theta=0.25$ )	88.36 ± 0.93	1.75 ± 0.01s	254.58 ± 1.16	13.54 ± 0.01s	
		nmfgpu4R (float)	als	233.16 ± 21.10	8.01 ± 0.48s	1147.49 ± 36.85	130.43 ± 0.35s
		gdcls ( $\lambda=0.01$ )	93.80 ± 1.53	2.26 ± 0.01s	218.83 ± 1.02	70.69 ± 0.69s	
		mu	74.25 ± 0.32	2.19 ± 0.02s	217.72 ± 0.94	6.95 ± 0.06s	
		nsNMF ( $\theta=0.25$ )	84.60 ± 0.39	1.39 ± 0.02s	246.74 ± 0.74	6.98 ± 0.02s	

**Table 5:** Benchmark results for the Yale Face Database with  $r = 32$  features and Cropped Extended Yale Face Database with  $r = 128$ . Each measurement was taken at iteration 2000 with  $n = 10$  computations.



**Figure 4:** Computation time for one iteration on the Yale Face Database with  $r = 32$  (top) and Cropped Extended Yale Face Database B with  $r = 128$  (bottom) shown on a logarithmic scale.

## Summary

In this work a new possibility to compute Non-negative Matrix Factorizations (NMF) using CUDA hardware is presented. As shown in the benchmarks, the performance gain is remarkable and therefore much larger datasets can be reduced, without having to wait on completion for weeks or even months. Currently the implementation is only limited by the available memory on the device, because all algorithms work directly in device memory without transferring intermediate results back to the host. Possible extensions to this library/package could make use of out-of-core computation and multiple CUDA devices, either to compute one distributed factorization or multiple factorizations with different initializations. Furthermore more complex algorithms and initialization strategies could be implemented in the future.

## Bibliography

- D. Bates and M. Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2014. URL <http://CRAN.R-project.org/package=Matrix>. R package version 1.1-4. [p386]
- P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, Jul 1997. ISSN 0162-8828. doi: 10.1109/34.598228. [p390]
- R. Bellman. Adaptive control processes: A guided tour. (A RAND corporation research study). Princeton, N. J.: Princeton University Press, XVI, 255 p., 1961. [p383]
- M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 52(1):155 – 173, 2007. ISSN 0167-9473. doi: <http://dx.doi.org/10.1016/j.csda.2006.11.006>. URL <http://www.sciencedirect.com/science/article/pii/S0167947306004191>. [p384, 385]
- D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3: 993–1022, Mar. 2003. ISSN 1532-4435. URL <http://dl.acm.org/citation.cfm?id=944919.944937>. [p383]
- C. Boutsidis and E. Gallopoulos. SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350 – 1362, 2008. ISSN 0031-3203. doi: <http://dx.doi.org/10.1016/j.patcog.2007.09.010>. URL <http://www.sciencedirect.com/science/article/pii/S0031320307004359>. [p384]
- O. G. Cula and K. J. Dana. Compact representation of bidirectional texture functions. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I-1041–I-1047, 2001. doi: 10.1109/CVPR.2001.990645. [p387]
- R. A. Fischer. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2): 179–188, 1936. ISSN 2050-1439. doi: 10.1111/j.1469-1809.1936.tb02137.x. URL <http://dx.doi.org/10.1111/j.1469-1809.1936.tb02137.x>. [p388, 390]
- R. Gaujoux and C. Seoighe. A flexible R package for nonnegative matrix factorization. *BMC Bioinformatics*, 11(1):367, 2010. ISSN 1471-2105. doi: 10.1186/1471-2105-11-367. URL <http://www.biomedcentral.com/1471-2105/11/367>. [p386]
- L. Gong and A. Nandi. An enhanced initialization method for non-negative matrix factorization. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, Sept 2013. doi: 10.1109/MLSP.2013.6661949. [p387]
- E. F. Gonzalez and Y. Zhang. Accelerating the Lee-Seung algorithm for nonnegative matrix factorization. CAAM Technical Report TR05-02, Rice University, Mar. 2005. URL <http://www.caam.rice.edu/caam/trs/2005/TR05-02.pdf>. [p384]
- R. Koenker and P. Ng. *SparseM: Sparse Linear Algebra*, 2015. URL <http://CRAN.R-project.org/package=SparseM>. R package version 1.6. [p386]
- A. N. Langville, C. D. Meyer, R. Albright, J. Cox, and D. Duling. Algorithms, initializations, and convergence for the nonnegative matrix factorization. *CoRR*, abs/1407.7299, 2014. URL <http://arxiv.org/abs/1407.7299>. [p384, 385, 387, 388]
- D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. [p383, 384, 385]

- D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 556–562. MIT Press, 2001. URL <http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf>. [p383, 384, 385, 388]
- K.-C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):684–698, May 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.92. [p390]
- S. T. Liu. *NMFN: Non-negative Matrix Factorization*, 2012. URL <http://CRAN.R-project.org/package=NMFN>. R package version 2.0. [p386]
- N. Lopes and B. Ribeiro. Non-negative matrix factorization implementation using graphic processing units. *Intelligent Data Engineering and Automated Learning – IDEAL 2010*, Jan. 2010. doi: 10.1007/978-3-642-15381-5\_34. URL [http://dx.doi.org/10.1007/978-3-642-15381-5\\_34](http://dx.doi.org/10.1007/978-3-642-15381-5_34). [p386]
- N. Lopes and B. Ribeiro. A fast optimized semi-supervised non-negative matrix factorization algorithm. In *The 2011 International Joint Conference on Neural Networks (IJCNN)*, pages 2495–2500, July 2011. doi: 10.1109/IJCNN.2011.6033543. [p388, 389]
- P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994. ISSN 1099-095X. doi: 10.1002/env.3170050203. URL <http://dx.doi.org/10.1002/env.3170050203>. [p384, 385, 388]
- A. Pascual-Montano, J. M. Carazo, K. Kochi, D. Lehmann, and R. D. Pascual-Marqui. Nonsmooth nonnegative matrix factorization (nsNMF). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):403–415, Mar. 2006. ISSN 0162-8828. doi: 10.1109/TPAMI.2006.60. [p388]
- V. P. Pauca, J. Piper, and R. J. Plemmons. Nonnegative matrix factorization for spectral data analysis. *Linear Algebra and its Applications*, 416(1):29 – 47, 2006. ISSN 0024-3795. doi: <http://dx.doi.org/10.1016/j.laa.2005.06.025>. URL <http://www.sciencedirect.com/science/article/pii/S002437950500340X>. Special Issue devoted to the Haifa 2005 conference on matrix theory. [p387]
- K. Pearson. On lines and planes of closest fit to system of points in space. *Philosophical Magazine*, 2: 559–572, 1901. [p383]
- R. Reinhardt, A. Hoffmann, and T. Gerlach. *Nichtlineare Optimierung - Theorie, Numerik und Experimente*. Springer, 2013. ISBN 978-3827429490. [p385]
- G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513 – 523, 1988. ISSN 0306-4573. doi: [http://dx.doi.org/10.1016/0306-4573\(88\)90021-0](http://dx.doi.org/10.1016/0306-4573(88)90021-0). URL <http://www.sciencedirect.com/science/article/pii/0306457388900210>. [p387]
- F. Shahnaz, M. W. Berry, V. Pauca, and R. J. Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing & Management*, 42(2):373–386, Mar. 2006. ISSN 0306-4573. URL <http://www.sciencedirect.com/science/article/pii/S0306457304001542>. [p384, 388]
- D. Skillicorn. *Understanding Complex Datasets: Data Mining with Matrix Decompositions*. Chapman & Hall/CRC, 2007. ISBN 978-1584888321. [p383]

*Sven Koitka*  
*Department of Computer Science*  
*University of Applied Sciences and Arts Dortmund (FHDO)*  
*Emil-Figge-Straße 42, 44227 Dortmund*  
*Germany*  
*sven.koitka@fh-dortmund.de*

*Christoph M. Friedrich*  
*Department of Computer Science*  
*University of Applied Sciences and Arts Dortmund (FHDO)*  
*Emil-Figge-Straße 42, 44227 Dortmund*  
*Germany*  
*christoph.friedrich@fh-dortmund.de*

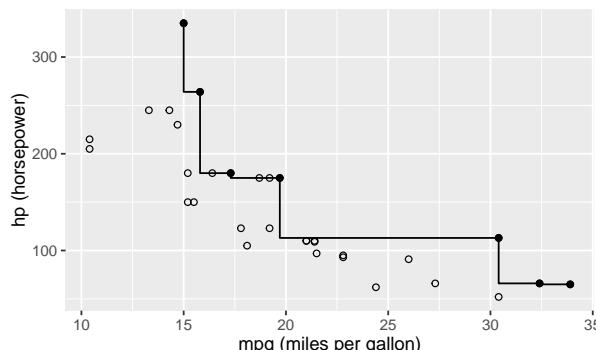
# Computing Pareto Frontiers and Database Preferences with the rPref Package

by Patrick Roocks

**Abstract** The concept of Pareto frontiers is well-known in economics. Within the database community there exist many different solutions for the specification and calculation of Pareto frontiers, also called *Skyline queries* in the database context. Slight generalizations like the combination of the Pareto operator with the lexicographical order have been established under the term *database preferences*. In this paper we present the **rPref** package which allows to efficiently deal with these concepts within R. With its help, database preferences can be specified in a very similar way as in a state-of-the-art database management system. Our package provides algorithms for an efficient calculation of the Pareto-optimal set and further functionalities for visualizing and analyzing the induced preference order.

## Introduction

A Pareto set is characterized by containing only those tuples of a data set which are not Pareto-dominated by any other tuple of the data set. We say that a tuple  $t$  Pareto-dominates another tuple  $t'$  if it is better or equal w.r.t. all relevant attributes and there exists at least one attribute where  $t$  is strictly better than  $t'$ . Typically, such sets are of interest when the dimensions looked upon tend to be anticorrelated. Consider the Pareto set of `mtcars` where the fuel consumption shall be minimized (i.e., `miles per gallon` shall be maximized) and the horsepower maximized, which is depicted in Figure 1. These dimensions tend to anticorrelate and hence a car buyer might Pareto-optimize both dimensions to get those cars which are *optimal compromises*.



**Figure 1:** The bold points represent the Pareto set of cars with high mpg and hp values. The Pareto front line connecting them isolates the dominated area.

In the database community this concept was introduced under the name *Skyline Operator* in Börzsönyi et al. (2001). In this pioneer paper they suggest a `SKYLINE OF` clause extending a usual SQL query to specify the optimization goals for Pareto sets. For example, if `mtcars` is an SQL table, the Pareto-optimal cars from the example above can be selected using the Skyline Operator by

```
SELECT * FROM mtcars SKYLINE OF hp MAX, mpg MAX
```

Such queries can be rewritten into standard SQL, cf. Kießling and Köstler (2002). But such a rewritten query contains a very complex `WHERE` clause, which is very inefficient to process for common query processors. To the best of our knowledge there is no open-source database management system supporting Skyline queries off-the-shelf.

There exists a commercial database management system *EXASolution* which supports such Skylines queries in a slightly generalized manner (Mandl et al., 2015). The idea is to construct *preference terms* within the SQL query, which induce strict orders (irreflexive and transitive). This approach was adapted from the preference framework presented in Kießling (2002). The syntactical schema of an SQL query in EXASolution using the *Skyline*-Feature is given by:

```
SELECT ... FROM ... WHERE ...
PREFERRING {pref-term} [PARTITION BY A_1, ..., A_k]
```

The PARITION BY clause splits the data set into groups in which the preference is evaluated separately. The preference term {pref-term} can contain several sub-constructs:

```
{pref-term} ::= [LOW | HIGH] {numerical-expression} | {logical-expression} |
{pref-term} PLUS {pref-term} | {pref-term} PRIOR TO {pref-term} |
INVERSE {pref-term}
```

The LOW and HIGH predicates induce orders where small or high values, respectively, are preferred. They have to be followed by an expression which evaluates to a numerical value. An expression which evaluates to a logical value is called a *Boolean preference*. In the induced order all tuples evaluating to TRUE are better than the FALSE ones.

These base constructs can be combined to obtain more complex preferences terms using one of the *preference operators*. The operator PLUS denotes the *Pareto composition*. This means that {p1} PLUS {p2} induces an order, where a tuple is better if and only if, it is better or equal w.r.t. both {p1} and {p2} and strictly better w.r.t. one of them. The PRIOR TO keyword combines the two given orders using the lexicographical order, which is also called *Prioritization* in the wording of Kießling (2002). Finally, the INVERSE keyword reverses the order. For example, LOW A is equivalent to INVERSE HIGH A.

The above Skyline example simply translates to the EXASolution query:

```
SELECT * FROM mtcars PREFERING HIGH hp PLUS HIGH mpg
```

In this paper we will present our **rPref** package (Roocks, 2016b) for computing optimal sets according to preferences within R. We decided to stick closely to the EXASolution semantics of preferences in our package, which is more general than Skylines but still has a clean and simple syntax.

Our first ideas to process preferences in R were published in Roocks and Kießling (2013) under the name “R-Pref”. In that approach the Pareto set calculation was done entirely in R, requiring nested for-loops, which made the preference evaluation very slow. The **rPref** package is a complete redesign of this first prototype, where we put special attention to the performance by implementing the main algorithms in C++.

There are some existing R package around Pareto optimization, e.g., **emoa** (Mersmann, 2012), **mco** (Mersmann, 2014) and **TunePareto** (Müssel et al., 2012). The two latter ones are designed for optimizing multi-dimensional functions and optimizing classification tasks, respectively. The **emoa** package does a selection of Pareto-optimal tuples from a data set similar to **rPref**. But it does not offer a semantic interface to specify the optimization goals and it is slower, as we will see in the performance evaluation later on.

The remainder of the paper is structured as follows: First, we give some motivating examples. We proceed with a formal specification of the preference model used in **rPref**. Next, we describe the implementation in our package together with some more specific examples. Finally we show some use cases visualizing preference orders.

## Motivating examples

In the following we give some examples based on the **mtcars** data set explaining the use of **rPref**. In this section we follow the introductory vignette of the package. First, we consider pure Pareto optimizations, followed by some generalizations. For all the following R code examples we assume that `library(rPref)` and `library(dplyr)` was executed before, i.e., our package and **dplyr** (Wickham and Francois, 2016) is loaded. The latter package is used for data manipulations like filtering, grouping, sorting, etc. before and after the preference selection.

### Pareto optima

In the simple example from the introduction the dimensions `mpg` and `hp` are simultaneously maximized, i.e., we are *not* interested in the *dominated* cars, which are strictly worse in at least one dimensions and worse/equal in the other dimensions. This can be realized in **rPref** with

```
p <- high(mpg) * high(hp)
psel(mtcars, p)
```

where `p` is the preference object and `psel` does the *preference selection*. The star `*` is the Pareto operator which means that both goals should be optimized simultaneously.

We can add a third dimension like minimizing the 1/4 mile time of a car, which is another practical criterion for the superiority of a car. Additional to the preference selection via `psel`, preference objects can be associated with data sets and then processed via `peval` (*preference evaluation*). For example,

```
p <- high(mpg, df = mtcars) * high(hp) * low(qsec)
creates a 3-dimensional Pareto preference which is associated with mtcars. The string output of p is:
> p
[Preference] high(mpg) * high(hp) * low(qsec)
* associated data source: data.frame "mtcars" [32 x 11]
```

We run the preference selection and select the relevant columns, where the selection is done using `select` from the **dplyr** package:

```
> select(peval(p), mpg, hp, qsec)
  mpg  hp  qsec
Mazda RX4    21.0 110 16.46
Merc 450SE   16.4 180 17.40
Merc 450SL   17.3 180 17.60
Fiat 128     32.4  66 19.47
Toyota Corolla 33.9  65 19.90
Porsche 914-2 26.0  91 16.70
Lotus Europa  30.4 113 16.90
Ford Pantera L 15.8 264 14.50
Ferrari Dino   19.7 175 15.50
Maserati Bora 15.0 335 14.60
```

All these cars have quite different values for the optimization goals `mpg`, `hp` and `qsec`, but no car is Pareto-dominant over another car in the result set. Hence all these results are *optimal compromises* without using a scoring function weighting the relative importance of the different criteria.

Using `pse1` instead of `peval` we can evaluate the preference on another data set (which does not change the association of `p`). We can first pick all cars with automatic transmission (`am == 0`) and then get the Pareto optima using `pse1` and the chain operator `%>%` (which is from **dplyr**):

```
mtcars %>% filter(am == 0) %>% pse1(p)
```

## Lexicographical order

Database preferences allow some generalizations of Skyline queries like combining the Pareto order with the lexicographical order. Assume we prefer cars with manual transmission (`am == 0`). If two cars are equivalent according to this criterion, then the higher number of gears should be the decisive criterion. This is known as the lexicographical order and can be realized in **rPref** with

```
p <- true(am == 1) & high(gear)
```

where `true` is a Boolean preference, where those tuples are preferred fulfilling the logical condition. The `&` operator is the non-commutative *Prioritization* creating a lexicographical order. Symbol and wording are taken from Kießling (2002) and Mandl et al. (2015).

The *base preferences* `high`, `low` and `true` accept arbitrary arithmetic (and accordingly logical, for `true`) expressions. For example, we can Pareto-combine `p` with a wish for a high power per cylinder ratio. Before doing the preference selection, we restrict our attention to the relevant columns:

```
> mtcars0 <- select(mtcars, am, gear, hp, cyl)
> p <- p * high(hp/cyl)
> pse1(mtcars0, p)
  am gear  hp cyl
Maserati Bora 1    5 335   8
```

Here the two goals of the lexicographical order, as defined above, and the high `hp/cyl` ratio are simultaneously optimized.

## Top-k selections

In the above preference selection we just have one Pareto-optimal tuple for the data set `mtcars`. Probably we are also interested in the tuples slightly worse than the optimum. **rPref** offers a top-*k* preference selection, iterating the preference selection on the remainder on the data set until *k* tuples are returned. To get the three best tuples we use:

```
> psel(mtcars0, p, top = 3)
   am gear hp cyl .level
Maserati Bora  1    5 335  8      1
Ford Pantera L 1    5 264  8      2
Duster 360     0    3 245  8      3
```

Additionally the column `.level` is added to the result, which is the number of iterations needed to get this tuple. The  $i$ -th level of a Skyline is also called *the i-th stratum*. We see that the first three tuples have levels  $\{1, 2, 3\}$ . The `top` parameter produces a nondeterministic cut, i.e., there could be more tuples in the third level which we do not see in the result above. To avoid this, we use the `at_least` parameter, returning all tuples from the last level and avoiding the cut:

```
> psel(mtcars0, p, at_least = 3)
   am gear hp cyl .level
Maserati Bora  1    5 335  8      1
Ford Pantera L 1    5 264  8      2
Duster 360     0    3 245  8      3
Camaro Z28     0    3 245  8      3
Ferrari Dino   1    5 175  6      3
```

Additionally there is a `top_level` parameter which allows to explicitly state the number of iterations. The preference selection with `top_level = 3` is identical to the statement above in this case, because just one tuple resides in each of the levels 1 and 2.

## Formal background

Now we will formally specify how strict orders are constructed from the preference language available in **rPref**. We mainly adapt from the formal framework from Kießling (2002) and the EXASolution implementation described in Mandl et al. (2015). An extensive consideration of theoretical aspects of this framework is given in Roocks (2016a).

For a given data set  $D$ , a preference  $p$  is associated with a strict order  $<_p$  (irreflexive and transitive) on the tuples in  $D$ . The predicate  $t <_p t'$  is interpreted as  $t'$  is better than  $t$ . A preference  $p$  is also associated with an equivalence relation  $=_p$ , modelling which tuples are equivalent for this preference. In the following,  $expr$  is an expression and  $\text{eval}(expr, t)$  is the evaluation of  $expr$  over a tuple  $t \in D$ .

### Base preferences

We define the following base preferences constructs:

- $\text{low}(expr)$ : Here  $expr$  must evaluate to a numerical value. For all  $t, t' \in D$  we have

$$\begin{aligned} t <_{\text{low}(expr)} t' &\Leftrightarrow \text{eval}(expr, t') < \text{eval}(expr, t), \\ t =_{\text{low}(expr)} t' &\Leftrightarrow \text{eval}(expr, t') = \text{eval}(expr, t), \end{aligned}$$

i.e., smaller values are preferred. Tuples with identical values are considered to be equivalent.

- $\text{high}(expr)$ : Analogously to  $\text{low}$ , larger values are preferred.
- $\text{true}(expr)$ : Here  $expr$  must evaluate to a logical value. Analogously to  $\text{high}$ , where logical values are converted to numerical ones ( $\text{false} \mapsto 0$ ,  $\text{true} \mapsto 1$ ). True values are preferred.

### Complex preferences

Building on these base constructs we define complex compositions of preferences. The formal symbols for these *preference operators* are taken from Kießling (2002). In the following  $p, q$  may either be a base preference or a complex preference. For tuples  $t, t' \in D$  we define the short hand notation

$$t \leq_p t' \Leftrightarrow t =_p t' \vee t <_p t'.$$

- The Pareto operator  $\otimes$  (“better in one dimension, better/equal in the other one”) is given by

$$\begin{aligned} t <_{p \otimes q} t' &\Leftrightarrow (t \leq_p t' \wedge t <_q t') \vee \\ &(t <_p t' \wedge t \leq_q t'). \end{aligned}$$

- The prioritisation & (lexicographical order) is defined by

$$t <_{p \& q} t' \Leftrightarrow t <_p t' \vee (t =_p t' \wedge t <_q t') .$$

- For the intersection preference  $\blacklozenge$  (“better in all dimensions”, product order) we have

$$t <_{p \blacklozenge q} t' \Leftrightarrow t <_p t' \wedge t <_q t' .$$

For all operators  $\star \in \{\otimes, \&, \blacklozenge\}$  we define that the resulting associated equivalence relation is given by the product of the equivalence relations, i.e.,  $t =_{p \star q} t' \Leftrightarrow t =_p t' \wedge t =_q t'$ .

From these definitions we see that all operators are associative. Moreover  $\otimes$  and  $\blacklozenge$  are commutative operators. All induced orders  $<_p$  are irreflexive and transitive. For the  $n$ -ary Pareto preference we can infer the representation

$$t <_{p_1 \otimes p_2 \otimes \dots \otimes p_n} t' \Leftrightarrow \exists i : t <_{p_i} t' \wedge \forall i : t \leq_{p_i} t' .$$

When all  $p_i$  preferences are low or high base preferences, then  $p_1 \otimes p_2 \otimes \dots \otimes p_n$  specifies a Skyline in the sense of Börzsönyi et al. (2001).

Finally there is the converse preference  $p^{-1}$ , reversing the induced order, given by

$$t <_{p^{-1}} t' \Leftrightarrow t' <_p t \quad \text{and} \quad t =_{p^{-1}} t' \Leftrightarrow t =_p t' .$$

## Preference evaluation

The most important function to apply these preference objects to data sets is the *preference selection*, selecting the not dominated tuples. For a data set  $D$  we define

$$\max_{<_p} D := \{t \in D \mid \nexists t' \in D : t <_p t'\} .$$

Next, we specify the *level* for all tuples in a data set  $D$  as the number of iterations of  $\max_{<_p}$  being required to retrieve this tuple. To this end we recursively define the disjoint sets  $L_i$  by

$$\begin{aligned} L_1 &:= \max_{<_p} (D) , \\ L_i &:= \max_{<_p} \left( D \setminus \bigcup_{j=1}^{i-1} L_j \right) \quad \text{for } i > 1 . \end{aligned} \tag{L}$$

We say that tuple  $t$  has level  $i$  if and only if  $t \in L_i$ . This quantifies how far a tuple is away from the optimum in a given data set. This is the key concept to do top- $k$  preference selections, where one is interested not only in the Pareto optimum but wants to obtain the  $k$  best tuples according to the preference order.

To investigate and visualize preference orders on smaller data sets, *Hasse diagrams* are a useful tool. Formally the set of edges of a Hasse diagram for a preference  $p$  on a data set  $D$  is given by the transitive reduction

$$\text{HassDiag}(D, <_p) := \{(t, t') \in D \times D \mid t <_p t' \wedge \nexists t'' \in D : t <_p t'' <_p t'\} . \tag{H}$$

## Implementation in rPref

In the following we describe how the formal framework from the section above is realized within our package.

### Preference objects

The base preferences `low`, `high` and `true` expect the (unquoted) expression to be evaluated as their argument, e.g., `high(hp/wt)` for searching for “maximal power per weight” on the `mtcars` data set.

The complex preferences are realized by overloading arithmetic operators. The Pareto operator  $\otimes$  is encoded with the multiplication operator  $*$ , the prioritization  $\&$  keeps its symbol  $\&$  and the intersection preference  $\blacklozenge$  is called with the operator  $|$ . The reverse preference  $(\cdot)^{-1}$  is implemented with an unary minus operator.

In `rPref` all the language elements of the Skyline feature from the EXASolution query language are supported. For example, a query on the `mtcars` data set like

... PREFERRING INVERSE ( $\text{wt} > \text{AVG}(\text{wt})$ ) PRIOR TO (HIGH  $\text{qsec}$  PLUS LOW ( $\text{hp}/\text{wt}$ )))

can be translated (manually) to the following R code:

```
p <- -(true(wt > mean(wt)) & (high(qsec) * low(hp/wt)))
```

Note that database specific function calls in the expressions (like `avg(wt)` to calculate the mean value of the weight column) have to be replaced by their corresponding R functions (here `mean(wt)`). The inverse translation of `p` to the EXASolution query string can be done with `show.query(p)` in `rPref` (but `mean` → `avg` has to be done manually).

Internally preference objects are S4 classes. This allows operator overloading to realize easy readable complex preference terms. Some generic methods are overloaded for preference objects, e.g., `as.expression(p)` returns the expression of a preference. For complex preferences, `length(p)` gives us the number of base preferences, and base preferences have length 1.

## Programming in rPref

For the three base preference constructors there are variants `low_`, `high_` and `true_`, which expect calls or strings as argument, i.e., `low(a)` is equivalent to `low_('a')`. This can be used to implement user defined preference constructors, e.g., the counterpart to the `true` preference negating the logical expression. This base preference `false(expr)` prefers tuples, where `expr` evaluates to FALSE values.

```
false <- function(x) true_(call('!', substitute(x)))
```

Internally the expressions are mapped to *lazy* expressions using the `lazyeval` package (Wickham, 2016). It is also possible to call these constructors with lazy expressions, e.g. `low_(as.lazy('mpg'))`.

Additionally, there is the special preference object `empty()` which acts as a neutral element for all complex operators. This can be useful as an initial element for generating preference terms using the higher-order function `Reduce` from R. Consider the following example, where a set of attributes of `mtcars`, given as a character vector, shall be simultaneously maximized, i.e., a Pareto composition of high preferences shall be generated.

```
> sky_att <- c('mpg', 'hp', 'cyl')
> p <- Reduce('*', lapply(sky_att, high_), empty())
> p
[Preference] high(mpg) * high(hp) * high(cyl)
```

## Preference selection

The main function for evaluating preferences is `psel(df, p)` (for **preference selection**), returning the optimal tuples of a data set w.r.t. a preference. It expects a data frame `df` and a preference object `p`. For example, using `p` from above, `psel(mtcars, p)` returns the optimal cars having a low fuel consumption, a high horsepower and a high number of cylinders.

Note that the preference selection of `rPref` is restricted to data frames. Working directly on the database, e.g., using `tbl` on a database connection from `dplyr`, is currently not supported. To our knowledge there are no open database management systems directly supporting the Skyline operator. Hence an implementation to work remotely on a database could either convert the `tbl` object to a data frame (i.e., gather the entire data set), or generate a Skyline query by rewriting the query into standard SQL, as done in Kießling and Köstler (2002). While an automatic conversion would be a misleading use of such an SQL table object, the rewriting approach would result in a very bad performance. In our experiences, usual database optimizers cannot process rewritten Skyline queries within reasonable costs.

As mentioned in the introductory examples, top- $k$  queries are also supported. Using the optional parameter `top = k` in `psel` the  $k$  best tuples are returned. For a formal specification we define, using the levels sets  $L_i$  from equation (L),

$$L^{(j)} := \bigcup_{i=1}^j L_i ,$$

and iterate over  $j \in \{1, 2, \dots\}$  until  $|L^{(j)}| \geq k$  is fulfilled. If  $|L^{(j)}| > k$  then some level- $j$  tuples are non-deterministically cut off such that  $k$  tuples are left. Using a top- $k$  selection the level values of the tuples are also provided in an additional column `.level`. By using `top = nrow(df)` we get the level values of all tuples from a data frame `df`.

There are also deterministic variants of the top- $k$  preference selection accessible by using one of the optional parameters `at_least` or `top_level` of the `psel` function. When `at_least = k` is set, all

tuples in  $L^{(j)}$  are returned where  $j$  is the minimal value fulfilling  $|L^{(j)}| > k$  (i.e., top- $k$  without cut-off). Finally `top_level = k` simply returns  $L^{(k)}$ , i.e., the first  $k$  levels.

### Grouped preference evaluation

Grouped preferences are also supported. We rely on the grouping functionality from the `dplyr` package. To get the Pareto-optimal cars with high weight and fast acceleration (i.e., low 1/4 mile time) for each group with a different number of cylinders we state:

```
grouped_cars <- group_by(mtcars, cyl)
opt_cars <- psel(grouped_cars, high(wt) * low(qsec))
```

As this again returns a grouped data frame, we can get the cardinality of each Pareto-optimal set by

```
> as.data.frame(summarize(opt_cars, n()))
   cyl n()
1    4   3
2    6   4
3    8   7
```

For grouped data sets, the optional `top`, `at_least` and `top_level` arguments are considered for each group separately. This means that a preference selection with `top = 4` on a data set with 3 groups, where each group contains at least 4 tuples, will return 12 tuples.

### Partial evaluation and associated data sets

Base preferences can be associated with a data set using the additional argument `df`. This association is propagated when preferences are composed. For example,

```
> p <- high(mpg, df = mtcars[1:10,]) * low(wt)
> p
[Preference] high(mpg) * low(wt)
  * associated data source: data.frame "mtcars[1:10, ]" [10 x 11]
```

creates a preference object `p` with a subset of `mtcars` as associated data source.

Associating a data frame also invokes a partial evaluation of all literals which are not column names of the data frame. If addressed with `df$column` the columns are also partially evaluated. For example, we can create a preference maximizing the normalized sum of `hp` and `mpg` by

```
> p <- high(mpg/max(mtcars$mpg) + hp/max(mtcars$hp), df = mtcars)
> p
[Preference] high(mpg/33.9 + hp/335)
  * associated data source: data.frame "mtcars" [32 x 11]
```

and we directly see the summed values within the preference expression.

To evaluate a preference on the associated data set, we can call `peval(p)` (for preference **evaluation**). Alternatively, by using `psel(df, p)` we can use another data source without overwriting the associated data frame of a preference object. The association can be changed using `assoc.df(p) <- df`, assigning a new data frame `df` to `p`, where `assoc.df(p)` shows us the current data source.

### Algorithms and performance

The default algorithm used for the preference selection is BNL from Börzsönyi et al. (2001). If the given preference is a pure Pareto composition, i.e.,  $p = p_1 \otimes \dots \otimes p_n$  where all  $p_i$  are base preferences, then the Scalagon algorithm from Endres et al. (2015) is used which is faster than BNL in most cases.

A further performance gain is possible by parallelization. A simple approach to speed up the preference selection on multi-processor systems is to split the calculation over the  $n$  different cores using the formula

$$\max_{\leq_p} D = \max_{\leq_p} \left( \max_{\leq_p} D_1 \cup \dots \cup \max_{\leq_p} D_n \right) \quad \text{where} \quad D = \bigcup_{i=1}^n D_i.$$

When the option `rPref.parallel` is set to `TRUE` then  $D$  is split in  $n$  parts and the calculation is done in  $n$  threads. This number can be specified using the option `rPref.parallel.threads`. By default,  $n$  is the number of processor cores. The  $\max_{\leq_p} D_i$  calculation is done in parallel, while the final step of

merging the maxima is done on one core. This is implemented in our package using `RcppParallel` from Allaire et al. (2016). By default, parallel calculation is not used. We will show that parallelization can lead to a (slight) performance gain.

As mentioned in the introduction, there is already the `emoa` package (Mersmann, 2012) which is also suited for calculating Pareto optima. There, all dimensions of the given data set are simultaneously minimized, i.e., there is no semantic preference model in that package.

In the following we will compare the performance of `emoa` and the (parallel) preference selection of the `rPref` package. For the benchmarks will use 2-dimensional weakly anti-correlated data sets (correlation value of  $-0.7$ ), which is a typical example for Skyline computation. The data generator reads as follows:

```
gen_data <- function(N, cor) {
  rndvals <- matrix(runif(2 * N), N, 2)
  corvals <- runif(N)
  corvals <- cbind(corvals, 1 - corvals)
  df <- as.data.frame((1 - abs(cor)) * rndvals + abs(cor) * (1 - corvals))
  colnames(df) <- c('x', 'y')
  return(df)
}
```

First we compare the results of `rPref` and `emoa` to ensure that both do the same. In the latter package, `nondominated_points` is the equivalent to `pse1` in our package. All dimensions are minimized simultaneously, corresponding to the preference  $\text{low}(x) * \text{low}(y)$ , and `nondominated_points` expects a matrix where tuples are columns. We calculate the Pareto optima with both variants and convince ourselves that the results are identical. As the order of the tuples does not matter, i.e., the routines may return a different sorted result, we use the `setequal` function from base R for comparison.

```
> df <- gen_data(1E6, -0.7)
> result1 <- pse1(df, low(x) * low(y))
> result2 <- as.data.frame(t(nondominated_points(t(as.matrix(df)))))
> setequal(result1, result2)
TRUE
```

Next, we compare the run times. Using the option `rPref.parallel` we can control if `rPref` uses multi-threaded calculation. For each test we generate 10 different data sets with  $5 \cdot 10^6$  tuples with a correlation value of  $-0.7$  and measure the run times to calculate the Pareto optima:

```
options(rPref.parallel = FALSE)
time_rpref_serial <- vapply(1:10, function(i)
  system.time({ pse1(gen_data(5E6, -0.7), low(x) * low(y)) })[3], 0)

options(rPref.parallel = TRUE)
time_rpref_parallel <- vapply(1:10, function(i)
  system.time({ pse1(gen_data(5E6, -0.7), low(x) * low(y)) })[3], 0)

time_emoa <- vapply(1:10, function(i)
  system.time({ nondominated_points(t(as.matrix(gen_data(5E6, -0.7)))) })[3], 0)
```

In Table 1 we summarize the results. We see a slight performance gain by parallelization in `rPref` and find out that `emoa` is slower.

Test setting	Run time (seconds)
<code>rPref</code> , serial	$1.30 \pm 0.03$
<code>rPref</code> , parallel	$1.25 \pm 0.03$
<code>emoa</code>	$8.29 \pm 0.29$

**Table 1:** Comparison of run times between `emoa` and `rPref` (mean value and standard deviation of 10 iterations) on a weakly anti-correlated data set containing  $5 \cdot 10^6$  tuples.

## Visualization use cases

In the following we show two different approaches to visualize a preference order on the entire data set. Both approaches implicitly assume a sufficiently small data set and are primarily intended to get a

better understanding of a (potentially complex) preference operating on a small number of tuples.

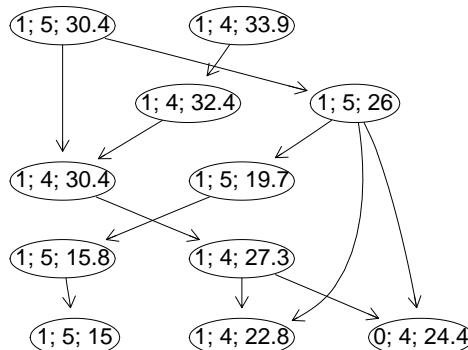
### Hasse diagram

The Hasse diagram as defined in equation (H), also called *Better-Than-Graph (BTG)* in the context of preferences, contains the transitive reduction of all better-than-relations. We can retrieve this in **rPref** with `get_hasse_diag` as an  $n \times 2$  matrix, where  $n$  is the number of better-than-relations. To visualize this diagram we rely of the Graphviz/dot layouter from [Ellson et al. \(2004\)](#). There is the R package **Rgraphviz** ([Hansen et al., 2016](#)), only available on Bioconductor, which is used by the `plot_btg` function if available (**Rgraphviz** is suggested by our package).

Let us consider again the preference from the introduction combining a Prioritization and a Pareto composition. We create appropriate labels showing the relevant values and plot the Better-Than-Graph. We restrict our attention to the first five Skyline levels.

```
p <- (true(am == 1) & high(gear)) * high(mpg)
df <- psel(mtcars, p, top_level = 5)
labels <- with(df, paste(am, gear, mpg, sep = ' ; '))
plot_btg(df, p, labels)
```

We get the diagram depicted in Figure 2. The row of a tuple node in the diagram, counted from top to bottom, coincides with the level-value as defined in equation (L). Here we have exactly 5 rows as we restricted the data set to be plotted with `top_level = 5`. The correspondence between level and row is ensured by **rPref** if the parameter `levelwise` of the `plot_btg` function is TRUE (which is the default). If this parameter is FALSE the vertical arrangement is subject to the dot layouter and it is not levelwise in general. In any case, the edges will point from top to bottom (unless the entire graph is flipped using the parameter `flip.edges`). Additionally we can get the dot source code of the graph with `get_btg_dot`. This is useful for using an external dot interpreter (we will not go into details here).



**Figure 2:** The Better-Than-Graph for the above example. The directed edges show the “is better than” relations and the labels denote the `am`, `gear` and `mpg` values for each tuple. The rows correspond to the levels of the tuples.

If **Rgraphviz** is not available, the **igraph** package ([Csardi and Nepusz, 2006](#)) is used alternatively. But **igraph** has, to our knowledge, no layouting method suited for general strict orders. The edges will not point from top to bottom in general and hence the diagram will look not very pretty.

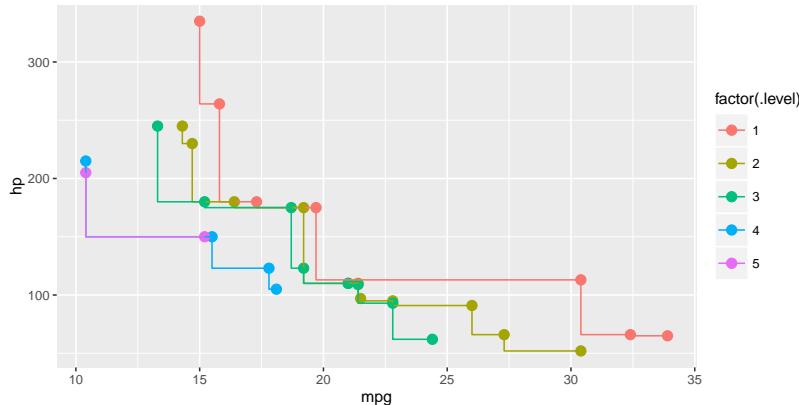
### Pareto front lines

In Figure 1 we already saw the points representing the Pareto set. To show more clearly how the preference orders the given tuples we do the following:

1. Retrieve the levels of all tuples,
2. plot them in a different color,
3. and show the Pareto front line for each level.

The Pareto front line is a stair-shaped line marking the border of the dominance area of these tuples. We can simply plot that line using `geom_step` from the **ggplot2** package from [Wickham \(2009\)](#). The following code returns the diagram in Figure 3.

```
res <- psel(mtcars, high(mpg) * high(hp), top = nrow(mtcars))
ggplot(res, aes(x = mpg, y = hp, color = factor(.level))) +
  geom_point(size = 3) + geom_step(direction = 'vh')
```



**Figure 3:** The Pareto front lines for each level for the Pareto preference of maximal mpg and hp values.

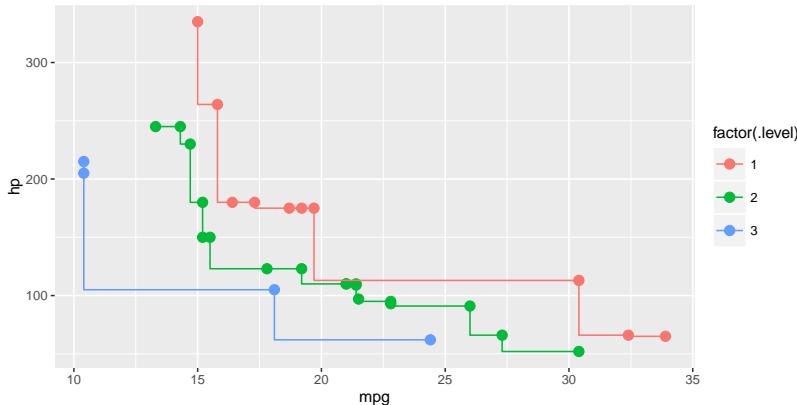
We see that the front lines in Figure 3 are overlapping at some points. This is because the Pareto order only requires a tuple to be strictly better in one dimension to dominate other tuples. For the other dimensions equivalent value are sufficient, which causes that both level-5 tuples in Figure 3 are on the Pareto front line of the level-4 tuples.

When substituting the Pareto preference by the intersection preference (`|` operator instead of `*`), where better tuples are required to be strictly better in both dimensions (i.e., the product order), there are no more overlapping front lines. We generate the corresponding diagram with the following R code, using `dplyr` for sorting the Pareto set:

```
res <- mtcars %>% psel(high(mpg) | high(hp), top = nrow(mtcars)) %>%
  arrange(mpg, -hp)
ggplot(res, aes(x = mpg, y = hp, color = factor(.level))) +
  geom_point(size = 3) + geom_step(direction = 'vh')
```

Note that we have to sort the resulting data set `res` by ascending `mpg` values and descending `hp` values to get the proper stair-shaped front line isolating the dominated area. Without that sorting we would get U-shaped lines.

The result of the `psel` function is never sorted by `rPref`, aside from that top- $k$  queries return tuples sorted by level. As all tuples within the same level are incomparable w.r.t. the preference, there is no natural order of these tuples justifying some specific ordering. Usually the order does not matter for further calculations or visualizations. Only for some visual representations (like in this example) we need an appropriate sorting, depending on the kind of representation.



**Figure 4:** The front lines for each level for the intersection preference of maximal mpg and hp values.

The result is shown in Figure 4. When compared to Figure 3, we clearly see the difference between the Pareto and intersection preference.

Note that the Pareto and intersection preference coincide in use cases where no duplicate values occur, e.g., measurement points in continuous domains.

## Conclusion

To our knowledge, **emoa** (Mersmann, 2012) is the most similar existing R package, which also computes Pareto sets. We have shown that our approach is more general and offers better performance. In addition to Pareto sets, some generalizations called *database preferences* from the database community are also provided in **rPref**. The preference semantics are adapted from the commercial implementation EXASolution Skyline, cf. Mandl et al. (2015).

We used existing approaches where possible and appropriate, e.g., **Rgraphviz** for plotting Better-Than-Graphs, **dplyr** for grouping data sets or **ggplot2** for plotting Pareto front lines. By doing so, we tried to keep the package small and focussed on its main task of specifying and computing database preferences. For the specification it supports a semantically rich preference model.

Although Pareto optima and generalizations are a hot topic in the database community since the pioneer work of Börzsönyi et al. (2001), there are no open source implementations of database management systems supporting Skylines. The large majority of papers describe research prototypes being not publicly accessible, and the only commercially available system is EXASolution. Through our work, the functionality of the “Skyline” feature of that commercial system is now fully available for the R ecosystem.

## Acknowledgement

The author thanks the two anonymous reviewers for their constructive comments and suggestions that have greatly improved the quality of the manuscript and the usability of the package.

## Bibliography

- J. Allaire, R. Francois, K. Ushey, G. Vandenbrouck, M. Geelnard, and Intel. *RcppParallel: Parallel Programming Tools for ‘Rcpp’*, 2016. URL <https://CRAN.R-project.org/package=RcppParallel>. R package version 4.3.20. [p401]
- S. Börzsönyi, D. Kossmann, and K. Stocker. The Skyline Operator. In *17th International Conference on Data Engineering*, pages 421–430, 2001. doi: 10.1109/ICDE.2001.914855. [p394, 398, 400, 404]
- G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*:1695, 2006. URL <http://igraph.sf.net>. [p402]
- J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull. Graphviz and Dynagraph – Static and Dynamic Graph Drawing Tools. In *Graph Drawing Software*, pages 127–148. Springer, 2004. [p402]
- M. Endres, P. Roocks, and W. Kießling. Scalagon: An Efficient Skyline Algorithm for All Seasons. In *DASFAA ’15: Database Systems for Advanced Applications*, pages 292–308. Springer, 2015. [p400]
- K. D. Hansen, J. Gentry, L. Long, R. Gentleman, S. Falcon, F. Hahne, and D. Sarkar. *Rgraphviz: Provides plotting capabilities for R graph objects*, 2016. R package version 2.18.0. [p402]
- W. Kießling. Foundations of Preferences in Database Systems. In *VLDB ’02: 28th International Conference on Very Large Data Bases*, pages 311–322, Hong Kong, China, 2002. [p394, 395, 396, 397]
- W. Kießling and G. Köstler. Preference SQL: Design, Implementation, Experiences. In *VLDB ’02: 28th International Conference on Very Large Data Bases*, pages 990–1001, 2002. [p394, 399]
- S. Mandl, O. Kozachuk, M. Endres, and W. Kießling. Preference Analytics in EXASolution. In *16th Conference on Database Systems for Business, Technology, and Web*, 2015. URL [http://www.btw-2015.de/res/proceedings/Hauptband/Ind/Mandl-Preference\\_Analytics\\_in\\_EXAS.pdf](http://www.btw-2015.de/res/proceedings/Hauptband/Ind/Mandl-Preference_Analytics_in_EXAS.pdf). [p394, 396, 397, 404]
- O. Mersmann. *emoa: Evolutionary Multiobjective Optimization Algorithms*, 2012. URL <https://CRAN.R-project.org/package=emoa>. R package version 0.5-0. [p395, 401, 404]
- O. Mersmann. *mco: Multiple Criteria Optimization Algorithms and Related Functions*, 2014. URL <https://CRAN.R-project.org/package=mco>. R package version 1.0-15.1. [p395]

- C. Müssel, L. Lausser, M. Maucher, and H. A. Kestler. Multi-Objective Parameter Selection for Classifiers. *Journal of Statistical Software*, 46(5):1–27, 2012. URL <http://www.jstatsoft.org/v46/i05/>. [p395]
- P. Roocks. *Relational and Algebraic Calculi for Database Preferences*. PhD thesis, University of Augsburg, 2016a. URL <https://opus.bibliothek.uni-augsburg.de/opus4/frontdoor/index/index/docId/3760>. [p397]
- P. Roocks. *rPref: Database Preferences and Skyline Computation*, 2016b. URL <http://www.p-roocks.de/rpref>. R package version 1.2. [p395]
- P. Roocks and W. Kießling. R-Pref: Rapid Prototyping of Database Preference Queries in R. In *DATA '13: 2nd International Conference on Data Management Technologies and Applications*, pages 104–111, 2013. [p395]
- H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. ISBN 978-0-387-98140-6. URL <http://ggplot2.org>. [p402]
- H. Wickham. *lazyeval: Lazy (Non-Standard) Evaluation*, 2016. URL <https://CRAN.R-project.org/package=lazyeval>. R package version 0.2.0. [p399]
- H. Wickham and R. Francois. *dplyr: A Grammar of Data Manipulation*, 2016. URL <https://CRAN.R-project.org/package=dplyr>. R package version 0.5.0. [p395]

Patrick Roocks  
University of Augsburg  
Institute of Computer Science  
Universitätsstr. 6a  
86159 Augsburg  
Germany  
[roocks@informatik.uni-augsburg.de](mailto:roocks@informatik.uni-augsburg.de)

# micompr: An R Package for Multivariate Independent Comparison of Observations

by Nuno Fachada, João Rodrigues, Vitor V. Lopes, Rui C. Martins and Agostinho C. Rosa

**Abstract** The R package **micompr** implements a procedure for assessing if two or more multivariate samples are drawn from the same distribution. The procedure uses principal component analysis to convert multivariate observations into a set of linearly uncorrelated statistical measures, which are then compared using a number of statistical methods. This technique is independent of the distributional properties of samples and automatically selects features that best explain their differences. The procedure is appropriate for comparing samples of time series, images, spectrometric measures or similar high-dimension multivariate observations.

## Introduction

The aim of this paper is to present the **micompr** package for R (Fachada, 2016), which implements a procedure for comparing multivariate samples associated with different factor levels or groups. The research goal is to differentiate among pre-specified, well-defined classes or groups of sampling entities generating highly multivariate observations in which the dimensions or dependent variables are correlated, and to test for significant differences among groups. The procedure uses principal component analysis (PCA; Jolliffe, 2002) to convert multivariate observations into a set of linearly uncorrelated statistical measures, which are then compared using a number of statistical methods, such as hypothesis tests and score plots.

This technique has several desirable attributes: a) it automatically selects observation features that best explain sample differences; b) it does not depend on the distributional properties of samples; and, c) it simplifies the researchers' work, as it can be used directly on multivariate observations. The procedure is appropriate for comparing samples of multivariate observations with highly correlated and similar scale dimensions, such as time series, images or spectrometric measures. However, the **micompr** package goes one step further by also accommodating the simultaneous comparison of multiple observation types, i.e., multiple *outputs* from a given "system". In this context, a "system" can be defined as an abstract entity capable of generating one or more stochastic data streams, i.e., *outputs*. Thus, **micompr** can determine if two or more instances of such a "system" display the same behavior by comparing observations of their *outputs*.

The remainder of this paper is organized as follows. First, in Section [Testing for significant differences in multivariate samples](#), commonly used techniques for differentiating samples of multivariate observations are discussed. The methodology employed by **micompr** is described in Section [Independent comparison of multivariate observations](#). Section [The micompr package](#) introduces the software and its architecture, namely the available objects and functions. Several concrete application examples, and how the "system"-*output* terminology fits each one, are presented in Section [Examples](#). The paper closes with Section [Summary](#), in which the overall approach and the R package are summarized.

## Testing for significant differences in multivariate samples

Two-sample or multi-sample hypothesis tests are commonly used for assessing statistically dissimilarity in univariate samples, i.e., samples composed of scalar observations. If samples are drawn from normally distributed populations, the *t* (two samples) and ANOVA (*n*-samples) tests are adequate (Montgomery and Rungé, 2014). Non-parametric tests are more appropriate if population normality cannot be assumed. The Mann-Whitney *U* test (Gibbons and Chakraborti, 2010) and the Kolmogorov-Smirnov test (Massey Jr, 1951) are typically employed for comparing two samples. The Kruskal-Wallis test (Kruskal and Wallis, 1952) extends the former for the *n*-sample case.

Multivariate analysis of variance (MANOVA; Krzanowski, 1988; Tabachnick and Fidell, 2013) can be used as a statistical test for comparing multivariate samples. In this context, samples are composed of multi-dimensional observations, for which each dimension is a dependent variable (DV). However, MANOVA is not appropriate for cases with highly correlated DVs and when the number of DVs or dimensions is higher than the number of observations. Additionally, MANOVA is a parametric method which makes a series of assumptions on the underlying data which are not always met in practice.

Analogous non-parametric tests exist, but they are not as widespread and are commonly oriented towards specific research topics. Multiple Response Permutation Procedures (MRPP; Mielke Jr et al., 1976) and associated permutation-based methods, such as ANOSIM (Clarke, 1993) or permutational MANOVA (Anderson, 2001), test for differences in distances between observations from each group. These tests are implemented in the `vegan` package (Oksanen et al., 2016), typically used in Ecology studies. The `Blossom` package (Talbert et al., 2016) also provides MRPP and other distance-function based permutation tests. In a similar note, Székely and Rizzo (2004) proposed a multi-sample test for equality of multivariate distributions based on the Euclidean distance between sample elements. The test statistic belongs to a class of multivariate statistics (energy statistics) proposed by the same authors. The `energy` package (Rizzo and Szekely, 2016) implements this test and other energy statistics-related functionality. The cross-match test is another distance-based test (Rosenbaum, 2005), with the particularity of not requiring permutation techniques. It is available for the R environment via the `crossmatch` package (Heller et al., 2012). In turn, the `cramer` package (Franz, 2014) provides a multivariate implementation of the non-parametric two-sample Cramér test, originally proposed by Baringhaus and Franz (2004). The critical value of the test can be determined with bootstrap (ordinary or permutation-based) or eigenvalue-based methods. Another test which avoids permutation was proposed by Duong et al. (2012). It is a kernel-based test, originally developed to assess the statistical differences between two cellular topologies. The test is implemented in the `ks` package (Duong, 2016), although limited to six-dimensional data.

An alternative to multivariate tests is to extract a number of statistical summaries (e.g., averages or extreme values) or specific points from individual multivariate observations, and then perform a univariate test for each summary measure. This approach also has its issues: a) it does not provide a single answer, i.e., it will yield as many  $p$ -values as there are summary measures; b) the choice of summary is problem-dependent and empirically driven, and consequently, error-prone, in the sense that the chosen summaries may not be representative of the original multivariate observations. While only careful analysis can minimize the latter issue, the former problem can be addressed with a multiple comparison adjustment procedure, such as the Bonferroni correction (Shaffer, 1995).

## Independent comparison of multivariate observations

Given a matrix  $\mathbf{X}_{(n \times m)}$  of  $n$  observations and  $m$  variables or dimensions, PCA can be used to obtain matrix  $\mathbf{T}_{(n \times r)}$ , which is the representation of  $\mathbf{X}_{(n \times m)}$  in the principal components (PCs) space, and vector  $\lambda_{(1 \times r)}$ , containing the eigenvalues of the covariance matrix of the original mean-centered data. Rows of  $\mathbf{T}$  directly correspond to the observations of the original samples, while columns correspond to PCs. Columns are ordered by decreasing variance, i.e., the first column corresponds to the first PC, and so on. Variance is given by the eigenvalues in vector  $\lambda$ , which are likewise ordered, each eigenvalue corresponding to the variance of the columns of  $\mathbf{T}$ . The percentage of variance explained by each PC can be obtained by dividing the respective eigenvalue with the sum of all eigenvalues. At this stage, PCA-reshaped observations associated with different groups can be compared using statistical methods. More specifically, hypothesis tests can be used to check if the sample projections on the PC space are drawn from populations with the same distribution. There are two possible lines of action:

1. Apply a MANOVA test to the samples, where each observation has  $q$ -dimensions, corresponding to the first  $q$  PCs (dimensions) such that these explain a user-defined minimum percentage of variance.
2. Apply a univariate test to observations in individual PCs. Possible tests include the  $t$ -test and the Mann-Whitney  $U$  test for comparing two samples, or ANOVA and Kruskal-Wallis test, which are the respective parametric and non-parametric versions for comparing more than two samples.

The MANOVA test yields a single  $p$ -value from the simultaneous comparison of observations along multiple PCs. An equally succinct answer can be obtained with a univariate test using the Bonferroni correction or a similar method for handling  $p$ -values from multiple comparisons. However, both approaches will not prioritize dimensions, even though the first PCs can be more important for characterizing an output, as they explain more variance. In the univariate case one can prioritize PCs according to the explained variance using the weighted Bonferroni procedure (Rosenthal and Rubin, 1983).

Conclusions concerning whether samples are statistically similar can be drawn by analyzing the  $p$ -values produced by the employed statistical tests, which should be below the typical 1% or 5% when samples are significantly different. In such case, less PCs should be required to explain the same percentage of variance than when, in the same context, no significant differences are found. The scatter

plot of the first two PC dimensions can also provide visual, although subjective feedback on sample similarity.

While the procedure is most appropriate for comparing multivariate observations with highly correlated and similar scale dimensions, assessing the similarity of “systems” with multiple outputs of different scales is also possible. The simplest approach would be to apply the proposed method to samples of individual outputs, and analyze the results in a multiple comparison context. An alternative approach consists in concatenating, observation-wise, all outputs, centered and scaled to the same order of magnitude, thus reducing a “system” with  $k$  outputs to a “system” with one output. The proposed method would then be applied to samples composed of concatenated observations encompassing the existing outputs. This technique is described in detail by [Fachada et al. \(2017\)](#) in the context of comparing simulation outputs.

## The **micompr** package

### Overview

The **micompr** package for the R statistical computing environment implements the methodology proposed in Section [Independent comparison of multivariate observations](#). Here we describe version 1.0.1 of the package, which is available at <https://CRAN.R-project.org/package=micompr/>. The development version is hosted at <https://github.com/fakenmc/micompr>. The package is covered by the MIT license.

The **micompr** package is built upon two functions, `cmpoutput` and `micomp`. The former compares two or more samples of multivariate observations collected from one output. The latter is used for comparing multiple outputs and/or comparing outputs in multiple contexts. `grpoutputs` is a helper function for loading data from two or more set of files and preparing the data to be processed by the `cmpoutput` and/or `micomp` functions. `assumptions` is a generic function for assessing the assumptions of the parametric tests used in sample comparisons.

### Architecture

**micompr** is structured according to the S3 object-oriented system. The `cmpoutput`, `micomp` and `grpoutputs` functions produce S3 objects with the same name. The package also provides the generic function `assumptions`, and two concrete implementations of methods for “`cmpoutput`” and “`micomp`” objects, which return objects of class “`assumptions_cmpoutput`” and “`assumptions_micomp`”, respectively. All classes have method implementations of the common S3 generic functions `print`, `summary` and `plot`. Additionally, method implementations of the `toLatex` function, for producing user-configurable L<sup>A</sup>T<sub>E</sub>X tables with information about the performed comparisons, are provided for “`cmpoutput`” and “`micomp`” objects.

### `grpoutputs`

This function groups outputs from sets of files containing multiple observations into samples. It returns a list of output matrices, ready to be processed by `micomp`. Alternatively, individual output matrices can be handled by `cmpoutput`. Separate files contain one multivariate observation of one or more outputs, one column per output, one row per dimension or variable. Each specified set of files is associated with a different sample. The function is also able to create an additional concatenated output, composed from the centered and scaled original outputs.

The `plot` method for “`grpoutputs`” objects shows  $k$  plots, one per output. Output observations are plotted on top of each other, with different samples colored distinctively. The `summary` method for “`grpoutputs`” objects returns a list containing two elements: a) the  $n \times m$  dimensions of each output matrix; and, b) the sizes of individual samples. The `print` method for “`grpoutputs`” objects simply outputs the summary in a more adequate presentation format.

### `cmpoutput`

The `cmpoutput` function is at the core of **micompr**. It compares two or more samples of multivariate observations using the technique described in Section [Independent comparison of multivariate observations](#). It accepts an output matrix,  $\mathbf{X}_{(n \times m)}$ , with  $n$  observations and  $m$  variables or dimensions, a factor vector of length  $n$ , specifying the group associated with each observation, and a vector of explained variances with which to determine the number of PCs to use in the MANOVA test (alternatively, the number of PCs can also be directly specified). The function returns matrix  $\mathbf{T}_{(n \times r)}$  of

PCA scores and the *p*-values for the performed statistical tests, namely: a) a MANOVA test for each explained variance (or number of PCs); and, b) parametric (*t*-test or ANOVA) and non-parametric (Mann-Whitney or Kruskal-Wallis) univariate tests for each PC. Regarding the latter, the function also returns *p*-values adjusted with the weighted Bonferroni correction, using the percentages of explained variance by PC as weights.

The plot implementation for “*cmpoutput*” objects shows six sub-plots, namely a scatter plot with the PC1 vs. PC2 scores and five bar plots. The horizontal scale of the latter consists of the *r* PCs, and the vertical bars represent the explained variance (one plot) or univariate parametric and non-parametric *p*-values, before and after weighted Bonferroni correction (four plots). The summary method for “*cmpoutput*” objects returns a list with the following items: a) percentage of variance explained by each PC; b) *p*-values of the MANOVA test or tests; c) *p*-values of the parametric test, per PC, before and after weighted Bonferroni correction; d) *p*-values of the non-parametric test, per PC, before and after weighted Bonferroni correction; and, e) name of the parametric and non-parametric univariate tests employed (either *t*-test and Mann-Whitney *U* test for comparing two samples, or ANOVA and Kruskal-Wallis for more than two samples). The *print* method for “*cmpoutput*” objects shows the information provided by the *summary* implementation, but the *p*-values of the univariate tests are only shown for the first PC.

### **micomp**

The *micomp* function performs one or more comparisons of multiple outputs, invoking *cmpoutput* for each comparison/output combination. It accepts a list of comparisons, where individual comparisons can have one of two configurations: a) a vector of folders and a vector of file sets containing data in the format required by *grpoutputs*, where each file set corresponds to a different sample; and, b) a “*grpoutputs*” object, passed directly. The returned objects, of class “*micomp*”, are basically two-dimensional lists of “*cmpoutput*” instances, with rows associated with individual outputs, and columns with separate comparisons.

The plot method for “*micomp*” objects shows the PC1 vs. PC2 score plots for each comparison/output combination. The summary implementation for “*micomp*” objects returns a list of comparisons, each one containing a  $a \times k$  matrix of *p*-values or number of PCs, associated with  $a \geq 6$  measures and  $k$  outputs. Four rows represent the *p*-values of the parametric and non-parametric univariate tests for the first PC, before and after weighted Bonferroni correction. The remaining pairs of rows are associated with the MANOVA test for a given percentage of variance to explain. One row shows the *p*-values, and the other displays the number of PCs required to explain the specified percentage of variance for the given output. As with other *micompr* objects, the *print* method for “*micomp*” objects also shows the summary with a better presentation.

### **assumptions**

*assumptions* is a generic function which performs a number of statistical tests concerning the assumptions of the parametric tests performed by the package functions. Implementations of this generic function exist for “*cmpoutput*” and “*micomp*” objects. The former method returns objects of class “*assumptions\_cmpoutput*” containing results of the assumptions tests for a single output comparison. The latter returns a two-dimensional list of “*assumptions\_cmpoutput*” objects, with rows associated to individual outputs, and columns to separate comparisons. These objects are tagged with the “*assumptions\_micomp*” class attribute.

The following assumptions are checked: a) observations are normally distributed within each sample along individual PCs (Shapiro-Wilk test); b) observations follow a multivariate normal distribution within each sample for all PCs used in MANOVA (Royston test); c) samples have homogeneous variance along individual PCs (Bartlett test); and, d) samples have homogeneous covariance matrices for all PCs used in MANOVA (Box’s *M* test). Assumptions a) and c) should be verified for the parametric test applied to each PC, while assumptions b) and d) should be verified for individual MANOVA tests performed for each variance to explain (or, alternatively, for each specified number of PCs).

The plot implementations for classes “*assumptions\_cmpoutput*” and “*assumptions\_micomp*” display a number of bar plots for the *p*-values of the performed tests. These are more detailed for “*assumptions\_cmpoutput*” objects, showing the *p*-values of the univariate test for all PCs. For “*assumptions\_micomp*” objects, one bar plot is shown per output/comparison combination, but in the case of the univariate tests only the *p*-values of the first PC are shown. Implementations of *summary* return a list of tabular data containing the *p*-values of the assumption tests. The *summary* method for “*assumptions\_cmpoutput*” objects returns a list with two matrices of *p*-values, one for the MANOVA tests, another for the univariate tests. The *summary* method for “*assumptions\_micomp*” objects follows the approach taken by the *summary* method for “*micomp*” objects, returning a list

of  $p$ -value matrices, one matrix per comparison. Rows of individual matrices correspond to the assumptions tests, and columns to outputs. The print methods for “assumptions\_cmppoutput” objects and for “assumptions\_micomp” objects again show the summary information in a printable format.

### toLatex methods for “cmppoutput” and “micomp” objects

These methods are implementations of the toLatex generic function, and convert “cmppoutput” and “micomp” objects to character vectors representing L<sup>A</sup>T<sub>E</sub>X tables. The generated tables are configurable via function arguments, with sensible defaults. Tables can present the following data for each output/comparison combination: a) number of principal components required to explain a user-specified percentage of variance; b) MANOVA  $p$ -value for a user-specified percentage of variance to explain or number of PCs; c) parametric test  $p$ -value for a given PC, before and/or after weighted Bonferroni correction; d) non-parametric test  $p$ -value for a given PC, before and/or after weighted Bonferroni correction; e) variance explained by a specific PC; and, f) a score plot with the output projection on the first two PCs.

### Other functions

The **micompr** package is bundled with additional functions whose purpose is to aid the main package methods do their job. However, some of these may be useful in other contexts.

The concat\_outputs function concatenates outputs collected from multiple observations. It accepts two arguments, namely a list of output matrices, and the centering and scaling method. Several centering and scaling methods, such as “range”, “iqrangle”, “vast” or “pareto” (Berg et al., 2006), are recognized in the second argument. The function returns an  $n \times p$  matrix of  $n$  observations with length  $p$ , which is the sum of individual output lengths. Lower-level centering and scaling of individual outputs is performed by the centerscale function, which accepts a numeric vector and returns a new vector, centered and scaled with the specified method.

The pvalf generic function formats  $p$ -values for L<sup>A</sup>T<sub>E</sub>X. A concrete default implementation is used by the **micompr** toLatex implementations if not specified otherwise. This implementation underlines and double-underlines  $p$ -values lower than 0.05 and 0.01, respectively, although these limits are configurable, and underlining can be turned off by setting both limits to zero. It is also possible to specify a limit below which  $p$ -values are capped. For example, if this limit is set to  $1 \times 10^{-5}$ , a  $p$ -value equal to  $1 \times 10^{-6}$  would be displayed as “ $< 1e^{-5}$ ”. The default method of pvalf will format  $p$ -values lower than  $5 \times 10^{-4}$  using scientific E notation, which is more compact and thus a better fit for tables.  $p$ -values between  $5 \times 10^{-4}$  and 1 are formatted using regular decimal notation with three decimal places. This aspect is not configurable. However, another implementation of pvalf can be passed to the **micompr** toLatex implementations if different formatting is desired.

Simple TikZ 2D scatter plots, as the ones produced by the **micompr** toLatex implementations, can be generated with the tikzscat function. The function accepts the data to plot, an  $n \times 2$  numeric matrix, of  $n$  observations and 2 dimensions, and a factor vector specifying the levels or groups associated with each observation. Several plot characteristics, such as mark types, scale and axes color, are configurable via function arguments. tikzscat returns a string containing the TikZ figure code for plotting the specified data.

### Included data

The package includes test data produced by several implementations of the Predator-Prey for High Performance Computing (PPHPC) simulation model (Fachada et al., 2015). The data is provided in rdata format, and is readily available on loading the package. The same data is also provided in TSV format. This is a limited subset of the complete data, and is included for package testing and exemplification purposes. The example discussed in Section [Simulation model with multiple outputs](#) uses a superset of this data, which is available for public download, but could not be included with the package due to its large size.

### Dependencies

**micompr** has a number of optional dependencies, not required for package installation and for using most of its functionality. The **biotools** (da Silva, 2016) and **MVN** (Korkmaz et al., 2014) packages are required by the assumptions functions, providing the statistical tests for assessing MANOVA and  $t$ -test assumptions. If these functions are invoked without the presence of the specified packages, they will inform the user of that fact, and terminate cleanly. The **testthat** (Wickham, 2011), **knitr**

(Xie, 2015) and **roxygen2** (Wickham et al., 2015) packages are required for package development. The **deseasonalize** package (McLeod and Gweon, 2013) is required for building one of the vignettes.

## Examples

In this section we discuss four concrete application examples for the **micompr** package. The complete scripts used in these examples are available at <https://github.com/fakenmc/micompr-examples>.

### Simulation model with multiple outputs

The replication of a simulation model in a new context highlights differences between the conceptual and implemented models, as well as inconsistencies in the conceptual model specification (Edmonds and Hales, 2003), promoting model verification, model validation (Wilensky and Rand, 2007), and model credibility (Thiele and Grimm, 2015). Some argue that a simulation model is untrustworthy until it has been successfully replicated (Edmonds and Hales, 2003; David, 2013). Model parallelization is an illustrative example of the importance of replication, as it is often required for simulating large models in practical time frames (Fachada et al., in press). By definition, model parallelization implies a number of changes, or even full reimplementation, of the original model, such that a robust comparison methodology, as provided by the **micompr** package, is required in order to make sure a parallelized model faithfully reproduces the behavior of the original serial model.

PPHPC is a reference model for studying and evaluating implementation strategies for spatial agent-based models, capturing important characteristics such as agent movement and local agent interactions (Fachada et al., 2015). The model describes a prototypical predator-prey system, and has six outputs, namely prey population,  $P^s$ , predator population,  $P^w$ , cell-bound food quantity,  $P^c$ , mean prey energy,  $\bar{E}^s$ , mean predator energy,  $\bar{E}^w$ , and mean cell-bound food levels,  $\bar{C}$ . Since outputs are collected once per iteration, each simulation run yields six time series, associated with the individual outputs. With several open source implementations publicly available, the model provides a good test case for multivariate comparison purposes.

Here we show the main comparison cases discussed in a previous article (Fachada et al., 2017), in which the model implementations are parameterized with size 400 and parameter set 1 (Fachada et al., 2015). A canonical PPHPC realization, implemented in NetLogo (Wilensky, 1999), is compared with three configurations of a parallel Java implementation (Fachada et al., in press). The NetLogo implementation and the first Java configuration follow the PPHPC conceptual model and the specified parameters. The second Java configuration disables agent shuffling prior to agent actions, which is explicitly mandated in the conceptual model description. The third Java configuration performs a minimal change in one of the parameters specified by parameter set 1. As such, we define three comparison cases:

**Case I** Compare the NetLogo implementation with the first Java configuration. These should yield distributionally equivalent results.

**Case II** Compare the NetLogo implementation with the second Java configuration. A small misalignment is to be expected.

**Case III** Compare the NetLogo implementation with the third Java configuration. There should be a mismatch in the outputs.

Independent samples of the six model outputs were obtained from  $n = 30$  replications for each implementation or configuration, in a total of  $4n = 120$  runs. Each replication  $r = 1, \dots, 4n$  was performed with a PRNG seed obtained by taking the MD5 checksum of  $r$ , guaranteeing independence between seeds, and consequently, between replications. The data generated by this computational experiment, as well as the scripts used to set up the experiment, are made available to other researchers at <https://zenodo.org/record/46848>.

The following script performs these comparisons. Note that the `concat = TRUE` option of the `micomp` function specifies that an additional concatenated output,  $\tilde{A}$ , should be generated from the original outputs and analyzed in a similar fashion. The `dir_data` variable specifies the location of the dataset.

```
R> # Load package
R> library(micompr)

R> # Output names
R> outputs <- c("$P^s$", "$P^w$", "$P^c$", "$\\mean{E}^s$",
+           "$\\overline{E}^w$", "$\\overline{C}$", "$\\widetilde{A}$")
```

```
R> # Outputs from the NetLogo implementation
R> dir_nl_ok <- paste0(dir_data, "nl_ok")
R> # Outputs from the Java implementation, first configuration
R> dir_jex_ok <- paste0(dir_data, "j_ex_ok")
R> # Outputs from the Java implementation, second configuration
R> dir_jex_noshuff <- paste0(dir_data, "j_ex_noshuff")
R> # Outputs from the Java implementation, third configuration
R> dir_jex_diff <- paste0(dir_data, "j_ex_diff")

R> # Files for model size 400, parameter set 1
R> filez <- glob2rx("stats400v1*.txt")

R> # Perform the three comparison cases
R> mic <- micomp(outputs,
+                 ve_ncps = 0.75,
+                 list(list(name = "I",
+                          folders = c(dir_nl_ok, dir_jex_ok),
+                          files = c(filez, filez),
+                          lvs = c("NLOK", "JEXOK")),
+                      list(name = "II",
+                          folders = c(dir_nl_ok, dir_jex_noshuff),
+                          files = c(filez, filez),
+                          lvs = c("NLOK", "JEXNS")),
+                      list(name = "III",
+                          folders = c(dir_nl_ok, dir_jex_diff),
+                          files = c(filez, filez),
+                          lvs = c("NLOK", "JEXDIF"))),
+                 concat = TRUE)
```

The `mic` object can be inspected at the R prompt using the common S3 generic functions `print`, `summary` and `plot`. For publication purposes, the `toLatex` method for “`micomp`” objects produces L<sup>A</sup>T<sub>E</sub>X tables with user-specified information. For example, to generate a table similar to Table 4 of our previous work (Fachada et al., 2017), `toLatex` is invoked as follows:

```
R> toLatex(mic, booktabs = TRUE,
+           data_show = c("npcs-1", "mnvp-1", "parp-1", "scoreplot"),
+           data_labels = c("$\\#PCs", "MNV", "$t$-test", "PCS"),
+           col_width = TRUE, pvalf_params = list(minval = 1e-8, na_str = "*"),
+           label = "tab:pphpc",
+           caption = paste("Comparison of a NetLogo implementation of",
+                           "the PPHPC model against three configurations",
+                           "of a parallel Java implementation."))
```

This call produces Table 1 with `booktabs` (Fear, 2005) table style (`booktabs = TRUE`) and width set to document column width (`col_width = TRUE`), since the table is somewhat large. The `label` and `caption` parameters set the label and caption of the L<sup>A</sup>T<sub>E</sub>X table, respectively, while the `pvalf_params` argument accepts a list of options for formatting *p*-values. The `data_show` parameter specifies what data to show, which in this case is: 1) `npcs-1`, the number of PCs for the first specified variance (the `micomp` function accepts and performs output comparison with one or more specified variances); 2) `mnvp-1`, the MANOVA *p*-value for the first specified variance; 3) `parp-1`, *t*-test *p*-value for the first PC; and, 4) `scoreplot` for the first two PCs.

In terms of comparison, the method does not find significant differences in case I. However, it successfully differentiates the configurations compared in cases II and III. This is in line with what would be expected, and is discussed in further detail by Fachada et al. (2017). While not shown here, the `assumptions(mic)` command reveals that most assumptions of the MANOVA and *t*-tests are verified.

## Monthly sunspots

This example uses the monthly sunspot data (WDC, 2016), included with R, which contains the monthly numbers of sunspots from 1749 to the present day. The solar cycle is an approximate 11-year period of changes in the number of sunspots and other associated phenomena. Thus, we divide the data into 11-year (132-month) periods, and consider each period to be an observation. In practice this is an oversimplification, since the cycles can be a bit longer or shorter than 11 years.

Comp.	Data	Outputs						
		$P^s$	$P^w$	$P^c$	$\bar{E}^s$	$\bar{E}^w$	$\bar{C}$	$\tilde{A}$
I	#PCs	13	18	14	22	28	14	22
	MNV	0.325	0.640	0.462	0.052	0.796	0.463	0.523
	$t$ -test	0.530	0.836	0.804	0.784	0.378	0.805	0.879
PCS								
II	#PCs	13	15	13	1	28	14	21
	MNV	<u>4e-05</u>	<u>&lt;1e-08</u>	<u>0.002</u>	*	0.089	<u>0.003</u>	<u>2e-07</u>
	$t$ -test	<u>0.042</u>	<u>&lt;1e-08</u>	<u>0.108</u>	<u>&lt;1e-08</u>	<u>0.017</u>	<u>0.109</u>	<u>0.390</u>
PCS								
III	#PCs	7	1	2	2	20	1	14
	MNV	<u>&lt;1e-08</u>	*	<u>&lt;1e-08</u>	<u>&lt;1e-08</u>	<u>&lt;1e-08</u>	*	<u>&lt;1e-08</u>
	$t$ -test	<u>&lt;1e-08</u>						
PCS								

**Table 1:** Comparison of a NetLogo implementation of the PPHPC model against three configurations of a parallel Java implementation.

Given the data, we define two samples of 10 observations each, over a period of 110 years or 1320 months. The first sample includes solar cycles from 1749 to 1859, while the second encompasses cycles from 1902 to 2012. We can now ask the following question: were the solar cycles during the 1749–1859 interval significantly different from the more recent observations? The following code compares observations from the two time intervals, and attempts to provide an answer:

```
R> # Load package
R> library(micompr)

R> # Months in the 1749-1859 interval (110 years)
R> # Months in the 1902-2012 interval (110 years)
R> m <- sunspot.month[c(1:1320, 1837:3156)]
R> m <- matrix(m, nrow = 20)

R> # Factor vector, two levels:
R> # a) ten 11-year cycles from 1749 to 1859
R> # b) ten 11-year cycles from 1902 to 2012
R> groups <- factor(c(rep("A", 10), rep("B", 10)))

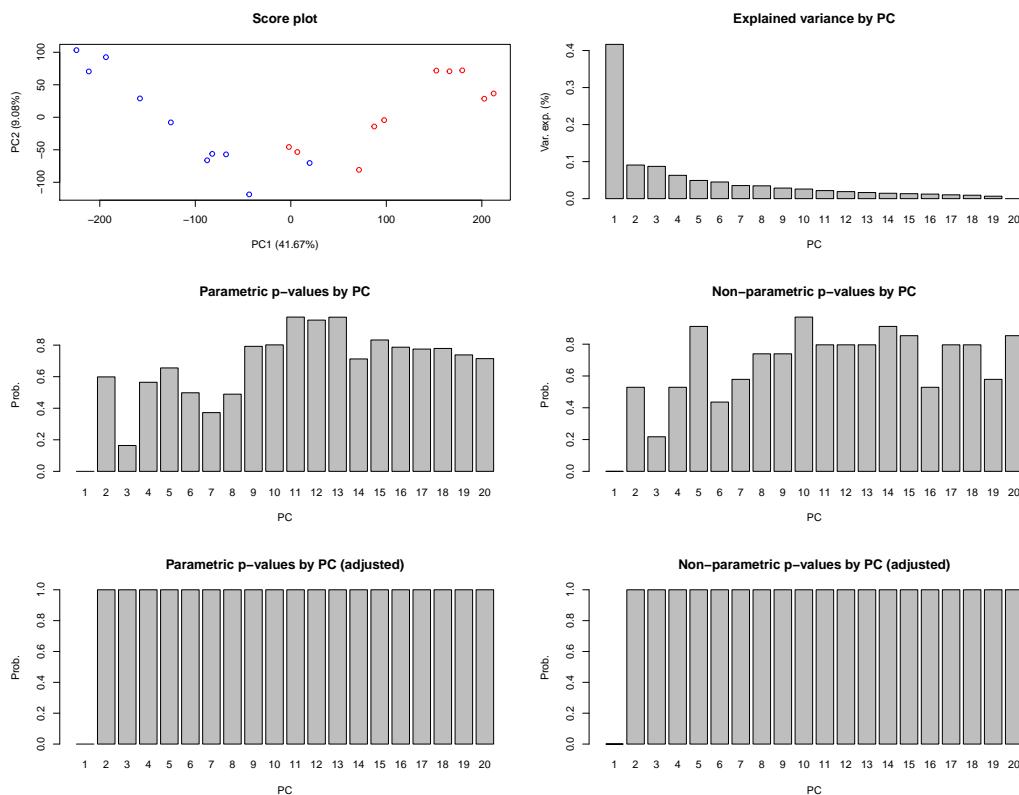
R> # Compare the two groups, use 9 PCs for MANOVA
R> cmp <- cmpoutput("SunSpots", 9, m, groups)
```

The `cmp` object can now be analyzed:

```
R> cmp

Output name: SunSpots
Number of PCs which explain 85.0% of variance: 9
P-Value for MANOVA along 9 dimensions: 3.40755e-06
P-Value for t-test (1st PC): 2.713985e-06
P-Value for Mann-Whitney U test (1st PC): 4.330035e-05
Adjusted p-Value for t-test (1st PC): 6.513579e-06
Adjusted p-Value for Mann-Whitney U test (1st PC): 0.0001039211
```

The MANOVA  $p$ -value is significant, as well as the  $t$ -test and Mann-Whitney PC1  $p$ -values, before and after weighted Bonferroni correction. As such, it is possible to conclude that solar cycles from 1749 to 1859 were significantly different from cycles between 1902 and 2012. However, the data



**Figure 1:** Plots produced by sunspots example.

in accordance with the assumptions for the MANOVA and *t*-test? This can be checked with the `assumptions` function:

```
R> assumptions(cmp)

==== MANOVA assumptions ====
    NPCs=9
Royston (A) 2.190796e-01
Royston (B) 3.627858e-01
Box's M      1.567180e-08

==== T-test assumptions ====
          PC1
Shapiro-Wilk (A) 0.7739058
Shapiro-Wilk (B) 0.3791168
Bartlett       0.9353299
```

Only Box's *M* test, which checks for homogeneity of variance-covariance matrices, is significant. However, this test is prone to false positives, and this assumption is not critical when samples are of the same size (Tabachnick and Fidell, 2013). Given this information, it seems plausible to consider the results provided by the parametric tests in our final decision, i.e., that there is in fact a significant difference between samples. A good way to visualize the overall results is to plot the "cmpoutput" object:

```
R> plot(cmp)
```

This command generates the plots shown in Figure 1. The score plot shows the samples to be distinctly separated, and the variance explained by PC decreases abruptly from the first PC to the second. Univariate *p*-values for PC1 are visibly significant, though not very much for the remaining PCs.

### Saugeen river flow

This example uses the Saugeen River daily flow data (Hipel and McLeod, 1994), included with the `deseasonalize` package. This data consists of a time series of the rivers' daily flow ( $\text{m}^3/\text{s}$ ) from 1915 to

1979. Considering one year as an observation, there are a total of 65 observations. We can, for example, define two samples of 30 observations each, with the first and last 30 years of records, and ask the following question: is there any statistical difference between the flow dynamics during the 1915–1944 and 1950–1979 periods (perhaps due to climate change or some other factor)? The following code compares observations from the two periods:

```
R> # Load packages
R> library(micompr)
R> library(deseasonalize)

R> # Unique years
R> years <- unique(sapply(rownames(SaugeenDay), substr, 1, 4))

R> # Number of days in each year
R> ndays <- sapply(years, function(x) sum(substr(rownames(SaugeenDay), 1, 4) == x))

R> # Indexes of last day in each year
R> lastdays <- cumsum(ndays)

R> # Prepare data for PCA
R> saugdata <- t(mapply(
+   function(nd, ld) {
+     rflows <- rep(NA, 366)
+     rflows[1:nd] <- SaugeenDay[(ld - nd + 1):ld]
+     # Discard last day in leap years
+     rflows[1:365]
+   },
+   ndays, lastdays))

R> # Consider first 30 years and last 30 years (discard 5 years in between)
R> saugdata <- saugdata[c(1:30, 36:65), ]

R> # Factor vector, two levels: first 30 years and last 30 years
R> groups <- factor(c(rep("A", 30), rep("B", 30)))

R> # Compare
R> cmp <- cmpoutput("SaugeenFlow", 0.9, saugdata, groups)
```

The `cmp` object can now be analyzed:

```
R> cmp

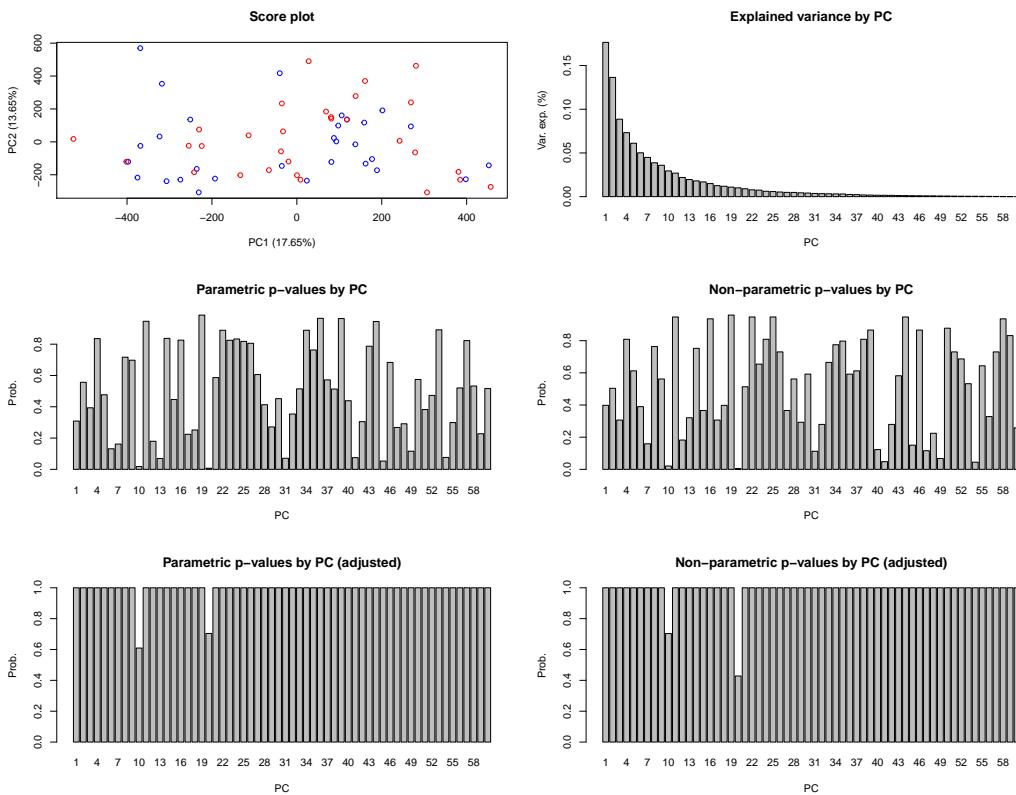
Output name: SaugeenFlow
Number of PCs which explain 90.0% of variance: 21
P-Value for MANOVA along 21 dimensions: 0.0740641
P-Value for t-test (1st PC): 0.3088125
P-Value for Mann-Whitney U test (1st PC): 0.3980033
Adjusted p-Value for t-test (1st PC): 1
Adjusted p-Value for Mann-Whitney U test (1st PC): 1
```

There are no significant  $p$ -values. As such, it is not possible to conclude that the river flow dynamic during the first 30 years of measurements is statistically different from the last 30 years. The MANOVA assumptions are not verified, as shown by the `assumptions` function:

```
R> assumptions(cmp)

==== MANOVA assumptions ====
      NPCs=21
Royston (A) 1.309146e-05
Royston (B) 3.947504e-05
Box's M      1.859561e-10

==== T-test assumptions ====
          PC1
Shapiro-Wilk (A) 0.02860029
Shapiro-Wilk (B) 0.83367671
Bartlett        0.79716243
```



**Figure 2:** Plots produced by the Saugeen river flow example.

The *t*-test assumptions mostly hold, in spite of the PC1 normality test for sample A (first 30 years) being significant at the  $\alpha = 0.05$  level. Nonetheless, the *U* test *p*-value provides a similar conclusion. Plotting the *cmp* object provides another perspective:

```
R> plot(cmp)
```

The previous command will produce plots in Figure 2. The PC1 vs. PC2 score plot does not show any clear sample separation and the decrease in explained variance from the first PC to the second is considerably less abrupt than what was observed for the sunspots example. Additionally, no *t* and *U* test *p*-values are significant for the 60 PCs after weighted Bonferroni correction, further reinforcing the conclusion that the yearly Saugeen river flow dynamic was similar during the compared periods of time.

## PH<sup>2</sup> database of dermoscopic images

In this example we use the tools provided by the **micompr** package to study the PH<sup>2</sup> database of dermoscopic images (Mendonça et al., 2013). This image database contains a total of 200 dermoscopic images of melanocytic lesions, including, from benign to more serious, 80 common nevi, 80 atypical nevi, and 40 melanomas. These are 8-bit RGB color images, with a resolution of purportedly  $768 \times 560$  pixels. We have found, however, that resolutions vary between  $761 \times 570$  and  $769 \times 577$ . As such, we resized all images to  $760 \times 570$  prior to our analysis. The goal is to verify if images of the three types of lesions form statistically distinguishable samples, i.e., this is not a classification exercise such as performed by Barata et al. (2014).

Each image is considered an observation of three outputs, red, green and blue, corresponding to the respective color channels. The concatenation of all outputs, i.e., channels, provides a 4<sup>th</sup> output. The three lesion samples are compared pairwise, as follows:

- 1v2 Common nevi and atypical nevi.
- 1v3 Common nevi and melanomas.
- 2v3 Atypical nevi and melanomas.

The following code reads the image dataset from disk and compares images grouped by lesion type. The *imgfolder* variable specifies the path containing the images (resized to  $760 \times 570$ ), while the *grpsfile* variable specifies the path to the file containing the sample to which each image belongs.

```

R> # Load packages
R> library(bmp)
R> library(micompr)

R> # Image definitions
R> imgs <- dir(imgfolder)
R> nimgs <- length(imgs)
R> npixels <- 760 * 570

R> # Specify image groups (Common nevi, atypical nevi, melanomas).
R> f <- read.table(grpsfile, row.names = 1)
R> grps <- f[order(row.names(f)), ]

R> # Read images from disk
R> # Use different color channels as outputs, and also use a concatenated output
R> rimgs <- matrix(nrow = nimgs, ncol = npixels)
R> gimgs <- matrix(nrow = nimgs, ncol = npixels)
R> bimgs <- matrix(nrow = nimgs, ncol = npixels)
R> rgbimgs <- matrix(nrow = nimgs, ncol = npixels * 3)

R> for (i in 1:nimgs) {
+   cimg <- read.bmp(paste0(imgfolder, imgs[i]))
+   rimgs[i, ] <- c(cimg[, , 1])
+   gimgs[i, ] <- c(cimg[, , 2])
+   bimgs[i, ] <- c(cimg[, , 3])
+   rgbimgs[i, ] <- c(cimg[, , 1], cimg[, , 2], cimg[, , 3])
+ }

R> # Perform multivariate independent comparison of images
R> mic <-
+   micomp(outputs = c("R", "G", "B", "RGB"),
+           ve_npcs = 0.9,
+           comps = list(
+             list(name = "1v2",
+                  grpout = list(
+                    data = list(R = rimgs[grps != 3, ],
+                               G = gimgs[grps != 3, ],
+                               B = bimgs[grps != 3, ],
+                               RGB = rgbimgs[grps != 3, ]),
+                    obs_lvls = factor(grps[grps != 3]))),
+             list(name = "1v3",
+                  grpout = list(
+                    data = list(R = rimgs[grps != 2, ],
+                               G = gimgs[grps != 2, ],
+                               B = bimgs[grps != 2, ],
+                               RGB = rgbimgs[grps != 2, ]),
+                    obs_lvls = factor(grps[grps != 2]))),
+             list(name = "2v3",
+                  grpout = list(
+                    data = list(R = rimgs[grps != 1, ],
+                               G = gimgs[grps != 1, ],
+                               B = bimgs[grps != 1, ],
+                               RGB = rgbimgs[grps != 1, ]),
+                    obs_lvls = factor(grps[grps != 1])))))

```

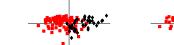
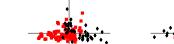
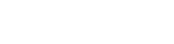
As in the [Simulation model with multiple outputs](#) example, the `mic` object can be inspected at the R prompt using the common S3 generic functions. Likewise, the `toLatex` function produces L<sup>A</sup>T<sub>E</sub>X tables summarizing the object. The following code generates Table 2:

```

R> toLatex(mic, booktabs = TRUE, data_show = c("parp-1", "nparp-1", "scoreplot"),
+           data_labels = c("$t$-test", "$U$ test", "PCS"),
+           pvalf_params = list(minval = 1e-8, na_str = "*"), label = "tab:ph2",
+           caption = paste("Comparison of PH$^2$ dataset images",
+                           "grouped by lesion type."))

```

Note that we did not request the MANOVA *p*-values in the `data_show` parameter, as in this case

Comp.	Data	Outputs			
		R	G	B	RGB
1v2	<i>t</i> -test	<u>2e-04</u>	<u>0.001</u>	0.033	<u>0.002</u>
	<i>U</i> test	<u>8e-05</u>	<u>5e-04</u>	0.024	<u>0.001</u>
	PCS				
1v3	<i>t</i> -test	<u>&lt;1e-08</u>	<u>&lt;1e-08</u>	7e-07	<u>&lt;1e-08</u>
	<i>U</i> test	<u>&lt;1e-08</u>	<u>&lt;1e-08</u>	7e-06	<u>&lt;1e-08</u>
	PCS				
2v3	<i>t</i> -test	<u>&lt;1e-08</u>	<u>&lt;1e-08</u>	3e-04	<u>&lt;1e-08</u>
	<i>U</i> test	<u>6e-08</u>	<u>&lt;1e-08</u>	0.001	<u>8e-07</u>
	PCS				

**Table 2:** Comparison of PH<sup>2</sup> dataset images grouped by lesion type.

the required assumptions do not appear to be verified. However, assumptions for the *t*-test on the first PC seem to hold. In any case, and to complement the information provided by the *t*-test, we specified the "nparp=1" option to the data\_show argument, such that the table shows the *p*-value of the Mann-Whitney *U* test on the first PC.

Results in Table 2 show that images of different lesions have statistically significant differences, when compared either by individual color channels or with the three channels concatenated. The latter seems to provide better differentiation, with the common nevi and melanoma samples (1v3 comparison) appearing to be the most dissimilar.

## Summary

In this paper we presented the R package **micompr**, which implements a procedure for comparing multivariate samples associated with different factor levels or groups. The package architecture and its core components were discussed and four examples were examined.

## Acknowledgments

This work was supported by the Fundação para a Ciência e a Tecnologia (FCT) projects UID/EEA/50009/2013 and UID/MAT/04561/2013, and partially funded with grant SFRH/BD/48310/2008, also from FCT.

## Bibliography

- M. J. Anderson. A new method for non-parametric multivariate analysis of variance. *Austral Ecology*, 26(1):32–46, Feb. 2001. doi: 10.1111/j.1442-9993.2001.01070.pp.x. [p407]
- C. Barata, M. Ruela, M. Francisco, T. Mendonça, and J. S. Marques. Two systems for the detection of melanomas in dermoscopy images using texture and color features. *IEEE Systems Journal*, 8(3):965–979, Sept. 2014. doi: 10.1109/JST.2013.2271540. [p416]
- L. Baringhaus and C. Franz. On a new multivariate two-sample test. *Journal of Multivariate Analysis*, 88(1):190–206, Jan. 2004. doi: 10.1016/s0047-259x(03)00079-4. [p407]
- R. A. Berg, H. C. Hoefsloot, J. A. Westerhuis, A. K. Smilde, and M. J. Werf. Centering, scaling, and transformations: improving the biological information content of metabolomics data. *BMC Genomics*, 7(1):142, 2006. doi: 10.1186/1471-2164-7-142. [p410]

- K. R. Clarke. Non-parametric multivariate analyses of changes in community structure. *Australian Journal of Ecology*, 18(1):117–143, Mar. 1993. doi: 10.1111/j.1442-9993.1993.tb00438.x. [p407]
- A. R. da Silva. *biotools: Tools for Biometry and Applied Statistics in Agricultural Science*, 2016. URL <https://CRAN.R-project.org/package=biotools>. R package version 3.0. [p410]
- N. David. Validating simulations. In *Simulating Social Complexity*, Understanding Complex Systems, pages 135–171. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-540-93813-2\\_8. [p411]
- T. Duong. *ks: Kernel Smoothing*, 2016. URL <https://CRAN.R-project.org/package=ks>. R package version 1.10.4. [p407]
- T. Duong, B. Goud, and K. Schauer. Closed-form density-based framework for automatic detection of cellular morphology changes. *Proceedings of the National Academy of Sciences*, 109(22):8382–8387, May 2012. doi: 10.1073/pnas.1117796109. [p407]
- B. Edmonds and D. Hales. Replication, replication and replication: Some hard lessons from model alignment. *Journal of Artificial Societies and Social Simulation*, 6(4):11, 2003. URL <http://jasss.soc.surrey.ac.uk/6/4/11.html>. [p411]
- N. Fachada. *micompr: Multivariate Independent Comparison of Observations*, 2016. URL <https://CRAN.R-project.org/package=micompr>. R package version 1.0.1. [p406]
- N. Fachada, V. V. Lopes, R. C. Martins, and A. C. Rosa. Towards a standard model for research in agent-based modeling and simulation. *PeerJ Computer Science*, 1:e36, Nov. 2015. doi: 10.7717/peerj-cs.36. [p410, 411]
- N. Fachada, V. V. Lopes, R. C. Martins, and A. C. Rosa. Model-independent comparison of simulation output. *Simulation Modelling Practice and Theory*, 72:131–149, Mar. 2017. doi: 10.1016/j.simpat.2016.12.013. [p408, 411, 412]
- N. Fachada, V. V. Lopes, R. C. Martins, and A. C. Rosa. Parallelization strategies for spatial agent-based models. *International Journal of Parallel Programming*, pages 1–33, in press. doi: 10.1007/s10766-015-0399-9. [p411]
- S. Fear. *Publication Quality Tables in L<sup>A</sup>T<sub>E</sub>X*, Apr. 2005. URL <https://www.ctan.org/pkg/booktabs>. [p412]
- C. Franz. *cramer: Multivariate Nonparametric Cramer-Test for the Two-Sample-Problem*, 2014. URL <https://CRAN.R-project.org/package=cramer>. R package version 0.9-1. [p407]
- J. D. Gibbons and S. Chakraborti. *Nonparametric Statistical Inference*. Statistics: Textbooks & Monographs. Chapman and Hall/CRC, Boca Raton, FL, USA, 5th edition, July 2010. [p406]
- R. Heller, D. Small, and P. Rosenbaum. *crossmatch: The Cross-Match Test*, 2012. URL <https://CRAN.R-project.org/package=crossmatch>. R package version 1.3-1. [p407]
- K. W. Hipel and A. I. McLeod. *Time Series Modelling of Water Resources and Environmental Systems*. Elsevier, 1994. [p414]
- I. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2nd edition, 2002. doi: 10.1007/b98835. [p406]
- S. Korkmaz, D. Goksuluk, and G. Zararsiz. MVN: An R package for assessing multivariate normality. *The R Journal*, 6(2):151–162, 2014. URL <http://journal.R-project.org/archive/2014-2/korkmaz-goksuluk-zararsiz.pdf>. [p410]
- W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952. doi: 10.1080/01621459.1952.10483441. [p406]
- W. J. Krzanowski. *Principles of Multivariate Analysis: A User's Perspective*. Oxford University Press, New York, USA, 1988. [p406]
- F. J. Massey Jr. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951. [p406]
- A. I. McLeod and H. Gweon. Optimal deseasonalization for monthly and daily geophysical time series. *Journal of Environmental Statistics*, 4(11), 2013. URL <http://www.jenvstat.org/v04/i11>. [p411]

- T. Mendonça, P. M. Ferreira, J. Marques, A. R. S. Marcal, and J. Rozeira. PH<sup>2</sup> – a dermoscopic image database for research and benchmarking. In *35th International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5437–5440. IEEE, July 2013. doi: 10.1109/EMBC.2013.6610779. [p416]
- P. W. Mielke Jr, K. J. Berry, and E. S. Johnson. Multi-response permutation procedures for a priori classifications. *Communications in Statistics – Theory and Methods*, 5(14):1409–1424, Jan. 1976. doi: 10.1080/03610927608827451. [p407]
- D. C. Montgomery and G. C. Runger. *Applied Statistics and Probability for Engineers*. John Wiley & Sons, 6th edition, 2014. [p406]
- J. Oksanen, F. G. Blanchet, M. Friendly, R. Kindt, P. Legendre, D. McGlinn, P. R. Minchin, R. B. O'Hara, G. L. Simpson, P. Solymos, M. H. H. Stevens, E. Szoecs, and H. Wagner. *vegan: Community Ecology Package*, 2016. URL <https://CRAN.R-project.org/package=vegan>. R package version 2.4-1. [p407]
- M. L. Rizzo and G. J. Szekely. *energy: E-Statistics: Multivariate Inference via the Energy of Data*, 2016. URL <https://CRAN.R-project.org/package=energy>. R package version 1.7-0. [p407]
- P. R. Rosenbaum. An exact distribution-free test comparing two multivariate distributions based on adjacency. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(4):515–530, Sept. 2005. doi: 10.1111/j.1467-9868.2005.00513.x. [p407]
- R. Rosenthal and D. B. Rubin. Ensemble-adjusted *p* values. *Psychological Bulletin*, 94(3):540–541, 1983. doi: 10.1037/0033-2909.94.3.540. [p407]
- J. P. Shaffer. Multiple hypothesis testing. *Annual Review of Psychology*, 46:561–584, Feb. 1995. doi: 10.1146/annurev.ps.46.020195.003021. [p407]
- G. J. Székely and M. L. Rizzo. Testing for equal distributions in high dimension. *InterStat*, 5:1–6, Nov. 2004. [p407]
- B. G. Tabachnick and L. S. Fidell. *Using Multivariate Statistics*. Pearson, 6th edition, July 2013. [p406, 414]
- M. Talbert, J. Richards, P. Mielke, and B. Cade. *Blossom: Statistical Comparisons with Distance-Function Based Permutation Tests*, 2016. URL <https://CRAN.R-project.org/package=Blossom>. R package version 1.4. [p407]
- J. C. Thiele and V. Grimm. Replicating and breaking models: Good for you and good for ecology. *Oikos*, 124(6):691–696, 2015. doi: 10.1111/oik.02170. [p411]
- Sunspot Number. WDC-SILSO, Solar Influences Data Analysis Center (SIDC), Royal Observatory of Belgium, Brussels, Belgium, Apr. 2016. URL <http://www.sidc.be/silso/datafiles>. [p412]
- H. Wickham. testthat: Get started with testing. *The R Journal*, 3:5–10, 2011. URL [http://journal.R-project.org/archive/2011-1/RJournal\\_2011-1\\_Wickham.pdf](http://journal.R-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf). [p410]
- H. Wickham, P. Danenberg, and M. Eugster. *roxygen2: In-Source Documentation for R*, 2015. URL <https://CRAN.R-project.org/package=roxygen2>. R package version 5.0.1. [p411]
- U. Wilensky. *NetLogo*. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, USA, 1999. URL <http://ccl.northwestern.edu/netlogo/>. [p411]
- U. Wilensky and W. Rand. Making models match: Replicating an agent-based model. *Journal of Artificial Societies and Social Simulation*, 10(4):2, 2007. URL <http://jasss.soc.surrey.ac.uk/10/4/2.html>. [p411]
- Y. Xie. *Dynamic Documents with R and knitr*. CRC Press, 2nd edition, 2015. [p411]

Nuno Fachada  
Institute for Systems and Robotics, LARSys,  
Instituto Superior Técnico,  
Universidade de Lisboa,  
Lisboa,  
Portugal [nfachada@laseeb.org](mailto:nfachada@laseeb.org)

João Rodrigues  
École Polytechnique Fédérale de Lausanne,

*Lausanne,*  
*Switzerland* [joao.rodrigues@epfl.ch](mailto:joao.rodrigues@epfl.ch)

*Vitor V. Lopes*  
*UTEC - Universidad de Ingeniería & Tecnología,*  
*Lima,*  
*Perú* [vitorvieiralopes@gmail.com](mailto:vitorvieiralopes@gmail.com)

*Rui C. Martins*  
*Life and Health Sciences Research Institute,*  
*School of Health Sciences,*  
*University of Minho,*  
*Braga,*  
*Portugal* [rui.martins@ecsaude.uminho.pt](mailto:rui.martins@ecsaude.uminho.pt)

*Agostinho C. Rosa*  
*Institute for Systems and Robotics, LARSyS,*  
*Instituto Superior Técnico,*  
*Universidade de Lisboa,*  
*Lisboa,*  
*Portugal* [acrosa@laseeb.org](mailto:acrosa@laseeb.org)

# mixtox: An R Package for Mixture Toxicity Assessment

by Xiang-Wei Zhu and Jian-Yi Chen

**Abstract** Mixture toxicity assessment is indeed necessary for humans and ecosystems that are continually exposed to a variety of chemical mixtures. This paper describes an R package, called **mixtox**, which offers a general framework of curve fitting, mixture experimental design, and mixture toxicity prediction for practitioners in toxicology. The unique features of **mixtox** include: (1) constructing a uniform table for mixture experimental design; and (2) predicting toxicity of a mixture with multiple components based on reference models such as concentration addition, independent action, and generalized concentration addition. We describe the various functions of the package and provide examples to illustrate their use and show the collaboration of **mixtox** with other existing packages (e.g., **drc**) in predicting toxicity of chemical mixtures.

## Introduction

Environmental chemicals usually occur as complex mixtures rather than a single compound. Humans and ecosystems are continually exposed to a variety of chemical mixtures with changing composition under different circumstances. Public concern over environmental chemicals demands extensive risk assessment of various mixtures. Today, risk assessment of the effect of environmental mixtures is mainly based on a complex framework in the context of reference models (Backhaus and Faust, 2012). Predicting toxicity of mixtures using reference models is usually based on the fitting information of individual concentration response curves (Faust et al., 2003, 2001). Knowledge on the synergistic, antagonistic and additive effects of multiple stressors will be helpful for regulatory agencies in the risk assessment of environmental chemicals.

Curve fitting of individual concentration responses is the basis for predicting the effect of mixtures. Many sigmoidal regression functions have been proposed for the fitting of monotonic nonlinear concentration response data (Goutelle et al., 2008; Scholze et al., 2001; Spiess and Neumeyer, 2010). Sigmoidal functions (e.g., Logit and Weibull) with a lower limit of 0 and an upper limit of 1 (Scholze et al., 2001) are suitable for quantal response data. Other functions like three- or four-parameter Hill functions (Goutelle et al., 2008) and three- or four-parameter Weibull and Logistic functions (Spiess and Neumeyer, 2010) could be used for both quantal and continuous response data. Both proprietary (e.g., Origin) and free-to-download software (e.g., EPA Probit analysis program and BMDS; <https://www.epa.gov/bmds>) are available for curve fitting. Focusing on R packages we must mention **drc** (Ritz and Streibig, 2005), **drcfit** (Ranke, 2016), and **ezec** (Kamvar, 2016). Package **drc** provides a suite of flexible and versatile model fitting and after-fitting functions to analyze concentration response data.

Appropriate mixture experiments need to be designed to analyze interactions (synergistic, antagonistic, or additive) between/among mixture components. Usually, fixed-ratio ray design is employed to systematically measure the effect of mixtures. One popular fixed-ratio ray design is the equal effect concentration ratio (EECR) (Faust et al., 2003, 2001), which designs mixtures according to the proportion of a particular effect concentration, say half maximal effect concentration (EC50), of individual chemicals. EECR has become a de facto standard for the experimental investigation of chemical interaction for all kind of mixtures. The call for simulating “environmentally realistic” mixtures (Backhaus and Faust, 2012) requires more sophisticated experimental design for mixtures. The uniform design concentration ratio (UDCR, or uniform design ray; Liu et al. 2015) was proposed to construct the “environmentally realistic” mixtures. Constructing appropriate uniform design tables (Ning et al., 2011) is the key to the sophisticated uniform experimental design. Uniform design tables with various number of runs, factors, and levels calculated by Professor Fang and his colleagues can be found on the website <http://sites.stat.psu.edu/~rli/DMCE/UniformDesign/>. However, those tables are not specifically tuned for mixture experimental design. The APTox program (Liu et al., 2012), assessment and prediction on toxicity of chemical mixtures, developed based on Visual Basic, provides a module to construct uniform design tables. To the best of our information, there is no R package available for the construction of uniform design table.

Concentration addition (CA) and independent action (IA) are two classical reference concepts that allow predicting toxicity of mixture substances (Backhaus and Faust, 2012). CA assumes to predict toxicity of mixture compounds with a similar mechanism of action. In contrast to CA, IA assumes mixture components act on different biological targets of an exposed organism independently. The

expected effect of mixtures can hence be calculated according to the joint probability of statistically independent events. The package **drc** (Ritz and Streibig, 2005, 2014) provides fitting models (e.g., CA, Hewlett, and Voelund) incorporating synergism/antagonism for binary and ternary mixtures. However, there is a need for evaluating the effect of mixtures with more than three components. The CA and IA models are suitable for fitting mixtures with quantal concentration response data (or quantal responses after proper transformation). Generalized concentration addition (GCA) (Howard and Webster, 2009) was proposed to examine mixtures containing partial agonists. GCA is appropriate for both quantal and continuous concentration response relationships. A following study (Hadrup et al., 2013) showed GCA could predict a larger range of the concentration response curve as compared with the CA and IA models. GCA is appropriate for both quantal and continuous concentration response relationships.

The goal of this paper is to describe an R package, called **mixtox** (Zhu, 2016), that allows to perform curve fitting, experimental design, and mixture toxicity assessment. In this paper, we will introduce curve fitting of individual concentration response data using a series of sigmoidal models, mixture experimental design based on different strategies, and mixture toxicity prediction based on CA, IA, and GCA, as well as follow-up analysis.

The paper is organized as follows. First, the statistical fundamentals underneath the curve fitting, experimental design, and mixture toxicity prediction are briefly recalled. Then the main functions in **mixtox** are listed and their use described. Finally, several examples are provided to illustrate the use of these functions.

## Outline of statistical fundamentals

Consider a set of  $n$  concentration response data points  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$  and model function  $y = f(x, \theta) + e$  with  $p$  parameters  $\theta = (\theta_1, \theta_2, \dots, \theta_p)$ . A non-linear least square regression was used to fit the concentration response data. The purpose of non-linear least square regression is to find the parameter  $\theta$  to minimize the sum of squares of  $e$  (a vector of errors). With the help of the modified Levenberg-Marquardt algorithm, package **minpack.lm** (Elzhov et al., 2016) is more efficient than the built-in **nls** function in solving nonlinear least-squares problems. It was employed to fit the concentration response data instead of **nls**.

### Goodness of fit statistics

The **mixtox** package provides 13 sigmoidal functions to fit concentration response data. A series of goodness of fit statistics were provided to select the best fit. These statistics include coefficient of determination ( $R^2$ ), the bias-corrected coefficient of determination ( $R^2_{adj}$ ), root mean squared error ( $RMSE$ ), mean absolute error ( $MAE$ ), Akaike information criterion ( $AIC$ ), the bias-corrected  $AIC_c$ , and Bayesian information criterion ( $BIC$ ). The mathematical expression of these statistics can be found in Liao and McGee (2003) and Spiess and Neumeyer (2010). A variety of functions can be used to fit the concentration responses of one compound on an organism. Previously, the best model was selected based on the highest  $R^2$  and  $R^2_{adj}$ , lowest  $RMSE$ , and lowest  $MAE$ . However,  $R^2$  or even  $R^2_{adj}$  are not sensitive measures for nonlinear models as compared with  $AIC$ ,  $AIC_c$ , and  $BIC$  (Spiess and Neumeyer, 2010). Here, we suggest that users select the best model based on the lowest  $AIC$ ,  $AIC_c$ , and  $BIC$ .

### Confidence intervals

The Delta method (Dybowski and Gant, 2001) is used to construct confidence intervals for predicted responses. The (non-simultaneous) confidence interval (CI) at the predictor value  $x$  is given as follows:

$$CI = f(\hat{\theta}) \pm t_{(n-p, (1-\alpha)/2)} \sqrt{(\nu^T Var(\hat{\theta}) \nu)}, \quad (1)$$

where  $t$  is the inverse of Student's  $t$  cumulative distribution function,  $\alpha$  is the confidence level (usually 95%); the superscript  $T$  denotes transpose;  $\nu$  is the row vector of the Jacobian evaluated at a specified predictor value.

$Var(\hat{\theta})$  in (1) is the covariance matrix of the parameter estimates.

$$Var(\hat{\theta}) = \sigma^2 \left( J(\theta)^T J(\theta) \right)^{-1}, \quad (2)$$

where  $\sigma^2$  is the squared residual standard error.  $J(\theta)$  is the Jacobian matrix of  $f(\theta)$  with respect to the coefficients.

The (non-simultaneous) prediction bounds for a new observation (Huet et al., 2004) (i.e., prediction interval,  $PI$ ) were found to show better characterization on responses at the extremely low concentrations (Zhu et al., 2013). The  $PI$  (De Gryze et al., 2007; Huet et al., 2004) was recommended as the primary choice for uncertainty characterization:

$$PI = f(\hat{\theta}) \pm t_{(n-m, (1-\alpha)/2)} \sqrt{\sigma^2 + (\nu^T Var(\hat{\theta})\nu)}. \quad (3)$$

The  $PI$  for new observations should be wider than the  $CI$  for the additional variance ( $\sigma^2$ ) in predicting new responses (the fit plus random errors).

### Uniform design table

Many experimental design methods such as generalized Latin hypercube design (Dette and Pepelyshev, 2010) and uniform design (Ning et al., 2011) can be used to construct chemical mixtures. The uniform design was incorporated into package **mixtox**. Uniform design is an efficient experimental design method to simulate environmentally realistic mixtures. It allows researchers to investigate mixtures with more chemicals (i.e., factors in uniform design) and concentrations (levels) while simultaneously minimizing the number of experiments (Liu et al., 2015).

As a prerequisite for constructing a uniform design table, researchers need to plan the number of runs, factors, and levels for a mixture experiment according to their experiment. Usually, the number of runs is an integer multiple of levels. Assume that we want to test the mixture effect of three compounds each with four different concentrations. Based on the three factors (compounds) and four levels (concentrations), a uniform design table with four runs would be suitable for this case. If we want to use a uniform design table with eight runs, then the levels (concentrations) need to be allocated in repetition to form pseudo-levels (Liang et al., 2001), the number of which equals that of runs.

The good lattice point method (Zhou and Fang, 2013) with a power generator was used to construct a uniform table based on the number of runs. The centered  $L_2$ -discrepancy ( $CD_2$ ) (Hickernell, 1996; Zhou and Fang, 2013) was employed to measure the uniformity and find the one with lowest discrepancy.

$$CD_2(P) = \left[ \left( \frac{13}{12} \right)^s - \frac{2^{1-s}}{n} \sum_{k=1}^n \prod_{i=1}^s \theta_{ki} + \frac{1}{n^2} \sum_{k,l=1}^n \prod_{i=1}^s \phi_{k,li} \right]^{1/2}, \quad (4)$$

with the definition of  $\theta_{ki}$  and  $\phi_{k,li}$  as follows:

$$\theta_{ki} = 2 + |x_{ki} - \frac{1}{2}| - |x_{ki} - \frac{1}{2}|^2, \quad (5)$$

$$\phi_{k,li} = 1 + \frac{1}{2} \left( |x_{ki} - \frac{1}{2}| + |x_{li} - \frac{1}{2}| - |x_{ki} - x_{li}| \right), \quad (6)$$

and where  $n$  and  $s$  are the number of runs (levels or pseudo-levels) and the number of factors, respectively.

### Reference models

For a well-defined mixture of  $n$  components, concentration addition is expressed mathematically as (Faust et al., 2001):

$$\sum_{i=1}^n \frac{c_i}{ECx_i} = 1, \quad (7)$$

where  $ECx_i$  is the effect concentration of the  $i^{th}$  component that causes an  $x\%$  effect when applied individually at  $c_i$ . The  $c_i$  is expressed as:

$$c_i = p_i \cdot c_{mix} = p_i \cdot EC_{x,mix}, \quad (8)$$

where  $p_i$  is the proportion of the  $i^{th}$  component in a mixture,  $c_{mix}$  the concentration of the mixture and  $EC_{x,mix}$  the concentration of the mixture that causes an effect of  $x\%$ . The formula to predict the effect

of a mixture can be expressed as:

$$EC_{x,mix} = \left( \sum_{i=1}^n \frac{p_i}{EC_{x,i}} \right)^{-1}. \quad (9)$$

The theoretical basis for independent action is defined as (Faust et al., 2001):

$$E(c_{mix}) = 1 - (1 - E(c_1))(1 - E(c_2)) \cdots (1 - E(c_n)) = 1 - \prod_{i=1}^n (1 - E(c_i)), \quad (10)$$

where  $E(c_{mix})$  is the total effect caused by a mixture at the concentration of  $c_{mix}$ , and  $E(c_i)$  is the effect caused at  $c_i$  of individual chemicals. For a fitted function ( $f_i$ ) based on the concentration response data of the  $i^{th}$  component,  $E(c_i)$  is equal to  $f_i(c_i)$ . When  $E(c_{mix}) = x$ , (10) can be expressed as:

$$x\% = 1 - \prod_{i=1}^n \left( 1 - f_i(p_i(EC_{x,mix})) \right). \quad (11)$$

(11) can be used to predict the mixture effect for IA.

Generalized concentration addition (GCA) is a natural extension of CA that could be applied to mixtures including full agonists and partial agonists or mixtures including full agonists and competitive antagonists (Howard and Webster, 2009). The general form of GCA is expressed as follows:

$$\sum_{i=1}^n \frac{c_i}{f_i^{-1}(E)} = 1. \quad (12)$$

Empirical data were used to fit function  $f_i(c_i)$ , and then predict the mixture response using the inverse function  $f_i^{-1}(E)$ . Previous studies used the Hill function with Hill coefficient equal to 1 to fit individual concentration response curves (Howard et al., 2010). It is suitable to fit response data with lower limit fixed at 0 and no fixed maximal effect.

$$f(c) = \frac{\alpha c}{K + c}, \quad (13)$$

where  $c$  is the concentration,  $K$  is the concentration causes 50% effect, and  $\alpha$  is the maximal effect level of a chemical on a test organism. In **mixtox**, function (13) was labeled as Hill\_two. The equation to predict the effect of a multiple-component mixture based on Hill\_two is expressed as follows (Howard and Webster, 2009):

$$E_{mix}^{GCA} = \left( \sum_{i=1}^n \frac{\alpha_i c_i}{K_i} \right) / \left( 1 + \sum_{i=1}^n \frac{c_i}{K_i} \right). \quad (14)$$

## Overview of the package

This section provides a description of functions in the R package **mixtox**. We first detail the main functions that allow to fit concentration response data. Second, we describe a function to construct uniform design tables. Then we discuss functions for mixture toxicity prediction.

### Concentration-response curve fitting

Package **mixtox** provides 13 sigmoidal functions to fit monotonic concentration responses. First, the name of 13 sigmoidal functions will display through the command `showEq("sigmoid")`:

```
[1] "Hill"
[4] "Hill_four"
[7] "Weibull"
[10] "Logit"
[13] "GL(Generalized Logit)"

[1] "Hill_two"
[4] "Weibull"
[7] "Logit"
[10] "BCW(Box-Cox-Weibull)"
[13] "BCL(Box-Cox-Logit)"
```

Then the formula of those functions can be displayed through querying the function name using `showEq` (e.g., `showEq("Hill")`). Six functions (i.e., Hill, Weibull, Logit, BCW, BCL, and GL) are suitable to fit quantal responses in the range of [0, 1]. The other 7 functions are suitable to fit continuous concentration responses with response range out of [0, 1].

The main function for curve fitting in **mixtox** is `curveFit`. This function provides various options. It requires concentration (x), experimental responses (expr), a suitable equation (eq), and corresponding

starting values (param).

```
R> curveFit <- function(x, expr, eq, param, effv, sigLev = 0.05)
```

The eq term can be any of the 13 sigmoidal equations. The default significant level (sigLev) is 0.05 to calculate the *PI* and *CI*. Dunnett test (Dunnett, 1964) was used to calculate the non-observed effect concentration (NOEC) and lowest observed effect concentration (LOEC) for quantal concentration response data or response data with lower limit fixed at 0. The response values for the control in those data were set to 0 in the calculation of NOEC and LOEC.

```
R> NOEC <- function(x, expr, sigLev = 0.05)
```

The starting values (param) are indispensable for curveFit. The function tuneFit could help users to find proper starting values. Users need to provide their experimental concentration (conc), corresponding responses (rspn) and the equation (eq) they choose to fit the data (e.g., Weibull). Other terms (effv, rsq, highBar, bar, and sav) are optional and the detailed explanation can be found in the reference manual.

```
R> tuneFit <- function(conc, rspn, eq = "Weibull", effv, rsq = 0.6, highBar = 5000,
+   bar = 1000, sav = FALSE)
```

tuneFit is essentially a sophisticated wrapper for nlsLM in the package **minpack.lm**. Like the function **drm** in package **drc** (Ritz and Streibig, 2005) which provides self starting values, tuneFit also provides starting values for all of the 13 sigmoidal functions. Those starting values were collected through the fitting of various concentration response data from various sources (e.g., our lab tests, published literature, and large databases such as ToxCast; Kavlock et al. 2012). The starting values for all of the 13 functions are stored in **staval** (e.g., **staval\$Logit** would show all of the 1786 starting values for the Logit function). tuneFit provides a high frequency trial and error approach to deploy those starting values one by one until getting the best fit. This approach is quite efficient especially for the quantal concentration response data.

### Construction of a uniform design table

The function **unidTab** can be called to generate uniform design tables.

```
R> unidTab <- function(lev, fac, algo = "cd2")
```

To call **unidTab**, users need to provide the number of runs (levels, lev) and factors (fac). Normally, the number of runs equals the number of levels. If users need to do more runs than levels (the number of runs is  $n$  times that of levels,  $n = 1, 2, \dots$ ), a pseudo-level design needs to be performed. Usually, **unidTab** is incorporated into functions (e.g., **caPred** and **iaPred**) to predict toxicity of mixtures. Once users provide the levels (pseudo-levels) and fitting information of individual compounds, the **unidTab** will be called by **caPred** and **iaPred** to generate uniform table(s) with the lowest discrepancy.

### Mixture toxicity prediction

The function **caPred** can be called to predict the effect of mixtures based on concentration addition.

```
R> caPred <- function(model, param, mixType = c("acr", "eecr", "udcr"), effv,
+   effPoints)
```

Mixture toxicity prediction is based on the fitting information of individual concentration response curves. Users need to fit the individual concentration response data first and provide the fitting information (model used to fit individual concentration responses and corresponding fitted parameters param) to **caPred**. Function **caPred** provides three optional fix-ratio ray design methods: (1) arbitrary concentration ratio (acr), users can arbitrarily define the proportion of a component in a mixture. It also allows users to design mixtures according to certain experimental design methods (e.g., Latin hypercube design; Dette and Pepelyshev 2010); (2) equal effect concentration ratio (eecr); and (3) uniform design concentration ratio (udcr). If the mixture type is acr, the term effv is a vector of ratios or concentrations that will eventually be converted into the proportion of mixture components. If the mixture type is eecr or udcr, the argument effv is a numeric vector with single or multiple effect values. The argument effPoints is a vector of effect values (with a range of [0, 1]).

The function **iaPred** can be called to predict the effect of mixtures based on IA.

```
R> iaPred <- function(model, param, mixType = c("acr", "eecr", "udcr"), effv,
+   effPoints, lb = 1e-9, ub = 6)
```

Most of the arguments in `iaPred` are the same as those in `caPred`. The arguments of `lb` and `ub` are the lower and upper bounds of the concentration range where to find a solution for the constructed IA equation based on (11), respectively. Users can change these values based on their practical experimental system.

The function `caPred` can only predict mixture effects of chemicals with quantal concentration responses. That is, `caPred` only covers concentration response curves fitted by six functions (i.e., Hill, Weibull, Logit, BCW, BCL, and GL).

The function `gcaHill` can be used to predict the effect of mixtures based on (14) on the condition that all individual concentration responses should be fitted by the function `Hill_two`.

```
R> gcaHill <- function(model, param, mixType = c("acr", "eecr", "udcr"), effv,
+   refEffv = c(0.10, 0.50))
```

The reference effects (`refEffv`) need to be provided by users to define the range of responses. To extend the GCA to be more general, 12 more functions (Hill, Hill\_three, Hill\_four, Weibull, Weibull\_three, Weibull\_four, Logit, Logit\_three, Logit\_four, BCW, BCL, and GL) were incorporated to `gcaPred` according to the general form of GCA equation in (12).

```
R> gcaPred <- function(model, param, mixType = c("acr", "eecr", "udcr"), effv,
+   refEffv = c(0.05, 0.50, 0.90), lb = 1E-8, ub = 0.90)
```

## Illustrations

In this section, we illustrate several examples of the use of the functions described above. Example 1 deals with curve fitting. Example 2 deals with experimental design and mixture toxicity prediction based on CA, IA, and GCA. Example 3 deals with the connection of mixture toxicity prediction in `mixtox` with curve fitting in `drc`. Example 4 deals with follow-up analysis on mixture prediction. The `mixtox` package provides sigmoidal concentration response data (`antibiotox`) to demonstrate the usage of different functions.

The `antibiotox` dataset includes the long term toxicity of seven aminoglycosides (paromomycin sulfate (PAR), spectinomycin dihydrochloridehydrate (SPE), kanamycin sulfate (KAN), streptomycin sulfate (STR), dihydrostreptomycin sesquisulfate hydrate (DIH), gentamycin sulfate (GEN), and Neomycin sulfate (NEO)) and their 12 mixtures designed using different methods on fresh water photobacteria *Vibro-qinghaiensis* sp. Q67. The concentration unit of these antibiotics is mole per liter (mol/L). The bioluminescence of photobacteria was transformed into quantal form with response approaches 0 if chemical's concentration was extremely low and 100% inhibition if chemical's concentration was infinitely high. Detailed information on these data and test systems can be found in the reference manual.

### Example 1: Curve fitting

Mixture toxicity prediction is based on the curve fitting information of individual compounds. For example, we need to use the `Logit` function to fit the concentration response data of PAR on photobacteria. To get proper starting values of  $\alpha$  (location parameter) and  $\beta$  (slope parameter) for `Logit`, users need to use the function `tuneFit`.

```
R> x <- antibiotox$PAR$x
R> expr <- antibiotox$PAR$y
R> y <- rowMeans(expr)
R> tuneFit(x, y, eq = "Logit")
```

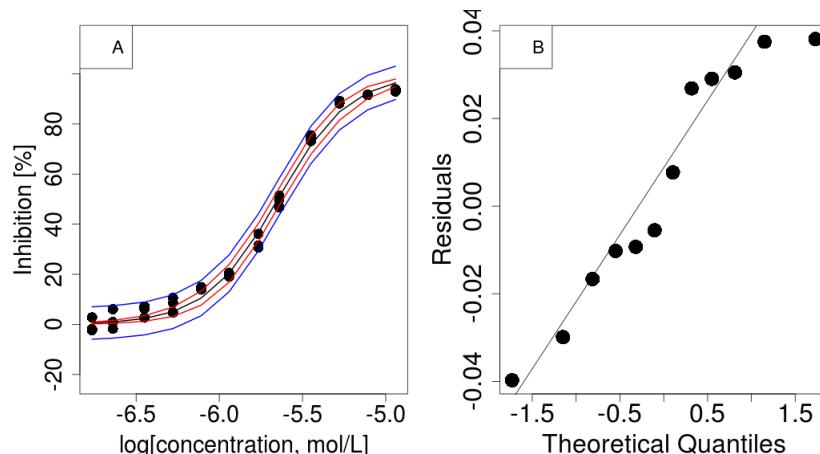
The function `tuneFit` will return a list as follows:

```
$sta
  Alpha Beta    r2 adjr2     MAE    RMSE    AIC    AICc    BIC      ecx
fit_1  26.3 4.66 0.995 0.994 0.0234 0.029 -49.1 -47.8 -48.2 1000001
```

The number 1000001 means the effect concentration (`ecx`) was not calculated because no `effv` was provided to `tuneFit`. To compare the result with Weibull:

```
R> tuneFit(x, y, eq = "Weibull")

$sta
  Alpha Beta    r2 adjr2     MAE    RMSE    AIC    AICc    BIC      ecx
fit_1  18.3 3.32 0.993 0.992 0.0223 0.0331 -46 -44.6 -45 1000001
```



**Figure 1:** The fitting information of antibiotic PAR on photobacteria: (A) concentration response curves. Dot: observed; Black line: fitted CRC; Red dotted lines: 95% CI; Blue dotted lines: 95% PI; and (B) normal Q-Q plots of residuals.

These two functions could well describe the concentration response data of PAR on photobacteria ( $r^2 > 0.99$ ). It is difficult to distinguish which fit is better purely based on  $r^2$ . Users need to choose the better fit (i.e., Logit) based on AIC or BIC (the lower, the better).

Function `tuneFit` also provides an important feature of fitting a batch of concentration response data of different chemicals. Here, we would like to example curve fitting of two compounds in a single run.

```
R> omim.x <- cytotox$Omim$x
R> omim.y <- rowMeans(cytotox$Omim$y)
R> emim.x <- cytotox$Emim$x
R> emim.y <- rowMeans(cytotox$Emim$y)
R> x <- rbind(omim.x, emim.x)
R> y <- rbind(omim.y, emim.y)
R> tuneFit(x, y, eq = "Weibull")

$sta
  Alpha Beta    r2 adjr2     MAE    RMSE     AIC   AICc     BIC     ecx
fit_1  6.42 2.25 0.946 0.941 0.0447 0.0625 -30.7 -29.3 -29.7 1000001
fit_2  5.57 2.27 0.974 0.972 0.0452 0.0545 -34.0 -32.6 -33.0 1000001
```

In this case,  $r^2$  is still useful to give an illustrative evaluation of the goodness of fit. In cases of various curves fitted by different functions, there is no way to compare their goodness of fit just based on AIC or BIC. Thus, we provide as many statistics as possible to allow users to choose the best fit in the context of their experimental systems.

Function `curveFit` could be used to fit the concentration response data of PAR using the starting values of 26.31 and 4.66 for Logit.

```
R> fit <- curveFit(x, expr, eq = "Logit", param = c(26.31, 4.66), effv = c(0.05, 0.5))
```

The plot of the fitted concentration response curve (Figure 1A) can be shown through function `figPlot`. The residuals of curve fitting can be found in `fit$res`. The normal Q-Q plot of residuals was plotted in Figure 1B using function `qq4res` in `mixtox`. The R code for Figure 1 is as follows:

```
R> par(mfrow = c(1, 2))
R> x <- antibiotox$PAR$x
R> expr <- antibiotox$PAR$y
R> fit <- curveFit(x, expr, eq = "Logit", param = c(26.31, 4.66), effv = c(0.05, 0.5))
R> figPlot(fit$crcInfo)
R> legend("topleft", legend = "A", border = "white", cex = 1.4)
R> qq4res(fit$res)
R> legend("topleft", legend = "B", border = "white", cex = 1.4)
```

The `effv` values of 0.05 and 0.5 imply that the effect concentrations that cause 5% and 50% effect (EC5 and EC50) are calculated. *PIs* and *CIs* were both calculated for comparison. The fitted  $\alpha$  and  $\beta$  are 26.31 and 4.66, respectively. The goodness of fit statistics are 0.995, 0.994, 0.02342, 0.02897,

$-49.13$ ,  $-47.80$ , and  $-48.16$  for  $R^2$ ,  $R_{adj}^2$ , *MAE*, *RMSE*, *AIC*, *AICc* and *BIC*, respectively. The function *curveFit* will call *ECx* to calculate effect concentration of *effv*. The EC50 of PAR on the inhibition of luminescence of phototobacteria is  $2.25\text{E-}06$  mol/L with 95% *CI* of  $[2.11\text{E-}06, 2.40\text{E-}06]$  mol/L and 95% *PI* of  $[1.95\text{E-}06, 2.57\text{E-}06]$  mol/L. The EC5 of PAR on the inhibition of luminescence of phototobacteria is  $5.24\text{E-}07$  mol/L with 95% *CI* of  $[4.37\text{E-}07, 6.44\text{E-}07]$  mol/L and 95% *PI* of  $[0, 8.44\text{E-}07]$  mol/L. Similarly, PAR will cause 50% inhibition with 95% *PI* of  $[42.7\%, 57.3\%]$  and 95% *CI* of  $[46.5\%, 53.5\%]$  at the concentration of  $2.25\text{E-}06$  mol/L. PAR will cause 5% inhibition with 95% *PI* of  $[0\%, 11.7\%]$  and 95% *CI* of  $[3.14\%, 6.86\%]$  at the concentration of  $5.24\text{E-}07$  mol/L.

The NOEC and LOEC can be calculated using the function *NOEC*:

```
R> NOEC(x, expr)
```

The values of NOEC and LOEC are  $2.29\text{E-}07$  mol/L and  $3.55\text{E-}07$  mol/L for PAR, respectively.

### Example 2: Mixture toxicity prediction

The function *caPred* can be called to predict the effect of mixtures based on concentration addition. It provides three optional mixture design methods as mentioned before (ACR, EECR, and UDCR). The effect of mixtures of seven aminoglycosides combined on photobacteria according to the EECR design for 5% and 50% effect levels can be predicted by:

```
R> model <- antibiotox$sgl$model
R> param <- antibiotox$sgl$param
R> caPred(model, param, mixType = "eecr", effv = c(0.05, 0.5))
```

Function *caPred* will return a series of effects *e* in a range of  $[0, 1]$ , predicted concentrations *ca* that cause effects *e*, and the proportion of individual components (*pct*) for the EECR mixture. The proportion of individual components is very useful for users to prepare mixtures in practical experiments.

Here, we want to design 10 UDCR mixtures based on seven antibiotics and five concentration levels (effect concentrations at the responses of 5%, 10%, 20%, 30% and 50%, respectively). The function *unidTab* can be called in *caPred* to generate a uniform design table of  $U_{10}(10^7)$ . The uniform table can also be generated with 10 runs (pseudo-levels) and 7 factors as follows:

```
R> unidTab(10, 7)
```

No	1	2	3	4	5	6	9
1	1	2	3	4	5	6	9
2	2	4	6	8	10	1	7
3	3	6	9	1	4	7	5
4	4	8	1	5	9	2	3
5	5	10	4	9	3	8	1
6	6	1	7	2	8	3	10
7	7	3	10	6	2	9	8
8	8	5	2	10	7	4	6
9	9	7	5	3	1	10	4
10	10	9	8	7	6	5	2

The CA prediction of those UDCR mixtures is as follows:

```
R> caPred(model, param, mixType = "udcr", effv = rep(c(0.05, 0.1, 0.2, 0.3, 0.5), 2))
```

The argument *effv* exemplifies the pseudo-level design of the five levels. Function *caPred* also returns the uniform table employed to construct UDCR mixtures.

Similarly, the effect of mixtures based on IA for the EECR and UDCR design can be predicted as follows:

```
R> iaPred(model, param, mixType = "eecr", effv = c(0.05, 0.5))
R> iaPred(model, param, mixType = "udcr", effv = rep(c(0.05, 0.1, 0.2, 0.3, 0.5), 2))
```

Both the *caPred* and *iaPred* would return the proportion of individual compounds in each mixture.

Users need to fit all of the individual concentration response data using *Hill\_two* (13) before using function *gcaHill*. Assume we have fitted four curves using *Hill\_two* with corresponding fitting information stored in *model\_hill12* and *param\_hill12*. The effect of mixtures predicted based on generalized concentration addition for the EECR and UDCR design can be calculated as follows:

```
R> model_hill2 <- c("Hill_two", "Hill_two", "Hill_two", "Hill_two")
R> param_hill2 <- matrix(c(3.94e-5, 0.97, 5.16e-4, 1.50, 3.43e-6, 1.04, 9.18e-6, 0.77),
+   nrow = 4, ncol = 2, byrow = TRUE)
R> gcaHill(model_hill2, param_hill2, mixType = "eecr", effv = c(0.05, 0.5))
R> effv <- c(0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.50)
R> gcaHill(model_hill2, param_hill2, mixType = "udcr", effv)
```

### Example 3: Connection of mixtox with drc

Package **drc** is widely used for curve fitting of concentration response data. Many users are accustomed to its convenience in curve fitting. The two-parameter log-logistic function (*LL.2*) and two-parameter Weibull functions (*W1.2* and *W2.2*) with the lower limit fixed at 0 and the upper limit fixed at 1 in **drc** are suitable for quantal responses. The function **drm** is a general model fitting function for analysis of concentration response data in **drc**.

Here we would like to show an example of connecting the curve fitting (**drm**) in **drc** with the mixture effects prediction (**caPred** and **iaPred**) in **mixtox**. Assume we need to predict the effect of mixtures of four antibiotics (PAR, SPE, KAN, and STR) on photobacteria. First, we need to fit these concentration response data (stored in **antibiotox**) using **drm**. The *LL.2* function is selected to fit these response data.

```
R> library(mixtox)
R> library(drc)
R> PAR.x <- antibiotox$PAR$x
R> PAR.y <- rowMeans(antibiotox$PAR$y)
R> PAR <- data.frame(x = PAR.x, y = PAR.y)
R> SPE <- data.frame(x = antibiotox$SPE$x, y = rowMeans(antibiotox$SPE$y))
R> KAN <- data.frame(x = antibiotox$KAN$x, y = rowMeans(antibiotox$KAN$y))
R> STR <- data.frame(x = antibiotox$STR$x, y = rowMeans(antibiotox$STR$y))

R> PAR.fit <- drm(y ~ x, data = PAR, fct = LL.2())
R> SPE.fit <- drm(y ~ x, data = SPE, fct = LL.2())
R> KAN.fit <- drm(y ~ x, data = KAN, fct = LL.2())
R> STR.fit <- drm(y ~ x, data = STR, fct = LL.2())
R> param.LL.2 <- rbind(PAR.fit$fit$par, SPE.fit$fit$par, KAN.fit$fit$par,
+   STR.fit$fit$par)
R> rownames(param.LL.2) <- c("PAR", "SPE", "KAN", "STR")
R> colnames(param.LL.2) <- c("b", "e")
```

The fitted parameters  $b$  and  $e$  for *LL.2* (Ritz and Streibig, 2005) are stored in **param.LL.2**. The formula of *LL.2* in **drc** and Logit in **mixtox** seem different, but they are essentially the same expression. The following equations can be used to transform  $b$  and  $e$  in *LL.2* into  $\alpha$  and  $\beta$  in Logit, respectively.

$$\alpha = -\log(e) \cdot \beta, \quad \beta = -b \cdot \ln(10). \quad (15)$$

The following code can be used to transform  $b$  and  $e$  in *LL.2* into  $\alpha$  and  $\beta$  in Logit numerically.

```
R> Beta <- -param.LL.2[, 1] * log(10)
R> Alpha <- -log10(param.LL.2[, 2]) * Beta
R> param.Logit <- cbind(Alpha, Beta)
```

The effect of mixtures of four antibiotics on photobacteria according to, say, the EECR design at 5% and 50% effect levels can be predicted by:

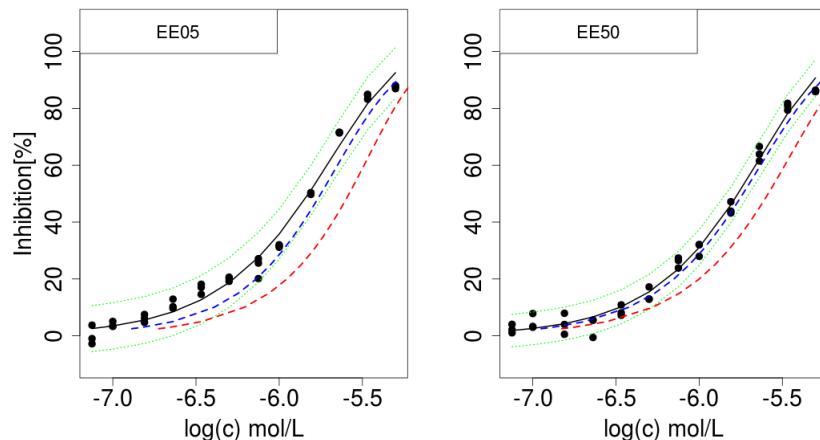
```
R> model.Logit <- c("Logit", "Logit", "Logit", "Logit")
R> caPred(model.Logit, param.Logit, mixType = "eecr", effv = c(0.05, 0.5))
R> iaPred(model.Logit, param.Logit, mixType = "eecr", effv = c(0.05, 0.5))
```

### Example 4: Follow-up analysis

Figure 2 shows the comparison of CA and IA prediction with experimental observation of EECR mixtures. The code for this comparison is shown in Appendix.

We can conclude that the mixture toxicity of seven antibiotics can be predicted by the IA model since the IA curve locates in the 95% PI of the experimental curve.

Despite the CA and IA prediction at particular effect concentrations, it is also meaningful to know the CA and IA prediction at particular effects. For example, the luminescence of photobacteria will



**Figure 2:** Experimental, predicted CA, and IA concentration response curves of 2 mixtures designed by equal effect concentration ratio (EE05 and EE50). Dot: observed; Black line: fitted CRC; Grey dotted lines: 95% PI; Blue dashed line: IA prediction; Red dashed line: CA. The code is listed in [Appendix](#).

be inhibited by 5% at the concentration of  $1.41\text{E}-7 \text{ mol/L}$  and  $1.77\text{E}-7 \text{ mol/L}$  for EE05 and EE50 mixtures, respectively. The CA and IA prediction at the concentration of  $1.41\text{E}-7 \text{ mol/L}$  (EE05) and  $1.77\text{E}-7 \text{ mol/L}$  (EE50) can be calculated based on (7) and (10), respectively. The dichotomy technique was employed to solve the constructed CA equation based on (7). This calculation requires fitted concentration response information of individual chemicals, their ratios in the mixtures, and the established concentration response information of mixtures. Function `eiaPred` and `ecaPred` can be called as follows:

```
R> model <- antibiotox$eecr.mix$model
R> param <- antibiotox$eecr.mix$param
R> pct <- antibiotox$eecr.pct
R> mix <- list(model = model, param = param)
R> eiaPred(effv = 0.05, antibiotox$sgl, mix, pct)
R> ecaPred(effv = 0.05, antibiotox$sgl, mix, pct)
```

The two functions first calculate the effect concentrations based on the fitted concentration response information of mixtures (i.e., the selected equations and associated coefficients contained in `antibiotox$eecr.mix`) according to the input effects `effv` (e.g., 5.0%). The concentration of an individual component ( $c_i$ ) is computed from (8) based on the mixture's effect concentration and the ratio of components in the mixture `antibiotox$pct` ( $p_i$ ). The IA predicted effect for EE5 and EE50 are 1.80% and 3.04%, respectively. The CA predicted effect of EE5 and EE50 are 2.78% and 4.25%, respectively.

## Conclusions

Package **mixtox** mainly targets toxicologists in the study of chemicals' toxicity and the effect of mixtures. It offers a general framework of curve fitting, experimental design, and mixture prediction for practitioners in toxicology. The unique features of **mixtox** include: (1) construction of uniform design table; and (2) mixture toxicity prediction for multiple components based on CA, IA, and GCA. Function `gcaHill` is capable of examining mixtures containing partial agonists or even full agonists. However, we are lacking experimental data in **mixtox** to verify its ability. **mixtox** aims to predict the effect of mixtures with more than two components. It cannot make isoboles for binary mixtures. Users may need to draw isoboles using package **drc** or other software package based on the results calculated with **mixtox**. The current version of **mixtox** (version number 1.3.1) is available on CRAN. We received a lot of feedback from users since its release last year. Now it is still under rapid development to improve its performance and try to include more features in toxicology.

## Acknowledgement

This work was financed by the National Natural Science Foundation of China (No. 21407087) and the Startup Foundation for Advanced Talents (No. 6631113336 and No. 6631114328) of Qingdao Agricultural University.

## Bibliography

- T. Backhaus and M. Faust. Predictive environmental risk assessment of chemical mixtures: A conceptual framework. *Environmental Science & Technology*, 46(5):2564–2573, 2012. doi: 10.1021/es2034125. [p422]
- S. De Gryze, I. Langhans, and M. Vandebroek. Using the correct intervals for prediction: A tutorial on tolerance intervals for ordinary least-squares regression. *Chemometrics and Intelligent Laboratory Systems*, 87(2):147–154, 2007. doi: 10.1016/j.chemolab.2007.03.002. [p424]
- H. Dette and A. Pepelyshev. Generalized latin hypercube design for computer experiments. *Technometrics*, 52(4):421–429, 2010. doi: 10.1198/tech.2010.09157. [p424, 426]
- C. W. Dunnett. New tables for multiple comparisons with a control. *Biometrics*, 30(3):482–491, 1964. doi: 10.2307/2528490. [p426]
- R. Dybowski and V. Gant. *Clinical Applications of Artificial Neural Networks*. Cambridge University Press, Cambridge, 1st edition, 2001. doi: 10.1017/cbo9780511543494. [p423]
- T. V. Elzhov, K. M. Mullen, A.-N. Spiess, and B. Bolker. *minpack.lm: R Interface to the Levenberg-Marquardt Nonlinear Least-Squares Algorithm Found in MINPACK, Plus Support for Bounds*, 2016. URL <https://CRAN.R-project.org/package=minpack.lm>. R package version 1.2-1. [p423]
- M. Faust, R. Altenburger, T. Backhaus, H. Blanck, W. Boedeker, P. Gramatica, V. Hamer, M. Scholze, M. Vighi, and L. H. Grimme. Predicting the joint algal toxicity of multi-component s-triazine mixtures at low-effect concentrations of individual toxicants. *Aquatic Toxicology*, 56(1):13–32, 2001. doi: 10.1016/s0166-445x(01)00187-4. [p422, 424, 425]
- M. Faust, R. Altenburger, T. Backhaus, H. Blanck, W. Boedeker, P. Gramatica, V. Hamer, M. Scholze, M. Vighi, and L. H. Grimme. Joint algal toxicity of 16 dissimilarly acting chemicals is predictable by the concept of independent action. *Aquatic Toxicology*, 63(1):43–63, 2003. doi: 10.1016/s0166-445x(02)00133-9. [p422]
- S. Goutelle, M. Maurin, F. Rougier, X. Barbaut, L. Bourguignon, M. Ducher, and P. Maire. The hill equation: a review of its capabilities in pharmacological modelling. *Fundamental & Clinical Pharmacology*, 22(6):633–648, 2008. doi: 10.1111/j.1472-8206.2008.00633.x. [p422]
- N. Hadrup, C. Taxvig, M. Pedersen, C. Nellemann, U. Hass, and A. M. Vinggaard. Concentration addition, independent action and generalized concentration addition models for mixture effect prediction of sex hormone synthesis in vitro. *PLoS ONE*, 8(8):e70490, 2013. doi: 10.1371/journal.pone.0070490. [p423]
- F. J. Hickernell. A generalized discrepancy and quadrature error bound. *Mathematics of Computation*, 67(211):299–322, 1996. doi: 10.1090/s0025-5718-98-00894-1. [p424]
- G. J. Howard and T. F. Webster. Generalized concentration addition: A method for examining mixtures containing partial agonists. *Journal of Theoretical Biology*, 259(3):469–477, 2009. doi: 10.1016/j.jtbi.2009.03.030. [p423, 425]
- G. J. Howard, J. J. Schlezinger, M. E. Hahn, and T. F. Webster. Generalized Concentration Addition Predicts Joint Effects of Aryl Hydrocarbon Receptor Agonists with Partial Agonists and Competitive Antagonists. *Environmental Health Perspectives*, 118(5):666–672, 2010. doi: 10.1289/ehp.0901312. [p425]
- S. Huet, A. Bouvier, M.-A. Poursat, and E. Jolivet. *Statistical Tools for Nonlinear Regression: A Practical Guide with S-PLUS and R Examples*. Springer-Verlag, 2004. doi: 10.1007/978-1-4757-2523-0. [p424]
- Z. N. Kamvar. *eze: Easy interface to effective concentration calculations*, 2016. URL <https://CRAN.R-project.org/package=eze>. R package version 1.0.1. [p422]
- R. Kavlock, K. Chandler, K. Houck, S. Hunter, R. Judson, N. Kleinstreuer, T. Knudsen, M. Martin, S. Padilla, D. Reif, A. Richard, D. Rotroff, N. Sipes, and D. Dix. Update on EPA’s ToxCast program: Providing high throughput decision support tools for chemical risk management. *Chemical Research in Toxicology*, 25(7):1287–1302, 2012. doi: 10.1021/tx3000939. [p426]
- Y.-Z. Liang, K.-T. Fang, and Q.-S. Xu. Uniform design and its applications in chemistry and chemical engineering. *Chemometrics and Intelligent Laboratory Systems*, 58(1):43–57, 2001. doi: 10.1016/s0169-7439(01)00139-3. [p424]

- J. G. Liao and D. McGee. Adjusted coefficients of determination for logistic regression. *The American Statistician*, 57(3):161–165, 2003. doi: 10.1198/0003130031964. [p<sup>423</sup>]
- S.-S. Liu, J. Zhang, Y.-H. Zhang, and L.-T. Qin. APTox: Assessment and prediction on toxicity of chemical mixtures. *Acta Chimica Sinica*, 70(14):1511–1517, 2012. doi: 10.6023/a12050175. [p<sup>422</sup>]
- S.-S. Liu, Q.-F. Xiao, J. Zhang, and M. Yu. Uniform design ray in the assessment of combined toxicities of multi-component mixtures. *Science Bulletin*, 61(1):52–58, 2015. doi: 10.1007/s11434-015-0925-6. [p<sup>422</sup>, p<sup>424</sup>]
- J.-H. Ning, K.-T. Fang, and Y.-D. Zhou. Uniform design for experiments with mixtures. *Communications in Statistics – Theory and Methods*, 40(10):1734–1742, 2011. doi: 10.1080/03610921003637470. [p<sup>422</sup>, p<sup>424</sup>]
- J. Ranke. *drlfit: Dose-Response Data Evaluation*, 2016. URL <https://CRAN.R-project.org/package=drlfit>. R package version 0.6.7. [p<sup>422</sup>]
- C. Ritz and J. C. Streibig. Bioassay analysis using R. *Journal of Statistical Software*, 12(5):1–22, 2005. doi: 10.18637/jss.v012.i05. [p<sup>422</sup>, p<sup>423</sup>, p<sup>426</sup>, p<sup>430</sup>]
- C. Ritz and J. C. Streibig. From Additivity to Synergism – A Modelling Perspective. *Synergy*, 1(1):22–29, 2014. doi: 10.1016/j.synres.2014.07.010. [p<sup>423</sup>]
- M. Scholze, W. Boedeker, M. Faust, T. Backhaus, R. Altenburger, and L. H. Grimme. A general best-fit method for concentration-response curves and the estimation of low-effect concentrations. *Environmental Toxicology and Chemistry*, 20(2):448–457, 2001. doi: 10.1002/etc.5620200228. [p<sup>422</sup>]
- A.-N. Spiess and N. Neumeyer. An evaluation of  $R^2$  as an inadequate measure for nonlinear models in pharmacological and biochemical research: A Monte Carlo approach. *BMC Pharmacology*, 10(6):11, 2010. doi: 10.1186/1471-2210-10-6. [p<sup>422</sup>, p<sup>423</sup>]
- Y.-D. Zhou and K.-T. Fang. An efficient method for constructing uniform designs with large size. *Computational Statistics*, 28(3):1319–1331, 2013. doi: 10.1007/s00180-012-0359-4. [p<sup>424</sup>]
- X. Zhu. *mixtox: Curve Fitting and Mixture Toxicity Assessment*, 2016. URL <https://CRAN.R-project.org/package=mixtox>. R package version 1.3.1. [p<sup>423</sup>]
- X.-W. Zhu, S.-S. Liu, L.-T. Qin, F. Chen, and H.-L. Liu. Modeling non-monotonic dose–response relationships: Model evaluation and hormetic quantities exploration. *Ecotoxicology and Environmental Safety*, 89:130–136, 2013. doi: 10.1016/j.ecoenv.2012.11.022. [p<sup>424</sup>]

## Appendix

```

library(mixtox)
par(mfrow = c(1, 2))
model <- antibiotox$sgl$model
param <- antibiotox$sgl$param
eeea <- caPred(model, param, mixType = "eecd", effv = c(0.05, 0.5))
eeia <- iaPred(model, param, mixType = "eecd", effv = c(0.05, 0.5))
# plot EE05 mixture
par(mar = c(5, 5, 1, 1))
x <- antibiotox$ee05$x
expr <- antibiotox$ee05$y
ee05fit <- curveFit(x, expr, eq = antibiotox$eecd.mix$model[1],
                      param = antibiotox$eecd.mix$param[1, ])
plot(rep(log10(x), ncol(expr)), expr * 100, pch = 20, ylim = c(-10, 110),
      xlab = "log(c) mol/L", ylab = "Inhibition[%]", cex = 1.8, cex.lab = 1.8,
      cex.axis = 1.8)
lines(log10(x), ee05fit$crcInfo[, 2] * 100, col = 1, lwd = 2)
lines(log10(x), ee05fit$crcInfo[, 6] * 100, col = "green", lwd = 1.5, lty = 3)
lines(log10(x), ee05fit$crcInfo[, 7] * 100, col = "green", lwd = 1.5, lty = 3)
lines(log10(eeia$ia[1, ]), eeia$e * 100, col = "red", lwd = 2.5, lty = 2)
lines(log10(eeca$ca[1, ]), eeca$e * 100, col = "blue", lwd = 2.5, lty = 2)
legend("topleft", legend = "EE05", border = "white", cex = 1.4)
# plot EE50 mixture
par(mar = c(5, 5, 1, 1))
x <- antibiotox$ee50$x
expr <- antibiotox$ee50$y
ee50fit <- curveFit(x, expr, eq = antibiotox$eecd.mix$model[1],
                      param = antibiotox$eecd.mix$param[1, ])
plot(rep(log10(x), ncol(expr)), expr * 100, pch = 20, ylim = c(-10, 110),
      xlab = "log(c) mol/L", ylab = "", cex = 1.8, cex.lab = 1.8, cex.axis = 1.8)
lines(log10(x), ee50fit$crcInfo[, 2] * 100, col = 1, lwd = 2)
lines(log10(x), ee50fit$crcInfo[, 6] * 100, col = "green", lwd = 1.5, lty = 3)
lines(log10(x), ee50fit$crcInfo[, 7] * 100, col = "green", lwd = 1.5, lty = 3)
lines(log10(eeia$ia[2, ]), eeia$e * 100, col = "red", lwd = 2.5, lty = 2)
lines(log10(eeca$ca[2, ]), eeca$e * 100, col = "blue", lwd = 2.5, lty = 2)
legend("topleft", legend = "EE50", border = "white", cex = 1.4)

```

*Xiang-Wei Zhu  
 Qingdao Agricultural University  
 College of Resource and Environment  
 and  
 Qingdao Engineering Research Center for Rural Environment  
 266109 Qingdao, China  
 xwzhunc@gmail.com*

*Jian-Yi Chen  
 Qingdao Agricultural University  
 Science and Information College  
 266109 Qingdao, China  
 chenjy@amss.ac.cn*

# Weighted Distance Based Discriminant Analysis: The R Package WeDiBaDis

by Itziar Irigoien, Francesc Mestres, and Concepcion Arenas

**Abstract** The **WeDiBaDis** package provides a user friendly environment to perform discriminant analysis (supervised classification). **WeDiBaDis** is an easy to use package addressed to the biological and medical communities, and in general, to researchers interested in applied studies. It can be suitable when the user is interested in the problem of constructing a discriminant rule on the basis of distances between a relatively small number of instances or units of known unbalanced-class membership measured on many (possibly thousands) features of any type. This is a current situation when analyzing genetic biomedical data. This discriminant rule can then be used both, as a means of explaining differences among classes, but also in the important task of assigning the class membership for new unlabeled units. Our package implements two discriminant analysis procedures in an R environment: the well-known distance-based discriminant analysis (DB-discriminant) and a weighted-distance-based discriminant (WDB-discriminant), a novel classifier rule that we introduce. This new procedure is based on an improvement of the DB rule taking into account the statistical depth of the units. This article presents both classifying procedures and describes the implementation of each in detail. We illustrate the use of the package using an ecological and a genetic experimental example. Finally, we illustrate the effectiveness of the new proposed procedure (WDB), as compared with DB. This comparison is carried out using thirty-eight, high-dimensional, class-unbalanced, cancer data sets, three of which include clinical features.

## Introduction

Discriminant analysis (supervised classification) is used to differentiate between two or more naturally occurring groups based on a suite of discriminating features. This analysis can be used as a means of explaining differences among groups and for classification. That is, to develop a rule based on features measured on a group of units with known membership (the so-called training set), and to use this classification rule to assign a class membership to new unlabeled units. Classification is used by researchers in a wide variety of settings and fields including biological and medical sciences. For example, in biology it is used for taxonomic classification, morphometric analysis for species identification, and to study species distribution. Discriminant analysis is applicable to a wide range of ecological problems, e.g., testing for niche separation by sympatric species or for the presence or absence of a particular species. Marine ecologists commonly use discriminant analysis to evaluate the similarity of distinct populations and to classify units of unknown origin to known populations. The discriminant technique is also used in genetic studies in order to summarize the genetic differentiation between groups. In studies with Single Nucleotide Polymorphism (SNP) or re-sequencing data sets, usually the number of variables (alleles) is greater than the number of observations (units), so discriminant methods are available for data sets with more variables than units, as necessary. Furthermore, class prediction is currently one of the most important tasks in biomedical studies. The diagnosis of diseases, as cancer type or psychiatric disorder, has recently received a great deal of attention. With actual data, classification presents serious difficulties, because diagnosis is based on both clinical/pathological features (usually nominal data) and gene expression information (continuous data). For this reason, classification rules that could be applied to all types of data are desirable. The most popular classification rules are the linear (LDA) and quadratic (QDA) discriminant analyses (Fisher, 1936), which are easy to use as they are found in most statistical packages. However, they require the assumption of normally distributed data; when this condition is violated, their use may yield poor classification results. Many distinct classifiers exist, differing in the definition of the classification rule and whether they utilize statistical (Golub et al., 1999; Hastie et al., 2001) or machine learning (Breiman, 2001; Boulesteix et al., 2008) methods. However, the problem of classification with data obtained from microarrays is challenging because there are a large number of genes and a relatively small number of samples. In this situation, the classification methods based on the within-class covariance matrix fail, as an inverse is not defined. This is known as the singularity or under-sample problem (Krzanowski et al., 1995). The shrunken centroid method can be seen as a modification of the diagonal discriminant analysis (Dudoit et al., 2002) and was developed for continuous high-dimensional data (Tibshirani et al., 2002). Nowadays, another issue that requires attention is the class-unbalanced situation, that is, the number of units belonging to each class is not the same. Some classifiers on class-unbalanced data tend to classify most of the new data in the majority class. This bias is higher when using high dimensional data. Recently, a method which improves the shrunken centroid method when the high-dimensional data is class-unbalanced was presented

(Blagus and Lusa, 2013). Furthermore, some statistical approaches are characterized by having an explicit underlying probability model, but it is not possible to always assume this requirement. One of the most popular nonparametric, machine-learning, classification methods is the  $k$ -nearest neighbor classification ( $k$ -NN) (Cover and Hart, 1967; Duda et al., 2000). Given a new unit to be classified, this method finds the  $k$  nearest neighbors and classifies the new unit in the class to which belong the majority of neighbours. This classification may depend on the selected value for  $k$ . As ecologists have repeatedly argued, the Euclidean distance is inappropriate for raw species abundance data involving null abundances (Orloci, 1967; Legendre and Legendre, 1998) and it is necessary to use discriminant analyses that incorporate adequate distances. In this situation, discriminant analysis based on distances (DB-discriminant), where any symmetric distance or dissimilarity function can be used, is a useful alternative (Cuadras, 1989, 1992; Cuadras et al., 1997; Anderson and Robinson, 2003). To our knowledge, this technique is only included in GINGKO a suite of programs for multivariate analysis, oriented towards ordination and classification of ecological data (De Caceres et al., 2003; Bouin, 2005; Kent, 2011). These programs are written in Java language, so it is therefore necessary to have a Java Virtual Machine to execute it. Even though GINGKO is a very useful tool, it does not provide the option of a class prediction for new unlabeled units or feature selection. Recently, data depth was proposed as the basis for nonparametric classifiers (Jornstein, 2004; Ghosh and Chaudhuri, 2005; Jin and Cui, 2010; Hlubinka and Vencalek, 2013). A depth of a unit is a nonnegative number, which measures the centrality of the unit. That is, depth in the sample version reflects position of the unit with respect to the observed data cloud. The so-called maximal depth classifier is the simple and natural classifier defined from a depth function: to allocate a new observation to the class to which it has maximal depth. There are many possibilities how to define the depth of the data (Liu, 1990; Vardi and Zhang, 2000; Zuo and Serfling, 2000; Serfling, 2002), nevertheless the computation of the most popular depth functions is very slow, in particular, for high-dimensional data the time needed for classification grows rapidly. A new less-computer intensive depth function  $I$  (Irigoinen et al., 2013a) was developed, but the authors did not study its use in relation to the classification problem.

A discriminant method should have several abilities. First, the classifier rule has to be able to properly separate the classes. In this sense, the classifier evaluation is most often based on the error rate, the percentage of incorrect prediction divided by the total number of predictions. Second, the rule has to be useful to classify new unlabeled units. Then, cross validation evaluation is needed. Cross-validation involves a series of sub-experiments, each of which involves the removal of a subset of objects from a data set (the test set), construction of a classifier using the remaining objects in the data set (the model building set), and subsequent application of the resulting model to the removed objects. The leave-one-out method is a special case of cross-validation; it considers each single object in the data set as a test set. Furthermore, other measures, such as the sensitivity, specificity, positive predictive value for each class, and the generalized correlation coefficient, are useful to know the ability of the rule in the prediction task.

Here we introduce **WeDiBaDis**, an R package which provides a user-friendly interface to run the DB-discriminant analysis and a new classification procedure, the weighted-distance-based discriminant (WDB-discriminant) that performs well and improves the DB-discriminant rule. It is based on both, the DB-discriminant rule and the depth function  $I$  (Irigoinen et al., 2013a). First, we will describe the DB and WDB discriminant rules. Then, we will provide details about the **WeDiBaDis** package and will illustrate its use and its main outputs using an ecological and a genetic data set. To compare both DB and WDB rules—and in order to avoid the criticism that artificial data can favour particular methods—we present a large analysis of thirty-eight, high-dimensional, class-unbalanced, cancer gene expression data sets, three of which include clinical features. Furthermore, the data sets include more than two classes. Finally, we conclude the paper presenting the main conclusions. **WeDiBaDis** is available at <https://github.com/ItziarI/WeDiBaDis>.

## Discriminant rules and evaluation criteria

Let  $\mathbf{y}_i$  ( $i = 1, 2, \dots, n$ ) be  $m$ -dimensional units measured in any kind of features, with associated class labels  $l_i \in \{1, 2, \dots, K\}$ , where  $n$  and  $K$  denote the number of units and classes, respectively. Let  $\mathbf{Y}$  be the matrix of all units and  $d$  a distance defined between any pair of units,  $d_{ij} = d(\mathbf{y}_i, \mathbf{y}_j)$ . Let  $\mathbf{y}^*$  be a new unlabeled unit to be classified in one of the given classes  $C_k$ ,  $k = 1, 2, \dots, K$ .

### DB-discriminant

The distance-based or DB-discriminant rule (Cuadras et al., 1997) takes as a discriminant score

$$\delta_k^1(\mathbf{y}^*) = \hat{\phi}^2(\mathbf{y}^*, C_k), \quad (1)$$

where  $\hat{\phi}^2(\mathbf{y}^*, C_k)$  is the proximity function which measures the proximity between  $\mathbf{y}^*$  and  $C_k$ . This function is defined by,

$$\hat{\phi}^2(\mathbf{y}^*, C_k) = \frac{1}{n_k} \sum_{i=1}^{n_k} d^2(\mathbf{y}^*, \mathbf{y}_i) - \frac{1}{2n_k^2} \sum_{i,j=1}^{n_k} d^2(\mathbf{y}_i, \mathbf{y}_j), \quad (2)$$

where  $n_k$  indicates the number of units in class  $k$ . Note that the second term in (2),

$$\hat{V}(C_k) = \frac{1}{2n_k^2} \sum_{i,j=1}^{n_k} d^2(\mathbf{y}_i, \mathbf{y}_j),$$

called geometric variability of  $C_k$ , measures the dispersion of  $C_k$ . When  $d$  is the Euclidean distance,  $\hat{V}(C_k)$  is the trace of the covariance matrix of  $\mathbf{Y}$ .

The DB classification rule allocates  $\mathbf{y}^*$  to the class which has the minimal proximity value:

$$C_{DB}(\mathbf{y}^*) = l \quad \text{where} \quad \delta_l^1(\mathbf{y}^*) = \min_{k=1,\dots,K} \left\{ \delta_k^1(\mathbf{y}^*) \right\}. \quad (3)$$

That is, this distance-based rule assigns a unit to the nearest group. Furthermore, using appropriate distances, Equation (3) reduces to some classic and well-studied rules (see Table 1 in Cuadras et al. (1997)). For example, under the normality assumption, Equation (3) is equivalent to a linear discriminant or to a quadratic discriminant if the Mahalanobis distance or the Mahalanobis distance plus a constant is selected, respectively.

### WDB-discriminant

For any unit  $\mathbf{y}$ , let  $I_k$  be the depth function in class  $C_k$  defined by (Irigoien et al., 2013a),

$$I_k(\mathbf{y}) = \left[ 1 + \frac{\hat{\phi}^2(\mathbf{y}, C_k)}{\hat{V}(C_k)} \right]^{-1}. \quad (4)$$

Function  $I$  takes values in  $[0, 1]$  and it verifies the following desirable properties: For a distribution having a uniquely defined "center"  $I$  attains maximum value at this center (maximality at center); When one unit moves away from the deepest unit (the unit at which the depth function attains maximum value; in particular, for a symmetric distribution, the center) along any fixed ray through the center, the depth at this unit decreases monotonically (monotonicity relative to the deepest point) and the depth of a unit  $\mathbf{y}$  should approach zero as  $\|\mathbf{y}\|$  approaches infinity (vanishing at infinity). According to the distance used, the depth of a unit may or may not depend on the underlying coordinate system or, in particular, of the scales of the underlying measurements. In any case the affine invariance holds for translations and rotations. Thus, according to Zuo and Serfling (2000),  $I$  is a *type C* depth function. As  $I$  is a depth function, it assigns to any observation a degree of centrality. While most of the depth functions assign zero depth to units outside a convex hull and then, it is possible that some training units have zero depth, the function in Equation (4) attains the zero value if  $V(C_k) = 0$ , that is, in presence of a constant distribution.

For each class  $C_k$  we weight the discriminant score  $\delta_k^1$  by  $1 - I_k(\mathbf{y}^*)$ , that is, given a new unit  $\mathbf{y}^*$ , we define a new discriminant score for class  $k$  by:

$$\delta_k^2(\mathbf{y}^*) = \delta_k^1(1 - I_k(\mathbf{y}^*)) = \phi^2(\mathbf{y}^*, C_k)(1 - I_k(\mathbf{y}^*)). \quad (5)$$

The shrinkage we use, reduces the proximity values, this reduction being greater for deeper units. Thus, this new classification rule,

$$C_{WDB}(\mathbf{y}^*) = l \quad \text{where} \quad \delta_l^2(\mathbf{y}^*) = \min_{k=1,\dots,K} \left\{ \delta_k^2(\mathbf{y}^*) \right\}, \quad (6)$$

allocates a new unit  $\mathbf{y}^*$  to the class which has the minimal proximity and maximal depth values.

### Evaluation criteria

First consider the case of two classes ( $K = 2$ ) and the most common measures of performance for a classification rule. As it is usual in medical statistics, for a fixed class  $k$ , let TP, FN, FP, and TN denote the true positive (number of units of class  $k$  correctly classified in class  $k$ ), the false negative (number of units of class  $k$  misclassified as units in class  $l$ , with  $l \neq k$ ), the false positive (number of units of class  $l$ , with  $l \neq k$  misclassified as units in class  $k$ ), and the true negative (number of units of

class  $l$ , with  $l \neq k$  correctly classified as units in class  $l$ ), respectively. Then (Zhou et al., 2002), the sensitivity (recall) for class  $k$  is defined as the ability of a rule to correctly classify units belonging to class  $k$ , thus  $Q_k^{se} = \frac{TP}{TP+FN}$ . The specificity is the ability of a rule to correctly exclude a unit from class  $k$  when it really belongs to another class, thus  $Q_k^{sp} = \frac{TN}{TN+FP}$ . Furthermore, the positive predictive value (precision) is the probability that a classification in class  $k$  is correct, thus  $P_k^+ = \frac{TP}{TP+FP}$  and the negative predictive value is the probability that a classification in class  $l$  with  $l \neq k$  is correct, thus  $P_k^- = \frac{TN}{TN+FN}$ . However, these measures do not take into account all the TP, FN, FP and TN values. For this reason, in biomedical applications the Matthew's correlation coefficient (Matthews, 1975) MC it is often used. This is defined by:

$$MC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}.$$

It ranges from  $-1$  if all the classifications are wrong to  $+1$  for perfect classification. A value equal to zero indicates that the classifications are random or the classifier always predicts only one of the two classes.

In the general case of  $K$  classes with  $K \geq 2$ , one obtains a  $K \times K$  contingency or confusion matrix  $Z = (z_{kl})$ , where  $z_{kl}$  is the number of times that units are classified to be in class  $l$  while belonging in reality to class  $k$ . Then,  $z_{k.} = \sum_l z_{kl}$  and  $z_{.l} = \sum_k z_{kl}$  represent the number of units belonging to class  $k$  and the number of units predicted to be in class  $l$ , respectively. Obviously  $n = \sum_{kl} z_{kl} = \sum_k z_{k.} = \sum_l z_{.l}$ .

One standard criterium to evaluate a classification rule is to compute the percentage of all correct predictions,

$$Q_t = 100 \frac{\sum z_{kk}}{n}, \quad (7)$$

the percentage of units correctly predicted to belong to class  $k$  relative to the total number of units in class  $k$  (sensitivity for class  $k$ ),

$$Q_k^{se} = 100 \frac{z_{kk}}{z_{k.}}, \quad (8)$$

the percentage of units correctly predicted to belong to any class  $l$  with  $l \neq k$  relative to the total number of units in any class  $l$  with  $l \neq k$  (specificity of class  $k$ ),

$$Q_k^{sp} = 100 \frac{\sum_{l \neq k} z_{l.} - \sum_{l \neq k} z_{lk}}{n - z_{k.}}, \quad (9)$$

and the percentage of units correctly classified to be in class  $k$  with respect to the total number of units classified in class  $k$  (positive predictive value for class  $k$ ),

$$P_k^+ = 100 \frac{z_{kk}}{z_{k.}}. \quad (10)$$

However, we also consider a generalization of the Matthew's correlation coefficient, the so called generalized squared correlation  $GC^2$  (Baldi et al., 2000), which is defined by

$$GC^2 = \frac{\sum_{k,l} (z_{kl} - e_{kl})^2 / e_{kl}}{n(K-1)}, \quad (11)$$

where  $e_{kl} = \frac{z_k z_l}{n}$ . This coefficient ranges between 0 and 1, and may often provide a much more balanced evaluation of the prediction than, for instance, the above percentages. A value equal to zero indicates that there is at least one class in which no units are classified.

Another interesting coefficient is the *Kappa* statistic, which measures the agreement of classification to the true class (Cohen, 1960; Landis and Koch, 1977). It can be calculated by:

$$Kappa = \frac{\frac{TP+TN}{n} - \frac{(TN+FP) \cdot (TN+FN) + (FN+TP) \cdot (FP+TP)}{n^2}}{1 - \frac{(TN+FP) \cdot (TN+FN) + (FN+TP) \cdot (FP+TP)}{n^2}},$$

and the interpretation is:  $Kappa < 0$ , less than chance agreement;  $Kappa$  in  $0.01 - 0.20$ , slight agreement;  $Kappa$  in  $0.21 - 0.40$ , fair agreement;  $Kappa$  in  $0.41 - 0.60$ , moderate agreement;  $Kappa$  in  $0.61 - 0.80$ , substantial agreement; and  $Kappa$  in  $0.81 - 0.99$ , almost perfect agreement.

Finally, another measure used as a result of classification is the  $F_1$  statistic (Powers, 2011). For each class, it is calculated based on the precision  $P_k^+$  and the recall  $Q_k^{se}$  as follows:  $F_1 = 2 \cdot \frac{P_k^+ Q_k^{se}}{P_k^+ + Q_k^{se}}$ . However, note that  $F_1$  does not take the true negatives into account.

## Distance functions

The DB and WDB procedures require the previous calculation of a distance between units. In biomedical, genetic, and ecological studies different types of dissimilarities are frequently used. For this reason, **WeDiBaDis** includes several distance functions. Although these distances can be found in other packages they were included for ease their use for non-expert R users.

The package contains the usual Euclidean distance,

$$d_E(\mathbf{y}_i, \mathbf{y}_j) = \sqrt{\sum_{k=1}^m (y_{ik} - y_{jk})^2}, \quad (12)$$

the well known correlation distance, where  $r$  is the Pearson correlation coefficient,

$$d_c(\mathbf{y}_i, \mathbf{y}_j) = \sqrt{(1 - r(\mathbf{y}_i, \mathbf{y}_j))}, \quad (13)$$

and the Mahalanobis distance (Mahalanobis, 1936) with  $S$  the variance-covariance matrix,

$$d_M(\mathbf{y}_i, \mathbf{y}_j) = \sqrt{(\mathbf{y}_i - \mathbf{y}_j)' S^{-1} (\mathbf{y}_i - \mathbf{y}_j)}. \quad (14)$$

The function named `mahalanobis()` that calculates the Mahalanobis distance already exists in the **stats** package, but it is not suitable in our context. While this function calculates the Mahalanobis distance with respect to a given center, our function is designed to calculate the Mahalanobis distance between each pair of units given a data matrix.

Next, we briefly comment on the other distances included in the package. The Bhattacharyya distance (Bhattacharyya, 1946) is a very well-known distance between populations in the genetic context. Each population is characterized by a vector  $(p_{i1}, \dots, p_{im})$  whose coordinates are the relative frequencies of the features (usually chromosomal arrangements), with

$$p_{ij} > 0, j = 1, \dots, m \quad \text{and} \quad \sum_{j=1}^m p_{ij} = 1, i = 1, \dots, n.$$

Then, the distance between two units (populations) with frequencies  $\mathbf{y}_i = (p_{i1}, \dots, p_{im})$  and  $\mathbf{y}_j = (p_{j1}, \dots, p_{jm})$  is defined by:

$$d_B(\mathbf{y}_i, \mathbf{y}_j) = \arccos \sum_{l=1}^m \sqrt{p_{il} p_{jl}}. \quad (15)$$

The Gower distance (Gower, 1971), used for mixed variables, is defined by:

$$d_G(\mathbf{y}_i, \mathbf{y}_j) = \sqrt{2(1 - s(\mathbf{y}_i, \mathbf{y}_j))}, \quad (16)$$

where  $s(\mathbf{y}_i, \mathbf{y}_j)$  is the similarity coefficient between unit  $\mathbf{y}_i = (\mathbf{x}_i, \mathbf{q}_i, \mathbf{b}_i)$  and unit  $\mathbf{y}_j = (\mathbf{x}_j, \mathbf{q}_j, \mathbf{b}_j)$ , and  $\mathbf{x}_i, \mathbf{q}_i, \mathbf{b}_i$  are the values for the  $m_1$  continuous,  $m_2$  binary and  $m_3$  qualitative features, respectively. The coefficient  $s(\mathbf{y}_i, \mathbf{y}_j)$  is calculated by:

$$s(\mathbf{y}_i, \mathbf{y}_j) = \frac{\sum_{l=1}^{m_1} \left(1 - \frac{|x_{il} - x_{jl}|}{R_l}\right) + a + \alpha}{m_1 + (m_2 - d) + m_3},$$

with  $R_l$  the range of the  $l$ th continuous variable ( $l = 1, \dots, m_1$ ); for the  $m_2$  binary variables,  $a$  and  $d$  represent the number of matches presence-presence and absence-absence, respectively; and  $\alpha$  is the number of matches between states for the  $m_3$  qualitative variables. Note that there is also the `daisy()` function in the **cluster** package, which can calculate the Gower distance for mixed variables. The difference between this function and `dGower()` in **WeDiBaDis** is that in `daisy()` the distance is calculated as  $d(\mathbf{y}_i, \mathbf{y}_j) = 1 - s(\mathbf{y}_i, \mathbf{y}_j)$  and in `dGower()` as  $d(\mathbf{y}_i, \mathbf{y}_j) = \sqrt{2(1 - s(\mathbf{y}_i, \mathbf{y}_j))}$ . Moreover, `dGower()` allows us to include missing values (such as NA) and therefore calculates distances based

on Gower's weighted similarity coefficients. The `dGower()` function improves the function `dgower()` included in the package **ICGE** (Irigoien et al., 2013b).

The Bray-Curtis distance (Bray and Curtis, 1957) is one of the most well-known ways of quantifying the difference between samples when the information is ecological abundance data collected at different sampling locations. It is defined by:

$$d_B(\mathbf{y}_i, \mathbf{y}_j) = \frac{\sum_{l=1}^m |y_{il} - y_{jl}|}{y_{i+} + y_{j+}}, \quad (17)$$

where  $y_{il}, y_{jl}$  are the abundance of specie  $l$  in samples  $i$  and  $j$ , respectively, and  $y_{i+}, y_{j+}$  are the total specie's abundance in samples  $i$  and  $j$ , respectively. This distance can be also found in the **vegan** package.

The Hellinger (Rao, 1995) and Orloci (or chord distance) (Orloci, 1967) distances are also measures recommended for quantifying differences between sampling locations when the ecological abundance of species is collected. The Hellinger distance is given by:

$$d_H(\mathbf{y}_i, \mathbf{y}_j) = \sqrt{\sum_{l=1}^m \left( \sqrt{\frac{y_{il}}{\sum_{k=1}^m y_{ik}}} - \sqrt{\frac{y_{jl}}{\sum_{k=1}^m y_{jk}}} \right)^2}, \quad (18)$$

and the Orloci distance that represents the Euclidean distance computed after scaling the site vectors to length 1 is defined by:

$$d_O(\mathbf{y}_i, \mathbf{y}_j) = \sqrt{\sum_{l=1}^m \left( \frac{y_{il}}{\sqrt{\sum_{k=1}^m y_{ik}^2}} - \frac{y_{jl}}{\sqrt{\sum_{k=1}^m y_{jk}^2}} \right)^2}. \quad (19)$$

This distance between two sites is equivalent to the length of a chord joining two points within a segment of a hypersphere of radius 1.

The Prevosti distance (Prevosti et al., 1975) is a very useful genetic distance between units representing populations. Now, we consider that genetic data is stored in a table where the rows represent the populations and the columns represent potential allelic states grouped by loci. The distance between two units at a single locus  $k$  with  $m(k)$  allelic states is:

$$d_P(\mathbf{y}_i, \mathbf{y}_j) = \frac{1}{2\nu} \sum_{k=1}^v \sum_{s=1}^{m(k)} |p_{iks} - p_{jks}|, \quad (20)$$

where  $v$  is the number of loci or chromosomes (in the case of chromosomal polymorphism) considered and  $p_{iks}, p_{jks}$  are the sample relative frequencies of the allele or chromosomal arrangement  $s$  in the locus or chromosome  $k$ , in the  $i$ th and  $j$ th population, respectively. With presence/absence data coded by 1 and 0, respectively, the term  $\frac{1}{2\nu}$  is omitted.

As we explain in the next section, **WeDiBaDis** allows the user to introduce alternative distances by means of a distance matrix. Therefore, the user can work with any distance matrix that is considered appropriate for their data set and analysis. For this reason, no more distances were included in our package.

## Using the package

We have developed the **WeDiBaDis** package to implement both the DB-discriminant and the new WDB-discriminant. It can be used with different distance functions and NA values are allowed. When an unit has a NA value in some features, those features are excluded in the computation of the distances for that unit and the computation is scaled up to the number  $m$  of features involved in the data set. Package **WeDiBaDis** requires a version 3.3.1 or a greater of R.

The principal function is `WDBdisc` with arguments:

```
WDBdisc(data, datatype, classcol, new.ind, distance, type, method)
```

Where:

- data a data matrix or a distance matrix. If the Prevosti distance will be used, data must

be a named matrix where the name of the loci and allele must be separated by a dot (Loci-Name.AlleleName)

- `datatype` if the data is a data matrix, `datatype="m"`; if the data is a distance matrix `datatype="d"`
- `classcol` a number indicating which column in the data contains the class variable. By default the class variable is in the first column
- `new.ind` is only required if there are new unlabeled units to be classified; if `datatype="m"` it is a matrix containing the feature values for the new units to be classified; if `datatype="d"` it is a matrix containing the distances between the new units to be classified and the units in the classes
- `distance` the distance measure to be used. This must be either "euclidean" (default option), "correlation", "Bhattacharyya", "Gower", "Mahalanobis", "BrayCurtis", "Orloci", "Hellinger", or "Prevosti".
- `type` is only required if `distance = "Gower"`. The value for `type` is a list (e.g., `type = list(cuant,nom,bin)`) indicating the position of the columns for continuous (`cuant`), nominal (`nom`) and binary (`bin`) features, respectively
- `method` the discriminant method to be used. This must be either "DB" or "WDB" for the DB-discriminant and WDB-discriminant, respectively. The default method is WDB

The function returns an object with associated plot and `summary` methods offering:

- The classification table obtained with the leave-one-out cross-validation
- The total well classification rate in percentage ( $Q_t$ )
- The generalized squared correlation ( $GC^2$ )
- The sensitivity, specificity, and positive predictive values for each class ( $Q_k^{se}$ ,  $Q_k^{sp}$ , and  $P_k^+$ , respectively)
- The  $Kappa$  and  $F_1$  statistics
- The assigned class for new unlabeled units to be classified
- A barplot for the classification table
- A barplot for the sensitivity, specificity, and positive predictive values for each class

Moreover, given a data set, the distances commented on in Section "Distance funtions" can be obtained through the functions: `dcor` (correlation distance); `dMahal` (Mahalanobis distance); `dBhatta` (Bhattacharyya distance); `dGower` (Gower distance); `dBrayCurtis` (Bray and Curtis distance); `dHellinger` (Hellinger distance); `dOrloci` (Orloci distance), and `dPrevosti` (Prevosti distance).

### Example 1: Ecological data

We consider the data from [Fielding \(2007\)](#), which relate to the core area (the region close to the nest) of the golden eagle *Aquila chrysaetos* in three regions of Western Scotland. The data consist of eight habitat variables: POST (mature planted conifer forest in which the tree canopy has closed); PRE (pre-canopy closure planted conifer forest); BOG (flat waterlogged land); CALL (Calluna (heather) heath land); WET (wet heath, mainly purple moor grass); STEEP (steeply sloping land); LT200 (land below 200 m), and L4-600 (land between 200 and 400 m). The values are the numbers of four-hectare grid cells covered by the habitat, whose values are the amounts of each habitat variable, measured as the number of four hectare blocks within a region defined as a "core area." In order to evaluate if the habitat variables allow to discriminate between these three regions, for example, a WDB-discriminant using the Euclidean distance using the following instructions may be performed:

```
library(WeDiBaDis)
out<-WDBdisc(data=datafile, datatype="m", classcol=1)
```

The `summary` method shows, as usual, the more complete information:

```
summary(out)
```

Discriminant method: WDB  
Leave-one-out confusion matrix:

		Predicted		
Real	1	2	3	
1	7	0	0	
2	0	14	2	
3	2	0	15	

```
Total correct prediction: 90%
Generalized squared correlation: 0.7361
Cohen's Kappa coefficient: 0.84375
Sensitivity for each class:
  1      2      3
100.00 87.50 88.24
Predictive value for each class:
  1      2      3
77.78 100.00 88.24
Specificity for each class:
  1      2      3
87.88 91.67 91.30
F1-score for each class:
  1      2      3
87.50 93.33 88.24
----- ----- ----- ----- -----
```

No predicted individuals

As we can observe, perfect classification is obtained for samples from region 1. For regions 2 and 3, only two samples were not correctly classified.

If we want to obtain the barplot for the classification table (see Figure 1), we use the command

```
plot(out)
```

These commands generate the sensitivity, specificity and positive predicted values barplot (see Figure 2):

```
outplot <- summary(out, show=FALSE)
plot(outplot)
```

Finally to perform a DB discriminant using a different distance than the Euclidean, the following commands are used:

```
library(WeDiBaDis)
out<-WDBdisc(data=datafile, datatype="m", distance="name of the distance",
               method="DB", classcol=1)
summary(out)
plot(out)
outplot <- summary(out, show=FALSE)
plot(outplot)
```

### Example 2: population genetics data

The chromosomal polymorphism for inversions is very useful to characterize the natural populations of *Drosophila subobscura*. Furthermore, lethal genes located in chromosomal inversions allow the understanding of important evolutionary events. We consider the data from a study of 40 samples of this polymorphism for the O chromosome of this species (Solé et al., 2000; Balanyà et al., 2004; Mestres et al., 2009). Four groups can be considered: NLE with 16 no lethal European samples, LE with 4 lethal European samples, NLA with 14 no lethal American samples and LA with 6 lethal American samples. In this example, two samples one of the group NLA and one of the group NLE were randomly selected, and considered as new unlabeled units to be classified. The Bhattacharyya distances between all pairs of units were calculated. Therefore, the input for the WDBdisc function is an  $n \times (n + 1)$  matrix  $\text{dat} = (l_i, d_B(y_i, y_1), \dots, d_B(y_i, y_n))_{i=1,\dots,n}$  where the first column contains the class label and the following columns the distance matrix. Furthermore,  $xnew$  is a two-row matrix where each row contains the distances between the new unlabeled units to be classified and the units in the four classes. In this situation, the commands to call the WDB procedure to classify the  $xnew$  units and to obtain the available graphics in the package, are:

```
library(WeDiBaDis)
out<-WDBdisc(data=dat, datatype="d", classcol=1, new.ind=xnew)
plot(out)
outplot <- summary(out, show=FALSE)
plot(outplot)
```

The summary method shows the following information. We can see that the  $xnew$  units were correctly classified:

```

summary(out)

Discriminant method: WDB
Leave-one-out confusion matrix:
  Predicted
  Real LA LE NLA NLE
    LA  6  0  0  0
    LE  0  3  0  1
    NLA 0  0 13  0
    NLE 0  3  0 12
Total correct prediction: 89.47%
Generalized squared correlation: 0.7442
Cohen's Kappa coefficient: 0.8509804
Sensitivity for each class:
  LA      LE      NLA      NLE
100.00 75.00 100.00 80.00
Predictive value for each class:
  LA      LE      NLA      NLE
100.00 50.00 100.00 92.31
Specificity for each class:
  LA      LE      NLA      NLE
87.50 91.18 84.00 95.65
F1-score for each class:
  LA      LE      NLA      NLE
100.00 60.00 100.00 85.71
----- ----- ----- -----
Prediction for new individuals:
Pred. class
1 "NLE"
2 "NLA"

```

Now, the two unlabeled new units were correctly classified. The barplots are in Figure 3 and Figure 4, respectively.

## Data files

The package contains some examples of data files, each with a corresponding explanation. The data sets are `corearea`, containing the data for the example presented in the subsection Example 1: Ecological data; abundances, which is a simulated data set for abundance data matrix; and `microsatt`, a data set containing allele frequencies for 18 cattle breeds (bull or zebu), of French and African descent, typed on 9 microsatellites.

## Computing time

To illustrate the time consumed by the WDB procedure, which requires more computation than DB, we performed the following simulation with artificial data. We generated multinormal samples containing 50, 100, 200, 300,...,900, 1000, 2000, and 3000 units, respectively. Then, for each sample size we created sets containing respectively 50, 100, 500, 1000, 1500, 2000, 2500, ..., 4500, and 5000 features. For each combination of sample sizes and features, we considered 2, 3, 4, and 10 classes. All the computations presented in this paper have been performed on a personal computer with Intel(R) Core(TM) i5-2450M and 6 GB of memory using a single 2.50GHz CPU processor. The results of the simulation for two classes are displayed in Figure 5, where the elapsed time (the actual elapsed time since the process started) is reported in seconds. We can observe that the runtime is mainly affected by the number of units (Figure 4, top), but affected very little by the number of variables (Figure 4, bottom). This is expected, as the procedure is based on distances and therefore the dimension of the distance matrix (number of units) determines the runtime required. The number of classes also affects the runtime, although its variation with increasing the number of classes is very slight. For example, with 300 units and 4000 variables, the elapsed time for 2, 3, 4, and 10 classes are 3.38, 3.40, 3.62, and 4.82 seconds, respectively.

## DB and WDB comparison using cancer data sets

In order to compare the performance of DB and WDB procedures, thirty-eight available cancer data sets were considered in our analysis (Table 1). These are available at <http://bioinformatics.rutgers.edu/Static/Supplements/CompCancer/datasets.htm> and Lê Cao et al. (2010). As we can observe in Table 1, three of them include clinical features and some of the data sets have unbalanced classes. We performed the evaluation for DB and WDB classifiers using the leave-one-out procedure. We present the total misclassification rate  $MQ_t = 100 - Q_t$  and the generalized squared correlation coefficient  $GC^2$  (Table 2). For simplicity, the sensitivity  $Q_k i^{se}$ , the specificity  $Q_k^{sp}$ , the positive predictive value  $P_k^+$  for each class, the *Kappa* and  $F_1$  statistics are not presented. For the microarray data sets with only continuous features we used the Euclidean distance, and for those including clinical and genetic data, we considered the Gower distance (Gower, 1971). As we can observe in Table 2, considering only  $MQ_t$ , the total misclassification percentage rate, WDB was the best classifier in 18 data sets and it shared this quality in 11 data sets with DB (Wilcoxon signed rank test; one side p-value = 0.0265). Using the generalized squared correlation  $GC^2$  coefficient (Table 2), WDB was the best rule in 16 data sets and it shared this quality in 11 data sets with DB (Wilcoxon signed rank test; one side p-value = 0.0378). Note that for data sets 30 and 38 the  $GC^2$  value is 0. For example, in the Risinger-2003 case, all units of the second class (class with 3 units) were badly classified with DB and WDB methods. However, while with the DB method, 4 units belonging to other classes were badly classified in class 2, with the WDB method none of the units of other classes were badly classified in class 2, and for this reason the  $GC^2$  is equal to 0. With the Yeoh-2002-v2 data set something similar happened. For all these results, WDB seems to obtain in general the best results and to be a slightly better in the case where classes are unbalanced with respect to their sizes.

## Conclusions

The package **WeDiBaDis**, available at <https://github.com/ItziarI/WeDiBaDis>, is an implementation of two discriminant analysis procedures in an R environment. The classifiers are the Distance-Based (DB) and the new proposed procedure Weighted-Distance-Based (WDB). They are useful to solve the classification problem for high-dimensional data sets with mixed features or when the input information is a distance matrix. This software provides functions to compute both discriminant procedures and to assess the performance of the classification rules it offers: the leave-one-out classification table; the general correlation coefficient; the sensitivity, specificity, and positive predictive value for each class; the *Kappa* and the  $F_1$  statistics. The package also presents these results in a graphical form (barplots for the classification table and, for sensitivity, specificity and positive predictive values, respectively). Furthermore, it allows the classification for new unlabeled units. **WeDiBaDis** provides a user-friendly environment, which can be of great utility in biology, ecology, biomedical, and, in general, any applied study involving discrimination between groups and classification of new unlabeled units. In addition, it can be very useful in multivariate methods courses aimed at biologists, medical researchers, psychologists, etc.

## Acknowledgements

This research was partially supported: II by the Spanish 'Ministerio de Economía y Competitividad' (TIN2015-64395-R) and by the Basque Government Research Team Grant (IT313-10) SAIOTEK Project SA-2013/00397 and by the University of the Basque Country UPV/EHU (Grant UFI11/45 (BAILab). FM by the Spanish 'Ministerio de Economía y Competitividad' (CTM2013-48163) and by Grant 2014 SGR 336 from the Departament d'Economia i Coneixement de la Generalitat de Catalunya. CA by the Spanish 'Ministerio de Economía y Competitividad' (SAF2015-68341-R), by the Spanish 'Ministerio de Economía y Competitividad' (TIN2015-64395-R) and by Grant 2014 SGR 464 (GRBIO) from the Departament d'Economia i Coneixement de la Generalitat de Catalunya.

## Bibliography

- M. J. Anderson and J. Robinson. Generalized discriminant analysis based on distances. *Australian and New Zealand Journal of Statistics*, 45:301–318, 2003. [p436]
- I. Balanyà, E. Solé, J. M. Oller, D. Sperlich, and L. Serra. Long-term changes in chromosomal inversion polymorphism of *D. subobscura*: II. European populations. *Journal of Zoological Systematics and Evolutionary Research*, 42:191–201, 2004. [p442]
- P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16:412–424, 2000. [p438]
- A. Bhattacharyya. On a measure of divergence of two multinomial populations. *Sankhya*, 7:401–406, 1946. [p439]
- R. Blagus and L. Lusa. Improved shrunken centroid classifiers for high-dimensional class-imbalanced data. *BMC Bioinformatics*, 14:64, 2013. [p436]
- G. Bouin. Computer program review: Ginkgo, a multivariate analysis package. *Journal of Vegetation Science*, 16:355–359, 2005. [p436]
- A. L. Boulesteix, C. Porzelius, and M. Daumer. Microarray-based classification and clinical predictors: On combined classifiers and additional predictive value. *Bioinformatics*, 24:1698–1706, 2008. [p435]
- J. R. Bray and J. T. Curtis. An ordination of upland forest communities of southern Wisconsin. *Ecological Monographs*, 27:325–349, 1957. [p440]
- L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. [p435]
- J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20: 37–46, 1960. [p438]
- T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967. [p436]
- C. M. Cuadras. *Statistical Data Analysis and Inference*, chapter Distance Analysis. In: Discrimination and Classification Using both Continuous and Categorical Variables, pages 459–473. Elsevier Science Publishers BV, Amsterdam, 1989. [p436]
- C. M. Cuadras. Some examples of distance based discrimination. *Biometrical Letters*, 29:3–20, 1992. [p436]
- C. M. Cuadras, J. Fortiana, and F. Oliva. The proximity of an individual to a population with applications in discriminant analysis. *Journal of Classification*, 14:117–136, 1997. [p436, 437]
- M. De Caceres, F. Oliva, and X. Font. GINKGO, a multivariate analysis program oriented towards distance-based classifications. In *International Conference on Correspondence Analysis and Related Methods (CARME' 03)*, 2003. [p436]
- R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience Publication. John Wiley and Sons, New York, 2000. [p436]
- S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of American Statistical Association*, 97:77–87, 2002. [p435]
- A. H. Fielding. *Cluster and Classification Techniques for the Biosciences*. Cambridge University Press, 2007. [p441]
- R. A. Fisher. The use of multiple measurements in taxonomic problems. *The Annals of Eugenics*, 7: 179–188, 1936. [p435]
- A. K. Ghosh and P. Chaudhuri. On data depth and distribution-free discriminant analysis using separating surfaces. *Bernoulli*, 11:1–27, 2005. [p436]
- T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, and C. D. Bloomfield. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999. [p435]
- J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27:857–871, 1971. [p439, 444]

- T. Hastie, R. Tibshirani, D. Botstein, and P. Brown. Supervised harvesting of expression trees. *Genome Biology*, 2:1–12, 2001. [p435]
- D. Hlubinka and O. Vencsek. Depth-based classification for distributions with nonconvex support. *Journal of Probability and Statistics*, 28:1–7, 2013. [p436]
- I. Irigoien, F. Mestres, and C. Arenas. The depth problem: Identifying the most representative units in a data group. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10:161–172, 2013a. [p436, 437]
- I. Irigoien, B. Sierra, and C. Arenas. ICGE: An R package for detecting relevant clusters and atypical units in gene expression. *BMC Bioinformatics*, 13:30–41, 2013b. [p440]
- J. Jin and H. Cui. Discriminant analysis based on statistical depth. *Journal of Systems Science and Complexity*, 23:362–371, 2010. [p436]
- R. Jornstein. Clustering and classification based on  $L_1$  data depth. *Journal of Multivariate Analysis*, 90: 67–89, 2004. [p436]
- M. Kent. *Vegetation Description and Data Analysis: A Practical Approach*. Wiley-Blackwey, 2011. [p436]
- W. I. Krzanowski, P. Jonathan, W. V. McCarthy, and M. R. Thomas. Discriminant analysis with singular covariance matrices: Methods and applications to spectroscopic data. *Applied Statistics*, 44:101–115, 1995. [p435]
- J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33:159–174, 1977. [p438]
- K. A. Lê Cao, E. Meugnier, and G. J. McLachlan. Integrative mixture of experts to combine clinical factors and gene markers. *Bioinformatics*, 26:1192–1198, 2010. [p444]
- P. Legendre and L. Legendre. *Numerical Ecology*. Elsevier, Amsterdam, 1998. [p436]
- R. Y. Liu. On a notion of data depth based on random simplices. *Annals of Statistics*, 18:405–414, 1990. [p436]
- P. V. Mahalanobis. On the generalized distance in statistics. *Procedures of the Natural Institute of Science of India*, 2:49–55, 1936. [p439]
- B. W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica Biophysica Acta*, 405:442–451, 1975. [p438]
- F. Mestres, J. Balanyà, M. Pascual, C. Arenas, G. W. Gilchrist, R. B. Huey, and L. Serra. Evolution of Chilean colonizing populations of *D. subobscura*: lethal genes and chromosomal arrangements. *Genetica*, 136:37–48, 2009. [p442]
- L. Orloci. An agglomerative method for classification of plant communities. *Journal of Ecology*, 55: 193–205, 1967. [p436, 440]
- D. M. W. Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlaction. *Journal of Machine Learning Technologies*, 2:37–63, 2011. [p439]
- A. Prevosti, J. Ocaña, and G. Alonso. Distances between populations of *D. subobscura*, based on chromosome arrangement frequencies. *Theoretical and Applied Genetics*, 45:231–241, 1975. [p440]
- C. R. Rao. A review of canonical coordinates and an alternative to correspondence analysis using Hellinger distance. *Qüestiió*, 19:23–63, 1995. [p440]
- R. Serfling. *Statistic and Data Analysis Based on  $L_1$ -Norm and Related Methods*, chapter A Depth Function and a Scale Curve Based on Spatial Quantiles, pages 25–38. Birkhäuser, Boston, 2002. [p436]
- E. Solé, F. Mestres, J. Balanyà, C. Arenas, and L. Serra. Colonization of America by *D. subobscura*: Spatial and temporal lethal-gene allelism. *Hereditas*, 133:65–72, 2000. [p442]
- R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences of the United States of America*, 99:6567–6572, 2002. [p435]
- Y. Vardi and C. Zhang. The multivariate  $L_1$ -median and associated data depth. *Proceedings of the National Academy of Sciences of the United States of America*, 97:1423–1426, 2000. [p436]

- X. H. Zhou, N. A. Obuchowski, and D. K. McClish. *Statistical Methods in Diagnostic Medicine*. Wiley Series in Probability and Statistics. John Wiley and Sons, New Jersey, 2002. [p<sup>438</sup>]
- S. Zuo and R. Serfling. General notions of statistical depth function. *Annals of Statistics*, 28:461–482, 2000. [p<sup>436</sup>, 437]

*Itziar Irigoien*  
Department of Computation Science and Artificial Intelligence  
University of the Basque Country UPV/EHU  
Donostia, Spain  
[itziar.irigoien@ehu.eus](mailto:itziar.irigoien@ehu.eus)

*Francesc Mestres*  
Department of Genetics, Microbiology and Statistics. Genetics Section  
University of Barcelona  
Barcelona, Spain  
[fmestres@ub.edu](mailto:fmestres@ub.edu)

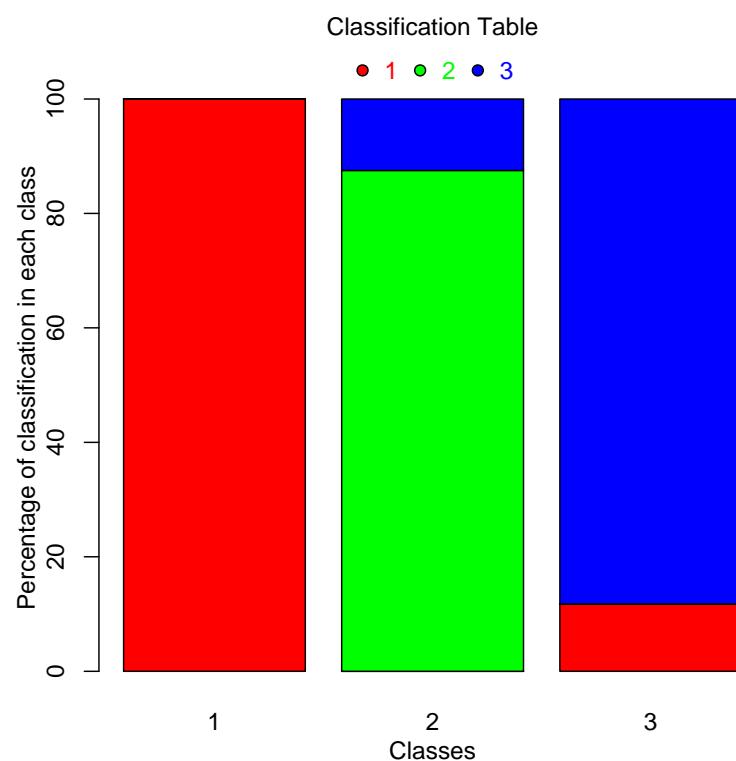
*Concepcion Arenas*  
Department of Genetics, Microbiology and Statistics. Statistics Section  
University of Barcelona  
Barcelona, Spain  
[carenas@ub.edu](mailto:carenas@ub.edu)

ID	Data set	K	n	$n_i$	p	cuant	quali
1	Alizadeh-2000-v1	2	42	21(50%), 21(50%)	1095	1095	
2	Alizadeh-2000-v2	3	62	42(67.74%), 9(14.52%), 11(17.74%)	2093	2093	
3	Armstrong-2002-v1	2	72	24(33.33%), 48(66.67%)	1081	1081	
4	Armstrong-2002-v2	3	72	24(33.33%), 20(27.78%), 28(38.89%)	2194	2194	
5	Bhattacharjee-2001	5	203	139(68.47%), 17(8.37%), 6(2.96%), 21(10.34%), 20(9.85%)	1543	1543	
6	Bittner-2000-V1	2	38	19(50%), 19(50%)	2201	2201	
7	Bittner-2000-V2	3	38	19(50%), 12(31.58%), 7(18.42%)	2201	2201	
8	Breast	2	256	75(29.30%), 181(70.70%)	5545	5537	8
9	Bredel-2005	3	50	31(62%), 14(28%), 5(10%)	1739	1739	
10	Chen-2002	2	179	104(58.10%), 75(41.90%)	85	85	
11	Chowdary-2006	2	104	62(59.62%), 42(38.89%)	182	182	
12	CNS	2	60	21(35%), 39(65%)	7134	7128	6
13	Dyrskjot-2003	3	40	9(22.5%), 20(50%), 11(27.5%)	1203	1203	
14	Garber-2001	4	66	17(25.76%), 40(60.61%), 4(6.06%), 5(7.58%)	4553	4553	
15	Golub-1999-v1	2	72	47(65.28%), 25(34.72%)	1877	1877	
16	Golub-1999-v2	3	72	38(52.78%), 9(12.5%), 25(34.72%)	1877	1877	
17	Gordon-2002	2	181	31(17.13%), 150(82.87%)	1626	1626	
18	Khan-2001	4	83	29(34.94%), 11(13.25%), 18(21.69%), 25(30.12%)	1069	1069	
19	Laiho-2007	2	37	8(21.62%), 29(78.38%)	2202	2202	
20	Lapointe-2004-v1	3	69	11(15.94%), 39(56.52%), 19(27.54%)	1625	1625	
21	Lapointe-2004-v2	4	110	11(10%), 39(35.45%), 19(17.27%), 41(37.27%)	2496	2496	
22	Liang-2005	3	37	28(75.67%), 6(16.22%), 3(8.11%)	1411	1411	
23	Nutt-2003-v1	4	50	14(50%), 7(14%), 14(28%), 15(30%)	1377	1377	
24	Nutt-2003-v2	2	28	14(50%), 14(50%)	1070	1070	
25	Nutt-2003-v3	2	22	7(31.82%), 15(68.18%)	1152	1152	
26	Pomeroy-2002-v1	2	34	25(73.53%), 9(26.47%)	857	857	
27	Pomeroy-2002-v2	5	42	10(23.81%), 10(23.81%), 10(23.81%), 4(9.52%) 8(19.05%)	1379	1379	
28	Prostate	2	79	37(46.84%), 42(53.16%)	7892	7884	8
29	Ramaswamy-2001	14	190	11(5.79%), 11(5.79%), 20(10.53%), 11(5.79%), 30(15.79%), 11(5.79%), 22(11.28%), 11(5.79%), 10(5.26%), 11(5.79%), 11(5.79%), 10(5.26%), 11(5.79%), 10(5.26%)	1363	1363	
30	Risinger-2003	4	42	13(30.95%), 3(7.14%), 19(45.24%), 7(16.67%)	1771	1771	
31	Shipp-2002-v1	2	77	58(75.32%), 19(24.67%)	798	798	
32	Singh-2002	2	102	50(49.02%), 52(50.98%)	339	339	
33	Su-2001	10	174	8(4.60%), 26(14.94%), 23(13.22%), 12(6.90%), 11(6.32%), 7(4.02%), 28(16.09%), 27(15.52%), 6(3.45%), 26(14.94%)	1571	1571	
34	Tomlins-2006-v1	5	104	27(25.96%), 20(19.23%), 32(30.77%), 13(12.5%), 12(11.54%)	2315	2315	
35	Tomlins-2006-v2	4	92	27(26.35%), 20(21.74%), 32(34.78%), 13(14.13%)	1288	1288	
36	West-2001	2	49	25(51.02%), 24 (48.98%)	1198	1198	
37	Yeoh-2002-v1	2	248	43(17.34%), 205(82.66%)	2526	2526	
38	Yeoh-2002-v2	6	248	15(6.05%), 27(10.89%), 64(25.81%), 20(8.06%), 43(17.34%), 79(31.85%)	2526	2526	

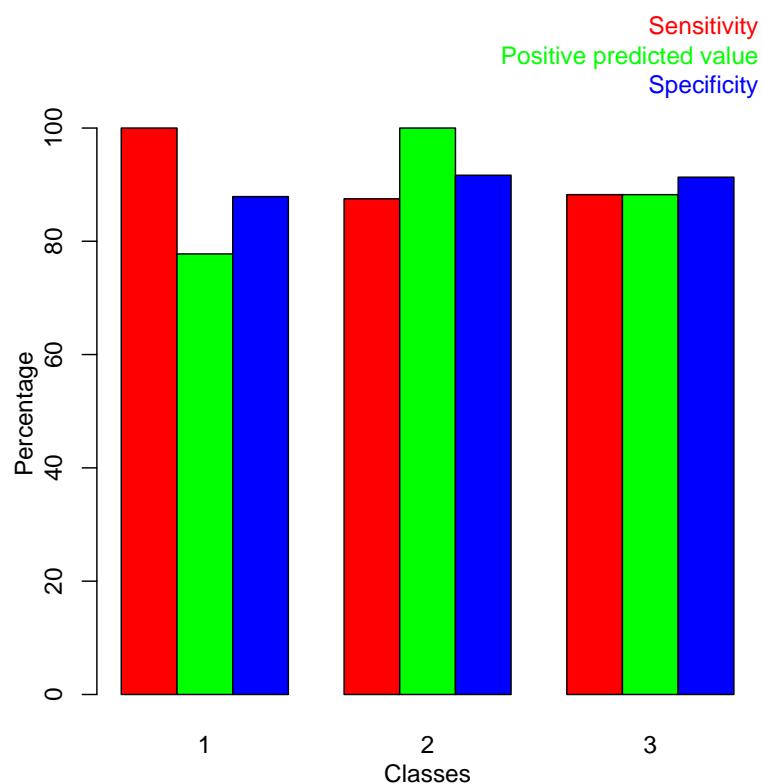
**Table 1:** Cancer data sets (ID = identification number). They present different number of classes (K), number of samples (n), number of samples in each class ( $n_i$ ), number of features (p), number of continuous features (cuant) and number of qualitative features (quali). The percentage corresponding to the number of samples belonging to each class is in brackets in column five.

ID	$100 - Q_t$ DB	$100 - Q_t$ WDB	$GC^2$ DB	$GC^2$ WDB
1	<b>7.14</b>	<b>7.14</b>	<b>0.74</b>	<b>0.74</b>
2	1.61	<b>0.00</b>	0.94	<b>1.00</b>
3	8.33	<b>5.56</b>	0.684	<b>0.77</b>
4	<b>4.17</b>	<b>4.17</b>	<b>0.88</b>	<b>0.88</b>
5	19.21	<b>15.27</b>	0.49	<b>0.56</b>
6	<b>13.16</b>	<b>13.16</b>	<b>0.56</b>	<b>0.56</b>
7	<b>36.84</b>	<b>36.84</b>	0.25	<b>0.25</b>
8	32.81	<b>30.47</b>	0.11	<b>0.13</b>
9	<b>18.00</b>	<b>18.00</b>	<b>0.34</b>	<b>0.34</b>
10	11.17	<b>8.94</b>	0.61	<b>0.67</b>
11	18.27	<b>9.62</b>	0.42	<b>0.64</b>
12	41.67	<b>38.33</b>	<b>0.01</b>	<b>0.01</b>
13	15.00	<b>12.50</b>	0.58	<b>0.65</b>
14	<b>21.21</b>	28.79	<b>0.38</b>	0.19
15	6.94	<b>4.17</b>	0.72	<b>0.82</b>
16	<b>6.94</b>	<b>6.94</b>	<b>0.81</b>	<b>0.81</b>
17	<b>12.71</b>	13.26	<b>0.47</b>	0.42
18	<b>1.20</b>	<b>1.20</b>	0.97	<b>0.97</b>
19	<b>21.62</b>	<b>21.62</b>	<b>0.23</b>	<b>0.23</b>
20	31.88	<b>30.43</b>	0.23	<b>0.26</b>
21	<b>30.91</b>	<b>30.91</b>	<b>0.34</b>	<b>0.34</b>
22	13.51	<b>10.81</b>	0.72	<b>0.76</b>
23	<b>32.00</b>	34.00	<b>0.40</b>	0.33
24	17.86	<b>10.71</b>	0.43	<b>0.65</b>
25	<b>4.55</b>	9.09	<b>0.80</b>	0.67
26	29.41	<b>20.59</b>	0.12	<b>0.16</b>
27	<b>16.67</b>	21.43	<b>0.65</b>	0.63
28	<b>34.18</b>	<b>34.18</b>	<b>0.10</b>	<b>0.10</b>
29	36.84	<b>29.47</b>	0.44	<b>0.53</b>
30	28.57	<b>26.19</b>	<b>0.36</b>	0.00
31	29.87	<b>12.99</b>	0.24	<b>0.48</b>
32	<b>30.39</b>	<b>30.39</b>	<b>0.18</b>	0.16
33	20.11	<b>16.67</b>	0.63	<b>0.70</b>
34	<b>17.31</b>	21.15	<b>0.66</b>	0.58
35	<b>23.91</b>	26.09	<b>0.46</b>	0.41
36	20.41	<b>14.29</b>	0.35	<b>0.52</b>
37	<b>1.61</b>	2.02	<b>0.89</b>	0.87
38	<b>21.77</b>	24.60	<b>0.57</b>	0.00

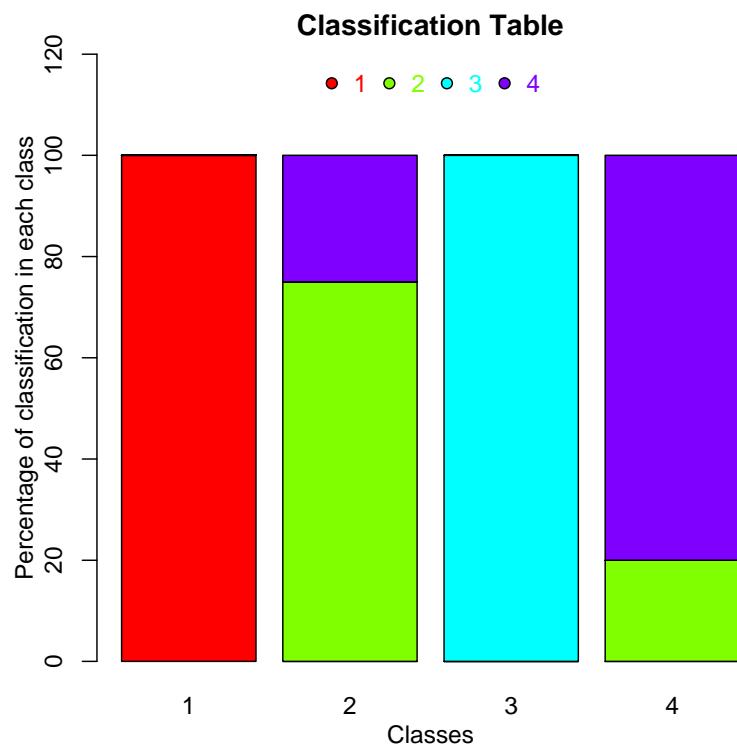
**Table 2:** In the first column identification number for cancer data sets. In the second and third columns, total leave-one-out misclassification rate  $100 - Q_t$  (in percentage) for classifiers  $DB$  and  $WDB$ , respectively. In bold the smallest misclassification rate. In the forth and fifth columns, generalized squared correlation  $GC^2$  coefficient for classifiers  $DB$  and  $WDB$ , respectively. In bold the greater  $GC^2$  value.



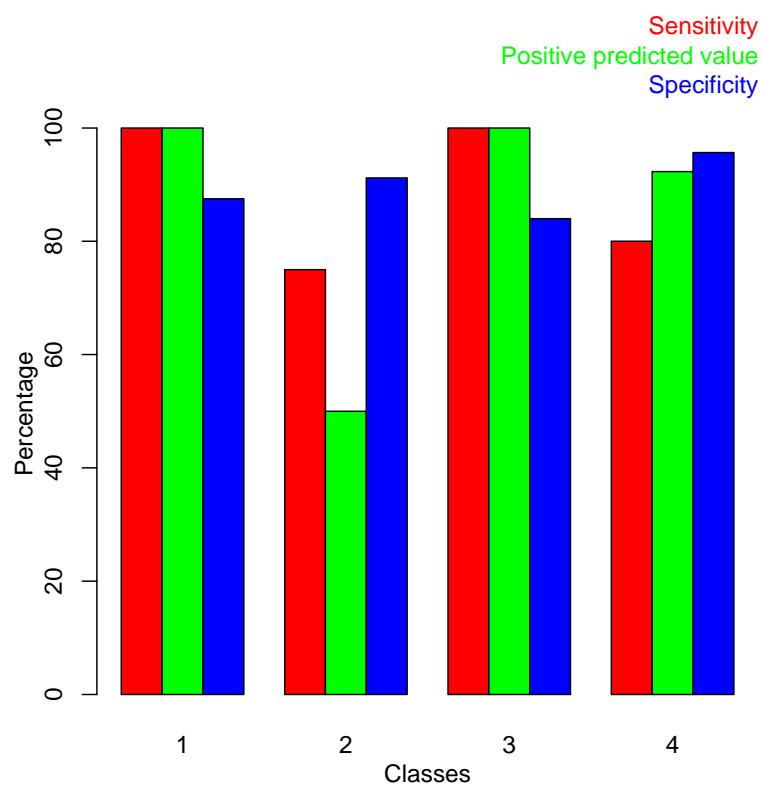
**Figure 1:** Plot of leave-one-out classification table for ecological data in Example 1.



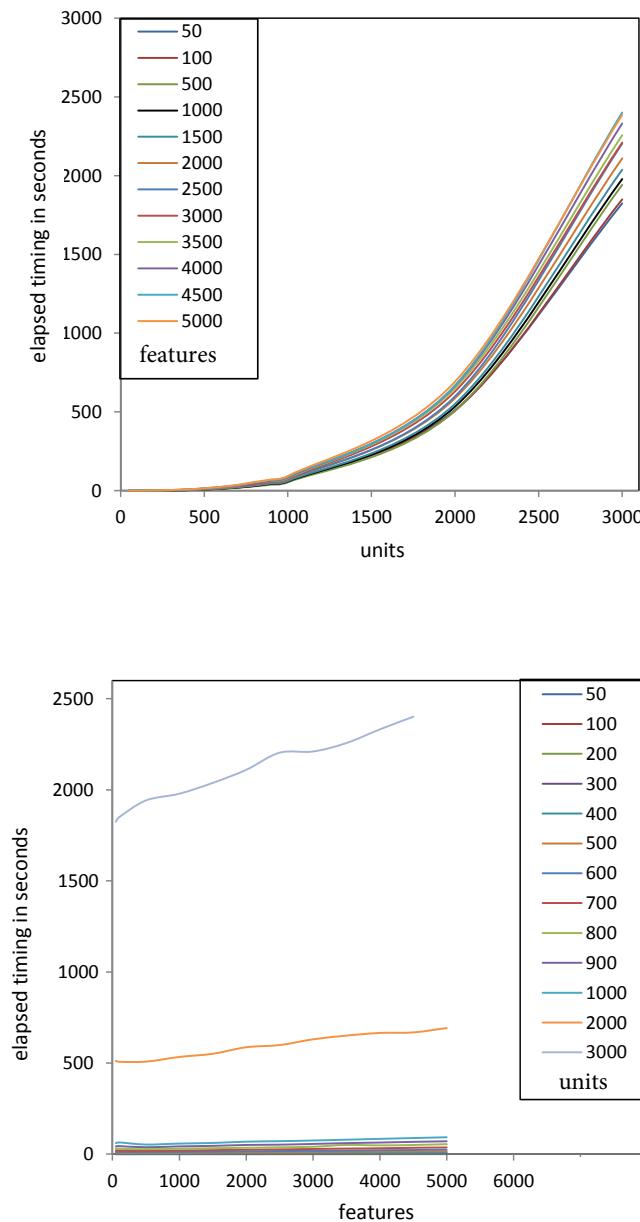
**Figure 2:** Plot of the sensitivity, specificity, and positive predicted value for each class for ecological data in Example 1.



**Figure 3:** Plot of leave-one-out classification table for population genetics data in Example 2.



**Figure 4:** Plot of the sensitivity, specificity, and positive predicted value for each class for population genetics data in Example 2.



**Figure 5:** Artificial data sets with two classes. Top: Elapsed timing in seconds (y axes) for WDB procedure with respect to the number of units (x axes). Each line (colours in the legend) corresponds to the set with identical number of features. Bottom: Elapsed timing in seconds (y axes) for WDB procedure with respect to the number of features (x axes). Each line (colours in the legend) corresponds to the set with identical number of units.

# Distance Measures for Time Series in R: the **TSdist** Package

by Usue Mori, Alexander Mendiburu and Jose A. Lozano

**Abstract** The definition of a distance measure between time series is crucial for many time series data mining tasks, such as clustering and classification. For this reason, a vast portfolio of time series distance measures has been published in the past few years. In this paper, the **TSdist** package is presented, a complete tool which provides a unified framework to calculate the largest variety of time series dissimilarity measures available in R at the moment, to the best of our knowledge. The package implements some popular distance measures which were not previously available in R, and moreover, it also provides wrappers for measures already included in other R packages. Additionally, the application of these distance measures to clustering and classification tasks is also supported in **TSdist**, directly enabling the evaluation and comparison of their performance within these two frameworks.

## Introduction

In recent years, the increase in data collecting technologies has triggered the creation of time series databases, where each instance consists of an entire time series. The main features of this type of data are its high dimensionality, dynamism, auto-correlation and noisy nature, all which complicate the study and pattern extraction to a large extent. However, in the past few years, tasks such as regression, classification, clustering or segmentation have been extended and modified successfully for time series databases (Fu, 2011; Bagnall et al., 2016). In many cases, these tasks require the definition of a distance measure, which will indicate the level of similarity between time series. Because of this, understanding suitable measures for this specific type of data has become a crucial area of study.

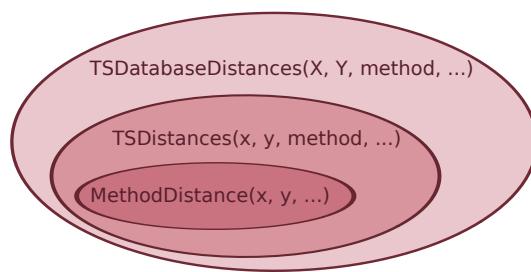
R is a popular programming language and a free software environment for statistical computing, data analysis and graphics (R Core Team, 2014), which can be extended by means of *packages*, contributed by the users themselves. A few of these R packages, such as **dtw** (Giorgino, 2009), **pdc** (Brandmaier, 2015), **proxy** (Meyer and Buchta, 2015), **longitudinalData** (Genolini, 2014) and **TSclust** (Montero and Vilar, 2014) provide implementations of some time series distance measures. However, many of the most popular distances reviewed by Esling and Agon (2012); Wang et al. (2012) and Bagnall et al. (2016) are not available in these R packages.

In this paper, the **TSdist** package (Mori et al., 2015) for the R statistical software is presented. In addition to providing wrapper functions to all the distance measures implemented in the previously mentioned packages, **TSdist** implements another 9 distance measures designed for univariate numerical time series. These distance measures have been selected based on their prevalence, and because they are mentioned in recent reviews on the topic (Liao, 2005; Esling and Agon, 2012; Wang et al., 2012). In this manner, and to the best of our knowledge, this package provides the most up-to-date coverage of the published time series distance measures in R.

## Design and implementation of the package

As can be seen in Figure 1, the core of the **TSdist** package consists of three types of functions. To begin with, in the lowest level, the functions of the type **MethodDistance** conform the basis of the package, and can be used to calculate distances between pairs of numerical and univariate vectors. Of course, **Method** must be substituted by the name of a specific distance measure. Most of them are implemented exclusively in R language but, the internal routines of a few of them are implemented in C language, for reasons of computational efficiency.

In the next level, the wrapper function called **TSDistances** enables the calculation of distance measures between univariate time series objects of type **ts**, **zoo** and **xts**, the latter two defined in their respective packages: **zoo** (Zeileis and Grothendieck, 2005) and **xts** (Ryan and Ulrich, 2013). All these objects are specific for temporal data and the corresponding packages provide a complete set of methods to work with them. However, there are slight differences between them. Objects of type **ts** are the most basic and are exclusively addressed for regularly sampled time series. The **zoo** objects incorporate the possibility of dealing with irregularly sampled time series. Finally, the **xts** package further extends the **zoo** package to provide a uniform handling of all the time series data types in R. To calculate the distance measure between two objects of one of these types, the **TSDistances** function just takes care of the conversion of data types and then makes use of the desired **MethodDistance**



**Figure 1:** Structure and organization of the **TSdist** package.

function. Note that, in addition to `ts`, `xts` and `zoo` objects, we can also introduce basic numeric vectors into the `TSDistances` function. In this sense, it generalizes and unifies the calculation of all the distance measures in one function.

Finally, on some occasions, it is necessary to calculate the distance between each pair of series in a given database of series ( $X = \{X_1, X_2, \dots, X_N\}$ ). This will result in a distance matrix such as the following:

$$D(X) = \begin{pmatrix} d(X_1, X_1) & d(X_1, X_2) & \cdots & d(X_1, X_N) \\ d(X_2, X_1) & d(X_2, X_2) & \cdots & d(X_2, X_N) \\ \vdots & \vdots & \ddots & \vdots \\ d(X_N, X_1) & d(X_N, X_2) & \cdots & d(X_N, X_N) \end{pmatrix}$$

The `TSDatabaseDistances` function is specifically designed to build distance matrices from time series databases saved in matrices, `mts` objects, `zoo` objects, `xts` objects or lists. Upon loading the **TSdist** package, the `TSDistances` function is automatically included in the `pr_DB` database, which is a list of similarity measures defined in the `proxy` package. This directly enables the use of the `dist` function, the baseline R function to calculate distance matrices, with the dissimilarity measures defined in the **TSdist** package. This is the general strategy followed by the `TSDatabaseDistances` function and, only for a few special measures, the distance matrix is calculated in other ad-hoc manners for efficiency purposes.

As an additional capability of the `TSDatabaseDistances` function, the distance matrices can not only be calculated for a single database, but also for two separate databases. In this second case, all the pairwise distances between the series in the first database and the second database are calculated:

$$D(X, Y) = \begin{pmatrix} d(X_1, Y_1) & d(X_1, Y_2) & \cdots & d(X_1, Y_N) \\ d(X_2, Y_1) & d(X_2, Y_2) & \cdots & d(X_2, Y_N) \\ \vdots & \vdots & \ddots & \vdots \\ d(X_M, Y_1) & d(X_M, Y_2) & \cdots & d(X_M, Y_N) \end{pmatrix}$$

This last feature is especially useful for classification tasks where train/test validation frameworks are frequently used.

## Summary of distance measures included in **TSdist**

In Table 1, a summary of the distance measures included in **TSdist** is presented. Since the package includes wrapper functions to distance measures hosted in other packages, the original package is also cited in the table.

Based on the literature, we have divided the distance measures into four groups. Shape-based distances compare the overall shape of the time series by measuring the closeness of the raw-values of the time series (Esling and Agon, 2012). Within this category, we separate the (i) lock-step measures, which compare the  $i$ -th point of one time series to the  $i$ -th point of another, and the (ii) elastic measures, which are more flexible and allow one-to-many points and one-to-one point matchings (Wang et al., 2012). Feature-based distances are based on comparing certain features extracted from the series, such as Fourier or wavelet coefficients, autocorrelation values, etc. Next, structure-based distances include (i) model-based approaches, where a model is fit to each series and the comparison is made between models, and (ii) complexity-based models, where the similarity between two series is measured based on the quantity of shared information. Finally, prediction-based distances analyze the similarity of the forecasts obtained for different time series.

	proxy	longitudinal Data	TSclust	dtw	pdc	TSdist
<b>Shape based distances</b>						
<i>Lock-step measures</i>						
$L_p$ distances		✓				✓
DISSIM						✓
Short Time Series Distance (STS)						✓
Cross-correlation based						✓
Pearson correlation based				✓		
CORT distance				✓		
<i>Elastic measures</i>						
Frechet distance		✓				
Dynamic Time Warping (DTW)				✓		
Keogh_LB for DTW						✓
Edit Distance for Real Sequences (EDR)						✓
Edit Distance with Real Penalty (ERP)						✓
Longest Common Subsequence (LCSS)						✓
<b>Feature-based distances</b>						
(Partial) Autocorrelation based			✓			
Fourier Decomposition based						✓
TQuest						✓
Wavelet Decomposition based			✓			
(Integrated) Periodogram based			✓			
SAX representation based			✓			
Spectral Density based			✓			
<b>Structure-based distances</b>						
<i>Model based</i>						
Piccolo distance			✓			
Maharaj distance			✓			
Cepstral based distances			✓			
<i>Compression based</i>						
Compression based distances			✓			
Complexity invariant distance			✓			
Permutation distribution based distance					✓	
<i>Prediction based</i>						
Non Parametric Forecast based			✓			

**Table 1:** Summary of distance measures for time series implemented in R.

As can be seen in Table 1, the distance measures implemented specifically in **TSdist** complement the set of measures already included in other packages, contributing to a more thorough coverage of the existing time series distance measures. As the most notable example, edit based distances for numeric time series (EDR, ERP and LCSS) have been introduced, which were completely overlooked in previous R packages.

For more extensive explanations on each of the distance measures, the readers can access the documentation of the **TSdist** package, where more details or suitable references are provided.

## User interface by example

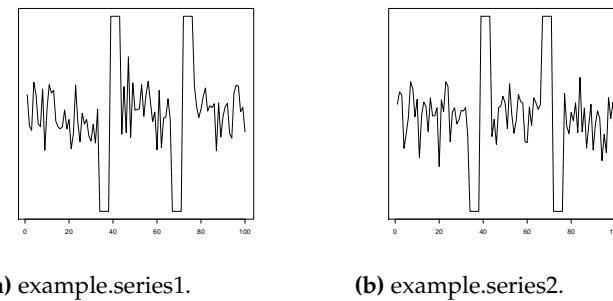
The **TSdist** package is available from the CRAN repository, where the source files for Unix platforms and the binaries for Windows and some OS-X distributions can be downloaded. For more information on software pre-requisites and detailed instructions on the installation process of **TSdist**, please see the README file included in the `inst/doc` directory of the package.

Note that, in the following sections, we will use several time series and time series databases included in **TSdist**. These databases are all synthetic, and have been chosen and designed specifically because of their simplicity and because they allow us to provide straightforward examples which clearly illustrate the usage of the different functions included in the package, and can be easily analyzed, replicated and visualized by the reader. However, once the practitioner becomes familiar with the examples provided in the following sections, it is straightforward to download any real dataset, such as those included in the UCR archive ([Keogh et al.](#)), and work on it.

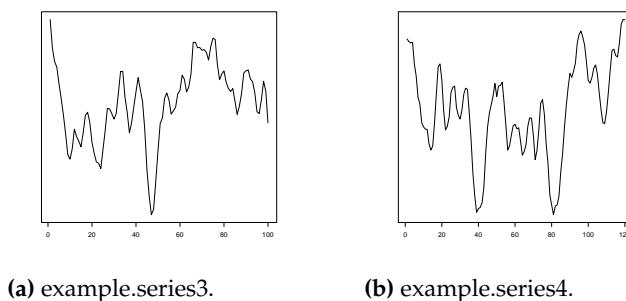
## Examples of distance calculations between numeric vectors

The `example.series1` and `example.series2` objects (see Figure 2) included in the **TSdist** package are two numeric vectors that represent two different synthetic series which were generated based on the shapes that define the Two Patterns synthetic database of series ([Geurts, 2002](#)).

Additionaly, `example.series3` and `example.series4` (see Figure 3) represent two ARMA(3,2) series of coefficients AR=(1, -0.24, 0.1) and MA=(1, 1.2) generated with different random seeds and with different lengths, 100 and 120, respectively.



**Figure 2:** The two example series of the same length included in the TSdist package.



**Figure 3:** The two example series of different length included in the TSdist package.

As mentioned previously, the basic calculation of the distance between two series, such as `example.series1` and `example.series2`, is done by using the `MethodDistance` functions and replacing `Method` with the reference name of the distance measure of choice (for a complete list of reference names, the user can access the help pages of `TSdist`):

```
> CCorDistance(example.series1, example.series2)  
[1] 1.192903  
  
> CorDistance(example.series1, example.series2)  
[1] 1.399347
```

Many of the distance measures require the definition of a parameter, which must be included in the call to the corresponding function:

```
> EDRDistance(example.series1, example.series2, epsilon=0.1)
[1] 80

> ERPDistance(example.series1, example.series2, g=0)
[1] 98.29833
```

Additionally, each distance measure has some characteristics which can impose some constraints on the input time series. For example, some distance measures such as the Euclidean distance can not deal with time series of different lengths. As such, if the conditions are not fulfilled, the distance can not be computed and the function will return NA together with the corresponding error message:

```
> EuclideanDistance(example.series3, example.series4)
Error : Both series must have the same length.
[1] NA

> EDRDistance(example.series3, example.series4, epsilon=0.1, sigma=105)
Error : The window size exceeds the length of the first series
[1] NA
```

Finally, note that all these distance calculations can be carried out by using the `TSdistances` wrapper function as follows:

```
> TSDistances(example.series1, example.series2, distance="ccor")
[1] 1.192903
> TSDistances(example.series1, example.series2, distance="cor")
[1] 1.399347
> TSDistances(example.series1, example.series2, distance="edr", epsilon=0.1)
[1] 80
> TSDistances(example.series1, example.series2, distance="erp", g=0)
[1] 98.29833
```

As can be seen, the distance of choice must be specified within the `distance` argument, followed by the necessary parameters.

We must emphasize that each distance measure is scaled differently and so, distance values obtained from different distance measures are not directly comparable, even when comparing the two same time series. As such, completely different values can be obtained from different distance measures, as can be seen in the previous example.

### Examples of distance calculations between time series objects

The `zoo.series1` and `zoo.series2` time series included in the package are replicas of the `example.series1` and `example.series2` objects introduced previously but saved as `zoo` objects with a specific time index. A basic distance calculation between two series like these is done using the `TSDistances` function exactly as shown in the previous section:

```
> TSDistances(zoo.series1, zoo.series2, distance="cor")
[1] 1.399347
> TSDistances(zoo.series1, zoo.series2, distance="dtw", sigma=10)
[1] 123.8757
```

The distance calculation between `ts` or `xts` objects is done in the same manner.

### Examples of distance matrix calculations

The `example.database` object included in the package is a matrix that represents a database with 6 ARMA(3,2) series of coefficients AR=(1, -0.24, 0.1) and MA=(1, 1.2), but generated with different random seeds. Each time series corresponds to a row of the matrix. Additionally, the `zoo.database` object included in the package is a multivariate `zoo` object that saves the series of `example.database` with a specific time index.

The `dist` function calculates the pairwise distance between all the rows in a matrix so, the calculation of the distance matrix can be done easily for the `example.database` object in the following manner:

```
> dist(example.database, method="TSDistances", distance="tquest",
+      tau=mean(example.database), diag=TRUE, upper=TRUE)

    series1    series2    series3    series4    series5    series6
series1 0.00000000 0.10310669 0.06460465 0.05345349 0.08355246 0.04768702
series2 0.10310669 0.00000000 0.05260503 0.07685220 0.12273356 0.03049604
series3 0.06460465 0.05260503 0.00000000 0.02003566 0.09874005 0.01984044
series4 0.05345349 0.07685220 0.02003566 0.00000000 0.04998743 0.02302477
series5 0.08355246 0.12273356 0.09874005 0.04998743 0.00000000 0.06191323
series6 0.04768702 0.03049604 0.01984044 0.02302477 0.06191323 0.00000000
```

When using the `dist` function with the distances included in **TSdist**, the method argument will always be left as "TSDistances", and the selected distance measure must be introduced in the `distance` argument, followed by its parameters. The `diag` and `upper` options are used to specify if the diagonal and upper triangle of the matrix should be shown. In any case, this calculation can also be done more directly by using the `TSDatabaseDistances` function:

```
> TSDatabaseDistances(example.database, distance="tquest",
+ tau=mean(example.database))
```

When the database is not saved as a matrix, such as with `zoo.database`, the distance matrix calculation can not be done by using the `dist` function directly. In this case, the calculation must necessarily be carried out by using `TSDatabaseDistances`:

```
> TSDatabaseDistances(zoo.database, distance="tquest",
+ tau=mean(zoo.database))
```

	series1	series2	series3	series4	series5
series2	0.10310669				
series3	0.06460465	0.05260503			
series4	0.05345349	0.07685220	0.02003566		
series5	0.08355246	0.12273356	0.09874005	0.04998743	
series6	0.04768702	0.03049604	0.01984044	0.02302477	0.06191323

Note that, by default, the `TSDatabaseDistances` function does not show the diagonal and upper triangle of the computed distance matrix. If we want the whole matrix to appear, we must include the options `diag=TRUE` and `upper=TRUE` as with the `dist` function.

Finally, as previously stated, an additional capability of the `TSDatabaseDistances` function is that it is capable of calculating distances between the time series in two separate databases:

```
> TSDatabaseDistances(example.database, zoo.database, distance="tquest",
+ tau=mean(zoo.database))
```

	series1	series2	series3	series4	series5	series6
series1	0.00000000	0.10310669	0.06460465	0.05345349	0.08355246	0.04768702
series2	0.10310669	0.00000000	0.05260503	0.07685220	0.12273356	0.03049604
series3	0.06460465	0.05260503	0.00000000	0.02003566	0.09874005	0.01984044
series4	0.05345349	0.07685220	0.02003566	0.00000000	0.04998743	0.02302477
series5	0.08355246	0.12273356	0.09874005	0.04998743	0.00000000	0.06191323
series6	0.04768702	0.03049604	0.01984044	0.02302477	0.06191323	0.00000000

Note that the two databases do not have to be provided in identical formats.

## Time series classification and clustering with the **TSdist** package

The most common usage of time series distance measures is within clustering and classification tasks,<sup>1</sup> and all the measures included in this package can be useful within these two frameworks. As a support for these two tasks, the **TSdist** package includes two well-known functions.

The first function (`OneNN`) implements the 1NN classifier. This classifier is commonly used to evaluate the performance of different distance measures, due to the influence the distance measure has on its performance together with its reduced number of parameters (Wang et al., 2012). Given a pair of train/test time series datasets and the class values of the series in the training set, the `oneNN` function outputs the predicted class values for the test series. Additionally, if the ground truth class values of the series in the testing set are provided by the user, the error obtained in the classification process is also calculated.

As an example of usage, suppose we want to classify the series in the `example.database2` database (included in **TSdist**), which contains 100 series from 6 classes. In order to simulate a typical classification framework, we divide the database into two sets by randomly selecting 30% of the series for training purposes and 70% for testing.<sup>2</sup> Then, we apply the 1-NN classifier to the testing set with any distance measure of choice:

<sup>1</sup>Beware when using these distance measures within kernel based classifiers. Some of them, such as DTW, do not necessarily issue positive definite Gram matrices when inserted directly into common kernel functions, such as the Gaussian RBF. More information and some possible solutions can be found in (Cuturi, 2011; Pree et al., 2014; Gaidon et al., 2011; Marteau and Gibet, 2014).

<sup>2</sup>The code to load and prepare the data is available in the documentation of the `OneNN` function.

```
> OneNN(train, trainclass, test, "euclidean")
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 4 4 4 4
[39] 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
```

Additionally, if the selected distance measure requires the definition of any parameters, these should be included at the end of the call:

```
> OneNN(train, trainclass, test, "tquest", tau=85)
[1] 1 3 3 3 2 3 3 3 5 1 2 3 3 3 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 4 6 4 4
[39] 4 6 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 6 4 6 6 4 4 6 4 6 6 6 6 6 6 6 6 6 6
```

If we also provide the true class labels of the test instances, we can obtain the classification error obtained by the 1NN algorithm and the distance measure of choice:

```
> OneNN(train, trainclass, test, testclass, "euclidean")$error
[1] 0

> OneNN(train, trainclass, test, testclass, "acf")$error
[1] 0.4142857

> OneNN(train, trainclass, test, testclass, "tquest", tau=85)$error
[1] 0.3285714

> OneNN(train, trainclass, test, testclass, "dtw", sigma=20)$error
[1] 0
```

For clustering tasks, the `k.medoids` function can be used, which, given the data and the number of clusters, outputs the clustering result together with the F evaluation measure ([Wagner and Wagner, 2007](#)), if the ground truth clustering is provided by the user. In the following example, the popular k-medoids algorithm is applied to the `example.database3` database, (which contains series from 5 classes obtained from ARMA processes), using different distance measures and setting the number of clusters to 5:

```
> KMedoids(data, 5, "euclidean")
[1] 1 1 1 2 1 2 3 2 1 2 2 4 1 4 5 1 4 1 4 1 5 2 5 5 5 5 2 4 2 4 3 3 2
[34] 3 2 2 3 2 3 2 5 5 2 5 1 2 5 2 5 2

> KMedoids(data, 5, "tquest", tau=0)
[1] 1 1 1 2 1 2 3 1 1 1 2 2 4 2 4 4 2 4 2 4 3 3 3 2 3 3 2 2 3 2 3 3 2
[34] 3 2 2 3 2 3 2 3 5 2 5 1 2 5 2 5 2
```

As mentioned, if we provide the ground truth clustering result, we can also obtain the F measure of the obtained clustering:

```
> KMedoids(data, 5, ground.truth, "euclidean")$F
[1] 0.5154762

> KMedoids(data, 5, ground.truth, "acf")$F
[1] 0.9799499

> KMedoids(data, 5, ground.truth, "tquest", tau=0)$F
```

```
[1] 0.594479
> KMedoids(data, 5, ground.truth, "dtw", sigma=20)$F
[1] 0.8933333
```

As can be seen, the best results are provided by the Euclidean distance and DTW when we classify the `example.database2` database, and the autocorrelation distance is the best performing measure from the selected options when clustering the `example.database3` database.

In this line, previous experiments show that there is no “best” distance measure which is suitable for all databases and all tasks, (Wang et al., 2012). In this context, a specific distance measure must be selected, in each case, in order to obtain satisfactory results (Mori et al., 2016). The large number of distance measures included in **TSdist** and the simple design of this package allows the user to try different distance measures directly, simplifying the distance measure selection process considerably.

## Summary and conclusions

The **TSdist** package enables the calculation of distances between time series and time series databases, by using a large variety of measures available in the literature. By including wrapper functions for time series distances already available in R, and implementing other unavailable popular measures reviewed in the literature, this package provides the largest selection of time series distance measures available at R at the moment, to the best of our knowledge. Additionally, it also simplifies the evaluation of these measures and their application in classification and clustering contexts by providing several ad-hoc functions.

For more detailed information on the databases and functions included in the **TSdist** package, and a more complete set of examples, the reader can consult the help pages or the manual of the **TSdist** package and the vignette included within.

## Bibliography

- A. Bagnall, A. Bostrom, J. Large, and J. Lines. The great time series classification bake off: An experimental evaluation of recently proposed algorithms. Extended version. *ArXiv e-prints*, Feb. 2016. [p455]
- A. M. Brandmaier. pdc: An “R” package for complexity-based clustering of time series. *Journal of Statistical Software*, 67(1):1–23, 2015. ISSN 1548-7660. doi: 10.18637/jss.v067.i05. URL <https://www.jstatsoft.org/index.php/jss/article/view/v067i05>. [p455]
- M. Cuturi. Fast Global Alignment Kernels. In *Proceedings of the 28th International Conference on Machine Learning*, pages 929–936, 2011. [p460]
- P. Esling and C. Agon. Time-series data mining. *ACM Computing Surveys*, 45(1):1–34, Nov. 2012. [p455, 456]
- T.-C. Fu. A Review on Time Series Data Mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, Feb. 2011. [p455]
- A. Gaidon, Z. Harchaoui, and C. Schmid. A time series kernel for action recognition. In *BMVC 2011 - British Machine Vision Conference*, pages 63.1–63.11, 2011. [p460]
- C. Genolini. *longitudinalData: Longitudinal Data*, 2014. URL <http://CRAN.R-project.org/package=longitudinalData>. R package version 2.2. [p455]
- P. Geurts. *Contributions to decision tree induction: bias/variance tradeoff and time series classification*. PhD thesis, University of Liege, Belgium., 2002. [p457]
- T. Giorgino. Computing and visualizing dynamic time warping alignments in R: the dtw package. *Journal of Statistical Software*, 31(7), 2009. [p455]

- E. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei, and C. Ratanamahatana. The UCR Time Series Classification/Clustering Homepage. URL [www.cs.ucr.edu/~eamonn/time\\_series\\_data/](http://www.cs.ucr.edu/~eamonn/time_series_data/). [p457]
- T. W. Liao. Clustering of time series data: a survey. *Pattern Recognition*, 38(11):1857–1874, Nov. 2005. [p455]
- P.-F. Marteau and S. Gchet. On recursive edit distance kernels with applications to time series classification. *IEEE Transactions on Neural Networks and Learning Systems*, PP(6):1–13, 2014. [p460]
- D. Meyer and C. Buchta. *proxy: Distance and Similarity Measures*, 2015. URL <http://CRAN.R-project.org/package=proxy>. R package version 0.4-14. [p455]
- P. Montero and J. A. Vilar. TSclust: An R package for time series clustering. *Journal of Statistical Software*, 62(1):1–43, 2014. [p455]
- U. Mori, A. Mendiburu, and J. A. Lozano. TSdist: Distance Measures for Time Series data., 2015. URL <http://CRAN.R-project.org/package=TSdist>. [p455]
- U. Mori, A. Mendiburu, and J. Lozano. Similarity measure selection for clustering time series databases. *Knowledge and Data Engineering, IEEE Transactions on*, 28(1):181–195, Jan 2016. ISSN 1041-4347. [p462]
- H. Pree, B. Herwig, T. Gruber, B. Sick, K. David, and P. Lukowicz. On general purpose time series similarity measures and their use as kernel functions in support vector machines. *Information Sciences*, 281:478–495, Oct. 2014. [p460]
- R Core Team. R: A Language and Environment for Statistical Computing, 2014. URL <http://www.r-project.org/>. [p455]
- J. A. Ryan and J. M. Ulrich. xts: eXtensible Time Series, 2013. URL <http://cran.r-project.org/package=xts>. [p455]
- S. Wagner and D. Wagner. Comparing Clusterings - An Overview. Technical Report 2006-04, Universität Karlsruhe (TH), 2007. URL <http://digbib.ubka.uni-karlsruhe.de/volltexte/1000011477>. [p461]
- X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2012. [p455, 456, 460, 462]
- A. Zeileis and G. Grothendieck. zoo: S3 Infrastructure for Regular and Irregular Time Series. *Journal of Statistical Software*, 14(6):1–27, 2005. [p455]

*Usue Mori*

*Department of Applied Mathematics, Statistics and Operational Research  
University of the Basque Country, UPV/EHU  
20018, Donostia/San Sebastian (Spain)*  
[usue.mori@ehu.eus](mailto:usue.mori@ehu.eus)

*Alexander Mendiburu*

*Department of Computer Architecture and Technology  
University of the Basque Country, UPV/EHU  
20018, Donostia/San Sebastian (Spain)*  
[alexander.mendiburu@ehu.eus](mailto:alexander.mendiburu@ehu.eus)

*Jose A. Lozano*

*Department of Computer Science and Artificial Intelligence  
University of the Basque Country, UPV/EHU  
20018, Donostia/San Sebastian (Spain)*  
[ja.lozano@ehu.eus](mailto:ja.lozano@ehu.eus)

# condSURV: An R Package for the Estimation of the Conditional Survival Function for Ordered Multivariate Failure Time Data

by Luis Meira-Machado and Marta Sestelo

**Abstract** One major goal in clinical applications of time-to-event data is the estimation of survival with censored data. The usual nonparametric estimator of the survival function is the time-honored Kaplan-Meier product-limit estimator. Though this estimator has been implemented in several R packages, the development of the **condSURV** R package has been motivated by recent contributions that allow the estimation of the survival function for ordered multivariate failure time data. The **condSURV** package provides three different approaches all based on the Kaplan-Meier estimator. In one of these approaches these quantities are estimated conditionally on current or past covariate measures. Illustration of the software usage is included using real data.

## Introduction

One major goal in survival studies is the estimation of the survival function. The most popular method for estimating this function is the well-known product-limit estimator also known as Kaplan-Meier estimator (Kaplan and Meier, 1958). The popularity of the product-limit estimator is explained by its simplicity and intuitive appeal while requiring very weak assumptions. It simply takes into account the empirical probability of surviving over a certain time. The method does not take into account of covariates, so it is mainly descriptive. Discrete covariates can be included by splitting the sample for each level of the covariate and applying the product-limit estimator for each subsample. This approach is not recommended for continuous covariates. To account for this extra difficulty several generalizations to the Kaplan-Meier estimator have been proposed throughout the last decades. Beran (1981) was the first one who proposed an estimator of the conditional distribution (survival) function with censored data in a fully nonparametric way. His estimator was further studied among others by Dabrowska (1987), Akritas (1994), Gonzalez-Manteiga and Cadarso-Suárez (1994) and Van Keilegom et al. (2001). All these estimators can be used to estimate the distribution (or survival) function conditional on a continuous covariate in a regression model, when data are subject to censoring. However, none of the above methods can be used to estimate the conditional survival when the covariate is censored.

Several software packages in the form of R packages have been developed to estimate the survival function. Though this function can be estimated parametrically or using nonparametric maximum likelihood estimation, the product limit Kaplan-Meier estimator is still one of the best options for estimating the survival function. Several R packages have been developed to implement the product-limit Kaplan-Meier estimator. For instance, the **survival** package (Therneau, 2015) and the **prolim** package (Gerds, 2015) can be used to obtain Kaplan-Meier estimates. A comprehensive list of the available packages which can be used to estimate the survival function can be seen in the CRAN Task View “Survival Analysis” (Allignol and Latouche, 2016) of the Comprehensive R Archive Network (CRAN).

In many longitudinal medical studies, patients may experience several events through a follow-up period. In these studies, the analysis of sequentially ordered events are often of interest. The events of concern can be of the same nature (e.g., recurrent disease episodes in cancer studies) or represent different states in the disease process (e.g., “alive and disease-free”, “alive with recurrence” and “dead”). If the events are of the same nature, this is usually referred as recurrent events (Cook and Lawless, 2007), whereas if they are based on different disease categories they are usually modeled through their intensity functions (Meira-Machado et al., 2009). Again, several R packages have been developed to deal with problems that arise in these processes (see for example, Allignol and Latouche 2016). Some of these packages can be used to estimate occupation probabilities, transition probabilities and the cumulative incidence functions. However, none can be used to estimate conditional survival probabilities such as:  $P(T_2 > y | T_1 > x)$ ,  $P(T_3 > y | T_1 < x_1, T_2 > x_2)$  or  $P(T_3 > y | T_1 > x_1, T_2 > x_2)$  where  $T_1$ ,  $T_2$  and  $T_3$  are ordered event times of successive events. This issue was recently considered by Meira-Machado et al. (2016) who proposed nonparametric and semiparametric estimators for such quantities.

This paper describes the R package **condSURV** (available from the Comprehensive R Archive

Network at <https://CRAN.R-project.org/package=condSURV/>) and its capabilities for implementing nonparametric and semiparametric estimators for conditional survival probabilities for two multivariate ordered times. The package can also be used to estimate more general functions involving more than two successive event times. The estimation of these quantities is essential for long-term survival prognosis which arises in many medical contexts such as cancer studies, asthma, HIV/AIDS, heart disease, dementia and Alzheimer's disease, etc. The methods implemented in the package also deal with the possible effect of covariates on the conditional survival probabilities (e.g.,  $P(T_2 > y | T_1 > x_1, Z = z)$  where  $Z$  denotes a continuous covariate). To account for the covariate effect, a flexible approach is based on local smoothing by means of kernel weights based on local constant (Nadaraya-Watson) regression. In this article we explain and illustrate how numerical and graphical output for all methods can be obtained using the **condSURV** package.

The remainder of this paper is organized as follows. The following section provides a brief introduction to the methodological background. All estimators for the conditional survival function are presented. Then, a detailed description of the package is presented, and its usage is illustrated through the analysis of a real data set; and finally, the last section contains the main conclusions of this work. The use of the package to more than two consecutive events is illustrated in the [Appendix](#).

## Methodology background

Suppose that an individual may experience  $K$  consecutive events at times  $T_1 < T_2 < \dots < T_K = T$ , which are measured from the start of the follow-up. In this section different methods are proposed to estimate conditional survival probabilities such as  $P(T_2 > y | T_1 > x)$  or  $P(T_2 > y | T_1 \leq x)$ , where  $T_1$  and  $T_2$  are ordered event times of two successive events. The proposed methods are all based on the Kaplan-Meier estimator and the ideas behind the proposed estimators can also be used to estimate more general functions involving more than two successive event times. However, for ease of presentation and without loss of generality, we take  $K = 2$  in this section. The extension to  $K > 2$  is straightforward.

Let  $(T_1, T_2)$  be a pair of successive event times corresponding to two ordered (possibly consecutive) events measured from the start of the follow-up. Let  $T = T_2$  denote the total time and assume that both  $T_1$  and  $T$  are observed subject to a (univariate) random right-censoring variable  $C$  assumed to be independent of  $(T_1, T)$ . Due to censoring, rather than  $(T_1, T)$  we observe  $(\tilde{T}_1, \Delta_1, \tilde{T}, \Delta_2)$  where  $\tilde{T}_1 = \min(T_1, C)$ ,  $\Delta_1 = I(T_1 \leq C)$ ,  $\tilde{T} = \min(T, C)$ ,  $\Delta_2 = I(T \leq C)$ , where  $I(\cdot)$  is the indicator function. Let  $(\tilde{T}_{1i}, \Delta_{1i}, \tilde{T}_i, \Delta_{2i})$ ,  $1 \leq i \leq n$  be independent and identically distributed data with the same distribution as  $(\tilde{T}_1, \Delta_1, \tilde{T}, \Delta_2)$ .

Let  $S_1$  and  $S$  be the marginal survival functions of  $T_1$  and  $T$ ; that is,  $S_1(y) = P(T_1 > y)$  and  $S(y) = P(T > y)$ . Introduce also the conditional survival probabilities  $P(T > y | T_1 > x)$  and  $P(T > y | T_1 \leq x)$ . without loss of generality, we only consider the estimation of  $S(y|x) = P(T > y | T_1 > x)$ .

The Kaplan-Meier estimator, also known as the product-limit estimator, is the most frequently used method to estimate survival for censored data. The most used representation of the Kaplan-Meier estimator of the total time is through a product of the following form

$$\hat{S}(y) = \prod_{\tilde{T}_i \leq t} \left( 1 - \frac{\Delta_{2i}}{R(\tilde{T}_i)} \right),$$

where  $R(t) = \sum_{i=1}^n I(\tilde{T}_i \geq t)$  denotes the number of individuals at risk just before time  $t$ . The censoring is assumed to be independent in the sense that the additional knowledge of the right-censoring times before any time  $y$  does not carry information on the risk of failure at time  $y$ . The Kaplan-Meier estimate is a step function with jumps at event times. The size of the steps depends on the number of events and the number of individuals at risk at the corresponding time. Below we introduce a weighted average representation of the Kaplan-Meier estimator which will be used later to introduce estimators for the conditional survival function

$$\hat{S}(y) = 1 - \sum_{i=1}^n W_i I(\tilde{T}_{(i)} \leq y),$$

where  $\tilde{T}_{(1)} \leq \dots \leq \tilde{T}_{(n)}$  denotes the ordered  $\tilde{T}$ -sample and

$$W_i = \frac{\Delta_{2[i]}}{n-i+1} \prod_{j=1}^{i-1} \left[ 1 - \frac{\Delta_{2[j]}}{n-j+1} \right]$$

is the Kaplan-Meier weight attached to  $\tilde{T}_{(i)}$ . In the expression of  $W_i$  notation  $\Delta_{2[i]}$  is used for the  $i$ -th

concomitant value of the censoring indicator (that is,  $\Delta_{2[i]} = \Delta_{2j}$  if  $\tilde{T}_{(i)} = \tilde{T}_j$ ).

In this work we are interested in the estimation of the conditional survival function,  $S(y | x) = P(T > y | T_1 > x)$ . Below we provide estimators for this quantity, all based on the Kaplan-Meier estimator.

The conditional survival function  $S(y | x)$  can be expressed as  $S(y | x) = P(T > y | T_1 > x) = 1 - P(T \leq y | T_1 > x) = 1 - P(T_1 > x, T \leq y) / (1 - P(T_1 \leq x))$ . Then, the denominator of the term at the right hand side can be estimated using the Kaplan-Meier estimator of survival of the first time; the quantity at the numerator involves transformations of the pair  $(T_1, T)$  which cannot be estimated so simply. This quantity can be estimated using Kaplan-Meier weights pertaining to the distribution of the total time to weight the bivariate data (Meira-Machado et al., 2016). The corresponding estimator is given as follows:

$$\hat{S}^{\text{KMW}}(y | x) = 1 - \frac{\sum_{i=1}^n W_i I(\tilde{T}_{1[i]} > x, \tilde{T}_{(i)} \leq y)}{\hat{S}_1(x)}. \quad (1)$$

Another way to introduce a (monotonic) nonparametric estimator for the conditional survival is by considering the landmark approach (Van Houwelingen, 2007). Given the time point  $x$ , to estimate  $S(y | x) = P(T > y | T_1 > x)$  the analysis can be restricted to the individuals with an observed first event time greater than  $x$ . Then, an estimator for the conditional survival function is just the Kaplan-Meier estimator of the survival function of  $T$  computed from such a subset

$$\hat{S}^{\text{LDM}}(y | x) = \hat{S}^x(y), \quad (2)$$

where  $\hat{S}^x(y)$  is the Kaplan-Meier estimator of survival computed from the  $(\tilde{T}, \Delta_2)$ -sample in  $\{i : \tilde{T}_{1i} > x\}$  ordered with respect to  $\tilde{T}$ .

The standard error of the nonparametric landmark estimator (LDM) may be large when the censoring is heavy, particularly with a small sample size. Interestingly, the variance of this estimator may be reduced by presmoothing (Dikta, 1998). Here, the idea of presmoothing involves replacing the censoring indicators (in the expression of the Kaplan-Meier weights) by some smooth fit before the Kaplan-Meier formula is applied. This preliminary smoothing may be based on a certain parametric family such as the logistic (thus leading to a semiparametric estimator), or on a nonparametric estimator of the binary regression curve. The corresponding presmoothed landmark estimator is then given by

$$\hat{S}^{\text{PLDM}}(y | x) = 1 - \sum_{i=1}^{n_x} w_i^x I(\tilde{T}_{(i)}^x \leq y), \quad (3)$$

where  $w_i^x$  is defined through

$$w_i^x = \frac{m(\tilde{T}_{(i)}^x)}{n_x - i + 1} \prod_{j=1}^{i-1} \left[ 1 - \frac{m(\tilde{T}_{(j)}^x)}{n_x - j + 1} \right], \quad 1 \leq i \leq n_x,$$

where  $(\tilde{T}_{(i)}^x, \Delta_{2[i]}^x)$ ,  $i = 1, \dots, n_x$ , is the  $(\tilde{T}, \Delta_2)$ -sample in  $\{i : \tilde{T}_{1i} > x\}$  ordered with respect to  $\tilde{T}$ .

Here,  $m(t) = P(\Delta_2 = 1 | \tilde{T} = t)$ , where  $m(\tilde{T})$  belongs to a parametric (smooth) family of binary regression curves, e.g., logistic. In practice, we assume that  $m(t) = m(t; \beta)$  where  $\beta$  is a vector of parameters which will be computed by maximizing the conditional likelihood of the  $\Delta_2$ 's given  $\tilde{T}$ .

Note that  $\hat{S}^{\text{PLDM}}(y | x)$  is the presmoothed Kaplan-Meier estimator of survival computed from the  $(\tilde{T}, \Delta_2)$ -sample in  $\{i : \tilde{T}_{1i} > x\}$  ordered with respect to  $\tilde{T}$ .

The practical performance of the proposed estimators for the conditional survival function has been investigated through simulations (Meira-Machado et al., 2016). It has been demonstrated that all of the studied estimators perform well, approaching their targets as the sample size increases. Besides, simulation results reveal that the landmark estimator (LDM) performs favorably when compared with the first method (KMW). Furthermore, the reported simulation results reveal relative benefits of presmoothing (PLDM) in the heavily censored scenarios or for small sample sizes.

Now we will explain how to introduce covariate information in the conditional survival function. Discrete covariates can be also included by splitting the sample for each level of the covariate and repeating the described procedures for each subsample. This approach is not valid for continuous covariates. To estimate the survival probabilities conditionally on continuous covariates

we propose to use local smoothing which is introduced using regression weights. Without loss of generality the methodology will be explained in the build of the conditional survival probability  $P(T > y | T_1 > x, Z = z)$ , where  $Z$  denotes a continuous covariate. To estimate  $S(y | x, z) = 1 - P(T_1 > x, T \leq y | Z = z) / (1 - P(T_1 \leq x | Z = z))$  we need to estimate the following conditional expectations:  $E(I(T_1 > x, T \leq y) | Z)$  and  $E(I(T_1 \leq x) | Z)$ .

In the absence of censoring, to estimate these quantities, we may use kernel smoothing techniques by calculating a local average of the indicator functions. For example,  $E[I(T_1 > x, T \leq y) | Z = z]$  could be estimated as follows

$$\hat{E}[I(T_1 > x, T \leq y) | Z = z] = \sum_{i=1}^n NW_i(z, a_n) I(\tilde{T}_1 > x, \tilde{T} \leq y),$$

where  $NW_i(z, a_n)$  is a weight function which corresponds to the Nadaraya-Watson (Nadaraya, 1965; Watson, 1964) estimator (NW) as follows

$$NW_i(z, a_n) = \frac{K((z - Z_i)/a_n)}{\sum_{j=1}^n K((z - Z_j)/a_n)},$$

where  $K$  is a known probability density function (the kernel function) and  $a_n$  is a sequence of bandwidths.

In our case, however, we allow the data to be right-censored. To handle right-censoring, inverse probability of censoring weighting (IPCW; see for example, Satten and Datta 2001) can be used. In order to introduce our estimators, note that, assuming that the support of the conditional distribution of  $T$  is contained in that of  $C | Z$ , we have  $E[I(T_1 > x, T \leq y) | Z] = E[I(\tilde{T}_1 > x, \tilde{T} \leq y) \Delta_2 / G_Z(\tilde{T}) | Z]$  and  $E[I(T_1 \leq x) | Z] = E[I(\tilde{T}_1 \leq x) \Delta_1 / H_Z(\tilde{T}_1) | Z]$  where  $G_Z$  and  $H_Z$  denote the conditional survival functions of the censoring variable of the total time and the first event time, respectively, given  $Z$ .

The estimation of the conditional survival function, given a covariate under random censoring has been considered in many papers. This topic was introduced by Beran (1981) and was further studied by several authors (Dabrowska, 1987; Akritas, 1994; Gonzalez-Manteiga and Cadarso-Suárez, 1994; Van Keilegom et al., 2001). Their proposals can also be used to estimate the conditional survival function of  $C | Z = z$ , say  $\hat{G}_Z$ . This can be done using the estimator introduced by Beran,

$$\hat{G}_Z(y) = \prod_{T_i \leq y, \Delta_{2i}=0} \left[ 1 - \frac{NW_i(z, a_n)}{\sum_{j=1}^n I(T_j \geq T_i) NW_j(z, a_n)} \right].$$

In order to introduce our estimators we propose to plug-in Beran's estimator  $\hat{G}_Z$  and use NW to compute

$$\hat{P}(T_1 > x, T \leq y | Z = z) = \sum_{i=1}^n NW_i(z, a_n) \frac{I(\tilde{T}_{1i} > x, \tilde{T}_i \leq y) \Delta_{2i}}{\hat{G}_{Z_i}(\tilde{T}_i)}.$$

Similarly, we propose to plug-in Beran's estimator  $\hat{H}_Z$  and use NW to compute

$$\hat{P}(T_1 \leq x | Z = z) = \sum_{i=1}^n NW_i(z, a_n) \frac{I(\tilde{T}_{1i} \leq x) \Delta_{1i}}{\hat{H}_{Z_i}(\tilde{T}_{1i})}.$$

Then, we may introduce the IPCW estimator as follows:

$$\hat{S}^{\text{IPCW}}(y | x, z) = 1 - \hat{P}(T_1 > x, T \leq y | Z = z) / (1 - \hat{P}(T_1 \leq x | Z = z)). \quad (4)$$

## condSURV in practice

This section introduces an overview of how the package is structured. **condSURV** is a shortcut for “conditional survival” and this is its major functionality: to provide estimates of the survival function conditional to previous (possibly censored) events. This software enables both numerical and graphical outputs to be displayed for all methods (KMW, LDM, PLDM and IPCW) described in the previous section. This software is intended to be used with the R environment for statistical computing and graphics. Our package is composed of 12 functions that allow users to obtain estimates for all proposed methods.

Function	Description
survCOND	Conditional survival probabilities based on Kaplan-Meier weights and the Landmark approaches. This function also implements estimation methods for these quantities conditionally on current or past covariate measures.
survCS	Create a "survCS" object, usually used as a response variable in a model formula.
plot.survCS	Plot for an object of class "survCS".
summary.survCS	Summary for an object of class "survCS".
print.survCS	Print for an object of class "survCS".
KM	Computes the Kaplan-Meier product-limit of survival.
PKM	Computes the presmoothed Kaplan-Meier product-limit of survival.
Beran	Computes the conditional survival probability $P(T > y   T_1 = x)$ using Beran's estimator.
KMW	Returns a vector with the Kaplan-Meier weights.
PKMW	Returns a vector with the presmoothed Kaplan-Meier weights.
LLW	Returns a vector with the local linear weights.
NWW	Returns a vector with the Nadaraya-Watson weights.

**Table 1:** Summary of functions in the **condSURV** package.

Details on the usage of the functions (described in Table 1) can be obtained with the corresponding help pages.

It should be noted that to implement the methods described in Section [Methodology background](#) one needs the following variables of data in a specific order (as shown): time1, event1, Stime and event. The variable time1 represents the observed time to the first event of interest, and event1 the corresponding status/censoring indicator (if the survival time is a censored observation, the value is 0 and otherwise the value is 1). The variable Stime represents the total survival time. If event1 = 0, then the total survival time is equal to the observed time to the first event. The variable event is the final status of the individual (takes the value 1 if the final event of interest is observed and 0 otherwise). The illustration of the **condSURV** package for more than two event times is discussed in the [Appendix](#).

### Example of application

For illustration, we apply the proposed methods to data from a large clinical trial on Duke's stage III patients, affected by colon cancer, that underwent a curative surgery for colorectal cancer ([Moertel et al., 1990](#)). This data set is freely available as part of the R **survival** package. The data is also available as part of the R package **condSURV**. From the total of 929 patients, 468 developed a recurrence and among these 414 died. For each individual, an indicator of his/her final vital status (censored or not), the survival times (time to recurrence, time to death) from the entry of the patient in the study (in days), and a vector of covariates including age (in years) and recurrence (coded as 1 = yes; 0 = no) were recorded. The covariate recurrence is a time-dependent covariate which can be expressed as an intermediate event. We will use the methods described in Section [Methodology background](#) to study survival as well as the effect of recurrence on the final outcome (death).

In the following, we will demonstrate the package capabilities using this data. Below is an excerpt of the data.frame with one row per individual.

```
> library("condSURV")
> data(colonCS)
> head(colonCS[, 1:7])
```

	time1	event1	Stime	event	rx	sex	age
1	968	1	1521	1	Lev+5FU	1	43
2	3087	0	3087	0	Lev+5FU	1	63
3	542	1	963	1	Obs	0	71
4	245	1	293	1	Lev+5FU	0	66
5	523	1	659	1	Obs	1	69
6	904	1	1767	1	Lev+5FU	0	57

Individuals represented in lines 1, 3, 4, 5 and 6 experienced a recurrence of the tumor and have died;

the individual represented in line 2 is alive and without recurrence at the end of follow-up. We note that event1 = 1 and event = 0 corresponds to individuals with an observed recurrence that remain alive at the end of the follow-up.

The development of the **condSURV** R package has been motivated by recent contributions that allow the estimation of the (conditional) survival function for ordered multivariate failure time data. This package contains the function `survCS` which takes the input data as an R formula and creates a survival object among the chosen variables for analysis. This function will verify if the data has been introduced correctly and will create a "survCS" object. Arguments in this function must be introduced in the following order `time1, event1, time2, event2, ..., Stime` and `event`, where `time1, time2, ..., Stime` are ordered event times and `event1, event2, ..., event` their corresponding indicator statuses. This function plays a similar role as the `Surv` function in the **survival** R package.

The effect of "recurrence" is important on the patient outcome and can be studied through the ordered multivariate event time data of time-to-event from enrolment, to recurrence and to death. Results obtained from the estimation of the conditional survival probabilities,  $S(y | x) = P(T > y | T_1 > x)$ , can be used to understand which individuals without recurring cancer after surgery are most likely to survive from their disease and which would benefit from more personal attention, closer follow-up and monitoring. Below we discuss how to estimate this and other quantities using the **condSURV** package.

Estimates for the conditional survival probabilities are obtained using function `survCOND`. The first argument of this function is a formula object with the response on the left of a `~` operator. The response must be a "survCS" object which is obtained using the `survCS` function. A single covariate (qualitative or quantitative) can be included in the right hand side of the formula allowing the estimation of survival probabilities conditionally on current or past covariate measures. The use of the main function `survCOND` is explained below.

In the absence of covariates, two methods can be used to estimate the conditional survival probabilities: the method based on the use of Kaplan-Meier weights (KMW) and the method based on the landmark approach (KMW). A smoothed version of the landmark approach is also implemented. Given  $x = 365$  (one year) and  $y = 1825$  (five years), estimates for  $S(y | x) = P(T > y | T_1 > x)$  can be obtained using function `survCOND` with the method based on the use of Kaplan-Meier weights (`method = "KMW"`):

```
> set.seed(123)
> colon.kmw.1 <- survCOND(survCS(time1, event1, Stime, event) ~ 1, x = 365, y = 1825,
+   data = colonCS, method = "KMW")
> summary(colon.kmw.1)

P(T>y|T1>365)

      y  estimate lower 95% CI upper 95% CI
1825 0.7303216    0.697005    0.7562444
```

As can be seen, the `survCOND` function provides, by default, 95% pointwise confidence intervals (`conf = TRUE`) using 200 bootstrap replicates (`n.boot = 200`). The construction of the pointwise confidence intervals is obtained by means of the bootstrap percentile method by randomly sampling the  $n$  items from the original data set with replacement (Davison and Hinkley, 1997). Intervals with other levels of confidence besides 95% (the default value) can be obtained by setting the argument `conf.level` to the desired level.

Given a fixed value of  $x$ , estimates for the conditional survival can be obtained for a vector of  $y$  values. An example is given below:

```
> colon.kmw.2 <- survCOND(survCS(time1, event1, Stime, event) ~ 1, x = 365,
+   y = 365 * 1:7, data = colonCS, method = "KMW")
> summary(colon.kmw.2)

P(T>y|T1>365)
```

y	estimate	lower 95% CI	upper 95% CI
365	1.0000000	1.0000000	1.0000000
730	0.9441430	0.9265015	0.9599035
1095	0.8624983	0.8353103	0.8843765
1460	0.7750519	0.7389898	0.8090082
1825	0.7303216	0.6920535	0.7664671
2190	0.6879923	0.6511133	0.7249555
2555	0.6548414	0.6114144	0.6940938

If argument  $y$  is omitted, then the `survCOND` function allows the user to obtain estimates for all possible  $y$  values. Then, one can use the `summary` function to get the estimated values at the desired values (through argument `times` of the `summary` function). A truncated output for the following input commands is shown below:

```
> colon.kmw.3 <- survCOND(survCS(time1, event1, Stime, event) ~ 1, x = 365,
+ data = colonCS, method = "KMW")
> summary(colon.kmw.3)
```

$P(T > y | T_1 > 365)$

$y$	estimate	lower	95% CI	upper	95% CI
365.0	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000
421.0	0.9985694	0.9956263	1.0000000	1.0000000	1.0000000
430.0	0.9971388	0.9928077	1.0000000	1.0000000	1.0000000
448.0	0.9957082	0.9900836	1.0000000	1.0000000	1.0000000
454.5	0.9942758	0.9871742	0.9985958	0.9985958	0.9985958
465.0	0.9928434	0.9853940	0.9985509	0.9985509	0.9985509
485.0	0.9914111	0.9826719	0.9971681	0.9971681	0.9971681
486.0	0.9899787	0.9808888	0.9958287	0.9958287	0.9958287
499.0	0.9885463	0.9797218	0.9956079	0.9956079	0.9956079
.....	.....	.....	.....	.....	.....

Similarly, one can obtain the results for the landmark methods (LDM and PLDM) using the same function `survCOND`. The unsmoothed landmark estimator is obtained using argument `method = "LDM"` whereas for obtaining the presmoothed landmark estimator the argument `presmooth = TRUE` is also required.

```
> colon.ldm.1 <- survCOND(survCS(time1, event1, Stime, event) ~ 1, x = 365,
+ data = colonCS, method = "LDM")
> summary(colon.ldm.1, times = 365 * 1:7)
```

$y$	estimate	lower	95% CI	upper	95% CI
365	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000
730	0.9441319	0.9296298	0.9614001	0.9614001	0.9614001
1095	0.8624695	0.8418715	0.8877858	0.8877858	0.8877858
1460	0.7750019	0.7413340	0.8002003	0.8002003	0.8002003
1825	0.7302521	0.6957642	0.7584186	0.7584186	0.7584186
2190	0.6878056	0.6515754	0.7196251	0.7196251	0.7196251
2555	0.6543273	0.6119221	0.6916915	0.6916915	0.6916915

```
> colon.pldm.1 <- survCOND(survCS(time1, event1, Stime, event) ~ 1, x = 365,
+ data = colonCS, method = "LDM", presmooth = TRUE)
> summary(colon.pldm.1, times = 365 * 1:7)
```

$y$	estimate	lower	95% CI	upper	95% CI
365	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000
730	0.9429609	0.9236513	0.9590418	0.9590418	0.9590418
1095	0.8624778	0.8373879	0.8844013	0.8844013	0.8844013
1460	0.7788757	0.7430835	0.8137728	0.8137728	0.8137728
1825	0.7411599	0.7046557	0.7710392	0.7710392	0.7710392
2190	0.6795849	0.6377276	0.7118881	0.7118881	0.7118881
2555	0.6467549	0.6028921	0.6821533	0.6821533	0.6821533

In addition, one may also be interested in calculating the conditional survival function,  $S(y | x) = P(T > y | T_1 \leq x)$ . This is the probability of the individual to be alive at time  $y$  conditional that he/she is alive with recurrence at a previous time  $x$ . This quantity can also be estimated using function `survCOND` by considering the argument `lower.tail = TRUE`:

```
> colon.ldm.2 <- survCOND(survCS(time1, event1, Stime, event) ~ 1, x = 365,
+ data = colonCS, method = "LDM", lower.tail = TRUE)
> summary(colon.ldm.2, times=c(90, 180, 365, 730, 1095, 1460, 1825))
```

$y$	estimate	lower	95% CI	upper	95% CI
90	0.96956522	0.94541818	0.99122998	0.99122998	0.99122998
180	0.89565217	0.85836820	0.93278055	0.93278055	0.93278055
365	0.66086957	0.60452616	0.73021864	0.73021864	0.73021864

```

730 0.25652174 0.20552495 0.30834239
1095 0.10434783 0.07551852 0.14219702
1460 0.06956522 0.04000000 0.10000000
1825 0.06086957 0.03553539 0.09006894

```

It is worth mentioning that, given  $x$ , `lower.tail = TRUE` provides the survival estimates conditional to  $T_1 \leq x$  whereas `lower.tail = FALSE` provides the survival estimates conditional to  $T_1 > x$ . It should be noted that conditioning on  $T_1 > x$  is the default behavior of `survCOND`.

The package also provides plots for all methods. The following input commands (shown below) provide the plots for the conditional survival function  $P(T > y|T_1 > x)$  along  $y \geq x$  where  $x$  is a predefined fixed value. The corresponding plots for the two landmark methods (LDM and PLDM) are shown in Figure 1. The plots were obtained for fixed values  $x$  equal to 365 and 1095 days, along time  $y$ . This figure allows for an inspection along time of the survival probability (i.e., of being alive with or without recurrence) for the individuals who are disease free 1 and 3 years after surgery. All curves are monotonously decreasing. It is also evident that the conditional survival probabilities are smaller for lower  $x$  values. This feature was expected since the survival time increases with an increase in the recurrence-free survival. Results also suggest that individuals with higher recurrence times are most likely to survive from their disease.

To illustrate the usage of the graphical parameter arguments of function `plot.survCS`, plots shown in the first row in Figure 1 were obtained using arguments `col`, `confcol`, `xlab`, `ylab` and `ylim`. Plots shown on the second row were obtained using the default values. For more details about the graphical parameter arguments, see the corresponding help file.

```

> colon.ldm.1 <- survCOND(survCS(time1, event1, Stime, event) ~ 1, x = 365,
+   data = colonCS, method = "LDM")
> plot(colon.ldm.1, col = 1, confcol = 2, xlab = "Time (days)", ylab = "S(y|365)",
+   ylim = c(0.3, 1))
> colon.pldm.1 <- survCOND(survCS(time1, event1, Stime, event) ~ 1, x = 365,
+   data = colonCS, method = "LDM", presmooth = TRUE)
> plot(colon.ldm.1, col = 1, confcol = 2, xlab = "Time (days)", ylab = "S(y|365)",
+   ylim = c(0.3, 1))
> colon.ldm.2 <- survCOND(survCS(time1, event1, Stime, event) ~ 1, x = 1095,
+   data = colonCS, method = "LDM")
> plot(colon.ldm.1)
> colon.pldm.2 <- survCOND(survCS(time1, event1, Stime, event) ~ 1, x = 1095,
+   data = colonCS, method = "LDM", presmooth = TRUE)
> plot(colon.ldm.1)

```

When comparing the results obtained through the two methods (LDM and PLDM), it is seen that the semiparametric estimator PLDM has less variability with more jump points, specially at the right tail. It can also be seen that the semiparametric estimator takes higher values at the right tail.

One important goal is to obtain estimates for the above estimated quantities (conditional survival probabilities) conditionally on current or past covariate measures. The current version of the package allows the inclusion of a single covariate. Below we illustrate its usage using two qualitative covariates `rx` (treatment: Obs(ervation), Lev(amisole), Lev(amisole)+5FU), sex (1 – male) and a continuous covariate `age` (in years). The following input commands provide the estimates of the conditional survival  $S(y | x) = P(T > y|T_1 > x)$  for the three treatment groups by including the covariate (`rx`) in the right hand side of the formula argument.

```

> colon.rx.ldm <- survCOND(survCS(time1, event1, Stime, event) ~ rx, x = 365,
+   data = colonCS, method = "LDM")
> summary(colon.rx.ldm, times = 365 * 1:6)

```

```

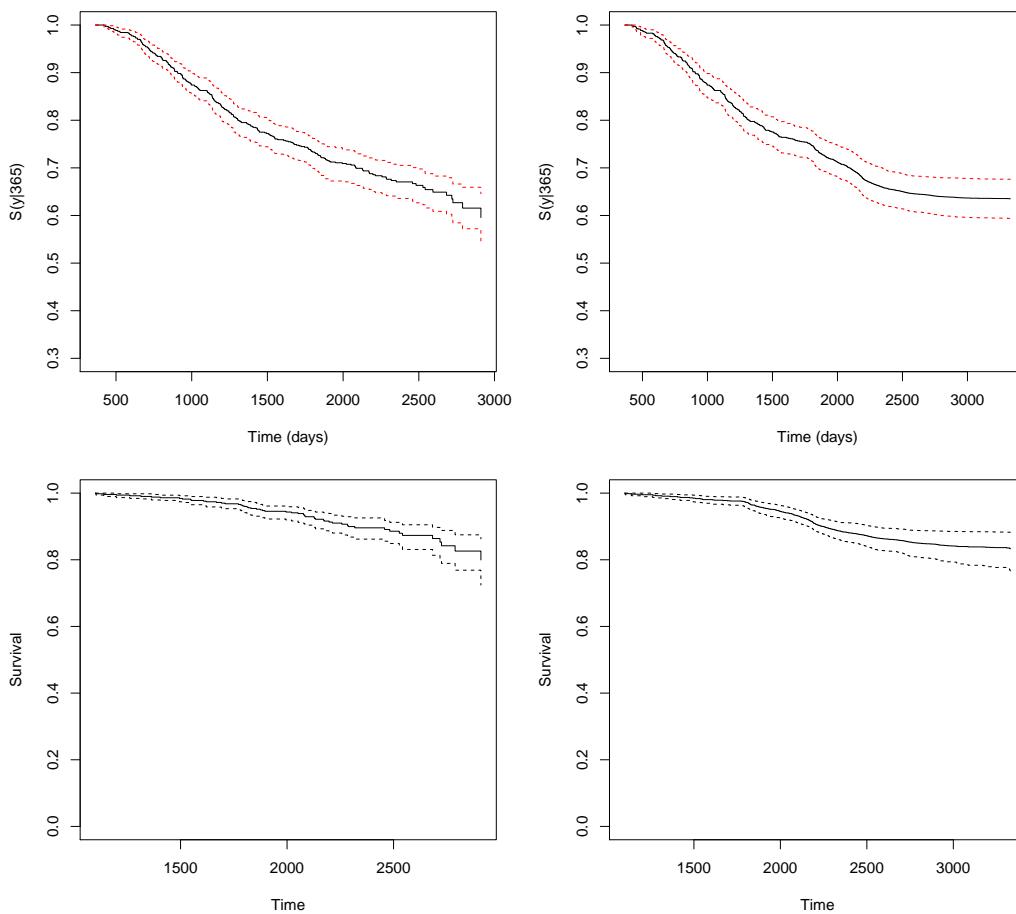
rx = Obs
y estimate lower 95% CI upper 95% CI
365 1.0000000 1.0000000 1.0000000
730 0.9469212 0.9095018 0.9780613
1095 0.8672736 0.8247419 0.9131253
1460 0.7655017 0.7187629 0.8095914
1825 0.7123480 0.6608400 0.7638970
2190 0.6562687 0.6038035 0.7170123

```

```

rx = Lev
y estimate lower 95% CI upper 95% CI
365 1.0000000 1.0000000 1.0000000

```



**Figure 1:** Estimation of the conditional survival function given that the subject is alive and disease-free at  $x = 365$  (top) and  $x = 1095$  (bottom row) days. Landmark estimators at the left and presmoothed landmark estimator on the right hand side. Colon cancer data.

```

730 0.9411765    0.9070484    0.9695116
1095 0.8280543    0.7757624    0.8773140
1460 0.7375566    0.6787002    0.7929553
1825 0.7102667    0.6541314    0.7709486
2190 0.6704293    0.6101410    0.7309958

```

```

rx = Lev+5FU
y estimate lower 95% CI upper 95% CI
365 1.0000000   1.0000000   1.0000000
730 0.9442231   0.9142632   0.9716684
1095 0.8884462   0.8431373   0.9243251
1460 0.8165244   0.7672650   0.8666731
1825 0.7639544   0.7092081   0.8144077
2190 0.7314409   0.6709950   0.7813303

```

Results obtained for the three treatment groups reveal that the combined treatment of levamisole plus fluorouracil have a benefit on overall survival. This is confirmed by the plot shown in Figure 2 which can be obtained using the following input command:

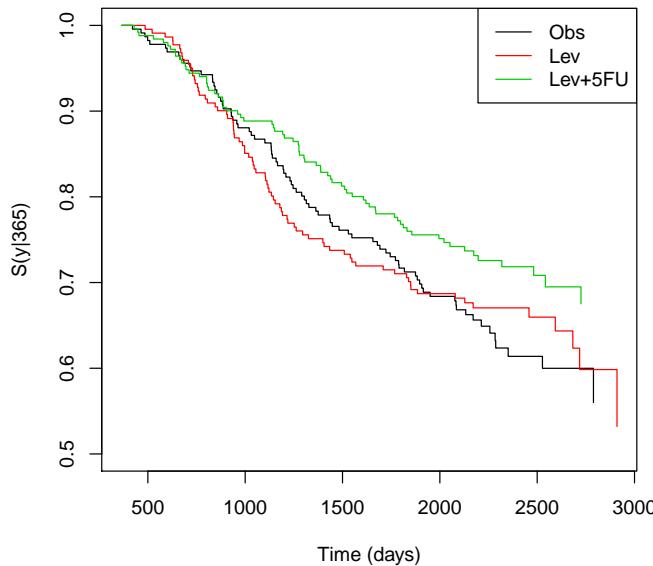
```
> plot(colon.rx.ldm, xlab = "Time (days)", ylab = "S(y|365)", conf = FALSE)
```

Similarly, one can obtain the corresponding survival probabilities  $S(y | x) = P(T > y | T_1 \leq x)$  for both genders (1 – male). Since this variable in the data.frame colonCS is of class "integer" it must be included in the formula using function factor.

```

> colon.sex.ldm <- survCOND(survCS(time1, event1, Stime, event) ~ factor(sex), x = 365,
+ data = colonCS, method = "LDM")
> summary(colon.sex.ldm, times = 365 * 1:6)

```



**Figure 2:** Estimates of the conditional survival function for the three treatment groups. Colon cancer data.

```

factor(sex) = 0
y estimate lower 95% CI upper 95% CI
365 1.0000000 1.0000000 1.0000000
730 0.9569231 0.9370979 0.9792300
1095 0.8769231 0.8417722 0.9172158
1460 0.7876565 0.7424245 0.8314511
1825 0.7475015 0.7084168 0.7982421
2190 0.6940773 0.6518322 0.7563382

factor(sex) = 1
y estimate lower 95% CI upper 95% CI
365 1.0000000 1.0000000 1.0000000
730 0.9329893 0.9037317 0.9615385
1095 0.8498782 0.8109783 0.8846094
1460 0.7639861 0.7247797 0.8053898
1825 0.7152471 0.6742324 0.7612994
2190 0.6822945 0.6336874 0.7277440

```

The **condSURV** package also allows the user to estimate the conditional survival given a continuous covariate (i.e., objects of class "integer" or "numeric"). For example, estimates and plots for the conditional survival for individuals aged 48 years,  $S(y|x, Z = z) = P(T > y | T_1 > x, \text{age} = 48)$ . This can be obtained using the following input commands:

```

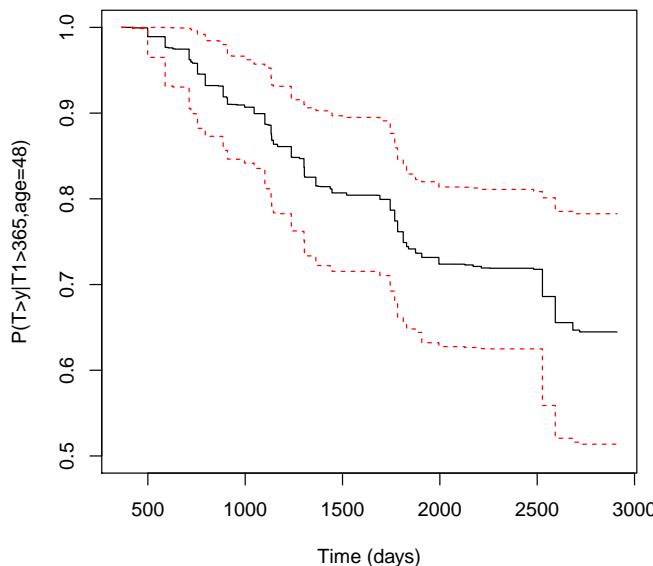
> colon.ipcw.age <- survCOND(survCS(time1, event1, Stime, event) ~ age, x = 365,
+ + z.value = 48, data = colonCS, lower.tail = FALSE)
> summary(colon.ipcw.age, times = 365 * 1:7)

y estimate lower 95% CI upper 95% CI
365 1.0000000 1.0000000 1.0000000
730 0.9582900 0.8993620 0.9960546
1095 0.8994077 0.8354449 0.9570992
1460 0.8069071 0.7154049 0.8968507
1825 0.7490154 0.6531423 0.8387582
2190 0.7211058 0.6265042 0.8126480
2555 0.6860070 0.5588995 0.8012140

> plot(colon.ipcw.age, col = 1, confcol = 2, xlab = "Time (days)",
+ + ylab = "P(T>y|T1>365,age=48)", ylim = c(0.5, 1))

```

The plot shown in Figure 3 depicts the conditional survival estimates taking into account the influ-



**Figure 3:** Estimates of the conditional survival function given that the subject is alive and disease-free at  $x = 365$  days given the continuous covariate age is equal to 48 years old. 95% pointwise confidence bands based on the percentile bootstrap. Colon cancer data.

ence of the covariate age together with the 95% pointwise confidence bands based on the percentile bootstrap which resamples each datum with probability  $1/n$ . The methods for implementing the conditional survival function conditionally on current or past covariate measures can be computationally demanding. In particular, the use of bootstrap resampling techniques are time-consuming processes because it is necessary to estimate the model a great number of times. The CPU time needed for running the input command required to obtain the plot shown in Figure 3 can take a few minutes. In such cases we recommend the use of parallelization (`cluster = TRUE`). This allows to run those repeated operations (for example, the estimation of the conditional probability in each of the bootstrap replicates) on multiple processors/cores on your computer, or on multiple nodes of a cluster. Thus, we can reduce the execution time in the construction of the bootstrap-based confidence interval.

The use of the `condSURV` package to more than two consecutive events is illustrated in the [Appendix](#).

## Conclusions

This paper discusses the implementation in R of some newly developed methods for the estimation of the conditional survival function. The `condSURV` package implements nonparametric and semi-parametric estimators for these quantities. The package also introduces and implements feasible estimation methods for these quantities conditionally on current or past covariate measures. Other related estimators are also implemented in the package. One of these estimators is the Kaplan-Meier estimator typically assumed to estimate the survival function. A modification of the Kaplan-Meier estimator based on a preliminary estimation (presmoothing) of the censoring probability for the survival time, given the available information is also implemented.

Software for multi-state survival analysis has been developed recently. These models deal with problems that are similar to those implemented in package `condSURV`. Among other quantities these packages deal with the estimation of the transition probabilities. It can be shown that in the progressive model with three states the conditional survival function  $P(T_2 > y | T_1 > x)$  can be expressed as the sum of two transition probabilities,  $p_{11}(x, y) + p_{12}(x, y)$ . However, for more than three states no formal relation can be established between the two quantities. To the best of our knowledge none of the available software packages can be used to estimate conditional survival probabilities such as:  $P(T_2 > y | T_1 > x)$ ,  $P(T_3 > y | T_1 < x_1, T_2 > x_2)$  or  $P(T_3 > y | T_1 > x_1, T_2 > x_2)$  where  $T_1$ ,  $T_2$  and  $T_3$  are ordered event times of successive events.

We mention some important topics that we shall consider in future versions of the package. One important issue is about the extension of the proposed methods for interval censoring. Another topic of much practical interest is to establish a more formal relation between our software and the `survival` package.

The results in this paper were obtained using R 3.2.5. The **condSURV** package is available from the Comprehensive R Archive Network at <https://CRAN.R-project.org/package=condSURV/>.

## Acknowledgments

This research was financed by Portuguese Funds through FCT - “Fundação para a Ciência e a Tecnologia”, within Project UID/MAT/00013/2013 and by research grant SFRH/BPD/93928/2013. We thank the reviewers for their constructive comments.

## Bibliography

- M. G. Akritas. Nearest neighbor estimation of a bivariate distribution under random censoring. *The Annals of Statistics*, 22:1299–1327, 1994. doi: 10.1214/aos/1176325630. [p464, 467]
- A. Allignol and A. Latouche. Cran task view: Survival analysis, 2016. URL <https://CRAN.R-project.org/view=Survival>. Version 2016-01-27. [p464]
- R. Beran. Nonparametric regression with randomly censored survival data. Technical report, University of California, Berkeley, 1981. [p464, 467]
- D. P. Byar. Veterans administration study of chemoprophylaxis for recurrent stage i bladder tumors: Comparisons of placebo, pyridoxine and topical thioptera. *Bladder Tumors and Other Topics in Urological Oncology*, 18:363–370, 1980. doi: 10.1007/978-1-4613-3030-1\_74. [p476]
- R. J. Cook and J. F. Lawless. *The Analysis of Recurrent Event Data*. Springer-Verlag, New York, 2007. [p464]
- D. Dabrowska. Non-parametric regression with censored survival data. *Scandinavian Journal of Statistics*, 14:181–197, 1987. doi: 10.1002/9781118558072.ch5. [p464, 467]
- A. C. Davison and D. V. Hinkley. *Bootstrap Methods and Their Application*. Cambridge University Press, New York, 1997. [p469]
- G. Dikta. On semiparametric random censorship models. *Journal of Statistical Planning and Inference*, 66:253–279, 1998. doi: 10.1016/s0378-3758(97)00091-8. [p466]
- T. A. Gerds. *prodlim: Product-Limit Estimation for Censored Event History Analysis*, 2015. URL <https://CRAN.R-project.org/package=prodlim>. R package version 1.5.7. [p464]
- W. Gonzalez-Manteiga and C. Cadarso-Suárez. Asymptotic properties of a generalized Kaplan-Meier estimator with some applications. *Communications in Statistics – Theory and Methods*, 4(1):65–78, 1994. doi: 10.1080/10485259408832601. [p464, 467]
- E. L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53:457–481, 1958. doi: 10.2307/2281868. [p464]
- L. Meira-Machado, J. de Uña-Álvarez, C. Cadarso-Suárez, and P. K. Andersen. Multi-state models for the analysis of time to event data. *Statistical Methods in Medical Research*, 18:195–222, 2009. doi: 10.1177/0962280208092301. [p464]
- L. Meira-Machado, M. Sestelo, and A. Goncalves. Nonparametric estimation of the survival function for ordered multivariate failure time data: A comparative study. *Biometrical Journal*, 58(3):623–634, 2016. doi: 10.1002/bimj.201500038. [p464, 466]
- C. G. Moertel, T. R. Fleming, J. S. McDonald, et al. Levamisole and fluorouracil for adjuvant therapy of resected colon carcinoma. *New England Journal of Medicine*, 322:352–358, 1990. doi: 10.1007/s11725-008-0076-x. [p468]
- E. A. Nadaraya. On nonparametric estimates of density functions and regression curves. *Theory of Applied Probability*, 10:186–190, 1965. doi: 10.1137/1110024. [p467]
- G. A. Satten and S. Datta. The Kaplan-Meier estimator as an inverse-probability-of-censoring weighted average. *The American Statistician*, 55(3):207–210, 2001. doi: 10.1198/000313001317098185. [p467]
- T. M. Therneau. *survival: A Package for Survival Analysis in S*, 2015. URL <https://CRAN.R-project.org/package=survival>. version 2.38. [p464]

- H. C. Van Houwelingen. Dynamic prediction by landmarking in event history analysis. *Scandinavian Journal of Statistics*, 34:70–85, 2007. doi: 10.1111/j.1467-9469.2006.00529.x. [p466]
- I. Van Keilegom, M. Akritas, and N. Veraverbeke. Estimation of the conditional distribution in regression with censored data: A comparative study. *Computational Statistics & Data Analysis*, 35: 487–500, 2001. doi: 10.1016/s0167-9473(00)00025-6. [p464, 467]
- G. S. Watson. Smooth regression analysis. *Sankhya*, 26(15):175–184, 1964. doi: 10.1093/biomet/51.1-2.175. [p467]

## Appendix

To illustrate the use of the **condSURV** package to more than two event times we use data from a bladder cancer study (Byar, 1980) conducted by the Veterans Administration Cooperative Urological Research Group. In this study, patients had superficial bladder tumors that were removed by transurethral resection. Many patients had multiple recurrences of tumors during the study, and new tumors were removed at each visit. For illustration purposes we use data from 85 individuals in the placebo and thiotepa treatment groups.

Here, only the first three recurrence times (in months) and the corresponding event times,  $T_1$ ,  $T_2$  and  $T_3$ , are considered. From the total of 85 patients, 47 relapsed at least once and, among these, 29 experienced a new recurrence and 22 individuals had a third recurrence.

Below we illustrate how to obtain estimates for the conditional survival function  $P(T_3 > y | T_1 \leq x_1, T_2 > x_2)$ . First we need to built a formula object using the `survCS` function as the response. The three event times and their corresponding indicator statuses have to be specified in this function. Then, the conditional survival function can be estimated using function `survCOND` by considering the argument `lower.tail = c(TRUE, FALSE)`. Below we show the corresponding input commands for the landmark method (LDM) and for its presmoothed version.

```
> bladder.ldm <- survCOND(survCS(t1, e1, t2, e2, t3, e3) ~ 1, x = c(8, 12),
+   lower.tail = c(TRUE, FALSE), data = bladderCS, method = "LDM")
> summary(bladder.ldm)

P(T>y|T1<=8,T2>12)

y estimate lower 95% CI upper 95% CI
12 1.0000000 1.0000000 1.0000000
19 0.9444444 0.8174641 1.0000000
22 0.8854167 0.7141053 1.0000000
23 0.7083333 0.4991667 0.9230769
24 0.6493056 0.4117563 0.8461538
25 0.5902778 0.3656623 0.8000893
46 0.3935185 0.0000000 0.7692308
47 0.0000000 0.0000000 0.6675000

> bladder.pldm <- survCOND(survCS(t1, e1, t2, e2, t3, e3) ~ 1, x = c(8, 12),
+   lower.tail = c(TRUE, FALSE), data = bladderCS, method = "LDM", presmooth = TRUE)
> summary(bladder.pldm)

P(T>y|T1<=8,T2>12)

y estimate lower 95% CI upper 95% CI
12 1.0000000 1.0000000 1.0000000
14 0.9734802 0.92306857 1.0000000
18 0.9473931 0.89833894 1.0000000
19 0.8935818 0.80656969 0.9932548
22 0.8663338 0.74760747 0.9691555
23 0.7795092 0.57123322 0.9414341
24 0.7485065 0.49188561 0.9316808
25 0.7163915 0.43525931 0.9189147
26 0.6475479 0.32652371 0.8674366
27 0.6102029 0.32640074 0.8358788
29 0.5709044 0.32624995 0.8159263
43 0.4475656 0.23840137 0.6916276
```

46	0.3217597	0.12021360	0.6010603
47	0.2062917	0.03004109	0.6010603

When comparing the unsmoothed estimator with the semiparametric presmoothed estimator it can be seen that the later has less variability with more jump points, specially at the right tail. Differences obtained for the estimates using the two methods are explained by the small sample size and the high censoring percentage. For such cases we recommend the use of the presmoothed estimator.

It is worth mentioning that the **condSURV** package can be used to estimate other quantities involving all possible combinations in the argument `lower.tail`. For example:  $P(T > y | T_1 > 6, T_2 \leq 32)$  or  $P(T > y | T_1 \leq 12, T_2 \leq 32)$ .

*Luis Meira-Machado*

*Centre of Mathematics and Department of Mathematics and Applications*

*University of Minho*

*Portugal*

[lmachado@math.uminho.pt](mailto:lmachado@math.uminho.pt)

*Marta Sestelo*

*Centre of Mathematics, University of Minho, Portugal*

*SiDOR Research Group and CINBIO, University of Vigo, Spain*

[sestelo@uvigo.es](mailto:sestelo@uvigo.es)

# ggfortify: Unified Interface to Visualize Statistical Results of Popular R Packages

by Yuan Tang, Masaaki Horikoshi, and Wenxuan Li

**Abstract** The **ggfortify** package provides a unified interface that enables users to use one line of code to visualize statistical results of many R packages using **ggplot2** idioms. With the help of **ggfortify**, statisticians, data scientists, and researchers can avoid the sometimes repetitive work of using the **ggplot2** syntax to achieve what they need.

## Background

R users have many plotting options to choose from, such as base graphics, grid graphics, and **lattice** graphics (Sarkar, 2008). Each has their own unique customization and extensibility options. In recent years, **ggplot2** has emerged as a popular choice for creating visualizations (Wickham, 2009) and provides a strong programming model based on a “grammar of graphics” which enables methodical production of virtually any kind of statistical chart. The **ggplot2** package makes it possible to describe a wide range of graphics with succinct syntax and independent components and is based on an object-oriented model that also makes it modular and extensible. It has become a widely used framework for producing statistical graphics in R.

The distinct syntax of **ggplot2** makes it a definite paradigm shift from base and **lattice** graphics and presents a somewhat steep learning curve for those used to existing R charting idioms. Often times users only want to quickly visualize some statistical results from key R packages, especially those focusing on clustering and time series analysis. Many of these packages provide default **base plot()** visualizations for the data and models they generate. These components require transformation before using them in **ggplot2** and each of those transformation steps must be replicated by others when they wish to produce similar charts in their analyses. Creating a central repository for common/popular transformations and default plotting idioms would reduce the amount of effort needed by all to create compelling, consistent and informative charts. To achieve this, we provide a unified **ggplot2** plotting interface to many statistics and machine-learning packages and functions in order to help these users achieve reproducibility goals with minimal effort.

The **ggfortify** (Horikoshi and Tang, 2015) package has a very easy-to-use and uniform programming interface that enables users to use one line of code to visualize statistical results of many popular R packages using **ggplot2** as a foundation. This helps statisticians, data scientists, and researchers avoid both repetitive work and the need to identify the correct **ggplot2** syntax to achieve what they need. With **ggfortify**, users are able to generate beautiful visualizations of their statistical results produced by popular packages with minimal effort.

## Software architecture

There are many ways to extend the functionality of **ggplot2**. One straightforward way is through the use of S3 generic functions<sup>1</sup>. Specifically, it is possible to provide custom functions for:

- **autoplot()**, which enables plotting a custom object with **ggplot2**, and
- **fortify()**, which enables converting a custom object to a tidy “**data.frame**”

The **ggfortify** package uses this extensibility to provide default **ggplot2** visualizations and data transformations.

To illustrate this, we consider the implementation for **fortify.prcomp()** and **autoplot.pca\_common()** used as a basis of other PCA related implementations:

```
fortify.prcomp <- function(model, data = NULL, ...) {
  if (is(model, "prcomp")) {
    d <- as.data.frame(model$x)
```

<sup>1</sup><http://adv-r.had.co.nz/S3.html>

```

    values <- model$x %*% t(model$rotation)
} else if (is(model, "princomp")) {
  d <- as.data.frame(model$scores)
  values <- model$scores %*% t(model$loadings[,])
} else {
  stop(paste0("Unsupported class for fortify.pca_common: ", class(model)))
}

values <- ggfortify::unscale(values, center = model$center,
                             scale = model$scale)
values <- cbind_wraps(data, values)
d <- cbind_wraps(values, d)
post_fortify(d)
}

```

This S3 function recognizes “prcomp” objects and will extract the necessary components from them such as the matrix whose columns contain the eigenvectors in “rotation” and rotated data in “x”, which can be drawn using `autoplot()` later on. The `if()` call is used here to handle different objects that are of essentially the same principal components family since they can be handled in the exactly same way once the necessary components are extracted from `ggfortify`.

The following `autoplot.pca_common()` function first calls `fortify()` to perform the component extraction for different PCA-related objects, then performs some common data preparation for those objects, and finally calls `ggbiplot()` internally to handle the actual plotting.

```

autoplot.pca_common <- function(object, data = NULL,
                                 scale = 1.0, ...) {

  plot.data <- ggplot2::fortify(object, data = data)
  plot.data$rownames <- rownames(plot.data)

  if (is_derived_from(object, "prcomp")) {
    x.column <- "PC1"
    y.column <- "PC2"
    loadings.column <- "rotation"

    lam <- object$sdev[1L:2L]
    lam <- lam * sqrt(nrow(plot.data))

  } else if (is_derived_from(object, "princomp")) {
    ...
  } else {
    stop(paste0("Unsupported class for autoplot.pca_common: ", class(object)))
  }

  # common and additional preparation before plotting
  ...

  p <- ggbiplot(plot.data = plot.data,
                loadings.data = loadings.data, ...)
  return(p)
}

```

Once `ggfortify` is loaded, users have instant access to 38 pre-defined `autoplot()` functions and 36 pre-defined `fortify()` functions, enabling them to immediately `autoplot()` numerous types of objects or pass those objects directly to `ggplot2` for manual customization. Furthermore, `ggfortify` is highly extensible and customizable and provides utility functions that make it easy for users to define `autoplot()` and `fortify()` methods for their own custom objects.

To present a streamlined API, `ggfortify` groups common implementations for various object-types, including:

- Time-series
- Principal components analysis (PCA), including clustering and multi-dimensional scaling (MDS)

**Table 1:** Supported packages

package	supported types	package	supported types
<b>base</b>	"matrix", "table"	<b>sp</b>	"SpatialPoints", "SpatialPolygons", "Line", "Lines", "Polygon", "Polygons", "SpatialLines", "SpatialLinesDataFrame", "SpatialPointsDataFrame", "SpatialPolygonsDataFrame"
<b>cluster</b>	"clara", "fanny", "pam"	<b>stats</b>	"HoltWinters", "lm", "acf", "ar", "Arima", "stepfun", "stl", "ts", "cmdscale", "decomposed.ts", "density", "factanal", "glm", "kmeans", "princomp", "spec" "survfit", "survfit.cox"
<b>changepoint</b>	"cpt"	<b>survival</b>	"breakpoints", "breakpointsfull"
<b>dlm</b>	"dlmFilter", "dlmSmooth"	<b>strucchange</b>	"timeSeries"
<b>fGarch</b>	"fGARCH"	<b>timeSeries</b>	"irts"
<b>forecast</b>	"bats", "forecast", "ets", "nnetar"	<b>tseries</b>	
<b>fracdiff</b>	"fracdiff"	<b>vars</b>	"varprd"
<b>glmnet</b>	"cv.glmnet", "glmnet"	<b>xts</b>	"xts"
<b>KFAS</b>	"KFS", "signal"	<b>zoo</b>	"zooreg"
<b>lfda</b>	"lfda", "klfda", "self"	<b>MASS</b>	"isoMDS", "sammon"
<b>maps</b>	"map"		

- 1d/2d kernel density estimation (KDE)
- Survival analysis
- Cartography

A list of currently supported packages and classes can be found in Table 1. Additional packages that are in development are not shown here but more than 50 object types are supported by **ggfortify**. Feedback is being collected from users<sup>2</sup> for possible bug fixes and future enhancements.

## Illustrations

As previously stated, **ggfortify** provides methods that enable **ggplot2** to work with objects in different classes from different R packages. The following subsections illustrate how to use **ggfortify** to plot results from several of these packages.

### Principal components analysis

The **ggfortify** package defines both `fortify()` and `autoplot()` methods for the two core PCA functions in the **stats** package: `stats::prcomp()` and `stats::princomp()`. The values returned by either function can be passed directly to `ggplot2::autoplot()` as illustrated in the following code and in Figure 1. Note that users can also specify a column to be used for the colour aesthetic.

```
library(ggfortify)
df <- iris[c(1, 2, 3, 4)]
autoplot(prcomp(df), data = iris, colour = "Species")
```

If `label = TRUE` is specified, as shown in Figure 2, **ggfortify** will draw labels for each data point. Users can also specify the size of the labels via `label.size`. If `shape = FALSE` is specified, the shape of the data points will be removed, leaving only the labels on the plot.

```
autoplot(prcomp(df), data = iris, colour = "Species", shape = FALSE, label.size = 3)
```

<sup>2</sup><https://github.com/sinhrks/ggfortify/issues>

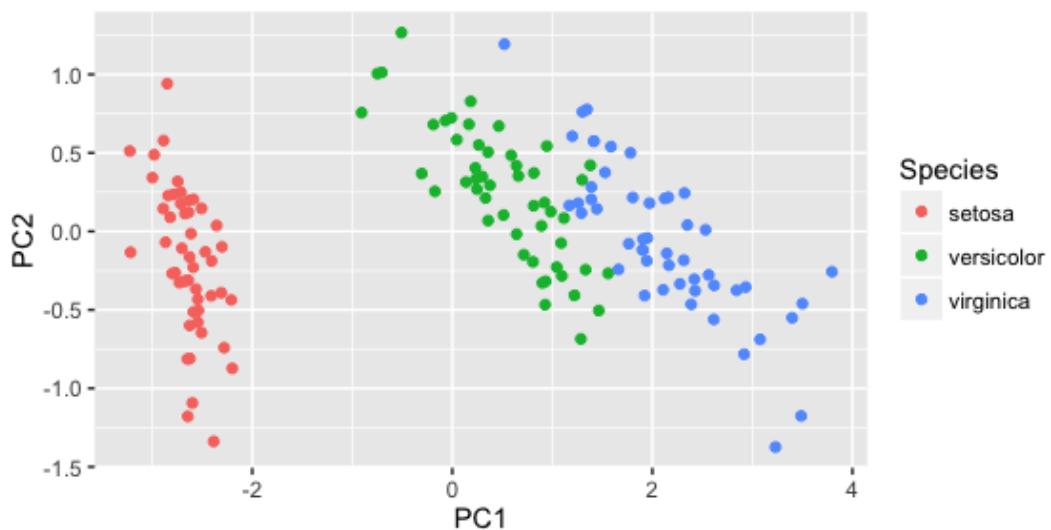


Figure 1: PCA with colors for each class.

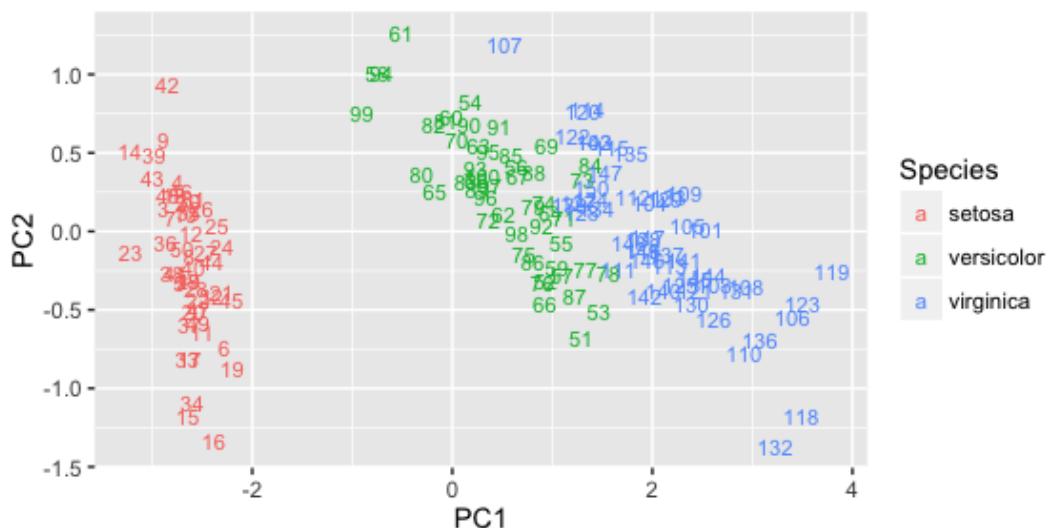


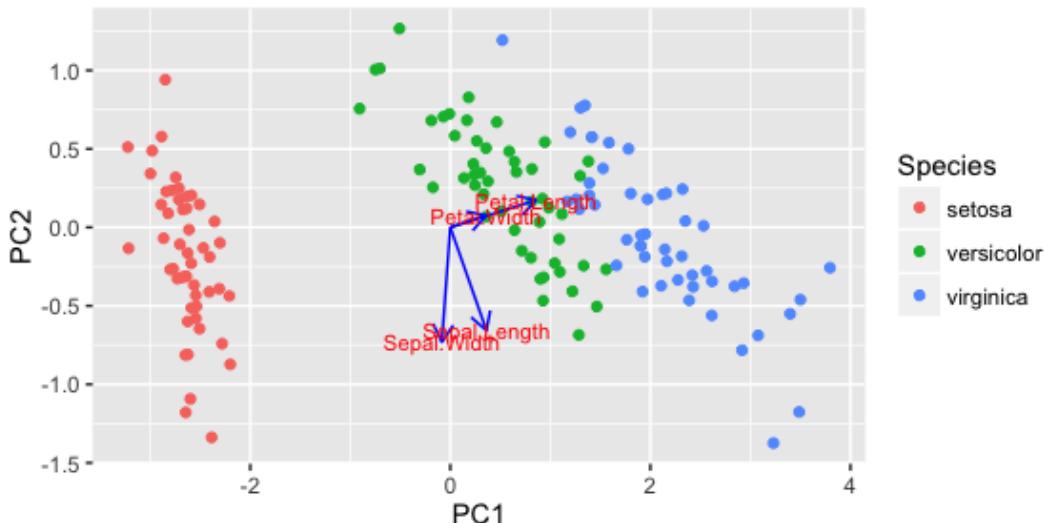
Figure 2: PCA with colors and labels for each class.

The autoplot function returns the constructed **ggplot2** object so users can apply additional **ggplot2** code to further enhance the plot. For example:

```
autoplot(prcomp(df), data=iris, colour = "Species", shape = FALSE, label.size = 3)
+ labs(title = "Principal Component Analysis")
```

Users can also specify loadings = TRUE to draw the PCA eigen-vectors. More aesthetic options such as size and colors of the eigen-vector labels can also be specified as shown in Figure 3 and the following code:

```
autoplot(prcomp(df), data = iris, colour = "Species",
loadings = TRUE, loadings.colour = 'blue',
loadings.label = TRUE, loadings.label.size = 3)
```



**Figure 3:** PCA with eigen-vectors and labels.

## Linear models

The **ggfortify** function is able to interpret `lm()` fitted model objects and allows the user to select the subset of desired plots through the `which` parameter (just like the `plot.lm()` function). The `ncol` and `nrow` parameters also allow users to specify the number of subplot columns and rows, as seen in Figure 4 and the following code:

```
par(mfrow = c(1, 2))
m <- lm(Petal.Width ~ Petal.Length, data = iris)
autoplot(m, which = 1:6, ncol = 3, label.size = 3)
```

Many plot aesthetics can be changed by using the appropriate named parameters. For example, the `colour` parameter is for coloring data points, the `smooth.colour` parameter is for coloring smoothing lines and the `ad.colour` parameter is for coloring the auxiliary lines, as demonstrated in Figure 5 and the following code:

```
autoplot(m, which = 1:6, colour = "dodgerblue3",
smooth.colour = "black", smooth.linetype = "dashed",
ad.colour = "blue",
label.size = 3, label.n = 5, label.colour = "blue",
ncol = 3)
```

## Clustering

The **ggfortify** package also supports various objects like "clara", "fanny", "pam", "kmeans", and "lfd", from the `cluster` (Maechler et al., 2015) and `lfd` (Tang and Deane-Mayer, 2016) packages. It automatically infers the object type and plots the results from those packages using **ggplot2** with a single function call. Users can specify `frame = TRUE` to easily draw the clustering boundaries as seen in Figure 6 and the following code:

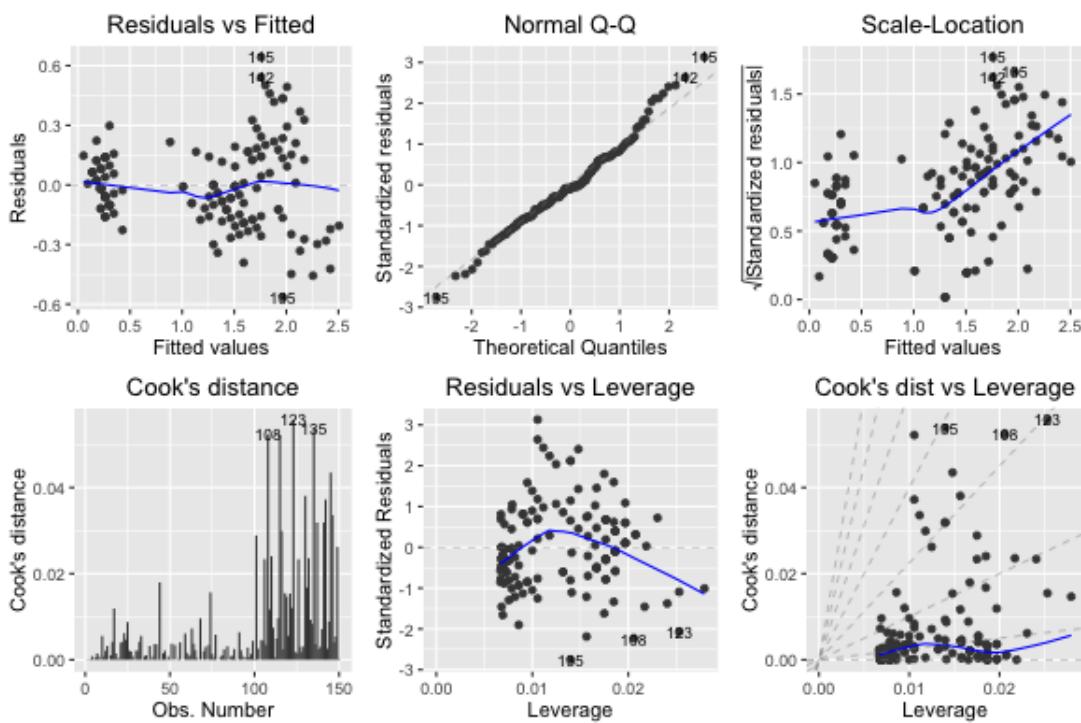


Figure 4: Linear model results.

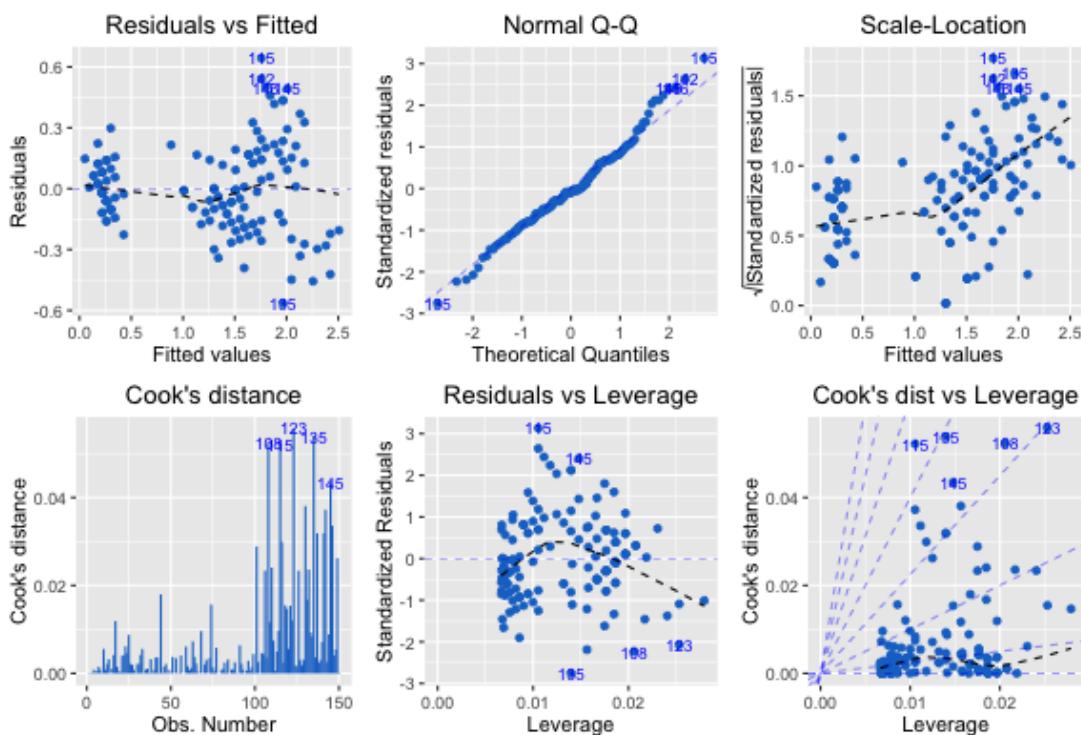
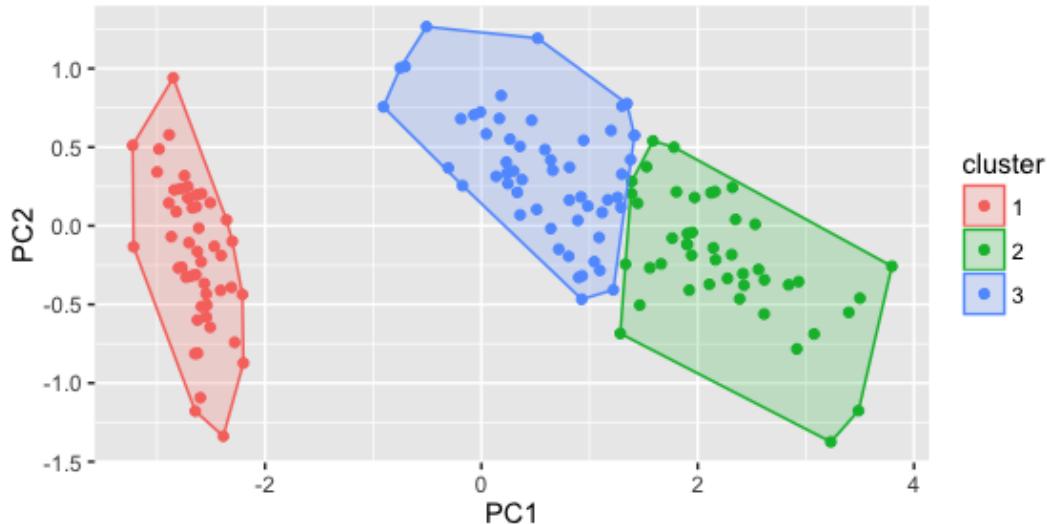


Figure 5: Linear model results with specified options.

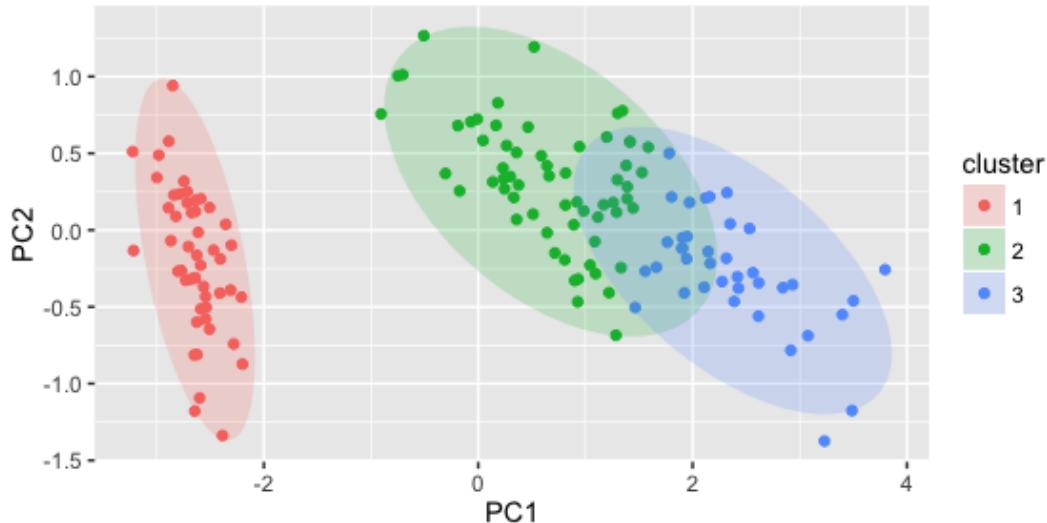
```
library(cluster)
autoplots(fanny(iris[-5], 3), frame = TRUE)
```



**Figure 6:** Clustering with boundaries.

As illustrated in Figure 7 with `frame.type = "norm"`, by specifying `frame.type` users are able to draw boundaries of different shapes. The different frame types can be found in `frame.type` option in `ggplot2::stat_ellipse()`.

```
autoplots(pam(iris[-5], 3), frame = TRUE, frame.type = "norm")
```



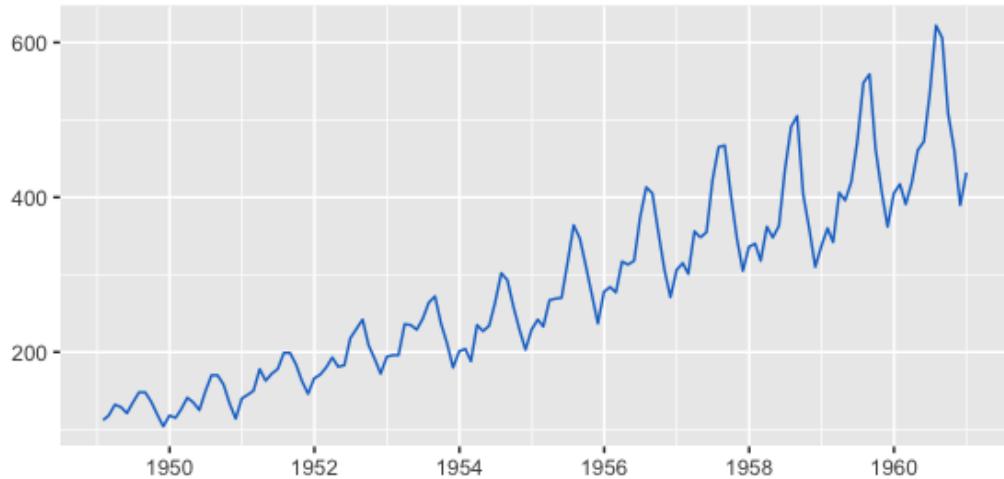
**Figure 7:** Clustering with boundaries in ellipse shape.

### Time series

The **ggfortify** package makes it much easier to visualize time series objects using **ggplot2** and provides `autoplots()` and `fortify()` implementations for objects from many time series libraries such as **zoo** (Zeileis and Grothendieck, 2005), **xts** (Ryan and Ulrich, 2014), and **timeSeries** (Team et al., 2015).

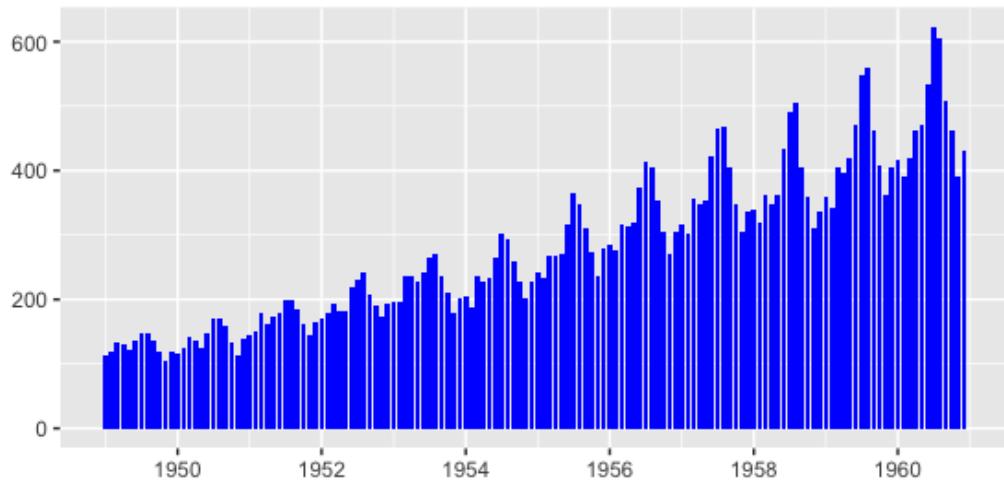
Here is an example of using **ggfortify** to plot the AirPassengers example time series data set from the **timeSeries** package, specifying color via `ts.colour`, geometric shape via `ts.geom` as seen in Figure 8, Figure 9, and Figure 10:

```
library(timeSeries)
autoplot(as.timeSeries(AirPassengers), ts.colour = "dodgerblue3")
```



**Figure 8:** AirPassengers time series.

```
autoplot(AirPassengers, ts.geom = "bar", fill = "blue")
```



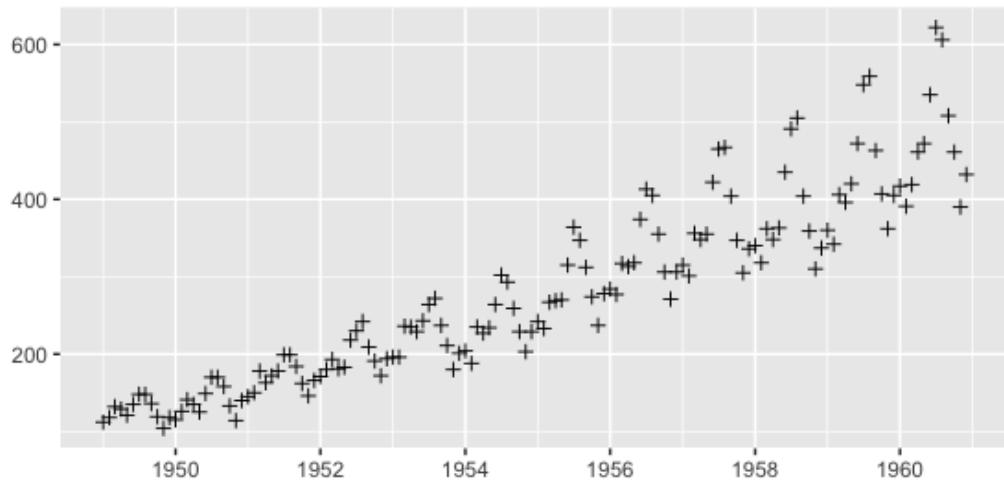
**Figure 9:** AirPassengers time series in bar shape.

```
autoplot(AirPassengers, ts.geom = "point", shape = 3)
```

## Forecasting

Forecasting packages such as **forecast** (Hyndman, 2015), **changepoint** (Killick et al., 2016), **strucchange** (Zeileis et al., 2002), and **dlm** (Petrис, 2010), are popular choices for statisticians and researchers. Predictions and statistical results from those packages can now be plotted automatically with **ggplot2** using the functions provided by **ggfortify**. Note that in these cases the order of loading packages matters. For example, since **forecast** has its own **autoplot()** function, if it is loaded before **ggfortify**, the **autoplot()** function in **forecast** will be used instead.

The **ggfortify** function automatically plots the original and smoothed line from Kalman filter function in the **dlm** package as shown in Figure 11 .

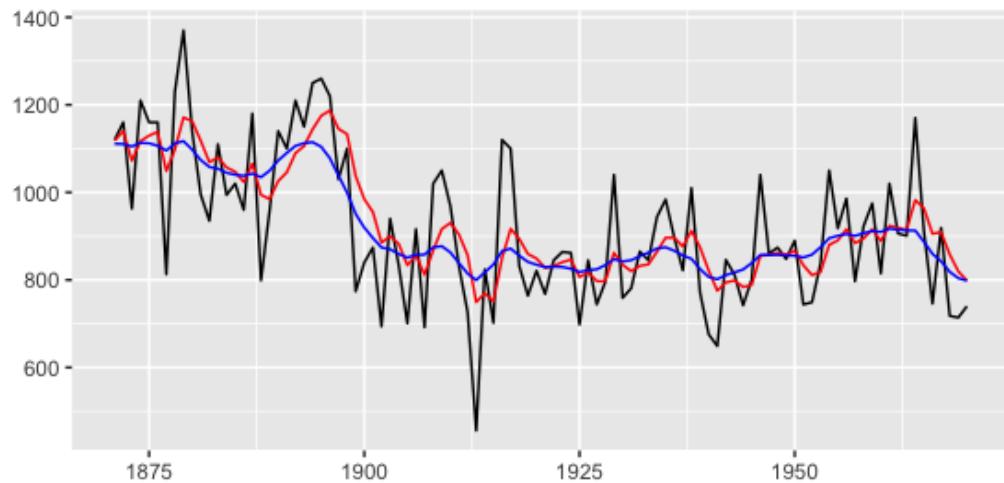


**Figure 10:** AirPassengers time series in point shape.

```
library(dlm)
form <- function(theta){
  dlmModPoly(order = 1, dV = exp(theta[1]), dW = exp(theta[2]))
}

model <- form(dlmMLE(Nile, parm = c(1, 1), form)$par)
filtered <- dlmFilter(Nile, model)

autoplot(filtered)
```



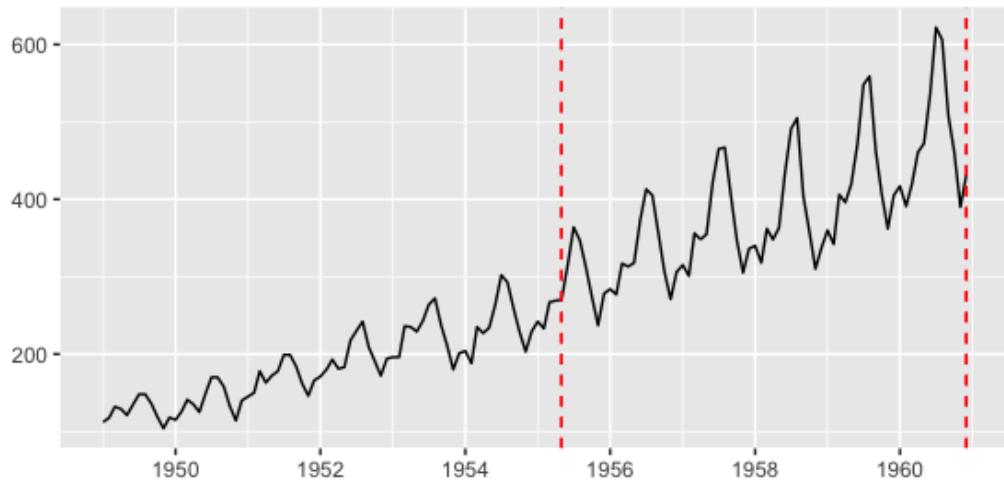
**Figure 11:** Smoothed time series by Kalman filter.

The **ggfortify** package automatically plots the change points with optimal positioning for the AirPassengers data set found in the **changepoint** package using the `cpt.meanvar()` function, shown in Figure 12 .

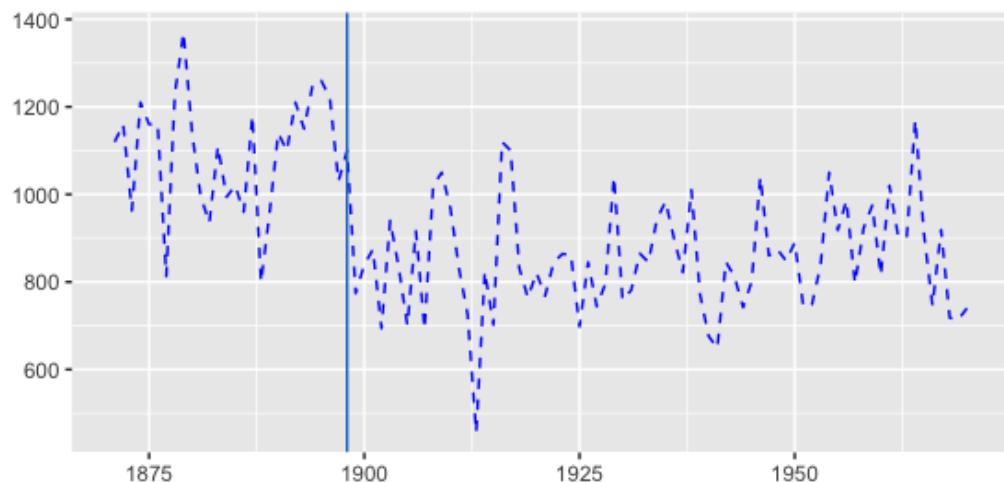
```
library(changepoint)
autoplot(cpt.meanvar(AirPassengers))
```

As well, **ggfortify** plots the optimal break points where possible structural changes happen in the regression models built by the `strucchange::breakpoints()`, shown in Figure 13.

```
library(strucchange)
autoplot(breakpoints(Nile ~ 1), ts.colour = "blue", ts.linetype = "dashed",
         cpt.colour = "dodgerblue3", cpt.linetype = "solid")
```



**Figure 12:** Change points with optimal positioning for AirPassengers.



**Figure 13:** Optimal break points with possible structural changes.

## Future development

We welcome suggestions and contributions from others. Providing default `autoplot()` and `fortify()` methods for additional R objects means researchers will spend less time focusing on `ggplot2` plotting details and more time on their work and research. We have provided a Github repository <https://github.com/sinhrks/ggfortify> where users can test out development versions of the package and provide feature requests, feedback and bug reports. We encourage you to submit your issues and pull requests to help us make this package better for the R community.

## Summary

The `ggfortify` package provides a very simple interface to streamline the process of plotting statistical results from many popular R packages. Users can spend more time and focus on their analyses instead of figuring out the details of how to visualize their results in `ggplot2`.

## Acknowledgement

We sincerely thank all developers for their efforts behind the packages that **ggfortify** depend on, namely, **dplyr** (Wickham and Francois, 2015), **tidyR** (Wickham, 2016b), **gridExtra** (Auguie, 2016), and **scales** (Wickham, 2016a).

## Bibliography

- B. Auguie. *gridExtra: Miscellaneous Functions for "Grid" Graphics*, 2016. URL <http://CRAN.R-project.org/package=gridExtra>. R package version 2.2.1. [p488]
- M. Horikoshi and Y. Tang. *ggfortify: Data Visualization Tools for Statistical Analysis Results*, 2015. URL <http://CRAN.R-project.org/package=ggfortify>. R package version 0.1.0. [p478]
- R. J. Hyndman. *forecast: Forecasting functions for time series and linear models*, 2015. URL <http://github.com/robjhyndman/forecast>. R package version 6.2. [p485]
- R. Killick, K. Haynes, and I. A. Eckley. *changepoint: An R package for changepoint analysis*, 2016. URL <http://CRAN.R-project.org/package=changepoint>. R package version 2.2.1. [p485]
- M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik. *cluster: Cluster Analysis Basics and Extensions*, 2015. R package version 2.0.3 — For new features, see the 'Changelog' file (in the package source). [p482]
- G. Petris. An R package for dynamic linear models. *Journal of Statistical Software*, 36(12):1–16, 2010. URL <http://www.jstatsoft.org/v36/i12/>. [p485]
- J. A. Ryan and J. M. Ulrich. *xts: eXtensible Time Series*, 2014. URL <http://CRAN.R-project.org/package=xts>. R package version 0.9-7. [p484]
- 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. [p478]
- Y. Tang and Z. Deane-Mayer. *lfda: Local Fisher Discriminant Analysis*, 2016. URL <https://github.com/terrytangyuan/lfda>. R package version 1.1.1. [p482]
- R. C. Team, D. Wuertz, T. Setz, and Y. Chalabi. *timeSeries: Rmetrics - Financial Time Series Objects*, 2015. URL <http://CRAN.R-project.org/package=timeSeries>. R package version 3022.101.2. [p484]
- H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer Science & Business Media, 2009. [p478]
- H. Wickham. *scales: Scale Functions for Visualization*, 2016a. URL <http://CRAN.R-project.org/package=scales>. R package version 0.4.0. [p488]
- H. Wickham. *tidyR: Easily Tidy Data with 'spread()' and 'gather()' Functions*, 2016b. URL <http://CRAN.R-project.org/package=tidyR>. R package version 0.4.1. [p488]
- H. Wickham and R. Francois. *dplyr: A Grammar of Data Manipulation*, 2015. URL <http://CRAN.R-project.org/package=dplyr>. R package version 0.4.3. [p488]
- 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/>. [p484]
- A. Zeileis, F. Leisch, K. Hornik, and C. Kleiber. *strucchange: An R package for testing for structural change in linear regression models*. *Journal of Statistical Software*, 7(2):1–38, 2002. URL <http://www.jstatsoft.org/v07/i02/>. [p485]

*Yuan Tang*  
*Uptake Technologies, Inc.*  
*600 West Chicago Ave, Chicago, IL 60654*  
*United States*  
[terrytangyuan@gmail.com](mailto:terrytangyuan@gmail.com)

*Masaaki Horikoshi*  
*Accenture Japan Ltd.*

*Akasaka Intercity 1-11-44 Akasaka Minato-ku, Tokyo  
Japan  
[sinhrks@gmail.com](mailto:sinhrks@gmail.com)*

*Wenxuan Li  
Department of Agricultural Economics, Purdue University  
403 W State Street, West Lafayette, IN, 47907  
United States  
[wenxuan.tess@gmail.com](mailto:wenxuan.tess@gmail.com)*

# Measurement units in R

Edzer Pebesma, Thomas Mailund, and James Hiebert

**Abstract** We briefly review SI units, and discuss R packages that deal with measurement units, their compatibility and conversion. Built upon `udunits2` and the UNIDATA udunits library, we introduce the package `units` that provides a class for maintaining unit metadata. When used in expression, it automatically converts units, and simplifies units of results when possible; in case of incompatible units, errors are raised. The class flexibly allows expansion beyond predefined units. Using `units` may eliminate a whole class of potential scientific programming mistakes. We discuss the potential and limitations of computing with explicit units.

## Introduction

Two quotes from Cobb and Moore (1997) – “*Data are not just numbers, they are numbers with a context*” and “*in data analysis, context provides meaning*” – illustrate that for a data analysis to be meaningful, knowledge of the data’s context is needed. Pragmatic aspects of this context include who collected or generated the data, how this was done, and for which purpose (Scheider et al., 2016); semantic aspects concern what the data represents: which aspect of the world do the data refer to, when and where were they measured, and what a value of ‘1’ means.

R does allow for keeping some context with data, for instance

- “`data.frame`” columns must have and “`list`” elements may have names that can be used to describe context, using freetext
- “`matrix`” or “`array`” objects may have `dimnames`
- for variables of class “`factor`” or “`ordered`”, levels may indicate, using freetext, the categories of nominal or ordinal variables
- “`POSIXt`” and “`Date`” objects specify how numbers should be interpreted as time or date, with fixed units (second and day, respectively) and origin (Jan 1, 1970, 00:00 UTC)
- “`difftime`” objects specify how time duration can be represented by numbers, with flexible units (secs, mins, hours, days, weeks); `lubridate` (Grolemund and Wickham, 2011) extends some of this functionality.

Furthermore, if spatial objects as defined in package `sp` (Pebesma and Bivand, 2005) have a proper coordinate reference system set, they can be transformed to other datums, or converted to various flat (projected) representations of the Earth (Iliffe and Lott, 2008).

In many cases however, R drops contextual information. As an example, we look at annual global land-ocean temperature index<sup>1</sup> since 1960:

```
> temp_data = subset(read.table("647_Global_Temperature_Data_File.txt",
+                                header=TRUE)[1:2], Year >= 1960)
> temp_data$date = as.Date(paste0(temp_data$Year, "-01-01"))
> temp_data$time = as.POSIXct(temp_data$date)
> Sys.setenv(TZ="UTC")
> head(temp_data, 3)
  Year Annual_Mean      date       time
81 1960      -0.03 1960-01-01 1960-01-01
82 1961       0.05 1961-01-01 1961-01-01
83 1962       0.02 1962-01-01 1962-01-01
> year_duration = diff(temp_data$date)
> mean(year_duration)
\end{Sinput}
\begin{Soutput}
Time difference of 365.2545 days
\end{Soutput}
\end{Schunk}

Here, the time difference units are reported for the \code{difftime} object \code{year_duration}, but if we would use it in a linear algebra operation
\begin{Schunk}
```

<sup>1</sup>data from <http://climate.nasa.gov/vital-signs/global-temperature/>

```
\begin{Sinput}
> year_duration %*% rep(1, length(year_duration)) / length(year_duration)
\end{Sinput}
\begin{Soutput}
[,1]
[1,] 365.2545
\end{Soutput}
\end{Schunk}
the unit is dropped. Similarly, for linear regression coefficients we see
\begin{Schunk}
\begin{Sinput}
> coef(lm(Annual_Mean ~ date, temp_data))
\end{Sinput}
\begin{Soutput}
(Intercept)      date
1.833671e-02 4.364763e-05
\end{Soutput}
\begin{Sinput}
> coef(lm(Annual_Mean ~ time, temp_data))
\end{Sinput}
\begin{Soutput}
(Intercept)      time
1.833671e-02 5.051809e-10

```

where the unit of change is in degrees Celsius but either *per day* (date) or *per second* (time). For purely mathematical manipulations, R often strips context from numbers when it is carried in attributes, the linear algebra routines being a prime example.

Most variables are somehow attributed with information about their *units*, which specify what the value 1 of this variable represents. This may be counts of something, e.g. ‘1 apple’, but it may also refer to some *physical unit*, such as distance in meter. This article discusses how strong unit support can be introduced in R.

## SI

The BIPM (Bureau International des Poids et Mesures) is the “*the intergovernmental organization through which Member States act together on matters related to measurement science and measurement standards. Its recommended practical system of units of measurement is the International System of Units (Système International d’Unités, with the international abbreviation SI)*<sup>2</sup>”. International Bureau of Weights and Measures et al. (2001) describe the SI units, where, briefly, *SI units*

- consist of seven base units (length, mass, time & duration, electric current, thermodynamic temperature, amount of substance, and luminous intensity), each with a name and abbreviation (Table 1)
- consist of *derived units* that are formed by products of powers of base units, such as ‘ $m/s^2$ ’, many of which have special names and symbols (e.g. angle: 1 rad = 1 m/m; force: 1 N = 1 m kg s $^{-2}$ )
- consist of *coherent derived units* when derived units include no numerical factors other than one (with the exception of ‘kg’<sup>3</sup>); an example of a coherent derived unit is 1 watt = 1 joule per 1 second,
- may contain SI prefixes (k = kilo for  $10^3$ , m = milli for  $10^{-3}$ , etc.)
- contain special quantities where units disappear (e.g., m/m) or have the nature of a count, in which cases the unit is ‘1’.

## Related work in R

Several R packages provide unit conversions. For instance, **measurements** (Birk, 2016) provides a collection of tools to make working with physical measurements easier. It converts between metric and imperial units, or calculates a dimension’s unknown value from other dimensions’ measurements. It does this by the `conv_unit` function:

<sup>2</sup><http://www.bipm.org/en/measurement-units/>

<sup>3</sup>as a base unit, kg can be part of coherent derived units

Base quantity		SI base unit	
Name	Symbol	Name	Symbol
length	$l, x, r$ , etc.	meter	m
mass	$m$	kilogram	kg
time, duration	$t$	second	s
electric current	$I, i$	ampere	A
thermodynamic temperature	$T$	kelvin	K
amount of substance	$n$	mole	mol
luminous intensity	$I_v$	candela	cd

**Table 1:** base quantities, SI units and their symbols (from International Bureau of Weights and Measures et al. (2001), p. 23)

```
> library(measurements)
> conv_unit(2.54, "cm", "inch")
[1] 1
> conv_unit(c("101 44.32", "3 19.453"), "deg_dec_min", "deg_min_sec")
\end{Sinput}
\begin{Soutput}
[1] "101 44 19.2000000000116" "3 19 27.1800000000003"
\end{Soutput}
\begin{Sinput}
> conv_unit(10, "cm_per_sec", "km_per_day")
[1] 8.64
```

but uses for instance kph instead of ‘km\_per\_hour’, and then ‘m3\_per\_hr’ for flow – unit names seem to come from convention rather than systematic composition. Object `conv_unit_options` contains all 173 supported units, categorized by the physical dimension they describe:

```
> names(conv_unit_options)
[1] "acceleration" "angle"          "area"           "coordinate"    "count"
[6] "duration"      "energy"         "flow"          "length"       "mass"
[11] "power"         "pressure"       "speed"         "temperature" "volume"
> conv_unit_options$volume
[1] "ul"            "ml"            "dl"            "l"             "cm3"          "dm3"
[7] "m3"            "km3"           "us_tsp"        "us_tbsp"      "us_oz"        "us_cup"
[13] "us_pint"       "us_quart"      "us_gal"        "inch3"        "ft3"          "mi3"
[19] "imp_tsp"       "imp_tbsp"      "imp_oz"        "imp_cup"      "imp_pint"     "imp_quart"
[25] "imp_gal"
```

Function `conv_dim` allows for the conversion of units in products or ratios, e.g.

```
> conv_dim(x = 100, x_unit = "m", trans = 3, trans_unit = "ft_per_sec", y_unit = "min")
[1] 1.822689
```

computes how many minutes it takes to travel 100 meters at 3 feet per second.

Package **NISTunits** (Gama, 2014) provides fundamental physical constants (Quantity, Value, Uncertainty, Unit) for SI and non-SI units, plus unit conversions, based on the data from NIST (National Institute of Standards and Technology). The package provides a single function for every unit conversion; all but 5 from its 896 functions are of the form ‘NISTxxxT0yyy’ where ‘xxx’ and ‘yyy’ refer to two different units. For instance, converting from  $W \text{ m}^{-2}$  to  $W \text{ inch}^{-2}$  is done by

```
> library(NISTunits)
> NISTwattPerSqrMeterTOwattPerSqrInch(1:5)
[1] 0.00064516 0.00129032 0.00193548 0.00258064 0.00322580
```

Both **measurements** and **NISTunits** are written entirely in R.

## UNIDATA’s udunits library and the udunits2 R package

Udunits, developed by UCAR/UNIDATA, advertises itself on its web page<sup>4</sup> as: “The *udunits* package supports units of physical quantities. Its C library provides for arithmetic manipulation of units and for

<sup>4</sup><https://www.unidata.ucar.edu/software/udunits/>

conversion of numeric values between compatible units. The package contains an extensive unit database, which is in XML format and user-extensible. The R package **udunits2** (Hiebert, 2015) provides an R level interface to the most important functions in the C library.

The functions provided by **udunits2** are

```
> library(udunits2)
> ls(2)
[1] "ud.are.convertible"  "ud.convert"           "ud.get.name"
[4] "ud.get.symbol"       "ud.have.unit.system" "ud.is.parseable"
[7] "ud.set.encoding"
```

Dropping the ud prefix, **is.parseable** verifies whether a unit is parseable

```
> ud.is.parseable("m/s")
[1] TRUE
> ud.is.parseable("q")
[1] FALSE
```

**are.convertible** specifies whether two units are convertible

```
> ud.are.convertible("m/s", "km/h")
[1] TRUE
> ud.are.convertible("m/s", "s")
[1] FALSE
```

**convert** converts units that are convertible, and throws an error otherwise

```
> ud.convert(1:3, "m/s", "km/h")
[1] 3.6 7.2 10.8
```

and **get.name**, **get.symbol** and **set.encoding** get name, get symbol or modify encoding of the character unit arguments.

```
> ud.get.name("kg")
[1] "kilogram"
> ud.get.symbol("kilogram")
[1] "kg"
> ud.set.encoding("utf8")
NULL
```

Unlike the **measurements** and **NISTunits**, **udunits2** parses units as expressions, and bases its logic upon the convertibility of expressions, rather than the comparison of fixed strings:

```
> m100_a = paste(rep("m", 100), collapse = "*")
> m100_b = "dm^100"
> ud.is.parseable(m100_a)
[1] TRUE
> ud.is.parseable(m100_b)
[1] TRUE
> ud.are.convertible(m100_a, m100_b)
[1] TRUE
```

This has the advantage that through complex computations, intermediate objects can have units that are arbitrarily complex, and that can potentially be simplified later on. It also means that the package practically supports an unlimited amount of derived units.

## Udunits versus the Unified Code for Units of Measure (UCUM)

Another set of encodings for measurement units is the Unified Code for Units of Measure (UCUM, Schadow and McDonald (2009)). A dedicated web site<sup>5</sup> describes the details of the differences between **udunits** and UCUM, and provides a conversion service between the two encoding sets.

The UCUM website refers to some Java implementations, but some of the links seem to be dead. UCUM is the preferred encoding for standards from the Open Geospatial Consortium. **udunits** on the other hand is the units standard of choice by the climate science community, and is adopted by the CF (Climate and Forecast) conventions, which mostly uses NetCDF. NetCDF (Rew and Davis, 1990) is a binary data format that is widely used for atmospheric and climate model predictions.

<sup>5</sup><http://coastwatch.pfeg.noaa.gov/erddap/convert/units.html>

The udunits library is a C library that has strong support from UNIDATA, and we decided to build our developments on this, rather than on Java implementations of UCUM with a less clear provenance.

## Handling data with units in R: the `units` package

The `units` package builds "units" objects from scratch, where `m`, created by

```
> library(units)
> m = make_unit("m")
> str(m)
Class 'units' atomic [1:1] 1
..- attr(*, "units")=List of 2
... $ numerator : chr "m"
... $ denominator: chr(0)
.. ..- attr(*, "class")= chr "symbolic_units"
```

represents '1 m', one meter. Other length values are obtained by using this unit in an expression:

```
> x1 = 1:5 * m
```

As an alternative to using `make_unit`, we can retrieve units directly from the `ud_units` database, which is part of `units`, and was derived from the xml units database that is part of udunits. Two ways of doing this are

```
> x2 = 1:5 * ud_units$m
> identical(x1, x2)
[1] TRUE
> x3 = 1:5 * with(ud_units, m)
> identical(x1, x3)
[1] TRUE
```

Although one could attach `ud_units` to use the units directly, there are over 3000 and this would not only clobber the namespace but also lead to conflicts, e.g. for T (Tesla, TRUE) or in (inch, reserved R language element). The last form using `with` has the advantage that it can take direct expressions:

```
> with(ud_units, m/s^2)
1 m/s^2
```

Several manipulations with "units" objects will now be illustrated.

```
> m = with(ud_units, m)
> km = with(ud_units, km)
> cm = with(ud_units, cm)
> s = with(ud_units, s)
> h = with(ud_units, h)
```

Manipulations that do not involve unit conversion are for instance addition:

```
> x = 1:3 * m/s
> x + 2 * x
Units: m/s
[1] 3 6 9
```

Explicit unit conversion is done by assigning new units:

```
> units(x) = cm/s
> x
Units: cm/s
[1] 100 200 300
> as.numeric(x)
[1] 100 200 300
```

similar to the behaviour of "difftime" objects, this modifies the numeric values without modifying their meaning (what the numbers refer to).

When mixing units in sums, comparisons or concatenation, units are automatically converted to those of the first argument:

```
> y = 1:3 * km/h
> x + y
Units: cm/s
[1] 127.7778 255.5556 383.3333
> y + x
Units: km/h
[1] 4.6 9.2 13.8
> x < y
[1] FALSE FALSE FALSE
> c(y, x)
Units: km/h
[1] 1.0 2.0 3.0 3.6 7.2 10.8
```

where `c(y, x)` concatenates `y` and `x` after converting `x` to the units of `y`. Derived units are created where appropriate:

```
> x * y
Units: cm*km/h/s
[1] 100 400 900
> x^3
Units: cm^3/s^3
[1] 1.0e+06 8.0e+06 2.7e+07
```

and meaningful error messages appear when units are not compatible:

```
> e = try(z <- x + x * y)
> attr(e, "condition")[[1]]
[1] "cannot convert cm*km/h/s into cm/s"
```

The full set of methods and method groups for `units` objects is shown by

```
> methods(class = "units")
[1] as.data.frame c diff format hist
[6] Math mean median Ops plot
[11] print quantile Summary [ units<-
[16] units weighted.mean
see '?methods' for accessing help and source code
```

where the method groups

- `Ops` include operations that require compatible units, converting when necessary (`+`, `-`, `==`, `!=`, `<`, `>`, `<=`, `>=`), and operations that create new units (`*`, `/`, `^` and `**`),
- `Math` include `abs`, `sign`, `floor`, `ceiling`, `trunc`, `round`, `signif`, `log`, `cumsum`, `cummax`, `cummin`, and
- `Summary` include `sum`, `min`, `max` and `range`, and all convert to the unit of the first argument.

When possible, new units are simplified:

```
> a = 1:10 * m/s
> b = 1:10 * h
> a * b
Units: m
[1] 3600 14400 32400 57600 90000 129600 176400 230400 291600 360000
> make_unit(m100_a) / make_unit(m100_b)
1e+100 1
```

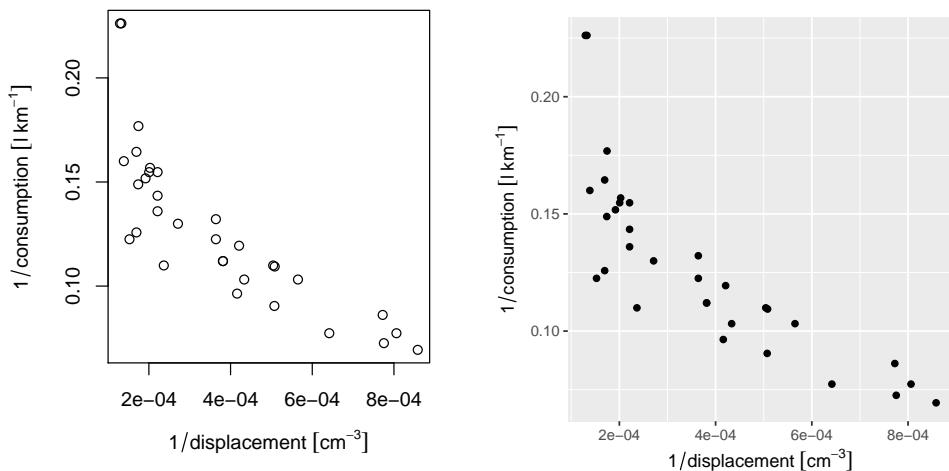
Units are printed as simple R expressions, e.g.

```
> m^5/s^4
1 m^5/s^4
```

Another way to print units commonly seen in Climate and Forecast Conventions<sup>6</sup> is '`m2 s-1`' for  $m^2/s$ . These are not R expressions, but as they are understood by udunits, they can be converted (by udunits) but not simplified (by R):

```
> x = make_unit("m2 s-1")
> y = km^2/h
> z = m^2/s
```

<sup>6</sup>CF, <http://cfconventions.org/Data/cf-standard-names/34/build/cf-standard-name-table.html>



**Figure 1:** Plot of 1/consumption against 1/displacement of dataset `mtcars`, illustrating automatic units in default axis labels (after conversion to SI) for base plot (left) and `ggplot` (right); `demo(ggforce)` illustrates how these plots are generated.

```
> x + y
278.7778 (m2 s-1)
> x/y
1 h*(m2 s-1)/km^2
> z/y
0.0036 1
```

However, `parse_unit` parses such units, and `as_cf` returns such unit strings from "units" objects:

```
> parse_unit("m2 s-1")
1 m^2/s
> as_cf(m^2*s^-1)
[1] "m2 s-1"
```

The `plot` and `hist` methods add units to default axis labels, an example is shown in Figure 1. For `ggplot2` plots (Wickham, 2009), automatic unit placement in default axis label is provided by package `ggforce` (Pedersen, 2016); `demo(ggforce)` gives an example.

Automatic conversion between "units" and "difftime" is provided:

```
> (dt = diff(Sys.time() + c(0, 1, 1+60, 1+60+3600))) # class difftime
Time differences in secs
[1] 1 60 3600
> (dt.u = as.units(dt))
Units: s
[1] 1 60 3600
> identical(as.dt(dt.u), dt) # as.difftime is not a generic
[1] TRUE
```

Objects of class "units" can be used as columns in "data.frame" objects, as well as in "tbl\_df" (Wickham et al., 2016).

## Discussion and conclusions

The `units` R package provides a new class, "units", for numeric data with associated measurement units. Operations on objects of this class retain the unit metadata and provide automated dimensional analysis: dimensions are taken into consideration in computations and comparisons. Combining different units that are compatible triggers automatic unit conversion, derived units are automatically generated and simplified where possible, and meaningful error messages are given when a user tries to add objects with incompatible units. This verifies that computations are not only syntactically and numerically allowed, but also semantically, and in the case of physical units, physically allowed, which may support code verification and provenance tracking. Using this package may eliminate a whole class of potential scientific programming mistakes.

Where the R packages `measurements` and `NISTunits` provide conversion between a fixed number of units, with the help of the `udunits` library and unit database R package `units` allows for arbitrarily complex derived units. By treating units as expressions it can derive, convert and simplify units. In addition, beyond the SI units packaged, `units` handles user-defined units not supported by `udunits`.

Data in "units" vectors can be stored as columns in "data.frame" or "tbl\_df" objects, and can be converted to and from "difftime". When "units" objects have associated time and location information, they could be stored in spatial or spatio-temporal objects provided by `sp` or `spacetime` (Pebesma, 2012) as these store attribute data in "data.frame" slots, but for instance not in "zoo" (Zeileis and Grothendieck, 2005) or "xts" (Ryan and Ulrich, 2014) objects, as these latter two set the class attribute of a vector or matrix.

Despite all standardization efforts, units may still be ambiguous, or subject to interpretation. For instance for the duration of one year `NISTunits` or `udunits2` give us an answer that depends on whether we want a common, leap, Gregorian, Julian, tropical or siderial year (Lang (2006), see also `demo(year)`). This illustrates that those who apply unit conversion should be aware of possible pitfalls. Support for calendars in `udunits` seems not as well developed as in R.

Future work includes extending packages that read external data from formats, databases or interfaces with support for measurement unit information into R, preserving the measurement unit information. Examples would be interfaces to HDF5 (e.g., `h5`, Annau (2016)), `RNetCDF` (Michna and Woods, 2016) or `sos4R` (Nüst et al., 2011). It would be nice to see units of measurements propagate into units of regression coefficient estimates.

## Acknowledgements

We acknowledge three anonymous reviewers and the handling editor for their constructive comments, and Thomas Lin Pedersen for implementing the ggplot extensions in package `ggeforce` that automatically add units to default ggplot axis labels (Fig 1).

## Bibliography

- M. Annau. *h5: Interface to the 'HDF5' Library*, 2016. URL <https://CRAN.R-project.org/package=h5>. R package version 0.9.7. [p497]
- M. A. Birk. *measurements: Tools for Units of Measurement*, 2016. URL <https://CRAN.R-project.org/package=measurements>. R package version 1.0.0. [p491]
- G. W. Cobb and D. S. Moore. Mathematics, statistics, and teaching. *American Mathematical Monthly*, pages 801–823, 1997. [p490]
- J. Gama. *NISTunits: Fundamental Physical Constants and Unit Conversions from NIST*, 2014. URL <https://CRAN.R-project.org/package=NISTunits>. R package version 1.0.0. [p492]
- G. Grolemund and H. Wickham. Dates and times made easy with lubridate. *Journal of Statistical Software*, 40(1):1–25, 2011. ISSN 1548-7660. doi: 10.18637/jss.v040.i03. URL <https://www.jstatsoft.org/index.php/jss/article/view/v040i03>. [p490]
- J. Hiebert. *udunits2: Udunits-2 Bindings for R*, 2015. R package version 0.9. [p493]
- J. Iliffe and R. Lott. *Datums and Map Projections: For Remote Sensing, GIS and Surveying*. CRC Inc, 2008. [p490]
- International Bureau of Weights and Measures, B. N. Taylor, and A. Thompson. The international system of units (SI). 2001. [p491, 492]
- K. Lang. *Astrophysical Formulae Volume II: Space, Time, Matter and Cosmology, 3rd Edition 1999. 2nd printing*. Springer, 2006. [p497]
- P. Michna and M. Woods. *RNetCDF: Interface to NetCDF Datasets*, 2016. URL <https://CRAN.R-project.org/package=RNetCDF>. R package version 1.8-2. [p497]
- D. Nüst, C. Stasch, and E. J. Pebesma. Connecting R to the sensor web. Lecture Notes in Geoinformation and Cartography, pages 227–246. Springer, 2011. [p497]
- E. Pebesma. *spacetime: Spatio-temporal data in R*. *Journal of Statistical Software*, 51(1):1–30, 2012. ISSN 1548-7660. doi: 10.18637/jss.v051.i07. URL <https://www.jstatsoft.org/index.php/jss/article/view/v051i07>. [p497]

- E. Pebesma and R. Bivand. Classes and methods for spatial data in R. *R News*, 5:9–13, 2005. URL <http://cran.r-project.org/doc/Rnews/>. [p490]
- T. L. Pedersen. *ggforce: Accelerating 'ggplot2'*, 2016. URL <https://CRAN.R-project.org/package=ggforce>. R package version 0.1.0. [p496]
- R. Rew and G. Davis. NetCDF: an interface for scientific data access. *IEEE computer graphics and applications*, 10(4):76–82, 1990. [p493]
- J. A. Ryan and J. M. Ulrich. *xts: eXtensible Time Series*, 2014. URL <https://CRAN.R-project.org/package=xts>. R package version 0.9-7. [p497]
- G. Schadow and C. J. McDonald. The unified code for units of measure. *Regenstrief Institute and UCUM Organization: Indianapolis, IN, USA*, 2009. [p493]
- S. Scheider, B. Gräler, E. Pebesma, and C. Stasch. Modeling spatiotemporal information generation. *International Journal of Geographical Information Science*, 30(10):1980–2008, 2016. URL <http://dx.doi.org/10.1080/13658816.2016.1151520>. [p490]
- H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. ISBN 978-0-387-98140-6. URL <http://ggplot2.org>. [p496]
- H. Wickham, R. Francois, and K. Müller. *tibble: Simple Data Frames*, 2016. URL <https://CRAN.R-project.org/package=tibble>. R package version 1.2. [p496]
- A. Zeileis and G. Grothendieck. zoo: S3 infrastructure for regular and irregular time series. *Journal of Statistical Software*, 14(6):1–27, 2005. doi: 10.18637/jss.v014.i06. [p497]

*Edzer Pebesma*  
Institute for Geoinformatics  
Heisenbergstraße 2, 48149 Münster  
Germany  
[edzer.pebesma@uni-muenster.de](mailto:edzer.pebesma@uni-muenster.de)

*Thomas Mailund*  
Bioinformatics Research Center  
Aarhus University  
[mailund@birc.au.dk](mailto:mailund@birc.au.dk)

*James Hiebert*  
Pacific Climate Impacts Consortium  
University of Victoria, Canada  
[hiebert@uvic.ca](mailto:hiebert@uvic.ca)

# mctest: An R Package for Detection of Collinearity among Regressors

by Muhammad Imdadullah, Muhammad Aslam, Saima Altaf

**Abstract** It is common for linear regression models to be plagued with the problem of multicollinearity when two or more regressors are highly correlated. This problem results in unstable estimates of regression coefficients and causes some serious problems in validation and interpretation of the model. Different diagnostic measures are used to detect multicollinearity among regressors. Many statistical software and R packages provide few diagnostic measures for the judgment of multicollinearity. Most widely used diagnostic measures in these software are: coefficient of determination ( $R^2$ ), variance inflation factor/tolerance limit (VIF/TOL), eigenvalues, condition number (CN) and condition index (CI) etc. In this manuscript, we present an R package, **mctest**, that computes popular and widely used multicollinearity diagnostic measures. The package also indicates which regressors may be the reason of collinearity among regressors.

## Brief introduction of collinearity

Consider the conventional multiple linear regression equation

$$y = X\beta + u,$$

where  $y$  is an  $n \times 1$  vector of observation on response variable,  $X$  is known design matrix of order  $n \times p$ ,  $\beta$  is an  $p \times 1$  vector of unknown parameters and  $u$  is an  $n \times 1$  vector of random errors with mean zero and variance  $\sigma^2 I_n$ , where  $I_n$  is an identity matrix of order  $n$ .

One of the important assumptions of the classical linear regression model is that there is no exact collinearity among the regressors otherwise, the issue is referred to as multicollinearity. Generally, the problem of multicollinearity may also refer to have not only exact linear relationship but also high correlations among some or all regressors of a regression model under study. Strictly speaking, multicollinearity is usually refers to the existence of more than one exact linear relationship among regressors, while collinearity refers to the existence of a single linear relationship among regressors. However, in general, the term multicollinearity may be referred to both the cases. Data collection method, constraints on the fitted regression model, model specification error, overdefined model, some common trend in time series data and naturally correlated data may be some potential sources of multicollinearity.

The problem of multicollinearity has potentially serious effect on the regression estimates such as implausible coefficient signs, impossible inversion of matrix  $X'X$  as it becomes either singular (in the case of perfect multicollinearity) or near to singular (in the case of near to perfect multicollinearity), large magnitude of coefficients in absolute value, large variance or standard errors with wider confidence intervals and small  $t$ -ratios. The ordinary least squared (OLS) estimators and standard errors also become sensitive to small change in data when regressors are collinear to each other (see Belsley et al., 1980; Dorsett et al., 1983; Farrar and Glauber, 1967; Gunst and Mason, 1977; Johnston, 1963; Mason et al., 1975). On the basis of theoretical considerations, these indications signify the need for detection of multicollinearity among regressors (Belsley et al., 1980; Greene, 2002; Younger, 1979).

This paper presents the overview of existing collinearity diagnostic measures along with commonly used threshold values for the judgment of existence of collinearity among regressors. These diagnostic measures are being implemented in R with the proposed **mctest** package (Imdadullah and Aslam, 2016).

## Collinearity diagnostic measures

Several numerical methods for the detection of collinearity are available in the existing literature proposed or discussed by various authors e.g., (see Belsley et al., 1980; Curto and Pinto, 2011; Koutsoyiannis, 1977; Kovács et al., 2005; Marquardt, 1970; Montgomery and Peck, 1982, etc.). Widely used and the most suggested collinearity diagnostic measures are values of pair-wise correlations, R-squared value (c.f. Gujarati and Porter (2008)), variance inflation factor (VIF), tolerance limit (TOL) (Kutner et al., 2004; Marquardt, 1970), eigenvalues (Kendall, 1957; Silvey, 1969), condition number (CN) and condition index (CI) (Belsley et al., 1980), Leamer's method (Greene, 2002), Kleins rule (Klein, 1962), three tests proposed by Farrar and Glauber (Farrar and Glauber, 1967), the Red indicator (Kovács et al., 2005), and Theil's measure (Theil, 1971). However, none of these can be regarded as the

best choice for the detection of collinearity (Kovács et al., 2005).

Following are the diagnostics that can be considered as the classical symptoms of harmfulness of multicollinearity. (i) If zero-order (pair-wise) correlation coefficient between two regressors is high (say >0.8) then multicollinearity may be a serious problem (Gujarati and Porter, 2008; Maddala, 1988). However, it is not sufficient and necessary condition for the detection of multicollinearity because a linear relation involves many of the regressors, therefore it may not be possible to detect such a relation with a simple correlation or pairs-wise plot (Chatterjee and Hadi, 2006; Judge et al., 1985). (ii) High  $R^2$  (say >0.8) may indicate the problem of multicollinearity (Gujarati and Porter (2008)). In most of the cases, overall  $F$ -test rejects the null hypothesis of partial slopes for being zero, but some or all individual  $t$ -ratios of partial slopes may be non-significant. Therefore, a model having no multicollinearity problem should have high  $R^2$  and larger (significant)  $t$ -ratios of partial slopes. (iii) High variance of regression coefficients' estimates and low  $t$ -ratios also suggest the existence of multicollinearity.

We classified other widely used collinearity diagnostics as overall and individual measures of collinearity. This classification is due to the fact that there are some diagnostic measures resulting in a single number, while others yield as many quantities as the number of regressors in the model. The overall diagnostic measures help to get an idea about only the existence of collinearity and they do not tell which regressor may be the reason of collinearity, while the individual measures point out the regressors causing collinearity. Since no specific collinearity diagnostic measure is superior and each of these measures has different collinearity detection criterion (threshold value) proposed by various authors in the textbooks and research articles, there is need to study multiple collinearity diagnostics. That is, there is no clear-cut criterion for evaluating multicollinearity in linear regression models. Similarly, some diagnostic measures are statistically criticized such as tests proposed by Farrar and Glauber (1967) while threshold values of many other diagnostic measures are subjective in nature as no unique or standard critical values exist for these measures. Moreover, different collinearity detection methods are not comparable with each other. That is why, many regression analysts often rely on more than one collinearity diagnostic measures.

Following is the list of overall and individual collinearity diagnostic measures along with short description, formula, detection criterion (threshold value) and reference for each measure. These diagnostic measures will assist the researchers in determining when and where some corrective action is necessary. According to Belsley et al. (1980), the investigations concerning the presence of multicollinearity have been based on judging the magnitudes of various diagnostic measures.

## Overall collinearity diagnostic measures

- **Determinant:**

The matrix  $X'X$  will be singular if it contains linearly dependent columns or rows. Therefore, determinant of normalized correlation matrix ( $R = X'X$ ) without intercept can be used to indicate existence of collinearity among regressors. However, determinant does not provide information about interdependence among regressors, it only provides information about singularity (departure from orthogonality) of a correlation matrix. The determinant of  $X'X$  on the scale is  $0 \leq |X'X| \leq 1$  (Cooley and Lohnes, 1971). If  $|X'X| \sim 0$ , then collinearity exists among regressors (Asteriou and Hall, 2007).

- **$R$ -squared:**

Coefficient of determination ( $R^2$ ) from regression of all  $x$  on  $y$ . The  $R^2$  is a monotonic non-decreasing function of number of regressors included in the model, that is,  $R^2$  indicates how well the regression fits the data (Gujarati and Porter, 2008; Stock and Watson, 2010). On the other hand, higher the  $R^2$  values, the more chances of regressors to be plagued with multicollinearity (Asteriou and Hall, 2007; Gujarati and Porter, 2008; Maddala, 1988), since  $R^2$  is affected by regressors sharing their variances (Gujarati and Porter, 2008; Maddala, 1988).

- **Farrar  $\chi^2$ :**

It is the Chi-square test for detecting the strength of collinearity over the complete set of regressors.  $\chi^2 = -\left[n - 1 - \frac{1}{6(2p+5)}\right] \times \log_e [X'X] \sim \psi_{v=\frac{1}{2}p(p-1)}^2$ . Collinearity exists among regressors if  $\chi^2 > \chi_{\frac{1}{2}p(p-1)}^2$  (Farrar and Glauber, 1967).

- **Condition index:**

$CI_j = \sqrt{\frac{\max(\lambda_j)}{\lambda_j}}$   $j = 1, 2, \dots, p$ ;  $\lambda_1 \geq \lambda_2 \dots \geq \lambda_p$ . Collinearity exists if any of  $CI_j > 10, 15$ , or 30 (Belsley et al., 1980; Chatterjee and Hadi, 2006; Maddala, 1988).

- **Sum of reciprocal of eigenvalues:**

In an orthogonal system  $\sum_{j=1}^p \frac{1}{\lambda_j} = p$ , therefore, for a sample based correlation matrix  $R$  with

eigenvalues  $\lambda_j$  comparing  $p$  with  $\sum_{j=1}^p \frac{1}{\lambda_j}$  can be used to indicate collinearity. If  $\sum_{j=1}^p \frac{1}{\lambda_j}$  is (say) five times larger than the number of regressors used in the model then collinearity exists among regressors (Chatterjee and Price, 1977; Dillon and Goldstein, 1984).

- **Theil's indicator:**

Theil (1971) proposed a measure of collinearity based on an incremental contribution ( $R^2 - R_j^2$ ) to the squared multiple correlation, where  $R_j^2$  is the  $R^2$  from auxiliary regression of regressors.  $m = R^2 - \sum_{i=1}^p (R^2 - R_{-i}^2)$ . If  $m = 0$  then all  $X$ 's are mutually uncorrelated (no redundancy exists) as the incremental contribution all add up to  $R^2$ . However, if  $m \sim 1$  then collinearity exists among regressors.

- **Red indicator:**

Kovács et al. (2005) presented a synthetic and new normalized indicator for diagnostic of collinearity by using eigenvalues or quantifying the average correlation of the data.  $Red = \frac{\sqrt{\sum_{j=1}^p (\lambda_j - 1)^2}}{\sqrt{p-1}}$ . If value of the Red indicator is zero ( $Red = 0$ ) then it indicates the absence of redundancy and value near to 1 ( $Red \sim 1$ ) indicates maximum redundancy ( $Red \sim 1$ ).

## Individual collinearity diagnostic measures

- **Klein's rule:**

If  $R_j$  from the auxiliary regression is greater than the overall  $R^2$  (obtained from the regression of  $y$  on all the regressors) then multicollinearity may be troublesome. The decision rule for detection of collinearity is,  $R_{x_j, x_1, x_2, \dots, x_p}^2 > R_{y, x_1, x_2, \dots, x_p}^2$  (Klein, 1962).

- **VIF and TOL:**

VIF measures how much variances of the estimated regression coefficients are increased over the case of no correlation among  $p$  regressors. The diagonal elements of  $(X'X)^{-1}$  matrix are considered as very important in detecting multicollinearity.  $VIF_j = (X'X)_{jj}^{-1} = \frac{1}{1-R_j^2}$  and  $TOL_j = \frac{1}{VIF_j} = 1 - R_j^2$ .

The criticism on VIF is that  $var(\hat{\beta}_j) = \frac{\sigma^2}{\sum x_j^2} VIF_j$  depends on  $\sigma^2$ ,  $\sum x_j^2$  and VIF, which shows that a high VIF can be counterbalanced by a low  $\sigma^2$  or high  $\sum x_j^2$ . So a high VIF is neither a necessary nor a sufficient measure of multicollinearity (Gujarati and Porter, 2008). The value of  $VIF > 3, 5, 10$  or value of  $TOL \sim 0$  indicates existence of collinearity among regressors (Kutner et al., 2004; Marquardt, 1970).

- **Eigenvalues:**

Kendall (1957) and Silvey (1969) suggested the use of eigenvalues of  $X'X$  (correlation matrix) to check the presence of multicollinearity and set the criteria that small eigenvalues (near to zero) are indication of high collinearity, however, they did not mentioned how much small it should be. One or more smaller eigenvalues of  $X'X$  or its related correlation matrix indicate collinearity (Kendall, 1957; Silvey, 1969).

- **CVIF:**

Curto and Pinto (2011) proposed new measure of multicollinearity to evaluate the impact of the correlation among regressors in the variance of the OLSEs.  $CVIF_j = VIF_j \times \frac{1-R_j^2}{1-R_0^2}$  where,  $R_0^2 = R_{yx_1}^2 + R_{yx_2}^2 + \dots + R_{yx_p}^2$ . Collinearity exists if  $CVIF_j \geq 10$  (Curto and Pinto, 2011).

- **Leamer's method:**

Leamer (in Greene (2002)) suggested a measure of the effect of multicollinearity for the  $j$ th variable;  $C_j = \left\{ \frac{\left( \sum_i^n (X_{ij} - \bar{X}_j)^2 \right)^{-1}}{(X'X)_{jj}^{-1}} \right\}^{\left( \frac{1}{2} \right)}$ . This measure is the square root of the ratio of variances of estimated coefficients ( $\hat{\beta}_j$ ) when estimated without and with the other regressors. If  $X_j$  is uncorrelated with the other regressors  $C_j$  would be 1 otherwise will be equal to  $(1 - R_j^2)^{\frac{1}{2}}$ , i.e.,  $C_j \sim 0$  indicates existence of collinearity among regressors.

- **F and  $R^2$  relation:**

The relationship of F-test and  $R^2$  from regressing  $X_j$  on the other remaining regressors can

be used to detect multicollinearity. The relationship is described as:  $F_j = \frac{\frac{R_{x_j, x_1, \dots, x_p}^2}{p-2}}{\frac{1-R_{x_j, x_1, \dots, x_p}^2}{n-p+1}} \sim F(p-2, n-p+1)$ , where  $F^* = F_{p-2, n-p+1}$ . If  $F_j > F^*$ , then it means that the regressor  $X_j$  is collinear with other regressors and it should be dropped from the model (Gujarati and Porter, 2008).

- **Farrar  $w_j$ :**

It is an  $F$ -test for locating the regressors which are collinear with others and it makes use of multiple correlation coefficients among regressors.  $w_j = \frac{R_j^2}{1-R_j^2} \left( \frac{n-p}{p-1} \right) \sim F_{(n-p, p-1)}$ . If  $w_j > F_{(n-p, p-1)}$ , there is indication of considerable collinearity (Farrar and Glauber, 1967).

There are few software and R packages that provide some collinearity diagnostic measures such as correlation matrix, VIF/TOL, eigenvalues/eigenvectors, and CN/CI. The design goal of our developed package **mctest** is primarily to provide a comprehensive suite of all the listed diagnostic measures. All R packages mentioned in Table 1 are compared with our **mctest** package regarding diagnostic measures in these packages. Other features in these packages and collinearity related measures available in different statistical software are also discussed.

	<b>perturb</b>	<b>HH</b>	<b>car</b>	<b>fmsb</b>	<b>rms</b>	<b>faraway</b>	<b>usdm</b>	<b>mctest</b>
<i>Overall collinearity diagnostics</i>								
$ X'X $								✓
R-squared								✓
Farrar $\chi^2$								✓
CN/CI			✓					✓
$\sum_{j=1}^p \frac{1}{\lambda_j}$								✓
Theil's indicator								✓
Red indicator								✓
<i>Individual collinearity diagnostics</i>								
Correlation matrix								✓
Var and $t$ -ratios								✓
Klein's rule								✓
VIF	✓	✓	✓	✓	✓	✓	✓	✓
TOL								✓
Eigenvalues								✓
CVIF								✓
Leamer's method								✓
Farrar $W_i$								✓
F and $R^2$ relation								✓

**Table 1:** Comparison of collinearity related R packages

There are few statistical software (SAS (SAS 9.3, 2011), Stata (StataCorp, 2015), Minitab (Minitab, Inc., 2014), NCSS (NCSS , 2016), and StatGraphics (Statgraphics Centurion XVII, 2015) etc.), giving different collinearity diagnostic measures such as ( $R^2$ , eigenvalues, VIF, CN, and correlation matrix etc.). The R packages mentioned in Table 1 have some other functionalities related to collinearity. For example, **perturb** (Hendrickx, 2012) evaluates collinearity by adding random noise to selected variables and computes the CN and variance decomposition proportion to test the collinearity and to uncover its sources. The package **car** (Fox and Weisberg, 2011) computes the VIF and GVIF for linear and generalized linear models. The function **vif** of package **usdm** (Naimi, 2015) computes the VIF for a set of variables and excludes highly correlated variables from the set through a stepwise procedure. The package **rms** Harrell Jr (2016) computes VIF from the covariance matrix of parameter estimates from binary or ordinal regression models, Cox regression, accelerated failure time models, ordinary linear models, the Buckley-James model, generalized least squares for serially or spatially correlated observations, generalized linear models, and quantile regression. The packages, **HH** (Heiberger, 2016), **fmsb** (Nakazawa, 2015) and **faraway** (Faraway, 2016) present different statistical methods and an extensive use of graphical display.

There are some other R packages such as **VIF** (Lin, 2012), **leaps** (Lumley, 2009), **bestglm** (McLeod and Xu, 2014), **glmulti** (Calcagno, 2013), and **meifly** (Wickham, 2014) that are used for collinear datasets. These packages involved procedures to search for adequate predictors and for parsimonious

models (subset or all subset regression). The availability of different collinearity diagnostic measures in R packages, shown in Table 1 and in different statistical software, suggests that the package **mctest** is a useful addition on CRAN.

## R implementation

In this section, we illustrate the use of our developed package **mctest**. The R package **mctest** mainly implements functions for the detection of collinearity among regressors by calling **omcdiag()** and **imcdiag()** functions. For the graphical representation of VIF values and eigenvalues, **mc.plot()** function can also be used. We try to build a simple interface to facilitate the usage of this package.

The functions, **omcdiag**, **imcdiag**, and **mctest** ensure that the number of regressors provided as  $x$  argument should be at least two. Similarly, the values of regressors and response variable ( $y$ ) should contain only numbers provided that both have equal number of observations. All the other arguments are optional and have default threshold values for different collinearity diagnostic measures. Following is the list of functions available in **mctest**:

Function	Description
<b>omcdiag()</b>	Computation of overall collinearity measures.
<b>imcdiag()</b>	Computation of individual collinearity measures for each regressor.
<b>mctest()</b>	Calls overall and individual collinearity measures.
<b>mc.plot()</b>	Graphical representation of VIF and eigenvalues.

**Table 2:** Functions available in **mctest** package

## Overall collinearity diagnostics

For overall collinearity diagnostic measures, the function **omcdiag()** has only two mandatory arguments: the vector of response variable  $y$  and the matrix of regressors  $x$ . The argument **na.rm** removes the missing values in dataset and is set to TRUE. Therefore, all calculations will be performed on newly created data after removing missing observations if any, otherwise, calculations will be performed on complete observation available in the provided dataset. The optional argument **Inter**, when it takes the value TRUE, allows to compute eigenvalues and condition index including intercept term in design matrix  $X'X$ , otherwise, without it. The other arguments **detr**, **red**, **theil**, **cn**, and **conf** are used as threshold values as collinearity detection criteria. If all these optional arguments are not used, the eigenvalues and CIs with intercept term will be computed and all these values will be compared with the default threshold values (can be provided by the user) for the indication of existence of collinearity by each of the diagnostic methods.

```
omcdiag(x,y,na.rm=TRUE,Inter=TRUE,detr=0.01,red=0.5,conf=0.95,theil=0.5,cn=30,...)
```

The results from each of overall collinearity diagnostic measures are displayed with an indication that whether existence of collinearity among regressors is correctly detected by diagnostic methods or not. The eigenvalues and CIs are also displayed for the confirmation of existence of collinearity.

### Example: **omcdiag()**

This section uses the Hald data (Hald, 1952) bundled in **mctest** package for checking of existence of collinearity among regressors using **omcdiag()** function. Different examples of **omcdiag()** with use of difference arguments are provided, however, results are shown only for the last command.

```
> library('mctest')
> head(Hald)
> x <- Hald[, -1]
> y <- Hald[, 1]

> omcdiag (x, y, detr = 0.001, red = 0.6, conf = 0.99, theil = 0.6, cn = 15)
> omcdiag (x, y, Inter = FALSE)
> omcdiag (x, y)
```

Call:

```
omcdiag(x = x, y = y)

Overall Multicollinearity Diagnostics

MC Results detection
Determinant |X'X|:      0.0011      1
Farrar Chi-Square:     59.8700      1
Red Indicator:        0.5414      1
Sum of Lambda Inverse: 622.3006      1
Theil's Method:        0.9981      1
Condition Number:     249.5783      1

1 --> COLLINEARITY is detected
0 --> COLLINEARITY is not detected by the test
```

```
=====
Eigenvalues with INTERCEPT
    Intercept      X1      X2      X3      X4
Eigenvalues:   4.1197 0.5539 0.2887 0.0376 0.0001
Condition Indexes: 1.0000 2.7272 3.7775 10.4621 249.5783
```

Results from `omcdiag` shows that all of the overall collinearity diagnostic measures correctly detected the presence of multicollinearity among regressors. Similarly, eigenvalues and CIs also indicate regressors are collinear since, some eigenvalues are small enough and at least one of the CIs is greater than 30.

### Individual collinearity diagnostics

For the individual collinearity diagnostic measures, `imcdiag()` also has two mandatory arguments like `omcdiag()` or `mctest()` has. The optional argument `method`, when it takes value "VIF", "TOL", "Wi", "Fi", "Klein", "conf", "CVIF", or "Leamer", will compute only provided method with an indication of whether regressor(s) is(are) possible reason of collinearity or not. The other optional arguments (such as `vif`, `tol` `conf`, `cvif`, and `leamer`) are threshold values to compare with diagnostic measures of VIF, TOL, confidence level for the Farrar-Glauber test of Wi, Fi, CVIF, and Leamer's method, respectively for possible detection of collinearity among regressors. The `corr` argument is set to FALSE, if it takes value as TRUE, the correlation matrix will also be produced along with collinearity diagnostic measures with the indication of which pair of regressors are collinear. The computed value of certain diagnostic measure, provided to `method` argument, is displayed with an indication of whether diagnostic measure correctly detected the existence of collinearity or not. The `all` argument is set to FALSE, if it takes value as TRUE, the individual collinearity diagnostics will be returned in form of 0 or 1. From "lm" function, non-significant *t*-values are also displayed for further subjective judgment and confirmation of the existence of collinearity among regressors.

```
imcdiag(x,y,method = NULL,na.rm = TRUE,corr = FALSE,vif = 10,tol = 0.1,
         conf = 0.95, cvif = 10, leamer = 0.1, all = FALSE,...)
```

### Example: `imcdiag()`

Different examples of `imcdiag()` function with use of different arguments are provided, however, results are shown only for the last command.

```
> imcdiag(x, y, corr = TRUE)
> imcdiag(x, y)
```

Call:

```
imcdiag(x = x, y = y)
```

### Individual Multicollinearity Diagnostics

	VIF	TOL	Wi	Fi	Leamer	CVIF	Klein
X1	38.4962	0.0260	112.4886	187.4811	0.1612	-0.5846	0
X2	254.4232	0.0039	760.2695	1267.1158	0.0627	-3.8635	1
X3	46.8684	0.0213	137.6052	229.3419	0.1461	-0.7117	0
X4	282.5129	0.0035	844.5386	1407.5643	0.0595	-4.2900	1

```

1 --> COLLINEARITY is detected
0 --> COLLINEARITY is not detected by the test

X1 , X2 , X3 , X4 , coefficient(s) are non-significant may be due to multicollinearity

* use method argument to check which regressors may be the reason of collinearity

```

Each column of output from `imcdiag(x,y)` indicates that the computed values of individual collinearity diagnostic measures for each regressor. The last column results in either 0 (no collinearity due  $X_j$ ) or 1 (collinearity due to  $X_j$ ) due to Klein's rule.

To get certain individual collinearity diagnostic with custom threshold can be obtained by using `method` argument. The first column of output contains the value of diagnostic measure. In the second column, 1 and 0 denotes the detection and non-detection of collinearity, respectively, for each of the regressor. The use of switch statement is made to fulfill the purpose of obtaining diagnostic values and the indication of collinearity detection for certain collinearity diagnostics provided as value to argument `method`. Some examples of obtaining certain individual collinearity diagnostic measures are;

```

> imcdiag(x, y, method = "VIF", vif = 5)
> imcdiag(x, y, method = "VIF", vif = 10, corr = TRUE)
> imcdiag(x, y, method = "CVIF", cvif = 10)

```

Call:  
`imcdiag(x = x, y = y, method = "CVIF", cvif = 10)`

#### Individual Multicollinearity Diagnostics

CVIF detection	
X1	-0.5846
X2	-3.8635
X3	-0.7117
X4	-4.2900
	0
	0
	0
	0

NOTE: CVIF Method Failed to detect multicollinearity

0 --> COLLINEARITY in not detected by the test

If argument `all` in `imcdiag` or `mctest` is set to TRUE, a matrix of either 0 or 1 will be displayed. Few examples for use of `all` argument are;

```

> imcdiag(x, y, all = TRUE)
> imcdiag(x, y, all = TRUE, vif = 15, conf = 0.99, )
> imcdiag(x, y, method = "VIF", all = TRUE)
> mctest(x, y, all = TRUE, type="i")

```

Call:  
`imcdiag(x = x, y = y, all = TRUE)`

#### All Individual Multicollinearity Diagnostics in 0 or 1

VIF TOL Wi Fi Leamer CVIF Klein							
X1	1	1	1	1	0	0	0
X2	1	1	1	1	1	0	1
X3	1	1	1	1	0	0	0
X4	1	1	1	1	1	0	1

1 --> COLLINEARITY is detected  
0 --> COLLINEARITY in not detected by the test

X1 , X2 , X3 , X4 , coefficient(s) are non-significant may be due to multicollinearity

R-square of y on all x: 0.9824

\* use method argument to check which regressors may be the reason of collinearity

```

mctest(x,y,type = c("o","i","b"),na.rm = TRUE,Inter = TRUE,method = NULL,
       corr = FALSE,detr = 0.01,red = 0.5,cn = 30,vif = 10,tol = 0.1,conf = 0.95,

```

```
cvif = 10,leamer = 0.1,all = FALSE,... )
```

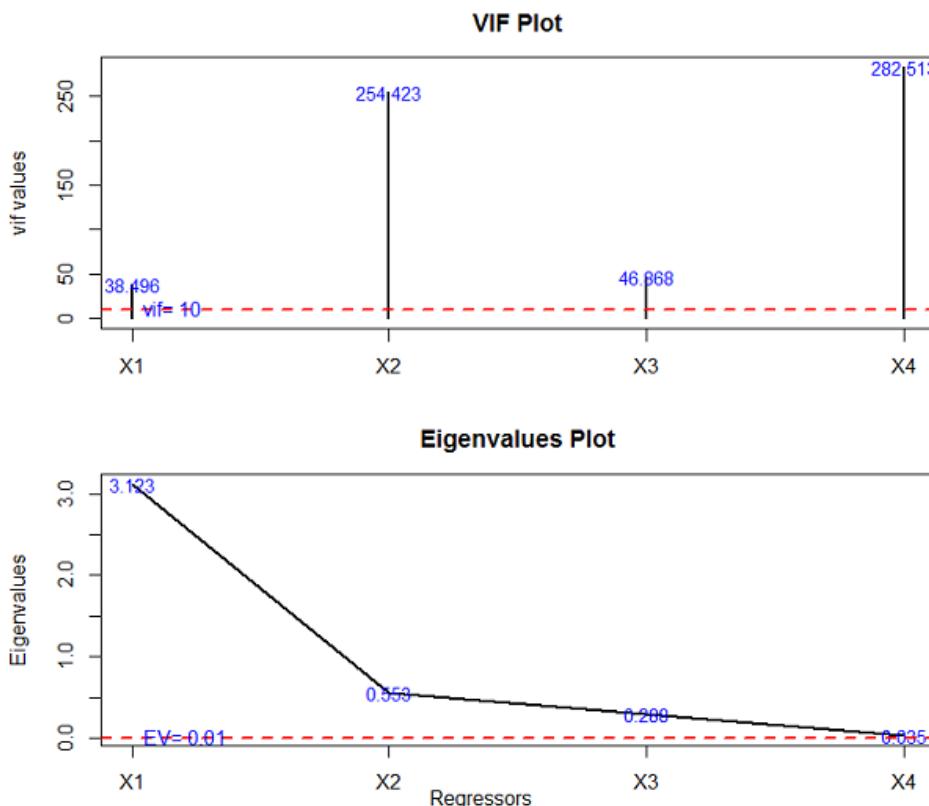
The `mctest()` function also has two mandatory arguments: the vector of response  $y$  and the matrix of regressors as  $x$ . The argument `type` is optional for computation of overall (from `omcdiag`) by setting `type="o"`, individual (from `imcdiag`) by setting `type="i"` or both overall and individual collinearity diagnostics by setting `type="b"`, if `type` argument is not used overall collinearity measures will be computed and displayed.

### Collinearity diagnostic plots: VIF and eigenvalues plot

The `mc.plot` function can also be used to draw the plots of VIF values and eigenvalues to graphically judge the existence of collinearity among regressors. The VIF values and eigenvalues are also drawn for each regressor along the  $y$ -axis. A horizontal red dashed line equal to either default threshold or may be provided by the user of `mctest`, for both VIF and eigenvalues.

```
> mc.plot(x, y)
> mc.plot(x, y, vif = 10, ev = 0.1)
```

The argument, `vif = 10` and `ev = 0.1` are user provided thresholds for VIF and eigenvalues, respectively and will be shown as horizontal red dashed line.



**Figure 1:** The VIF and Eigenvalues Plots.

From VIF plot, the VIF values of each regressor greater than 30 indicates the existence of multicollinearity among regressors. Similarly, the eigenvalues plot indicates that few regressors have relatively smaller eigenvalues than others, indicating the existence of collinearity. Note that the graphical output (shown in Figure 1) from `mc.plot()` and numerical results from, for example, `mctest()` are all equivalent. Only difference exists in the way of their representation.

### Dealing with multicollinearity

Complete elimination of multicollinearity is not possible, but the degree of collinearity can be reduced. Depending on the severity of the collinearity problem, there are two schools of thought (a) do nothing or (b) follow some rules of thumb. According to the first school of thought, [Blanchard \(1967\)](#) suggested to do nothing with the regressors or model, since multicollinearity is essentially a data deficiency

problem and sometimes there is no choice over the data available for empirical analysis. Regarding the second approach, some rules to alleviate the problem of multicollinearity are: (i) Drop one of the highly collinear regressor. If model has two or more regressors with high VIF, drop one from the model, because it supplies redundant information. Dropping one of the correlated regressor usually does not drastically reduce the  $R^2$ . However, omission of relevant regressor(s) from the model, may result in a specification error. Hence, the remedy may be worse than the disease in some situations, because, multicollinearity may prevent the precise estimation of parameters of the regression model. Therefore, omitting some regressor(s) may seriously mislead to the true values of the parameters (Gujarati and Porter, 2008, pg. 344). (ii) Use an appropriate experimental design and increase the sample if possible. However, obtaining additional or better data is not always easy. (iii) Transform the regressors (iv) Use some alternative methods to the OLS such as principal component regression and ridge regression etc. to control variance and instability of the OLS estimates. (v) Use stepwise regression, best subset regression or specialized knowledge of the dataset to remove the redundant regressors. (vi) Combine the redundant variables, if possible.

## Summary

Strong linear relationship among regressors i.e. , the issue of multicollinearity results in unstable estimated regression coefficients and other inadequate statistical measures. Therefore, its severity should be tested. An R package, **mctest** has been designed with the goal of providing the most widely used and discussed collinearity diagnostic related statistics. Two main functions **omcdiag()** and **imcdiag** facilitate the users to get information about the existence of collinearity among regressors and also to get idea about which regressor may be the reason of multicollinearity. A function, **mc.plot()** can also be used to detect existence of collinearity among regressors by drawing a graph of VIF and eigenvalues. For further details about use of the said package and related functions, interested readers are referred to the documentation of the package.

## Bibliography

- D. Asteriou and S. G. Hall. *Applied Econometrics: A Modern Approach using Eviews and Microfit*. Palgrave Macmillan, New York, 2007. [p500]
- D. A. Belsley, E. Kuh, and R. E. Welsch. *Diagnostics: Identifying Influential Data and Sources of Collinearity*. John Wiley & Sons, New York, 1980. chap. 3. [p499, 500]
- O. J. Blanchard. Commnet. *Journal of Business and Economic Statistics*, 5:449–451, 1967. [p506]
- V. Calcagno. *glmulti: Model Selection and Multimodel Inference Made Easy*, 2013. URL <https://CRAN.R-project.org/package=glmulti>. R package version 1.0.7. [p502]
- S. Chatterjee and A. S. Hadi. *Regression Analysis by Example*. John Wiley & Sons, 4th edition, 2006. [p500]
- S. Chatterjee and B. Price. *Regression Analysis by Examples*. John Wiley & Sons, New York, 1977. [p501]
- W. W. A. Cooley and P. R. A. Lohnes. *Multivariate Data Analysis*. John Wiley & Sons, Australia, 1971. ISBN 9780471170600. [p500]
- J. D. Curto and J. C. Pinto. The corrected VIF (CVIF). *Journal of Applied Statistics*, 38(7):1499–1507, 2011. [p499, 501]
- W. R. Dillon and M. Goldstein. *Multivariate Analysis: Methods and Applications*. John Wiley & Sons, 1984. [p501]
- D. Dorsett, R. F. Gunst, and E. C. J. Gartland. Multicollinear effects of weighted least squares regression. *Statistics & Probability Letters*, 1(4):207–211, 1983. [p499]
- J. Faraway. *faraway: Functions and Datasets for Books by Julian Faraway*, 2016. URL <https://CRAN.R-project.org/package=faraway>. R package version 1.0.7. [p502]
- D. E. Farrar and R. R. Glauber. Multicollinearity in regression analysis: The problem revisited. *The Review of Economics and Statistics*, 49:92–107, 1967. [p499, 500, 502]
- J. Fox and S. Weisberg. *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, second edition, 2011. URL <http://socserv.socsci.mcmaster.ca/jfox/Books/Companion>. [p502]

- W. H. Greene. *Econometric Analysis*. Prentic-Hall, New Jersey, 5th edition, 2002. [p499, 501]
- D. N. Gujarati and D. C. Porter. *Basic Econometrics*. McGraw Hill, 5 edition, 2008. [p499, 500, 501, 502, 507]
- R. F. Gunst and R. L. Mason. Advantages of examining multicollinearities in regression analysis. *Biometrics*, 33:249–260, 1977. [p499]
- A. Hald. *Statistical Theory with Engineering Applications*. John Wiley & Sons, New York, 1952. [p503]
- F. E. Harrell Jr. *rms: Regression Modeling Strategies*, 2016. URL <https://CRAN.R-project.org/package=rms>. R package version 4.5-0. [p502]
- R. M. Heiberger. *HH: Statistical Analysis and Data Display: Heiberger and Holland*, 2016. URL <http://CRAN.R-project.org/package=HH>. R package version 3.1-32. [p502]
- J. Hendrickx. *perturb: Tools for Evaluating Collinearity*, 2012. URL <https://CRAN.R-project.org/package=perturb>. R package version 2.05. [p502]
- M. Imdadullah and M. Aslam. *mctest: Multicollinearity Diagnostic Measures*, 2016. URL <https://CRAN.R-project.org/package=mctest>. R package version 1.1. [p499]
- J. Johnston. *Econometric Methods*. McGraw Hill, New York, 1963. [p499]
- G. Judge, W. Griffiths, H. Lutkepohl, and T. Lee. *The Theory and Practice of Econometrics*. John Wiley & Sons, 1985. [p500]
- M. G. Kendall. *A Course in Multivariate Analysis*. Griffin, London, 1957. pp. 70–75. [p499, 501]
- L. R. Klein. *An Introduction to Econometrics*. Prentic-Hall, Englewood, Cliffs, N. J., 1962. pp. 101. [p499, 501]
- A. Koutsoyiannis. *Theory of Econometrics*. Macmillan Education Limited, 1977. [p499]
- P. Kovács, T. Petres, and Tóth. A new measure of multicollinearity in linear regression models. *International Statistical Review / Revue Internationale de Statistique*, 73(3):405–412, 2005. [p499, 500, 501]
- M. H. Kutner, C. J. Nachtsheim, and J. Neter. *Applied Linear Regression Models*. McGraw Hill Irwin, 4th edition, 2004. [p499, 501]
- D. Lin. *VIF: VIF regression: A Fast Regression Algorithm for Large Data*, 2012. URL <https://CRAN.R-project.org/package=VIF>. R package version 1.0. [p502]
- T. Lumley. *leaps: Regression Subset Selection using Fortran code by Alan Miller Including Exhaustive Search*, 2009. URL <https://CRAN.R-project.org/package=leaps>. R package version 2.9. [p502]
- G. S. Maddala. *Introduction to Econometrics*. Macmillan, New York, 1988. [p500]
- D. W. Marquardt. Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation. *Technometrics*, 12(3):591–612, 1970. [p499, 501]
- R. L. Mason, R. F. Gunst, and J. T. Webster. Regression analysis and problems of multicollinearity. *Communications in Statistics*, 4(3):277–292, 1975. [p499]
- A. McLeod and C. Xu. *bestglm: Best Bubset GLM*, 2014. URL <https://CRAN.R-project.org/package=bestglm>. R package version 0.34. [p502]
- Minitab, Inc. Minitab Statistical Software, Release 17 for Windows, 2014. State College, Pennsylvania. [p502]
- D. Montgomery and E. A. Peck. *Introduction to Linear Regression Analysis*. John Wiley & Sons, New York, 1982. [p499]
- B. Naimi. *usdm: Uncertainty Analysis for Species Distribution Models*, 2015. URL <https://CRAN.R-project.org/package=usdm>. R package version 1.1-15. [p502]
- M. Nakazawa. *fmsb: Functions for Medical Statistics Book with Some Demographic Data*, 2015. URL <https://CRAN.R-project.org/package=fmsb>. R package version 0.5.2. [p502]
- NCSS . NCSS 11 Statistical Software, 2016. URL [ncss.com/software/ncss](http://ncss.com/software/ncss). NCSS, LLC Paysville, Utah, USA. [p502]

- SAS 9.3. SAS Institute Inc., 2011. Cary, NC, USA. [p<sup>502</sup>]
- S. D. Silvey. Multicollinearity and imprecise estimation. *Journal of the Royal Statistical Society, Series B (Methodological)*, 31(3):539–552, 1969. [p<sup>499</sup>, <sup>501</sup>]
- StataCorp. Stata statistical software: Release 14, 2015. College Station, Texas 77845 USA. [p<sup>502</sup>]
- Statgraphics Centurion XVII. Statpoint Technologies, Inc., 2015. Warrenton, Virginia. [p<sup>502</sup>]
- J. H. Stock and M. W. Watson. *Introduction to Econometrics*. Pearson Addison-Wesley, 3rd edition, 2010. [p<sup>500</sup>]
- H. Theil. *Principles of Econometrics*. John Wiley & Sons, New York, 1971. [p<sup>499</sup>, <sup>501</sup>]
- H. Wickham. *meifly: Interactive Model Exploration using GGobi*, 2014. URL <https://CRAN.R-project.org/package=meifly>. R package version 0.3. [p<sup>502</sup>]
- M. S. Younger. *A Handbook for Linear Regression*. MA: Duxbury Resource Center, North Scituate, 1979. [p<sup>499</sup>]

*Muhammad Imdadullah*  
Ph.D scholar (Statistics)  
Department of Statistics  
Bahauddin Zakariya University, Multan, Pakistan  
[mimdadasad@gmail.com](mailto:mimdadasad@gmail.com)

*Muhammad Aslam*  
Department of Statistics  
Bahauddin Zakariya University, Multan, Pakistan  
[aslamasadi@bzu.edu.pk](mailto:aslamasadi@bzu.edu.pk)

*Saima Altaf*  
Department of Statistics  
Bahauddin Zakariya University, Multan, Pakistan  
[drsaimaaltaf27@gmail.com](mailto:drsaimaaltaf27@gmail.com)

# R Foundation News

by Torsten Hothorn

## Donations and new members

### New benefactors

INWT Statistics, Germany

### Donations

Radosav Andric, in memory of Ramiro Zurkowski, Canada Greater Good, Netherlands  
Koen-Woong Moon, Korea Daniel Neumann, Germany Schukat Electronic Vertriebs GmbH,  
Germany Somewhat Retired, USA Richard Vlasimsky, IMIDEX, USA

### New supporting institutions

Inference Technologies, Czech Republic

### New supporting members

Ayala S. Allon, Israel Michael Hahsler, USA Matthias Häni, Switzerland Christian Kohlberg,  
Germany Wojciech Niemczyk, Poland

*Torsten Hothorn*

*Universität Zürich, Switzerland* [Torsten.Hothorn@R-project.org](mailto:Torsten.Hothorn@R-project.org)

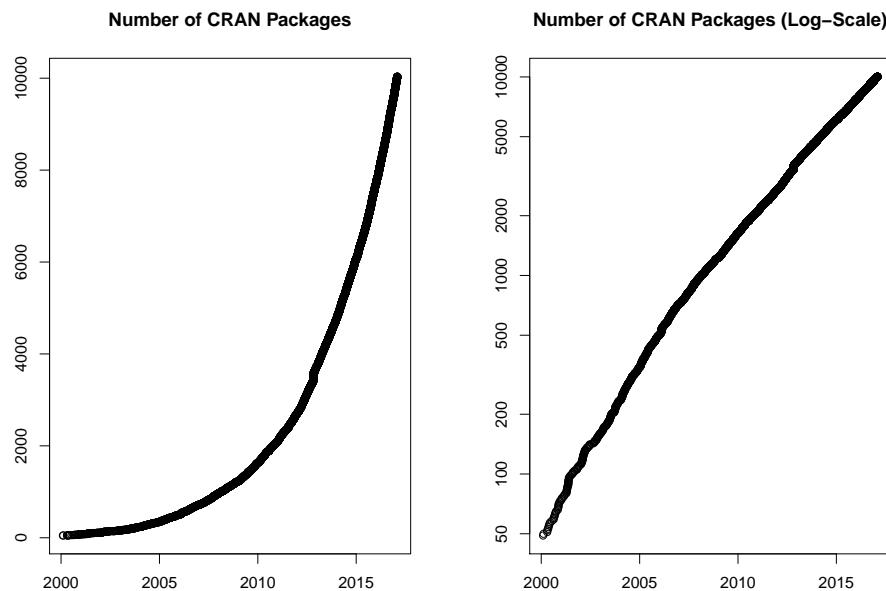
# Changes on CRAN

2016-08-01 to 2017-01-31

by Kurt Hornik and Achim Zeileis

## CRAN Growth

In the past 6 months, 1169 new packages were added to the CRAN package repository. 18 packages were unarchived and 36 archived. The following shows the growth of the number of active packages in the CRAN package repository:



Around 2017-01-27, the number of active packages went above 10000!

## New CRAN task views

*ExtremeValue* Topic: Extreme Value Analysis. Maintainer: Christophe Dutang, Kevin Jaunatre. Packages: [QRM](#), [RTDE](#), [ReIns](#), [SpatialExtremes](#), [VGAM](#), [copula](#), [evd](#)\*, [evdbayes](#), [evir](#)\*, [extRemes](#), [extremeStat](#), [extremefit](#), [fExtremes](#), [ismev](#), [lmom](#), [lmomRFA](#), [lmomco](#), [mev](#), [texmex](#).

(\* = core package)

## New packages in CRAN task views

*Bayesian* [BAS](#), [Boom](#), [BoomSpikeSlab](#), [LaplaceDemon](#), [abn](#), [bayesImageS](#), [bayesmeta](#), [bsts](#), [nimble](#).

*ChemPhys* [enpls](#), [wccsom](#).

*ClinicalTrials* [ThreeGroups](#).

*Cluster* [ADPclust](#), [CEC](#), [bmixture](#), [clustMixType](#), [edci](#), [largeVis](#).

*DifferentialEquations* [Sim.DiffProc](#).

*Distributions* Compositional, QRM, ReIns, bmixture.

*Econometrics* ExtremeBounds, clubSandwich, clusterSEs, decompr, gvc, pwt9, rdd, rdtools, rdrobust.

*ExperimentalDesign* BOIN, BayesMAMS, CombinS, GroupSeq, ICAOD, JMdesign, OBsMD, OptimaRegion, OptimalDesign, PGM2, PwrGSD, RPPairwiseDesign, SLHD, ThreeArmedTrials, VNM, acebayes, binseqtest, choiceDes, crmPack, designGLMM, designmatch, desplot, dfcomb, dfcrm, dfmta, dfpk, docopulae, dynaTree, easypower, ez, gset, hiPOD, ibd, minimaxdesign, optDesignSlopeInt, ph2bayes, ph2bye, pid, powerAnalysis, powerGWASinteraction, powerbydesign, qualityTools, seqDesign, ssize.fdr, ssizeRNA, vdg.

*Finance* Dowd, FinancialMath, GetHFDData, GetTDData, InfoTrad, MSGARCH, NetworkRiskMeasures, PortfolioEffectHFT, QuantTools, factorstochvol, fmdates, pinbasic, ragtop, sharpeRratio, tidyquant.

*HighPerformanceComputing* batchtools, pbapply, permGPU.

*MachineLearning* biglasso, gmmr.r, rnn, spa.

*MedicalImaging* Morpho\*, RNifti\*, Rvcg\*, adaptsmoFMRI, bayesImageS, divest\*, edfReader\*, eegkit\*.

*MetaAnalysis* MetaAnalyser, MetaIntegrator, esc, metaplotr.

*NaturalLanguageProcessing* PGRdup, gutenbergr, hunspell, monkeylearn, mscstexta4r, mscsweblm4r, phonics, quanteda, tesseract, text2vec, tidytext, tokenizers.

*NumericalMathematics* RSpectra, rmumps, schumaker.

*OfficialStatistics* BIFIEsurvey, CalibrateSSB, Frames2, GeomComb, MBHdesign, PracTools, RRTCS, RcmdrPlugin.sampling, gridsample, mapStats, panelaggregation, quantification, rpms, rspa, samplesize4surveys, srvyr, surveybootstrap, surveydata, surveyoutliers, svyPVPack.

*Phylogenetics* outbreaker, phyloTop, rmetasim, rotl.

*Psychometrics* BayesFM, BayesLCA, BigSEM, CAvariants, ClustVarLV, DistatisR, GDINA, IRTpp, LNIRT, LVMMCOR, LatentREGpp, MCAvariants, ML-CIRTwithin, SEMID, SOD, SparseFactorAnalysis, TestDataImputation, aspect, cIRT, cabootcrs, cds, cncaGUI, cocor, covLCA, ctsem, dlsem, edstan, elasticnet, emIRT, esaBcv, faoutlier, fourPNO, gSEM, gtheory, immer, influence.SEM, irtDemo, lba, lcda, lsl, ltbayes, nsprcomp, optiscale, paran, piecewiseSEM, plotSEMM, regsem, rsem, semGOF, semdiag, semtree, smds, soc.ca, sparseSEM, subscore, superMDS, xxIRT.

*SocialSciences* optmatch.

*Spatial* ExceedanceTools, RQGIS, dggridR\*, gear, geojson, geojsonio, ggsn, postGIS-tools, rgif, rpostgis, sf\*, smacpod, smerc, spacom, spselect.

*SpatioTemporal* CARBayesST, GeoLight, SimilarityMeasures.

*Survival* AHR, APtools, AdapEnetClass, BayesPiecewiseICAR, Biograph, CFC, Coxnet, Cyclops, ELYP, FamEvent, GORCure, GSSE, ICBayes, ICGOR, InformativeCensoring, JointModel, LTRCTrees, LexisPlotR, PenCoxFrail, SSRMST, SimHaz, Sim-SCRPiecewise, SurvCorr, SurvLong, SurvRank, SurvRegCensCov, TP.idm, Trans-Model, addhazard, asaur, bnnSurvival, cmprskQR, compareC, condSURV, controlTest, coxsei, crrp, crskdiag, discSurv, emplik2, fastpseudo, flexPM, flexrsurv, frailtySurv, gcerisk, glmnet, gte, hdnom, icRSF, icenReg, imputeYn, intercure, isoph, jackknifeKME, joint.Cox, landest, mexhaz, miCoPTCM, msmttools, npsurv, pch, plac, popEpi, ranger, reReg, reda, rstpm2, smcure, survMisc, survRM2, survminer, tdROC, thregI, tranSurv, uniah.

*TimeSeries*, **BETS**, **BigVAR**, **GAS**, **GeomComb**, **InspectChangepoint**, **Tcomp**, **WaveletComp**, **cointReg**, **dCovTS**, **dynr**, **ecm**, **factorstochvol**, **forecastHybrid**, **gdpc**, **ggseas**, **mclcar**, **onlineCPD**, **opera**, **pcdpca**, **pdc**, **robets**, **roll**, **rucrdtw**, **scoringRules**, **seasonalview**, **smooth**, **sparsevar**, **spectral**, **stR**, **thief**, **tsdisagg2**, **tswge**, **uroot**, **x13binary**.

*WebTechnologies* **ROpenFIGI**, **RSmartlyIO**, **RStripe**, **anametrix**, **dataone**, **datarobot**, **europepmc**, **fiery**, **geoparser**, **googleAnalyticsR**, **googleCloudStorageR**, **htmltab**, **jqr**, **jsonvalidate**, **mscstexta4r**, **mscsweblm4r**, **nomadlist**, **openadds**, **opencage**, **osi**, **osmplotr**, **placement**, **plumber**, **rgeospatialquality**, **rosetteApi**, **uaparserjs**.

(\* = core package)

*Kurt Hornik*

*WU Wirtschaftsuniversität Wien, Austria*

[Kurt.Hornik@R-project.org](mailto:Kurt.Hornik@R-project.org)

*Achim Zeileis*

*Universität Innsbruck, Austria*

[Achim.Zeileis@R-project.org](mailto:Achim.Zeileis@R-project.org)

# News from the Bioconductor Project

by *Bioconductor Core Team*

The [Bioconductor](#) project provides tools for the analysis and comprehension of high-throughput genomic data. Bioconductor 3.4 was released on 18 October, 2016. It is compatible with R 3.3 and consists of 1296 software packages, 309 experiment data packages, and 933 up-to-date annotation packages. The [release announcement](#) includes descriptions of 101 new packages, and updated NEWS files for many additional packages. Start using Bioconductor by installing the most recent version of R and evaluating the commands

```
source("https://bioconductor.org/biocLite.R")
biocLite()
```

Install additional packages and dependencies, e.g., [AnnotationHub](#), with

```
BiocInstaller::biocLite("AnnotationHub")
```

Docker and [Amazon](#) images provides a very effective on-ramp for power users to rapidly obtain access to standardized and scalable computing environments. Key resources include:

- [bioconductor.org](#) to install, learn, use, and develop Bioconductor packages.
- A listing of [available software](#), linked to pages describing each package.
- A question-and-answer style [user support site](#) and developer-oriented [mailing list](#).
- The [F1000Research Bioconductor channel](#) for peer-reviewed Bioconductor work flows.
- Our [package submission](#) repository for open technical review of new packages.

Our annual conference, [BioC 2017: Where Software and Biology Connect](#), will be on June 26 ('developer day'), 27 and 28, in Boston, MA.

*Bioconductor Core Team  
Biostatistics and Bioinformatics  
Roswell Park Cancer Institute, Buffalo, NY  
USA [maintainer@bioconductor.org](mailto:maintainer@bioconductor.org)*

# Changes in R

From version 3.2.4 to version 3.3.1 patched

by the R Core Team

## CHANGES IN R 3.3.2 patched

### NEW FEATURES

- The internal methods of `download.file()` and `url()` now report if they are unable to follow the redirection of a ‘`http://`’ URL to a ‘`https://`’ URL (rather than failing silently).

### INSTALLATION on a UNIX-ALIKE

- The `configure` check for the `zlib` version is now robust to versions longer than 5 characters, including 1.2.10.

### UTILITIES

- Environmental variable `_R_CHECK_TESTS_NLINES_` controls how `R CMD check` reports failing tests (see §8 of the ‘R Internals’ manual).

### BUG FIXES

- `rep(x, times)` and `rep.int(x, times)` now both work also when `times` is larger than the maximal integer, including when it is of length greater than one. ([PR#16932](#))
- `vapply(x, *)` now works with long vectors `x`. ([PR#17174](#))
- `isS3method("is.na.data.frame")` and similar are correct now. ([PR#17171](#))
- `grepRaw(<long>, <short>, fixed = TRUE)` now works, thanks to a patch by Mikko Korpela. ([PR#17132](#))
- Package installation into a library where the package exists via symbolic link now should work wherever `()` works, resolving [PR#16725](#).
- “Cincinnati” was missing an “n” in the `precip` dataset.
- Fix buffer overflow vulnerability in `pdf()` when loading an encoding file. Reported by Talos (TALOS-2016-0227).
- `getDLLRegisteredRoutines()` now produces its warning correctly when multiple DLLs match, thanks to Matt Dowle’s [PR#17184](#).
- `Sys.timezone()` now returns non-NA also on platforms such as ‘Ubuntu 14.04.5 LTS’, thanks to Mikko Korpela’s [PR#17186](#).
- `format(x)` for an illegal “`POSIXlt`” object `x` no longer segfaults.
- `methods(f)` now also works for `f` “`(`” or “`{`”.
- (Windows only) `dir.create()` did not check the length of the path to create, and so could overflow a buffer and crash R. ([PR#17206](#))
- On some systems, very small hexadecimal numbers in hex notation would underflow to zero. ([PR#17199](#))

- `pmin()` and `pmax()` now work again for ordered factors and 0-length S3 classed objects, thanks to Suharto Anggono's [PR#17195](#) and [PR#17200](#).
- `bug.report()` did not do any validity checking on a package's 'BugReports' field. It now ignores an empty field, removes leading whitespace and only attempts to open '`http://`' and '`https://`' URLs, falling back to emailing the maintainer.
- Bandwidth selectors `bw.ucv()` and `bw.SJ()` gave incorrect answers or incorrectly reported an error (because of integer overflow) for inputs longer than 46341. Similarly for `bw.bcv()` at length 5793.  
Another possible integer overflow is checked and may result in an error report (rather than an incorrect result) for much longer inputs (millions for a smooth distribution).
- `findMethod` failed if the active signature had expanded beyond what a particular package used. (Example with packages `XR` and `XRJulia` on CRAN).
- `qbeta()` underflowed too early in some very asymmetric cases. ([PR#17178](#))

## CHANGES IN R 3.3.2

### NEW FEATURES

- `extSoftVersion()` now reports the version (if any) of the `readline` library in use.
- The version of LAPACK included in the sources has been updated to 3.6.1, a bug-fix release including a speedup for the non-symmetric case of `eigen()`.
- Use `options(deparse.max.lines=)` to limit the number of lines recorded in .Traceback and other deparsing activities.
- `format(<AsIs>)` looks more regular, also for non-character atomic matrices.
- `abbreviate()` gains an option named = TRUE.
- The online documentation for package **methods** is extensively rewritten. The goals are to simplify documentation for basic use, to note old features not recommended and to correct out-of-date information.
- Calls to `setMethod()` no longer print a message when creating a generic function in those cases where that is natural: S3 generics and primitives.

### INSTALLATION and INCLUDED SOFTWARE

- Versions of the `readline` library  $\geq 6.3$  had been changed so that terminal window resizes were not signalled to `readline`: code has been added using a explicit signal handler to work around that (when R is compiled against `readline`  $\geq 6.3$ ). ([PR#16604](#))
- `configure` works better with Oracle Developer Studio 12.5.

### UTILITIES

- `R CMD check` reports more dubious flags in files 'src/Makevars[.in]', including '-w' and '-g'.
- `R CMD check` has been set up to filter important warnings from recent versions of `gfortran` with '-Wall -pedantic': this now reports non-portable GNU extensions such as out-of-order declarations.
- `R CMD config` works better with paths containing spaces, even those of home directories (as reported by Ken Beath).

## DEPRECATED AND DEFUNCT

- Use of the C/C++ macro ‘NO\_C\_HEADERS’ is deprecated (no C headers are included by R headers from C++ as from R 3.3.0, so it should no longer be needed).

## BUG FIXES

- The check for non-portable flags in R CMD check could be stymied by ‘src/Makevars’ files which contained targets.
- (Windows only) When using certain desktop themes in Windows 7 or higher, Alt-Tab could cause Rterm to stop accepting input. ([PR#14406](#); patch submitted by Jan Gleixner.)
- `pretty(d, ...)` behaves better for date-time d ([PR#16923](#)).
- When an S4 class name matches multiple classes in the S4 cache, perform a dynamic search in order to obey namespace imports. This should eliminate annoying messages about multiple hits in the class cache. Also, pass along the package from the `ClassExtends` object when looking up superclasses in the cache.
- `sample(NA_real_)` now works.
- Packages using non-ASCII encodings in their code did not install data properly on systems using different encodings.
- `merge(df1,df2)` now also works for data frames with column names “`na.last`”, “`decreasing`”, or “`method`”. ([PR#17119](#))
- `contour()` caused a segfault if the `labels` argument had length zero. (Reported by Bill Dunlap.)
- `unique(warnings())` works more correctly, thanks to a new `duplicated.warnings()` method.
- `findInterval(x, vec = numeric(), all.inside = TRUE)` now returns 0s as documented. (Reported by Bill Dunlap.)
- (Windows only) R CMD SHLIB failed when a symbol in the resulting library had the same name as a keyword in the ‘.def’ file. ([PR#17130](#))
- `pmax()` and `pmin()` now work with (more ?) classed objects, such as “`Matrix`” from the `Matrix` package, as documented for a long time.
- `axis(side, x = D)` and hence `Axis()` and `plot()` now work correctly for “`Date`” and time objects `D`, even when “time goes backward”, e.g., with decreasing `xlim`. (Reported by William May.)
- `str(I(matrix(..)))` now looks as always intended.
- `plot.ts()`, the `plot()` method for time series, now respects `cex`, `lwd` and `lty`. (Reported by Greg Werbin.)
- `parallel::mccollect()` now returns a named list (as documented) when called with `wait = FALSE`. (Reported by Michel Lang.)
- If a package added a class to a class union in another package, loading the first package gave erroneous warnings about “undefined subclass”.
- `c()`’s argument `use.names` is documented now, as belonging to the (C internal) default method. In “`parallel`”, argument `recursive` is also moved from the generic to the default method, such that the formal argument list of `base` generic `c()` is just (...).

- `rbeta(4,NA)` and similarly `rgamma()` and `rbinom()` now return NaN's with a warning, as other `r<dist>()`, and as documented. ([PR#17155](#))
- Using `options(checkPackageLicense = TRUE)` no longer requires acceptance of the licence for non-default standard packages such as `compiler`. (Reported by Mikko Korpela.)
- `split(<very_long>,*)` now works even when the split off parts are long. ([PR#17139](#))
- `min()` and `max()` now also work correctly when the argument list starts with `character(0)`. ([PR#17160](#))
- Subsetting very large matrices (`prod(dim(.)) >= 2^31`) now works thanks to Michael Schubmehl's [PR#17158](#).
- `bartlett.test()` used residual sums of squares instead of variances, when the argument was a list of `lm` objects. (Reported by Jens Ledet Jensen).
- `plot(<lm>,which = *)` now correctly labels the contour lines for the standardized residuals for `which = 6`. It also takes the correct `p` in case of singularities (also for `which = 5`). ([PR#17161](#))
- `xtabs(~ exclude)` no longer fails from wrong scope, thanks to Suharto Anggono's [PR#17147](#).
- Reference class calls to `methods()` did not re-analyse previously defined methods, meaning that calls to methods defined later would fail. (Reported by Charles Tilford).
- `findInterval(x, vec, left.open = TRUE)` misbehaved in some cases. (Reported by Dmitriy Chernykh.)