The R Journal

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

Contents

Contributed Research Articles

News and Notes

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

Editor-in-Chief:
Simon Urbanek, University of Auckland, New Zealand

Executive editors:

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

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

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

0.1 NA

Editorial

by Simon Urbanek

On behalf of the editorial board, I am pleased to present Volume 15 Issue 2 of the R Journal.

Behind the scenes, several people assist with the journal operations. Mitchell O'Hara-Wild continues to work on infrastructure, H. Sherry Zhang continues to develop the rjtools package under the direction of Professor Dianne Cook. In addition, articles in this issue have been carefully copy edited by Adam Bartonicek and Chase Robertson.

In this issue

This issue features 18 contributed research articles the majority of which relate to R packages on a diverse range of topics. All packages are available on CRAN. Supplementary material with fully reproducible code is available for download from the Journal website. Topics covered in this issue are

Graphics and Visualisation

- langevitour: smooth interactive touring of high dimensions, demonstrated with scRNA-Seq data
- ggdensity: Improved Bivariate Density Visualization in R
- Taking the Scenic Route: Interactive and Performant Tour Animations
- vivid: An R package for Variable Importance and Variable Interactions Displays for Machine Learning Models

Multivariate Statistics

- Generalized Estimating Equations using the package glmtoolbox
- genpathmox: An R Package to Tackle Numerous Categorical Variables and Heterogeneity in Partial Least Squares Structural Equation Modeling

Bayesian Inference

- bqror: An R package for Bayesian Quantile Regression in Ordinal Models
- A framework for estimating and visualising excess mortality during the COVID-19 pandemic

Social Sciences

- PINstimation: An R Package for Estimating Models of Probability of Informed Trading
- mutualinf: An R Package for Computing and Decomposing the Mutual Information Index of Segregation
- Three-way Correspondence Analysis in R
- Difficult Choices? Estimating Heteroskedastic and Instrumental Variable Models for Binary Dependent Variables in R

Mixture Models and Optimization

- nlstac: Non-gradient Separable Nonlinear Least Squares Fitting
- Univariate Gaussian mixtures in R

Clustering and Graphs

- Identifying Counterfactual Queries with the R package cfid
- clustAnalytics: An R Package for Assessing Stability and Significance of Clusters in Networks

Other

- hydrotoolbox: a Package for Hydrometeorological Data Management
- EviewsR: an R Package for Dynamic and Reproducible Research Using EViews, R, R Markdown and Quarto

Simon Urbanek
University of Auckland

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

nlstac: Non-Gradient Separable Nonlinear Least Squares Fitting

by J. A. F. Torvisco, R. Benítez, M. R. Arias, and J. Cabello Sánchez

Abstract A new package for nonlinear least squares fitting is introduced in this paper. This package implements a recently developed algorithm that, for certain types of nonlinear curve fitting, reduces the number of nonlinear parameters to be fitted. One notable feature of this method is the absence of initialization which is typically necessary for nonlinear fitting gradient-based algorithms. Instead, just some bounds for the nonlinear parameters are required. Even though convergence for this method is guaranteed for exponential decay using the max-norm, the algorithm exhibits remarkable robustness, and its use has been extended to a wide range of functions using the Euclidean norm. Furthermore, this data-fitting package can also serve as a valuable resource for providing accurate initial parameters to other algorithms that rely on them.

1 Introduction

Experimental data often exhibits non-linear patterns. As such, researchers in applied science often have to try to fit these data with non-linear models which can be challenging to fit. In this paper, we introduce the **nlstac** package (Rodriguez-Arias et al., 2023), which implements the TAC algorithm for solving separable nonlinear regression problems, among others. Unlike other solvers, it does not require initialization values. Throughout the paper, we emphasize the potential synergistic usage of **nlstac** alongside other commonly used solvers, as it can provide reliable initialization values for them. The syntax of **nlstac** follows a similar structure to the solvers in the **minpack.lm** package (Elzhov et al., 2023), making it familiar to researchers experienced with those solvers.

The motivation behind developing the **nlstac** package stems from an approximation problem involving time series data. Specifically, we were working with data corresponding to measurements obtained from a thermometer reaching thermal equilibrium with the surrounding medium, particularly oceanic water. In accord with Newton's law of cooling, the temporal evolution of these data exhibits an exponential pattern described by the expression:

$$a_1 e^{-k_1 t} + a_2, \text{ where } a_1, a_2, k_1 \in \mathbb{R}, \ k_1 > 0, \tag{1}$$

where t represents the time variable.

Fitting data with the exponential pattern described by equation (1) is a nonlinear optimization problem. One challenge we encountered with widely used algorithms for such problems was the need to initialize the parameters in (1), as the solutions often strongly depended on the chosen initial values —a bad choice of initial values could lead to a sub-optimal local minimum or even make the algorithm not to converge at all. The TAC algorithm, around which the present package is built, is presented in Torvisco et al. (2018) and it overcomes this issue by eliminating the requirement for parameter initialization. It only needs to specify a broad interval in which to search for the nonlinear parameters. As we worked with the TAC algorithm, its robustness became increasingly evident—robustness in the sense of stability of the algorithm in relation with noisy data and the convergence for a great variety of problems. In our opinion, this advantage, along with the lack of initialization, outweighs the need to specify the exact pattern to be used.

While the convergence of TAC is proven using the max norm, as shown in Torvisco et al. (2018), we employ the Euclidean norm in the **nlstac** package and consider more general patterns beyond equation (1). This extension of TAC beyond its proven convergence conditions is supported by its reliable performance, as mentioned earlier.

We acknowledge the widespread use of other algorithms for nonlinear fitting, such as Gauss-Newton or Levenberg-Marquardt, with the former being the default choice for the `nls` function in the **stats** package (R Core Team, 2021). Hence, in the present paper, we aimed to showcase the similarities and differences between **nlstac** and the `nls` fit, not as a competition, but as a demonstration of how well these two algorithms can work in synergy. Researchers sometimes encounter difficulties in finding suitable initialization values for `nls` to achieve convergence. In this regard, since **nlstac** does not require users to specify good starting values to converge, the resulting estimates can be used to provide good starting values to `nls` or any other initialization-dependent algorithm.

1.1 Nonlinear regression, or more broadly, a problem of approximation

Nonlinear regression or nonlinear fitting is a standard procedure commonly used in a wide range of scientific fields. Typically, we start with a dataset of q observations of n regressors (or predictor variables) and a response variable, namely $\{(\mathbf{x}_i, y_i), i = 1, \ldots, q\}$, and a mathematical expression relating the regressors and the response variable. This mathematical expression may present a nonlinear dependency on several parameters. For instance, the aforementioned general expression is usually given by

$$y = f(\mathbf{x}; \boldsymbol{\theta}) + \varepsilon(\mathbf{x}; \boldsymbol{\theta}), \tag{2}$$

being $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$ a function, $\varepsilon(\mathbf{x}; \cdot)$, independent and identically distributed random variables following a spherical normal distribution and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$ being the vector of parameters. Thus, the problem reduces to finding an estimate of the parameter vector $\boldsymbol{\theta}^*$ such that some cost function is minimized. A usual choice for the cost function is the well-known least-squares cost function so that we are solving an optimization problem. Namely, finding $\boldsymbol{\theta}^* \in \mathbb{R}^p$ such that

$$g(\boldsymbol{\theta}^*) = \min_{\boldsymbol{\theta}} g(\boldsymbol{\theta}), \quad g(\boldsymbol{\theta}) = \sum_{i=1}^{q} (y_i - f(\mathbf{x}_i; \boldsymbol{\theta}))^2. \tag{3}$$

Please observe it is assumed that only y_i's are observed with error, whereas x_i's are measured exactly. The possibility of generalizing this to measurement-error models is not implemented in **nlstac**. More information about Least Squares Problems can be found in, for example, Björck (1996) or Nocedal and Wright (2006).

The above problem can be described in Mathematical Analysis as an approximation problem. This kind of problem is determined by three elements: a set A, which is the object to be approximated; a family of functions $\mathcal{F}$, whose elements are known as approximants; and finally an approximation criterion—a procedure to measure how close to A each element of $\mathcal{F}$ lies. In an approximation problem, a canonical question arises: does exist an element $f \in \mathcal{F}$ which is closest to A —attending to the approximation criterion— than every other element in $\mathcal{F}$? When the answer to this question happens to be affirmative, a method to locate one of such elements is needed and this is what **nlstac** is designed for.

Let us identify those elements in the problem described above. The element A to be approximated is the dataset of observations $\{(\mathbf{x}_i, y_i), i = 1, \ldots, q\}$, which is a subset of $\mathbb{R}^n \times \mathbb{R}$. The family of approximants, $\mathcal{F}$, is given by the following p-parametric family

$$\mathcal{F} = \left\{ f_{\boldsymbol{\theta}} : \mathbb{R}^n \longmapsto \mathbb{R} \,|\, f_{\boldsymbol{\theta}}(\mathbf{x}) = f(\mathbf{x}; \boldsymbol{\theta}), \, \boldsymbol{\theta} \in \mathbb{R}^p \right\},$$

being f the mathematical expression relating the regressors and the response variable given in (2). Finally, the approximation criterion corresponds to the function g in (3). From now on, we will refer to one or the other definition depending on which one is more clear within the context.

Algorithms for finding the best set of parameters $\boldsymbol{\theta}^*$ of (3) can be divided into local solvers and global solvers, depending on whether they are designed to find a local or a global minimum of the cost function, respectively. In general, local solvers are, under certain conditions, fast and accurate. They are usually based on some sort of gradient descent algorithm and are iterative in nature. That is, they start at a given initial guess for θ and, at each iteration (hopefully) they find a better approximation of the minimum. Under some assumptions (e.g. convexity), the local and global minima may coincide. However, in many cases, we will find that there are many different local minima and, consequently, the solution given by those algorithms may depend on the initial guess. Therefore, a bad initial guess could land us in a sub-optimal local minimum and there are even cases for which that initial conditions may cause the algorithm to not converge. Some local algorithms are the steepest descent method, incremental steepest descent method, Newton's method, Quasi-Newton methods, Newton's methods with Hessian modification, BFGS algorithm, Gauss-Newton method, or Levenberg-Marquardt method. More information about this and other methods can be found in, for example, Nocedal and Wright (2006), Arora (2015) or Rhinehart (2016).

On the other hand, global solvers do not depend that heavily on an initial condition, but they require an interval or area in which to start looking for the minimum. Some global algorithms, like grid-search, are known to converge in any case, but they scale very poorly, such that the computational time grows exponentially with the number of parameters to be found. There are also heuristic solvers, which are not guaranteed to converge to global minimum, but they give a reasonable approximation in cases where other algorithms either take too long or do not converge at all. Some examples of this last kind are Nelder–Mead method, genetic algorithms, particle swarm optimization, simulated annealing, or ant colony optimization, see Arora (2015).

1.2 Separable nonlinear regression problems

Some of the previous problems fall into the family of **separable nonlinear regression** problems. In this kind of problem, the nonlinear function f in (2) can be written as a linear combination of nonlinear functions. In particular, f takes the expression

$$f(\mathbf{x}; \boldsymbol{\theta}) = \sum_{i=1}^{r} a_i \phi_i(\mathbf{x}; \mathbf{b}). \tag{4}$$

With this formulation, the original set of parameters $\boldsymbol{\theta}$ has been split into two subsets: the linear parameters $\mathbf{a} = (a_1, \ldots, a_r)$ and the nonlinear parameters $\mathbf{b} = (b_1, \ldots, b_s)$. Obviously $r + s = p$ and thus the number of nonlinear parameters to be determined is smaller than the total number of parameters. Therefore, the number of parameters to be found using nonlinear algorithms can be reduced. Separable nonlinear least squares methods are described, for example, in Golub and Pereyra (2003) and Golub and Pereyra (1973).

In this work we present the package **nlstac**, based on the TAC algorithm described in Torvisco et al. (2018) for solving the separable nonlinear regression problem given in (4).

1.3 Related packages

There are some widely used R packages that also deal with the problems **nlstac** is designed to solve. However, they rely on different algorithms which are often dependent on the choice of initial values. These packages are mainly **nlsr** (Nash and Murdoch, 2023) and **minpack.lm**, both solving the problems with variants of the Levenberg-Marquardt algorithm, **lbfgs** package (Coppola et al., 2022), which provides an interface to L-BFGS and OWL-QN algorithms, or **minqa** package (Bates et al., 2014), implementing derivative-free optimization algorithms. The algorithms used by this package fall into the aforementioned category of local algorithms.

Other R packages that use global algorithms, mostly of a stochastic nature, are, for example, **DEoptimR** (Conceicao, 2022), **GenSA** (Xiang et al., 2013), **GA** (Scrucca, 2013), **ABCoptim** (Vega Yon and Muñoz, 2017) or **pso** (Bendtsen., 2022).

There are also packages for fitting models in separable non-linear regression models, although these tend to be more specialized for specific problem domains. For example, the **TIMP** package (Mullen and van Stokkum, 2007) is used for physics and chemistry problems whereas **spant** package (Wilson, 2021) deals with magnetic resonance spectroscopy problems. For partially separable nonlinear fitting we find **psqn** package (Christoffersen, 2022).

2 The nlstac package

The **nlstac** package was developed with two objectives: first, to implement the algorithm described in Torvisco et al. (2018) in functions that could be used for estimating separable nonlinear regression models, and second, to implement these functions with standard syntax such that they would be convenient for users familiar with other curve-fitting functions such as `lm`, `nls`, or `nlsLM` from the **minpack.lm** package.

The package consists of three units: a formula decomposer, a linear least squares solver, and a grid search unit.
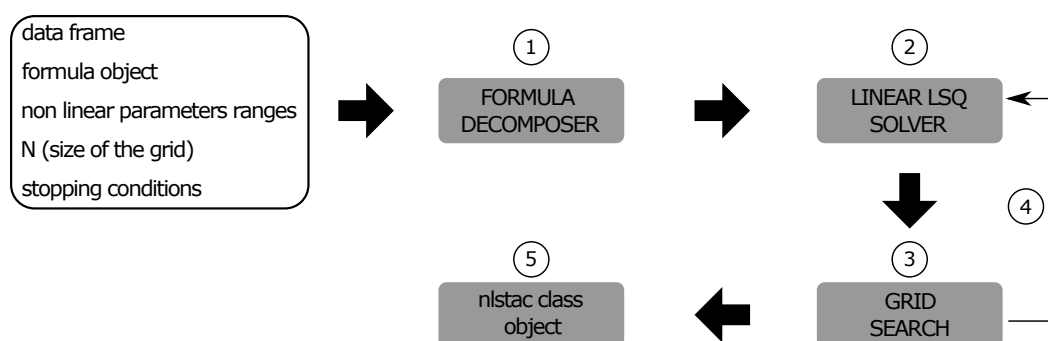
The workflow is depicted in Figure 1:

Figure 1: Schematic workflow of the algorithm used in the **nlstac** package.

1. From the dataset contained in a `dataframe`, an object of class `formula`, and a list of nonlinear parameters along with the initial ranges defined for each parameter, the formula decomposer, coded in internal function `get_functions` determines the nonlinear functions, ϕ_i, defined in (4).

2. From the ranges (intervals) of each nonlinear parameter and the number of nodes, N, of each partition of such intervals, all possible combinations of nonlinear parameters are determined. For each combination, a linear least square problem is solved, obtaining thus a set of *plausible parameters*. This is done by the `get_best_parameters` internal function. Such a set of 'plausible parameters' is obtained in the following way: Let $b_1, \ldots, b_s$ be the nonlinear parameters in (4). For each nonlinear parameter b_i let $[c_i, d_i]$ denote the interval where to seek the estimation of b_i and let $c_i = b_i^1 < b_i^2 < \cdots < b_i^{N-1} < b_i^N = d_i$ denote the partition of such interval. Then, from these partitions, we construct a mesh in the rectangle $[c_1, d_1] \times \cdots \times [c_s, d_s]$. Next, for each node of the grid, we obtain the optimal linear parameters by solving a linear least square problem. The nodes of the grid, along with the optimal linear parameters for each node, constitute the set of *plausible parameters*.

3. For each set of plausible parameters, the loss function (namely the sum of the squares of the residuals) is computed, and a grid search is performed to obtain the minimum value of the loss function. Let $(b_1^{m_1}, \ldots, b_s^{m_s})$ be the node of the grid in which the loss function minimizes.

4. Stopping criteria are met when either the maximum number of iterations is reached or when the size of the partition of every interval $[c_i, d_i]$ is lower than the tolerance level. If stopping criteria are not met, the grid is refined: for each parameter, b_i, a new subinterval where to seek the estimation of such parameter is considered, $[b_i^{m_i-1}, b_i^{m_i+1}]$ (note that if m_i is either 1 or N, the new subinterval will be $[c_i, b_i^2]$ or $[b_i^{N-1}, d_i]$, respectively). The new grid will be established by repeating steps 2 to 4 until one stopping criterion is met.

5. When stopping criteria are met, the result is returned as an object of class `nlstac`.

As indicated in step 5, the output given by the `nls_tac` function is an object of class `nlstac`. It is a list containing the following fields:

- `resid`: The residuals.
- `data`: The original data.
- `coefficients`: A named vector containing the values of the parameters.
- `stdError`: A named vector with the standard error of the estimation of the coefficients.
- `convInfo`: Convergence information. Namely, the number of iterations (`niter`), and the tolerance reached (`tolerance`).
- `SSR`: The sum of the squares of the residuals obtained by the fit.
- `fitted`: A vector containing the fitted values.
- `dataset`: A string with the name of the variable containing the data.
- `formula`: The formula used in the call of the function.
- `df`: The degrees of freedom
- `sigma`: The standard deviation estimate.
- `Rmat`: R matrix in the QR decomposition of the gradient matrix used for the computation of the standard errors of the coefficients.

The class `nlstac` has also some extraction methods similar to `lm`, `nls`, `glm`. For instance, the methods `summary.nlstac`, `predict.nlstac`, and `predict.summary.nlstac` produce identically formatted output as the `summary` functions for the `lm` and `nls` fits, as will be shown later.

The **nlstac** package (Rodriguez-Arias et al., 2023) is available in CRAN. The development version of the package can also be installed from the GitHub repository using the `install_github` function from the remotes package (Csárdi et al., 2021): `remotes::install_github("rbensua/nlstac")`.

2.1 Arguments

As was mentioned above, the inputs for the `nls_tac` function are, at least, the fields `data`, `formula`, `tol`, `N` and `nlparam`.

The `data` field is a data frame containing the data to be fitted; `tol` is the tolerance measured as the relative difference between the values of the parameters in two consecutive iterations; its default value is 10^{-4}; `N` is the number of divisions we make in each nonlinear parameter interval in each iteration (defaults to 10); `formula` is either an object of `formula` class or a character string that can be coerced into a `formula`, and it must contain the pattern or formula which will be fitted, and `nlparam` is a list containing the nonlinear parameters as well as their initial ranges.

2.2 Summary of nlstac class

Information about the fit is stored in an `nlstac` class, and the function `summary` can display the most numerical relevant information about the fit.

Considering the example shown in Subsection 2.3.1 (Example 1. Exponential Decay), once the analysis is done, the `summary` function shows us the information of the analysis as follows:

```
  > summary(tacfit)


Formula: temp ~ a1 * exp(-k1 * time) + a2


Parameters:
      Estimate   Std. Error  t value   Pr(>|t|)
k1 1.399458e-02 8.107657e-05 172.6095 < 2.22e-16 ***
a1 4.951112e+01 1.447617e-01 342.0182 < 2.22e-16 ***
a2 2.382372e+01 5.877739e-02 405.3212 < 2.22e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


Residual standard error: 0.1647017 on 180 degrees of freedom


Number of iterations to convergence:  13
Achieved convergence tolerance:  2.87864e-08
```

As can be seen, function `summary` gives us information about the formula used in the fitting, the estimated parameters along with some statistical information, the residual standard error, the number of iterations necessary to achieve convergence, and, finally, the tolerance value when convergence is achieved.

3 Examples

In this section, we present several examples to illustrate the use of the **nlstac** package in various scenarios. Each example highlights different aspects of **nlstac**'s behavior compared to another commonly used R function for these types of problems: `nls`, which is included in the **stats** package and utilizes the Gauss-Newton algorithm by default.

For each example, we explore two different initializations for `nls`. First, we initialize `nls` with a reasonable set of initial values. Note that the actual values of the parameters are unknown and no *a priori* estimation of these values are available; therefore we denote as reasonable a set of values which are similar to the estimation obtained by TAC. This approach allows us to compare the fit achieved by **nlstac** with the widely used `nls` function. In the second initialization approach, we initialize `nls` with the estimation provided by **nlstac** for the same problem. This second approach serves as a starting point to ease the convergence of the algorithm used by `nls` and enables us to observe how effectively both algorithms can work in tandem. These examples shed light on the versatility and potential of the **nlstac** package in conjunction with the established `nls` function, providing valuable insights into their combined performance.

In Subsection 2.3.1 and 2.3.2 two examples with real data are presented. In both cases, the **nlstac** package obtains a solution, while the `nls` function does not converge with seemingly reasonable set of initial values. However, if the `nls` function is initialized with the output of **nlstac**, it successfully converges to a solution. In the first example, the fit remains the same, and in the second one, `nls` slightly improves the fit obtained by **nlstac**. These examples highlight the versatility of **nlstac**, which can be used either independently for fitting or to provide accurate initialization values for `nls` to converge effectively.

In Subsection 2.3.3 we present an example with simulated data. In this example, way beyond TAC proven convergence, we obtain a good fit to the model when running the **nlstac** package. However, if we initialize the `nls` function with seemingly reasonable set of initial values, it fails to converge. Nevertheless, by using the output of **nlstac** as initialization values for `nls`, we achieve an even better fit than what **nlstac** alone provides. This example also shows that **nlstac** can serve not only as a standalone fitting algorithm but also as a tool that enhances the performance of `nls` by providing reliable initialization values for improved fitting.

In Subsection 2.3.4, we present another example with simulated data where the **nlstac** package accurately fits the given pattern. However, in this case, the `nls` function converges using both initialization approaches. Interestingly, when a non-optimal initialization is used, the fit obtained by `nls` is poor.

In Subsection 2.3.5, we fit an exponential autoregressive model. This multi-variable example showcases the robustness of the TAC algorithm showing how the **nlstac** package can be utilized in wide range of scenarios. In this example, we generate three different datasets by making perturbations to a common underlying pattern. This example is quite interesting because for each dataset the behavior of nls differs.

Furthermore, in Subsection 2.3.6, we utilize the **nlstac** package to fit real-world data to an autoregressive model.

In Subsection 2.3.7, we illustrate how the function parameter in the nls_tac function can be utilized to explicitly provide the functions within the family of approximants. This feature proves useful in cases where the algorithm does not accurately recognize the pattern.

Lastly, we want to mention that all figures presented in this section have been generated using the **ggplot2** package (Wickham, 2016).

3.1 Example 1. Exponential Decay

Although we implement **nlstac** to fit data with virtually any nonlinear function, as mentioned in the introduction, the original purpose of the TAC algorithm was to fit exponential decays models such as (1). The convergence of TAC for exponential decays patterns is proved using the max-norm as the approximation criterion.

Patterns

$$a_1 e^{-k_1 t} + a_2, \text{ where } a_1, a_2, k_1 \in \mathbb{R}, \ k_1 > 0,$$

presented in (1) are widely used to fit data coming from measuring the temperature of a body during a time interval, and their use for this propose is based on Newton's law of cooling. Let us see an example of this use.

Five parameters: data, tol, N, formula and nlparam need to be passed, as indicated in Subsection 2.2.1. The first parameter, data, must be a 2-columns matrix containing data: instants and observations.

We intend to fit pattern (1) to dataset *Coolingwater* from **mosaicData** package (Pruim et al., 2022). First, we define variable data.

```
data <- CoolingWater[40:222,]
```

Once data is loaded, we specify the tolerance, tol, or stopping criterion, and the number of divisions to be made in each step, N.

```
tol <- 1e-7
N <- 10
```

We usually set the number of divisions to 10. However, if the search intervals for the nonlinear parameters are very wide or if we suspect that there may be many local minima, it might be advisable to increase the number of divisions to avoid converging to a sub-optimal local minimum. On the contrary, if we suspect that the computing time may be too high (for example, if the number of nonlinear parameters is large), it might be advisable to reduce the number of divisions.

Next, we specify the model to be used in the fitting, form, specifying the nonlinear parameters included in the model, nlparam, as well as the interval in which we assume they can be found. Please observe that the function does not require us to initialize the parameters whatsoever, we are just asked to provide a (wide) interval where to seek them. In this example, we have chosen the interval $[10^{-7}, 1]$ as the interval where k_1 must be sought.

```
form <- 'temp ~ a1*exp(-k1*time) + a2'
nlparam <- list(k1 = c(1e-7,1))
```

Finally, we run the nls_tac function to obtain the fit.

```
tacfit <- nls_tac(formula = form, data = data,  nlparam = nlparam, N = N, tol = tol,
 parallel = FALSE)
```

Note that the input formula is either an R formula object or an object coercible to it. For example, in this case, variable form is a string that can be coerced to a formula object. Also note that we only need to specify the names and the initial intervals for the nonlinear parameters in the nlparam input. Once the nonlinear parameters are given, the function nls_tac will call the formula decomposer that will try to determine the rest of the elements of the formula —i.e. the linear parameters and nonlinear functions described in equation (4). Finally, note that tacfit is an object of class nlstac containing the following fields: coefficients, stdError, convInfo, SSR, residuals, data and formula.

So far we have only used one algorithm for the fitting: the TAC algorithm. A rightful question to be asked would be how much this fit resembles the one provided by the widely used nonlinear Least Squares (NLS) algorithm.

For this purpose we will use the NLS algorithm, by means of `nls` function, to fit the same pattern to the same data. Since NLS requires initialization values, we initialize parameter k_1 as 0.1, and parameters a_1 as 50, and a_2 as 20. Then we run `nls`.

```
nlsfit1 <- nls(formula = form, data = data, start = list(k1 = 0.1, a1 = 50, a2=20),
control = nls.control(maxiter = 1000, tol = tol))
```

Although we have chosen reasonable initialization values, the algorithm did not converge. This shows a strength of TAC, which converges without the need for initialization values.

Besides the use of TAC and NLS as algorithms that can fit by themselves, they can also be used jointly. Since **nlstac** does not require initialization, just a set of intervals in which to seek the nonlinear parameters, **nlstac** can provide to `nls` good initialization values so that `nls` can successfully converge. The lack of dependence on initialization values of **nlstac** paired with the speed of `nls` and extended use among researchers, make them quite a good team.

When using **nlstac** and `nls` together, we use the coefficients obtained in the fit with **nlstac** to initialize and run `nls` for a second time.

```
nlsfit2 <- nls(formula = form, data = data, start = coef(tacfit),
control = nls.control(maxiter = 1000, tol = tol))
```

In this case, the `nls` did indeed converge, but the fit coincides with **nlstac**'s. This show that `nls` was not able to improve **nlstac** fit, even though it was initialized with its output.

We show the summary of **nlstac** and `nls` in Table 1. For both methods, the residual standard error is 0.1647017 on 180 degrees of freedom. For **nlstac**, the number of iterations to convergence is 13 and the achieved convergence tolerance is 2.87864×10^{-8}. For `nls` when fitting with **nlstac** output as initialization, the number of iterations to convergence is 1 —meaning the initial values were close to the optimal values; furthermore, the algorithm was unable to improve those values— and the achieved convergence tolerance is 1.391929×10^{-8}.

Parameter	Estimate	Std. Error	t value	Pr(>|t|)
k_1	0.01399458	8.107657e-05	172.6095	$< 2.22 \times 10^{-16}$ ***
a_1	49.51112	0.1447617	342.0182	$< 2.22 \times 10^{-16}$***
a_2	23.82372	0.05877739	405.3212	$< 2.22 \times 10^{-16}$ ***

Table 1: Example 1. Summary of **nlstac** and `nls` for *CoolingWater* dataset with the model given in (1). Note that all outputs coincide for both methods.

Figure 2 shows the data as gray dots. Both fits, the one provided by **nlstac** and the one provided by `nls` initialized using **nlstac**'s best approximation, are shown in green.
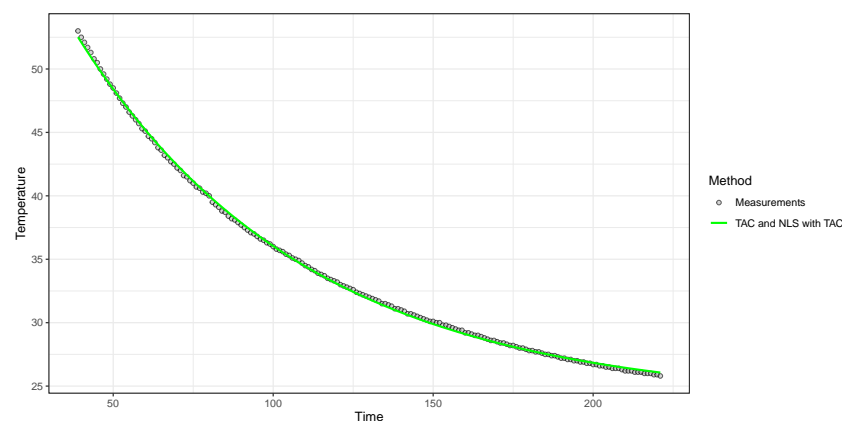

Figure 2: Example 1. Fitting an exponential decay for *CoolingWater* dataset. The figure shows the original data (grey points) and the **nlstac** fit along with the `nls` fit (green line).

For more examples of using TAC algorithm on real-world data, see section 4 of Torvisco et al. (2018) or subsection 5.1 of Cabello Sánchez et al. (2021).

3.2 Example 2. Bi-exponential Decay

In some approximation problems it is necessary to fit the sum of two exponential decays, as published in Moreno-Flores et al. (2010). While **nlstac** completely solves this problem, nls, given a bad initial values, does not converge. However, if we initialize nls with **nlstac** output, nls fit the data in a very similar way that **nlstac** does.

In this example, we intend to fit a function such as

$$f(t) = a_1 e^{-k_1 t} + a_2 e^{-k_2 t} + a_3, \text{ where } a_1, a_2, a_3, k_1, k_2 \in \mathbb{R}, k_1, k_2 > 0. \qquad (5)$$

Models of the form such as (5) were used in Moreno-Flores et al. (2010) in the fitting of data produced in indentation experiments carried out by scanning probe microscopes (e.g., Atomic Force Microscopes) in studies of viscoelastic mechanical properties of soft matter.

We intend to fit pattern (5) to *Indometh* data from the **datasets** package (R Core Team, 2021). We define parameter data and specify the tolerance, tol, and the number of divisions made in each step, N:

```
data <- Indometh[Indometh$Subject == 3, ]
tol <- 1e-7
N <- 10
```

We set the model to be used in the fitting, form, specifying the nonlinear parameters included in the model, nlparam, as well as the interval in which we assume they can be found. Finally, we apply the nls_tac function to get the fit.

```
form <- 'conc ~ a1*exp(-k1*time) + a2*exp(-k2*time) + a3'
nlparam <- list(k1 = c(1e-7,10), k2 = c(1e-7,10))
tacfit <- nls_tac(formula = form, data = data,  nlparam = nlparam, N = N, tol = tol,
  parallel = FALSE)
```

In a similar way as indicated in Example 2.3.1, we run nls initializing every parameter, that is, k_1, k_2, a_1, a_2 and a_3, as 1. Later we use the coefficients obtained with **nlstac** to initialize and run nls for a second time.

```
nlsfit1 <- nls(formula = form, data = data,
  start = list(k1 = 1,k2 = 1, a1 = 1, a2 = 1, a3 = 1),
  control = nls.control(maxiter = 1000, tol = tol))
nlsfit2 <- nls(formula = form, data = data, start = coef(tacfit),
  control = nls.control(maxiter = 1000, tol = tol))
```

While running this last piece of code we encountered an error because of bad initial values when using a vector of ones to initialize nls. We get a gradient error and nls does not converge. However, if parameters are initialized using **nlstac** output, nls does converge.

We show the summaries of **nlstac** and nls in Table 2 and Table 3, respectively.

Parameter	Estimate	Std. Error	t value	Pr(>|t|)
k_1	0.94813244	0.14656766	6.46891	0.00064760 ***
k_2	9.75308642	5.32150841	1.83277	0.11654040
a_1	2.11675784	0.29303012	7.22369	0.00035686 ***
a_2	11.09789500	12.99197213	0.85421	0.42577278
a_3	0.08168448	0.03165979	2.58007	0.04176551 *

Table 2: Example 2. Summary of **nlstac** for data of subject 3 in *Indometh* dataset with model given in (5). Residual standard error: 0.05351802 on 6 d.o.f. Number of iterations to convergence: 12. Achieved convergence tolerance: 3.824026×10^{-8}.

Figure 3 shows the data as gray dots; **nlstac** fit is shown in green and, in dashed red, nls fit is shown.

Although it is now hidden from the user, this implementation used to show some warnings related to a deficiency in a matrix rank. The explanation is that we are assuming both parameters k_1 and k_2 are contained within an interval $[10^{-7}, 10]$, so when we make two equal partitions of the same interval, one for each parameter, at some point both values will be the same: nodes of $[10^{-7}, 10] \times [10^{-7}, 10]$ where $k_1 = k_2$ and therefore we do not have a bi-exponential decay but only an exponential decay. Since these values are used in the resolution of a linear equation system that does not have a unique

Parameter	Estimate	Std. Error	t value	Pr(>|t|)
k_1	0.89971458	0.14914067	6.03266	0.00093743 ***
k_2	7.96454599	3.19653381	2.49162	0.04705893 *
a_1	2.00446255	0.29689160	6.75150	0.00051489 ***
a_2	7.63334977	4.94487377	1.54369	0.17361052
a_3	0.07663298	0.03262943	2.34858	0.05716893 .

Table 3: Example 2. Summary of `nls` for data of subject 3 in *Indometh* dataset with model given in (5). Residual standard error: 0.0527844 on 6 d.o.f. Number of iterations to convergence: 8. Achieved convergence tolerance: 3.646981×10^{-8}.
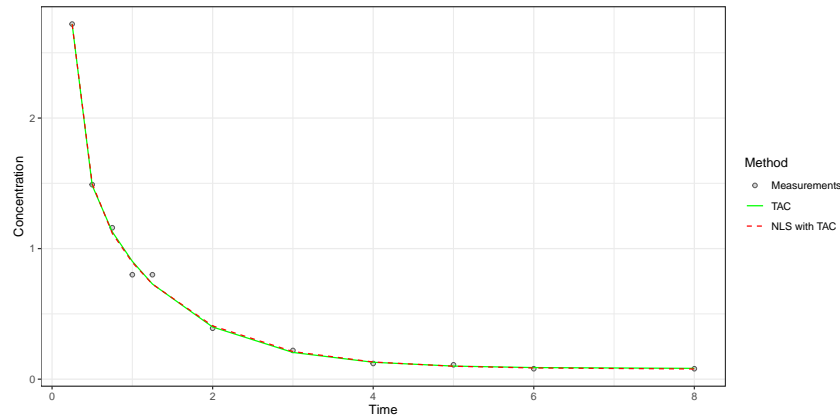

Figure 3: Example 2. Fit of a bi-exponential decay for data of subject 3 in *Indometh* dataset. The figure shows the original data (grey points), the **nlstac** fit (green line) and the `nls` fit (red line)

solution, we obtain a warning because such a unique solution can not be found. The algorithm takes care of that problem removing these indeterminate parameter sets, as well as the warnings.

Also note that, for some choice of models, permutations of parameters may give the same results. For example, using the model above, for each pair of parameters (k_1, k_2) there will be another pair of parameters (k_2, k_1) which will offer the same fit. However, that is not a problem for us, other than for an increase in the time of execution of the code. Another similar scenario is when adjusting two sinusoidal waves and two exponential decays. If we wanted to avoid making the same calculations multiple times, we would have to change the code, forcing the user to specify which functions are non-identifiable for permutations of parameters, so we would get a more time-efficient code at the cost of simplicity. However we have chosen simplicity over time efficiency.

Another bi-exponential decay example with real data can be found in section 5 of Torvisco et al. (2018).

3.3 Example 3. Exponential decays with phase displacement

In this example, **nlstac** converges and `nls`, even when reasonable initialization values are given, does not. However, if we use **nlstac** output as an initialization for `nls`, `nls` not only converges but improves **nlstac** fitting.

Here we get to see two advantages of TAC algorithm. Firstly, **nlstac** converges. Secondly, when given its approximation for `nls` initialization, `nls` improves upon this fit. That shows us, again, the two ways of using **nlstac**: directly estimating models or providing initial values.

We intend to fit a function such as

$$f(t) = a_1 e^{-b_1(t-d_1)^2} + a_2 e^{-b_2(t-d_2)^2} + a_3 e^{-b_3(t-d_3)^2} + a_4, \tag{6}$$

where $a_1, a_2, a_3, a_4, b_1, b_2, b_3, d_1, d_2, d_3 \in \mathbb{R}$. As indicated in Subsection 2.2.1 we need to pass five parameters: `data`, `tol`, `N`, `formula` and `nlparam`.

We create `data` and determine the tolerance, `tol`, and the number of divisions we make in each step, `N`.

```
set.seed(12345)
```

```
x <- seq(from = 0, to = 20, length.out = 65)
y <- 2*exp(-10*(x-0.5)^2) + 3*exp(-1*(x-2)^2) + 4*exp(-0.1*(x-5)^2) + 1 + .05*rnorm(65)
data <- data.frame(x,y)
tol <- 1e-5
N <- 6
```

We set the number of divisions to 6 because otherwise the computation time for **nlstac** is too high due to the increased number of nonlinear parameters. This is one of the downsides of **nlstac**, the increase of time processing is too high when several nonlinear parameters need to be fitted.

Later, we specify the model to be fitted, form, specifying the nonlinear parameters included in the model, nlparam, as well as the intervals in which we assume they can be found.

```
form <- 'y ~ a2*exp(-b1*(x-d1)^2) + a3*exp(-b2*(x-d2)^2)+ a4*exp(-b3*(x-d3)^2)+ a1'
nlparam <- list(b1 = c(7.7,15), b2 = c(0,5.1), b3 = c(1e-4,1.1),
        d1 = c(1e-2,1.5), d2 = c(0.1,4), d3 = c(0.11,11))
```

Finally, we apply the nls_tac function to get the fit.

```
tacfit <- nls_tac(formula = form, data = data,  nlparam = nlparam, N = N, tol = tol,
parallel = FALSE)
```

As indicated in previous examples, we will run nls. First, we will initialize it with the following values: $a_1 = 0.5, a_2 = 1, a_3 = 3, a_4 = 5, b_1 = 10, b_2 = 0.5, b_3 = 0.1, d_1 = 0, d_2 = 1$ and $d_3 = 1$. Later, we will provide **nlstac** output as nls initialization

```
nlsfit1 <- nls(formula = form, data = dat,
  start = list(a1 = 0.5, a2 = 1, a3 = 3, a4 = 5, b1 = 10, b2 = 0.5, b3 = 0.1,
  d1 = 0, d2 = 1, d3 = 1), control = nls.control(maxiter = 1000, tol = tol))
nlsfit2 <- nls(formula = form, data = dat, start = coef(tacfit),
  control = nls.control(maxiter = 1000, tol = tol))
```

As commented before, nls does not converge with the first initialization but does converge with **nlstac** initialization.

We show the results of both implementations in Table 4.

Method	a_1	a_2	a_3	a_4	b_1
nlstac	1.00896226	-1.5798685	3.38784985	3.95157736	7.7
nls	1.00522131	-2.00224548	3.79346203	3.98814647	3.86837135

Method	b_2	b_3	d_1	d_2	d_3
nlstac	0.53116510	0.10359427	1.21392	1.64537609	5.08417547
nls	0.61731929	0.09950229	1.26058001	1.57407878	5.01184086

Table 4: Example 3. Parameter estimates corresponding to **nlstac** and nls fit for dataset considered in example 3 with the model given in (6). Values have been rounded to the eighth decimal place. Please recall that nls initialized without the estimate from **nlstac** does not converge.

We show the summaries of **nlstac** and nls in Table 5 and Table 6, respectively.

Figure 4 shows the data as gray dots. In green, **nlstac** fit is shown. Dashed red line shows nls fit initialized with the parameters of **nlstac**'s best approximation.

This example is particularly significant since **nlstac** is outperformed by nls, both in time (**nlstac**'s computing time is significantly higher) and precision (compare the residual standard error value for both methods in Tables 5 and 6), although nls function needs to be initialized with **nlstac** best approximation to be able to converge.

3.4 Example 4. Exponential decay mixed with a sinusoidal signal

In this example, where we mix an exponential decay with a sinusoidal signal, we obtain a good fit with **nlstac** and a similar fit with nls when we provide **nlstac** output as initialization values. However, if we initialize nls with a bad initialization values, we get a poor fit: nls' fit identifies the exponential decay quite properly but fails to identify the sinusoidal signal.

We intend to fit a function such as

$$f(t) = a_1 e^{-k_1 t} + a_2 \sin(b_1 t) + a_3, \text{ where } a_1, a_2, a_3, k_1, b_1 \in \mathbb{R}, k_1, b_1 > 0. \tag{7}$$

Parameter	Estimate	Std. Error	t value	Pr(>|t|)
a_1	1.008962	0.026108	38.646	$< 2\times10^{-16}$ ***
a_2	-1.579868	0.200716	-7.871	1.41×10^{-10} ***
a_3	3.387850	0.189397	17.888	$< 2\times10^{-16}$ ***
a_4	3.951577	0.060719	65.080	$< 2\times10^{-16}$ ***
b_1	7.700000	2.148880	3.583	0.00072 ***
b_2	0.520107	0.030797	16.888	$< 2\times10^{-16}$ ***
b_3	0.103594	0.058464	1.772	0.08195 .
d_1	1.213920	0.048245	25.162	$< 2\times10^{-16}$ ***
d_2	1.645376	0.007954	206.864	$< 2\times10^{-16}$ ***
d_3	5.084175	0.099682	51.004	$< 2\times10^{-16}$ ***

Table 5: Example 3. Summary of **nlstac** for dataset considered in example 3 with the model given in (6). Residual standard error: 0.141 on 55 d.o.f. Number of iterations to convergence: 14. Achieved convergence tolerance: 9.871×10^{-6}.

Parameter	Estimate	Std. Error	t value	Pr(>|t|)
a_1	1.005221	0.024285	41.392	$< 2\times10^{-16}$ ***
a_2	-2.002245	0.350988	-5.705	4.80×10^{-7} ***
a_3	3.793462	0.311814	12.166	$< 2\times10^{-16}$ ***
a_4	3.988146	0.054770	72.816	$< 2\times10^{-16}$ ***
b_1	3.868371	1.061791	3.643	0.000597 ***
b_2	0.617319	0.078514	7.863	1.46×10^{-10} ***
b_3	0.099502	0.006762	14.715	$< 2\times10^{-16}$ ***
d_1	1.260580	0.033273	37.886	$< 2\times10^{-16}$ ***
d_2	1.574079	0.047734	32.976	$< 2\times10^{-16}$ ***
d_3	5.011841	0.087899	57.018	$< 2\times10^{-16}$ ***

Table 6: Example 3. Summary of nls for dataset considered in example 3 with the model given in (6). Residual standard error: 0.1307 on 55 d.o.f. Number of iterations to convergence: 16. Achieved convergence tolerance: 2.8×10^{-6}.

As indicated in 2.2.1 we need to pass five parameters: data, tol, N, formula and nlparam.

We create data and determine the tolerance, tol, or stopping criterion, and the number of divisions to be made in each step, N.

```
set.seed(12345)
x <- seq(from = 0, to = 10, length.out = 500)
y <- 3*exp(-0.85*x) + 1.5*sin(2*x) + 1 + rnorm(length(x), mean = 0, sd = 0.3)
data <- data.frame(x,y)
tol <- 1e-7
N <- 10
```

Later we set the model to be used in the fitting, form, specifying the nonlinear parameters included in the model, nlparam, as well as the intervals in which we assume they can be found.

```
form <- 'y ~ a1*exp(-k1*x) + a2*sin(b1*x) + a3'
nlparam <- list(k1 = c(0.1,1), b1 = c(1.1,5))
```

Finally, we apply the nls_tac function to adjust the data.

```
tacfit <- nls_tac(formula = form, data = data,  nlparam = nlparam, N = N, tol = tol,
  parallel = FALSE)
```

As in previous examples, we compare the **nlstac** and nls output. For the first comparison we will run nls initializing every parameter, that is, k_1, b_1, a_1, a_2 and a_3, as 1, and for the second comparison, we will use **nlstac** output to initialize and run nls.

```
nlsfit1 <- nls(formula = form, data = data, start = list(k1 = 1, b1 = 1, a1 = 1,
  a2 = 1, a3 = 1) , control = nls.control(maxiter = 1000))
nlsfit2 <- nls(formula = form, data = data, start = coef(tacfit), control =
  nls.control(maxiter = 1000))
```
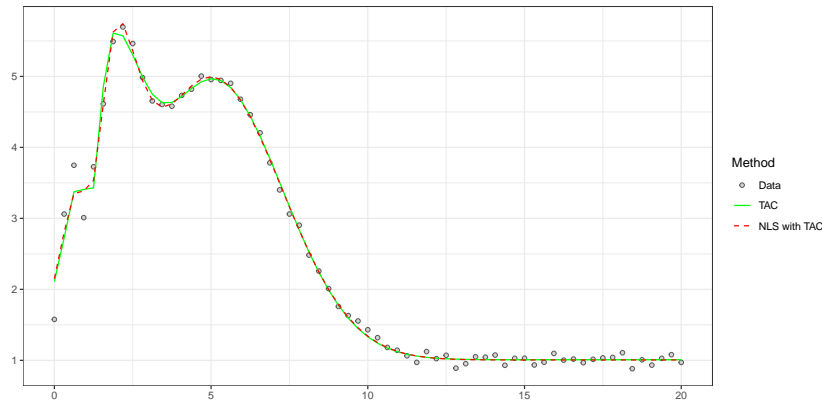
Figure 4: Example 3. The combined trend of three exponential decays with phase displacement. The figure shows the original data (grey points), the **nlstac** fit (green line) and the `nls` fit using **nlstac**'s best approximation (red line).

We present the results obtained in Table 7. Without looking at any graphic, it is quite evident from Table 7 alone that the last fit is different from the two others.

Method	k_1	b_1	a_1	a_2	a_3
nlstac	0.81234568	1.99851628	3.01166130	1.51095645	1.00766609
`nls` (with **nlstac** initialization)	0.81904149	1.99847422	3.01996411	1.51073313	1.00969794
`nls` (without **nlstac** initialization)	0.82435634	0.83083729	4.49199629	-0.23904801	0.91892869

Table 7: Example 4. Parameters corresponding to **nlstac** and `nls` fits for dataset considered in example 4 with the model given in (7). Values have been rounded off to the eighth decimal place.

Summary of **nlstac** is shown in Table 8 and Table 9 shows the summary of `nls` initialized with the best approximation obtained with **nlstac**. Finally, summary of `nls` initialized with a vector of ones appears in Table 10.

Parameter	Estimate	Std. Error	t value	Pr(>|t|)
k_1	0.812346	0.035145	23.11	$<2 \times 10^{-16}$ ***
b_1	1.998516	0.002132	937.35	$<2 \times 10^{-16}$ ***
a_1	3.011661	0.077098	39.06	$<2 \times 10^{-16}$ ***
a_2	1.510956	0.019734	76.57	$<2 \times 10^{-16}$ ***
a_3	1.007666	0.018885	53.36	$<2 \times 10^{-16}$ ***

Table 8: Example 4. Summary of **nlstac** for dataset considered in example 4 with the model given in (7). Residual standard error: 0.2974 on 495 d.o.f. Number of iterations to convergence: 11. Achieved convergence tolerance: 3.184×10^{-8}.

Figure 5 shows the data as gray dots. Green line represents the **nlstac** fit and dashed red line represents `nls` fit. Blue line represents `nls` fit initialized with a vector of ones. It is clear that the last fit is not accurate. It seems that the nonlinear least squares algorithm has managed to fit the exponential part of the pattern but it seems to have missed the sinusoidal part.

This example shows a situation where **nlstac** works perfectly, as well as `nls` if initialized correctly. However, if the user does not provide good initialization values, the nonlinear least squares algorithm might fail to obtain a good fit since it may get stuck in a local minimum.

Parameter	Estimate	Std. Error	t value	Pr(>|t|)
k_1	0.819041	0.035388	23.14	$<2\times10^{-16}$ ***
b_1	1.998474	0.002132	937.31	$<2\times10^{-16}$ ***
a_1	3.019964	0.077382	39.03	$<2\times10^{-16}$ ***
a_2	1.510733	0.019733	76.56	$<2\times10^{-16}$ ***
a_3	1.009698	0.018813	53.67	$<2\times10^{-16}$ ***

Table 9: Example 4. Summary of `nls` initialized with **nlstac**'s best approximation for dataset considered in example 4 with the model given in (7). Residual standard error: 0.2974 on 495 d.o.f. Number of iterations to convergence: 4. Achieved convergence tolerance: 5.597×10^{-9}.

Parameter	Estimate	Std. Error	t value	Pr(>|t|)
k_1	0.82436	0.10069	8.187	2.3×10^{-15} ***
b_1	0.83084	0.06042	13.751	$< 2\times10^{-16}$ ***
a_1	4.49200	0.26879	16.712	$< 2\times10^{-16}$ ***
a_2	-0.23905	0.07680	-3.112	0.00196 **
a_3	0.91893	0.07455	12.326	$< 2\times10^{-16}$ ***

Table 10: Example 4. Summary of `nls` initialized with a vector of ones for dataset considered in example 4 with the model given in (7). Residual standard error: 1.056 on 495 d.o.f. Number of iterations to convergence: 23. Achieved convergence tolerance: 7.587×10^{-8}.

3.5 Example 5. Exponential autoregressive model: a multi-variable approach (p-variable)

Nonlinear time series models are used in a wide range of fields. In this example, we deal with an especially relevant nonlinear time series model: the exponential autoregressive model. Given a time series $\{x_1, x_2, x_3, \ldots\}$, the exponential autoregressive model is defined as

$$x_t = \left[\sum_{i=1}^{p} \left(a_i + b_i e^{-cx_{t-1}^2}\right) x_{t-i}\right] + \varepsilon_t,$$

where ε_t are independent and identically distributed random variables and independent with x_i, p denotes the system degree, $t \in \mathbb{N}$, $t > p$, and the parameters to be estimated from observations are c, a_i and b_i (for $i = 1, \ldots, p$). This model can be found in, for example, Xu et al. (2019) or Chen et al. (2018).

Some generalizations for the exponential autoregressive model have been made, and in this example, we will deal with a generalization of Teräsvirta's extended model that can be found in Chen et al. (2018, equation (10)) and we present here:

$$x_t = a_0 + \left[\sum_{i=1}^{p} \left(a_i + b_i e^{-c(x_{t-d} - z_i)^2}\right) x_{t-i}\right] + \varepsilon_t, \tag{8}$$

where z_i (for $i = 1, \ldots, p$) are scalar parameters and $d \in \mathbb{Z}$.

We would like to point out that the convergence of TAC algorithm has not been established for this type of problem. Further, this example is substantially different from the above examples since every observation depends on the previous ones. This model can not be described by a function of just one real variable. Instead, a vector of p real variables needs to be used. This approach can be developed considering a function from $(\mathbb{R}^{n-p})^p$ into $\mathbb{R}^{n-p}$. Therefore we transform a one-dimensional problem into a p-dimensional one. Let us explain this process. Let $x = (x_1, \ldots, x_n)$ denote the observations and let us define p variables $v1, \ldots, vp \in \mathbb{R}^{n-p}$, being $vi = (x_{p-i+1}, \ldots, x_{n-i})$ for $i = 1, \ldots, p$, which will allow us to redefine Equation (8) in terms of these new variables:

$$x_{t+p} = a_0 + \left[\sum_{i=1}^{p} \left(a_i + b_i e^{-c((vd)_t - z_i)^2}\right) (vi)_t\right] + \varepsilon_t, \text{ with } t = 1, \ldots, n - p, \tag{9}$$

where vd is fixed with d such that $1 \leq d \leq p$ and $(vi)_t$ denote the t-th component of variable vi.

For this example, first, we are going to fix one exponential autoregressive time series. Then we are going to generate three different datasets: Dataset 1, Dataset 2, and Dataset 3. Each of these datasets is generated by setting three different seeds (Seed 1, Seed 2, and Seed 3, respectively) in order to add a
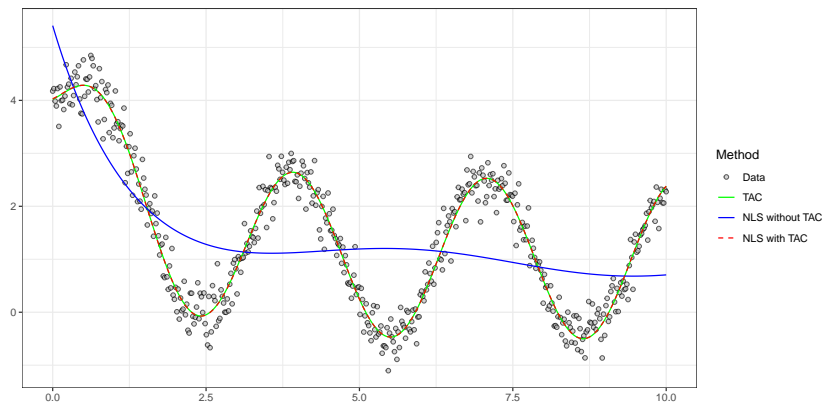
Figure 5: Example 4. Fit of an exponential decay mixed with a sinusoidal signal for dataset considered in example 4 with the model given in (7). The figure shows the original data (grey points), the **nlstac** fit (green line), the `nls` fit initialized with **nlstac** output (red line) and the `nls` fit initialized with a vector of ones (blue line).

random perturbation to the terms of the previously fixed time series.

The aim of this example is to fit model (9) with $p = 2$, $d = 2$, and $n = 100$ for each dataset. We start defining a vector, seed, containing the three seeds. We define the tolerance level, `tol`, and the parameters for the time series as well as initialize the time series with the first two terms. We also define the pattern to be fitted.

```
seed <- c('12','123','1234')
tol <- 1e-5
a0 <- -1.45
a1 <- 1.66
b1 <- -0.47
a2 <- 0.543
b2 <- -0.82
c <- 1.27
z1 <- 2.53
z2 <- 3.85
x <- numeric(100)
x[1] <- 2.7
x[2] <- 3.12
form <- 'y ~ a0+ a1*v1 + b1*v1*exp(-c*(v2-z1)^2) + a2*v2 + b2*v2*exp(-c*(v2-z2)^2)'
```

We intend to run **nlstac** in parallel using package **doParallel** (Corporation and Weston, 2022). We set up the parallelization:

```
no_cores <- detectCores() - 1
cl <- makeCluster(no_cores)
registerDoParallel(no_cores)
```

We are going to define a loop that will iterate three times, one for each dataset. In each iteration we set a seed and generate its corresponding dataset: seeds '12', '123', and '1234' will generate Dataset 1, Dataset 2, and Dataset 3, respectively. Then we transform the problem into a two-dimensional one by defining variables y, v1, and v2 as previously described. We create the dataframe, run **nlstac** in parallel and run `nls` with two different initializations: first initialized with $c = 1$, $z_1 = 2.25$, $z_2 = 4$, $a_0 = -1$, $a_1 = 1$, $b_1 = -1$, $a_2 = 1$, $b_2 = -1$ and then initialized with **nlstac** output. The function `tryCatch` is used in order to keep the loop running in the event that `nls` does not converge for some choice of initial parameters.

```
for (j in 1:3) {
  set.seed(seed[j])
  for (i in 3:100){
    x[i] <- a0 + (a1+b1*exp(-c*(x[i-2]-z1)^2))*x[i-1] +
      (a2+b2*exp(-c*(x[i-2]-z2)^2))*x[i-2] + rnorm(1, mean=0, sd=0.1)}
  y <- x[3:100]
  v1 <- x[2:99]
  v2 <- x[1:98]
```

```
data <- data.frame(v1 = v1, v2 = v2, y = y)
tacfit <- nls_tac(form, data = data,
                 nlparam = list(c = c(0,2), z1 = c(1,3.5), z2 = c(3,5)),
                 N = 10, tol=tol, parallel = TRUE)
tryCatch(nlsfit1 <- nls(formula = form, data = data, start = list(c = 1, z1 = 2.25,
      z2 = 4, a0 = -1, a1 = 1, b1 = -1, a2 = 1, b2 = -1),
      control = nls.control(maxiter = 1000, tol = tol, minFactor = 1e-5)),
      error = function(e) {nlsfit1 <<- NULL})
tryCatch(nlsfit2 <- nls(formula = form, data = data, start = coef(tacfit),
      control = nls.control(maxiter = 1000, tol = tol, minFactor = 1e-5)),
      error = function(e) {nlsfit2 <<- NULL}) }
```

Finally we stop the parallelization:

```
stopImplicitCluster()
```

For Data 1, pattern (9) has successfully been fitted with all three methods. For Data 2, **nlstac** did converge but nls did not converge for either of the two initializations. Finally, for Data 3, nls only converged if initialized with **nlstac** output, and in this case nls fit slightly improves **nlstac**'s.

We present the results for all three datasets in Table 11.

| | Parameter | Dataset 1 | | | Dataset 2 | Dataset 3 | |
		nlstac	nls (without **nlstac** init.)	nls (with **nlstac** init.)	**nlstac**	**nlstac**	nls (with **nlstac** init.)
c	Estimate	1.038256	1.019213	1.019233	0.246914	1.728395	5.441692
	Std. Error	0.487366	0.477747	0.477748	3.816922	2.300992	4.654574
z_1	Estimate	2.388889	2.436606	2.436604	3.211423	1.438958	2.261739
	Std. Error	0.521781	0.501099	0.501096	0.082645	1.707290	0.287022
z_2	Estimate	3.855393	3.881704	3.881695	3.006097	3.843975	3.628858
	Std. Error	0.251267	0.231838	0.231837	2.613194	0.296154	0.123689
a_0	Estimate	-1.929269	-2.691265	-2.691134	5.616140	-8.359206	-3.401572
	Std. Error	7.038082	6.687379	6.687395	45.308458	9.373772	5.347638
a_1	Estimate	1.778348	1.823764	1.823752	-3.721976	1.312138	1.329813
	Std. Error	0.409774	0.437640	0.437630	77.352777	0.075125	0.067164
b_1	Estimate	-0.813623	-0.832519	-0.832507	5.169974	2.888960	0.327253
	Std. Error	0.360567	0.380349	0.380342	77.339916	11.657008	0.542132
a_2	Estimate	1.055334	1.349577	1.349517	1.794767	2.716872	0.851547
	Std. Error	2.698594	2.549761	2.549766	40.429455	3.641374	1.854555
b_2	Estimate	-1.272387	-1.392871	-1.392838	-4.026071	-0.873735	-0.310493
	Std. Error	1.115143	1.090259	1.090251	55.467877	1.228695	0.348730
	SSR	0.701835	0.701740	0.701740	0.690280	0.87582	0.856081

Table 11: Example 5. Summary of all three methods (**nlstac**, nls without **nlstac** initialization, nls with **nlstac** initialization) for all three datasets considered in example 5 with the model given in (8). Missing methods for Dataset 2 and Dataset 3 are the result of the non-convergence of such methods.

Figure 6 shows the fitting for those three datasets.

3.6 Example 6. Exponential autoregressive model: a multi-variable approach (p-variable) with real data.

In this example, we intend to fit model (8) to returns on daily closing prices data from Financial Times Stock Exchange (FTSE) using **nlstac** package. More precisely, if vector $x = (x_1, \ldots, x_n)$ denotes the daily closing prices data, we are going to fit the returns, that is, vector $\left(\frac{x_2 - x_1}{x_1}, \ldots, \frac{x_i - x_{i-1}}{x_{i-1}}, \ldots, \frac{x_n - x_{n-1}}{x_{n-1}} \right)$. This data was obtained by *EuStockMarkets* dataset which is accessible from **datasets** package. As in the previous example, we are going to consider $p = 2$ and $d = 2$.
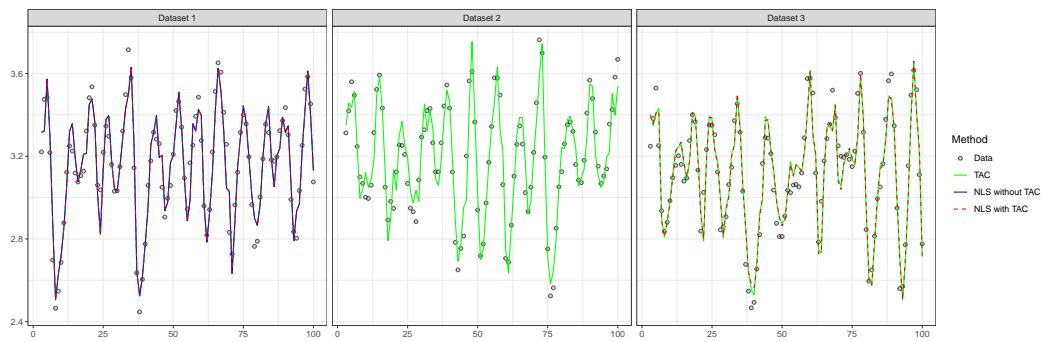
Figure 6: Example 5. Fits for three exponential autorregresive model datasets with model given in (8) as determined at the beginning of Example 5. The figure shows the original data (grey points), the **nlstac** fit (green line), the `nls` fit initialized with **nlstac** output (red line) and the `nls` fit initialized with **nlstac** output (blue line). Note that, in Dataset 1, blue line overlaps the green one.

First, we read in the data and specify the tolerance level:

```
x <- EuStockMarkets[,4]
x <- diff(x)/x[-length(x)]
tol <- 1e-7
```

Then we transform the problem into a two-dimensional one:

```
y <- x[3:length(x)]
v1 <- x[2:(length(x)-1)]
v2 <- x[1:(length(x)-2)]
data <- data.frame(v1 = v1, v2 = v2, y = y)
form <- 'y ~ a0+ a1*v1 + b1*v1*exp(-c*(v2-z1)^2) + a2*v2 + b2*v2*exp(-c*(v2-z2)^2)'
```

Finally, we make use of package **doParallel** to run **nlstac** in parallel:

```
no_cores <- detectCores() - 1
cl <- makeCluster(no_cores)
registerDoParallel(no_cores)
tacfit <- nls_tac(form, data = data,
            nlparam = list(c = c(1e-7,5), z1 = c(1e-7,5), z2 = c(1e-7,5)),
            N = 15, tol=tol, parallel = TRUE)
stopImplicitCluster()
```

Results are summarized in Table 12 and a plot with both the data and the fit obtained by `nls_tac` function is depicted in Figure 6

Parameter	Estimate	Std. Error	t value	$Pr(>|t|)$
c	2.857144e-01	4.246881e+02	0.00067	0.99946
z_1	3.668653e-03	4.806933e-03	0.76320	0.44544
z_2	7.164164e-02	2.281940e-01	0.31395	0.75359
a_0	7.571841e-05	2.219138e-04	0.34121	0.73299
a_1	-9.276088e+02	1.378917e+06	-0.00067	0.99946
b_1	9.277265e+02	1.378917e+06	0.00067	0.99946
a_2	-1.374292e+02	2.038046e+05	-0.00067	0.99946
b_2	1.376215e+02	2.038049e+05	0.00068	0.99946

Table 12: Example 6. Summary of **nlstac** for returns from FTSE in *EuStockMarkets* dataset with the model given in (8). Residual standard error: 0.00791658 on 1849 d.o.f. Number of iterations to convergence: 11. Achieved convergence tolerance: 1.94289×10^{-16}.

Another example of an exponential autoregressive model with real data can be consulted in subsection 6.3 of Cabello Sánchez et al. (2021).
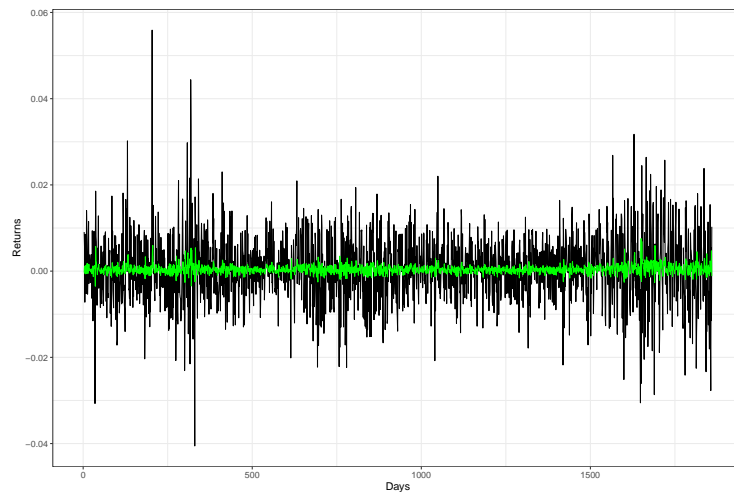
Figure 7: Example 6. Fit of an exponential autorregresive model for returns from FTSE in *EuStockMarkets* dataset with model given in (8). The figure shows the original data (black lines) and the **nlstac** fit (green line).

3.7 Example 7. Explicitly providing the functions of the pattern

Package **nlstac** relies on the `formula` infrastructure of the R language to determine the number and expressions of the nonlinear functions ϕ_i in (4). This is done internally by the function `get_functions`. However, in some cases, the user may want to explicitly state which are the nonlinear functions that define the separable nonlinear problem (e.g. the formula decomposer fails to automatically identify those functions). In that case, there is an optional parameter in the `nls_tac` function, named `functions`, which is an array of character strings defining the nonlinear functions. In practical terms, we have not found an example in which we needed to manually specify the functions defining the model, but the optional parameter is available nonetheless.

Next, just for illustration purposes, we present here the example shown in Subsection 2.3.4 adding the `functions` parameter to the function `nls_tac` so we explicitly provide the functions:

```
set.seed(12345)
x <- seq(from = 0, to = 10, length.out = 500)
y <- 3*exp(-0.85*x) + 1.5*sin(2*x) + 1 + rnorm(length(x), mean = 0, sd = 0.3)
data <- data.frame(x,y)
tol <- 1e-7
N <- 10
form <- 'y ~ a1*exp(-k1*x) + a2*sin(b1*x) + a3'
nlparam <- list(k1 = c(0.1,1), b1 = c(1.1,5))
tacfit <- nls_tac(formula = form, data = data,
  functions=c('exp(-k1*x)','sin(b1*x)','1'),  nlparam = nlparam, N = N, tol = tol,
  parallel = FALSE)
```

4   Code parallelization

The basic idea of the TAC algorithm is to find the optimal values for the linear parameters (by means of the linear least-square method) for each combination of the nonlinear parameters. Therefore, for every such combination of nonlinear parameters we have to solve a completely independent optimization problem, and thus this algorithm can take advantage of parallelization.

The **nlstac** package implements a parallelization of this stage of the algorithm in the `nls_tac` function. Setting the option `parallel=TRUE`, the function makes use of the `%dopar%` and `foreach` functions of the **foreach** (Microsoft and Weston, 2022) package and the infrastructure provided by the **parallel** (R Core Team, 2021) and **doParallel** packages.

One might think that parallelization always speeds up the algorithm, but in reality, initializing and stopping the cluster requires a certain amount of time. Therefore, in some cases it may be convenient to parallelize and in others it might not be worth it.

As was mentioned in the Introduction, the TAC algorithm, as all grid-search algorithms, scales poorly with the dimension of the problem (i.e. the number of nonlinear parameters). However, even

for low-dimensional problems, the speed of the algorithm depends on the number of subdivisions of each parameter search interval (i.e. the width of the grid), which is defined by the parameter N in the nls_tac function. Previous affirmations rely on the fact that, for each iteration, the number of plausible nonlinear parameters happens to be N^p, representing N, for each parameter, the number of nodes belonging to the partition of the interval where the parameter is assumed to be in and p the number of nonlinear parameters. Note that the number of plausible nonlinear parameters depends on N and exponentially increases with the number of nonlinear parameters p.

As an illustration of the convenience of using the parallel = TRUE option, Figure 8 depicts a comparison of non-parallel and parallel modes of the nls_tac function for two standard separable nonlinear least square problems. Namely, two exponential decays and three exponential decays. That is:

$$y = a_0 + \sum_{k=1}^{n} a_k e^{-b_k x}, \qquad n = 2, 3.$$

As could be expected, we can see how as the number of nonlinear parameters increases, the computation time rises exponentially. Also, it shows that only for very small problems (e.g. two nonlinear parameters) the parallelization is not worth it in some cases (N up to 35). To run this simulation we had to make use of **dplyr** package (Wickham et al., 2022).
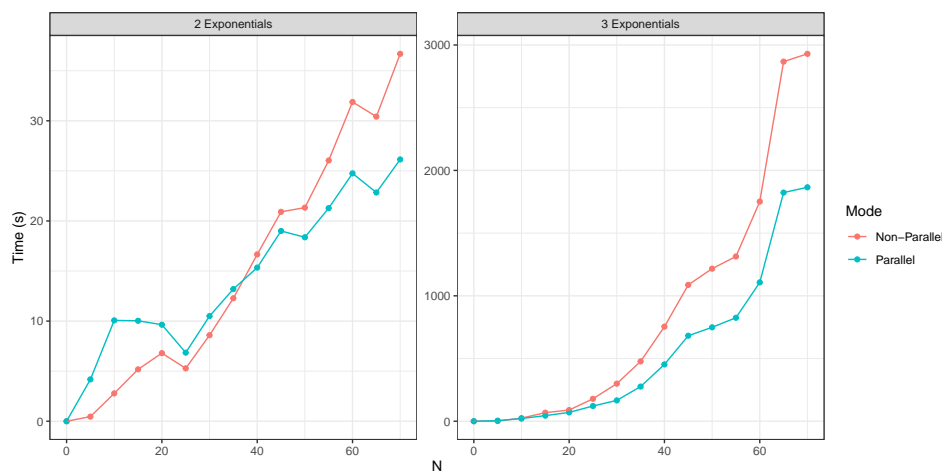

Figure 8: Comparison between the parallel and the non-parallel implementations of the nls_tac function for the fitting of two (left) and three (right) exponential decays, for different values of the number of subdivisions, N. Note how the time of processing is increased around a hundred times when changing from two nonlinear parameters to three nonlinear parameters.

5 Conclusions

Many popular packages for nonlinear function estimation depend heavily on the choice of starting values. This package, however, implements an algorithm that needs no initialization and can handle a wide variety of approximation problems.

Our goal has been to create a package for nonlinear regression using the TAC algorithm and to show how this algorithm can work either by itself or when combined with other nonlinear estimation algorithms.

Processing times on problems with a large number of nonlinear parameters can be a problem. In those cases, it might be advisable to consider the use of a gradient-based algorithm. In future versions, the implemented grid search could be refined to reduce those processing times.

Despite this possible drawback, we strongly believe that this package will be found useful by researchers in nonlinear regression problems.

References

R. K. Arora. *Optimization. Algorithms and applications*. Taylor & Francis Group, LLC, 2015. [p6]

D. Bates, K. M. Mullen, J. C. Nash, and R. Varadhan. *minqa: Derivative-free optimization algorithms*

by quadratic approximation, 2014. URL https://CRAN.R-project.org/package=minqa. R package version 1.2.4. [p7]

C. Bendtsen. *pso: Particle Swarm Optimization*, 2022. URL https://CRAN.R-project.org/package=pso. R package version 1.0.4. [p7]

A. Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, United States of America, 1 edition, 1996. doi: 10.1137/1.9781611971484. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611971484. [p6]

J. Cabello Sánchez, J. A. Fernández Torvisco, and M. R. Arias. Tac method for fitting exponential autoregressive models and others: Applications in economy and finance. *Mathematics*, 9(8), 2021. ISSN 2227-7390. doi: 10.3390/math9080862. URL https://www.mdpi.com/2227-7390/9/8/862. [p11, 20]

G.-Y. Chen, M. Gan, and G.-L. Chen. Generalized exponential autoregressive models for nonlinear time series: Stationarity, estimation and applications. *Information Sciences*, 438, 04 2018. doi: 10.1016/j.ins.2018.01.029. [p17]

B. Christoffersen. *psqn: Partially Separable Quasi-Newton*, 2022. URL https://CRAN.R-project.org/package=psqn. R package version 0.3.1. [p7]

E. L. T. Conceicao. *DEoptimR: Differential Evolution Optimization in Pure R*, 2022. URL https://CRAN.R-project.org/package=DEoptimR. R package version 1.0-10. [p7]

A. Coppola, B. Stewart, and N. Okazaki. *lbfgs: Limited-memory BFGS Optimization*, 2022. URL https://CRAN.R-project.org/package=lbfgs. R package version 1.2.1.2. [p7]

M. Corporation and S. Weston. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*, 2022. URL https://CRAN.R-project.org/package=doParallel. R package version 1.0.17. [p18]

G. Csárdi, J. Hester, H. Wickham, W. Chang, M. Morgan, and D. Tenenbaum. *remotes: R Package Installation from Remote Repositories, Including 'GitHub'*, 2021. URL https://CRAN.R-project.org/package=remotes. R package version 2.4.2. [p8]

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*, 2023. URL https://CRAN.R-project.org/package=minpack.lm. R package version 1.2-3. [p5]

G. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on Numerical Analysis*, 10:413–432, 04 1973. doi: 10.1137/0710036. [p7]

G. Golub and V. Pereyra. Separable nonlinear least squares: the variable projection method and its applications. *Inverse Problems*, 19:R1–R26(1), 01 2003. [p7]

Microsoft and S. Weston. *foreach: Provides Foreach Looping Construct*, 2022. URL https://CRAN.R-project.org/package=foreach. R package version 1.5.2. [p21]

S. Moreno-Flores, R. Benitez, M. dM Vivanco, and J. L. Toca-Herrera. Stress relaxation and creep on living cells with the atomic force microscope: a means to calculate elastic moduli and viscosities of cell components. *Nanotechnology*, 21(44):445101, 2010. URL http://stacks.iop.org/0957-4484/21/i=44/a=445101. [p12]

K. M. Mullen and I. H. M. van Stokkum. Timp: an r package for modeling multi-way spectroscopic measurements. *Journal of Statistical Software*, 18(3), 2007. doi: 10.18637/jss.v018.i03. [p7]

J. C. Nash and D. Murdoch. *nlsr: Functions for Nonlinear Least Squares Solutions - Updated 2022*, 2023. URL https://CRAN.R-project.org/package=nlsr. R package version 2023.5.8. [p7]

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006. [p6]

R. Pruim, D. Kaplan, and N. Horton. *mosaicData: Project MOSAIC Data Sets*, 2022. URL https://CRAN.R-project.org/package=mosaicData. R package version 0.20.3. [p10]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.R-project.org/. [p5, 12, 21]

R. R. Rhinehart. *Nonlinear regression modeling for engineering applications: modeling, model validation, and enabling design of experiments*. John Wiley & Sons, 2016. [p6]

M. Rodriguez-Arias, J. A. F. Torvisco, J. Cabello, and R. Benitez. *nlstac: An R Package for Fitting Separable Nonlinear Models*, 2023. URL https://CRAN.R-project.org/package=nlstac. R package version 0.1.0. [p5, 8]

L. Scrucca. GA: A package for genetic algorithms in R. *Journal of Statistical Software*, 53(4):1–37, 2013. doi: 10.18637/jss.v053.i04. [p7]

J. A. F. Torvisco, M. R. Arias, and J. Cabello Sánchez. A new algorithm to fit exponential decays without initial guess. *Filomat*, 32:4233–4248, 01 2018. doi: 10.2298/FIL1812233T. [p5, 7, 11, 13]

G. Vega Yon and E. Muñoz. *ABCoptim: An implementation of the Artificial Bee Colony (ABC) Algorithm*, 2017. URL https://github.com/gvegayon/ABCoptim. R package version 0.15.0. [p7]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p10]

H. Wickham, R. François, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2022. URL https://CRAN.R-project.org/package=dplyr. R package version 1.0.8. [p22]

M. Wilson. spant: An r package for magnetic resonance spectroscopy analysis. *Journal of Open Source Software*, 6(67):3646, 2021. doi: 10.21105/joss.03646. [p7]

Y. Xiang, S. Gubian, B. Suomela, and J. Hoeng. Generalized simulated annealing for efficient globaloptimization: the gensa package for r. *The R Journal*, 5/1, 2013. URL https://journal.r-project.org. [p7]

H. Xu, F. Ding, and E. Yang. Modeling a nonlinear process using the exponential autoregressive time series model. *Nonlinear Dynamics*, 95, 02 2019. doi: 10.1007/s11071-018-4677-0. [p17]

J. A. F. Torvisco
Facultad de Ciencias, Universidad de Extremadura
Avda de Elvas s/n, 06006 Badajoz, Spain
ORCiD: 0000-0001-8373-3477
jfernandck@alumnos.unex.es

R. Benítez
Departmento de Matemáticas para la Economía y la Empresa, Universidad de Valencia
Avda Tarongers s/n, 46022 Valencia, Spain
ORCiD: 0000-0002-9443-0209
rabesua@uv.es

M. R. Arias
Facultad de Ciencias, Universidad de Extremadura
Avda de Elvas s/n, 06006 Badajoz, Spain
ORCiD: 0000-0002-4885-4270
arias@unex.es

J. Cabello Sánchez
Facultad de Ciencias, Universidad de Extremadura
Avda de Elvas s/n, 06006 Badajoz, Spain
ORCiD: 0000-0003-2687-6193
coco@unex.es

hydrotoolbox, a Package for Hydrometeorological Data Management

by Ezequiel Toum and Pierre Pitte

Abstract The hydrometeorological data provided by federal agencies, research groups and private companies tend to be heterogeneous: records are kept in different formats, quality control processes are not standardized and may even vary within a given agency, variables are not always recorded with the same temporal resolution, and there are data gaps and incorrectly recorded values. Once these problems are dealt with, it is useful to have tools to safely store and manipulate the series, providing temporal aggregation, interactive visualization for analysis, static graphics to publish and/or communicate results, techniques to correct and/or modify the series, among others. Here we introduce a package written in the R language using object-oriented programming and designed to accomplish these objectives, giving to the user a general framework for working with any kind of hydrometeorological series. We present the package design, its strengths, limitations and show its application for two real cases.

1 Introduction

Data management has generated growing interest among the scientific community (Venkatraman 2013; Donoho 2017), driven by the proliferation of data sharing through the internet, new measurement techniques and equipment, and open and free access to information obtained from remote sensing. It is also fueled by new and growing demands from agencies and governments (national and/or regional) for data-informed decision-making. To realize the full potential of the available information, there is a need for accurate and efficient tools that can pre-process these vast and heterogeneous data into more convenient data sets (Addor et al. 2020).

The hydrological community uses data from hydrometeorological stations, remote sensing observations and outputs from climatic and/or hydrological models (Beven 2012). These data types may include (but are not limited to):

a. **remote sensing**: snow cover images, soil humidity, glacier cover areas, digital elevation models.

b. **hydro-met stations**: air temperature, relative humidity, incoming solar radiation, atmospheric pressure, precipitation, streamflow records.

c. **numerical weather models**: air temperature, precipitation, and wind speed forecasting.

d. **hydrological models**: evolution of the snowpack, simulated streamflow, infiltration rates.

e. **large-sample hydrology datasets**

Data from remote sensing and climate models is distributed in standardized formats (e.g., *.nc, .tiff, .grb*) and therefore there are standardized tools to manipulate them (Hijmans 2017; Conrad et al. 2015; Pierce 2019). This is not the case for hydrometeorological data as each agency, research group or private company has its own standards. Therefore, the formats, publication frequency, temporal resolution and quality control vary. As an example, in Argentina, the organizations that manage hydrological data tend to use proprietary software: a) the *Autoridad Interjurisdiccional de las Cuencas de los ríos Limay, Neuquén y Negro* (AIC) uses DIMS (Dynamic Integrated Monitoring System), b) the *Sistema Nacional de Información Hídrica* (SNIH) works with Mnemos, and c) the *Instituto Nacional del Agua* (INA) uses their own software (personal communication).

The R language has gained a central role within the hydrological community during the last decade because it allows the user to: a) download and work with multiple data types and formats, b) extract, manipulate and order data, c) develop hydrological models, d) conduct statistical analyses, e) view results, and f) export images and documents ready for publication. Slater et al. (2019) pointed out the benefits and advantages of using R in the hydrological field: (1) it democratizes science and numerical literacy; (2) it helps the research to be reproducible and open; (3) provides statistical tools; (4) allows connection to and from other languages; and (5) has the support from a constantly growing community. Indeed the R community has developed many packages for managing hydrological data (https://cran.r-project.org/web/views/Hydrology.html), but they tend to be either focused on Europe and North America or lack a general-purpose framework for efficiently and safely storing and manipulating hydrometeorological series:

1. **hddtools** (Vitolo 2017): is designed to facilitate access to a variety of online open data sources relevant for hydrologists and, more generally, environmental scientists and practitioners. It

includes functionality for Koppen Climate Classification, Global Runoff Data Centre, Data60UK, MOPEX (USA) and SEPA (Scotland).

2. **climate** (Czernecki, Głogowski, and Nowosad 2020): allows automatic downloading of OGIMET, University of Wyoming (atmospheric vertical profiling data), Polish Institute of Meteorology and Water Management (National Research Institute) and National Oceanic & Atmospheric Administration (NOAA).

3. waterData (Ryberg and Vecchia 2017): imports U.S. Geological Survey (USGS) daily hydrologic data from USGS web services, plots the data, addresses some common data problems, calculates and plots anomalies.

4. hydroTSM (Zambrano-Bigiarini 2020): provides functions for management, analysis, interpolation and plotting of time series used in hydrology and related environmental sciences. This package is highly oriented towards hydrological modeling tasks.

5. dataRetrieval (De Cicco et al. 2018): is a collection of functions that help to retrieve U.S. Geological Survey (USGS) and U.S. Environmental Protection Agency (EPA) water quality and hydrology data from web services.

6. hyfo (Xu 2020): is a package with focus on processing and visualization of hydrological data and climate forecasting. Main function includes data extraction, data downscaling, resampling, gap filler of precipitation, bias correction of forecasting data, flexible time series plot, and spatial map generation.

hydrotoolbox (Toum 2022) is an R package developed using R's object-oriented programming system (S4 classes; Chambers (2017)) that provides a general framework for efficiently storing and manipulating multiple hydrometeorological datasets, exploiting not only S4 advantages but base R's copy on modify semantics. The series and its metadata (e.g.: geo-referenced location, river basin name, country, among others) are agglomerated in a single class object, allowing for an effective management of vast and heterogeneous hydrometeorological data. **hydrotoolbox** also provides a general set of methods regarding: a) reading multiple datasets with their unique data formats (i.e. delimited files such as comma-separated values (CSV) and tab-separated values (TSV) and excel files), b) static and interactive time-series plotting, c) data manipulation, d) data temporal aggregation, among others. In addition, the current version (v 1.1.2) has specific functions for reading data from Argentina and Chile. Despite this fact, we want to stress that every user can combine other R functions or packages for downloading and reading data (including those coming from large-sample hydrological datasets; Addor et al. (2020)) with the functionality of the package.

The rest of this work is organized as follows: in the **Package design** section we describe the underlying ideas with a brief description of the classes, subclasses, methods and functions that **hydrotoolbox** offers. In the **Case studies section** we show two applied examples: the first deals with a hydrometerological data set and the second deals with post-processing the results of a modeling exercise with the HBV.IANIGLA model (Toum 2021; Toum et al. 2021). These cases lead to discussion about the package performance. We conclude with a discussion of possible improvements and ways to extend the functionality **hydrotoolbox**.

2 Package design

hydrotoolbox uses the object-oriented programming (OOP) paradigm, a feature that makes the package flexible and allows other programmers to extend its functionality. In this package, the properties are related to the hydrometeorological variables and the methods with functions for these objects manipulation. The following principles guided the design of **hydrotoolbox**:

1. **The time series of each variable must be grouped into stations**, as they are organized based on their geographic location. This allows series registered in the same station or at different ones to be compared without losing their position, a fundamental aspect for a correct physical interpretation of the data.

2. **Modifications must be recorded in the same file**. Most time series have errors that need to be corrected. This could happen because the time series are discontinuous, the instruments fail, the variables are not all measured in the same temporal resolution, because of power cuts, or because local natural conditions induce erroneous measurements (e.g., snowfall undercatch due to wind; Goodinson and Louie (1998)). Errors are corrected in successive steps but it is crucial to have access to all versions of the series. This principle also seeks to avoid having multiple records of the same variable in a given station. The reader will find an example in the *Case studies* section.
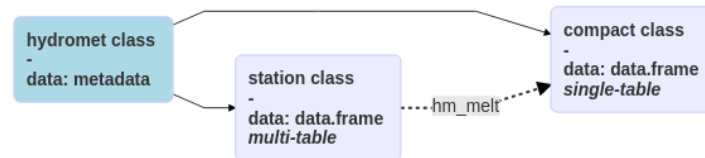
Figure 1: Classes for hydrometeorological data in **hydrotoolbox** package. Arrows indicate the class that each entity inherits from. The dotted arrow means that an `hydromet_station` class can be transformed (by the `hm_melt` method) to a `hydromet_compact` class object.

Table 1: Eight out of forty hydrometeorological variables supported in `hydromet_station` class (use `slotNames(x = "hydromet_station")` to see all of them). In the package most of them are rectangular tables containing date series in their first column (`Date` and `POSIX*` classes are allowed).

Slot	Variable
hq	water-height vs stream-discharge measurements
hw	water height level records
qh	hourly mean river discharge
qd	daily mean river discharge
qm	monthly mean river discharge
qa	annual river discharge
wspd	wind speed
wdir	wind direction
.	.
.	.
.	.

3. **Expedited visualization**. Fast and flexible visualization techniques are a powerful tool for analyzing and communicating results. The series should be able to be viewed statically and dynamically.

4. **There should be general functions for data manipulation** to reduce the need to permanently create custom functions.

5. **Open-ended design** to incorporate new objects and/or methods to continue expanding the package's functionality.

2.1 Classes and subclasses

Following the first design principle, there is a class and two subclasses for managing hydrometeorological data (Figure 1). The class, called `hydromet`, contains the station's metadata (e.g., geographic coordinates, name of the basin, province, country, unique identifier, etc.). The two subclasses correspond to **(a)** data from meteorological stations and **(b)** data derived from modeling. The `hydromet_station` subclass contains a table for each measured variable (some of them are shown in Table 1) in combination with the metadata, allowing users to store in a single object all the information concerning an hydrometeorological station.

The remaining subclass, called `hydromet_compact`, stores all series in a single table (called `compact`). It was created so that users could store input/output data from models or to save the same variable to perform regional analyzes (e.g., precipitation series recorded in different rain gauges).

2.2 Methods

The present package's version provides a set of data processing operations (methods) widely required in hydrology such as: temporal aggregation, interactive and static visualization, series modification and statistical summaries. Table 2 provides a complete list accompanied with a brief description. Also, to make the package more user-friendly, all the methods for object manipulation follow the syntax `hm_action`; the `hm_` prefix being indicative of the superclass (`hydromet`) and the action being indicative of what it does. We believe that this feature greatly eases its application.

Although more details and examples about each of the methods are given in the **Case studies** section and in the package documentation and vignettes (Toum 2022), in the next lines we explain

Table 2: Processing operations for object manipulation in the **hydrotoolbox** package.

Method	Description
hm_agg	temporally aggregates data
hm_build_generic	automatically loads raw data inside the hydromet_station object
hm_create	constructs the hydromet class and its subclasses
hm_get	extracts the required table (or metadata) from the object
hm_melt	merges several tables into a single one and set it into a hydromet_compact class object
hm_mutate	creates, modifies and deletes columns inside an object table (slot)
hm_name	changes data frame column names
hm_plot	makes static and interactive graphs of the required hydrometeorological variables
hm_report	gets basic statistic and a table with missing data
hm_set	assigns the (meta)data to an hydromet class or subclass object
hm_show	shows the head or tail of the tables inside the object
hm_subset	subsets the required table

the use of hm_build_generic. This is a core package function, since it allows to automatically load raw hydrometeorological data inside an hydromet_station class object. Figure 2 depicts five different compatible rectangular data configurations. The upper scheme shows allowed configurations for any delimited type file (Wickham, Hester, and Bryan 2022): a) all time series are stored in a single file with the first column being the date and the others numeric time series (measured variables), b) every variable is saved in a single file containing the date, the time series and other kind of data (like data-quality flags).

Another widely used file type to store hydrometeorological measurements are excel files. The bottom scheme of Figure 2 depicts the three approaches: a) all time series are stored in a single file and sheet, b) every sheet (all in the same file) contains a variable (non-numeric columns allowed), and c) multiple files storing a single variable.

The use of this function is illustrated with two examples (the data can be downloaded from https://gitlab.com/ezetoum27/hydrotoolbox/-/tree/master/my_data), but the user can find full-covered documentation using the ??hm_build_generic command.

```
#* This code example shows how to use
#* the hm_build_generic() method to
#* automatically load raw hydrometeorological
#* data

library(hydrotoolbox)
library(readr)
library(readxl)

# path to data
my_path <- "./home/my_folder/my_data"

#++++++++++++++++++++
# Rectangular data
#++++++++++++++++++++
#* Case B: multiple files (one per variable)
hm_create(class_name = "station") %>%
 hm_build_generic(path = my_path,
                 file_name = c("h_relativa_cuevas.csv",
                              "p_atm_cuevas.csv",
                              "precip_total_cuevas.csv",
                              "temp_aire_cuevas.csv",
                              "vel_viento_cuevas.csv"),
                 slot_name = c("rh", "patm", "precip",
                              "tair", "wspd"),
                 by = c("hour", "45 min", "30 min", "1 hour", "15 min"),
                 FUN = read_csv  ) %>%
 hm_show()
```
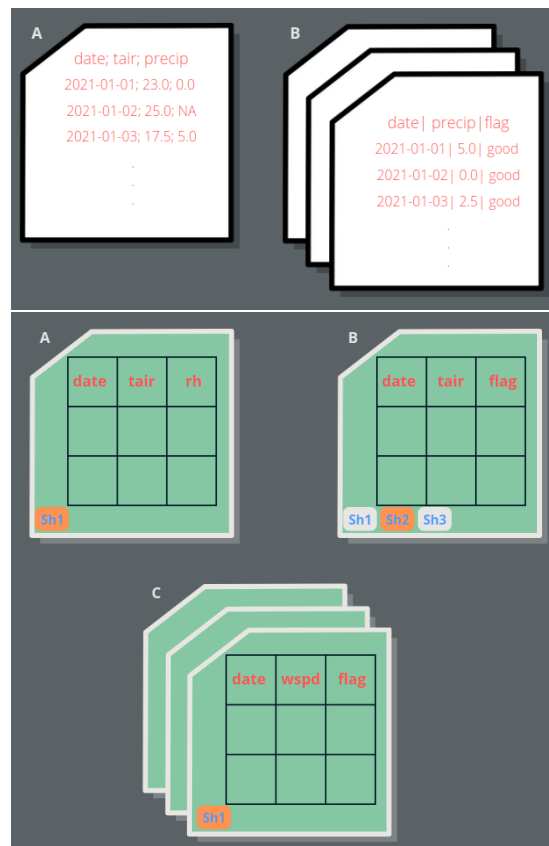
Figure 2: Rectangular data formats/file types compatible with the `hm_build_generic` function. Top scheme depicts allowed delimited file arrangements while the bottom scheme is for excel files.

```
#++++++++++++++++++++
# Excel files
#++++++++++++++++++++
#* Case B: single file - multiple sheets (one per variable)
hm_create(class_name = "station") %>%
 hm_build_generic(path = my_path,
                  file_name = "mnemos_guido.xlsx",
                  slot_name = c("qd", "evap",
                               "tair","tmax",
                               "tmin"),
                  by = c(q = "day", evap =  "day",
                        tair = "6 hour", tmax = "day",
                        tmin = NULL),
                  FUN = read_excel,
                  sheet = c(1L:5L),
                  skip = 3,
                  out_name = list( c("q_m3/s", "flag"),
                                  c("evap_mm", "flag"),
                                  c("tair", "flag"),
                                  c("tmax", "flag"),
                                  c("tmin", "flag")
                  )
   ) %>%
   hm_show()
```

3 Case studies

3.1 Daily streamflow record at the Guido station (Mendoza - Argentina)

The following example illustrates how to work with a time-series recorded in an Excel file with a single sheet (Case **A** in the **bottom** panel of Figure 2). This variable is measured at a station located in the Mendoza River basin from Argentina, a watershed that covers an area of approximately $7110\,km^2$ and provides water to almost $1m$ inhabitants (Figure 3).
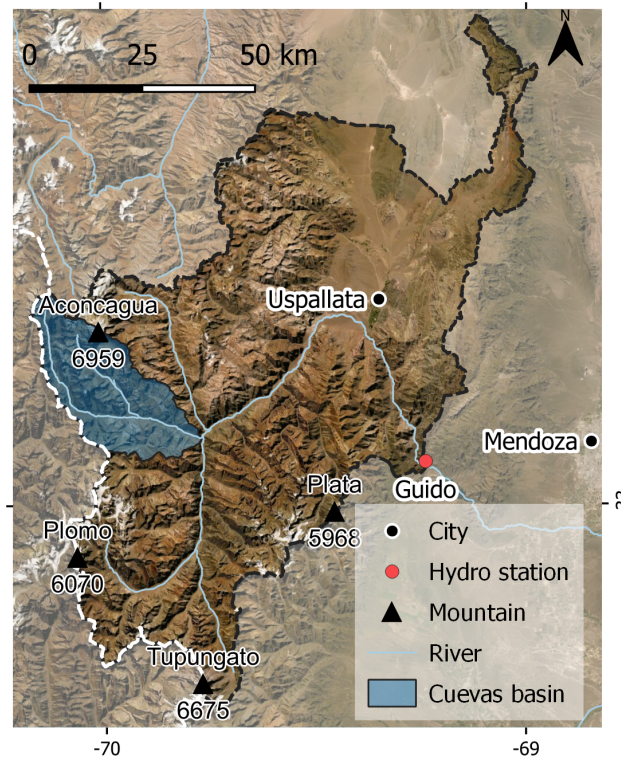

Figure 3: Map of the Mendoza basin including Guido hydrological station, cities, river and main mountain summits for reference.

In order to explore the data, we propose the following operations:

a. Build the station and obtain basic statistics of the data (`hm_build_generic`, `hm_report` and `hm_set` methods will be used).

b. Make a visual inspection of the series with `hm_plot`.

c. Smooth the streamflow record using a five days moving average windows. `hm_mutate` allows data manipulation and it can be combined with the user's or other package's functions via `FUN` and ... arguments.

d. Aggregate the series to monthly average values (via `hm_agg`).

e. Plot results for publishing using the **ggplot2** (Wickham 2016) output of `hm_plot`.

The first step is to load the original data in a `hydromet_station` that we are going to called *guido*.

```r
library(hydrotoolbox)
library(readxl)

# package's data-base
path <- system.file("extdata", package = "hydrotoolbox")

# station building
guido <-
  hm_create(class_name = "station") %>%
  hm_build_generic(path = path,
                   file_name = "snih_qd_guido.xlsx",
```

```
                    slot_name = "qd",
                    by = "1 day",
                    out_name = list("q_m3/s"),
                    sheet = 1L,
                    FUN = read_excel)
```

After executing these lines, the user will have an object that contains a table (a `tibble` in this case) with the streamflow record in the qd slot. When setting the argument by = "1 day", the function automatically fill with `NA_real_` values the missing dates and also removes duplicated records. On the other hand, using the default value (by = NULL) will ignore gaps, a feature that can be useful when loading irregular time series.

As it was previously mentioned, the package also allows setting the metadata inside the object. In this case we proposed to add the basin area using `hm_set`, but the user can apply the same command to incorporate other information (see ??hm_set). Then `hm_report` is used to get basic statistics on the streamflow record and also a table with a summary of the missing data. Note that this function also contains the string "Missings total are in the last rows." (under $miss_data$comment).

```
# set the basin area
guido <-
  guido %>%
  hm_set(basin_area = 7110)

# get streamflow's report
guido %>%
  hm_report(slot_name = "qd")

#> $stats
#>            date     q_m3/s
#> min  1956-07-01   8.00000
#> max  2020-06-30 401.00000
#> mean        <NA>  44.47951
#> sd          <NA>  36.53992
#>
#> $miss_data
#> $miss_data$`q_m3/s`
#>        first        last time_steps
#> 1 1962-09-01 1962-09-30         30
#> 2 1970-02-13 1970-02-14          2
#> 3 1976-06-22 1976-07-31         40
#> 4 1985-06-01 1985-06-30         30
#> 5       <NA>       <NA>        102
#>
#> $miss_data$comment
#> [1] "Missings total are in the last row."
```

Another essential part of the workflow is data visualization (Wickham and Grolemund 2017). The user of **hydrotoolbox** can use the `hm_plot` function and switch between static and interactive versions of the plot (using the `interactive = TRUE` argument). Internally, this function uses **ggplot2** and **plotly** (Wickham 2016; Sievert 2020) to reproduce time-series plots and provides several arguments to set axes labels, define colors, line thickness and type, transparency, among others. The following example shows how to plot the mean daily streamflow. To see the interactive version of Figure 4 please visit the HTML (online) version of the article.

```
# ggplot2 daily flow
guido %>%
  hm_plot(slot_name = "qd",
          col_name = list( "q_m3/s" ),
          interactive = FALSE,
          line_color = "dodgerblue",
          line_size = .7,
          y_lab = "Q(m3/s)",
          from = "2010-07-01",
          to = "2014-06-30")
```
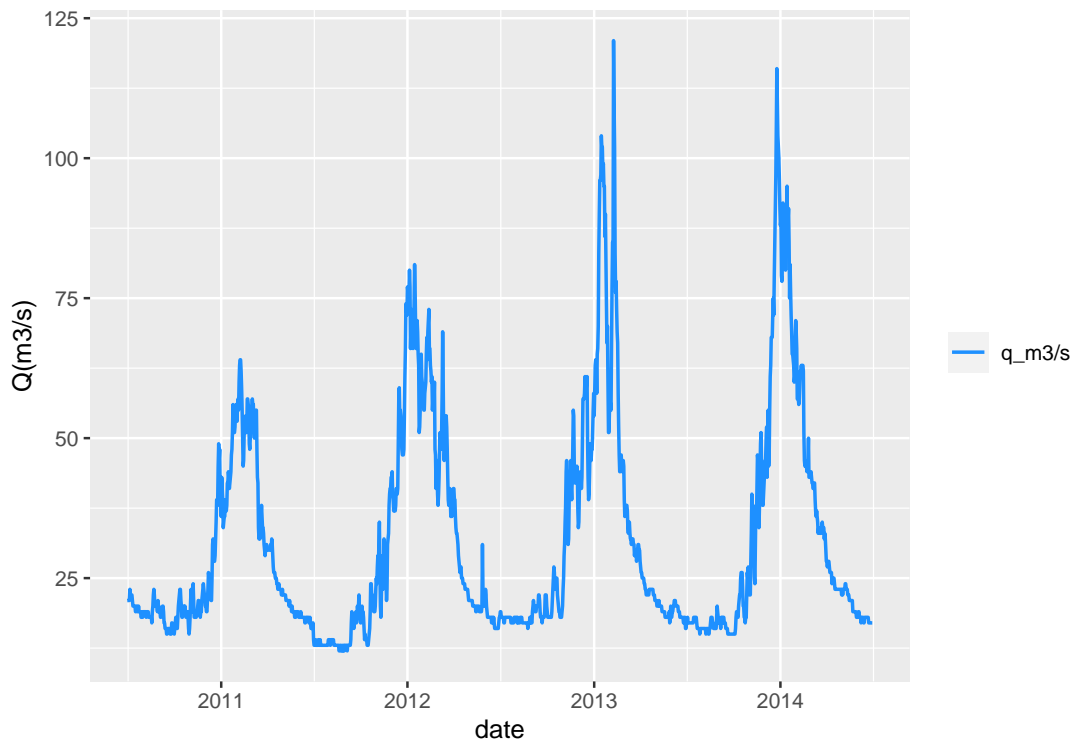
Figure 4: Plot of the Guido's streamflow time series (hydrologial years 2010/11 to 2013/14).

```
# plotly daily flow
guido %>%
  hm_plot(slot_name = "qd",
          col_name = list( "q_m3/s" ),
          interactive = TRUE,
          line_color = "dodgerblue",
          line_size = .7,
          y_lab = "Q(m3/s)",
          from = "2010-07-01",
          to = "2014-06-30")
```

As depicted in Figure 4 the time series shows the effect of instrumental oscillations during the low water flow periods. One possible solution is to smooth the series using a moving average window. To do this **hydrotoolbox** provides the hm_mutate method, a function that enable object's data manipulation. In this example we combined the aforementioned method with the package's function roll_fun. Furthermore, in order to show another hm_mutate case, we removed two periods with records below a minimum admissible value by setting them to NA_real_.

```
# smooth with roll_fun
guido <-
  guido %>%
  hm_mutate(slot_name = "qd",
            FUN = roll_fun,
            col_name = "last",
            pos = "c",
            k = 5,
            mean,
            out_name = "q_smooth")

# remove doubtful records with set_value
guido <-
  guido %>%
  hm_mutate(slot_name = "qd",
            FUN = set_value,
            col_name = "q_smooth",
            out_name = "q_set",
```

```
              value = rep(NA_real_, 2),
              from = c("1965-08-09", "1974-06-26"),
              to = c("1965-08-25", "1974-07-04") )
```

After executing these expressions, the original and new series remain stored in the same object. This feature avoids having multiple file versions and allows to track the time series modifications, achieving the second package's design principle *Modifications must be recorded in the same file*.

Moving forwards, the following code lines show how to temporally aggregate this series at a monthly resolution via hm_agg and how to extract this new series and the basin area out of the *guido* object using hm_get.

```
# aggregate daily mean streamflow
# to mean monthly values
guido <-
  guido %>%
  hm_agg(slot_name = "qd",
         col_name = "q_set",
         fun = "mean",
         period = "monthly",
         out_name = "q_mean",
         relocate = "qm",
         allow_na = 2)

# extract the table
tb_q_month <-
  guido %>%
  hm_get(slot_name = "qm")

# extract baisn area
basin_area <-
  guido %>%
  hm_get(slot_name = "basin_area")
```

In hm_agg we allow a maximum of two missing daily discharge records to compute the mean monthly streamflow and then, hm_get extracts this table so that the user may save and share it with others in *CSV* format. At this point is important to clarify that while hm_get method extracts any slot's data, the function hm_show just prints the desired slot.

To conclude this first case study the monthly streamflow is plotted (Figure 5). As the hm_plot function generates a ggplot2 object (when interactive = FALSE) the user can save and customize the graph to any style requirement.

```
# library
library(ggplot2)

# plot
gg_hm <-
  guido %>%
    hm_plot(slot_name = "qm",
         col_name = list( c("q_mean") ),
         line_color = "dodgerblue",
         line_size = .7,
         y_lab = "Q(m3/s)", x_lab = "",
         legend_lab = "Mendoza River",
         from = "1980-07-01", to = "1990-06-30")

 # customize the graph
gg_out <-
  gg_hm +
    geom_point(col = "red", size = .8) +
    theme_light() +
    scale_x_date( date_breaks = "4 month", date_labels = "%Y-%m" ) +
    scale_y_continuous(breaks = seq(0, 300, 50), limits = c(0, 300) ) +
    theme(axis.text = element_text(size = 8),
```
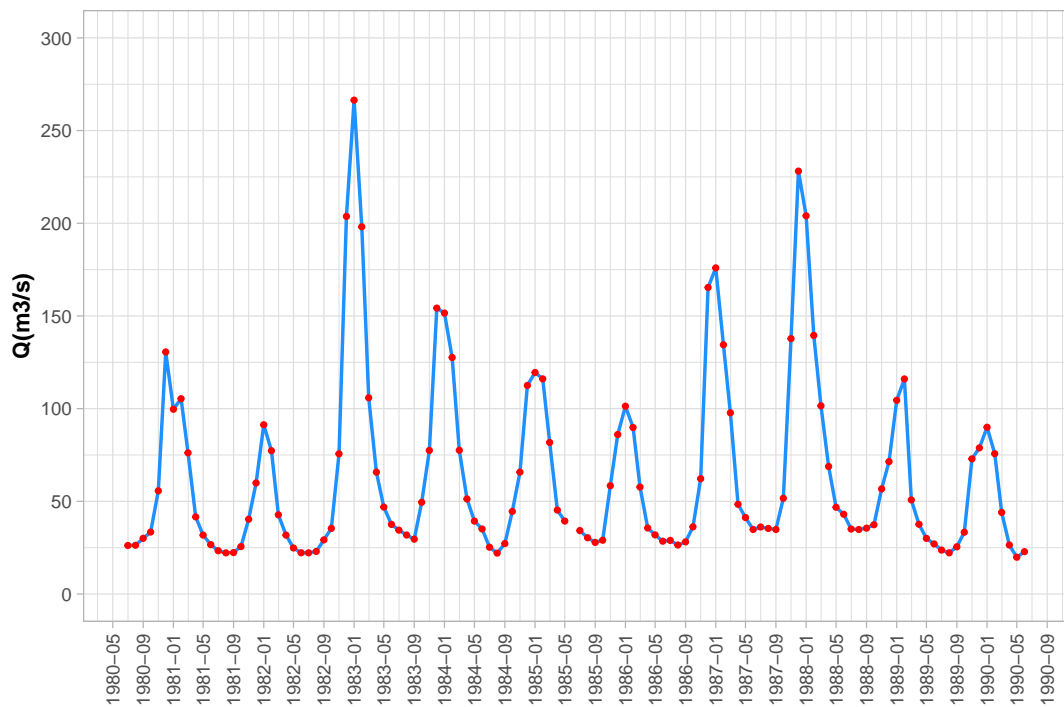
Figure 5: Example of customized time-series plot using the `hm_plot()` method in combination with labeling, scaling and theme options available on **ggplot2**.

```
              axis.title.x = element_text(size = 10, face = "bold"),
              axis.text.x = element_text(angle = 90, vjust = 0.5),
              axis.title.y = element_text(size = 10, face = "bold"),
              legend.position = "none")


gg_out
```

3.2 Post processing of the HBV.IANIGLA hydrological model

The second case study shows how to use the package for post-processing a numerical model output series. Specifically, it illustrates another `hm_mutate` case but in combination with another package function (`mutate` from **dplyr** package; Wickham et al. (2021)). The exercise proposes to transform the units of the Cuevas River basin (Figure 3) glacier mass balance simulations from *millimeters of water equivalent* to *meters of water equivalent* and then plot it (Figure 6). The data for this example has been previously loaded into an `hydromet_compact` class object and the reader can download the file from https://gitlab.com/ezetoum27/hydrotoolbox/-/tree/master/my_data).

```
# dplyr contains mutate()
library(dplyr)

# glacier mass balance simulation
cuevas_mb <- readRDS(file = "data/cuevas_mb.rds" )

cuevas_mb <-
  cuevas_mb %>%
  hm_mutate(slot_name = "compact",
            FUN = mutate,
            `bm (m we)` = round( cuevas / 1000, 2 ),
            .keep = "all"
            )

cuevas_mb %>% hm_show()

#> $compact
```
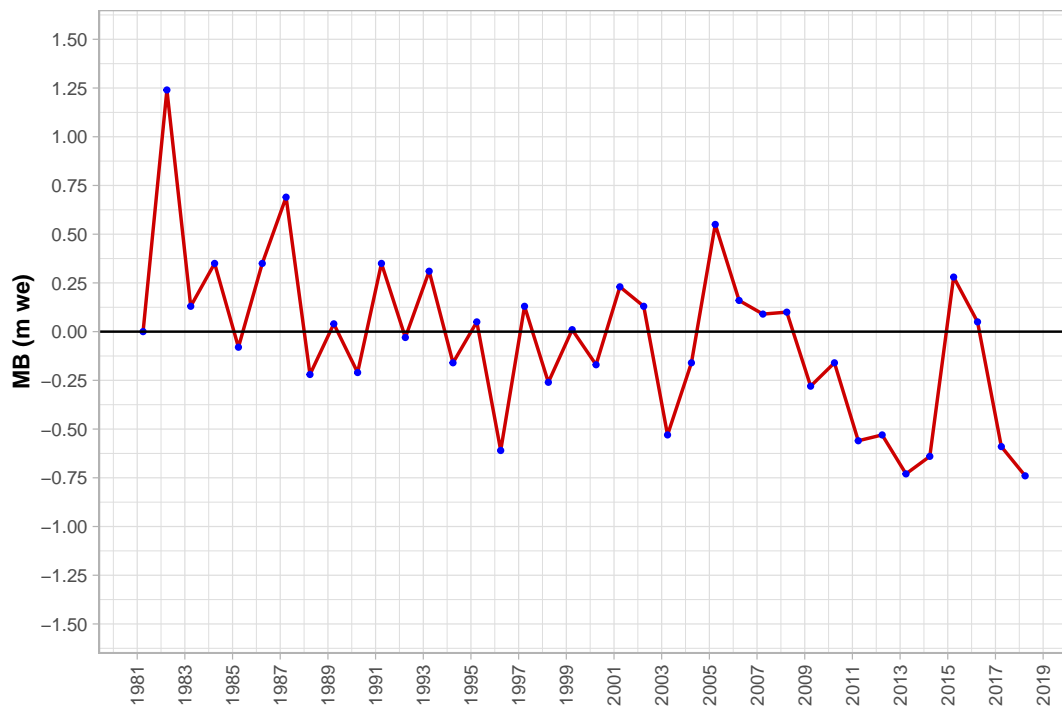
Figure 6: Model post-processing case example for the Cuevas basin annual glacier mass balance.

```
#>          date cuevas bm (m we)
#> 1 1981-04-01      0      0.00
#> 2 1982-04-01   1240      1.24
#> 3 1983-04-01    130      0.13
#> 4 1984-04-01    350      0.35
#> 5 1985-04-01    -80     -0.08
#> 6 1986-04-01    350      0.35


# use hm_plot()
gg_out <-
  cuevas_mb %>%
  hm_plot(slot_name = "compact",
          col_name = list("bm (m we)"),
          line_color = "red3",
          line_size = .7,
          x_lab = "", y_lab = "MB (m we)"  )

# customize the figure
gg_out +
geom_point(col = "blue", size = .8) +
  geom_hline(yintercept = 0) +
  theme_light() +
  scale_x_date( date_breaks = "2 year", date_labels = "%Y" ) +
  scale_y_continuous(breaks = seq(-1.5, 1.5, 0.25),
                     limits = c(-1.5, 1.5) ) +
  theme(axis.text = element_text(size = 8),
        axis.title.x = element_text(size = 10, face = "bold"),
        axis.text.x = element_text(angle = 90, vjust = 0.5),
        axis.title.y = element_text(size = 10, face = "bold"),
        legend.position = "none")
```

This simple example suggests that **hydrotoolbox** is not only suited for managing hydrometeorological series, but also for pre and post processing hydrological modeling data.

4   Discussion

The R language community has made significant advances in the development of hydrological packages, with many of them focusing on data access, numerical modeling, and pre/post-processing (Slater et al. 2019). The new **hydrotoolbox** package offers a general framework and methods for working with hydrometeorological time series data management. To our knowledge, this is the first general-purpose package for manipulating and visualizing hydrometeorological data to this date.

In the literature there are at least two other packages available in the CRAN repository that have some functionalities that could be considered similar to **hydrotoolbox**, namely **hyfo** and **hydroTSM** (Xu 2020; Zambrano-Bigiarini 2020). The first one was designed as part of the European project EUPORIAS, and it is mainly focused on data processing and visualization for hydrological and weather forecasting. Therefore, this package's functions are specialized in spatial data (e.g., NetCDF) in contrast to **hydrotoolbox** which focuses on hydrometeorological stations recorded time series. Additionally, **hyfo** uses **ggplot2** for visualization, which can be used to create publication-quality static graphs but not interactive visualizations. The other package, **hydroTSM**, is oriented towards tasks related to modeling, a shared feature with **hydrotoolbox** but lacks of a general-purpose system for hydrometeorological time series manipulation. In programming terms, we think that object-oriented programming (encapsulated or functional) is an essential mechanism for dealing with the diversity of available data, while keeping things simpler for the user. In addition, this paradigm makes **hydrotoolbox** robust in the following aspects:

  a. There are specific methods for manipulating and visualizing the objects that can be created with the package. This feature makes the workflow less prone to user error because specific and dedicated functions should be used to access the data.

  b. The objects and the modifications that the user makes to the data time series are stored in the same object, avoiding the duplication of records that can occur in other datasets.

  c. Once data has been incorporated, it is easy to get access and plot them. Additionally, **hydrotoolbox** allows saving the metadata which is a desirable feature when working with meteorological stations, data from numerical modeling, or series from large-sample hydrological datasets (Addor et al. 2020).

Although **hyfo** and **hydroTSM** overlap with **hydrotoolbox** in terms of numerical modeling and pre- and post-processing tasks, neither of these two packages (or those mentionned previously) explicitly cover hydrometeorological stations data management nor provide a general working framework for this kind of records.

   The package can be used in many applications and the user can also adapt existing functions or packages with **hydrotoolbox**'s framework. As case examples, the package's vignettes show how the tidyhydat (Albers 2017) and **weathercan** (LaZerte and Albers 2018) can be combined to work with Canadian hydrometeorological records (see `vignette(topic = "tidyhydat_can", package = "hydrotoolbox")` and `vignette(topic = "weathercan_can", package = "hydrotoolbox")`).

   In a recent publication, Addor et al. (2020) made a review of the current state of large-sample hydrology (LSH) datasets. The authors proposed general guidelines to support the creation of future LSH with some of them listed here,

  1. **provide basic data for each basin**, with streamflow records being the cornerstone. The metadata should include the name, unique identifier, river and geographical coordinates of each streamgauge, catchment area and elevation info.
  hydrotoolbox provides all the objects with this metadata.

  2. **following standards when naming variables**. Despite the fact that the package doesn't have the WATER ML-2 (https://www.ogc.org/standards/waterml - last access 2022-10-13) variable names, the purpose of this feature is to ensure the consistency and comparability of environmental datasets. This R package provides with a description for each variable, allowing possible inter-dataset comparisons.

  3. **use publicly available code for data processing**, making code for producing the LSH dataset available. This feature is important among free and open source software (FOSS).
  As a case example, the CAMELS dataset (Alvarez-Garreton et al. 2018; Addor et al. 2017) provides their users with a GitHub repository with code written in R (https://github.com/naddor/camels). Despite this, the repository contains a set of functions instead of a package that can be installed in many operating systems, with documented functions and reproducible examples. The specialized **hydrotoolbox** may provide a general framework to integrate this functions.

   Though there are similar and commonly used software in departments related to the management of hydrometeorological data, they are closed-source proprietary programs. **hydrotoolbox** is a FOSS,

which is a key feature not only from the research perspective but also for hydrological practice. Open source brings transparency to the decision-making and hydrological design processes, and helps to save license money in emerging countries (Hutton et al. 2016).

To conclude this discussion, some users may have used the older **hydroToolkit** package. This turned out to be an early version of **hydrotoolbox** that we decided to reform after using it in a large project. We realized that the original five subclasses could be reduced to two, simplifying not only their use but also improving their long-term maintainability. In addition, we decided to improve several methods (e.g., time series visualization) that were limited in functionality. Also, to elaborate the syntax of the functions, methods and their arguments, we followed the suggestions of *The tidyverse style guide* document (Wickham 2023), adapting the package to the new standards used within the community of R programmers.

5 Summary

hydrotoolbox is an new contribution to the R hydrological community, as it is specifically designed for working with hydrometeorological station records. The package allows the data to be viewed interactively3 (**plotly**) or statically (**ggplot2**) using the same method (`hm_plot`),3 it also allows the use of functions from other packages or created by the user via the `hm_mutate` method.

The case studies show two examples with different aims: the first processed a streamflow record and the second uses a numerical model output to plot results. More complete and varied examples are in the documentation of the package and in3 its vignettes (`vignette(package = "hydrotoolbox")`).

hydrotoolbox is designed to incorporate future improvements. One of them could be functions for visualizing station's geographical position (using the metadata) on an interactive map, and perhaps to allow the user to plot and compare recorded time series. This new function could use capabilities of the **leaflet** package, a JavaScript library for producing interactive maps. The main class, hydromet, could also combine the functionality of existing packages such as **raster** (Hijmans 2017) or **sf** (Pebesma 2018) to include other kind of spatial data (e.g., catchment boundary, soil types, gridded meteorological data, among others). Finally, new classes and methods could be included for processing field data from glacier mass balance studies, an activity closely related to the study of the hydrological cycle in cold regions. This kind of improvements may also make the package useful in other scientific communities such as glaciology.

References

Addor, Nans, Hong X. Do, Camila Alvarez-Garreton, Gemma Coxon, Keirnan Fowler, and Pablo A. Mendoza. 2020. "Large-Sample Hydrology: Recent Progress, Guidelines for New Datasets and Grand Challenges." *Hydrological Sciences Journal* 65 (April): 712–25. https://doi.org/10.1080/02626667.2019.1683182.

Addor, Nans, Andrew J. Newman, Naoki Mizukami, and Martyn P. Clark. 2017. "The CAMELS Data Set: Catchment Attributes and Meteorology for Large-Sample Studies." *Hydrology and Earth System Sciences* 21 (10): 5293–313. https://doi.org/10.5194/hess-21-5293-2017.

Albers, Sam. 2017. "Tidyhydat: Extract and Tidy Canadian Hydrometric Data." *The Journal of Open Source Software* 2 (20). https://doi.org/10.21105/joss.00511.

Alvarez-Garreton, Camila, Pablo A. Mendoza, Juan Pablo Boisier, Nans Addor, Mauricio Galleguillos, Mauricio Zambrano-Bigiarini, Antonio Lara, et al. 2018. "The CAMELS-CL Dataset: Catchment Attributes and Meteorology for Large Sample Studies – Chile Dataset." *Hydrology and Earth System Sciences* 22 (11): 5817–46. https://doi.org/https://doi.org/10.5194/hess-22-5817-2018.

Beven, Keith J. 2012. *Rainfall - Runoff Modelling*. 2 edition. Chichester: Wiley.

Chambers, John M. 2017. *Extending R*. CRC Press.

Conrad, O., B. Bechtel, M. Bock, H. Dietrich, E. Fischer, L. Gerlitz, J. Wehberg, V. Wichmann, and J. Böhner. 2015. "System for Automated Geoscientific Analyses (SAGA) v. 2.1.4." *Geoscientific Model Development* 8 (7). https://doi.org/https://doi.org/10.5194/gmd-8-1991-2015.

Czernecki, Bartosz, Arkadiusz Głogowski, and Jakub Nowosad. 2020. "Climate: An R Package to Access Free In-Situ Meteorological and Hydrological Datasets For Environmental Assessment." *Sustainability* 12 (1). https://doi.org/10.3390/su12010394.

De Cicco, Laura A., David Lorenz, Robert M. Hirsch, William Watkins, and Mike Johnson. 2018. *dataRetrieval: R Packages for Discovering and Retrieving Water Data Available from u.s. Federal Hydrologic Web Services* (version 2.7.7). Reston, VA: U.S. Geological Survey; U.S. Geological Survey. https://doi.org/10.5066/P9X4L3GE.

Donoho, David. 2017. "50 Years of Data Science." *Journal of Computational and Graphical Statistics* 26 (4): 745–66. https://doi.org/10.1080/10618600.2017.1384734.

Goodinson, and Louie. 1998. "WMO Solid Precipitation Measurement Intercomparison." 67. WMO.

Hijmans, Robert J. 2017. *Raster: Geographic Data Analysis and Modeling*. https://CRAN.R-project.org/package=raster.

Hutton, Christopher, Thorsten Wagener, Jim Freer, Dawei Han, Chris Duffy, and Berit Arheimer. 2016. "Most Computational Hydrology Is Not Reproducible, so Is It Really Science?" *Water Resources Research* 52 (10): 7548–55. https://doi.org/10.1002/2016WR019285.

LaZerte, Stefanie E, and Sam Albers. 2018. "weathercan: Download and Format Weather Data from Environment and Climate Change Canada." *The Journal of Open Source Software* 3 (22): 571. https://joss.theoj.org/papers/10.21105/joss.00571.

Pebesma, Edzer. 2018. "Simple Features for R: Standardized Support for Spatial Vector Data." *The R Journal* 10 (1): 439–46. https://doi.org/10.32614/RJ-2018-009.

Pierce, David. 2019. *Ncdf4: Interface to Unidata netCDF (Version 4 or Earlier) Format Data Files*. https://CRAN.R-project.org/package=ncdf4.

Ryberg, Karen R., and Aldo V. Vecchia. 2017. *waterData: Retrieval, Analysis, and Anomaly Calculation of Daily Hydrologic Time Series Data*. https://CRAN.R-project.org/package=waterData.

Sievert, Carson. 2020. *Interactive Web-Based Data Visualization with r, Plotly, and Shiny*. Chapman; Hall/CRC. https://plotly-r.com.

Slater, Louise J., Guillaume Thirel, Shaun Harrigan, Olivier Delaigue, Alexander Hurley, Abdou Khouakhi, Ilaria Prodoscimi, Claudia Vitolo, and Katie Smith. 2019. "Using R in Hydrology: A Review of Recent Developments and Future Directions." *Hydrology and Earth System Sciences Discussions*, February, 1–33. https://doi.org/https://doi.org/10.5194/hess-2019-50.

Toum, Ezequiel. 2021. *HBV.IANIGLA: Modular Hydrological Model*. https://CRAN.R-project.org/package=HBV.IANIGLA.

———. 2022. *Hydrotoolbox: Hydrological Tools for Handling Hydro-Meteorological Data Records*. https://CRAN.R-project.org/package=hydrotoolbox.

Toum, Ezequiel, Mariano H. Masiokas, Ricardo Villalba, Pierre Pitte, and Lucas Ruiz. 2021. "The HBV.IANIGLA Hydrological Model." *The R Journal* 13 (1): 378–95. https://doi.org/10.32614/RJ-2021-059.

Venkatraman, Vijaysree. 2013. "When All Science Becomes Data Science." *Science | AAAS*. https://www.sciencemag.org/careers/2013/05/when-all-science-becomes-data-science.

Vitolo, Claudia. 2017. "Hddtools: Hydrological Data Discovery Tools." *The Journal of Open Source Software* 2 (9). https://doi.org/10.21105/joss.00056.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

———. 2023. *The Tidyverse Style Guide*. https://style.tidyverse.org/.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2021. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Wickham, Hadley, and Garrett Grolemund. 2017. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. Edición: 1. Sebastopol, CA: O'Reilly Media.

Wickham, Hadley, Jim Hester, and Jennifer Bryan. 2022. *Readr: Read Rectangular Text Data*. https://CRAN.R-project.org/package=readr.

Xu, Yuanchao. 2020. *Hyfo: Hydrology and Climate Forecasting*. https://CRAN.R-project.org/package=hyfo.

Zambrano-Bigiarini, Mauricio. 2020. *hydroTSM: Time Series Management, Analysis and Interpolation for Hydrological Modelling*. https://github.com/hzambran/hydroTSM.

Ezequiel Toum
IANIGLA-CONICET
Av. Ruiz Leal s/n Parque General San Martin - Mendoza
Argentina
ORCiD: 0000-0002-4482-0559
etoum@mendoza-conicet.gob.ar

Pierre Pitte
IANIGLA-CONICET
Av. Ruiz Leal s/n Parque General San Martin - Mendoza
Argentina
ORCiD: 0000-0001-5625-8287
pierrepitte@mendoza-conicet.gob.ar

bqror: An R package for Bayesian Quantile Regression in Ordinal Models

by Prajual Maheshwari and Mohammad Arshad Rahman

Abstract This article describes an R package bqror that estimates Bayesian quantile regression in ordinal models introduced in Rahman (2016). The paper classifies ordinal models into two types and offers computationally efficient yet simple Markov chain Monte Carlo (MCMC) algorithms for estimating ordinal quantile regression. The generic ordinal model with 3 or more outcomes (labeled OR_I model) is estimated by a combination of Gibbs sampling and Metropolis-Hastings algorithm, whereas an ordinal model with exactly 3 outcomes (labeled OR_{II} model) is estimated using a Gibbs sampling algorithm only. In line with the Bayesian literature, we suggest using the marginal likelihood for comparing alternative quantile regression models and explain how to compute it. The models and their estimation procedures are illustrated via multiple simulation studies and implemented in two applications. The article also describes several functions contained within the bqror package, and illustrates their usage for estimation, inference, and assessing model fit.

1 Introduction

Quantile regression defines the conditional quantiles of a continuous dependent variable as a function of the covariates without assuming any error distribution (Koenker and Bassett 1978). The method is robust and offers several advantages over least squares regression such as desirable equivariance properties, invariance to monotone transformation of the dependent variable, and robustness against outliers (Koenker 2005; Davino, Furno, and Vistocco 2014; Furno and Vistocco 2018). However, quantile regression with discrete outcomes is more complex because quantiles of discrete data cannot be obtained through a simple inverse operation of the cumulative distribution function (cdf). Besides, discrete outcome (binary and ordinal) modeling requires location and scale restrictions to uniquely identify the parameters (see Section 2 for details). Kordas (2006) estimated quantile regression with binary outcomes using simulated annealing, while Benoit and Van den Poel (2010) proposed Bayesian binary quantile regression where a working likelihood for the latent variable was constructed by assuming the error follows an asymmetric Laplace (AL) distribution (Yu and Moyeed 2001). The estimation procedure for the latter is available in the **bayesQR** package of R software (Benoit and Van den Poel 2017). A couple of recent works on Bayesian quantile regression with binary longitudinal (panel) outcomes are Rahman and Vossmeyer (2019) and Bresson, Lacroix, and Rahman (2021). Extending the quantile framework to ordinal outcomes is more intricate due to the difficulty in satisfying the ordering of cut-points while sampling. Rahman (2016) introduced Bayesian quantile analysis of ordinal data and proposed two efficient MCMC algorithms. Since Rahman (2016), ordinal quantile regression has attracted some attention, such as in Alhamzawi (2016), Alhamzawi and Ali (2018), Ghasemzadeh, Ganjali, and Baghfalaki (2018), Rahman and Karnawat (2019), Ghasemzadeh, Ganjali, and Baghfalaki (2020), and Tian et al. (2021).

Ordinal outcomes occur in a wide class of applications in economics, finance, marketing, and the social sciences. Here, ordinal regression (e.g. ordinal probit, ordinal logit) is an important tool for modeling, analysis, and inference. Given the prevalence of ordinal models in applications and the recent theoretical developments surrounding ordinal quantile regression, an estimation package is essential so that applied econometricians and statisticians can benefit from a more comprehensive data analysis. At present, no statistical software (such as R, MATLAB, Python, Stata, SPSS, and SAS) have any package for estimating quantile regression with ordinal outcomes. The current paper fills this gap in the literature and describes the implementation of **bqror** package (version 1.6.0) for estimation and inference in Bayesian ordinal quantile regression.

The **bqror** package offers two MCMC algorithms. Ordinal model with 3 or more outcomes is estimated through a combination of Gibbs sampling (Casella and George 1992) and Metropolis-Hastings (MH) algorithm (S. Chib and Greenberg 1995). The method is implemented in the function quantregOR1. For ordinal models with exactly 3 outcomes, the package presents a Gibbs sampling algorithm that is implemented in the function quantregOR2. We recommend using this procedure for an ordinal model with 3 outcomes, since its simpler and faster. Both functions, quantregOR1 and quantregOR2, report typical posterior summaries such as the mean, standard deviation, 95% credible interval, and inefficiency factor of the model parameters. To compare alternative quantile regression models, we recommend using the marginal likelihood over the deviance information criterion (DIC). This is because the "Bayesian approach" to compare models is via the marginal likelihood (Siddhartha Chib 1995; Siddhartha Chib and Jeliazkov 2001). So, the **bqror** package also provides functions for

computing the marginal likelihood. Additionally, the package includes functions for calculating the covariate effects and example codes to produce trace plots of MCMC draws. Lastly, this paper uses the **bqror** package to demonstrate the estimation of quantile ordinal models on simulated data and real-life applications.

2 Quantile regression in ordinal models

Ordinal outcomes are common in a wide class of applications in economics, finance, marketing, social sciences, statistics in medicine, and transportation. In a typical study, the observed outcomes are ordered and categorical; so for the purpose of analysis scores/numbers are assigned to each outcome level. For example, in a study on public opinion about offshore drilling (Mukherjee and Rahman 2016), responses may be recorded as follows: 1 for 'strongly oppose', 2 for 'somewhat oppose', 3 for 'somewhat support', and 4 for 'strongly support'. The numbers have an ordinal meaning but have no cardinal interpretation. We cannot interpret a score of 2 as twice the support compared to a score of 1, or the difference in support between 2 and 3 is the same as that between 3 and 4. With ordinal outcomes, the primary modeling objective is to express the probability of outcomes as a function of the covariates. Ordinal models that have been extensively studied and employed in applications include the ordinal probit and ordinal logit models (Johnson and Albert 2000; Greene and Hensher 2010), but they only give information about the average probability of outcomes conditional on the covariates.

Quantile regression with ordinal outcomes can be estimated using the monotone equivariance property and provides information on the probability of outcomes at different quantiles. In the spirit of Albert and Chib (1993), the ordinal quantile regression model can be presented in terms of an underlying latent (or unobserved) variable z_i as follows:

$$z_i = x_i'\beta_p + \epsilon_i, \qquad \forall\, i = 1, \cdots, n, \tag{1}$$

where x_i' is a $1 \times k$ vector of covariates, β_p is a $k \times 1$ vector of unknown parameters at the p-th quantile, ϵ_i follows an AL distribution i.e., $\epsilon_i \sim AL(0, 1, p)$, and n denotes the number of observations. Note that unlike the Classical (or Frequentist) quantile regression, the error is assumed to follow an AL distribution in order to construct a (working) likelihood (Yu and Moyeed 2001). The latent variable z_i is related to the observed discrete response y_i through the following relationship,

$$\gamma_{p,j-1} < z_i \leq \gamma_{p,j} \Rightarrow y_i = j, \qquad \forall\, i = 1, \cdots, n; \; j = 1, \cdots, J, \tag{2}$$

where $\gamma_p = (\gamma_{p,0} = -\infty, \gamma_{p,1}, \ldots, \gamma_{p,J-1}, \gamma_{p,J} = \infty)$ is the cut-point vector and J denotes the number of outcomes or categories. Typically, the cut-point $\gamma_{p,1}$ is fixed at 0 to anchor the location of the distribution required for parameter identification (Jeliazkov and Rahman 2012). Given the observed data $y = (y_1, \cdots, y_n)'$, the joint density (or likelihood when viewed as a function of the parameters) for the ordinal quantile model can be written as,

$$f(y|\Theta_p) = \prod_{i=1}^{n} \prod_{j=1}^{J} P(y_i = j|\Theta_p)^{I(y_i=j)} \tag{3}$$

where $\Theta_p = (\beta_p, \gamma_p)$, $F_{AL}(\cdot)$ denotes the *cdf* of an AL distribution and $I(y_i = j)$ is an indicator function, which equals 1 if $y_i = j$ and 0 otherwise.

Working directly with the AL likelihood (3) is not convenient for MCMC sampling. Therefore, the latent formulation of the ordinal quantile model (1), following Kozumi and Kobayashi (2011), is expressed in the normal-exponential mixture form as follows,

$$z_i = x_i'\beta_p + \theta w_i + \tau \sqrt{w_i}\, u_i, \qquad \forall\, i = 1, \cdots, n, \tag{4}$$

where $\epsilon_i = \theta w_i + \tau \sqrt{w_i}\, u_i \sim AL(0, 1, p)$, $w_i \sim \mathcal{E}(1)$ is mutually independent of $u_i \sim N(0, 1)$, N and $\mathcal{E}$ denotes normal and exponential distributions, respectively; $\theta = (1 - 2p)/[p(1 - p)]$ and $\tau = \sqrt{2/[p(1 - p)]}$. Based on this formulation, we can write the conditional distribution of the latent variable as $z_i|\beta_p, w_i \sim N(x_i'\beta_p + \theta w_i, \tau^2 w_i)$ for $i = 1, \ldots, n$. This allows access to the properties of normal distribution which helps in constructing efficient MCMC algorithms.

OR$_I$ model

The term "OR$_I$ model" describes an ordinal model in which the number of outcomes (J) is equal to or greater than 3, location restriction is imposed by setting $\gamma_{p,1} = 0$, and scale restriction is achieved via constant variance (for a given value of p, variance of a standard AL distribution is constant; see Rahman (2016)). Note that in contrast to Rahman (2016), our definition of OR_I model includes an

ordinal model with exactly 3 outcomes. The location and scale restrictions are necessary to uniquely identify the parameters (see Jeliazkov, Graves, and Kutzbach (2008), and Jeliazkov and Rahman (2012) for further details and a pictorial representation).

During the MCMC sampling of the OR$_I$ model, we need to preserve the ordering of cut-points $(\gamma_{p,0} = -\infty < \gamma_{p,1} < \gamma_{p,2} < \cdots < \gamma_{p,J-1} < \gamma_{p,J} = \infty)$. This is achieved by using a monotone transformation from a compact set to the real line. Many such transformations are available (e.g., log-ratios of category bin widths, arctan, arcsin), but the **bqror** package utilizes the logarithmic transformation (Albert and Chib 2001; Rahman 2016),

$$\delta_{p,j} = \ln(\gamma_{p,j} - \gamma_{p,j-1}), \qquad 2 \leq j \leq J - 1. \tag{5}$$

The cut-points $(\gamma_{p,1}, \gamma_{p,2}, \cdots, \gamma_{p,J-1})$ can be obtained from a one-to-one mapping to $(\delta_{p,2}, \cdots, \delta_{p,J-1})$.

With all the modeling ingredients in place, we employ the Bayes' theorem and express the joint posterior distribution as proportional to the product of the likelihood and the prior distributions. As in Rahman (2016), we employ independent normal priors: $\beta_p \sim N(\beta_{p0}, B_{p0})$, $\delta_p \sim N(\delta_{p0}, D_{p0})$ in the **bqror** package. The augmented joint posterior distribution for the OR$_I$ model can thus be written as,

$$\pi(z, \beta_p, \delta_p, w | y) \propto f(y | z, \beta_p, \delta_p, w) \ \pi(z | \beta_p, w) \ \pi(w) \ \pi(\beta_p) \ \pi(\delta_p),$$

$$\propto \left\{ \prod_{i=1}^{n} f(y_i | z_i, \delta_p) \right\} \pi(z | \beta_p, w) \ \pi(w) \ \pi(\beta_p) \ \pi(\delta_p),$$

$$\propto \prod_{i=1}^{n} \left\{ \prod_{j=1}^{J} 1\{\gamma_{p,j-1} < z_i \leq \gamma_{p,j}\} \ N(z_i | x_i' \beta_p + \theta w_i, \tau^2 w_i) \ \mathcal{E}(w_i | 1) \right\}$$

$$\times \ N(\beta_p | \beta_{p0}, B_{p0}) \ N(\delta_p | \delta_{p0}, D_{p0}). \tag{6}$$

where in the likelihood function of the second line, we use the fact that the observed y_i is independent of (β_p, w) given (z_i, δ_p). This follows from equation (2) which shows that y_i given (z_i, δ_p) is determined with probability 1. In the third line, we specify the conditional distribution of the latent variable and the prior distribution on the parameters.

The conditional posterior distributions are derived from the augmented joint posterior distribution (6), and the parameters are sampled as per Algorithm 1. This algorithm is implemented in the **bqror** package. The parameter β_p is sampled from an updated multivariate normal distribution and the latent weight w is sampled element-wise from a generalized inverse Gaussian (GIG) distribution. The cut-point vector δ_p is sampled marginally of (z, w) using a random-walk MH algorithm. Lastly, the latent variable z is sampled element-wise from a truncated normal (TN) distribution.

Algorithm 1: Sampling in OR$_I$ model.

- Sample $\beta_p | z, w \sim N(\tilde{\beta}_p, \tilde{B}_p)$, where,

  - $\tilde{B}_p^{-1} = \left(\sum_{i=1}^{n} \frac{x_i x_i'}{\tau^2 w_i} + B_{p0}^{-1} \right)$ and $\tilde{\beta}_p = \tilde{B}_p \left(\sum_{i=1}^{n} \frac{x_i(z_i - \theta w_i)}{\tau^2 w_i} + B_{p0}^{-1} \beta_{p0} \right)$.

- Sample $w_i | \beta_p, z_i \sim GIG(0.5, \tilde{\lambda}_i, \tilde{\eta})$, for $i = 1, \cdots, n$, where,

  - $\tilde{\lambda}_i = \left(\frac{z_i - x_i' \beta_p}{\tau} \right)^2$ and $\tilde{\eta} = \left(\frac{\theta^2}{\tau^2} + 2 \right)$.

- Sample $\delta_p | y, \beta$ marginally of w (latent weight) and z (latent data), by generating δ_p' using a random-walk chain $\delta_p' = \delta_p + u$, where $u \sim N(0_{J-2}, \iota^2 \hat{D})$, ι is a tuning parameter and $\hat{D}$ denotes the negative inverse Hessian, obtained by maximizing the log-likelihood with respect to δ_p. Given the current value of δ_p and the proposed draw δ_p', return δ_p' with probability,

$$\alpha_{MH}(\delta_p, \delta_p') = \min \left\{ 1, \frac{f(y | \beta_p, \delta_p') \ \pi(\beta_p, \delta_p')}{f(y | \beta_p, \delta_p) \ \pi(\beta_p, \delta_p)} \right\};$$

  otherwise repeat the old value δ_p. The variance of u may be tuned as needed for appropriate step size and acceptance rate.
- Sample $z_i | y, \beta_p, \gamma_p, w \sim TN_{(\gamma_{p,j-1}, \gamma_{p,j})}(x_i' \beta_p + \theta w_i, \tau^2 w_i)$ for $i = 1, \cdots, n$, where TN denotes a truncated normal distribution and γ_p is obtained via δ_p using equation (5)

OR_{II} model

The term "OR_{II} model" is used for an ordinal model with exactly 3 outcomes (i.e., $J = 3$) where both location and scale restrictions are imposed by fixing the cut-points. Since there are only 2 cut-points and both are fixed at some values, the scale of the distribution needs to be free. Therefore, a scale parameter σ_p is introduced and the quantile ordinal model is rewritten as follows:

$$
\begin{aligned}
z_i &= x_i'\beta_p + \sigma_p\epsilon_i = x_i'\beta_p + \sigma_p\theta w_i + \sigma_p\tau\sqrt{w_i}\,u_i, & \forall\, i = 1,\cdots,n, \\
\gamma_{j-1} &< z_i \le \gamma_j \;\Rightarrow\; y_i = j, & \forall\, i = 1,\cdots,n;\; j = 1,2,3,
\end{aligned}
\tag{7}
$$

where $\sigma_p\epsilon_i \sim AL(0,\sigma_p,p)$, (γ_1,γ_2) are fixed, and recall $\gamma_0 = -\infty$ and $\gamma_3 = \infty$. In this formulation, the conditional mean of z_i is dependent on σ_p which is problematic for Gibbs sampling. So, we define a new variable $\nu_i = \sigma_p w_i \sim \mathcal{E}(\sigma_p)$ and rewrite the model in terms of ν_i. In this representation, $z_i|\beta_p,\sigma_p,\nu_i \sim N(x_i'\beta_p + \theta\nu_i, \tau^2\sigma_p\nu_i)$, the conditional mean is free of σ_p and the model is conducive to Gibbs sampling.

The next step is to specify the prior distributions required for Bayesian inference. We follow Rahman (2016) and assume $\beta_p \sim N(\beta_{p0}, B_{p0})$ and $\sigma_p \sim IG(n_0/2, d_0/2)$; where IG stands for an inverse-gamma distribution. These are the default prior distributions in the **bqror** package. Employing the Bayes' theorem, the augmented joint posterior distribution can be expressed as,

$$
\begin{aligned}
\pi(z,\beta_p,\nu,\sigma_p|y) &\propto f(y|z,\beta_p,\nu,\sigma_p)\,\pi(z|\beta_p,\nu,\sigma_p)\,\pi(\nu|\sigma_p)\,\pi(\beta_p)\,\pi(\sigma_p), \\
&\propto \left\{ \prod_{i=1}^{n} f(y_i|z_i,\sigma_p) \right\} \pi(z|\beta_p,\nu,\sigma_p)\,\pi(\nu|\sigma_p)\,\pi(\beta_p)\,\pi(\sigma_p), \\
&\propto \left\{ \prod_{i=1}^{n}\prod_{j=1}^{3} 1(\gamma_{j-1} < z_i \le \gamma_j)\, N(z_i|x_i'\beta_p + \theta\nu_i, \tau^2\sigma_p\nu_i)\, \mathcal{E}(\nu_i|\sigma_p) \right\} \\
&\quad \times\; N(\beta_p|\beta_{p0}, B_{p0})\, IG(\sigma_p|n_0/2, d_0/2),
\end{aligned}
\tag{8}
$$

where the derivations in each step are analogous to those for the OR_I model.

The augmented joint posterior distribution, given by equation (8), can be utilized to derive the conditional posterior distributions and the parameters are sampled as presented in Algorithm 2. This involves sampling β_p from an updated multivariate normal distribution and sampling σ_p from an updated IG distribution. The latent weight ν is sampled element-wise from a GIG distribution and similarly, the latent variable z is sampled element-wise from a truncated normal distribution.

Algorithm 2: Sampling in OR_{II} model.

- Sample $\beta_p|z,\sigma_p,\nu \sim N(\tilde{\beta}_p, \tilde{B}_p)$, where,

  - $\tilde{B}_p^{-1} = \left(\sum_{i=1}^{n} \frac{x_i x_i'}{\tau^2\sigma_p\nu_i} + B_{p0}^{-1} \right)$ and $\tilde{\beta}_p = \tilde{B}_p\left(\sum_{i=1}^{n} \frac{x_i(z_i - \theta\nu_i)}{\tau^2\sigma_p\nu_i} + B_{p0}^{-1}\beta_{p0} \right)$

- Sample $\sigma_p|z,\beta_p,\nu \sim IG(\tilde{n}/2, \tilde{d}/2)$, where,

  - $\tilde{n} = (n_0 + 3n)$ and $\tilde{d} = \sum_{i=1}^{n}(z_i - x_i'\beta_p - \theta\nu_i)^2/\tau^2\nu_i + d_0 + 2\sum_{i=1}^{n}\nu_i$.

- Sample $\nu_i|z_i,\beta_p,\sigma_p \sim GIG(0.5, \tilde{\lambda}_i, \tilde{\eta})$, for $i = 1,\cdots,n$, where,

  - $\tilde{\lambda}_i = \frac{(z_i - x_i'\beta_p)^2}{\tau^2\sigma_p}$ and $\tilde{\eta} = \left(\frac{\theta^2}{\tau^2\sigma_p} + \frac{2}{\sigma_p} \right)$

- Sample $z_i|y,\beta_p,\sigma_p,\nu_i \sim TN_{(\gamma_{j-1},\gamma_j)}(x_i'\beta_p + \theta\nu_i, \tau^2\sigma_p\nu_i)$ for $i = 1,\cdots,n$, and $j = 1,2,3$.

---

3 Marginal likelihood

Rahman (2016) employed DIC (Spiegelhalter et al. 2002; Gelman et al. 2013) for model comparison. However, in the Bayesian framework alternative models are typically compared using the marginal likelihood or the Bayes factor (Poirier 1995; Greenberg 2012). Therefore, we prefer using the marginal likelihood (or the Bayes factor) for comparing two or more regression models at a given quantile.

Consider a model $\mathcal{M}_s$ with parameter vector Θ_s. Let $f(y|\mathcal{M}_s, \Theta_s)$ be its sampling density and $\pi(\Theta_s|\mathcal{M}_s)$ be the prior distribution; where $s = 1, \ldots, S$. Then, the marginal likelihood for the model

$\mathcal{M}_s$ is given by the expression, $m(y|\mathcal{M}_s) = \int f(y|\Theta_s, \mathcal{M}_s)\pi(\Theta_s|\mathcal{M}_s)\, d\Theta_s$. The Bayes factor is the ratio of marginal likelihoods. So, for any two models $\mathcal{M}_q$ versus $\mathcal{M}_r$, the Bayes factor, $B_{qr} = \frac{m(y|\mathcal{M}_q)}{m(y|\mathcal{M}_r)} = \frac{\int f(y|\mathcal{M}_q,\Theta_q)\,\pi(\Theta_q|\mathcal{M}_q)\,d\Theta_q}{\int f(y|\mathcal{M}_r,\Theta_r)\,\pi(\Theta_r|\mathcal{M}_r)\,d\Theta_r}$, can be easily computed once we have the marginal likelihoods.

Siddhartha Chib (1995) and later Siddhartha Chib and Jeliazkov (2001) showed that a simple and stable estimate of marginal likelihood can be obtained from the MCMC outputs. The approach is based on the recognition that the marginal likelihood can be written as the product of likelihood function and prior density over the posterior density. So, the marginal likelihood $m(y|\mathcal{M}_s)$ for model $\mathcal{M}_s$ is expressed as,

$$m(y|\mathcal{M}_s) = \frac{f(y|\mathcal{M}_s,\Theta_s)\,\pi(\Theta_s|\mathcal{M}_s)}{\pi(\Theta_s|\mathcal{M}_s,y)}. \tag{9}$$

Siddhartha Chib (1995) refers to equation (9) as the *basic marginal likelihood identity* since it holds for all values in the parameter space, but typically computed at a high density point (e.g., mean, mode) denoted Θ^* to minimize estimation variability. The likelihood ordinate $f(y|\mathcal{M}_s,\Theta^*)$ is directly available from the model and the prior density $\pi(\Theta^*|\mathcal{M}_s)$ is assumed by the researcher. The novel part is the computation of posterior ordinate $\pi(\Theta^*|y,\mathcal{M}_s)$, which is estimated using the MCMC outputs. Since the marginal likelihood is often a large number, it is convenient to express it on the logarithmic scale. An estimate of the logarithm of marginal likelihood is given by,

$$\ln \hat{m}(y) = \ln f(y|\Theta^*) + \ln \pi(\Theta^*) - \ln \hat{\pi}(\Theta^*|y), \tag{10}$$

where we have dropped the conditioning on $\mathcal{M}_s$ for notational simplicity. The next two subsections explain the computation of marginal likelihood for the OR_I and OR_{II} quantile regression models.

Marginal likelihood for OR_I model

We know from Section 2.1 that the MCMC algorithm for estimating the OR_I model is defined by the following conditional posterior densities: $\pi(\beta_p|z,w)$, $\pi(\delta_p|\beta_p,y)$, $\pi(w|\beta_p,z)$, and $\pi(z|\beta_p,\delta_p,w,y)$. The conditional posteriors for β_p, w, and z have a known form, but that of δ_p is not tractable and is sampled using an MH algorithm. Consequently, we adopt the approach of Siddhartha Chib and Jeliazkov (2001) to calculate the marginal likelihood for the OR_I model.

To simplify the computational process (specifically, to keep the computation over a reasonable dimension), we estimate the marginal likelihood marginally of the latent variables (w, z). Moreover, we decompose the posterior ordinate as,

$$\pi(\beta_p^*, \delta_p^*|y) = \pi(\delta_p^*|y)\pi(\beta_p^*|\delta_p^*, y),$$

where $\Theta^* = (\beta_p^*, \delta_p^*)$ denotes a high density point. By placing the intractable posterior ordinate first, we avoid the MH step in the *reduced run* – the process of running an MCMC sampler with one or more parameters fixed at some value – of the MCMC sampler. We first estimate $\pi(\delta_p^*|y)$ and then the reduced conditional posterior ordinate $\pi(\beta_p^*|\delta_p^*, y)$.

To obtain an estimate of $\pi(\delta_p^*|y)$, we need to express it in a computationally convenient form. The parameter δ_p is sampled using an MH step, which requires specifying a proposal density. Let $q(\delta_p, \delta_p'|\beta_p, w, z, y)$ denote the proposal density for the transition from δ_p to δ_p', and let,

$$\alpha_{MH}(\delta_p, \delta_p') = \min\left\{1, \frac{f(y|\beta_p,\delta_p')\,\pi(\beta_p)\pi(\delta_p')}{f(y|\beta_p,\delta_p)\,\pi(\beta_p)\pi(\delta_p)} \times \frac{q(\delta_p', \delta_p|\beta_p, w, z, y)}{q(\delta_p, \delta_p'|\beta_p, w, z, y)}\right\}, \tag{11}$$

denote the probability of making the move. In the context of the model, $f(y|\beta_p,\delta_p)$ is the likelihood given by equation (3), $\pi(\beta_p)$ and $\pi(\delta_p)$ are normal prior distributions (i.e., $\beta_p \sim N(\beta_{p0}, B_{p0})$ and $\delta_p \sim N(\delta_{p0}, D_{p0})$ as specified in Section 2.1), and the proposal density $q(\delta_p, \delta_p'|\beta_p, w, z, y)$ is normal given by $f_N(\delta_p'|\delta_p, \iota^2 \hat{D})$ (see Algorithm 1 in Section 2.1). There are two points to be noted about the proposal density. First, the conditioning on (β_p, w, z, y) is only for generality and not necessary as illustrated by the use of a random-walk proposal density. Second, since our MCMC sampler utilizes a random-walk proposal density, the second ratio on the right hand side of equation (11) reduces to 1.

We closely follow the derivation in Siddhartha Chib and Jeliazkov (2001) and arrive at the following expression of the posterior ordinate,

$$\pi(\delta_p^*|y) = \frac{E_1\{\alpha_{MH}(\delta_p, \delta_p^*|\beta_p, w, z, y)\, q(\delta_p, \delta_p^*|\beta_p, w, z, y)\}}{E_2\{\alpha_{MH}(\delta_p^*, \delta_p|\beta_p, w, z, y)\}}, \tag{12}$$

where E_1 represents expectation with respect to the distribution $\pi(\beta_p, \delta_p, w, z|y)$ and E_2 represents

expectation with respect to the distribution $\pi(\beta_p, w, z | \delta_p^*, y) \times q(\delta_p^*, \delta_p | \beta_p, w, z, y)$. The quantities in equation (12) can be estimated using MCMC techniques. To estimate the numerator, we take the draw $\{\beta_p^{(m)}, \delta_p^{(m)}, w^{(m)}, z^{(m)}\}_{m=1}^M$ from the *complete MCMC run* and take an average of the quantity $\alpha_{MH}(\delta_p, \delta_p^* | \beta_p, w, z, y) \, q(\delta_p, \delta_p^* | \beta_p, w, z, y)$, where $\alpha_{MH}(\cdot)$ is given by equation (11) with δ_p' replaced by δ_p^*, and $q(\delta_p, \delta_p^* | \beta_p, w, z, y)$ is given by the normal density $f_N(\delta_p^* | \delta_p, \iota^2 \hat{D})$.

The estimation of the quantity in the denominator is tricky. This requires generating an additional sample (say of H iterations) from the reduced conditional densities: $\pi(\beta_p | w, z)$, $\pi(w | \beta_p, z)$, and $\pi(z | \beta_p, \delta_p^*, w, y)$, where note that δ_p is fixed at δ_p^* in the MCMC sampling, and thus corresponds to a *reduced run*. Moreover, at each iteration, we generate

$$\delta_p^{(h)} \sim q(\delta_p^*, \delta_p | \beta_p^{(h)}, w^{(h)}, z^{(h)}, y) \equiv f_N(\delta_p | \delta_p^*, \iota^2 \hat{D}).$$

The resulting quadruplet of draws $\{\beta_p^{(h)}, \delta_p^{(h)}, w^{(h)}, z^{(h)}\}$, as required, is a sample from the distribution $\pi(\beta_p, w, z | \delta_p^*, y) \times q(\delta_p^*, \delta_p | \beta_p, w, z, y)$. With the numerator and denominator now available, the posterior ordinate $\pi(\delta_p^* | y)$ is estimated as,

$$\hat{\pi}(\delta_p^* | y) = \frac{M^{-1} \sum_{m=1}^M \{\alpha_{MH}(\delta_p^{(m)}, \delta_p^* | \Lambda^{(m)}, y) \, q(\delta_p^{(m)}, \delta_p^* | \Lambda^{(m)}, y)\}}{H^{-1} \sum_{h=1}^H \{\alpha_{MH}(\delta_p^*, \delta_p^{(h)} | \Lambda^{(h)}, y)\}}. \tag{13}$$

where $\Lambda^{(m)} = (\beta_p^{(m)}, w^{(m)}, z^{(m)})$ and $\Lambda^{(h)} = (\beta_p^{(h)}, w^{(h)}, z^{(h)})$.

The computation of the posterior ordinate $\pi(\beta_p^* | \delta_p^*, y)$ is trivial. We have the sample of H draws $\{w^{(h)}, z^{(h)}\}$ from the reduced run, which are marginally of β_p from the distribution $\pi(w, z | \delta_p^*, y)$. These draws are utilized to estimate the posterior ordinate as,

$$\hat{\pi}(\beta_p^* | \delta_p^*, y) = H^{-1} \sum_{h=1}^H \pi(\beta_p^* | \delta_p^*, w^{(h)}, z^{(h)}, y). \tag{14}$$

Substituting the two density estimates given by equations (13) and (14) into equation (10), an estimate of the logarithm of marginal likelihood for OR_I model is obtained as,

$$\ln \hat{m}(y) = \ln f(y | \beta_p^*, \delta_p^*) + \ln \left[\pi(\beta_p^*) \pi(\delta_p^*) \right] - \ln \left[\hat{\pi}(\delta_p^* | y) \, \hat{\pi}(\beta_p^* | \delta_p^*, y) \right], \tag{15}$$

where the likelihood $f(y | \beta_p^*, \delta_p^*)$ and prior densities are evaluated at $\Theta^* = (\beta_p^*, \delta_p^*)$.

Marginal likelihood for OR_{II} model

We know from Section 2.2 that the OR_{II} model is estimated by Gibbs sampling and hence we follow Siddhartha Chib (1995) to compute the marginal likelihood. The Gibbs sampler consists of four conditional posterior densities given by $\pi(\beta_p | \sigma_p, \nu, z)$, $\pi(\sigma_p | \beta_p, \nu, z)$, $\pi(\nu | \beta_p, \sigma_p, z)$, and $\pi(z | \beta_p, \sigma_p, \nu, y)$. However, the variables (ν, z) are latent. So, we integrate them out and write the posterior ordinate as $\pi(\beta_p^*, \sigma_p^* | y) = \pi(\beta_p^* | y) \pi(\sigma_p^* | \beta_p^*, y)$, where the terms on the right hand side can be written as,

$$\pi(\beta_p^* | y) = \int \pi(\beta_p^* | \sigma_p, \nu, z, y) \, \pi(\sigma_p, \nu, z | y) \, d\sigma_p \, d\nu \, dz,$$

$$\pi(\sigma_p^* | \beta_p^*, y) = \int \pi(\sigma_p^* | \beta_p^*, \nu, z, y) \, \pi(\nu, z | \beta_p^*, y) \, d\nu \, dz,$$

and $\Theta^* = (\beta_p^*, \sigma_p^*)$ denotes a high density point, such as the mean or the median.

The posterior ordinate $\pi(\beta_p^* | y)$ can be estimated as the ergodic average of the conditional posterior density with the posterior draws of (σ_p, ν, z). Therefore, $\pi(\beta_p^* | y)$ is estimated as,

$$\hat{\pi}(\beta_p^* | y) = G^{-1} \sum_{g=1}^G \pi(\beta_p^* | \sigma_p^{(g)}, \nu^{(g)}, z^{(g)}, y). \tag{16}$$

The term $\pi(\sigma_p^* | \beta_p^*, y)$ is a reduced conditional density ordinate and can be estimated with the help of a *reduce run*. So, we generate an additional sample (say another G iterations) of $\{\nu^{(g)}, z^{(g)}\}$ from $\pi(\nu, z | \beta_p^*, y)$ by successively sampling from $\pi(\sigma_p | \beta_p^*, \nu, z)$, $\pi(\nu | \beta_p^*, \sigma_p, z)$, and $\pi(z | \beta_p^*, \sigma_p, \nu, y)$, where

note that β_p is fixed at β_p^* in each conditional density. Next, we use the draws $\{\nu^{(g)}, z^{(g)}\}$ to compute,

$$\hat{\pi}(\sigma_p^* | \beta_p^*, y) = G^{-1} \sum_{g=1}^{G} \pi(\sigma_p^* | \beta_p^*, \nu^{(g)}, z^{(g)}, y). \tag{17}$$

which is a simulation consistent estimate of $\pi(\sigma_p^* | \beta_p^*, y)$.

Substituting the two density estimates given by equations (16) and (17) into equation (10), we obtain an estimate of the logarithm of marginal likelihood,

$$\ln \hat{m}(y) = \ln f(y | \beta_p^*, \sigma_p^*) + \ln \left[\pi(\beta_p^*) \pi(\sigma_p^*) \right] - \ln \left[\hat{\pi}(\beta_p^* | y) \, \hat{\pi}(\sigma_p^* | \beta_p^*, y) \right], \tag{18}$$

where the likelihood function and prior densities are evaluated at $\Theta^* = (\beta_p^*, \sigma_p^*)$. Here, the likelihood function has the expression,

$$f(y | \beta_p^*, \sigma_p^*) = \prod_{i=1}^{n} \prod_{j=1}^{3} \left[F_{AL} \left(\frac{\gamma_j - x_i' \beta_p^*}{\sigma_p^*} \right) - F_{AL} \left(\frac{\gamma_{j-1} - x_i' \beta_p^*}{\sigma_p^*} \right) \right]^{I(y_i = j)},$$

where the cut-points γ are known and fixed for identification reasons as explained in Section 2.2.

4   Simulation studies

This section explains the data generating process for simulation studies, the functions offered in the **bqror** package, and usage of the functions for estimation and inference in ordinal quantile models.

OR_I model: data, functions, and outputs

Data Generation: The data for the simulation study of the OR_I model is generated from the regression: $z_i = x_i' \beta + \epsilon_i$, where $\beta = (-4, 5, 6)$, $(x_2, x_3) \sim U(0, 1)$, and $\epsilon_i \sim AL(0, \sigma = 1, p)$ for $i = 1, \dots, n$. Here, U and AL denote a uniform distribution and an asymmetric Laplace distribution, respectively. The z values are continuous and are classified into 4 categories based on the cut-points $(0, 2, 4)$ to generate ordinal values of y, the outcome variable. We follow the above procedure to generate 3 data sets with 500 observations (i.e., $n = 500$) each. The 3 data sets correspond to the quantile p equaling 0.25, 0.50, and 0.75, and are stored as data25j4, data50j4, and data75j4, respectively. Note that the last two letters in the name of the data object (i.e., j4) denote the number of unique outcomes in the y variable.

We now describe the major functions for Bayesian quantile estimation of OR_I model, demonstrate their usage, and note the inputs and outputs of each function.

quantregOR1: The quantregOR1 is the primary function for estimating Bayesian quantile regression in ordinal models with 3 or more outcomes (i.e., OR_I model) and implements Algorithm 1. In the code snippet below, we first read in the data and then do the following: define the ordinal response variable (y) and covariate matrix (xMat), specify the number of covariates (k) and number of outcomes (J), and set the prior means and covariances for β_p and δ_p. We then call the quantregOR1 function and specify the inputs: ordinal outcome variable (y), covariate matrix including a column of ones (xMat), prior mean (b0) and prior covariance matrix (B0) for the regression coefficients, prior mean (d0) and prior covariance matrix (D0) for the transformed cut-points, burn-in size (burn), post burn-in size (mcmc), quantile (p), the tuning factor (tune) to adjust the MH acceptance rate, and the auto correlation cutoff value (accutoff). The last input verbose, when set to TRUE will print the summary output.

In the code below, we use a diffuse normal prior $\beta_p \sim N(0_k, 10 * I_k)$ where 0_k and I_k are matrices of dimension $k \times 1$ and $k \times k$, respectively. The prior distribution can be further diffused (i.e., made less informative) by increasing the prior variance from 10 to say 100. Besides, the prior variance for δ_p should be small, such as $0.25 * I_{J-2}$, since the distribution is on the transformed cut-points, which is on the logarithmic scale. If there is a need for prior elicitation, they can be designed from previous subject based knowledge or by estimating the model on a training sample and then using the results to form prior distributions (See Greenberg (2012), for examples.)

```
library('bqror')
data("data25j4")
y <- data25j4$y
xMat <- data25j4$x
k <- dim(xMat)[2]
J <- dim(as.array(unique(y)))[1]
```

```
b0 <- array(rep(0, k), dim = c(k, 1))
B0 <- 10*diag(k)
d0 <- array(0, dim = c(J-2, 1))
D0 <- 0.25*diag(J - 2)
modelORI <- quantregOR1(y = y, x = xMat, b0, B0, d0, D0, burn = 1125,
                        mcmc = 4500, p = 0.25, tune = 1, accutoff = 0.5,
                        verbose = TRUE)
```

```
[1] Summary of MCMC draws:


         Post Mean Post Std Upper Credible Lower Credible  Inef Factor
beta_1     -3.6434   0.4293         -2.8369        -4.5073       2.3272
beta_2      4.8283   0.5597          5.9577         3.7624       2.4529
beta_3      5.9929   0.5996          7.2474         4.8819       2.7491
delta_1     0.7152   0.1110          0.9616         0.5004       3.2261
delta_2     0.7456   0.0940          0.9281         0.5543       2.1497

[1] MH acceptance rate: 31.82%
[1] Log of Marginal Likelihood: -545.72
[1] DIC: 1060.56
```

The outputs from the quantregOR1 function are the following quantities: summary.bqrorOR1, postMeanbeta, postMeandelta, postStdbeta, postStddelta, gammacp, acceptancerate, logMargLike, dicQuant, ineffactor, betadraws, and deltadraws. A detailed description of each output is presented in the **bqror** package help file. In the summary, we report the posterior mean, posterior standard deviation, 95% posterior credible (or probability) interval, and the inefficiency factor of the quantile regression coefficients β_p and transformed cut-points δ_p. These quantities are presented in the last five columns and labeled appropriately.

The posterior means of (β_p, δ_p) are close to the true values used to generate the data with small standard deviations. So, the quantregOR1 function is successful in recovering the true values of the parameters. The inefficiency factor is computed from the MCMC samples using the batch-means method (Greenberg 2012). They indicate the cost of working with correlated samples. For example, an inefficiency factor of 3 implies that it takes 3 correlated draws to get one independent draw. As such, low inefficiency factor indicates better mixing and a more efficient MCMC algorithm. Inefficiency factor also bears a direct relationship with effective sample size, where the latter can be obtained as the total number of (post burn-in) MCMC draws divided by the inefficiency factor (Siddhartha Chib 2012). The inefficiency factors for (β_p, δ_p) are stored in the object modelOR1 of class bqrorOR1 and can be obtained by calling modelORI$ineffactor.

The third last row displays the random-walk MH acceptance rate for δ_p, for which the preferred acceptance rate is around 30 percent. The last two rows present the model comparison measures, the logarithm of marginal likelihood and the DIC. The logarithm of marginal likelihood is computed using the MCMC outputs from the complete and reduced runs as explained in Section 3, while the principle for computing the DIC is presented in Gelman et al. (2013). For any two competing models at the same quantile, the model with a higher (lower) marginal likelihood (DIC) provides a better model fit.

While the two model comparison measures are printed as part of the summary output, they can also be called individually. For example, the logarithm of marginal likelihood can be obtained by calling modelORI$logMargLike. Whereas, the DIC can be obtained by calling modelORI$dicQuant$DIC. Two more quantities that are part of the object modelORI$dicQuant are effective number of parameters denoted p_D and the deviance computed at the posterior mean. They can be obtained by calling modelORI$dicQuant$pd and modelORI$dicQuant$dev, respectively. Besides, post estimation, one may also use the command modelORI$summary or summary.bqrorOR1(modelORI) to extract and print the summary output.

covEffectOR1: The function covEffectOR1 computes the average covariate effect for different outcomes of OR_I model at a specified quantile, marginally of the parameters and the remaining covariates. While a demonstration of this function is best understood in a real-life study and is presented in the application section, here we present the mechanics behind the computation of average covariate effect.

Suppose, we want to compute the average covariate effect when the l-th covariate $\{x_{i,l}\}$ is set to the values a and b, denoted as $\{x_{i,l}^a\}$ and $\{x_{i,l}^b\}$, respectively. We split the covariate and parameter vectors as follows: $x_i^a = (x_{i,l}^a, x_{i,-l})$, $x_i^b = (x_{i,l}^b, x_{i,-l})$, and $\beta_p = (\beta_{p,l}, \beta_{p,-l})$, where $-l$ in the subscript denotes all covariates (parameters) except the l-th covariate (parameter). We are interested in the distribution of the difference $\{\Pr(y_i = j|x_{i,l}^b) - \Pr(y_i = j|x_{i,l}^a)\}$ for $1 \leq j \leq J$, marginalized over $\{x_{i,-l}\}$

and (β_p, δ_p), given the data $y = (y_1, \cdots, y_n)'$. We marginalize the covariates using their empirical distribution and the parameters based on the posterior distribution.

To obtain draws from the distribution $\{\Pr(y_i = j | x_{i,l}^b) - \Pr(y_i = j | x_{i,l}^a)\}$, we use the method of composition (see Siddhartha Chib and Jeliazkov (2006), Rahman and Vossmeyer (2019), and Bresson, Lacroix, and Rahman (2021), for additional details). In this process, we randomly select an individual, extract the corresponding sequence of covariate values, draw a value (β_p, δ_p) from the posterior distributions, and evaluate $\{\Pr(y_i = j | x_i^b, \beta_p, \delta_p) - \Pr(y_i = j | x_i^a, \beta_p, \delta_p)\}$, where,

$$\Pr(y_i = j | x_i^q, \beta_p, \delta_p)$$
$$= F_{AL}(\gamma_{p,j} - x_{i,l}^q \beta_{p,l} - x_{i,-l}' \beta_{p,-l}) - F_{AL}(\gamma_{p,j-1} - x_{i,l}^q \beta_{p,l} - x_{i,-l}' \beta_{p,-l}),$$

for $q = b, a$ and $1 \leq j \leq J$. This process is repeated for all remaining individuals and other MCMC draws from the posterior distribution. Finally, the average covariate effect (ACE) for outcome j (for $1 \leq j \leq J$) is calculated as the mean of the difference in pointwise probabilities,

$$\frac{1}{M} \frac{1}{n} \sum_{m=1}^{M} \sum_{i=1}^{n} \left[\Pr(y_i = j | x_i^b, \beta_p^{(m)}, \delta_p^{(m)}) - \Pr(y_i = j | x_i^a, \beta_p^{(m)}, \delta_p^{(m)}) \right], \qquad (19)$$

where $(\beta_p^{(m)}, \delta_p^{(m)})$ is an MCMC draw of (β_p, δ_p), and M is the number of post burn-in MCMC draws.

OR_{II} model: data, function, and outputs

Data Generation: The data generating process for the OR_{II} model closely resembles that of OR_I model. In particular, 500 observations are generated for each value of p from the regression model: $z_i = x_i'\beta + \epsilon_i$, where $\beta = (-4, 6, 5)$, $(x_2, x_3) \sim U(0,1)$ and $\epsilon_i \sim AL(0, \sigma = 1, p)$ for $i = 1, \ldots, n$. The continuous values of z are classified based on the cut-points $(0, 3)$ to generate 3 ordinal values for y, the outcome variable. Once again, we choose p equal to 0.25, 0.50, and 0.75 to generate three samples from the model, which are referred to as data25j3, data50j3, and data75j3, respectively. The last two letters in the names of the data objects (i.e., j3) denote the number of unique outcomes in the y variable.

quantregOR2: The function quantregOR2 implements Algorithm 2 and is the main function and for estimating Bayesian quantile regression in OR_{II} model i.e., an ordinal model with exactly 3 outcomes. In the code snippet below, we first read the data, define the required quantities, and then call the quantregOR2 for estimating the quantile model. The function inputs are as follows: the ordinal outcome variable (y), covariate matrix including a column of ones (xMat), prior mean (b0) and prior covariance matrix (B0) for β_p, prior shape (n0) and scale (d0) parameters for σ_p, second cut-point (gammacp2), burn-in size (burn), post burn-in size (mcmc), quantile (p), auto correlation cutoff value (accutoff), and the verbose option which when set to TRUE (FALSE) will (not) print the outputs. We use a relatively diffuse prior distributions on (β_p, σ_p) to allow the data to speak for itself.

```
library('bqror')
data("data25j3")
y <- data25j3$y
xMat <- data25j3$x
k <- dim(xMat)[2]
b0 <- array(rep(0, k), dim = c(k, 1))
B0 <- 10*diag(k)
n0 <- 5
d0 <- 8
modelORII <- quantregOR2(y = y, x = xMat, b0, B0, n0, d0, gammacp2 = 3,
                        burn = 1125, mcmc = 4500, p = 0.25, accutoff = 0.5,
                        verbose = TRUE)

[1] Summary of MCMC draws:


        Post Mean Post Std Upper Credible Lower Credible  Inef Factor
beta_1   -3.8900   0.4560        -3.0578        -4.8346      2.4784
beta_2    5.8257   0.5341         6.9263         4.8243      2.1231
beta_3    4.7194   0.5227         5.7502         3.7194      2.2693
sigma     0.8968   0.0763         1.0587         0.7626      2.4079


[1] Log of Marginal Likelihood: -404.34
```

```
[1] DIC: 790.72
```

The outputs from the `quantregOR2` function are the following: `summary.bqrorOR2`, `postMeanbeta`, `postMeansigma`, `postStdbeta`, `postStdsigma`, `logMargLike`, `dicQuant`, `ineffactor`, `betadraws`, and `sigmadraws`. A detailed description of each output is presented in the **bqror** package help file. Once again, we summarize the MCMC draws by reporting the posterior mean, posterior standard deviation, 95% posterior credible (or probability) interval, and the inefficiency factor of the quantile regression coefficients β_p and scale parameter σ_p. The output also exhibits the logarithm of marginal likelihood and the DIC for OR_{II} model, where the former is computed using the Gibbs output as explain in Section 3.

covEffectOR2: The function `covEffectOR2` computes the average covariate effect for the 3 outcomes of OR_{II} model at a specified quantile, marginally of the parameters and remaining covariates. The principle underlying the computation is analogous to that of OR_I model and is explained below. An implementation of the function is presented in the tax policy application.

Suppose, we are interested in computing the average covariate effect for the l-th covariate $\{x_{i,l}\}$ for two different values a and b, and split the covariate and parameter vectors as: $x_i^a = (x_{i,l}^a, x_{i,-l})$, $x_i^b = (x_{i,l}^b, x_{i,-l})$, $\beta = (\beta_{p,l}, \beta_{p,-l})$. We are interested in the distribution of the difference $\{\Pr(y_i = j|x_{i,l}^b) - \Pr(y_i = j|x_{i,l}^a)\}$ for $1 \leq j \leq J = 3$, marginalized over $\{x_{i,-l}\}$ and (β_p, σ_p), given the data $y = (y_1, \cdots, y_n)'$. We again employ the method of composition i.e., randomly select an individual, extract the corresponding sequence of covariate values, draw a value (β_p, σ_p) from their posterior distributions, and lastly evaluate $\{\Pr(y_i = j|x_i^b, \beta_p, \sigma_p) - \Pr(y_i = j|x_i^a, \beta_p, \sigma_p)\}$, where

$$
\begin{aligned}
&\Pr(y_i = j|x_i^q, \beta_p, \sigma_p) \\
&= F_{AL}\left(\frac{\gamma_{p,j} - x_{i,l}^q \beta_{p,l} - x_{i,-l}' \beta_{p,-l}}{\sigma_p}\right) - F_{AL}\left(\frac{\gamma_{p,j-1} - x_{i,l}^q \beta_{p,l} - x_{i,-l}' \beta_{p,-l}}{\sigma_p}\right),
\end{aligned}
$$

for $q = b, a$, and $1 \leq j \leq J = 3$. This process is repeated for other individuals and the remaining Gibbs draws to compute the *ACE* for outcome $j (= 1, 2, 3)$ as below,

$$
\frac{1}{G}\frac{1}{n}\sum_{g=1}^{G}\sum_{i=1}^{n}\left[\Pr(y_i = j|x_i^b, \beta_p^{(g)}, \sigma_p^{(g)}) - \Pr(y_i = j|x_i^a, \beta_p^{(g)}, \sigma_p^{(g)})\right], \tag{20}
$$

where $(\beta_p^{(g)}, \sigma_p^{(g)})$ is a Gibbs draw of (β_p, σ_p) and G is the number of post burn-in Gibbs draws.

5 Applications

In this section, we consider the educational attainment and tax policy applications from Rahman (2016) to demonstrate the real data applications of the proposed **bqror** package. While the educational attainment study shows the implementation of ordinal quantile regression in OR_I model, the tax policy study highlights the use of ordinal quantile regression in OR_{II} model. Data for both the applications are included as a part of the **bqror** package.

Educational attainment

In this application, the goal is to study the effect of family background, individual level variables, and age cohort on educational attainment of 3923 individuals using data from the National Longitudinal Study of Youth (NLSY, 1979) (Jeliazkov, Graves, and Kutzbach 2008; Rahman 2016). The dependent variable in the model, education degrees, has four categories: *(i) Less than high school, (ii) High school degree, (iii) Some college or associate's degree*, and *(iv) College or graduate degree*. A bar chart of the four categories is presented in Figure 1. The independent variables in the model include intercept, square root of family income, mother's education, father's education, mother's working status, gender, race, indicator variables to point whether the youth lived in an urban area or South at the age of 14, and three indicator variables to indicate the individual's age in 1979 (serves as a control for age cohort effects).

To estimate the Bayesian ordinal quantile model on educational attainment data, we load the **bqror** package, prepare the required inputs and feed them into the `quantregOR1` function. Specifically, we specify the outcome variable, covariate matrix (with covariates in order as in Rahman 2016), prior distributions for (β_p, δ_p), burn-in size, number of post burn-in MCMC iterations, quantile value ($p = 0.5$ for this illustration), and the values for tuning factor and autocorrelation cutoff.
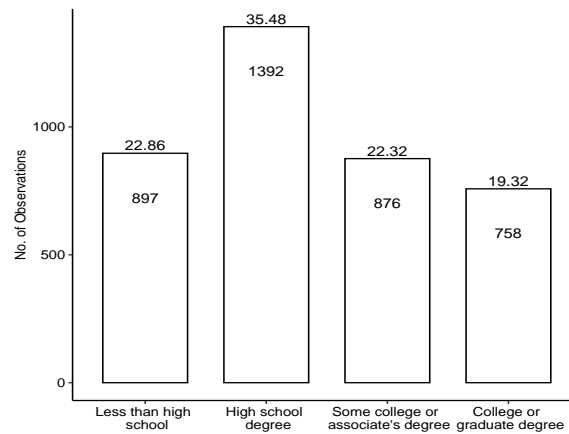
```
library('bqror')
```

Figure 1: Bar chart showing the different categories of educational attainment. The number of responses (percentage) for each category are shown inside (at the top) of each bar.

```
data("Educational_Attainment")
data <- na.omit(Educational_Attainment)
data$fam_income_sqrt <- sqrt(data$fam_income)
cols <- c("mother_work","urban","south", "father_educ","mother_educ",
          "fam_income_sqrt","female", "black","age_cohort_2","age_cohort_3",
          "age_cohort_4")
x <- data[cols]
x$intercept <- 1
xMat <- x[,c(12,6,5,4,1,7,8,2,3,9,10,11)]
yOrd <- data$dep_edu_level
k <- dim(xMat)[2]
J <- dim(as.array(unique(yOrd)))[1]
b0 <- array(rep(0, k), dim = c(k, 1))
B0 <- 1*diag(k)
d0 <- array(0, dim = c(J-2, 1))
D0 <- 0.25*diag(J - 2)
p <- 0.5

EducAtt <- quantregOR1(y = yOrd, x = xMat, b0, B0, d0, D0, burn = 1125,
                       mcmc = 4500, p, tune=1, accutoff = 0.5, TRUE)

[1] Summary of MCMC draws:
```

	Post Mean	Post Std	Upper Credible	Lower Credible	Inef Factor
intercept	-3.2546	0.2175	-2.8350	-3.6798	2.3143
fam_income_sqrt	0.2788	0.0230	0.3254	0.2337	2.1396
mother_educ	0.1242	0.0190	0.1619	0.0878	1.9499
father_educ	0.1866	0.0154	0.2165	0.1578	2.3062
mother_work	0.0664	0.0821	0.2287	-0.0930	1.9349
female	0.3492	0.0786	0.5070	0.2022	1.9086
black	0.4413	0.0997	0.6400	0.2506	1.9270
urban	-0.0777	0.0971	0.1104	-0.2712	1.9019
south	0.0842	0.0880	0.2529	-0.0895	1.9153
age_cohort_2	-0.0345	0.1192	0.1963	-0.2660	1.7502
age_cohort_3	-0.0426	0.1223	0.2033	-0.2849	1.9053
age_cohort_4	0.4938	0.1212	0.7256	0.2570	1.7512
delta_1	0.8988	0.0276	0.9534	0.8461	4.6186
delta_2	0.5481	0.0313	0.6146	0.4890	3.6003

```
[1] MH acceptance rate: 26.8%
[1] Log of Marginal Likelihood: -4923.48
[1] DIC: 9781.91
```
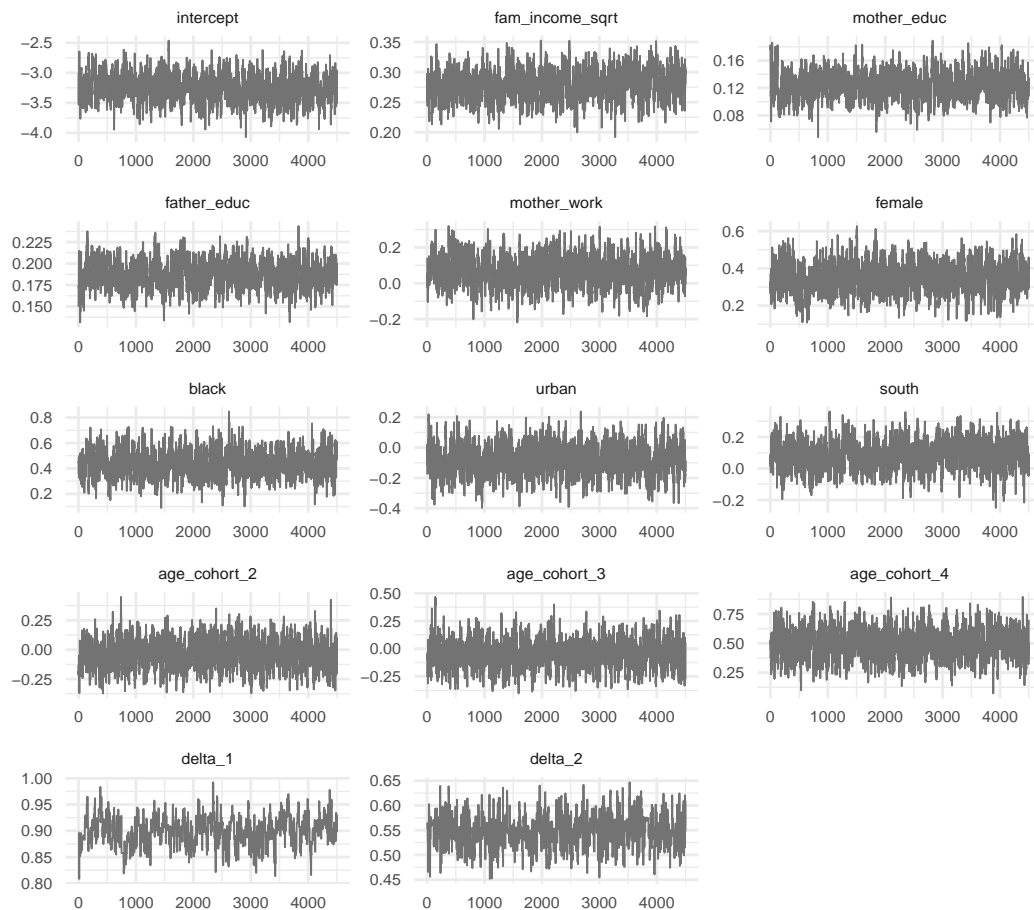
Figure 2: Trace plots of the MCMC draws in the educational attainment study.

The posterior results[1] from the MCMC draws are summarized above. In the summary, we report the posterior mean and posterior standard deviation of the parameters (β_p, δ_p), 95% posterior credible interval, and the inefficiency factors. Additionally, the summary displays the MH acceptance rate of δ_p, the logarithm of marginal likelihood, and the DIC.

```
mcmc <- 4500
burn <- round(0.25*mcmc)
nsim <- mcmc + burn
mcmcDraws <- cbind(t(EducAtt$betadraws), t(EducAtt$deltadraws))
color_scheme_set('darkgray')
bayesplot_theme_set(theme_minimal())
mcmc_trace(mcmcDraws[(burn+1):nsim, ], facet_args = list(ncol = 3))
```

Figure 2 presents the trace plots of the MCMC draws, which can be generated by loading the **bayesplot** package and using the codes presented above. The purpose of trace plots is to show that the Markov chains have converged to the joint posterior distribution, such as shown in Figure 2. The idea is that the trace plots should show variation around a central value if the chain has converged, rather than drift without settling down.

Next, we utilize the covEffectOR1 function to compute the average covariate effect for a \$10,000 increase in family income on the four categories of educational attainment. In general, the calculation of average covariate effect requires creation of either one or two new covariate matrices depending whether the covariate is continuous or indicator (binary), respectively. Since income is a continuous

---

[1]The results reported here are slightly different from those presented in Rahman (2016). This difference in results, aside from lesser number of MCMC draws, is due to a different approach in sampling from the GIG distribution. Rahman (2016) employed the ratio of uniforms method to sample from the GIG distribution (Dagpunar 2007), while the current paper utilizes the rgig function in the **GIGrvg** package that overcomes the disadvantages associated with sampling using the ratio of uniforms method (see **GIGrvg** documentation for further details). Also, see Devroye (2014) for an efficient sampling technique from a GIG distribution.

variable, a modified covariate matrix is created by adding \$10,000 to each observation of family income. This is xMod2 in the code below and the increased income variable corresponds to $x_{i,l}^b$ for $i = 1, \cdots, n$ in Section 4.1. The second covariate matrix xMod1 (in the code below) is simply the covariate or design matrix xMat; so with reference to Section 4.1, $x_{i,l}^a = x_{i,l}$ for $i = 1, \cdots, n$. When the covariate of interest is an indicator variable, then xMod1 also requires modification as illustrated in the tax policy application.

We now call the covEffectOR1 function and supply the inputs to get the results.

```
xMat1 <- xMat
xMat2 <- xMat
xMat2$fam_income_sqrt <- sqrt((xMat1$fam_income_sqrt)^2 + 10)
EducAttCE <- covEffectOR1(EducAtt, yOrd, xMat1, xMat2, p = 0.5, verbose = TRUE)

[1] Summary of Covariate Effect:

          Covariate Effect
Category_1         -0.0314
Category_2         -0.0129
Category_3          0.0193
Category_4          0.0250
```

The results shows that at the 50th quantile and for a \$10,000 increase in family income, the probability of obtaining *less than high school* (*high school degree*) decreases by 3.14 (1.29) percent, while the probability of achieving *some college or associate's degree* (*college or graduate degree*) increases by 1.93 (2.50) percent.

Tax Policy

Here, the objective is to analyze the factors that affect public opinion on the proposal to raise federal taxes for couples (individuals) earning more than \$250,000 (\$200,000) per year in the United States (US). The proposal was designed to extend the Bush Tax cuts for the lower and middle income classes, but restore higher rates for the richer class. Such a policy is considered pro-growth, since it is aimed to promote economic growth in the US by augmenting consumption among the low-middle income families. After extensive debate, the proposed policy received a two year extension and formed a part of the "Tax Relief, Unemployment Insurance Reauthorization, and Job Creation Act of 2010".

The data for the study was taken from the 2010-2012 American National Election Studies (ANES) on the Evaluations of Government and Society Study 1 (EGSS 1) and contains 1,164 observations. The dependent variable in the model, individual's opinion on tax increase, has 3 categories: *Oppose*, *Neither favor nor oppose*, and *Favor* (see Figure 3). The covariates included in the model are the intercept, indicator variables for employment status, income above \$75,000, bachelors' degree, post-bachelors' degree, computer ownership, cell phone ownership, and white race.

To estimate the quantile model on public opinion about federal tax increase, we load the **bqror** package, prepare the data, and provide the inputs into the quantregOR2 function. Specifically, we
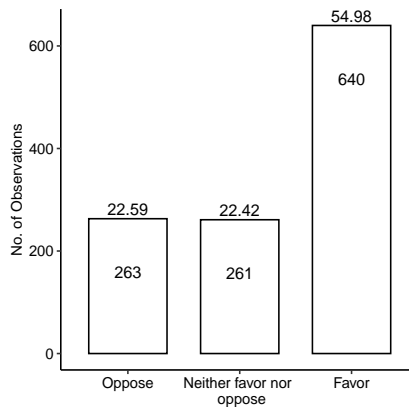

Figure 3: Bar chart for public opinion on tax increase. The number of responses (percentage) for each category are shown inside (at the top) of each bar.

define the outcome variable, covariate matrix (with covariates in order as in Rahman 2016), specify the prior distributions for (β_p, σ_p), choose the second cut-off value, burn-in size, number of post burn-in MCMC iterations, and values for quantile ($p = 0.5$ for this illustration) and autocorrelation cutoff.

```
library(bqror)
data("Policy_Opinion")
data <- na.omit(Policy_Opinion)
cols <- c("Intercept","EmpCat","IncomeCat","Bachelors","Post.Bachelors",
          "Computers","CellPhone", "White")
x <- data[cols]
xMat <- x[,c(1,2,3,4,5,6,7,8)]
yOrd <- data$y
k <- dim(x)[2]
b0 <- array(rep(0, k), dim = c(k, 1))
B0 = 1*diag(k)
n0 <- 5
d0 <- 8
FedTax <- quantregOR2(y = yOrd, x = xMat, b0, B0, n0, d0, gammacp2 = 3,
                      burn = 1125, mcmc = 4500, p = 0.5, accutoff = 0.5, TRUE)


[1] Summary of MCMC draws :

                Post Mean Post Std Upper Credible Lower Credible  Inef Factor
Intercept          2.0142   0.4553          2.9071         1.0959       1.4473
EmpCat             0.2496   0.2953          0.8294        -0.3270       1.6760
IncomeCat         -0.5083   0.3323          0.1329        -1.1580       1.6700
Bachelors          0.0809   0.3744          0.8569        -0.6324       1.6726
Post.Bachelors     0.5082   0.4406          1.3964        -0.3435       1.5053
Computers          0.7167   0.3509          1.4078         0.0219       1.4975
CellPhone          0.8464   0.4027          1.6191         0.0444       1.4524
White              0.0627   0.3659          0.7579        -0.6502       1.4931
sigma              2.2205   0.1442          2.5300         1.9552       2.3161


[1] Log of Marginal Likelihood: -1174.11
[1] DIC: 2334.58
```

The results (see Footnote 1) from the MCMC draws are summarized above, where we report the posterior mean, posterior standard deviation, 95% posterior credible interval, and the inefficiency factor of the parameters (β_p, σ_p). Additionally, the summary displays the logarithm of marginal likelihood and the DIC. Figure 4 presents the trace plots of the Gibbs draws, which can be generated by loading the **bayesplot** package and using the codes below.

```
mcmc <- 500
burn <- round(0.25*mcmc)
nsim <- mcmc + burn
mcmcDraws <- cbind(t(FedTax$betadraws), t(FedTax$sigmadraws))
color_scheme_set('darkgray')
bayesplot_theme_set(theme_minimal())
mcmc_trace(mcmcDraws[(burn+1):nsim, ], facet_args = list(ncol = 3))
```

Finally, we utilize the `covEffectOR2` function to demonstrate the calculation of average covariate effect within the OR_{II} framework. Below, we compute the average covariate effect for computer ownership (assume this is the l-th variable) on the 3 categories of public opinion about the tax policy. Here, the covariate is an indicator variable since you may either own a computer (coded as 1) or not (coded as 0). So, we need to create two modified covariate matrices. In the code snippet below, the first matrix `xMat1` and the second matrix `xMat2` are created by replacing the column on computer ownership with a column of zeros and ones, respectively. With reference to notations in Section 4.1 and Section 4.2, $x^a_{i,l} = 0$ and $x^b_{i,l} = 1$ for $i = 1, \cdots, n$. We then call the `covEffectOR2` function and supply the inputs to get the results.

```
xMat1 <- xMat
xMat1$Computers <- 0
xMat2 <- xMat
xMat2$Computers <- 1
```
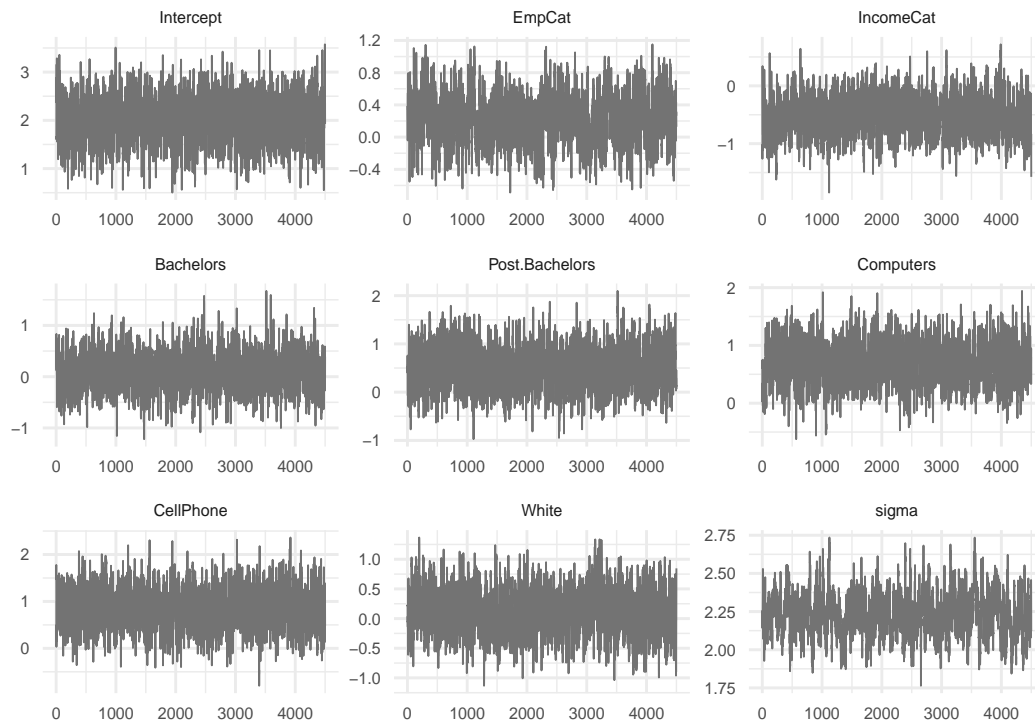
Figure 4: Trace plots of the MCMC draws in the tax policy study.

```
FedTaxCE <- covEffectOR2(FedTax, yOrd, xMat1, xMat2, gammacp2 = 3, p = 0.5,
                         verbose = TRUE)

[1] Summary of Covariate Effect:

            Covariate Effect
Category_1           -0.0396
Category_2           -0.0331
Category_3            0.0726
```

The result on covariate effect shows that at the 50th quantile, ownership of computer decreases probability for the first category *Oppose* (*Neither favor nor oppose*) by 3.96 (3.31) percent, and increases the probability for the third category *Favor* by 7.26 percent.

6 Conclusion

A wide class of applications in economics, finance, marketing, and the social sciences have dependent variables which are ordinal in nature (i.e., they are discrete and ordered, and are characterized by an underlying continuous variable). Modeling and analysis of such variables has been typically confined to ordinal probit or ordinal logit models, which offer information on the average probability of outcome variable given the covariates. However, a recently proposed method by Rahman (2016) allows Bayesian quantile modeling of ordinal data and thus presents the tool for a more comprehensive analysis and inference. The prevalence of ordinal responses in applications is well known and hence a software package that allows Bayesian quantile analysis with ordinal data will be of immense interest to applied researchers from different fields, including economics and statistics.

The current paper presents an implementation of the **bqror** package – the only package available for estimation and inference of Bayesian quantile regression in ordinal models. The package offers two MCMC algorithms for estimating ordinal quantile models. An ordinal quantile model with 3 or more outcomes is estimated by a combination of Gibbs sampling and MH algorithm, while estimation of an ordinal quantile model with exactly 3 outcomes utilizes a simpler and computationally faster algorithm that relies solely on Gibbs sampling. For both forms of ordinal quantile models, the **bqror** package also provides functions for calculating the covariate effects (for continuous as well as binary regressors) and measures for model comparison – marginal likelihood and the DIC. The paper explains

how to compute the marginal likelihood from the MCMC outputs and recommends its use over the DIC for model comparison. Additionally, this paper demonstrates the usage of functions for estimation and analysis of Bayesian quantile regression with ordinal data on simulation studies and two applications related to educational attainment and tax policy. In the future, the current package will be extended to include ordinal quantile regression with longitudinal data and variable selection in ordinal quantile regression with cross section and/or longitudinal data.

References

Albert, James, and Siddhartha Chib. 1993. "Bayesian Analysis of Binary and Polychotomous Response Data." *Journal of the American Statistical Association* 88 (422): 669–79. https://doi.org/10.1080/01621459.1993.10476321.

———. 2001. "Sequential Ordinal Modeling with Applications to Survival Data." *Biometrics* 57: 829–36. https://www.jstor.org/stable/3068422.

Alhamzawi, Rahim. 2016. "Bayesian Model Selection in Ordinal Quantile Regression." *Computational Statistics and Data Analysis* 103: 68–78. https://doi.org/10.1016/j.csda.2016.04.014.

Alhamzawi, Rahim, and Haithem Taha Mohammad Ali. 2018. "Bayesian Quantile Regression for Ordinal Longitudinal Data." *Journal of Applied Statistics* 45 (5): 815–28. https://doi.org/10.1080/02664763.2017.1315059.

Benoit, Dries F., and Dirk Van den Poel. 2010. "Binary Quantile Regression: A Bayesian Approach Based on the Asymmetric Laplace Distribution." *Journal of Applied Econometrics* 27 (7): 1174–88. https://doi.org/10.1002/jae.1216.

———. 2017. "BayesQR: A Bayesian Approach to Quantile Regression." *Journal of Statistical Software* 76 (7). https://cran.r-project.org/web/packages/bayesQR/index.html.

Bresson, Goerges, Guy Lacroix, and Mohammad Arshad Rahman. 2021. "Bayesian Panel Quantile Regression for Binary Outcomes with Correlated Random Effects: An Application on Crime Recidivism in Canada." *Empirical Economics* 60 (1): 227–59. https://doi.org/10.1007/s00181-020-01893-5.

Casella, George, and Edward I George. 1992. "Explaining the Gibbs Sampler." *The American Statistician* 46 (3): 167–74. https://doi.org/10.1080/00031305.1992.10475878.

Chib, S., and E. Greenberg. 1995. "Understanding the Metropolis-Hastings Algorithm." *The American Statistician* 49: 327–35. https://doi.org/10.1080/00031305.1995.10476177.

Chib, Siddhartha. 1995. "Marginal Likelihood from the Gibbs Output." *Journal of the American Statistical Association* 90 (432): 1313–21. https://doi.org/10.1080/01621459.1995.10476635.

———. 2012. "Introduction to Simulation and MCMC Methods." In *The Oxford Handbook of Bayesian Econometrics*, edited by John Geweke, Gary Koop, and Herman Van Dijk, 183–218. Oxford University Press, Oxford. https://doi.org/10.1093/oxfordhb/9780199559084.013.0006.

Chib, Siddhartha, and Ivan Jeliazkov. 2001. "Marginal Likelihood from the Metropolis-Hastings Output." *Journal of the American Statistical Association* 96 (453): 270–81. https://doi.org/10.1198/016214501750332848.

———. 2006. "Inference in Semiparametric Dynamic Models for Binary Longitudinal Data." *Journal of the American Statistical Association* 101 (474): 685–700. https://doi.org/10.1198/016214505000000871.

Dagpunar, John. 2007. *Simulations and Monte Carlo: With Applications in Finance and MCMC*. John Wiley & Sons Ltd., UK. https://onlinelibrary.wiley.com/doi/book/10.1002/9780470061336.

Davino, Cristino, Marileno Furno, and Domenico Vistocco. 2014. *Quantile Regression: Theory and Applications*. John Wiley & Sons, Chichester. https://doi.org/10.1002/9781118752685.

Devroye, Luc. 2014. "Random Variate Generation for the Generalized Inverse Gaussian Distribution." *Statistics and Computing* 24 (2): 239–46. https://doi.org/10.1007/s11222-012-9367-z.

Furno, Marilena, and Domenico Vistocco. 2018. *Quantile Regression: Estimation and Simulation*. John Wiley & Sons, New Jersey. https://doi.org/10.1002/9781118863718.

Gelman, Andrew, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. 2013. *Bayesian Data Analysis*. Chapman & Hall, New York. https://doi.org/10.1201/b16018.

Ghasemzadeh, Siamak, Mojtaba Ganjali, and Taban Baghfalaki. 2018. "Bayesian Quantile Regression for Analyzing Ordinal Longitudinal Responses in the Presence of Non-Ignorable Missingness." *METRON* 76 (3): 321–48. https://doi.org/10.1007/s40300-018-0136-4.

———. 2020. "Bayesian Quantile Regression for Joint Modeling of Longitudinal Mixed Ordinal Continuous Data." *Communications in Statistics − Simulation and Computation* 49 (2): 1375–95. https://doi.org/10.1080/03610918.2018.1484482.

Greenberg, Edward. 2012. *Introduction to Bayesian Econometrics*. 2nd Edition, Cambridge University Press, New York. https://doi.org/10.1017/CBO9780511808920.

Greene, William H., and David A. Hensher. 2010. *Modeling Ordered Choices: A Primer*. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511845062.

Jeliazkov, Ivan, Jennifer Graves, and Mark Kutzbach. 2008. "Fitting and Comparison of Models

for Multivariate Ordinal Outcomes." *Advances in Econometrics: Bayesian Econometrics* 23: 115–56. https://doi.org/10.1016/S0731-9053(08)23004-5.

Jeliazkov, Ivan, and Mohammad Arshad Rahman. 2012. "Binary and Ordinal Data Analysis in Economics: Modeling and Estimation." In *Mathematical Modeling with Multidisciplinary Applications*, edited by Xin She Yang, 123–50. John Wiley & Sons Inc., New Jersey. https://doi.org/10.1002/9781118462706.ch6.

Johnson, Valen E., and James H. Albert. 2000. *Ordinal Data Modeling*. Springer, New York. https://doi.org/10.1007/b98832.

Koenker, Roger. 2005. *Quantile Regression*. Cambridge University Press, Cambridge. https://doi.org/10.1257/jep.15.4.143.

Koenker, Roger, and G. W. Bassett. 1978. "Regression Quantiles." *Econometrica* 46 (1): 33–50. https://doi.org/10.2307/1913643.

Kordas, Gregory. 2006. "Smoothed Binary Regression Quantiles." *Journal of Applied Econometrics* 21 (3): 387–407. https://doi.org/10.1002/jae.843.

Kozumi, Hideo, and Genya Kobayashi. 2011. "Gibbs Sampling Methods for Bayesian Quantile Regression." *Journal of Statistical Computation and Simulation* 81 (11): 1565–78. https://doi.org/10.1080/00949655.2010.496117.

Mukherjee, Deep, and Mohammad Arshad Rahman. 2016. "To Drill or Not to Drill? An Econometric Analysis of US Public Opinion." *Energy Policy* 91: 341–51. https://doi.org/10.1016/j.enpol.2015.11.023.

Poirier, Dale J. 1995. *Intermediate Statistics and Econometrics*. The MIT Press, Cambridge. https://mitpress.mit.edu/9780262660945/intermediate-statistics-and-econometrics/.

Rahman, Mohammad Arshad. 2016. "Bayesian Quantile Regression for Ordinal Models." *Bayesian Analysis* 11 (1): 1–24. https://doi.org/10.1214/15-BA939.

Rahman, Mohammad Arshad, and Shubham Karnawat. 2019. "Flexible Bayesian Quantile Regression in Ordinal Models." *Advances in Econometrics* 40B: 211–51. https://doi.org/10.1108/S0731-90532019000040B011.

Rahman, Mohammad Arshad, and Angela Vossmeyer. 2019. "Estimation and Applications of Quantile Regression for Binary Longitudinal Data." *Advances in Econometrics* 40B: 157–91. https://doi.org/10.1108/S0731-90532019000040B009.

Spiegelhalter, David J., Nicola G. Best, Bradley P. Carlin, and Angelika Van Der Linde. 2002. "Bayesian Measures of Model Complexity and Fit." *Journal of the Royal Statistical Society – Series B* 64 (4): 583–639. https://doi.org/10.1111/1467-9868.00353.

Tian, Yu-Zhu, Man-Lai Tang, Wai-Sum Chan, and Mao-Zai Tian. 2021. "Bayesian Bridge-Randomized Penalized Quantile Regression for for Ordinal Longitudinal Data, with Application to Firm's Bond Ratings." *Computational Statistics* 36: 1289–319. https://doi.org/10.1007/s00180-020-01037-4.

Yu, Keming, and Rana A. Moyeed. 2001. "Bayesian Quantile Regression." *Statistics and Probability Letters* 54 (4): 437–47. https://doi.org/10.1016/S0167-7152(01)00124-9.

Prajual Maheshwari
Quantitative Researcher, Ogha Research
2123, 14th Main Road, HAL 3rd Stage, Kodihalli, Bengaluru, Karnataka, India
https://prajual.netlify.app
prajual1391@gmail.com

Mohammad Arshad Rahman
Department of Economic Sciences, Indian Institute of Technology Kanpur
Room 672, Faculty Building, Indian Institute of Technology Kanpur, India
https://www.arshadrahman.com
ORCiD: 0000-0001-8434-0042
marshad@iitk.ac.in, arshadrahman25@gmail.com

Gaussian Mixture Models in R

by Bastien Chassagnol, Antoine Bichat, Cheïma Boudjeniba, Pierre-Henri Wuillemin, Mickaël Guedj, David Gohel, Gregory Nuel, and Etienne Becht

Abstract Gaussian mixture models (GMMs) are widely used for modelling stochastic problems. Indeed, a wide diversity of packages have been developed in R. However, no recent review describing the main features offered by these packages and comparing their performances has been performed. In this article, we first introduce GMMs and the EM algorithm used to retrieve the parameters of the model and analyse the main features implemented among seven of the most widely used R packages. We then empirically compare their statistical and computational performances in relation with the choice of the initialisation algorithm and the complexity of the mixture. We demonstrate that the best estimation with well-separated components or with a small number of components with distinguishable modes is obtained with REBMIX initialisation, implemented in the rebmix package, while the best estimation with highly overlapping components is obtained with *k*-means or random initialisation. Importantly, we show that implementation details in the EM algorithm yield differences in the parameters' estimation. Especially, packages mixtools (Young et al. 2020) and Rmixmod (Langrognet et al. 2021) estimate the parameters of the mixture with smaller bias, while the RMSE and variability of the estimates is smaller with packages bgmm (Ewa Szczurek 2021) , EMCluster (W.-C. Chen and Maitra 2022) , GMKMcharlie (Liu 2021), flexmix (Gruen and Leisch 2022) and mclust (Fraley, Raftery, and Scrucca 2022). The comparison of these packages provides R users with useful recommendations for improving the computational and statistical performance of their clustering and for identifying common deficiencies. Additionally, we propose several improvements in the development of a future, unified mixture model package.

1 Introduction to Mixture modelling

Formally, let's consider a pair of random variables (X, S) with $S \in \{1, \ldots, k\}$ a discrete variable and designing the component identity of each observation. When observed, S is generally denoted as the labels of the individual observations. k is the number of mixture *components*. Then, the density distribution of X is given in Equation (1):

$$
\begin{aligned}
f_\theta(X) &= \sum_S f_\theta(X, S) \\
&= \sum_{j=1}^{k} p_j f_{\zeta j}(X), \quad X \in \mathbb{R}
\end{aligned}
\tag{1}
$$

where $\theta = (p, \zeta) = (p_1, \ldots, p_k, \zeta_1, \ldots, \zeta_k)$ denotes the parameters of the model: p_j is the proportion of component j and ζ_j represents the parameters of the density distribution followed by component j. In addition, since S is a categorical variable parametrized by p, the prior weights must enforce the unit simplex constraint (Equation (2)):

$$
\begin{cases}
p_j \geq 0 & \forall j \in \{1, \ldots, k\} \\
\sum_{j=1}^{k} p_j = 1
\end{cases}
\tag{2}
$$

In terms of applications, mixture models can be used to achieve the following goals:

- *Clustering*: hard clustering consists in determining a complete partition of the n observations $x_{1:n}$ into k disjoint non-empty subsets. In the context of *mixture model-based clustering*, this is done by assigning each observation i to the cluster $\hat{s}_i = \arg\max_j \eta_i(j)$ that maximises the posterior distribution (MAP) (see Equation (3)):

$$
\eta_i(j) := \mathbb{P}_\theta(S_i = j | X_i = x_i)
\tag{3}
$$

- *Prediction*: the purpose is to predict a response variable Y from an explanatory variable X. The dependent variable Y can either be discrete, taking values in classes $\{1, \ldots, G\}$ (*classification* task) or continuous (*regression* task). In this paper, we do not extensively discuss application of mixture models to regression purposes but refer the reader to Bouveyron and Girard (2009) for mixture classification and Shimizu and Kaneko (2020) for mixtures of regression models.

In section Univariate and multivariate Gaussian distributions in the context of mixture models, we describe the most commonly used family, the Gaussian Mixture Model (GMM). We then present the MLE estimation of the parameters of a GMM, introducing the classic EM algorithm in section Parameter estimation in finite mixtures models. Finally, we introduce bootstrap methods used to evaluate the quality of the estimation and metrics used for the selection of the best model in respectively appendices *Derivation of confidence intervals in GMMs* and *Model selection*.

1.1 Univariate and multivariate Gaussian distributions in the context of mixture models

We focus our study on the finite Gaussian mixture models (GMM) in which we suppose that each of the k components follows a Gaussian distribution.

We recall below the definition of the Gaussian distribution in both univariate and multivariate context. In the finite univariate Gaussian mixture model, the distribution of each component $f_{\zeta j}(X)$ is given by the following univariate Gaussian p.d.f. (probability density function) (Equation (4)):

$$f_{\zeta j}(X = x) = \varphi_{\zeta j}(x|\mu_j, \sigma_j) := \frac{1}{\sqrt{2\pi}\sigma_j} \exp^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}} \tag{4}$$

which we note: $X \sim \mathcal{N}(\mu_j, \sigma_j)$.

In the univariate case, the parameters to be inferred from each component, ζ_j, are: μ_j, the *location* parameter (equal to the mean of the distribution) and σ_j, the *scale* parameter (equal to the standard deviation of the distribution with a Gaussian distribution).

Following parsimonious parametrisations with respect to univariate GMMs are often considered:

- *homoscedascity*: variance is considered equal for all components, $\sigma_j = \sigma, \forall j \in \{1, \dots, k\}$, as opposed to heteroscedascity where each sub-population has its unique variability.

- *equi-proportion* among all mixtures: $p_j = \frac{1}{k} \quad j \in \{1, \dots, k\}$ [1]

In the finite multivariate Gaussian mixture model, the distribution $f_{\zeta j}(\boldsymbol{X})$ of each component j, where $\boldsymbol{X} \in \mathbb{R}^D = (X_1, \dots, X_D)^\top$ is a multivariate random variable of dimension D, is given by the following multivariate Gaussian p.d.f. (Equation (5)):

$$f_{\zeta j}(\boldsymbol{X} = \boldsymbol{x}) = \det(2\pi\boldsymbol{\Sigma}_j)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_j)\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_j)^\top\right) \tag{5}$$

which we note $\boldsymbol{X} \sim \mathcal{N}_D(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$. The parameters to be estimated for each component can be decomposed into:

- $\boldsymbol{\mu}_j = \begin{pmatrix} \mu_{1j} \\ \vdots \\ \mu_{Dj} \end{pmatrix} \in \mathbb{R}^D$, the D-dimensional mean vector.

- $\boldsymbol{\Sigma}_j$, the $\mathcal{M}_D(\mathbb{R})$ positive-definite [2] covariance matrix, whose diagonal terms are the individual variances of each feature and the off-diagonal terms are the pairwise covariance terms.

Three families of multivariate GMMs are often considered:

- the *spherical* family, $\boldsymbol{\Sigma}_j = \sigma_j^2 \boldsymbol{I}_D$, with $\sigma_j \in \mathbb{R}_+^*$, refers to GMMs whose covariance matrix is diagonal with an unique standard deviation term. The corresponding volume representation is a $D-$hypersphere of radius σ_j.

---

[1] A rarer constraint considered implies to enforce a linear constraint over the clusters' means, of the following general form: $\sum_{j=1}^{k} a_j\mu_j = 0$, with $\{a_1, \dots, a_k\}$. For instance, the R package **epigenomix** considers a $k = 3$ component mixture in the context of transcriptomic (differential analyses) and epigenetic (histone modification) to automatically identify undifferentiated, over and under-expressed genes between case and control samples. A common constraint then is to enforce the distribution of fold changes corresponding to the undifferentiated expressed genes to have a distribution centred on 0. Combining equality of means and equality of variances is irrelevant, as the model is then degenerate. Additionally, setting constraints on the means makes the estimation of the parameters challenging, as detailed in Appendix *Extensions of the EM algorithm to overcome its limitations*.

[2] The positive-definiteness constraint can be interpreted from a probabilistic point of view as a necessary condition such that the generalised integral of the multivariate distribution is defined and sum-to-one over $\mathbb{R}$ or from the statistical definition of the covariance. A symmetric real matrix X of rank D is said to be *positive-definite* if for any non-zero vector $\mathbf{v}, \in \mathbb{R}^D$, the following constraint $\mathbf{v}^\top X\mathbf{v} > 0$ is enforced.

- the *diagonal* family, $\Sigma_j = \text{diag}\left(\sigma_{1j}^2, \ldots, \sigma_{1D}^2\right)$, with $\sigma_j \in \mathbb{R}_+^D$, refers to GMMs whose covariance matrix is diagonal. Its associated volume representation is an ellipsoid whose main axes are aligned with the D canonical basis of $\mathbb{R}^D$. Of note, the null constraint imposed over the off-diagonal terms in the spherical and diagonal families imply that the multivariate distribution can be further decomposed and analysed as the product of univariate independent Gaussian realisations.

- the *ellipsoidal* family, also named the *general* family, refer to GMMs whose covariance matrix, Σ_j, can be any arbitrary positive-definite $D \times D$ matrix. Thus, the corresponding clusters for each component J are ellipsoidal, centred at the mean vector $\boldsymbol{\mu}_j$, and volume and orientation respectively determined by the eigenvalues and the eigenvectors of the covariance matrix Σ_j.

In the multivariate setting, the volume, shape, and orientation of the covariances can be constrained to be equal or variable across clusters, generating 14 possible parametrisations with different geometric characteristics (Banfield and Raftery 1993; Celeux and Govaert 1992). We review them in Appendix *Parameters estimation in a high-dimensional context* and Table 5. Of note, the correlation matrix can be easily derived from the covariance matrix with the following normalisation: $\text{cor}(\boldsymbol{X}) = \left(\frac{\text{cov}(x_l, x_m)}{\sqrt{\text{var}(x_l)} \times \sqrt{\text{var}(x_m)}}\right)_{(l,m) \in D \times D}$. Correlation if strictly included between -1 and 1, the strength of the correlation is given by its absolute value while the type of the interaction is returned by its sign. A correlation of 1 or -1 between two features indicates a strictly linear relationship.

For the sake of simplicity and tractability, we will only consider the fully unconstrained model in both the univariate (heteroscedastic and unbalanced classes) and multivariate dimension (unbalanced and complete covariance matrices for each cluster) in the remainder of our paper.

1.2 Parameter estimation in finite mixtures models

A common way for estimating the parameters of a parametric distribution is the *maximum likelihood estimation* (MLE) method. It consists in estimating the parameters by maximising the likelihood, or equivalently the log-likelihood of a sample. In what follows, $\ell(\theta|x_{1:n}) = \log(f(x_{1:n}|\theta))$ is the log-likelihood of a n-sample. When all observations are independent, it simplifies to $\ell(\theta|x_{1:n}) = \sum_{i=1}^n \log(f(x_i|\theta))$. The MLE consists in finding the parameter estimate $\hat{\theta}$ which maximises the log-likelihood $\hat{\theta} = \arg\max \ell(\theta|x_{1:n})$.

Recovering the maximum of a function is generally performed by finding the values at which its derivative vanishes. The MLE in GMMs has interesting properties, as opposed to the *moment estimation* method: it is a consistent, asymptotically efficient and unbiased estimator (Chen 2016; McLachlan and Peel 2000).

When S is completely observed, for pairs of observations $(x_{1:n}, s_{1:n})$, the log-likelihood of a finite mixture model is simply given by Equation (6):

$$\ell(\theta|X_{1:n} = x_{1:n}, S_{1:n} = s_{1:n}) = \sum_{i=1}^n \sum_{j=1}^k \left[\log\left(f_{\zeta_j}(x_i, s_i = j)\right) + \log(p_j)\right]_{\mathbf{1}_{s_i=j}} \tag{6}$$

where an analytical solution can be computed provided that a closed-form estimate exists to retrieve the parameters ζ_j for each components' parametric distribution. The MLE maximisation, in this context, involves the estimation of the parameters for each cluster, denoted as ζ_j. The corresponding proportions, p_j, can be straightforwardly computed as the ratios of observations assigned to cluster j relative to the total number of observations, n.

However, when S is unobserved, the log-likelihood, qualified as incomplete with respect to the previous case, is given by Equation (7):

$$\ell(\theta|x_{1:n}) = \sum_{i=1}^n \log\left(\underbrace{\sum_{j=1}^k p_j f_{\zeta_j}(x_i)}_{\text{sum of of logs}}\right) \tag{7}$$

The sum of terms embed in the log function (see underbrace section in Equation (7)) makes it intractable in practice to derive the null values of its corresponding derivative. Thus, no closed form of the MLE is available, including for the basic univariate GMM model. This is why most parameter estimation methods derive instead from the *EM algorithm*, first described in Dempster, Laird, and Rubin (1977). We describe its main theoretical properties, the reasons for its popularity, and its main limitations in the next section.

1.3 The EM algorithm

In cases where both S and the parameters associated to each cluster are unknown, there is no available closed-form solution that would jointly maximise the log-likelihood, as defined in Equation (7), with respect to the set of parameters (θ, S). However, when either S or θ are known, the estimation of the other parameters is straightforward. Hence, the general principle of EM-like algorithms is splitting this complex non-closed joint MLE estimation of (S, θ) into the iterative estimation of S_q from $\hat{\theta}_{q-1}$ and X (expectation phase, or *E-step* of the algorithm) and the estimation of $\hat{\theta}_q$ from (S_q and X (maximisation phase, or *M-step*), with $\hat{\theta}_{q-1}$ being the estimated parameters at the previous step $q - 1$, until we reach the convergence.

The EM algorithm sets itself apart from other commonly used methods by taking into account all possible values taken by the latent variable S. To do so, it computes the expected value of the log likelihood of θ, conditioned by the posterior distribution $\mathbb{P}_{\hat{\theta}_{q-1}}(S|X)$, also named as the *auxiliary function*. Utilising the assumption of independence among observations in a mixture model, the general formula of this proxy function of the incomplete log-likelihood is given in finite mixture models by Equation (8).

$$
\begin{aligned}
Q(\theta|\hat{\theta}_{q-1}) &:= \mathbb{E}_{S_{1:n}|X_{1:n},\hat{\theta}_{q-1}} \left[\ell(\theta|X_{1:n}, S_{1:n}) \right] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{k} \eta_i(j) \left(\log(p_j) + \log(\mathbb{P}(X_i|S_i = j, \theta)) \right)
\end{aligned}
\tag{8}
$$

with $\hat{\theta}_{q-1} = \hat{\theta}$ the current estimated parameter value.

In practice, the EM algorithm consists in performing alternatively E-step and M-step until convergence, as described in the pseudocode below (Box 1):

Box 1: the EM algorithm

- *step E*: determine the posterior probability function $\eta_i(j)$ for each observation of X for each possible discrete latent class, using the initial estimates $\hat{\theta}_0$ at step $q = 0$, or the previously computed estimates $\hat{\theta}_{q-1}$. The general formula is given by Equation (9):

$$
\eta_i(j) = \frac{p_j f_{\zeta_j}(x_i)}{\sum_{j=1}^{k} p_j f_{\zeta_j}(x_i)}
\tag{9}
$$

- *step M*: compute the mapping function $\hat{\theta}_q := M(\theta|\hat{\theta}_{q-1}) = \arg\max Q(\theta|\hat{\theta}_{q-1})$ which maximises the auxiliary function. One way of retrieving the MLE associated to the auxiliary function is to determine the roots of its derivative, namely solving Equation (10)[a]:

$$
\frac{\partial Q(\theta|\hat{\theta}_{q-1})}{\partial \theta} = 0
\tag{10}
$$

---

[a]To ensure that we reach a maximum, we should assert that the Hessian matrix evaluated at the MLE is indeed negative definite.

Interestingly, the decomposition of the incomplete log-likelihood associated to a mixture model $\ell(\theta|X)$ reveals an entropy term and the so-called auxiliary function (Dempster, Laird, and Rubin 1977). It can be used to prove that maximising the auxiliary function at each step induces a bounded increase of the incomplete log-likelihood. Namely, the convergence of the EM algorithm, defined by comparisons of consecutive log-likelihood, is guaranteed, provided the mapping function returns the maximum of the auxiliary function. Yet, the convergence of the series of estimated parameters $(\theta_q)_{q\geq 0} \xrightarrow[i\to+\infty]{} \hat{\theta}$ is harder to prove but has been formally demonstrated for the *exponential family* (a superset of the Gaussian family), as stated in Dempster, Laird, and Rubin (1977).

Additionally, the EM algorithm is *deterministic*, meaning that for a given initial estimate θ_0 the parameters returned by the algorithm at a given step q are fixed. However, this method requires the user to provide an initial estimate, denoted as θ_0, of the model parameters and to specify the number of components in the mixture. We review some classic initialisation methods in Initialisation of the EM algorithm and some algorithms used to overcome the main limitations of the EM algorithm in the Appendix *Extensions of the EM algorithm to overcome its limitations*.

Finally, the prevalent choice of Gaussian distributions to characterize the distribution of random observations is guided by a set of interesting properties. In particular, Geary (1936) has shown that the Normal distribution is the only distribution for which the Cochran's theorem (Cochran 1934) is guaranteed, namely for which the the mean and variance of the sample are independent of each other. Additionally, similar to any distribution proceeding from the exponential family, the MLE statistic is *sufficient*[3].

1.4 Initialisation of the EM algorithm

EM-like algorithms require an initial estimate of the parameters, θ_0, to optimise the maximum likelihood. *Initialisation* is a crucial step, as a bad initialisation can possibly lead to a local sub-optimal solution or trap the algorithm in the boundary of the parameter space. The most straightforward initialisation methods, such as random initialisation, are standalone and do not require any additional initialisation algorithms, whereas *meta-methods*, such as short-EM, still need to be initialised by alternative methods. The commonly-used initialisation methods encompass:

- The *Model-based Hierarchical Agglomerative Clustering* (MBHC) is an agglomerative hierarchical clustering based on MLE criteria applied to GMMs (Scrucca and Raftery 2015). First, the MBHC is initialised by assigning each observation to its own cluster. Next, the pair of clusters that maximises the likelihood of the underlying statistical model among all possible pairs is merged. This procedure is repeated until all clusters are merged. The final resulting clusters are then simply the last k cuts of the resulting dendrogram. When the data is univariate and homoscedastic, or when the underlying distribution has a diagonal covariance matrix, the merging criterion performs similarly to *Ward's criterion*, in that merging of the two clusters also simultaneously minimizes the sum of squares. As opposed to the other initialisation methods described hereafter, MBHC is a deterministic method which does not require careful calibration of hyperparameters. However, as acknowledged by the author of the method (Fraley 1998), the resulting partitions are generally suboptimal compared to other initialisation methods.

- The conventional *random* initialization method, frequently employed for the initialization step of the k-means algorithm, involves the random selection of k distinct observations, which are referred to as *centroids*. Subsequently, each observation is assigned to the nearest centroid, a process reminiscent of the C-step in the CEM algorithm (Biernacki, Celeux, and Govaert 2003). This is the method used in this paper, unless otherwise stated. Alternative versions of this method have been developed: for instance, the package **mixtools** draws the proportions of the components from a Dirichlet distribution, whose main advantage lies in respecting the unit simplex constraint (Equation (2))[4], but uses binning methods to guess the means and standard deviations of the components. Similarly, Kwedlo (2013) proposes a method in which the means of the components are randomly chosen, but with an additional constraint of maximising the Mahalanobis distance between the selected centroids. This enables to cover a larger portion of the parameters' space.

- k-means is a CEM algorithm, in which the additional assumption of balanced classes and homoscedascity implies that each observation in the E-step is assigned to the cluster with the nearest mean (the one with the shortest Euclidean distance). K-means is initialised by randomly selecting k points, known as the *centroids*. It is often chosen for its fast convergence and memory-saving consumption.

- The *quantile* method sorts each observation x_i in an increasing order and splits them into equi-balanced quantiles of size $1/k$. Then, all observations for a given quantile are assumed to belong to the same component. [5]

- The *Rough-Enhanced-Bayes mixture* (REBMIX) algorithm is implemented in the **rebmix** (Nagode 2022) package and the complete pseudo-code is described thoroughly in (Nagode 2015; Panic, Klemenc, and Nagode 2020). The key stages implemented by the rebmix algorithm for initialising the parameters of GMMs encompass:

---

[3]The Pitman–Koopman–Darmois theorem (Koopman 1936) states that only the exponential family provides distributions whose statistic can summarize arbitrary amounts of iid draws using a finite number of values

[4]Without prior knowledge favouring one component over another, the Dirichlet distribution is generally parametrised by $\alpha = \frac{1}{k}$, implicitly stating that any observation has equal chance to proceed from a given cluster. In that case, the corresponding distribution is parametrised by a single scalar value α, called the *concentration parameter*.

[5]This method is only available in the univariate framework, since it is not possible to define a unique partition of the observable space into k-splits. For example, in bivariate setting, a binning with $k = 2$ components on each axis leads to a total of $2 \times 2 = 4$ binned regions, which raises the selection issue of the best k hyper-squared volumes for the initial parameters estimation. More generally, $\binom{D}{k}$ binning choices are possible in the multivariate setting.

- First, the observations are processed using one of these three methods: k-nearest neighbours (KNN), Parzen kernel density estimation, or binned intervals. With the binned interval method, the observations are initially divided into $\sqrt{n}^D$ intervals of equal lengths. The mode of one of the components' distribution is subsequently determined by the midpoint of the interval with the highest frequency. The observations lying within the interval are used as preliminary estimates, referred to as "rough" parameters in Nagode (2015).
- All other observations and intervals are then iteratively assigned to the currently estimated component or to residual components, the ones that have not yet been characterised. The decision to assign an interval to either the currently estimated component or one of the residual components depends on the magnitude of the discrepancy between the observed and the expected frequency within the interval.
- Finally, all intervals assigned to the currently estimated component (and not only the interval including the mode of the distribution) are used to determine the parameters of the associated Gaussian distribution. Since this step relies on a more comprehensive number of observations for parameter estimation, guaranteeing in principle more robust estimates, this stage is referred to as "enhanced" estimation in Nagode (2015). The algorithm terminates when all intervals have been assigned to a cluster, and the parameters of the various distribution components have been estimated.

The rebmix algorithm can thus be seen as a natural extension of the quantile method, with more rigorous statistical support. Two drawbacks of the algorithm include the need for intensive calibration of hyperparameters and its inadequacy for the estimation of highly overlapping or high dimensional mixture distributions[6].

- The *meta-methods* consist generally in short runs of EM-like algorithms, namely CEM, SEM and EM (see Appendix B: *Extensions of the EM algorithm to overcome its limitation*), with alleviated convergence criterion. The main idea is to use several random initial estimates with shorter runs of the algorithm to explore larger regions of the parameter space and avoid being trapped in a local maximum. Yet, these methods are highly dependent on the choice of the initialisation algorithm (Biernacki, Celeux, and Govaert 2003).

- In the high-dimensional setting, if the number of dimensions D exceeds the number of observations n, all previous methods must be adjusted, usually by first projecting the dataset into a smaller, suitable subspace and then inferring prior parameters in it. In particular, **EM-MIXmfa**, in the mixture of common factor analysers (MCFA) approach, initialises the shared projection matrix Q by either keeping the first d eigen vectors generated from standard principal component analysis or uses custom random initialisations (Baek, McLachlan, and Flack 2010).

Following this theoretical introduction, we empirically evaluate the performance of the aforementioned R packages, considering various initialization algorithms and the complexity of the GMMs distributions. Precisely, we outline the simulation framework used to compare the seven packages in Methods and report the results in Results. We conclude by providing a general simplified framework to select the combination of package and initialisation method best suited to its objectives and the nature of the distribution of the dataset.

2 A comprehensive benchmark comparing estimation performance of GMMs

We searched CRAN and Bioconductor mirrors for packages that can retrieve parameters of GMM models. Briefly, out of 54 packages dealing with GMMs estimation, we focused on seven packages that all estimate the MLE in GMMs using the EM algorithm, were recently updated and allow the users to specify their own initial estimates: **bgmm**, **EMCluster**, **flexmix**, **GMKMcharlie**, **mclust**, **mixtools** and **Rmixmod**. The complete inclusion process is detailed in Appendix C, *the meta-analysis workflow for the final selection of CRAN and Bioconductor platforms*. The flowchart summarising our choices is represented in Figure 1.

---

[6]The method we describe here to preprocess the observations in order to estimate the empirical density estimation, namely the "histogram method" is not well suited for high dimensional data, as the exponential growth of the volume with respect to dimensionality leads to data sparsity, related to the well-known issue of the "curse of dimensionality". Indeed, $\sqrt{n}^D$ distinct intervals will be parsed by the method and the probability with an increasing number of features and decreasing number of observations that no clear local maximum emerges converges to 1. In high-dimensional context, the Parzen window or the KNN method should be favoured, see (Nagode 2015), p. 16.
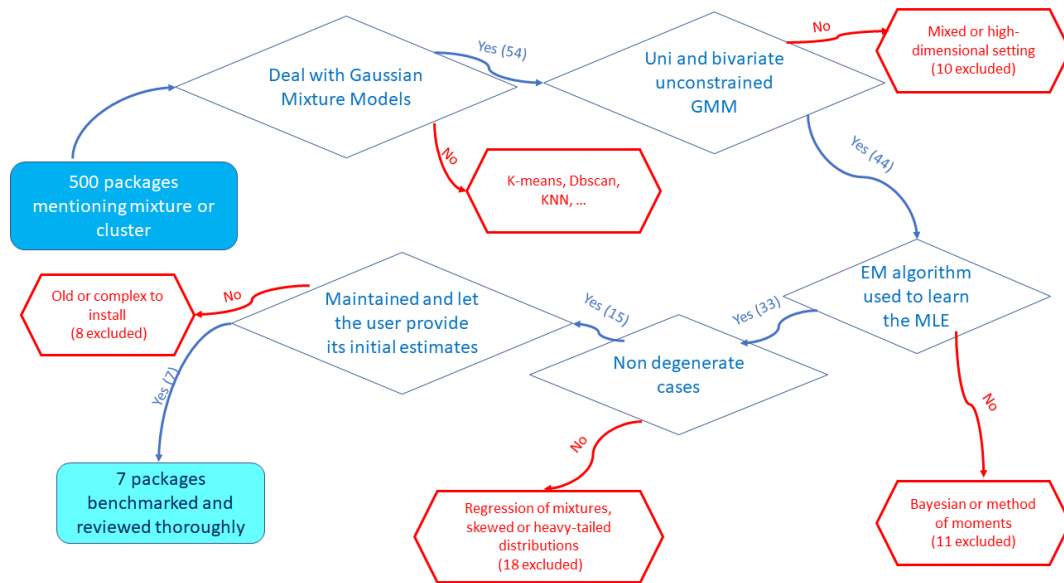
Figure 1: A minimal roadmap used for the selection of the packages reviewed in our benchmark.

We also include two additional packages dedicated specifically to high-dimensional settings, namely **EMMIXmfa** (Rathnayake et al. 2019) and **HDclassif** (Berge, Bouveyron, and Girard 2019) to compare their performance with standard multivariate approaches in complex, but non degenerate cases. We summarise the main features and use cases of the seven + two reviewed packages in Table 1. The three most commonly used packages are **mixtools**, **mclust** and **flexmix**. However, the **mclust** package is by far the most complete with many features provided to visualise and evaluate the quality of the GMM estimate. **bgmm** has the greatest number of dependencies, while **mclust** only depends of base R packages. Additionally, in parallel to clustering tasks, **flexmix** and **mixtools** packages perform regression of mixtures and implement mixture models using other parametric distributions or non-parametric methods via kernel-density estimation.

Table 1: Main features of the reviewed packages, sorted by decreasing number of daily downloads. *Downloads per day* returns the daily average number of downloads for each package on the last 2 years. *Recursive dependencies* column counts the complete set of non-base packages required, as first-order dependencies depend on other packages as well.

Package	Version	Regression	Implemented models	Downloads per day	Last update	Imports	Recursive dependencies	Language
mclust	5.4.7	☒	☒	5223	31/10/2022	R ($\geq$ 3.0)	0	Fortran
flexmix	2.3-17	☑	Poisson, binary, non-parametric, semi-parametric	3852	07/06/2022	R ($\geq$ 2.15.0), modeltools, nnet, stats4	3	R
mixtools	1.2.0	☑	multinomial, gamma, Weibull, non-parametric, semi-parametric	178	05/02/2022	R ($\geq$ 3.5.0), kernlab, segmented, survival	6	C
Rmixmod	2.1.5	☒	☒	39	18/10/2022	R ($\geq$ 2.12.0), Rcpp, RcppEigen	4	C++
EMCluster	0.2-13	☒	☒	33	12/08/2022	R ($\geq$ 3.0.1), Matrix	3	C
bgmm	1.8.4	☒	☒	27	10/10/2021	R ($\geq$ 2.0), mvtnorm, combinat	77	R
GMKMcharlie	1.1.1	☒	☒	12	29/05/2021	Rcpp, RcppParallel, RcppArmadillo	3	C++
EMMIXmfa	2.0.11	☒	☒	12	16/12/2019	NA	0	C
HDclassif	2.2.0	☒	☒	35	12/10/2022	rARPACK	13	R

We further detail features specifically related to GMMs in Table 2. We detail row after row its content below:

- The parametrisations used to provide parsimonious estimation of the GMMs are reviewed in Parameter estimation in finite mixtures models and summarised in rows 1 and 2 (Table 2) for the univariate and multivariate setting. We refer to the package as "canonical" when it implements both homoscedastic and heteroscedastic parametrisations in the univariate setting, and the 14 parametrisations listed in Supplementary Table 3 in the multivariate setting. Adding the additional constraint of equi-balanced clusters results in a total to $14 \times 2 = 28$ distinct models and $2 \times 2 = 4$ parametrisations, respectively in the univariate and multivariate setting. Since

EMMIXmfa and **HDclassif** are dedicated to the analysis of high-dimensional datasets, they project the observations in a smaller subspace and are not available in the univariate setting. Given an user-defined or prior computed intrinsic dimension, we can imagine using any of the standard parametrisations available for instance in the **mclust** package, and listed in Appendix *Parsimonious parametrisation of multivariate GMMs*. In addition, **HDclassif** allows each cluster j to be represented with its own subspace intrinsic dimension d_j, as we describe in further details in Appendix *Parameters estimation in a high-dimensional context*.

- The EM algorithm is the most commonly employed method for estimating the parameters of GMMs, however, alternative algorithms based on the EM framework, are reviewed in Appendix B: *Extensions of the EM algorithm to overcome its limitations* and row 3 of Table 2. Especially, GMMs estimation is particularly impacted by the presence of outliers, justifying a specific benchmark (see Appendix *A small simulation to evaluate the impact of outliers*). We briefly review the most common initialisation algorithms in section Initialisation of the EM algorithm and row 4 of Table 2, a necessary and tedious task for both the EM algorithm and its alternatives.

- To select the best parametrisations and number of components that fit the mixture, several metrics are provided by the reviewed packages (*Model selection* and row 5). Due to the complexity of computing the true distribution of the estimated parameters, bootstrap methods are commonly used used to derive confidence intervals (see Appendix *Derivation of confidence intervals in GMMs* and row 6 in Table 2).

- Six packages supply several functions for visualisation, summarised in the last row of Table 2, to display either the distributions corresponding to the estimated parameters or compare quickly the performance across packages. However, **mclust** is by far the most complete one, with density plots (in the univariate setting) and isodensity plots (bi-dimensional in the bivariate setting or in higher dimensions after appropriate dimensionality reduction), with the option to plot custom confidence intervals and critical regions, and finally boxplot bootstrap representations for displaying the distribution of the benchmarked estimated parameters.

High-dimensional packages provide specific representations adjusted to the high-dimensional settings, notably allowing the user to visualise the projected factorial representation of its dataset in a two or three-dimensional subspace. They also provide specialised performance plots, notably scree plots or BIC scatter plots to represent in a compact way numerous projections and parametrisations.

Table 2: Custom features associated to GMMs estimation for any of the benchmarked packages.

	mclust	flexmix	mixtools	Rmixmod	EMCluster	bgmm	GMKMcharlie	EMMIXmfa	HDclassif
Models Available (univariate)	canonical	unconstrained	canonical	canonical	unconstrained	canonical	unconstrained	NA	NA
Models Available (multivariate)	canonical	unconstrained diagonal or general	unconstrained	canonical	unconstrained	4 models (diagonal and general, either component specific or global)	unconstrained	4 models (either component-wise or common, on the intrinsic and diagonal residual error covariance matrices)	canonical on projected dimension
Variants of the EM algorithm	VBEM	SEM, CEM	ECM	SEM, CEM	☒	☒	CW-EM, MML	AECM	SEM, CEM
Initialisation	hierarchical clustering, quantile	short-EM, random	random	random, short-EM, CEM, SEM	random, short-EM	k-means, quantile	k-means	k-means, random, heuristic	short-EM, random, k-means
Model or Cluster Selection	BIC, ICL, LRTS	AIC, BIC, ICL	AIC, BIC, ICL, CAIC, LRTS	BIC, ICL, NEC	AIC, BIC, ICL, CLC	GIC	☒	☒	BIC, ICL, CV
Bootstrap Confidence Intervals	☑	☑	☑	☒	☒	☒	☒	☒	☒
Visualisation	performance, histograms and boxplots of bootstrapped estimates, density plots (univariate), scatter plots with uncertainty regions and boundaries (bivariate), isodensity (bivariate , 2D projected PCA or selecting coordinates)	☒	density curves	density curves, scatter plots with uncertainty boundaries	☒	performance, scatter plots with uncertainty boundaries	☒	projected factorial map	projected factorial map, performance (Cattell's scree plot, BIC performance, slope heuristic)

2.1 Methods

In addition to the the seven packages selected for our benchmark, we include a custom R implementation of the EM algorithm used as baseline, referred to as *RGMMBench*, and for the high-dimensional

setting we select packages **EMMIXmfa** and **HDclassif**, on the basis of criteria detailed in Appendix C, *General workflow*. Code for *RGMMBench* is provided in Appendix *Application of the EM algorithm to GMMs*. To compare the statistical performances of these packages, we performed *parametric bootstrap* (*Derivation of confidence intervals in GMMs*) and built an experimental design to cover distinct mixture distributions parameter configurations, using prior user-defined parameters.

For each experiment, we assign each observation to an unique cluster by drawing n labels $S_{1:n}$ from a multinomial distribution whose parameters were the prior user-defined proportions $p = (p_1, \ldots, p_k)$. Then, each observation x_i assigned to hidden component j is drawn from a Normal distribution using the `stats::rnorm()` function for the univariate distribution and `MASS::mvrnorm` for the multivariate distribution. The complete code used for simulating data is available on GitHub at RGMMBench. Finally, we obtain an empirical distribution of the estimated parameters by computing the MLE of each randomly generated sample.

For all the packages, we used the same convergence threshold, 10^{-6}, and maximum of 1,000 iterations, as a numerical criterion for convergence. We generated simulated data with $n = 200$ observations in the univariate setting and $n = 500$ observations in the bivariate setting. We set the number of observations in order to minimise the probability of generating a sample without drawing any observations from one of the components[7]. Unless stated explicitly, we kept the default hyper-parameters and custom global options provided by each package. For instance, the **flexmix** package has a default option, *minprior*, set by default to 0.05, which removes any component present in the mixture with a ratio below 0.05. Besides, the fully unconstrained model was the only one which we implemented both in the univariate and multivariate settings, as it is the only parametrisation implemented in all the seven packages.

We compared the packages' performances using five initialisation methods: random, quantile, *k*-means, rebmix and hierarchical clustering in the univariate setting. We benchmarked the same initialisation methods in the multivariate setting, except for the quantile method which has no multivariate equivalent (see section Initialisation of the EM algorithm):

- We used the function `EMCluster::rand.EM()` with 10 random restarts and minimal cluster size of 2 for the random initialisation. The method implemented by **EMCluster** is the most commonly used, described in details in Biernacki, Celeux, and Govaert (2003) and in section Initialisation of the EM algorithm.

- To implement the k-means initialisation, we used the `stats::kmeans()` function with a convergence criterion fo 10^{-2} and maximum of 200 iterations. The initial centroid and covariance matrix for each component were computed by restricting to the sample observations assigned to the corresponding component. The approach is close to the one adopted by the CEM algorithm (see Appendix B: *Extensions of the EM algorithm to overcome its limitations*).

- We used the `mclust::hcV()` function for the MBHC algorithm. This method has two main limitations: just like the *k*-means implementation, it only returns a cluster assignment to each observation instead of the posterior probabilities, and the splitting process to generate the clusters sometimes results in clusters composed of only one observation. To avoid this, we added a small epsilon to each posterior probability.

- We used in the univariate setting `bgmm::init.model.params` for the quantiles initialisation.

- To implement the rebmix method, we used the `rebmix::REBMIX` function, using the *kernel density estimation* for the estimation of the empirical density distribution coupled with *EMcontrol* set to one to prevent the algorithm from starting EM iterations.

- Any of the seven packages could be used to implement the small EM method. We decided to use the `mixtools::normalmixEM` as it is the closest one to our custom implementation. We specified 10 random restarts, a maximal number of iterations of 200 and an alleviated absolute threshold of 10^{-2}. Preliminary experiments have led us to consider the removal of the small EM initialization method from the simulation benchmark. This decision is based on the observation that the differences of performance observed between the packages were no longer significant (see supplementary Figure 9).

We sum up in Table 3 the general configuration used to run the scripts. Additionally, all simulations were run with the same R (R Core Team 2023) version 4.0.2 (2020-06-22).

Preliminary experiments suggested that the quality of the estimation of a GMM is mostly affected by the overlap between components' distribution and level of unbalance between components. We quantified the overlap between two components by the following overlap score (OVL, see Equation (11)), with a smaller score denoting well-separated components:

---

[7]It is especially critical in cases of highly unbalanced configurations, as detailed in Appendix *Practical details for the implementation of our benchmark*

Table 3: Global options shared by all the benchmarked packages.

Initialisation methods	Algorithms	Criterion threshold	Maximal iterations	Number of observations
midrule hc, kmeans, small EM,rebmix, quantiles, random	EM R, Rmixmod, bgmm, mclust, flexmix, EMCluster, mixtools, GMKMCharlie	10^{-6}	1000	100, 200, 500, 1000, 2000, 5000, 10000

$$\text{OVL}(i,j) = \int \min(f_{\zeta_i}(x), f_{\zeta_j}(x))dx \quad \text{with } i \neq j \tag{11}$$

We may generalise this definition to k components by averaging the individual components' overlap. We use the function `MixSim::overlap` from the **MixSim** package (Melnykov, Chen, and Maitra 2021) that approximates this quantity using a Monte-Carlo based method (see appendices *An analytic formula of the overlap for univariate Gaussian mixtures* and *Practical details for the implementation of our benchmark* for further details).

The level of imbalance may be evaluated with entropy measure (Equation (12)):

$$H(S) = -\sum_{j=1}^{k} p_j \log_k(p_j) \tag{12}$$

with k is the number of components and $p_j = \mathbb{P}(S = j)$ is the frequency of class j.

We considered 9 distinct configuration parameters, associated with distinct values of OVL and entropy in the univariate setting, 20 configurations in the bivariate setting, and 16 configurations in the high-dimensional setting. Briefly, in the univariate setting, we simulated components with the same set of four means (0, 4, 8, and 12), three sets of mixture proportions $[(0.25, 0.25, 0.25, 0.25); (0.2, 0.4, 0.2, 0.2); (0.1, 0.7, 0.1, 0.1)]$ and three variances: $(0.3, 1, 2)$. In the bivariate setting, we consider two sets of proportions: $[(0.5, 0.5); (0.9, 0.1)]$, two sets of coordinate centroids: $[(0; 20), (20, 0)]$ and $[(0; 2), (2, 0)]$, the same variance of 1 for each feature and for each component for illustrative purposes of the direct relation linking the correlation and the level of OVL and five sets of correlation:$[(-0.8, -0.8); (0.8, -0.8); (-0.8, 0.8); (0.8, 0.8); (0, 0)]$.

Finally, we tested eight configurations in the high-dimensional framework, setting to $D = 10$ the number of dimensions. We modified the level of overlap (definition is reported in Equation (11)) and the imbalance between the component proportions across our simulations. Additionally, we tested two types of constraints on the covariance matrix: fully parametrised and spherical (see Appendix *Parsimonious parametrisation of multivariate GMMs*). Each of the parameter configurations tested in the high-dimensional setting was evaluated with $n = 200$ observations and $n = 2000$ observations. Additionally, instead of manually defining the parameters for the high-dimensional simulation, we used the `MixSim` function from the **MixSim** package (Melnykov, Chen, and Maitra 2021). This function returns the user a fully parametrised GMM, with a prior defined level of maximum or average overlap[8].

The complete list of parameters used is reported respectively in Table 4 for the univariate setting, Table 5 for the bivariate setting and 6 for the high-dimensional setting. We benchmarked simulations where the components were alternatively very distinct or instead very overlapping, as well as of equal proportions or instead very unbalanced. The adjustments made to meet the specific requirements of high dimensional packages are detailed in *Practical details for the implementation of our benchmark*.

We report the most significant results and features and the associated recommendations in next section Results.

2.2 Results

All figures and performance overview tables are reported in *Supplementary Figures and Tables in the univariate simulation* for the univariate setting, *Supplementary Figures and Tables in the bivariate simulation* for the bivariate scenario and *Supplementary Figures and Tables in the HD simulation* for the high dimensional scenario.

Balanced and non-overlapping components

---

[8]Unfortunately, as discussed in further details in Appendix *An analytic formula of the overlap for univariate Gaussian mixtures*, the MixSim package does not compute the global distribution overlap, but instead returns the mean of pairwise overlap between any component (however, with two components, these two alternative definitions match.) Finally, it is not possible to set the proportions of the components before the generation of the parameters, except for clusters with equal proportions, and contrary to the expect behaviour of additional parameter *PiLow*, supposed to define the smallest mixing proportion.

In the univariate setting, with balanced components and low OVL (scenario U1 in Table 4), the parameter estimates are identical in most cases across initialisation methods and packages, notably the same estimates are returned with *k*-means or rebmix initialisation. However, the random initialisation method leads to a higher variance and bias on the parameter estimates than other methods (Supplementary Figure 4 and Supplementary Table 6), with some estimates fitting only local maxima, far from the optimal value.

Similarly, the scenarios in the bivariate setting (configurations B6-B10 in Table 5), with a focus on B6, B7 and B10 visualised in Supplementary Figure 16, feature well-separated and balanced components. Consistent with conclusions from the corresponding univariate configurations, all benchmarked packages return the same estimates across initialisation methods.

Unbalanced and non-overlapping components

However, with unbalanced classes and low OVL (scenario U7 in 4), the choice of the initialisation method is crucial, with quantiles and random methods yielding more biased estimates and proned to fall in local maximum. Rebmix initialisation provides the best estimates, with the smallest MSE and bias across packages (Supplementary Figure 5 and supplementary Table 7, always associated with the highest likelihood. Overall, with well-discriminated components, most of the differences on the estimation originate from the choice of initialisation method, while the choice of the package has only small impact.

In the bivariate framework, two configurations featured both strongly unbalanced and well-separated components, similarly to scenario U3 in Table 4: the configurations B12 (Supplementary Figure 12 and Table 12) and B14 (Supplementary Figure 13 and Supplementary Table 13). Similarly, configurations B16, B17 and B20 (Table 5) with similar characteristics are summarised in supplementary Figure 17. In all these configurations, neither the initialisation method nor the package have a statistical significant impact on the overall performance.

Similarly, configurations HD1a-HD4b in Table 6) in the high dimensional setting display well-separated clusters, with a representative outcome represented in Supplementary Figure 19 and Supplementary Table 16. Consistent with the results obtained in the analogous univariate and bivariate scenarios, in the unbalanced and non-overlapping framework, the majority of the benchmarked packages produce highly consistent and similar estimates when hierarchical clustering and k-means were used for parameter initialisation. However, **bgmm** and **EMCluster** clearly perform worse when the rebmix initialization method is used (however, overall, rebmix performs poorly, regardless of the package used for estimation). Notably, initialisations with the rebmix package tend to display a much larger number of poor estimations, some of which can be identified with the local maxima associated with parameter switching between the two classes. Finally, the two additional packages dedicated to high-dimensional clustering display the worst performances, with **EMMIXmfa** returning the most biased parameters and **HDclassif** the most noisy estimates. **EMMIXmfa** is the only package that returned highly biased estimates of the components' proportions in this setting.

Balanced and overlapping components

When the overlap between components increases, the bias and variability of the estimates tends to increase, and the choice of initialisation method becomes more impactful. The least biased and noisy estimations with balanced components in the univariate setting (scenario U3 in Table 4) are obtained with the *k*-means initialisation (Supplementary Figure 3 and Table 8) while the rebmix initialisation returns the most biased and noisy estimates. Similar results are found in the bivariate setting with a balanced and highly overlapping two-component GMM (configurations B1-B5 from Table 5), with the best performance reached with the *k*-means initialisation method, followed by MBHC. This is emphasised in supplementary Figure 16, in the top three most complex configurations, namely B1, B2 and B5. If the shape of the covariance matrix is well-recovered, no matter the package, the Hellinger distances are significantly higher (and thus the estimate further away from the target distribution) with the random and rebmix methods.

Similarly, in the high-dimensional scenario HD7 of Table 6), presenting balanced but highly overlapping clusters with a full covariance structure, the best performance was obtained with *k*-means initialisation, while the rebmix initialisation returned the most biased and noisy estimates. While **EMMIXmfa** performed well when it converged, it returned an error in most cases (see Column *Success* of supplementary Table 17). The least biased estimates were returned by **mixtools** and **Rmixmod** and the least noisy by **flexmix**, **mclust** and **GMKMCharlie** (smaller MSE). Interestingly, in the high-dimensional setting, the packages **EMCluster** and **bgmm** exhibited worse performance. In particular, as can be seen in panel E of supplementary Figure 20, the proportions of the components recovered the $]0-1[^k$ simplex.

Conversely, the **EMCluster** package, and to a lesser extent, the **bgmm** package, performed surprisingly well when datasets were simulated with an underlying spherical covariance structure, even though the estimation was not performed explicitly with this constraint (Supplementary Table 19). Indeed, it seems like that the off-diagonal terms tended to converge towards 0, as showcased in

Supplementary Figure 21, in Panel C, in which the fourth row from top represents the bootstrap intervals associated to the pairwise covariance between dimension 1 and 2 of each cluster.

Unbalanced and overlapping components

With unbalanced components and high OVL (scenario U9 in Table 4), all packages, no matter the initialisation method, provided biased estimates, with a higher variability of the parameter estimates compared to other configurations. The least biased estimates were obtained with k-means or random initialisation, but with a higher variability on the estimates with random initialisation (Supplementary Table 9). Delving further into the individual analysis of the parameter estimates associated to each component, we found out that the least biased estimates were achieved with rebmix initialisation for the most distinguishable components. For instance, in our configuration, the clusters 2 and 4 (see Supplementary Figure 7 and Table 9) were better characterised with the rebmix method. This observation aligns with the rebmix's underlying framework, using the modes of the distribution for initialising the component (Nagode 2015). With highly-overlapping distributions and unbalanced components, both the choice of the initialisation algorithm and the package have a substantial impact on the quality of the estimation of this mixture.

Two configurations in our bivariate simulation feature distributions with both strong OVL and unbalanced components. Especially, the scenario B11 (Table 5) has the strongest OVL overall, with notably a risk of wrongly assigning minor component 2 to major component 1 of 0.5 (a random method classifying each observation to cluster 1 or 2 would have the same performance).

First, we observe that the the random and rebmix initialisation methods have similar performance, significantly better than k-means or MBHC (Supplementary Figure 11). Specifically, the rebmix method returns the least biased estimates, while the random method is associated with the lowest MSE (respectively for configurations B11 and B15, the supplementary Tables 11 and 14). Second, the estimates differ across packages only in these two complex configurations, with packages **Rmixmod** and **mixtools** returning more accurate estimates than the others. It it is particularly emphasised in Scenario B15, where the component-specific covariance matrices are diagonal with same non-null input, and thus should present spherical density distributions. However, only the first class of packages correctly recovers the spherical bivariate 95% confidence regions while they are slightly ellipsoidal with the second class of packages (Panel B, Supplementary Figure 14).

With full covariance structures and unbalanced proportions, as depicted in the high-dimensional Scenario HD8a) and b) of Table 6, the general observations stated in the previous subsection for the high dimensional setting hold, namely that the least biased estimates are returned by packages not specifically designed for high-dimensional data, with the k-means initialisation (Supplementary Table 12 and supplementary Figure 22). Furthermore, the **EMCluster** and **bgmm** packages and the two packages dedicated to high-dimensional, perform similarly with $n = 200$ observations (sub-scenario a) and $n = 2000$ observations (sub-scenario b), whereas we would expect narrower and less biased confidence intervals by increasing the number of observations by a factor of 10.

Finally, with spherical covariance structures and unbalanced proportions, the best performances, both in terms of bias and variability, are obtained with **flexmix**, **mclust** and **GMKMCharlie**. Indeed, as detailed later in Conclusions, these packages are more sensitive to the choice of the initialisation method and have a greater tendency to get trapped in the neighbourhood of the initial estimates (Supplementary Table 19 and supplementary Figure 22). Accordingly, k-means initialisation performs best since it assumes independent and homoscedastic features for each cluster. Furthermore, **EMMIXmfa** is the package that best estimates the off-diagonal terms in this setting, as highlighted in supplementary Table 19.

Identification of two classes of packages with distinct behaviours

By summarizing the results obtained across all simulations, we identify two classes of packages with distinct behaviours (Figure 2):

- The first class of packages, represented by **Rmixmod** and **mixtools**, returns similar estimates to our baseline EM implementation. The estimates returned by these packages are less biased but at the extent of a higher variability on the estimates. Additionally, with overlapping mixtures, they tend to be slower compared to the second class, since they require additional steps to reach convergence.

- The second class of packages, composed of the other reviewed packages, is more sensitive to the initialisation method. This leads to more biased but less variable estimates, especially when assumptions done by the initialisation algorithm are not met.

Panels A, B and C display, respectively in the univariate, bivariate and high-dimensional setting, the heatmap of the Pearson correlation between the estimated parameters across the benchmarked packages for the most discriminative scenario (the one featuring the most unbalanced and overlapping components): scenario U9, Table 4 in the univariate setting, scenario B11, Table 5, for the bivari-
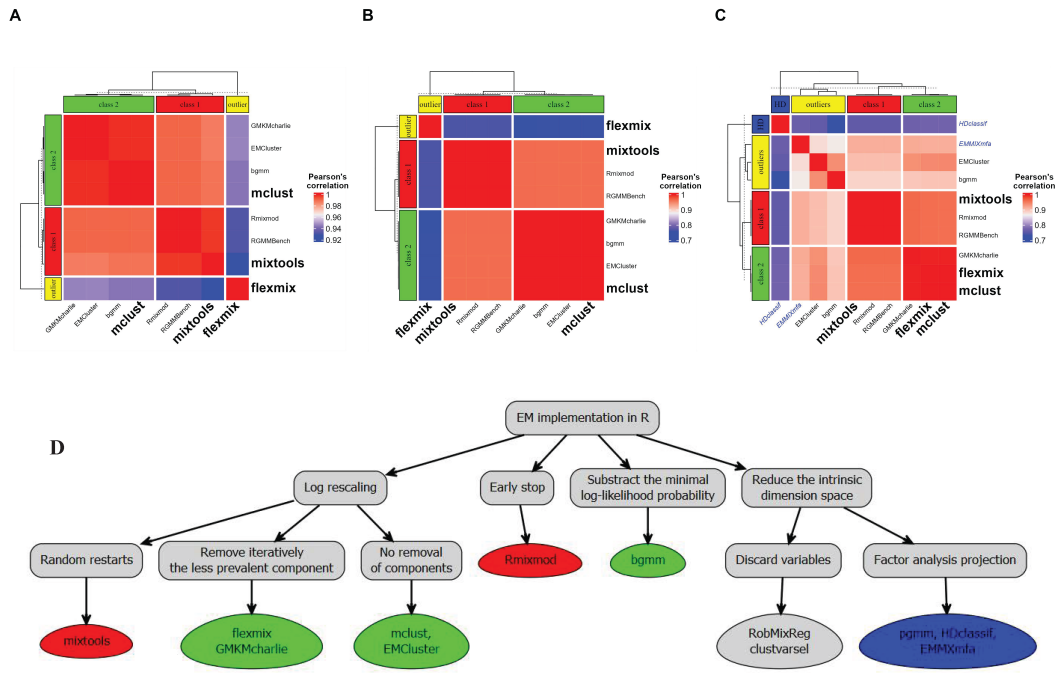
Figure 2: Panels A, B and C show respectively the heatmap of the Pearson correlation in the univariate, bivariate and high-dimensional framework between the parameters estimated by the packages, evaluated for the most discriminating and complex scenario. The correlation matrix was computed using the function stats::cor with option *complete* to remove any missing value related to a failed simulation, and the heatmap generated with the Bioconductor package ComplexHeatmap. Panel D represents a tree summarising the main differences between the benchmarked packages, in terms of the EM implementation. They are discussed in more detail in Appendix *EM-implementation differences across reviewed packages*.

ate simulation and scenario HD8, Table 6 for the high-dimensional simulation, with the *k*-means initialisation.

We further identified with this representation minor differences for the estimation of the parameters between **Rmixmod** and **mixtools**, while three subgroups can be identified in the second class of packages: the first subset with **bgmm** and **mclust**, the second subset with **EMCluster** and **GMKMcharlie** packages and the **flexmix** package, which clearly stands out from the others, as being the one most likely to be trapped at the boundaries of the parameter space. After examining the source codes of the packages, we attribute this differences to custom implementation choices of the EM algorithm, such as the way numerical underflow is managed or the choice of a relative or absolute scale to compare consecutive computed log-likelihoods (see Appendix *EM-implementation differences across reviewed packages* and Panel D, Figure 2). In the high-dimensional setting, the second class of packages showed additional heterogeneity, with **EMCluster** and **bgmm** setting themselves apart from the other three packages.

Failed estimations

Finally, in some cases, neither the specific EM algorithm implemented by each package nor the initialisation method were able to return an estimate with the expected number of components, or converged to a degenerate distribution (e.g., with infinite or zero variances). In that case, we considered the estimation as failed and accordingly we did not include it into the visualisations and the summary metric tables.

Most of the failed estimations occurred with the rebmix initialisation. Indeed, an updated version of the package forced the user to provide a set of possible positive integer values for the number of components, with at least a difference of two between the model with the most components and the model with the least components (we therefore set the parameter *cmax* to $k + 1$ and *cmin* to $k - 1$).In scenarios where the distributions associated with each cluster exhibit significant overlap, there is an increased risk of incorrectly estimating the number of components. This arises from the inherent difficulty of discerning the modes within the overall distribution. For instance, in the most complex scenario B11, characterized by strong overlap and imbalanced clusters (refer to Table 5), up to 20% of initialisations were unsuccessful. Similarly, in the second most challenging scenario, B15, approximately 10% of initializations failed against an averaged number of 4% of the simulations

exhibiting an inaccurate estimation of the number of components.

Removing errors proceeding from the initialisation method, only the **flexmix** package failed in returning an estimate matching the user criteria in some configurations of the univariate and bivariate settings. In both cases, the strong assumption that any cluster with less than 5% of the observations is irrelevant, results in trimming one or more components[9]. This strong agnostic constraint on component proportions led to failures in configurations featuring strongly overlapping clusters, with up to 20% failed estimations with the random initialisation method in scenario B11 (Table 5) and 80% failed estimations in the univariate setting[10] with the rebmix initialisation with scenario U9, Table 4.

In a relatively high dimensional framework, as tested on our third benchmark (Table 6), none of the algorithms that were initialised with the random method (`EMCluster::rand.EM()`) converged successfully. Indeed, of the 16 configurations tested, the covariance returned during the initialisation was systematically non-positive definite for at least one of the components, violating the properties of covariance matrices. Furthermore, an analysis of summary metrics in scenarios HD1 and HD8, reported in supplementary Tables 20 and 21, revealed a notably higher rate of failures when employing rebmix initialisation in conjunction with packages tailored for high dimensionality, such as **HDclassif EMMIXmfa**. This discrepancy was in stark contrast to the more reliable and consistent initial estimates returned by k-means or hierarchical clustering.

Furthermore, as shown by the comparison of summary metrics with $n = 200$ and $n = 2000$ observations in supplementary Tables 20 and 21, respectively for the simplest scenario HD1 and the most complex one HD8, the rebmix initialisation on the one hand, and the packages dedicated to high dimensionality or those of the second class of packages that show a particular behaviour, present much more failures than the k-means or hierarchical clustering initialisation.

3 Conclusions

There are many packages that implement the EM algorithm for estimating the parameters of GMMs. But only few are regularly updated, implement both the unconstrained univariate and multivariate GMM, and enable the user to provide its own initial estimates. Hence, among the 54 packages dealing with GMMs available on CRAN or Bioconductor repositories, we focused our review on 7 packages which implement all of these features. We believe that our in-depth review of the packages can help users to quickly find the best package for their clustering pipeline and highlight limitations in the implementation of some packages. Our benchmark covers a much broader range of configurations than the previously-published studies (Nityasuddhi and Böhning 2003; Lourens et al. 2013; Leytham 1984; Xu and Knight 2010), as we studied the impact of the level of overlap and the imbalance of the mixture proportions on the quality of the estimation.

Interestingly, the EM algorithm occasionally yields biased and inefficient estimates when the components overlap a lot, which agrees with the past literature (Lourens et al. 2013; Leytham 1984; Xu and Knight 2010). This appears to go counter to the theoretical results presented by Leytham (1984), which demonstrated the asymptotic consistency, unbiasedness, and efficiency of maximum likelihood estimates of GMMs. However, it's important to note that this theoretical demonstration relies on the definition of a "local" environment, necessitating the prior setting of boundaries within which the theorem's conditions are met (in other words, the definition of the *support*, which delineates the region where the initial values can be sampled from). It's not then surprising that the EM algorithm struggles in reaching the global maximum of the distribution in the presence of multiple local extremes.

When all components are well-separated or have a relatively small number of components (three or fewer), we found that the best estimation (lowest MSE and bias) is reached with the latest initialisation method developed, namely the rebmix one. Notably, the global maximum is always properly found in our simulations with distinguishable components. Yet, with overlapping components, the least biased and variable estimates overall are obtained with k-means initialisation, enforcing the robustness of the method while the assumptions for using it are not met.

On the contrary, with unbalanced and numerous components (above three), the quantiles initialisation leads to the most biased estimates while the rebmix initialisation induces the highest variability. Indeed, rebmix initialisation is not fit for highly overlapping mixtures, since one of its most restrictive assumption is that each generated interval of the empirical mixture distribution can be associated unambiguously to a component (see Initialisation of the EM algorithm and Nagode (2015)).

Furthermore, rebmix is not particularly adjusted to deal with high-dimensional mixtures, displaying systematically poorer performance compared to other initialisation strategies, such as k-means or hierarchical clustering, as illustrated by the summary metrics listed in Appendix *Supplementary Figures*

---

[9]With a two-components mixture like our bivariate scenario, this even implies to consider an unimodal distribution of the dataset

[10]the gap proceeds from the stronger level of imbalance and the greater number of components

and Tables in the HD simulation. Higher risk of returning a sub-optimal extremum likely arises from the increased data sparsity in high dimensional datasets, which grows as the square root of the number of dimensions $\sqrt{D}$ (Convergence of distance definitions). Thus, we expect that most of the equally-large intervals binning the sampling space and that are used to initiate the rebmix algorithm contain either no or only observation, preventing from retrieving the numerically defined mode of the distribution and increasing the risk of initiating the algorithm in a spurious neighbourhood.

About the remaining initialisation strategies, we observed that, even in the well-separated case, random initializations can sometimes yield highly biased estimates, far from the true parameter values. Consistent with our observations, it was shown in Jin et al. (2016) that the probability for the EM algorithm to converge from randomly initialised estimates to a local suboptimal maximum is non null above two components, increasing with the number of components. Additionally, the local maximum of the likelihood function obtained can be arbitrarily worse than the global maximum. Finally, hierarchical clustering does not take into account any uncertainty on the assignment for an observation to a given class, which explains its rather bad performances with overlapping components. Overall, there is always an initialisation algorithm performing better than the hierarchical clustering, and further it is also by far the slowest and most computationally intensive initialisation method (see supplementary Figure 10).

Our study reveals that while the EM algorithm is supposed to be deterministic, the estimates obtained from its implementations can differ across packages. We were able to link these differences with custom choices of EM implementations across the benchmarked packages. Two distinct classes of packages emerge, each with specific approaches to address certain limitations of the EM algorithm. The first class, exemplified by **mixtools** and **Rmixmod** typically yields smaller but less biased estimates that exhibit lower sensitivity to the choice of initialization method. However, these estimates tend to have higher variability and require longer running times to achieve convergence. The second class, composed of the remaining packages, provide estimates with reduced MSE, but at the extent of a higher bias on the estimates. One plausible explanation is that the first class of packages, comparing absolute iterations of the function to be maximised, tends on average to perform more iterations. The estimated results are accordingly more consistent and closer to the true MLE estimation but at the expense of an increased risk of getting trapped in a local extrema or a plateau, explaining the greater number of outliers observed. Among them, **flexmix** stands out by choosing an unbiased but non MLE-estimate of the covariance matrix, without any clear improvement of the overall performance in our simulations.

Based on these results, we design a decision tree indicating the best choice of package and initialisation method in relation with the shape of the distribution, displayed in Figure 3. Interestingly, our conclusions are consistent between the univariate and bivariate settings. Furthermore, most of the general recommendations on the best choices of packages with respect to the characteristics of the mixture model generally hold in a relatively higher dimensional setting[11]. From this, we assume that projection into a lower-dimensional space is only beneficial in a very high-dimensional setting, for example when the number of dimensions exceeds the number of observations, or when unrestricted parameter estimation (with the full covariance structure) is practically infeasible for computational reasons.

Comparing all these packages suggest several improvements.

1. The use of C++ code speeds up the convergence of the EM algorithm compared to a native R implementation.

2. All packages dealing with GMMs should use *k*-means for overlapping, complex mixtures and rebmix initialisation for well-separated components, provided that the dimension of the dataset is relatively small. The final choice between these two could be set after a first run or visual inspection aiming at determining roughly the level of entropy across mixture proportions and the degree of overlap between components.

3. The packages should allow the user to set their own termination criteria (either relative or absolute log-likelihood or over the estimates after normalisation). Interestingly, **EMMIXmfa** is the only package among those examined that allows the user to consider an absolute or relative convergence endpoint of the EM algorithm, through the `conv_measure` attribute, with `diff` and `ratio` options respectively.

4. With a great number of components or complex overlapping distributions, the optimal package should integrate prior information when available, e.g. via Bayesian estimation.

While **mclust** appeared as the most complete package to model GMMs in R, none of the packages reviewed in this report features all the characteristics mentioned above. We thus strongly believe that

---

[11]We should note, however, that a larger sample space revealed that the packages **bgmm** and **EMCluster** display more biased and noisy parameters compared to the other packages benchmarked and that their performance was surprisingly unaffected by the number of simulated realisations
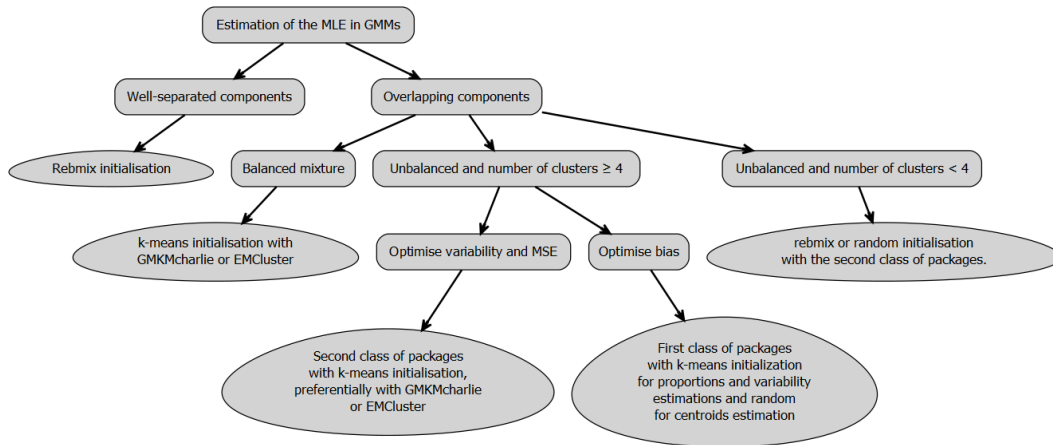
Figure 3: A decision tree to select the best combination of package and initialisation method with respect to the main characteristics of the mixture. It's worth pointing that in both univariate and low dimension multivariate settings, the recommandations are similar.

our observations will help users identify the most suitable packages and parameters for their analyses and guide the development or updates of future packages.

4 Simulation settings

For ease of reading, we reproduce below the parameter configurations used to run the three benchmarks, respectively for the univariate (Table 4), bivariate (5) and high dimensional setting (Table 6).

Table 4: The 9 parameter configurations tested to generate the samples of the univariate experiment, with $k = 4$ components.

ID	Entropy	OVL	Proportions	Means	Correlations
U1	1.00	3.3e-05	0.25 / 0.25 / 0.25 / 0.25	0 / 4 / 8 / 12	0.3 / 0.3 / 0.3 / 0.3
U2	1.00	5.7e-03	0.25 / 0.25 / 0.25 / 0.25	0 / 4 / 8 / 12	1 / 1 / 1 / 1
U3	1.00	2.0e-02	0.25 / 0.25 / 0.25 / 0.25	0 / 4 / 8 / 12	2 / 2 / 2 / 2
U4	0.96	3.3e-05	0.2 / 0.4 / 0.2 / 0.2	0 / 4 / 8 / 12	0.3 / 0.3 / 0.3 / 0.3
U5	0.96	5.8e-03	0.2 / 0.4 / 0.2 / 0.2	0 / 4 / 8 / 12	1 / 1 / 1 / 1
U6	0.96	2.0e-02	0.2 / 0.4 / 0.2 / 0.2	0 / 4 / 8 / 12	2 / 2 / 2 / 2
U7	0.68	2.7e-05	0.1 / 0.7 / 0.1 / 0.1	0 / 4 / 8 / 12	0.3 / 0.3 / 0.3 / 0.3
U8	0.68	4.4e-03	0.1 / 0.7 / 0.1 / 0.1	0 / 4 / 8 / 12	1 / 1 / 1 / 1
U9	0.68	1.5e-02	0.1 / 0.7 / 0.1 / 0.1	0 / 4 / 8 / 12	2 / 2 / 2 / 2

5 Additional files

- Additional files related to the univariate setting
    - **S1.** Bootstrap distributions of the estimated parameters for each scenario described in 4.
    - **S2.** Mean, standard deviation, bias and MSE for each individually estimated parameter in configurations listed in 4.
    - **S3.** Distribution of the running times taken for the EM estimation of the parameters of the GMM, across all nine configurations described in 4, for each benchmarked package. We selected the k-means algorithm to initialise the EM algorithm, as being the least variable for a given package and scenario.

Table 5: The 20 parameter configurations tested to generate the samples of the bivariate experiment.

ID	Entropy	OVL	Proportions	Means	Correlations
B1	1.00	0.15000	0.5 / 0.5	(0,2);(2,0)	-0.8 / -0.8
B2	1.00	0.07300	0.5 / 0.5	(0,2);(2,0)	-0.8 / 0.8
B3	1.00	0.07300	0.5 / 0.5	(0,2);(2,0)	0.8 / -0.8
B4	1.00	0.00078	0.5 / 0.5	(0,2);(2,0)	0.8 / 0.8
B5	1.00	0.07900	0.5 / 0.5	(0,2);(2,0)	0 / 0
B6	1.00	0.00000	0.5 / 0.5	(0,20);(20,0)	-0.8 / -0.8
B7	1.00	0.00000	0.5 / 0.5	(0,20);(20,0)	-0.8 / 0.8
B8	1.00	0.00000	0.5 / 0.5	(0,20);(20,0)	0.8 / -0.8
B9	1.00	0.00000	0.5 / 0.5	(0,20);(20,0)	0.8 / 0.8
B10	1.00	0.00000	0.5 / 0.5	(0,20);(20,0)	0 / 0
B11	0.47	0.06600	0.9 / 0.1	(0,2);(2,0)	-0.8 / -0.8
B12	0.47	0.01600	0.9 / 0.1	(0,2);(2,0)	-0.8 / 0.8
B13	0.47	0.05000	0.9 / 0.1	(0,2);(2,0)	0.8 / -0.8
B14	0.47	0.00045	0.9 / 0.1	(0,2);(2,0)	0.8 / 0.8
B15	0.47	0.03900	0.9 / 0.1	(0,2);(2,0)	0 / 0
B16	0.47	0.00000	0.9 / 0.1	(0,20);(20,0)	-0.8 / -0.8
B17	0.47	0.00000	0.9 / 0.1	(0,20);(20,0)	-0.8 / 0.8
B18	0.47	0.00000	0.9 / 0.1	(0,20);(20,0)	0.8 / -0.8
B19	0.47	0.00000	0.9 / 0.1	(0,20);(20,0)	0.8 / 0.8
B20	0.47	0.00000	0.9 / 0.1	(0,20);(20,0)	0 / 0

- **S4.** Distribution of the time computations taken by the six initialisation methods listed in Table 3.

- Additional files related to the outliers setting:

  - **S5.** Bootstrap distributions of the estimated parameters used to generate Supplementary Figure 2. We additionally include the **otrimle** package, dedicated to these extreme distributions. Two configurations were tested, introducing 2% and 4% of outliers drawn from an improper uniform distribution.
  - **S6.** Mean, standard deviation, bias and MSE for each individually estimated parameter in both configurations visualised on Supplementary Figure 2, for each combination of package and initialisation method.

- Additional files related to the bivariate benchmark:

  - **S7.** Bootstrap distributions of the estimated parameters for each scenario described in 5.

  - **S8.** Mean, standard deviation, bias and MSE for each individually estimated parameter in configurations listed in 5.

Table 6: The 16 parameter configurations tested to generate the samples in a high dimensional context. The first digit of each ID index refers to an unique parameter configuration (identified by its level of overlap, entropy and topological structure, either circular or ellipsoidal, of the covariance matrix, while the lowercase letter depicts the number of observations, a) with $n = 200$ and b) with $n = 2000$.

ID	OVL	Number of observations	Proportions	Spherical
HD1a	1e-04	200	0.5 / 0.5	☑
HD1b	1e-04	2000	0.5 / 0.5	☑
HD2a	1e-04	200	0.19 / 0.81	☑
HD2b	1e-04	2000	0.19 / 0.81	☑
HD3a	1e-04	200	0.5 / 0.5	☒
HD3b	1e-04	2000	0.5 / 0.5	☒
HD4a	1e-04	200	0.21 / 0.79	☒
HD4b	1e-04	2000	0.21 / 0.79	☒
HD5a	2e-01	200	0.5 / 0.5	☑
HD5b	2e-01	2000	0.5 / 0.5	☑
HD6a	2e-01	200	0.15 / 0.85	☑
HD6b	2e-01	2000	0.15 / 0.85	☑
HD7a	2e-01	200	0.5 / 0.5	☒
HD7b	2e-01	2000	0.5 / 0.5	☒
HD8a	2e-01	200	0.69 / 0.31	☒
HD8b	2e-01	2000	0.69 / 0.31	☒

- **S9.** Distribution of the running times taken for the EM estimation of the parameters of the GMM, across all twenty configurations described in 5, for each benchmarked package. We selected the k-means algorithm to initialise the EM algorithm, as being the least variable for a given package and scenario.

- Additional files related to the high-dimensional benchmark:

  - **S10.** Bootstrap distributions of the estimated parameters for each scenario described in 6.

  - **S11.** Mean, standard deviation, bias and MSE for each individually estimated parameter in configurations listed in 6.

  - **S12.** Distribution of the running times taken for the EM estimation of the parameters of the GMM, across all twenty configurations described in 6, for each benchmarked package. We selected the k-means algorithm to initialise the EM algorithm, as being the least variable for a given package and scenario.

References

Baek, Jangsun, Geoffrey J. McLachlan, and Lloyd K. Flack. 2010. "Mixtures of Factor Analyzers with Common Factor Loadings: Applications to the Clustering and Visualization of High-Dimensional Data." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2009.149.

Banfield, Jeffrey D., and Adrian E. Raftery. 1993. "Model-Based Gaussian and Non-Gaussian Clustering." *Biometrics*. https://doi.org/10.2307/2532201.

Berge, Laurent, Charles Bouveyron, and Stephane Girard. 2019. *HDclassif: High Dimensional Supervised Classification and Clustering*.

Biernacki, Christophe, Gilles Celeux, and Gérard Govaert. 2003. "Choosing Starting Values for the EM Algorithm for Getting the Highest Likelihood in Multivariate Gaussian Mixture Models." *Computational Statistics & Data Analysis*. https://doi.org/10.1016/S0167-9473(02)00163-9.

Bouveyron, Charles, and Stéphane Girard. 2009. "Robust Supervised Classification with Mixture Models: Learning from Data with Uncertain Labels." *Pattern Recognition*. https://doi.org/10.1016/j.patcog.2009.03.027.

Celeux, Gilles, and Gérard Govaert. 1992. "A Classification EM Algorithm for Clustering and Two Stochastic Versions." *Computational Statistics & Data Analysis*. https://doi.org/10.1016/0167-9473(92)90042-E.

Chen, Jiahua. 2016. "Consistency of the MLE Under Mixture Models." https://doi.org/10.1214/16-STS578.

Cochran, W. G. 1934. "The Distribution of Quadratic Forms in a Normal System, with Applications to the Analysis of Covariance." *Mathematical Proceedings of the Cambridge Philosophical Society*. https://doi.org/10.1017/S0305004100016595.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. "Maximum Likelihood from Incomplete Data Via the *EM* Algorithm." *Journal of the Royal Statistical Society*. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x.

Fraley, Chris. 1998. "Algorithms for Model-Based Gaussian Hierarchical Clustering." *SIAM Journal on Scientific Computing*. https://doi.org/10.1137/S1064827596311451.

Geary, R. C. 1936. "The Distribution of "Student's" Ratio for Non-Normal Samples." *Supplement to the Journal of the Royal Statistical Society*. https://doi.org/10.2307/2983669.

Jin, Chi, Yuchen Zhang, Sivaraman Balakrishnan, et al. 2016. "Local Maxima in the Likelihood of Gaussian Mixture Models: Structural Results and Algorithmic Consequences." Curran Associates, Inc. https://doi.org/https://doi.org/10.48550/arXiv.1609.00978.

Koopman, B. O. 1936. "On Distributions Admitting a Sufficient Statistic." *Transactions of the American Mathematical Society*. https://doi.org/10.2307/1989758.

Kwedlo, Wojciech. 2013. "A New Method for Random Initialization of the EM Algorithm for Multivariate Gaussian Mixture Learning." In *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013*, edited by Robert Burduk, Konrad Jackowski, Marek Kurzynski, Michał Wozniak, and Andrzej Zolnierek. Springer International Publishing. https://doi.org/10.1007/978-3-319-00969-8/_8.

Leytham, K. M. 1984. "Maximum Likelihood Estimates for the Parameters of Mixture Distributions." *Water Resources Research*. https://doi.org/10.1029/WR020i007p00896.

Lourens, Spencer, Ying Zhang, Jeffrey D Long, et al. 2013. "Bias in Estimation of a Mixture of Normal Distributions." *Journal of Biometrics & Biostatistics*. https://doi.org/10.4172/2155-6180.1000179.

McLachlan, Geoffrey, and David Peel. 2000. *Finite Mixture Models: McLachlan/Finite Mixture Models*. John Wiley & Sons, Inc. https://doi.org/10.1002/0471721182.

Melnykov, Volodymyr, Wei-Chen Chen, and Ranjan Maitra. 2021. *MixSim: Simulating Data to Study Performance of Clustering Algorithms*.

Nagode, Marko. 2015. "Finite Mixture Modeling via REBMIX." *Journal of Algorithms and Optimization*.
———. 2022. *Rebmix: Finite Mixture Modeling, Clustering & Classification*.

Nityasuddhi, Dechavudh, and Dankmar Böhning. 2003. "Asymptotic Properties of the EM Algorithm Estimate for Normal Mixture Models with Component Specific Variances." *Computational Statistics & Data Analysis*. https://doi.org/10.1016/S0167-9473(02)00176-7.

Panic, Branislav, Jernej Klemenc, and Marko Nagode. 2020. "Improved Initialization of the EM Algorithm for Mixture Model Parameter Estimation." *Mathematics*. https://doi.org/10.3390/math8030373.

R Core Team. 2023. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Rathnayake, Suren, Geoff McLachlan, David Peel, et al. 2019. *EMMIXmfa: Mixture Models with Component-Wise Factor Analyzers*.

Scrucca, Luca, and Adrian E. Raftery. 2015. "Improved Initialisation of Model-Based Clustering Using Gaussian Hierarchical Partitions." *Advances in Data Analysis and Classification*. https:

//doi.org/10.1007/s11634-015-0220-z.

Shimizu, Naoto, and Hiromasa Kaneko. 2020. "Direct Inverse Analysis Based on Gaussian Mixture Regression for Multiple Objective Variables in Material Design." *Materials & Design*. https://doi.org/10.1016/j.matdes.2020.109168.

Xu, Dinghai, and John Knight. 2010. "Continuous Empirical Characteristic Function Estimation of Mixtures of Normal Parameters." *Econometric Reviews*. https://doi.org/10.1080/07474938.2011.520565.

Bastien Chassagnol
Laboratoire de Probabilités, Statistiques et Modélisation (LPSM), UMR CNRS 8001
4 Place Jussieu Sorbonne Université
75005, Paris, France
ORCiD: 0000-0002-8955-2391
bastien_chassagnol@laposte.net

Antoine Bichat
Les Laboratoires Servier
50 Rue Carnot
92150, Suresnes, France
https://rdrr.io/github/abichat/abutils/
ORCiD: 0000-0001-6599-7081
antoine.bichat@servier.com

Cheïma Boudjeniba
Systems Biology Group, Dept. of Computational Biology, Institut Pasteur
25 Rue du Dr Roux
75015 Paris
cheima.boudjeniba@servier.com

Pierre-Henri Wuillemin
Laboratoire d'Informatique de Paris 6 (LIP6), UMR 7606
4 Place Jussieu Sorbonne Université
75005, Paris, France
http://www-desir.lip6.fr/~phw/
ORCiD: 0000-0003-3691-4886
pierre-henri.wuillemin@lip6.fr

Mickaël Guedj
Les Laboratoires Servier
50 Rue Carnot
92150, Suresnes, France
https://michaelguedj.github.io/
ORCiD: 0000-0001-6694-0554
mickael.guedj@gmail.com

David Gohel
ArData
59 rue Voltaire
92800, PUTEAUX, France
https://www.ardata.fr/expertise-r/
ORCiD: 0000-0003-2837-8884
david.gohel@ardata.fr

Gregory Nuel
Laboratoire de Probabilités, Statistiques et Modélisation (LPSM), UMR CNRS 8001
4 Place Jussieu Sorbonne Université

75005, Paris, France
http://nuel.perso.math.cnrs.fr/
ORCiD: 0000-0001-9910-2354
Gregory.Nuel@math.cnrs.fr

Etienne Becht
Les Laboratoires Servier
50 Rue Carnot
92150, Suresnes, France
ORCiD: 0000-0003-1859-9202
etienne.becht@servier.com

mutualinf: An R Package for Computing and Decomposing the Mutual Information Index of Segregation

by Rafael Fuentealba-Chaura, Daniel Guinea-Martin, Ricardo Mora, and Julio Rojas-Mora

Abstract In this article, we present the R package mutualinf for computing and decomposing the mutual information index of segregation by means of recursion and parallelization techniques. The mutual information index is the only multigroup index of segregation that satisfies strong decomposability properties, both for organizational units and groups. The mutualinf package contributes by (1) implementing the decomposition of the mutual information index into a "between" and a "within" term; (2) computing, in a single call, a chain of decompositions that involve one "between" term and several "within" terms; (3) providing the contributions of the variables that define the groups or the organizational units to the overall segregation; and (4) providing the demographic weights and local indexes employed in the computation of the "within" term. We illustrate the use of mutualinf using Chilean school enrollment data. With these data, we study socioeconomic and ethnic segregation in schools.

1 Introduction

Typically, segregation is measured using groups of individuals assigned to organizational units. A segregation index is a mathematical function that maps the joint distribution of groups and organizational units into $\mathbb{R}^+$. Traditional measures, such as the dissimilarity (Duncan and Duncan, 1955) or the Gini (Flückiger and Silber, 1999) indices, are appropriate for computing segregation when (i) there are two groups and (ii) the organizational units lack a hierarchical or multilevel structure.

One classic example from the segregation literature addresses measuring segregation among Black and White students in schools. However, even in this simple scenario, there may be information in the data that calls for a less restrictive use of the indices. First, there are often multiple sources of identity and affiliation that may result in nondichotomous groups (Akerlof and Kranton, 2010). For example, ethnic classification may include more than two categories. Additionally, the definition of groups can be extended by operating a Cartesian product among multiple sources of segregation: ethnicity, gender, religion, income, and language are consequential factors defining group membership in many societies. In this case, we would need to use a multigroup segregation index (Reardon and Firebaugh, 2002).

Second, we may hypothesize that multigroup ethnic segregation in schools is produced via two channels. One is the segregation between the ethnic majority and the remainder. The second channel is the segregation among minorities. In this scenario, we should be interested in decomposing the overall value of a segregation index into (i) a "between" term (gauging segregation in schools between the majority and the minorities) and (ii) a "within" term (computing segregation among the minorities). An index that satisfies the so-called *strong group decomposability* (*SGD*) property achieves this goal in the best possible way (Frankel and Volij, 2011).

We are also interested in another multilevel structure of the situation. For example, in many countries, schools belong to districts, and district authorities may have powers over the assignment of students to schools. We should examine: (i) how much ethnic segregation is in districts and (ii) how much is in schools. For this, we'll break down the total segregation into that of larger areas (districts) and specific units (schools). Complying with the so-called *strong unit decomposability* (*SUD*) property addresses this goal (Frankel and Volij, 2011).

As far as we know, the mutual information or *M* index—originally proposed to study race segregation in Chicago's public schools (Theil and Finizza, 1971)—is the only multigroup segregation index that simultaneously satisfies the two abovementioned properties.

There are two packages in R to compute *M* and its decompositions. First, the `mutual_total` function of the **segregation** package (Elbers, 2021) computes the index itself and the "within" term of the decomposition. By calling `mutual_total` twice, we obtain a simple "between"-"within" decomposition.[1] Second, in this article, we present the **mutualinf** package (Fuentealba-Chaura et al., 2021) for

---

[1]This package has additional features not directly related to the decompositions discussed in this paper. It (i) computes local segregation scores; (ii) uses bootstrap procedures to conduct inferential analyses; (iii) decomposes pairwise comparisons of indices to solve the problem of marginal dependence; and (iv) computes the *H* index, which is a normalization of the *M* index.

computing complex "between"-"within" decompositions, i.e., decompositions with multiple "within" terms.

In the next section, we introduce the M index and its decomposability properties. Next, we illustrate its usage with an application to a Chilean school enrollment dataset. We then explain the package structure. The last section summarizes the main contributions of the **mutualinf** package.

2 Statistical model

Consider two discrete random variables, *unit* and *group*. Let p_{ng} represent the joint proportion of individuals for whom $unit = n$ and $group = g$. Then,

$$P_{unit,group} = \{p_{ng} | \forall n \in 1, \dots, N, \forall g \in 1, \dots, G\} \tag{1}$$

denotes the joint distribution of the discrete random variables *unit* and *group*, and $\sum_{n=1}^{N} \sum_{g=1}^{G} p_{ng} = 1$.

In addition, let $p_{n\bullet} = \sum_{g=1}^{G} p_{ng}$ such that

$$P_{unit} = \{p_{n\bullet} | \forall n \in 1, \dots, N\} \tag{2}$$

denotes the marginal distribution of individuals across units. Similarly, let $p_{\bullet g} = \sum_{n=1}^{N} p_{ng}$, and

$$P_{group} = \{p_{\bullet g} | \forall g \in 1, \dots, G\} \tag{3}$$

be the marginal distribution across groups. Finally, let

$$P_{unit|g} = \left\{ \frac{p_{ng}}{p_{\bullet g}} | \forall g \text{ and } n \right\} \tag{4}$$

and

$$P_{group|n} = \left\{ \frac{p_{ng}}{p_{n\bullet}} | \forall g \text{ and } n \right\} \tag{5}$$

be the conditional distributions across units and groups.

The M index is the weighted average of the natural logarithm of the ratio between (i) the actual joint distribution of units and groups and (ii) the joint distribution under the hypothesis of independence or no association.

$$M = \sum_{g=1}^{G} \sum_{n=1}^{N} p_{ng} \log \left(\frac{p_{ng}}{p_{n\bullet} p_{\bullet g}} \right), \tag{6}$$

where we set $0 \times \log_b \left(\frac{1}{0} \right) = 0$. For simplicity, the M index uses the natural logarithm, although any base will work:

$$M_b = \frac{M}{\log_e (b)}. \tag{7}$$

As a measure of association, the index captures the excess uncertainty that exists when we learn about someone's unit and group separately rather than jointly. When groups and units are independent, the joint proportions p_{ng} equal the proportions under independent assignment: $p_{ng} = p_{n\bullet} p_{\bullet g}$.[2] Then, naturally, $M = 0$. Note that this is the minimum of the index. In effect, the M index is nonnegative and less than or equal to the logarithm of $\min\{G, N\}$. Often, $G < N$; and so the index reaches its maximum value, $\log(G)$, whenever equal-sized groups are isolated in separate units.

The M index builds on the concept of entropy from information theory (Kullback, 1959): the average information attained when we learn the value of a discrete variable. Frankel and Volij (2011) characterize M with six ordinal axioms. Interestingly, and rather conveniently, the M index can be expressed in two different but equivalent ways that represent two notions of segregation that Massey and Denton (1988) propose: segregation as the average departure of (i) $P_{group|n}$ from P_{group} and (ii) of $P_{unit|g}$ from P_{unit}.

---

[2]It can be shown that the M index is a monotonic transformation of the likelihood ratio test of random assignment across units and groups (Zoloth, 1974).

2.1 Decomposability properties

Social scientists who study sources of segregation should use an index that satisfies either *SUD* or *SGD* or both depending on whether the interest lies in sources stemming from units, groups, or both (Mora and Ruiz-Castillo, 2003; Frankel and Volij, 2011; Mora and Ruiz-Castillo, 2011).

Let *S* be an *SUD* index of segregation defined over variables *group* and *unit*. Consider a partition of all the units, $n = 1, \ldots, N$, into *K* major units, $k = 1, \ldots, K$. Let S_K be the index of segregation that we obtain after taking the *K* major units as organizational units. S_K is usually referred to as the "between" term. Following Frankel and Volij (2011), we decompose an *SUD* index as:

$$S = S_K + \sum_{k=1}^{K} p_{k\bullet} S(k), \tag{8}$$

where $p_{k\bullet}$ is the demographic share of major unit *k*, and $S(k)$ is the segregation index of major unit *k*. With an *SUD* index, the following scenario holds: if the differences in group proportions across the units of each major unit *k* vanished but the differences in group proportions across major units remained, segregation would decrease by the amount in $\sum_{k=1}^{K} p_{k\bullet} S(k)$. (The latter expression is commonly referred to as the "within" term.) In other words, $\sum_{k=1}^{K} p_{k\bullet} S(k)$ represents the exclusive contribution of *unit* segregation that arises within major units (e.g., school segregation within districts); therefore, it is unrelated to segregation that arises in major units (Mora and Ruiz-Castillo, 2011). Conversely, for a partition of *group*, we can decompose an *SGD* index into a "between" and a "within" term following a similar procedure.

As mentioned above, we can study sources of segregation only with indices that satisfy the *SUD* and/or *SGD* properties. Consider, for example, the situation in which the Cartesian product of several discrete variables define *group*. Let *A* be the subset of all the variables defining *group*. If index *S* satisfies the *SGD* property, we can decompose it card(*A*) times. Each decomposition would take the categories resulting from excluding one variable at a time as the major groups. Denote $S(A \backslash j), \forall j \in A$ as the "within" term in the decomposition that takes the Cartesian product of all the variables in $A \backslash j$ as supergroups. This term, $S(A \backslash j)$, identifies the exclusive contribution of the variable *j* to the overall group segregation in variable *unit*. For notational simplicity, let us express this as follows:

$$S^j := S(A \backslash j). \tag{9}$$

In general, $S \neq \sum_{j=1}^{J} S^j$. We can always define:

$$I = S - \sum_{j=1}^{J} S^j. \tag{10}$$

Therefore, $I \in \mathbb{R}$ can be interpreted as the "interaction" among all the variables in *A*, i.e., the slack or surplus in *S* that cannot exclusively be attributed to any variable in *A*. Hence:

$$S = \sum_{j=1}^{J} S^j + I. \tag{11}$$

Conversely, with index *S* satisfying *SUD*, we can study the segregation stemming from variable *unit*. If *S* satisfies both *SGD* and *SUD*, we can identify the exclusive contributions to overall segregation that come from all variables defining *group and unit*. The contributions of the *group* variables are computed on the segregation defined along all *unit* variables. Conversely, the contributions of the *unit* variables are computed on the segregation defined using all the *group* variables. Correspondingly, the resulting conceptual framework does not allow for simultaneously measuring the contributions of the *group* and *unit* variables. Nonetheless, it is possible to compute the segregation that stems from a subset of the *unit* and *group* variables.

The *M* index is the only multigroup segregation index that is known to satisfy the *SUD* and *SGD* properties simultaneously and that can, therefore, implement decompositions 8 and 11.

3 Illustration of mutualinf with school data

We illustrate the use of mutualinf with Chilean school enrollment data from the 2016-2018 period. The data include all students ($n = 287{,}546$) in the schools of the regions of Biobío, La Araucanía, and Los Ríos. These are the three administrative regions with the largest proportion of Mapuche people, the main ethnic minority in Chile.

Variable	Description
Individual Characteristics	
year	Student enrollment/academic year.
gender	Student gender code.
csep	Preferential student allowance category.
grade	Student grade/year.
ethnicity	Self-reported Native ethnicity.
School Characteristics	
school	School ID.
district	Administrative district where the school is located.
region	Administrative region where the school is located.
rural	School with multiage classrooms.
sch_type	Whether the school is public, charter, or private.

Table 1: Variable definitions. Individual-level enrollment data, 2016–2018, Biobío, La Araucanía, and Los Ríos regions (Chile).

We merged three datasets using unique student identifiers and enrollment years. *Datos Abiertos*, "Open Data" in Spanish, (Ministry of Education, https://datosabiertos.mineduc.cl/) is the main dataset. It includes individual-level information about all the students enrolled in grades/years 4 and 8 (grade). This includes their school (school), enrollment/academic year (year), gender (gender), and whether they receive (partially or fully) the government means-tested allowance for students (csep, the acronym of "preferential student allowance" in Spanish *subvención escolar preferencial*).[3] A total of 168,684 (58.7%) students received either the partial or full allowance. Moreover, the data include school-level information: the school ID (there are 2,454 schools), the school's ownership status (public, private, or charter; sch_type), rural or urban location (rural), administrative district (there are 98 administrative districts; district), and region (the three regions abovementioned; region).

The two other datasets are the *Cuestionario de Calidad y Contexto de la Educación para Padres y Apoderados* ("Quality and Education Survey for Parents and Guardians") and the *Cuestionario de Estudiantes* ("Student Survey", Education Quality Agency, https://www.agenciaeducacion.cl/). With them, we construct a proxy for belonging to the Mapuche ethnic minority (ethnicity): we classify a student as Mapuche if he or she, or at least one parent, self-identifies as such. With this, the broadest possible definition of the Mapuche group with these data, there are 41,884 (14.6%) Mapuche students in the sample[4]

3.1 School segregation

In this section, we work with the aggregated school enrollment data described in Table 1. First, we load the package **mutualinf**. The data.table function of this package automatically allows us to use our examples (DT_Seg_Chile):[5]

```
> library(mutualinf)
```

---

[3]The criteria for receiving a full or partial subsidy during the period 2016-2018 are closely related to the socioeconomic level of the student's family. Hence, the type of student subsidy received by each student is a proxy for his or her socioeconomic status, SES. In particular, students who receive no allowance are high SES students; students who receive partial allowance are middle SES students; and students with full allowance are low SES students.

[4]According to the Chilean Statistical Office, in the 2017 Census 12.8 percent of the population consider themselves to belong to one of the native peoples of the country. The "Quality and Education Survey for Parents and Guardians" and the "Student Survey" record only whether a student self-identifies as a member of a native group. Still, the Mapuches account for over 99 percent of the native population in the three regions that we study (Instituto Nacional de Estadísticas, 2018).

[5]The categories for the variable csep in DT_Seg_Chile have been modified so that the table fits the margins: s for subsidized, ps for partially-subsidized, and ns for non-subsidized.

```
> DT_Seg_Chile
      year school district csep    ethnicity rural    region sch_type gender grade fw

   1: 2016   4531     8101   ps non-Mapuche urban    Biobio   public female     4 22
   2: 2016   4531     8101    s non-Mapuche urban    Biobio   public female     4 19
   3: 2016   4531     8101    s     Mapuche urban    Biobio   public female     4  2
   4: 2016   4531     8101   ns non-Mapuche urban    Biobio   public female     4  5
   5: 2016   4531     8101   ps     Mapuche urban    Biobio   public female     4  2
  ---
55956: 2018  22495    14204   ps non-Mapuche urban Los Rios  charter   male     8  1
55957: 2018  22495    14204   ps     Mapuche urban Los Rios  charter   male     8  2
55958: 2018  22495    14204    s     Mapuche urban Los Rios  charter   male     8  5
55959: 2018  22495    14204   ns non-Mapuche urban Los Rios  charter   male     8  1
55960: 2018  22495    14204   ns     Mapuche urban Los Rios  charter   male     8  1
```

Each row in the database is a unique combination of the values that the variables in Table 1 take, representing a subset of the students. The last column, fw, contains the number of students enrolled in each unique subset. The remaining columns correspond to the variables in Table 1.

The mutual function can calculate the M index at its simplest level, i.e., as a measure of group segregation in a set of units. For example, suppose that the objective is to compute the M index of socioeconomic school segregation. In that case, schools define the units, and the preferential student allowance categories (proxies of socioeconomic status) define the three socioeconomic groups. The following code computes socioeconomic segregation in schools:

```
> mutual(data = DT_Seg_Chile,
        group = "csep",
        unit = "school")
        M
1: 0.1995499
```

As previously stated, $\log(\min(G, N))$ is the upper bound of the M index. Hence, a given index value represents different segregation levels depending on G and N. Normalization can help in this situation. Given that there are only three socioeconomic groups but many more schools, the upper bound in this case is $\log(3) \approx 1.0986$. We can use this value to normalize M, i.e., to rescale its value as a proportion of maximum segregation: $0.1995/1.0986 = 0.1816$ or, in other words, in our data, socioeconomic groups generate only $0.1816 \times 100 = 18.16\%$ of the maximum segregation that there could be. The cardinality of *any* normalized index, i.e., the index value we calculate with it, only warrants this limited interpretation. However, it comes at a high price for the normalized M index because it no longer satisfies the *SGD* property. Conversely, if the upper bound were defined by N, the resulting normalized index would no longer satisfy the *SUD* property (Guinea-Martin and Mora, 2021).

To return to our example, we can also compute ethnic segregation in schools:

```
> mutual(data = DT_Seg_Chile,
        group = "ethnicity",
        unit = "school")
         M
1: 0.06213906
```

The mutual function can also take on board multiple group dimensions. For example, we can measure socioeconomic *and* ethnic segregation in schools:

```
> mutual(data = DT_Seg_Chile,
        group = c("csep", "ethnicity"),
        unit = "school")
        M
1: 0.2610338
```

In the above example, the mutual function defines groups as combinations of socioeconomic and ethnic categories. By design, the value of the overall segregation thus obtained (0.2610338) must be greater than or equal to the value for the segregation measured for socioeconomic or ethnic groups separately: 0.1995499 and 0.06213906, respectively. Note that 0.1995499 (0.06213906) is the value of the "between" term in the decomposition of the total segregation in schools (0.2610338) that there is within socioeconomic (ethnic) categories. Given that the "within" term must be nonnegative, it follows that the "between" term cannot be greater than total segregation.

More generally, segregation analyses can be computed using a variety of unit- and/or group-defining variables. For example,

```
> mutual(data = DT_Seg_Chile,
         group = c("csep", "ethnicity"),
         unit = c("school", "district"))
            M
1: 0.2610338
```

computes the socioeconomic and ethnic segregation in combinations of schools and districts. Note that the result, 0.2610338, is identical to that obtained in the previous case. The reason is that each school belongs to only one district; hence, the combinations of schools and districts coincide with the set of schools. This occurs because the two variables are hierarchically related and districts partition schools. In other words, districts do not add information about segregation beyond what is obtained from schools.

However, the variables that define the units may not lie in a hierarchy. For example, defining the units with the school identification code (school) and ownership type (sch_type), we obtain a different value for the overall socioeconomic and ethnic segregation:

```
> mutual(data = DT_Seg_Chile,
         group = c("csep", "ethnicity"),
         unit = c("school", "sch_type"))
            M
1: 0.2610865
```

Why is this so? It turns out that a few schools changed ownership during the sample period, making for a nonhierarchical relationship between school and sch_type, the two unit-defining variables. The consequence is a slight increase in the overall measure of socioeconomic and ethnic segregation from 0.261033 to 0.2610865 that is entirely due to the enlargement of the set of units from schools to the combination of schools and ownership type.

Setting the option by allows us to compute the index for subsets of data separately. In our database, the variable region partitions data into three regions:

```
> mutual(data = DT_Seg_Chile,
         group = c("csep", "ethnicity"),
         unit = c("school", "sch_type"),
         by = "region")
        region        M
1:      Biobio 0.2312423
2: La Araucania 0.2367493
3:    Los Rios 0.2125013
```

The segregation in La Araucania is greater than that in either Biobio or Los Rios. By including more than one variable in option by, the subsets are defined by the Cartesian product of the categories of these variables. To illustrate it, we include the variables region and year to option by:

```
> mutual(data = DT_Seg_Chile,
         group = c("csep", "ethnicity"),
         unit = c("school", "sch_type"),
         by = c("region", "year"))
        region year         M
1:      Biobio 2016 0.2423257
2:      Biobio 2017 0.2599383
3:      Biobio 2018 0.2696983
4: La Araucania 2016 0.2818232
5: La Araucania 2017 0.2749189
6: La Araucania 2018 0.2873032
7:    Los Rios 2016 0.2489342
8:    Los Rios 2017 0.2540016
9:    Los Rios 2018 0.2664027
```

In the above example, we obtain a segregation index for each combination of region and year: the socioeconomic segregation and ethnic segregation in Biobio and Los Rios increase during the sample period (2016-2018); however, the segregation in La Araucania falls in 2017 and grows in 2018.

The within option additively decomposes the total segregation index into a "between" and a "within" term. We return to the by="region" example:

```
> mutual(data = DT_Seg_Chile,
         group = c("csep", "ethnicity"),
         unit = c("school", "sch_type"),
```

```
            by = "region",
            within = "ethnicity")
            region          M M_B_ethnicity M_W_ethnicity
1:          Biobio 0.2312423    0.02582674     0.2054156
2: La Araucania 0.2367493    0.04840892     0.1883404
3:     Los Rios 0.2125013    0.03324738     0.1792539
```

We obtain three terms for each region. The first, M, contains total segregation and matches the values without the within option. The second, M_B_ethnicity, contains values for the "between" term that measures ethnic segregation in the combinations of schools and types of schools. The third, M_W_ethnicity, contains values for the "within" term. These values are the weighted averages of the socioeconomic segregation (in the combinations of schools and types of schools) computed for each ethnic group (with weights equal to the demographic importance of each ethnicity). This "within" term is the part of the total segregation, M, that stems exclusively from socioeconomic differences. Hereafter, we will refer to these "within" terms, which isolate sources of segregation, as "contributions". In this case, M_W_ethnicity is the socioeconomic contribution to total segregation M.

It is also possible to decompose the M index into a "between" and a "within" socioeconomic term:

```
> mutual(data = DT_Seg_Chile,
         group = c("csep", "ethnicity"),
         unit = c("school", "sch_type"),
         by = "region",
         within = "csep")
         region          M   M_B_csep    M_W_csep
1:          Biobio 0.2312423 0.2030819 0.02816039
2: La Araucania 0.2367493 0.1906641 0.04608521
3:     Los Rios 0.2125013 0.1774420 0.03505928
```

Now, we obtain, again, three terms for each region. The first, M, captures total segregation as before. The second, M_B_csep, is the socioeconomic segregation in the combinations of schools and types of schools. The third, M_W_csep, is the ethnic contribution.

The within option also allows us to sequentially conduct more than one decomposition, using either major units and/or supergroups. Consider parsing the combinations of csep, ethnicity, and gender as supergroups:

```
> mutual(data = DT_Seg_Chile,
         group = c("csep", "ethnicity", "gender"),
         unit = c("school", "grade"),
         by = "region",
         within = c("csep", "ethnicity"))
         region          M   M_B_csep    M_W_csep M_W_csep_ethnicity
1:          Biobio 0.3026811 0.2086340  0.04019188         0.05385515
2: La Araucania 0.3261384 0.1970565  0.07945498         0.04962692
3:     Los Rios 0.3328145 0.1833409  0.06585940         0.08361417
```

In the above output, we obtain four terms for each region. The first, M, is the socioeconomic, ethnic, and gender segregation in the combinations of schools and grades, i.e., in the organizational units. The second, M_B_csep, is the socioeconomic segregation in the combinations of schools and grades. The third, M_W_csep, presents the weighted average across all socioeconomic levels of ethnic segregation in schools and grades. (This term is not the contribution of ethnicity because it also includes the interaction between ethnicity and gender.) The fourth, M_W_csep_ethnicity, is the contribution of gender to total segregation. Note that labels M_W_csep and M_W_csep_ethnicity are shortcuts for M_W_csep_B_ethnicity and M_W_csep_W_ethnicity_B_gender, respectively.

Returning to the example without gender, we can directly obtain contributions using the contribution.from option. Take the following example:

```
> mutual(data = DT_Seg_Chile,
         group = c("csep", "ethnicity"),
         unit = c("school", "sch_type"),
         by = "region",
         contribution.from = "group_vars")
         region          M    C_csep C_ethnicity   interaction
1:          Biobio 0.2312423 0.2054156  0.02816039 -0.002333648
2: La Araucania 0.2367493 0.1883404  0.04608521  0.002323710
3:     Los Rios 0.2125013 0.1792539  0.03505928 -0.001811897
```

Following Equations (9) and (10), we obtain four terms for each region: M, C_csep, C_ethnicity, and interaction. M is total segregation, as already presented. C_csep is the socioeconomic contribution. It matches the "within" ethnicity term (M_W_ethnicity) in the first within example above. C_ethnicity is the ethnic contribution. It matches the "within" socioeconomic term (M_W_csep) in the second within example above. Finally, interaction is equal to M minus the sum of C_csep and C_ethnicity, as defined in Equation (10). In other words, the interaction is the part of the total socioeconomic and ethnic segregation that exists in the combinations of schools and school types that cannot exclusively be attributed to either socioeconomic status or ethnicity. The socioeconomic contribution is largest in Biobio (0.2054156), and the ethnicity contribution is largest in La Araucania (0.04608521).

A positive interaction term, such as the one in La Araucania, signals that socioeconomic status and ethnicity are sources pushing segregation together in the same direction. Many authors employ the term *intersectionality* to refer to this sort of "double disadvantage" scenario, where people from a poor minority are more segregated from the rest than they would be if they were only poor or from an ethnic minority (Crenshaw, 1990).

By contrast, a negative interaction term, as that for Biobio and Los Rios, reflects that socioeconomic status and ethnicity pull in opposite directions: their effects cancel each other out to a certain extent. This situation has been reported empirically previously (Guinea-Martin et al., 2015). However, it remains undertheorized in the literature on the multidimensional nature of social inequality and segregation.

We can perform the same analysis conditional on group variables using the option within:

```
> mutual(data = DT_Seg_Chile,
        group = c("csep", "ethnicity"),
        unit = c("school", "sch_type"),
        by = "region",
        within = "ethnicity",
        contribution.from = "unit_vars")
         region        M M_B_ethnicity  C_school   C_sch_type interaction
1:        Biobio 0.2312423    0.02582674 0.1053177 3.885868e-05  0.10005903
2: La Araucania 0.2367493    0.04840892 0.1276039 7.777505e-06  0.06072869
3:     Los Rios 0.2125013    0.03324738 0.1065172 7.811955e-05  0.07265861
```

In the above output, there are five terms for each region. The first term, M, matches the first term in the previous example. The second term, M_B_ethnicity, contains measures of ethnic segregation in schools and school types. The following two columns, (C_school and C_sch_type), are the contributions of schools and school types, respectively, to socioeconomic segregation after controlling for ethnic segregation. The last term, interaction, is the part of segregation that cannot exclusively be attributed to either schools or their ownership types.

The contribution.from option can also be used to display the contributions of a subset of variables. For example,

```
> mutual(data = DT_Seg_Chile,
        group = c("csep", "ethnicity"),
        unit = c("school", "sch_type"),
        by = "region",
        contribution.from = "csep")
         region        M    C_csep
1:        Biobio 0.2312423 0.2054156
2: La Araucania 0.2367493 0.1883404
3:     Los Rios 0.2125013 0.1792539
```

returns M and C_csep.

The contributions can also be displayed for organizational units. Take the following example:

```
> mutual(data = DT_Seg_Chile,
        group = c("csep", "ethnicity"),
        unit = c("school", "sch_type"),
        by = "region",
        contribution.from = "unit_vars")
         region        M  C_school   C_sch_type interaction
1:        Biobio 0.2312423 0.1293566 4.860549e-05  0.10183706
2: La Araucania 0.2367493 0.1709480 8.563946e-06  0.06579272
3:     Los Rios 0.2125013 0.1351602 1.903942e-04  0.07715072
```

The first column in the above output holds measures of total segregation, M, as before. The second column, C_school, contains the contributions of schools. The third column, C_sch_type, contains

the contribution of ownership type. The fourth column, interaction, is the part of socioeconomic and ethnic segregation that cannot exclusively be attributed to segregation in either schools or by ownership type. As previously stated, most schools in the sample period retain their ownership type, so each school can almost uniquely be classified by its sch_type. Hence, ownership is a negligible source of information.

The display of the contributions is simpler when variables are hierarchically related:

```
> mutual(data = DT_Seg_Chile,
         group = c("csep", "ethnicity"),
         unit = c("school", "district"),
         by = "region",
         contribution.from = "unit_vars")
        region         M  C_school C_district interaction
1:      Biobio 0.2311937 0.1558457          0  0.07534802
2: La Araucania 0.2367407 0.1635589         0  0.07318187
3:    Los Rios 0.2123109 0.1605696          0  0.05174127
```

The contribution of districts, C_district, is zero because each school is in only one district; consequently, there is no segregation by district within schools.

The analysis of contributions using option contributions.from can be generalized to situations with more than two sources of segregation on either the group or the unit dimension but not both (see the Statistical Model section). For example, in the following code, we consider three sources of group segregation as given by the variables csep, ethnicity, and gender:

```
> mutual(data = DT_Seg_Chile,
         group = c("csep", "ethnicity", "gender"),
         unit = c("school", "sch_type"),
         by = "region",
         contribution.from = "group_vars")
        region         M    C_csep C_ethnicity   C_gender interaction
1:      Biobio 0.2732082 0.2143819  0.03442763 0.04196586 -0.01756726
2: La Araucania 0.2718253 0.2017662  0.05742349 0.03507595 -0.02244036
3:    Los Rios 0.2838373 0.1942702  0.04659460 0.07133600 -0.02836349
```

In the output, there are columns for total segregation (M), the contributions of each of the three sources of group segregation (C_csep, C_ethnicity, and gender), and the interaction term. Total segregation increases when the students' gender is considered. It is the highest in Los Rios (0.2838373).

In the index decompositions, the "within" terms are weighted averages of local indices with demographic weights. To assess the relative importance of demographic weights versus local indices, we set the option components=TRUE.

```
> mutual(data = DT_Seg_Chile,
         group = c("csep", "ethnicity"),
         unit = c("school", "sch_type"),
         within = "csep",
         components = TRUE)
$Total
          M   M_B_csep    M_W_csep
1: 0.2610865 0.1995741 0.06151239


$W_Decomposition
                 csep         p      within
1: partially-subsidized 0.2668582 0.04907351
2:          subsidized 0.5866331 0.07331278
3:      non-subsidized 0.1465087 0.03691946
```

In the element labeled W_Decomposition, we obtain all the components of the linear combination that constitutes the "within" term. In particular, we obtain (i) the demographic weights (p) and (ii) the indexes of ethnic segregation in schools and ownership type (within) for each of the three socioeconomic groups (csep) in the data. The average of within, weighted by p, is the value of the "within" term displayed in the output labeled $Total, i.e., M_W_csep: 0.06151239. By inspecting the components of the within term, we conclude that students who obtain the full allowance drive the "within" term: they are both (i) the largest demographic group and (ii) the group where ethnic segregation in schools and ownership type is the highest.

4 Package structure

The **mutualinf** package requires **data.table** (Dowle and Srinivasan, 2021) to speed up data processing. The package provides two functions. The first, prepare_data, allows the user to convert a microdata file to a multidimensional frequency table of the data.table and mutual.data classes. The mutual function calculates and decomposes the M index. Figure 1 represents the package graphically, showing the relationships between the objects and functions of the package.
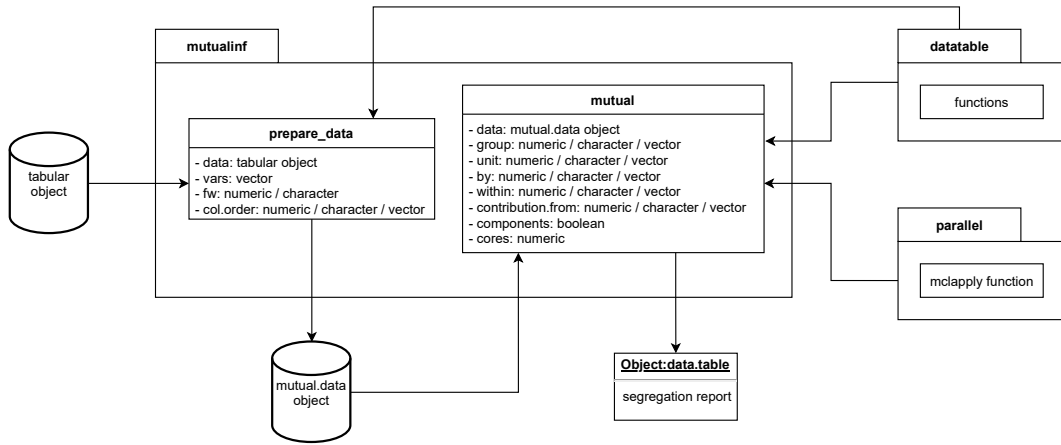

Figure 1: Package structure

Algorithm 1 mutual_within

Require: data,group,unit,within,by,i,$p_{k\bullet}$,*result*
  if $i == 1$ **then**
    result $\leftarrow \emptyset$
    Compute M for the first variable of within (the *between* term)
    Compute $p_{k\bullet}$
    Remove the first variable from within and append it to by
    $i \leftarrow i + 1$
    Append to *result* the *between* term
    Append to *result* mutual_within(data,group,unit,within,by,i,$p_{k\bullet}$,*result*) **return** *result*
  else if within $== \emptyset$ **then**
    $M_k \leftarrow \emptyset$
    for $k \in$ the Cartesian product of the variables in by **do**
      Append to M_k the M for k
    end for
    if i>2 **then**
      Subtract the dot product between $p_{k\bullet}$ and M_k from the last element in *result*
    end if
    Append to result the dot product between $p_{k\bullet}$ and M_k **return** *result*
  else
    $M_k \leftarrow \emptyset$
    for $k \in$ the Cartesian product of the variables in by **do**
      Append to M_k the M for k
    end for
    if i>2 **then**
      Subtract the dot product between $p_{k\bullet}$ and M_k from the last element in *result*
    end if
    $i \leftarrow i + 1$
    Append to *result* the dot product between $p_{k\bullet}$ and M_k
    Remove the first variable from within and append it to by
    Compute $p_{k\bullet}$
    Append to *result* mutual_within(data,group,unit,within,by,i,$p_{k\bullet}$,*result*) **return** *result*
  end if

Depending on its parameters, the mutual function carries out three different levels of analysis—basic, intermediate, and advanced—and outputs either indices alone or indices and their components.

At its most basic level, the algorithm computes the M index. In its first level of analysis, the `mutual` function computes the index using the variables that define groups and units established by the parameters `group` and `unit`, respectively. In its second level, `mutual` generates subsets (identified by the by parameter) on which it computes the index.

At the intermediate degree of complexity, `mutual` performs "between" and "within" decompositions of the index. In its first level, computes the decomposition with a single variable in the `within` parameter. If this variable belongs to the `unit` parameter, the function computes the decomposition that is shown in Equation (8) by applying the *SUD* property. Conversely, if this variable belongs to the `group` parameter, the function computes the decomposition that relies on the *SGD* property. In its second level, `mutual` generates subsets over the vector of variables defined in the by parameter. It then computes the index and its decompositions for each subset.

At its highest complexity, `mutual` allows for multiple decompositions. In its first level of analysis, it computes the index and decompositions for each element in the `within` parameter.In its second level, the function generates subsets using the by parameter and computes the index and its multiple decompositions for each subset.

In Algorithm 1, we illustrate the recursive calculation of all the terms in a decomposition by groups or units. This algorithm receives as inputs `data`, the dataset or multidimensional frequency table processed with `prepare_data`; group, the set of variables that identify the groups; unit, the set of variables that identify the units; `within`, the set of variables in which the index is decomposed; by, the set of variables that identify the subsets of the dataset; *i*, a control variable to identify if the decomposition is the first one performed; $p_{k\bullet}$, the demographic weights of the subsets obtained with by; and *result*, the variable that contains the output of the algorithm. See that *result* is both an input and an output variable due to the recursive implementation of the algorithm.

5 Conclusions

In this paper, we introduce the **mutualinf** package that implements a general approach for using the strong decomposability properties of the mutual information index in R. **mutualinf** exploits both recursion and parallelization techniques to facilitate the chained computation of "within" terms in complex decompositions of the index. We use Chilean primary school enrollment data to illustrate the usefulness and flexibility of the package. Of all the sources of segregation in schools considered, socioeconomic differences among students constitute the main source. The contributions to the overall segregation of ethnicity and gender are substantially lower.

6 Availability

The package is available in CRAN `https://cran.r-project.org/web/packages/mutualinf/`. The development version is available in GitHub `https://github.com/RafaelFuentealbaC/mutualinf`.

7 Acknowledgement

References

G. Akerlof and R. E. Kranton. *Identity economics how our identities shape our work, wages and well-being*. Princeton University Press, 2010. [p77]

K. Crenshaw. Mapping the margins: Intersectionality, identity politics, and violence against women of color. *Stan. L. Rev.*, 43:1241, 1990. [p84]

M. Dowle and A. Srinivasan. *data.table: Extension of 'data.frame'*, 2021. URL `https://CRAN.R-project.org/package=data.table`. R package version 1.14.0. [p86]

O. D. Duncan and B. Duncan. A methodological analysis of segregation indexes. *American Sociological Review*, 20(2):210, 1955. [p77]

B. Elbers. *segregation: Entropy-Based Segregation Indices*, 2021. URL https://CRAN.R-project.org/package=segregation. R package version 0.5.0. [p77]

Y. Flückiger and J. Silber. *The measurement of segregation in the labor force*. Physica-Verlag Heidelberg, 1999. [p77]

D. M. Frankel and O. Volij. Measuring school segregation. *Journal of Economic Theory*, 146(1):1–38, 2011. [p77, 78, 79]

R. Fuentealba-Chaura, R. Mora, and J. Rojas-Mora. *mutualinf: Computation and Decomposition of the Mutual Information Index*, 2021. URL https://CRAN.R-project.org/package=mutualinf. R package version 1.1.1. [p77]

D. Guinea-Martin and R. Mora. Computing decomposable multigroup indexes of segregation. Technical report, Universidad Carlos III de Madrid. Departamento de Economía, 2021. [p81]

D. Guinea-Martin, R. Mora, and J. Ruiz-Castillo. The joint effect of ethnicity and gender on occupational segregation. an approach based on the mutual information index. *Social Science Research*, 49:167–178, 2015. [p84]

Instituto Nacional de Estadísticas. Radiografía de genero: Pueblos originarios en Chile 2017, 2018. [p80]

S. Kullback. *Information Theory and Statistics*. Wiley Publication in Mathematical Statistics, 1959. [p78]

D. S. Massey and N. A. Denton. The dimensions of residential segregation. *Social Forces*, 67(2):281–315, 1988. [p78]

R. Mora and J. Ruiz-Castillo. Additively decomposable segregation indexes. the case of gender segregation by occupations and human capital levels in Spain. *The Journal of Economic Inequality*, 1 (2):147–179, 2003. [p79]

R. Mora and J. Ruiz-Castillo. Entropy-based segregation indices. *Sociological Methodology*, 41(1):159–194, 2011. [p79]

S. F. Reardon and G. Firebaugh. Measures of multigroup segregation. *Sociological Methodology*, 32(1): 33–67, 2002. [p77]

H. Theil and A. J. Finizza. A note on the measurement of racial integration of schools by means of informational concepts. *The Journal of Mathematical Sociology*, 1(2):187–193, 1971. [p77]

B. S. Zoloth. *An investigation of alternative measures of school segregation*. University of Wisconsin, 1974. [p78]

Rafael Fuentealba-Chaura
School of Computer Science
Universidad Católica de Temuco
Temuco, Chile

Daniel Guinea-Martin
Department of Sociology
Universidad de Málaga (UMA)
Málaga, Spain

Ricardo Mora
Department of Economics
Universidad Carlos III de Madrid
Getafe, Spain

Julio Rojas-Mora
Department of Computer Science
Universidad Católica de Temuco
Temuco, Chile

A Workflow for Estimating and Visualising Excess Mortality During the COVID-19 Pandemic

by Garyfallos Konstantinoudis, Virgilio Gómez-Rubio, Michela Cameletti, Monica Pirani, Gianluca Baio, and Marta Blangiardo

Abstract COVID-19 related deaths estimates underestimate the pandemic burden on mortality because they suffer from completeness and accuracy issues. Excess mortality is a popular alternative, as it compares the observed number of deaths versus the number that would be expected if the pandemic did not occur. The expected number of deaths depends on population trends, temperature, and spatio-temporal patterns. In addition to this, high geographical resolution is required to examine within country trends and the effectiveness of the different public health policies. In this tutorial, we propose a workflow using R for estimating and visualising excess mortality at high geographical resolution. We show a case study estimating excess deaths during 2020 in Italy. The proposed workflow is fast to implement and allows for combining different models and presenting aggregated results based on factors such as age, sex, and spatial location. This makes it a particularly powerful and appealing workflow for online monitoring of the pandemic burden and timely policy making.

1 Introduction

Estimating the burden of the effect of the COVID-19 pandemic on mortality is an important challenge (Weinberger et al., 2020). The estimates of COVID-19 related deaths are subject to testing capacity and changes in definition and reporting guidelines, raising accuracy and completeness considerations (Aburto et al., 2021; Konstantinoudis et al., 2022). In addition, the estimates of COVID-19-related deaths give an incomplete picture of the mortality burden of the COVID-19 pandemic, as they do not account for the indirect pandemic effects due to, for instance, disruption to health services (Kaczorowski and Del Grande, 2021). Alternatively, excess mortality has been extensively used to evaluate the impact of the COVID-19 pandemic on mortality (Rossen et al., 2020; Islam et al., 2021; Kontis et al., 2020; Konstantinoudis et al., 2022; Verbeeck et al., 2021).

Excess mortality is estimated by comparing the observed number of deaths in a particular time period with the expected number of deaths under the counterfactual scenario of the event (pandemic) not having had occurred. Typically, this counterfactual scenario is calculated using a comparison period, for instance, several previous years (https://www.euromomo.eu/). Calculating the expected number of deaths accurately requires accounting for factors such as population trends, seasonality, temperature, public holidays and spatio-temporal dependencies. While accounting for the above-mentioned factors, most studies to date have estimated excess mortality at the national level (see Rossen et al., 2020; Weinberger et al., 2020), and some have examined excess mortality across countries (Islam et al., 2021; Kontis et al., 2020, 2021).

While national-level estimates of excess mortality give valuable insights into the total number of excess deaths, they do not allow for evaluation of within-country geographical differences. However, such information is essential to understand the country's transmission patterns and effectiveness of local policies and measures to contain the pandemic (Kontopantelis et al., 2021). Temporal trends in excess mortality can substantially differ across regions of the same country (Blangiardo et al., 2020), which makes national-based comparisons even more challenging. Therefore, understanding the effect of the COVID-19 pandemic on mortality requires higher spatial resolution and models that account for spatial, temporal and spatio-temporal dependencies.

When working at high spatio-temporal resolution, data are generally sparse, and this leads to highly variable excess mortality estimates. This is aggravated by the fact that excess deaths tend to be subject to spatial and temporal correlation. This makes it essential to use statistical methods that can account for these dependencies and provide robust and accurate estimates. The disease mapping framework, which is commonly employed in epidemiology to study the spatio-temporal variation of diseases (Waller et al., 1997; Moraga, 2018), can be exploited to estimate excess mortality at subnational and weekly level. The Bayesian approach is naturally suited in this context, as it is able to model complex dependency structures, as well as to incorporate uncertainty in the data generating and modelling process. In addition, while fully propagating the uncertainty, it allows for summaries of results at any desired level of spatio-temporal aggregation (using for instance coarser geographical units suitable for policy implementation). This in combination with fast approximate methods to inference, such as the Integrated Laplace Approximation (INLA; Rue et al., 2009), make this framework
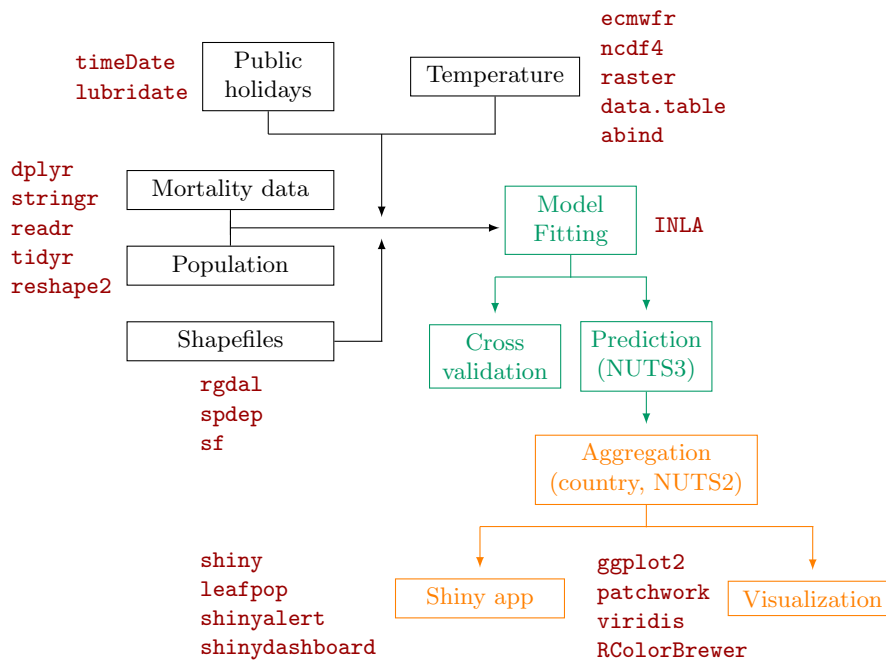
Figure 1: Diagram of the workflow: data wrangling in black, analysis in green, post-processing in orange and packages in red (timeDate; Wuertz et al., 2023, lubridate; Grolemund and Wickham, 2011, dplyr; Wickham et al., 2023a, stringr; Wickham, 2022, readr; Wickham et al., 2022, tidyr; Wickham et al., 2023b, reshape2; Wickham, 2007, rgdal; Bivand et al., 2023, spdep; Bivand, 2022, sf; Pebesma, 2018, shiny; Chang et al., 2022, leafpop; Appelhans and Detsch, 2021, shinyalert; Attali and Edwards, 2021, shinydashboard; Chang and Borges Ribeiro, 2021, ecwmfr; Hufkens et al., 2019, ncdf4; Pierce, 2023, raster; Hijmans (2023), data.table; Dowle and Srinivasan, 2022, abind; Plate and Heiberger, 2016, ggplot2; Wickham, 2016, patchwork; Pedersen, 2022, viridis; Garnier et al., 2021, RColorBrewer; Neuwirth, 2022). NUTS stands for Nomenclature of Territorial Units for Statistics with NUTS3 being the highest spatial resolution available and NUTS2 coarser but appropriate for policy making.

particularly powerful and appealing for the monitoring of the pandemic burden and timely policy making.

Here, we describe how to run a study on excess mortality at high spatial and temporal resolution using a Bayesian approach and R. This tutorial provides a detailed explanation of the modelling approach used previously to quantify excess mortality in 5 European regions (Konstantinoudis et al., 2022). We have further modified the way we model long-term trends in Konstantinoudis et al. (2022), as we showed that it provides more accurate predictions (Riou et al., 2023). Figure 1 illustrates the workflow followed in this paper together with the main R packages used. Source code for replicating the data wrangling (R scripts prefixed with 01, 02 or 03), analysis (R scripts prefixed with 04) and post-processing steps (R scripts prefixed with 05 or 06 and the App folder) and data files are available from GitHub at https://github.com/gkonstantinoudis/TutorialExcess.

This tutorial is structured as follows: we first describe the modelling framework and present the case study in Italy. We then show how to run and evaluate the model, and extract and visualise the results. Finally, we present an R-shiny app which makes the results effectively and easily disseminated.

2 Bayesian hierarchical spatio-temporal model to estimate excess mortality

We propose a Bayesian hierarchical model to quantify the effect of spatio-temporal location on excess mortality under extreme events such as the COVID-19 pandemic, stratified by specific age-sex population groups. To do so, we first describe the statistical model for predicting the number of deaths from all-causes based on historical data, in the counterfactual scenario in which the pandemic did not take place. Then, we show how to estimate the magnitude of excess deaths over a specific period of time, with associated uncertainty, by comparing the predicted versus the actual number of deaths.

Let y_{jstk} and P_{jstk} be the number of all-cause deaths and the population at risk for the j-th week, $j = 1, \ldots, J_t$, where J_t is the total number of weeks of the year t ($t = 2015, \ldots 2019$), the s-th spatial

unit ($s = 1, \ldots S$, where S is the number of provinces in Italy), and k-th age-sex group ($k = 1, \ldots 10$). Also let x_{1jt}, x_{2t} and z_{jst} denote the adjustment covariates, respectively: public holidays ($x_{1jt} = 1$ if week j of the t-th year contains a public holiday and 0 otherwise), the year that the j-the week belongs to, and temperature. We assume that the historical observed number of deaths, conditional on the risk r_{jstk}, follows a Poisson distribution, with a log-linear model for the risk. To simplify notation, we omit k, although the following model was fitted to all k age-sex groups (with the elements of the linear predictor also depending on k):

$$
\begin{aligned}
y_{jst}|r_{jst} &\sim \text{Poisson}(\mu_{jst} = r_{jst}P_{jst}), \\
\log\left(r_{jst}\right) &= \beta_{0jt} + \beta_1 x_{1jt} + \beta_2 x_{2t} + f(z_{jst}) + b_s + w_{jt}.
\end{aligned}
\tag{1}
$$

Here, β_{0jt} is the week-specific intercept in year t given by $\beta_{0jt} = \beta_0 + \epsilon_{jt}$ for the k-th age-sex group, where β_0 is the global intercept and ϵ_{jt} is an unstructured random effect representing the deviation of each week from the global intercept, which is modelled using independent and identically (iid) distributed Gaussian prior distribution with zero-mean and variance equal to τ_ϵ^{-1}. The parameters β_1 and β_2 are unknown regression coefficients associated to the public holidays covariate x_{1jt} and a long term linear trend. The effect of temperature, $f(\cdot)$, is allowed to be non-linear by specifying a second-order random walk (RW2) model:

$$
z_{jst} \mid z_{(j-1)st}, z_{(j-2)st}, \tau_z \sim \text{Normal}\left(2z_{(j-1)st} + z_{(j-2)st}, \tau_z^{-1}\right),
$$

where τ_z^{-1} is the variance.

Terms b_s and w_j are spatial and temporal random effects, respectively. We specify the spatial random effect term, b_s, with a convolution prior (Besag et al., 1991), and the temporal random effect term, w_j with a non-stationary in time prior. In detail, we model $\boldsymbol{b}$ using a reparameterisation of the popular Besag-York-Mollié prior, which is a convolution of an intrinsic conditional autoregressive (CAR) model and an iid Gaussian model. Let u_s be the spatially structured component defined by an intrinsic CAR (Besag, 1974) prior $u_s|u_i, i \in \partial_s \sim (\bar{u}, \tau_u^{-1}/m_s)$, where $\bar{u}$ is the mean of the neighbours and ∂_s and m_s are respectively the set and the number of neighbours of area s, τ_u^{-1} the conditional variance, and v_s the unstructured component with prior $v_s \overset{iid}{\sim} \text{Normal}(0, \tau_v^{-1})$. We model b_s as follows (Besag et al., 1991; Riebler et al., 2016; Konstantinoudis et al., 2020):

$$
b_s = \frac{1}{\sqrt{\tau_b}}\left(\sqrt{1-\phi}v_s^\star + \sqrt{\phi}u_s^\star\right)
$$

where $u_s^\star$ and $v_s^\star$ are standardised versions of u_s and v_s such that their variance is equal to 1 (Simpson et al., 2017). The term $0 \leq \phi \leq 1$ is a mixing parameter, which measures the proportion of the marginal variance explained by the structured effect. Finally, we assign to the temporal random effect term, w_{jt}, a Gaussian random walk model of order 1 (RW1). This component captures seasonality and is specified as:

$$
w_{jt} \mid w_{(j-1)t}, \tau_w \sim \text{Normal}(w_{(j-1)t}, \tau_w^{-1}).
$$

The Bayesian representation of the above model is completed once we select priors for the fixed effects β_0 and $\boldsymbol{\beta}$ and the hyperparameters: $\tau_\epsilon, \tau_z, \tau_b, \tau_w$, and ϕ. For the fixed effects we selected minimally informative Normal distributions, whereas for the hyperparameters we specified "penalising complexity" (PC) priors (Simpson et al., 2017). PC priors are defined by penalising deviations from a "base" model (e.g., specified in terms of a specific value of the hyperparameters) and have the effect of regularising inference, while not implying too strong prior information. Technically, PC priors imply an exponential distribution on a function of the Kullback–Leibler divergence between the base model and an alternative model in which the relevant parameter is unrestricted. This translates to a suitable "minimally informative", regularising prior on the natural scale of the parameter.

In order to quantify the weekly excess mortality at sub-national level for specific age-sex population groups, we need to predict the number of deaths that would be expected if the COVID-19 pandemic had not occurred. In Bayesian analysis, this can be performed by drawing random samples from the posterior predictive distribution (that is, the distribution of unobserved values conditional on the observed values from previous years). Specifically, letting $\boldsymbol{\theta}$ be the model parameters, $\mathcal{D}$ be the observed data, and y_{jst^*} be the count of deaths that we want to predict, we have:

$$
p(y_{jst^*} \mid \mathcal{D}) = \int p(y_{jst^*} \mid \boldsymbol{\theta})p(\boldsymbol{\theta} \mid \mathcal{D})d\boldsymbol{\theta}.
\tag{2}
$$

Operationally, we first generate random samples from the joint posterior marginal of the parame-

ters specified in Equation (1) at the highest spatial resolution available (NUTS3 regions; Nomenclature of Territorial Units for Statistics 3 regions, https://ec.europa.eu/eurostat/web/nuts/background/). Following that, we use the samples to compute the linear predictor, compute the mean parameter of the Poisson distribution via inverse link function, and obtain the predicted number of deaths, which represents the baseline number of deaths assuming the pandemic did not take place.

Finally, to estimate the excess deaths, the predicted number of deaths is compared against the actual observed number of deaths. Further, this allows us to compute the relative change in mortality (relative to what we would expect if the pandemic did not occur). This is obtained by (i) subtracting the predicted number of deaths from the observed number of deaths in each time point j in the t^*-th year and spatial unit s (number of excess deaths or NED), and (ii) dividing NED by the predicted number of deaths for each sample and multiplying by 100 (yielding % relative excess mortality or REM).

The model estimates are computed using Integrated Nested Laplace Approximation (INLA; Rue et al., 2009, which performs approximate Bayesian inference on the class of latent Gaussian models (Rue and Held, 2005). Unlike simulation based Markov chain Monte Carlo method, INLA is a deterministic algorithm, which employs analytical approximations and efficient numerical integration schemes to provide accurate approximations of the posterior distributions in short computing times. The INLA software is provided through the R package INLA, which can be downloaded from https://www.r-inla.org/.

3   Motivating example: Italy

3.1 Outcome data

We retrieved all-cause mortality data during 2015-2020 in Italy from the Italian National Institute of Statistics (https://www.istat.it/). Data were available weekly (ISO week), by age (5-year age groups), sex and NUTS3 regions. As the COVID-19 mortality rates increase with age, we aggregated mortality counts based on the following age groups: <40, 40-59, 60-69, 70-79 and 80 years and older (Davies et al., 2021).

3.2 Population data

Population data in Italy during 2015-2020 were retrieved from the Italian National Institute of Statistics. The data represent the population in Italy on first of January of every year stratified by age (5-year age groups), sex and NUTS3 regions. To retrieve weekly population, we performed linear interpolation by the selected age groups (<40, 40-59, 60-69, 70-79 and 80+), sex, and NUTS3 regions using populations on the first of January of the current and next year. Population counts on the first of January 2021, which takes COVID-19 deaths in 2020 into consideration, were available at the time of analysis. Our goal was, however, to predict mortality for 2020, as if the pandemic had not occurred. Thus we performed an additional linear interpolation by age, sex and NUTS3 regions to predict the population at January 1st 2021, using the years 2015-2020 (Figure 2, panel A). Object pop is a tibble containing the NUTS3 region ID (NUTS318CD), age group (ageg), sex (sex), year (year) and population counts (population):

```
pop
# A tibble: 6,420 x 5
# Groups:   ageg, sex [10]
   NUTS318CD ageg    sex      year population
   <chr>     <fct>   <chr>   <dbl>      <dbl>
 1 TO        less40  female   2015     435758
 2 TO        less40  female   2016     427702
 3 TO        less40  female   2017     420498
 4 TO        less40  female   2018     413141
 5 TO        less40  female   2019     406937
 6 TO        less40  female   2020     402768
 7 TO        less40  male     2015     449605
 8 TO        less40  male     2016     443941
 9 TO        less40  male     2017     439522
10 TO        less40  male     2018     433365
# ... with 6,410 more rows
```

We can use the following code (based on `tidyverse` and "piping" principles) to calculate the population of the 1st of January 2021 by NUTS3 regions, sex and age:

```
pop %>% group_by(NUTS3, sex, age) %>%
        summarise(pop = as.vector(coef(lm(pop ~ year)) %*% c(1, 2021))) %>%
        mutate(year = 2021) -> pop2021
```

We acknowledge that the linear trend in the population is a rather simplistic assumption. In subsequent analyses in Switzerland, we proposed a spatio-temporal approach similar with equation (1) to model the population counts had the pandemic not occurred (Riou et al., 2023). The code for that analysis is also online available online (https://github.com/jriou/covid19_ascertain_deaths). Once we obtained the year 2021 we performed an additional linear interpolation to calculate weekly number of population as shown on Figure 2, panel B.

3.3 Covariates

We used covariates related to ambient temperature, national holidays, and year of death, to improve the model's predictions. Data on air temperature during 2015-2020 in Italy at 2m above the surface of land were retrieved from the ERA5 reanalysis data set of the Copernicus climate change program
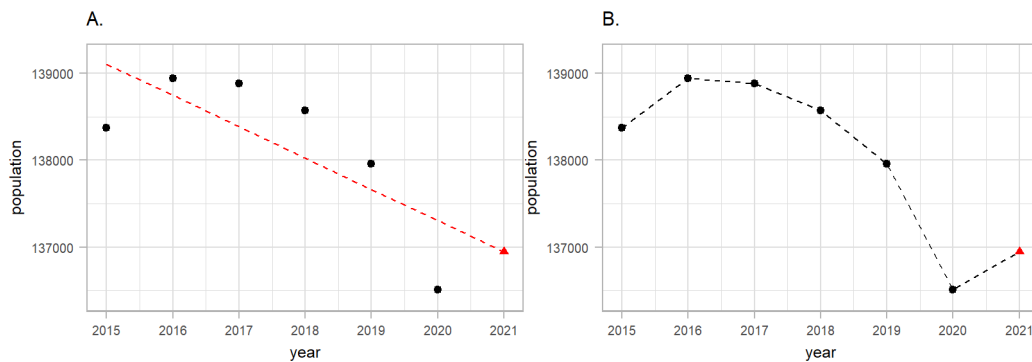
Figure 2: A schematic representation of the weekly population estimation procedure focusing on females aged 40-49 in Venice during 2015-2019 as an example. On panel A we show how we used the historical data (black points) and fit a linear regression (red dashed line) to predict 2021 (red triangle). On the panel B we show how we predicted weekly population by drawing lines between the years.
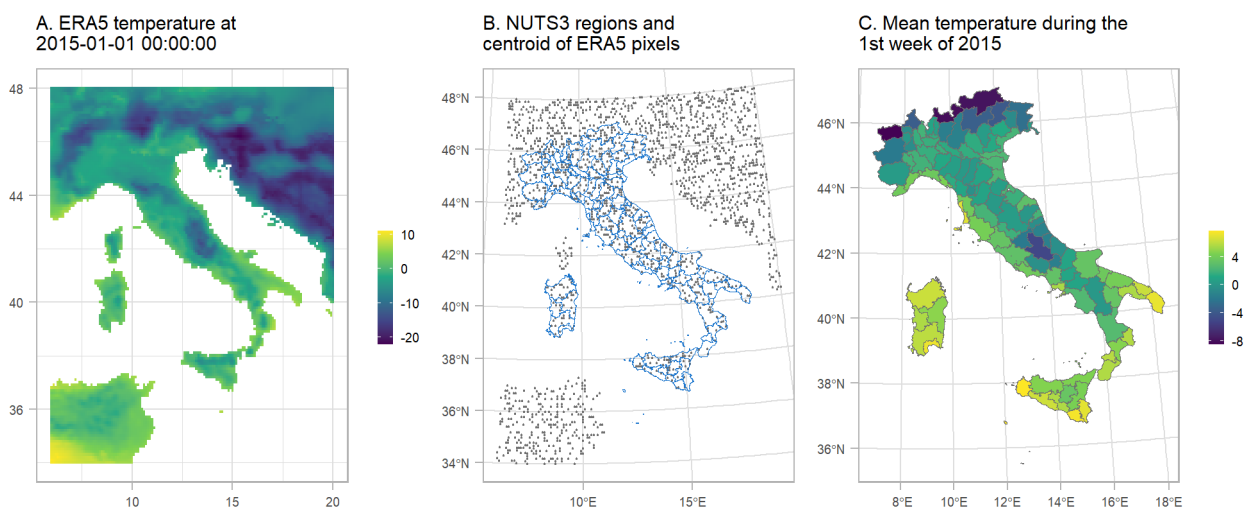
Figure 3: Schematic representation of the temperature misalignment procedure. A) The temperature obtained by ERA5 at 2015-01-01 00:00:00. B) NUTS3 regions in blue and a sample of the centroids of the pixels from the ERA5 raster. C. Mean temperature per NUTS3 region during the 1st week of 2015.

(Hersbach et al., 2020). The geographical resolution of the ERA5 estimates is $0.25° \times 0.25°$ (panel A of Figure 3). We calculated the weekly mean by the centroids of the $0.25° \times 0.25°$ grid (panel B of Figure 3) and then averaged the weekly temperature over the ERA5 centroids that overlay with the NUTS3 regions (panels B and C of Figure 3).

3.4 Fitting the model

The modelling process in INLA consists of three main steps: (1) the selection of priors, (2) definition of the model "formula" (which sets out the expression for the generalised linear predictor), and (3) the call to the main function inla, which computes the estimates.

In particular, we constructed the PC priors for $\sigma_\epsilon = \sqrt{1/\tau_\epsilon}, \sigma_z = \sqrt{1/\tau_z}, \sigma_b = \sqrt{1/\tau_b}$ and $\sigma_w = \sqrt{1/\tau_w}$ based on the assumption that it is unlikely to have a relative risk higher than $\exp(2)$ based solely on spatial, yearly and seasonal variation, Figure 4, panel A. For the mixing parameter ϕ, we set $\Pr(\phi < 0.5) = 0.5$ reflecting our lack of knowledge about whether overdispersion or strong spatial autocorrelation should dominate the field b, Figure 4, panel B.

These assumptions can be encoded using the following code:

```
# Defines the priors
hyper.bym <- list(
        theta1 = list('PCprior', c(1, 0.01)),
        theta2 = list('PCprior', c(0.5, 0.5))
)
hyper.iid <- list(theta = list(prior="pc.prec", param=c(1, 0.01)))

# Defines the model "formula"
formula =
        deaths ~ 1 + offset(log(population)) + hol + id.year +
        f(id.tmp, model = 'rw2', hyper = hyper.iid, constr = TRUE, scale.model = TRUE) +
        f(id.wkes, model = 'iid', hyper = hyper.iid, constr = TRUE) +
        f(id.time, model = 'rw1', hyper = hyper.iid, constr = TRUE, scale.model = TRUE,
         cyclic = TRUE) +
        f(id.space, model = 'bym2', graph = "W.adj", scale.model = TRUE, constr = TRUE,
        hyper = hyper.bym)

control.family = inla.set.control.family.default()

# Calls INLA to fit the model
inla.mod = inla(formula,
        data = dat,
        family = "Poisson",
        verbose = TRUE,
        control.family = control.family,
        control.compute = list(config = TRUE),
        control.mode = list(restart = T),
        num.threads = round(parallel::detectCores()*.8),
        control.predictor = list(link = 1))
```

After fitting the model, we take 1000 samples from the (approximated) posterior distribution of the linear predictor and we use each drawn sample as the mean of a Poisson distribution to retrieve the predicted mortality counts:

```
post.samples <- inla.posterior.sample(n = 1000, result = inla.mod)
predlist <- do.call(cbind, lapply(post.samples, function(X)
exp(X$latent[startsWith(rownames(X$latent), "Pred")])))

pois.samples <- apply(predlist, 2, function(Z) rpois(n = length(Z), lambda = Z))
```

This allows us to estimate the entire predictive posterior distribution of the mortality counts, incorporating both the sampling and the linear predictor uncertainty.

3.5 Model validation

To examine model validity, we performed a cross validation, leaving out one historical year at a time and predicting the weekly number of deaths by NUTS3 regions for the year left out. As part
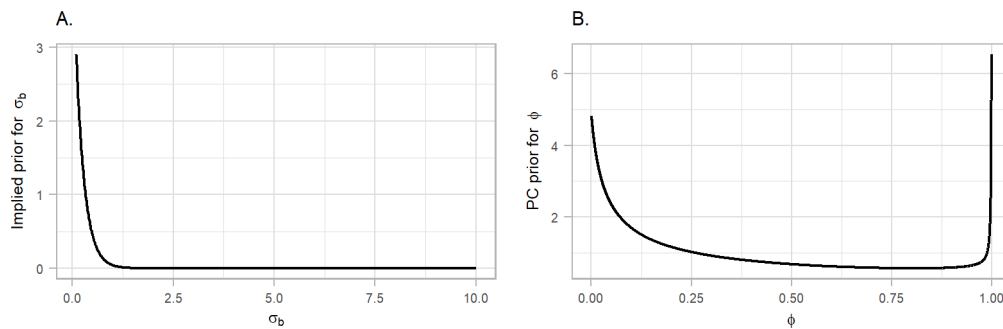
Figure 4: Penalised complexity (PC) priors for the hyperparameters of the spatial field. A) The implied PC prior for the standard deviation (as the original scale of the prior is the precision). B) The PC prior for the mixing parameter ϕ.

of this validation we used future years to train predictions for past, but in a previous study we also showed that the model selected was performing well when other types of cross-validations were used (Riou et al., 2023). For each stratum, we calculated the correlation between observed and fitted and a coverage probability, i.e. the probability that the observed death fall into the 95% credible interval (95% CrI) of the predicted.

4 Results

4.1 Cross-validation

Overall, we found that the model performed well in predicting the expected number of deaths. The correlation between true and expected number of deaths varied from 0.39 (95% CrI: 0.37, 0.40) in females 40< to 0.95 (95% CrI: 0.94, 0.95) in females 80> and coverage probability from 0.92 in females 40< to 0.96 in males 60-69, 70-79:

```
#              Correlation Coverage
# less40F 0.39 (0.37, 0.40)    0.92
# 40-59F  0.78 (0.77, 0.78)    0.95
# 60-69F  0.83 (0.82, 0.83)    0.95
# 70-79F  0.91 (0.90, 0.91)    0.95
# 80plusF 0.95 (0.94, 0.95)    0.95
# less40M 0.51 (0.50, 0.52)    0.93
# 40-59M  0.83 (0.83, 0.84)    0.95
# 60-69M  0.87 (0.86, 0.87)    0.96
# 70-79M  0.92 (0.91, 0.92)    0.96
# 80plusM 0.94 (0.94, 0.94)    0.95
```

4.2 Expected number of deaths

The object `pois.samples.list` contains 1000 samples from the posterior predictive distribution (2), i.e. 1000 samples of the expected number of deaths by age, sex, NUTS3 regions and week, had the pandemic not occurred. We can access the different age-sex groups as follows:

```
names(pois.samples.list)
# "F_less40" "F_40_59"  "F_60_69"  "F_70_79"  "F_80plus"
# "M_less40" "M_40_59"  "M_60_69"  "M_70_79"  "M_80plus"
```

where F stands for females and M for males across the different age groups. To get an idea about the structure of the data, we can use the `head()` function for females 60-69 and check the first 10 samples (V1 through to V10) of excess deaths for the first 6 weeks of 2020 in the 001 region (Torino)

```
pois.samples.list$F_60_69 %>%
select(paste0("V", 1:10), EURO_LABEL, ID_space, year) %>%
head()
#     V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 EURO_LABEL ID_space year
```

```
# 262 22 18 17 15 17 18 13 19 16  17   2020-W01     001 2020
# 263 18 16 21 20 16 23 13 12 23  17   2020-W02     001 2020
# 264 17 23 20 26 12 16 12 19 17  13   2020-W03     001 2020
# 265 13  8 13 30 22 17 16 22 17  19   2020-W04     001 2020
# 266 14 20 15 15 19 18 23 14 12  18   2020-W05     001 2020
# 267 17 14 14  9 14 15 17 19 19  14   2020-W06     001 2020
```

We can also calculate median and 95% CrI expected number of deaths for this specific age-sex group across all years by NUTS3 region:

```
pois.samples.list$F_60_69 %>%
select(starts_with("V"), "ID_space") %>%
        group_by(ID_space) %>%
        summarise_all(sum) %>%
        rowwise(ID_space) %>%
        mutate(median = median(c_across(V1:V1000)),
LL = quantile(c_across(V1:V1000), probs= 0.025),
UL = quantile(c_across(V1:V1000), probs= 0.975)) %>%
        select(ID_space, median, LL, UL) %>%
        head()
```

```
# A tibble: 107 × 4
# Rowwise:   ID_space
# ID_space median    LL    UL
# <chr>       <dbl> <dbl> <dbl>
# 1 001          814  748  885.
# 2 002           72   55   90
# 3 003          139  116  167
# 4 004          212  183. 243.
# 5 005           86   67  107
# 6 006          178  150  208
# 7 007           46   33   62
# 8 008           86   68  107
# 9 009          113   93  135.
# 10 010         341  301.  380
# ... with 97 more rows
```

4.3 Excess mortality

The above results can be combined in different ways using the functions get2020data() and get2020weeklydata() to calculate excess mortality (in the R script functions.R). The function get2020data() aggregates over the entire country, NUTS2 regions, sex, age and time resulting in the object d, whereas the function get2020weeklydata() aggregates over the entire country, NUTS2 regions, sex and age but not over time resulting in the object d_week.

```
names(d)
# "province" "region"    "country"
names(d$province)
# "none"    "age"      "sex"      "agesex"
names(d$province$age)
# "40<"    "40-59" "60-69" "70-79" "80+"
names(d$province$sex)
# "F" "M"
names(d$province$agesex)
# "F40<"    "F40-59" "F60-69" "F70-79" "F80+"    "M40<"    "M40-59" "M60-69" "M70-79" "M80+"
```

Province stands for the NUTS3 regions (the resolution we used to fit the models), region for the NUTS2 (coarser than NUTS3, appropriate for policy making) and country for the nationwide aggregation. Within these aggregations users can select the option "none" being the total aggregation by age and sex, "age" by sex, "sex" by age and "agesex" refers to no age and sex aggregation. The objects d and d_week have similar structure and contain summary statistics for REM and NED and posterior probabilities of a positive REM or NED:

```
head(d$province$none)
# Simple feature collection with 6 features and 24 fields
```

```
# Geometry type: POLYGON
# Dimension:     XY
# Bounding box:  xmin: 6.626865 ymin: 44.06028 xmax: 9.21355 ymax: 45.95041
# CRS:           +proj=longlat +datum=WGS84
# ID_space SIGLA    DEN_UTS observed population mean.REM median.REM    sd.REM
# 1      001   TO     Torino    32478 2226317.2 21.37414    21.36545 1.116582
# 2      002   VC   Vercelli     3216  168761.7 32.27082    32.15534 3.133142
# 3      003   NO     Novara     5274  364529.4 23.62994    23.71569 2.220801
# 4      004   CN      Cuneo     8716  585479.3 20.16665    20.22069 1.742638
# 5      005   AT       Asti     3745  211437.1 26.37080    26.30691 2.660622
# 6      006   AL Alessandria    7916  416090.6 28.27706    28.21510 2.052881
# LL.REM    UL.REM exceedance.REM median.REM.cat exceedance.REM.cat median.pred
# 1 19.19396 23.50963             1          20%>         (0.95, 1]       26761
# 2 26.51331 38.74179             1          20%>         (0.95, 1]        2434
# 3 19.18442 27.97865             1          20%>         (0.95, 1]        4263
# 4 16.67966 23.54490             1          20%>         (0.95, 1]        7250
# 5 21.19545 31.77340             1          20%>         (0.95, 1]        2965
# 6 24.48449 32.33309             1          20%>         (0.95, 1]        6174
# LL.pred UL.pred mean.NED median.NED    sd.NED    LL.NED    UL.NED exceedance.NED
# 1   26296   27249 5717.153     5717.5 246.42447 5229.975 6182.075              1
# 2    2318    2543  783.265      782.5  57.49419  673.975  898.025              1
# 3    4121    4428 1006.667     1011.0  76.69947  848.925 1153.000              1
# 4    7055    7471 1461.215     1466.0 105.25269 1245.975 1661.075              1
# 5    2842    3092  780.189      780.0  62.31281  654.950  903.000              1
# 6    5982    6360 1743.404     1742.0  98.74248 1556.975 1934.125              1
# median.NED.cat exceedance.NED.cat                        geometry
# 1          1000>         (0.95, 1] POLYGON ((7.859044 45.59758...
# 2     [500, 1000)         (0.95, 1] POLYGON ((8.204465 45.93567...
# 3          1000>         (0.95, 1] POLYGON ((8.496878 45.83934...
# 4          1000>         (0.95, 1] POLYGON ((7.990897 44.82381...
# 5     [500, 1000)         (0.95, 1] POLYGON ((8.046805 45.12815...
# 6          1000>         (0.95, 1] POLYGON ((8.405489 45.20148...
```

Notice that the object d$province$none is a simple feature collection, making mapping it straightforward using the ggplot2 package and geom_sf function. In Figure 5 (plots 1A, 2A and 3A) we show the median posterior of REM for total age and sex at the national, NUTS2 and NUTS3 regional level. For these plots we used the object d with the selection "none" and plot the median REM (median.REM), for example for 3A:

```
# prov <- "Foggia"
# d$province$none %>%
#       filter(DEN_UTS == prov) %>%
#       select(geometry) %>%
#       ggplot() +
#             geom_sf(data = d$province$none, aes(fill = median.REM.cat)) +
#             geom_sf(fill = NA, col = col.highlight, size = .8) +
#             scale_fill_manual(values=colors, name = "", drop = FALSE) +
#       theme_light() + ggtitle(paste0("3A. NUTS3 regions: ", prov))
```

Overall, the REM in Italy during 2020 was between 15-20%, meaning that 15-20% more people died that year than how many would be expected based on historical data, see Figure 5, panel 1A. When the higher geographical resolution is assessed, it was revealed that north and in particular Lombardia was the region hit the worst, with the REM exceeding 20%, see Figure 5, panels 1B and 1C. Figure 6 shows a measure of uncertainty of the REM, now for the different age groups and both sexes (selection "age"). The probability of a positive excess (exceedance.REM) in older people was larger than 0.95 almost everywhere, see Figure 6.

Panels 1B and 1C of Figure 5 show the median temporal nationwide excess mortality together with 95% CrI by sex after aggregating the different age groups (using d_week and the "sex" selection). We observe a clear first pandemic wave during March and May and a second one during mid October and December in 2020. During the first pandemic wave, there were weeks when the median REM reached almost 100% in males, Figure 5.
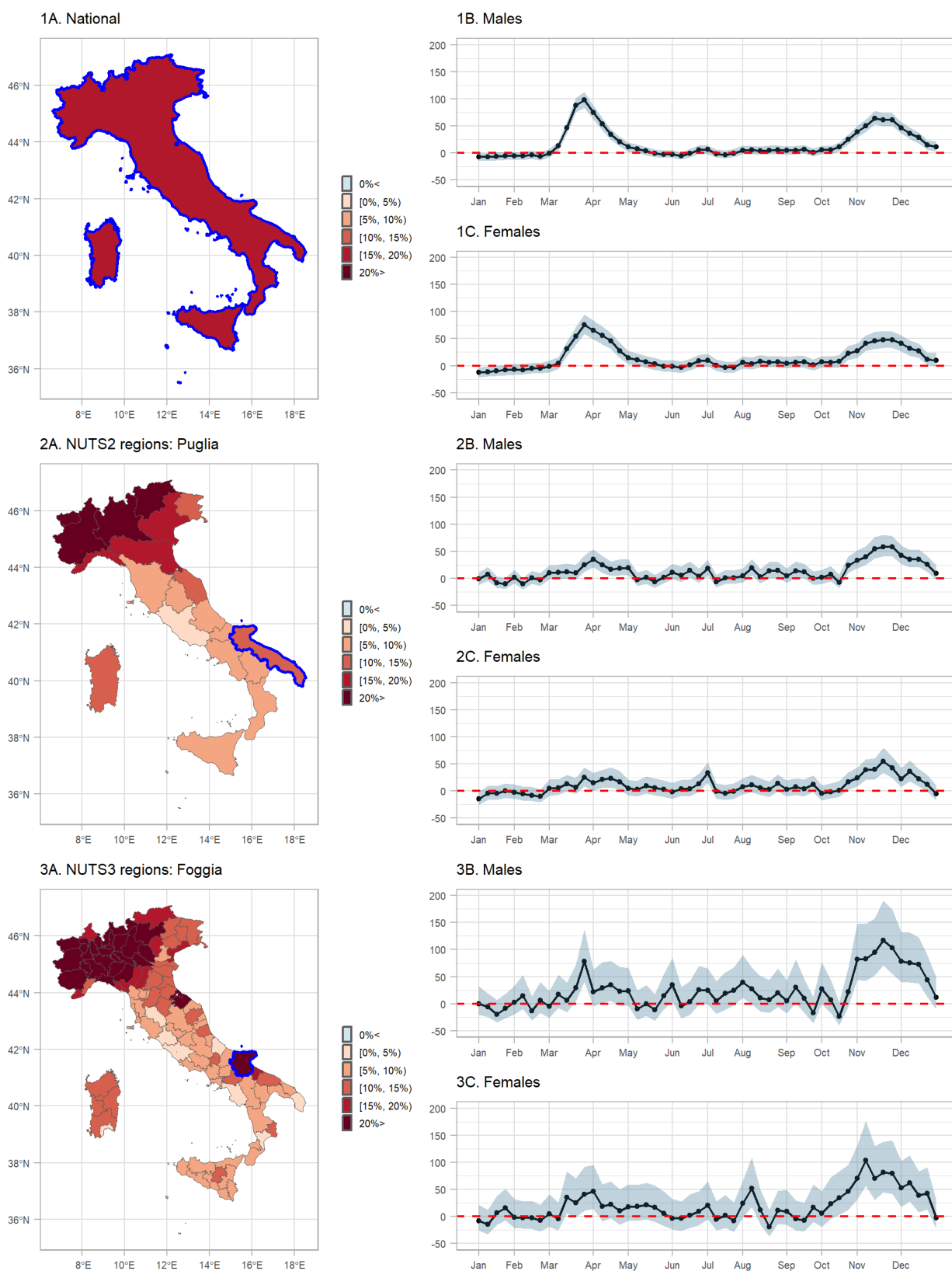
Figure 5: Median relative excess mortality by spatial region during 2020: 1A) nationwide, 2A) NUTS2 and 3A) NUTS3 level, and weekly median relative excess mortality and 95% credible intervals (95% probability that the true value lies within this interval) by sex during 2020 in: 1B) males nationwide, 1C) females nationwide, 2B) males in Puglia, 2C) females in Puglia, 3B) males in Foggia and 3C) females in Foggia.

Panels 2B, 2C, 3B and 3C of Figure 5 show the median spatio-temporal excess mortality together with 95% CrI by sex after aggregating over the different age groups. Panels 2B and 2C highlight the region of Puglia, where during the first wave of the pandemic experienced increases excess mortality. This increase follows similar trends as the nationwide excess mortality (Panels 1B and 1C). When we increase the spatial resolution in panels 3B and 3C, we highlight the province of Foggia, where there was insufficient evidence of a positive excess during the first wave, but strong during the second.
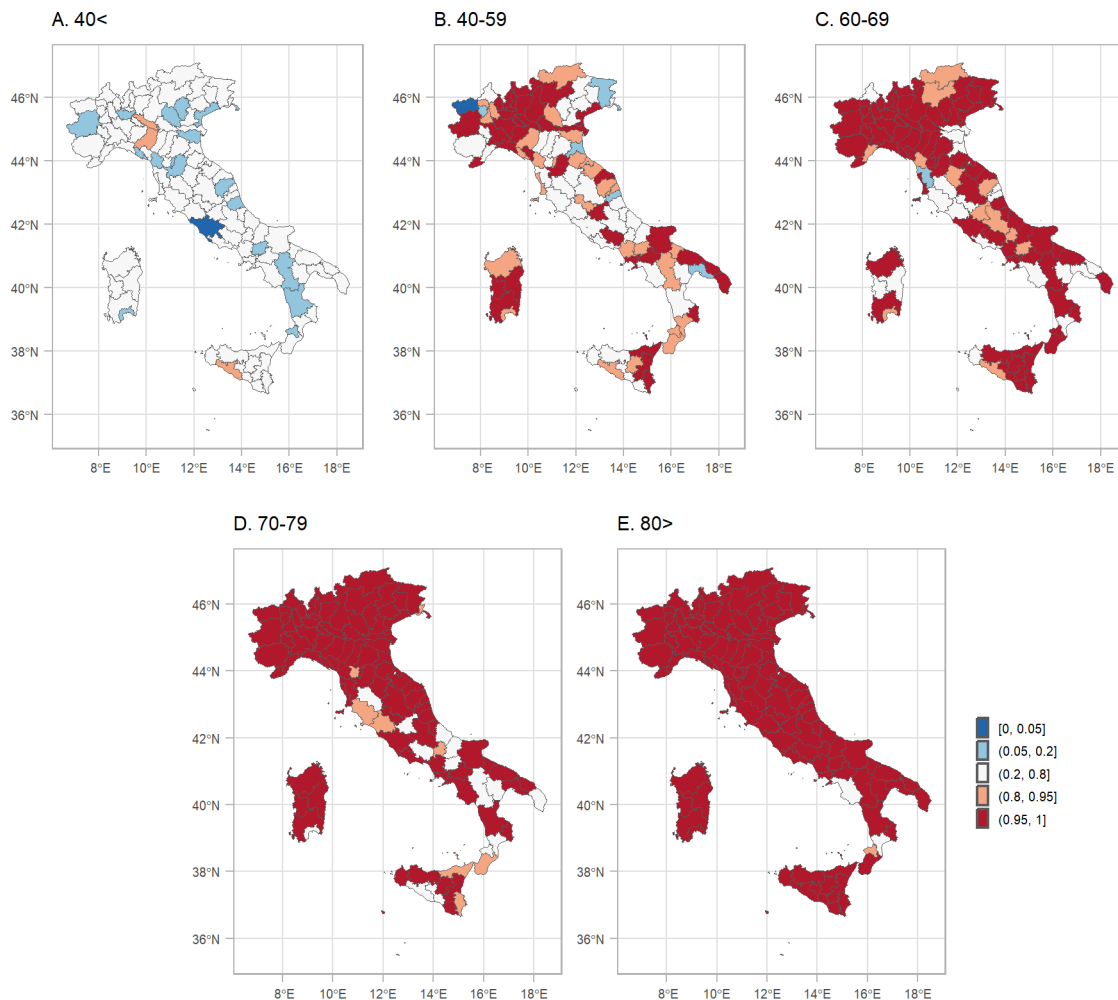
Figure 6: Posterior probability that the relative excess mortality is positive for both sexes during 2020 by age group and NUTS3 region.

4.4 Shiny Web-Application

To be able to effectively examine and communicate the different aggregation levels of the output of our modelling framework, we have also developed a Shiny Web-Application (WebApp), Figure 7. The WebApp provides spatial, temporal and spatio-temporal analysis tabs, and within each tab there are plots and summary statistics for the level of aggregation selected from the drop-down menu. Users can select across different variables (REM or NED), statistics (median or posterior probability), sex (males, females or both), age group ($40 <, 40 - 59, 60 - 69, 70 - 79, 80 >$ and all) and different geographical level (national, NUTS2 or NUTS3 regions). Summary statistics for each area are available and they are displayed in a pop-up window, which is activated by clicking on the area of interest. In addition, graphical pop-ups are provided to show the weekly estimates for each area with the leafpop R-package (Appelhans and Detsch, 2021), in the spatio-temporal analysis tab.

The WebApp that we have developed is hosted at http://atlasmortalidad.uclm.es/italyexcess/.
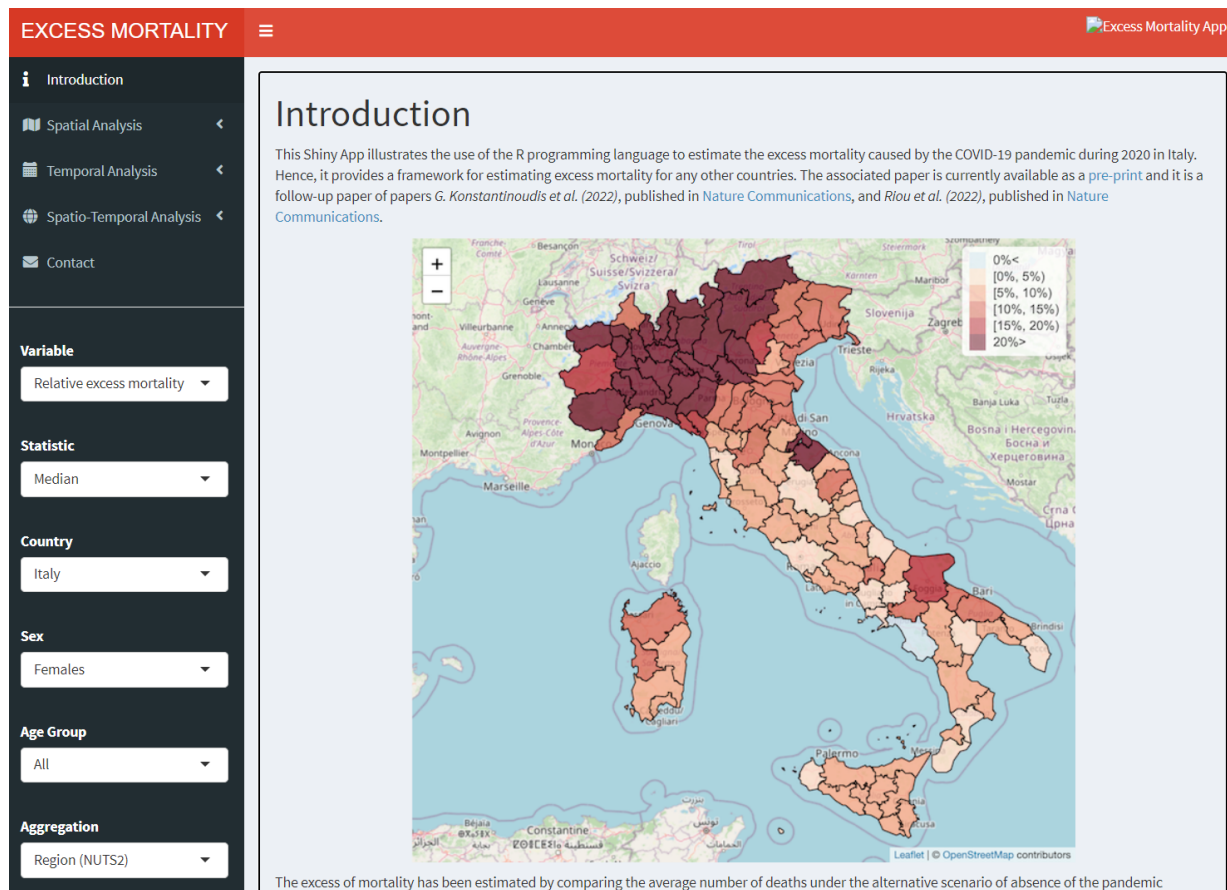
Figure 7: Shiny App developed to navigate through the excess mortality estimates during 2020 in Italy across the different aggregations.

5 Summary

This tutorial provides a detailed explanation of the workflow used previously to calculate excess mortality during the COVID-19 pandemic in 5 European regions (Konstantinoudis et al., 2022). The main model used here is slightly modified based on updated results (Riou et al., 2023). We have proposed a Bayesian workflow for estimating excess mortality and shown how to use R and INLA to retrieve fast and accurate estimates. The proposed workflow also allows for combining different models and presenting the results stratified by age, sex, spatial and temporal location. We have given a practical example of how to use the proposed framework to model the excess mortality during the 2020 COVID-19 pandemic in Italy at small area level. We also developed a Shiny App to effectively communicate the results. The methodological framework can be extended to monitor excess mortality caused by other extreme events; for instance, natural hazards such as tropical cyclones (Parks et al., 2022) or heatwaves (Konstantinoudis et al., 2023). Potential methodological extensions of the proposed framework also include modelling the younger age groups with a zero-inflated Poisson distribution. The proposed framework can also be extended to provide an automated tool for online disease surveillance and policy making.

Acknowledgements

Author contributions

V.G.R. conceived the study. M.B. supervised the study. G.K. developed the initial study protocol and discussed it with M.B., M.C., M.P. and G.B.. G.K. developed the statistical model, prepared the population and covariate data and led the acquisition of mortality data. M.C. and V.G.R. validated and modified accordingly the code. G.K. ran the analysis. G.K., V.G.R., M.B., M.C. and M.P. wrote the initial draft and all the authors contributed in modifying the paper and critically interpreting the results. V.G.R. developed the Shiny app. All authors read and approved the final version for publication.

References

J. M. Aburto, R. Kashyap, J. Schöley, C. Angus, J. Ermisch, M. C. Mills, and J. B. Dowd. Estimating the burden of the COVID-19 pandemic on mortality, life expectancy and lifespan inequality in England and Wales: A population-level analysis. *J Epidemiol Community Health*, 2021. [p89]

T. Appelhans and F. Detsch. *leafpop: Include Tables, Images and Graphs in Leaflet Pop-Ups*, 2021. URL https://CRAN.R-project.org/package=leafpop. R package version 0.1.0. [p90, 99]

D. Attali and T. Edwards. *shinyalert: Easily Create Pretty Popup Messages (Modals) in 'Shiny'*, 2021. URL https://CRAN.R-project.org/package=shinyalert. R package version 3.0.0. [p90]

J. Besag. Spatial interactions and the statistical analysis of lattice systems (with discussion). *Journal of the Royal Statistical Society: Series B*, 36:192–236, 1974. [p91]

J. Besag, J. York, and A. Mollié. Bayesian image restoration, with two applications in spatial statistics. *Annals of the Institute of Statistical Mathematics*, 43(1):1–20, 1991. [p91]

R. Bivand. *R Packages for Analyzing Spatial Data: A Comparative Case Study with Areal Data*, 2022. [p90]

R. Bivand, T. Keitt, and B. Rowlingson. *rgdal: Bindings for the 'Geospatial' Data Abstraction Library*, 2023. URL https://CRAN.R-project.org/package=rgdal. R package version 1.6-4. [p90]

M. Blangiardo, M. Cameletti, M. Pirani, G. Corsetti, M. Battaglini, and G. Baio. Estimating weekly excess mortality at sub-national level in Italy during the COVID-19 pandemic. *PLOS ONE*, 15(10): 1–15, 10 2020. doi: 10.1371/journal.pone.0240286. URL https://doi.org/10.1371/journal.pone.0240286. [p89]

W. Chang and B. Borges Ribeiro. *shinydashboard: Create Dashboards with 'Shiny'*, 2021. URL https://CRAN.R-project.org/package=shinydashboard. R package version 0.7.2. [p90]

W. Chang, J. Cheng, J. Allaire, C. Sievert, B. Schloerke, Y. Xie, J. Allen, J. McPherson, A. Dipert, and B. Borges. *shiny: Web Application Framework for R*, 2022. URL https://CRAN.R-project.org/package=shiny. R package version 1.7.4. [p90]

B. Davies, B. L. Parkes, J. Bennett, D. Fecht, M. Blangiardo, M. Ezzati, and P. Elliott. Community factors and excess mortality in first wave of the COVID-19 pandemic in England. *Nature Communications*, 12(1):1–9, 2021. [p92]

M. Dowle and A. Srinivasan. *data.table: Extension of 'data.frame'*, 2022. URL https://CRAN.R-project.org/package=data.table. R package version 1.14.6. [p90]

S. Garnier, N. Ross, R. Rudis, P. A. Camargo, M. Sciaini, and C. Scherer. *viridis - Colorblind-Friendly Color Maps for R*, 2021. URL https://sjmgarnier.github.io/viridis/. R package version 0.6.2. [p90]

G. Grolemund and H. Wickham. *Dates and Times Made Easy with lubridate*, 2011. URL https://www.jstatsoft.org/v40/i03/. [p90]

H. Hersbach, B. Bell, P. Berrisford, S. Hirahara, A. Horányi, J. Muñoz Sabater, J. Nicolas, C. Peubey, R. Radu, D. Schepers, et al. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020. [p94]

R. J. Hijmans. *raster: Geographic Data Analysis and Modeling*, 2023. URL https://CRAN.R-project.org/package=raster. R package version 3.6-14. [p90]

K. Hufkens, R. Stauffer, and E. Campitelli. *The ecwmfr package: An interface to ECMWF API endpoints*, 2019. URL https://bluegreen-labs.github.io/ecmwfr/. [p90]

N. Islam, V. M. Shkolnikov, R. J. Acosta, I. Klimkin, I. Kawachi, R. A. Irizarry, G. Alicandro, K. Khunti, T. Yates, D. A. Jdanov, et al. Excess deaths associated with COVID-19 pandemic in 2020: age and sex disaggregated time series analysis in 29 high income countries. *BMJ*, 373, 2021. [p89]

J. Kaczorowski and C. Del Grande. Beyond the tip of the iceberg: direct and indirect effects of COVID-19. *The Lancet Digital Health*, 3(4):e205–e206, 2021. [p89]

G. Konstantinoudis, D. Schuhmacher, H. Rue, and B. D. Spycher. Discrete versus continuous domain models for disease mapping. *Spatial and Spatio-temporal Epidemiology*, 32:100319, 2020. ISSN 1877-5845. doi: https://doi.org/10.1016/j.sste.2019.100319. URL https://www.sciencedirect.com/science/article/pii/S1877584519301297. [p91]

G. Konstantinoudis, M. Cameletti, V. Gómez-Rubio, I. L. Gómez, M. Pirani, G. Baio, A. Larrauri, J. Riou, M. Egger, P. Vineis, et al. Regional excess mortality during the 2020 COVID-19 pandemic in five European countries. *Nature communications*, 13(1):482, 2022. [p89, 90, 100]

G. Konstantinoudis, A. Hauser, and J. Riou. Bayesian ensemble modelling to monitor excess deaths during summer 2022 in Switzerland. *arXiv preprint arXiv:2308.15251*, 2023. [p100]

V. Kontis, J. E. Bennett, T. Rashid, R. M. Parks, J. Pearson-Stuttard, M. Guillot, P. Asaria, B. Zhou, M. Battaglini, G. Corsetti, et al. Magnitude, demographics and dynamics of the effect of the first wave of the COVID-19 pandemic on all-cause mortality in 21 industrialized countries. *Nature Medicine*, pages 1–10, 2020. [p89]

V. Kontis, J. Bennett, R. Parks, T. Rashid, J. Pearson-Stuttard, P. Asaria, B. Zhou, M. Guillot, C. Mathers, Y. Khang, M. McKee, and M. Ezzati. Lessons learned and lessons missed: impact of the coronavirus disease 2019 (COVID-19) pandemic on all-cause mortality in 40 industrialised countries prior to mass vaccination [version 1; peer review: 2 approved with reservations]. *Wellcome Open Research*, 6 (279), 2021. doi: 10.12688/wellcomeopenres.17253.1. [p89]

E. Kontopantelis, M. A. Mamas, J. Deanfield, M. Asaria, and T. Doran. Excess mortality in England and Wales during the first wave of the COVID-19 pandemic. *J Epidemiol Community Health*, 75(3): 213–223, 2021. [p89]

P. Moraga. Small Area Disease Risk Estimation and Visualization Using R. *The R Journal*, 10(1):495–506, 2018. doi: 10.32614/RJ-2018-036. URL https://doi.org/10.32614/RJ-2018-036. [p89]

E. Neuwirth. *RColorBrewer: ColorBrewer Palettes*, 2022. URL https://CRAN.R-project.org/package=RColorBrewer. R package version 1.1-3. [p90]

R. M. Parks, J. Benavides, G. B. Anderson, R. C. Nethery, A. Navas-Acien, F. Dominici, M. Ezzati, and M.-A. Kioumourtzoglou. Association of tropical cyclones with county-level mortality in the US. *JAMA*, 327(10):946–955, 2022. [p100]

E. Pebesma. Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal*, 10(1): 439–446, 2018. doi: 10.32614/RJ-2018-009. URL https://doi.org/10.32614/RJ-2018-009. [p90]

T. L. Pedersen. *patchwork: The Composer of Plots*, 2022. URL https://CRAN.R-project.org/package=patchwork. R package version 1.1.2. [p90]

D. Pierce. *ncdf4: Interface to Unidata netCDF (Version 4 or Earlier) Format Data Files*, 2023. URL https://CRAN.R-project.org/package=ncdf4. R package version 1.21. [p90]

T. Plate and R. Heiberger. *abind: Combine Multidimensional Arrays*, 2016. URL https://CRAN.R-project.org/package=abind. R package version 1.4-5. [p90]

A. Riebler, S. H. Sørbye, D. Simpson, and H. Rue. An intuitive Bayesian spatial model for disease mapping that accounts for scaling. *Statistical Methods in Medical Research*, 25(4):1145–1165, 2016. [p91]

J. Riou, A. Hauser, A. Fesser, C. L. Althaus, M. Egger, and G. Konstantinoudis. Direct and indirect effects of the COVID-19 pandemic on mortality in Switzerland. *Nature communications*, 14(1):90, 2023. [p90, 93, 95, 100]

L. M. Rossen, A. M. Branum, F. B. Ahmad, P. Sutton, and R. N. Anderson. Excess deaths associated with COVID-19, by age and race and ethnicity – United States, January 26–October 3, 2020. *Morbidity and Mortality Weekly Report*, 69(42):1522, 2020. [p89]

H. Rue and L. Held. *Gaussian Markov random fields: theory and applications*. CRC press, 2005. [p92]

H. Rue, S. Martino, and N. Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B (statistical methodology)*, 71(2):319–392, 2009. [p89, 92]

D. Simpson, H. Rue, A. Riebler, T. G. Martins, S. H. Sørbye, et al. Penalising model component complexity: A principled, practical approach to constructing priors. *Statistical Science*, 32(1):1–28, 2017. [p91]

J. Verbeeck, C. Faes, T. Neyens, N. Hens, G. Verbeke, P. Deboosere, and G. Molenberghs. A linear mixed model to estimate COVID-19-induced excess mortality. *Biometrics*, pages 1–9, 2021. doi: https://doi.org/10.1111/biom.13578. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/biom.13578. [p89]

L. A. Waller, B. P. Carlin, H. Xia, and A. E. Gelfand. Hierarchical spatio-temporal mapping of disease rates. *Journal of the American Statistical association*, 92(438):607–617, 1997. [p89]

D. M. Weinberger, J. Chen, T. Cohen, F. W. Crawford, F. Mostashari, D. Olson, V. E. Pitzer, N. G. Reich, M. Russi, L. Simonsen, et al. Estimation of excess deaths associated with the COVID-19 pandemic in the United States, March to May 2020. *JAMA Internal Medicine*, 180(10):1336–1344, 2020. [p89]

H. Wickham. *Reshaping Data with the reshape Package*, 2007. URL http://www.jstatsoft.org/v21/i12/. [p90]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p90]

H. Wickham. *stringr: Simple, Consistent Wrappers for Common String Operations*, 2022. URL https://CRAN.R-project.org/package=stringr. R package version 1.5.0. [p90]

H. Wickham, J. Hester, and J. Bryan. *readr: Read Rectangular Text Data*, 2022. URL https://CRAN.R-project.org/package=readr. R package version 2.1.3. [p90]

H. Wickham, R. François, L. Henry, K. Müller, and D. Vaughan. *dplyr: A Grammar of Data Manipulation*, 2023a. URL https://CRAN.R-project.org/package=dplyr. R package version 1.1.0. [p90]

H. Wickham, D. Vaughan, and M. Girlich. *tidyr: Tidy Messy Data*, 2023b. URL https://CRAN.R-project.org/package=tidyr. R package version 1.3.0. [p90]

D. Wuertz, T. Setz, Y. Chalabi, and G. N. Boshnakov. *timeDate: Rmetrics - Chronological and Calendar Objects*, 2023. URL https://CRAN.R-project.org/package=timeDate. R package version 4022.108. [p90]

Garyfallos Konstantinoudis
MRC Centre for Environment and Health, Imperial College London,
St Mary's Campus, Praed St,
W2 1NY, London
United Kingdom

Virgilio Gómez-Rubio
Departamento de Matemáticas, Escuela Técnica Superior de Ingeniería Industrial-Albacete, Universidad de Castilla-La Mancha,
Albacete, Spain

Michela Cameletti
Department of Economics, University of Bergamo,

Bergamo, Italy

Monica Pirani
MRC Centre for Environment and Health, Imperial College London,
St Mary's Campus, Praed St,
W2 1NY, London
United Kingdom

Gianluca Baio
Department of Statistical Sciences, University College London,
United Kingdom

Marta Blangiardo
MRC Centre for Environment and Health, Imperial College London,
St Mary's Campus, Praed St,
W2 1NY, London
United Kingdom

Generalized Estimating Equations using the new R package glmtoolbox

by L.H. Vanegas, L.M. Rondón, and G.A. Paula

Abstract This paper introduces a very comprehensive implementation, available in the new R package glmtoolbox, of a very flexible statistical tool known as Generalized Estimating Equations (GEE), which analyzes cluster correlated data utilizing marginal models. As well as providing more built-in structures for the working correlation matrix than other GEE implementations in R, this GEE implementation also allows the user to: (1) compute several estimates of the variance-covariance matrix of the estimators of the parameters of interest; (2) compute several criteria to assist the selection of the structure for the working-correlation matrix; (3) compare nested models using the Wald test as well as the generalized score test; (4) assess the goodness-of-fit of the model using Pearson-, deviance- and Mahalanobis-type residuals; (5) perform sensibility analysis using the global influence approach (that is, dfbeta statistic and Cook's distance) as well as the local influence approach; (6) use several criteria to perform variable selection using a hybrid stepwise procedure; (7) fit models with nonlinear predictors; (8) handle dropout-type missing data under MAR rather than MCAR assumption by using observation-specific or cluster-specific weighted methods. The capabilities of this GEE implementation are illustrated by analyzing four real datasets obtained from longitudinal studies.

1   Introduction

The Generalized Estimating Equations (GEE), proposed by Liang and Zeger (1986), extend the theoretical framework of the Generalized Least Squares (GLS) by allowing the variance of the response variable distribution to be proportional to a known function of its mean, resulting thus in a very flexible statistical tool for the analysis of heteroskedastic discrete and continuous cluster correlated data. Unlike conditional models such as random-effect models, the GEE approach is based on marginal models. In addition, and according to Lipsitz and Fitzmaurice (2008), GEE can also be regarded as a multivariate generalization of the quasi-likelihood approach to Generalized Linear Models (GLMs) introduced by Wedderburn (1974). The main advantage of GEE over other approaches to analyzing cluster correlated data lies in that this methodology does not require the full specification of the multivariate distribution of the (discrete or continuous) response vector measured on each subject or cluster, reducing the possibility of model misspecification. Indeed, GEE just requires the following:

- Specification of a variance function, which describes the mechanism of heteroscedasticity (if there is any), that is, it describes the way in which the variance of the response variable distribution is assumed to be dependent on its mean.
- Specification of a regression structure, very similar to that described in the theoretical framework of the GLMs (see, for instance, McCullagh and Nelder (1989)), that includes a link function and a linear predictor, which describe the way the mean of the response variable distribution is assumed to be dependent on some continuous and/or discrete regressors.
- Specification of a correlation matrix structure. This matrix describes the dynamic of the linear association between the different measurements of the response variable performed on the same subject or cluster.

This paper introduces the package glmtoolbox, which, besides providing more built-in structures for the working correlation matrix than other GEE implementations available in R, has several features, including: (1) compute several estimates of the variance-covariance matrix of the estimators of the parameters of interest; (2) compute several criteria to assist the selection of the structure for the working correlation matrix; (3) compare nested models using the Wald test as well as the generalized score test; (4) assess the goodness-of-fit of the model using Pearson-, deviance- and Mahalanobis-type residuals; (5) perform sensibility analysis using the global influence approach (that is, dfbeta statistic and Cook's distance) as well as the local influence approach; (6) use several criteria to perform variable selection using a hybrid stepwise procedure; (7) fit models with nonlinear predictors; (8) handle dropout-type missing data under MAR rather than MCAR assumption by using observation-specific or cluster-specific weighted methods. The rest of this paper is organized as follows: in Section 2 the main features of the GEE model setup are described; in Section 3 the main features of the implementation of GEE in the package glmtoolbox are described and compared with those available in the packages gee (Carey, 2022), geepack (Yan, 2002; Højsgaard et al., 2005) and geeM (McDaniel et al., 2013), which are the most widely used packages in R to analyze cluster correlated data using GEE; in Section 4 the capabilities of this implementation are illustrated by analyzing four real datasets obtained from longitudinal studies.

2 Generalized Estimating Equations

Let $\mathbf{y}_i = (y_{i1}, \ldots, y_{ij}, \ldots, y_{i\,n_i})^\top$ for $i = 1, \ldots, n$ be the multivariate response of interest measured on n subjects or clusters, which are assumed to be realizations of independent random vectors denoted here by $\mathbf{Y}_i = (Y_{i1}, \ldots, Y_{ij}, \ldots, Y_{i\,n_i})^\top$ for $i = 1, \ldots, n$, where n_i represents the size of the ith cluster or the number of measurements performed on the ith subject. So, the total number of observations is $N = n_1 + \ldots + n_n$. The random variables associated with the ith subject or cluster, given by Y_{ij} for $j = 1, \ldots, n_i$, are assumed to satisfy the following:

$$\text{Var}(Y_{ij}) = \frac{\phi}{\omega_{ij}} V(\mu_{ij}) \qquad \text{and} \qquad \text{Corr}(Y_{ij}, Y_{ik}) = r_{jk}(\boldsymbol{\rho}),$$

where $\mu_{ij} = \text{E}(Y_{ij})$, $\phi > 0$ is the dispersion parameter, $\omega_{ij} > 0$ are known weights typically specified to be 1, $V(\mu) > 0$ is the variance function, and $r_{jk}(\boldsymbol{\rho})$ is the Pearson's linear correlation coefficient, which is assumed to be dependent just on j, k and the unknown nuisance parameter vector denoted here by $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_q)^\top$. In addition, μ_{ij} is assumed to be dependent on a vector of p continuous and/or discrete regressors, denoted here by $(x_{1ij}, \ldots, x_{pij})$, in the following way:

$$g(\mu_{ij}) = \mathbf{x}_{ij}^\top \boldsymbol{\beta}, \tag{1}$$

where $g(\mu)$ is a strictly monotone and twice-differentiable known function better known as link function, $\mathbf{x}_{ij} = (1, x_{1ij}, \ldots, x_{pij})^\top$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^\top$ is the interest parameter vector.

According to Liang and Zeger (1986), the estimate of $\boldsymbol{\beta}$, denoted here by $\hat{\boldsymbol{\beta}}$, reduces to the solution to the $(p + 1)$ equations given by $\mathbf{U}(\hat{\boldsymbol{\beta}}) = \mathbf{0}$, where

$$\mathbf{U}(\boldsymbol{\beta}) = \phi^{-1} \sum_{i=1}^{n} \mathbf{X}_i^\top \mathbf{K}_i \mathbf{V}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i) = \phi^{-1} \sum_{i=1}^{n} \mathbf{X}_i^\top \mathbf{W}_i \mathbf{K}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i) = \phi^{-1} \mathbf{X}^\top \mathbf{W} \mathbf{K}^{-1}(\mathbf{y} - \boldsymbol{\mu}), \tag{2}$$

in which $\mathbf{X}_i = (\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i})^\top$, $\mathbf{W}_i = \mathbf{K}_i \mathbf{V}_i^{-1} \mathbf{K}_i$, $\mathbf{K}_i = \text{diag}\{1/g'(\mu_{i1}), \ldots, 1/g'(\mu_{in_i})\}$, $\mathbf{V}_i = \mathbf{A}_i^{\frac{1}{2}} \mathbf{R}_i \mathbf{A}_i^{\frac{1}{2}}$, $\mathbf{A}_i = \text{diag}\{V(\mu_{i1})/\omega_{i1}, \ldots, V(\mu_{in_i})/\omega_{in_i}\}$, $\mathbf{R}_i$ is a square matrix whose (j, k)th entry is $r_{jk}(\boldsymbol{\rho})$, $\boldsymbol{\mu}_i = (\mu_{i1}, \ldots, \mu_{in_i})^\top$, $\mathbf{X} = (\mathbf{X}_1^\top, \ldots, \mathbf{X}_n^\top)^\top$, $\mathbf{W} = \text{blockdiag}\{\mathbf{W}_1, \ldots, \mathbf{W}_n\}$, $\mathbf{K} = \text{blockdiag}\{\mathbf{K}_1, \ldots, \mathbf{K}_n\}$, $\mathbf{y} = (\mathbf{y}_1^\top, \ldots, \mathbf{y}_n^\top)^\top$ and $\boldsymbol{\mu} = (\boldsymbol{\mu}_1^\top, \ldots, \boldsymbol{\mu}_n^\top)^\top$. Moreover, the estimate of ϕ may be written as follows:

$$\hat{\phi} = \frac{1}{N - p - 1} \sum_{i=1}^{n} \sum_{j=1}^{n_i} \frac{(y_{ij} - \hat{\mu}_{ij})^2}{V(\hat{\mu}_{ij})/\omega_{ij}},$$

where $\hat{\mu}_{ij} = g^{-1}(\mathbf{x}_{ij}^\top \hat{\boldsymbol{\beta}})$. If the model for the mean (μ) is correctly specified, then, under certain regularity conditions, $\hat{\boldsymbol{\beta}}$ is consistent for $\boldsymbol{\beta}$ and its distribution is such that (Liang and Zeger, 1986):

$$\sqrt{n}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \xrightarrow[n \to \infty]{\mathcal{D}} \mathcal{N}(\mathbf{0}, \text{Var}(\hat{\boldsymbol{\beta}})),$$

where

$$\text{Var}(\hat{\boldsymbol{\beta}}) = \lim_{n \to \infty} \left(\frac{1}{n} \mathbf{X}^\top \mathbf{W} \mathbf{X}\right)^{-1} \left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{X}_i^\top \mathbf{W}_i \mathbf{K}_i^{-1} \text{Var}(\mathbf{Y}_i) \mathbf{K}_i^{-1} \mathbf{W}_i \mathbf{X}_i\right) \left(\frac{1}{n} \mathbf{X}^\top \mathbf{W} \mathbf{X}\right)^{-1}.$$

Therefore, if the mean model is correctly specified, then $\hat{\boldsymbol{\beta}}$ remain consistent and asymptotically normal distributed regardless of whether or not the correlation matrix structure is correctly specified. Indeed, if the structure of the correlation matrix is also correctly specified, that is, if $\text{Var}(\mathbf{Y}_i) = \phi \mathbf{V}_i$ for $i = 1, \ldots, n$, then $\text{Var}(\hat{\boldsymbol{\beta}})$ reduces to

$$\text{Var}(\hat{\boldsymbol{\beta}}) = \lim_{n \to \infty} \phi \left(\frac{1}{n} \mathbf{X}^\top \mathbf{W} \mathbf{X}\right)^{-1}.$$

3 R package `glmtoolbox`

The function `glmgee()` is the GEE solver available in the package `glmtoolbox`. That function includes the typical arguments present in regression routines such as `lm()` and `glm()`, that is, it includes arguments such as `formula`, `weights`, `start`, `data` and `subset`. In addition, the objects generated by the function `glmgee()` are associated with the typical extraction methods such as `summary()`, `print()`, `coef()`, `vcov()`, `fitted()`, `confint()`, `anova()`, `residuals()`, `predict()`, `leverage()`, `dfbeta()` and

`cooks.distance()`. Next, the main features of the implementation of GEE in `glmtoolbox` are described and compared with those available in the packages `gee`, `geepack` and `geeM`.

3.1 Link and variance functions

The available options for the link ($g(\mu)$) and variance function ($V(\mu)$) in the routine `glmgee()` are the following:

family	$V(\mu)$	$g(\mu)$
gaussian	1	inverse (μ^{-1}), identity (μ), log $(\log(\mu))$
binomial	$\mu(1-\mu)$	logit $\left(\log\left(\frac{\mu}{1-\mu}\right)\right)$, cloglog $(\log(-\log(1-\mu)))$, probit $(\Phi^{-1}(\mu))$, cauchit $(\tan(\frac{\pi}{2}(2\mu-1)))$
poisson	μ	sqrt $(\mu^{\frac{1}{2}})$, identity (μ), log $(\log(\mu))$
Gamma	μ^2	inverse (μ^{-1}), identity (μ), log $(\log(\mu))$
inverse.gaussian	μ^3	1/mu^2 (μ^{-2}), inverse (μ^{-1}), identity (μ), log $(\log(\mu))$
negative.binomial(θ)[1]	$\mu(1+\mu/\theta)$	log $(\log(\mu))$, identity (μ), sqrt $(\mu^{\frac{1}{2}})$
tweedie(θ,γ)[2]	μ^θ	$\log(\mu)$ if $\gamma=0$ and μ^γ if $\gamma\neq 0$

1 function available in package MASS
2 function available in package statmod

Moreover, new families and new link functions may be defined by the user as described on the help page of the routine `glm()`. The variance functions $V(\mu)=\mu^3$ and $V(\mu)=\mu(1+\mu/\theta)$ are not available in gee.

3.2 Estimating algorithm

The $(p+1)$ equations given by $\mathbf{U}(\hat{\boldsymbol{\beta}})=\mathbf{0}$ may be solved using the following algorithm:

Step 0: Start the counter at $t=0$; set the tolerance limit, $\epsilon>0$; set the maximum number of iterations, $n_{\max}$; and set the initial value for $\boldsymbol{\beta}$, say $\boldsymbol{\beta}^{[0]}$.

Step 1: Compute $\rho^{[t]}$ from the Pearson's residuals evaluated at $\boldsymbol{\beta}^{[t]}$, denoted here by $r_{ij}^{[t]}$.

Step 2: Compute $\boldsymbol{\beta}^{[t+1]}=\boldsymbol{\beta}^{[t]}+[\mathbf{K}(\boldsymbol{\beta}^{[t]})]^{-1}\mathbf{U}(\boldsymbol{\beta}^{[t]})=(\mathbf{X}^\top\mathbf{W}^{[t]}\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{W}^{[t]}\widetilde{\mathbf{y}}^{[t]}$.

Step 3: Compute $\delta^{(t+1)}=\delta(\boldsymbol{\beta}^{[t]},\boldsymbol{\beta}^{[t+1]})$.

Step 4: Update the counter by $t=t+1$.

Step 5: Repeat Steps 1,2,3 and 4 until $\delta^{(t)}<\epsilon$ or $t>n_{\max}$.

Step 6: If $\delta^{(t)}<\epsilon$, then $\hat{\boldsymbol{\beta}}$ is defined to be $\boldsymbol{\beta}^{[t]}$. Otherwise, convergence was not achieved.

Note that,

- $\boldsymbol{\beta}^{[0]}$ is specified to be the estimate of $\boldsymbol{\beta}$ in the GLM under which the random variables Y_{ij} for $i=1,\ldots,n$ and $j=1,\ldots,n_i$ are assumed to be independent. This may be easily obtained by using the function `glm()`. However, the starting value, $\boldsymbol{\beta}^{[0]}$, also may be supplied by the user with the argument `start` of the function `glmgee()`.

- $r_{ij}^{[t]}=\dfrac{y_{ij}-\mu_{ij}^{[t]}}{\sqrt{\phi^{[t]}\,V(\mu_{ij}^{[t]})/\omega_{ij}}}$ for $i=1,\ldots,n$ and $j=1,\ldots,n_i$, with $\phi^{[t]}=\dfrac{1}{N-p-1}\sum_{i=1}^{n}\sum_{j=1}^{n_i}\dfrac{(y_{ij}-\mu_{ij}^{[t]})^2}{V(\mu_{ij}^{[t]})/\omega_{ij}}$.

- $\delta(\mathbf{a},\mathbf{b})$ is a non-negative and strictly increasing function of the "difference" between the vectors $\mathbf{a}=(a_1,\ldots,a_{p+1})^\top$ and $\mathbf{b}=(b_1,\ldots,b_{p+1})^\top$. For instance, $\delta(\mathbf{a},\mathbf{b})=||\mathbf{b}-\mathbf{a}||_r$ or $\delta(\mathbf{a},\mathbf{b})=||(\mathbf{b}-\mathbf{a})^*||_r$, where $||\mathbf{a}||_r=(|a_1|^r+\ldots+|a_{p+1}|^r)^{1/r}$ for any $r\geq 1$, $||\mathbf{a}||_\infty=\max\{|a_1|,\ldots,|a_{p+1}|\}$ and $(\mathbf{b}-\mathbf{a})^*:=((b_1-a_1)/|a_1|,\ldots,(b_{p+1}-a_{p+1})/|a_{p+1}|)$. The comparison criterion in the function `glmgee()` is $\delta(\mathbf{a},\mathbf{b})=||(\mathbf{b}-\mathbf{a})^*||_\infty=\max\{|b_1-a_1|/|a_1|,\ldots,|b_{p+1}-a_{p+1}|/|a_{p+1}|\}$. In addition, the values of the tolerance limit, ϵ, and the maximum number of iterations, $n_{\max}$, may be specified in the function `glmgee()` via its arguments `toler` and `maxit`, respectively. By default, `toler = 0.00001` and `maxit = 50`. If `trace=TRUE` is specified in the function `glmgee()`, then the values of $\delta(\boldsymbol{\beta}^{[t]},\boldsymbol{\beta}^{[t+1]})$ are printed until convergence is reached or the maximum number of iterations is exceeded.

- $\mathbf{K}(\boldsymbol{\beta}) = \mathrm{E}\left(\dfrac{\partial \mathbf{U}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}^{\top}}\right) = \phi^{-1} \sum\limits_{i=1}^{n} \mathbf{X}_i^{\top} \mathbf{W}_i \mathbf{X}_i = \phi^{-1} \mathbf{X}^{\top} \mathbf{W} \mathbf{X}$ and $\tilde{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta} + \mathbf{K}^{-1}(\mathbf{y} - \boldsymbol{\mu})$. Therefore, $\hat{\boldsymbol{\beta}}$ may be written as $\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\top} \hat{\mathbf{W}} \mathbf{X})^{-1} \mathbf{X}^{\top} \hat{\mathbf{W}} \hat{\tilde{\mathbf{y}}}$, in which $\hat{\mathbf{W}}$ and $\hat{\tilde{\mathbf{y}}}$ represent to $\mathbf{W}$ and $\tilde{\mathbf{y}}$ evaluated at $\hat{\boldsymbol{\beta}}$, respectively. Thus, $\hat{\boldsymbol{\beta}}$ can be regarded as the GLS estimate of $\boldsymbol{\beta}$ in a linear model such that $\mathrm{E}(\tilde{\mathbf{Y}}) = \mathbf{X}\boldsymbol{\beta}$ and $\mathrm{Var}(\tilde{\mathbf{Y}}) = \sigma^2 \hat{\mathbf{W}}^{-1}$, with $\hat{\tilde{\mathbf{y}}}$ being the observed value of the random vector $\tilde{\mathbf{Y}}$.

The package glmtoolbox also includes an extracting method, named estequa(), associated with the objects generated by the function glmgee(), which allows the user to verify if, actually, the parameter estimates satisfy the generalized estimating equations, that is, the extracting method estequa() provides the value of the vector $\mathbf{U}(\boldsymbol{\beta})$ evaluated at the parameter estimates and the observed data.

3.3 Structures for the working-correlation matrix

The available options for the structure of the working correlation matrix in the function glmgee() are the following:

- corstr="Independence":

$$\mathrm{Corr}(Y_{ij}, Y_{ik}) = \begin{cases} 1, & \text{if } j = k, \\ 0, & \text{if } j \neq k \end{cases}$$

- corstr="Exchangeable":

$$\mathrm{Corr}(Y_{ij}, Y_{ik}) = \begin{cases} 1, & \text{if } j = k, \\ \rho, & \text{if } j \neq k, \end{cases} \quad \text{and} \quad \rho^{[t]} = \frac{1}{M - p - 1} \sum_{i=1}^{n} \sum_{j < k} r_{ij}^{[t]} r_{ik}^{[t]},$$

where $M = \dfrac{1}{2} \sum\limits_{i=1}^{n} n_i(n_i - 1)$.

- corstr="AR-M-dependent(m)":
  If $m = 1$, then the values of $\mathrm{Corr}(Y_{ij}, Y_{ik})$ become

$$\mathrm{Corr}(Y_{ij}, Y_{ik}) = \begin{cases} 1, & \text{if } j = k, \\ \rho^{|j-k|}, & \text{if } j \neq k, \end{cases} \quad \text{and} \quad \rho^{[t]} = \frac{1}{M - p - 1} \sum_{i=1}^{n} \sum_{j=1}^{n_i - 1} r_{ij}^{[t]} r_{i,j+1}^{[t]},$$

where $M = \sum\limits_{i=1}^{n} (n_i - 1)$.

- corstr="Stationary-M-dependent(m)":

$$\mathrm{Corr}(Y_{ij}, Y_{i,j+l}) = \begin{cases} 1, & \text{if } t = 0, \\ \rho_l, & \text{if } l = 1, \dots, m, \\ 0, & \text{if } l > m, \end{cases} \quad \text{and} \quad \rho_l^{[t]} = \frac{1}{M_l - p - 1} \sum_{i=1}^{n} \sum_{j=1}^{n_i - l} r_{ij}^{[t]} r_{i,j+l}^{[t]},$$

where $M_l = \sum\limits_{i=1}^{n} (n_i - l)$.

- corstr="Non-Stationary-M-dependent(m)":

$$\mathrm{Corr}(Y_{ij}, Y_{ik}) = \begin{cases} 1, & \text{if } j = k, \\ \rho_{jk}, & \text{if } 0 < |j - k| \leq m, \\ 0, & \text{if } |j - k| > m, \end{cases} \quad \text{and} \quad \rho_{jk}^{[t]} = \frac{1}{n - p - 1} \sum_{i=1}^{n} r_{ij}^{[t]} r_{ik}^{[t]},$$

- corstr="Unstructured":

$$\mathrm{Corr}(Y_{ij}, Y_{ik}) = \begin{cases} 1, & \text{if } j = k, \\ \rho_{jk}, & \text{if } j \neq k, \end{cases} \quad \text{and} \quad \rho_{jk}^{[t]} = \frac{1}{n - p - 1} \sum_{i=1}^{n} r_{ij}^{[t]} r_{ik}^{[t]},$$

- corstr="User-defined":
  Supplied by the user at the argument corr.

In geepack the structure Stationary-M-dependent is not available. Furthermore, Non-Stationary-M-dependent and AR-M-dependent (for $m > 1$) structures are not available in geepack nor in geeM.

3.4 Missing values

The rows of the data set with the same value in the variable specified by the argument id of the function glmgee() are assumed to belong to the same cluster regardless of whether they are located in consecutive rows. If the data are longitudinal, that is, if the data consist of measurements performed on the same subject/cluster but at different time points, then, by default, the function glmgee() assumes that the rows belonging to the same subject/cluster are ordered in time. However, the function glmgee() allows specifying, via its argument waves, an integer-valued vector, which by default is set to be $1, \ldots, n_i$, with the time points of the rows corresponding to each subject/cluster.

In longitudinal data, in which the structures for the working correlation matrix such as AR-M-dependent, Stationary-M-dependent, Non-Stationary-M-dependent and Unstructured become meaningful, the missing values may be present in one of the following ways:

- Missing values are located at the first time points.
- Intermittent missing values, that is, missing values intermixed with non-missing values in time.
- Missing values located at the last time points, also known as dropouts.

Similar to the packages geepack and geeM, the GEE solver in the package glmtoolbox allows the user to specify, via its argument waves, the way in which the missing values occurred, that is, it allows the user to specify an integer-valued vector with the time points of the non-missing values. By default, waves is set to be $1, \ldots, n_i$, meaning the missing values, if any, occurred at the last time points. The missing-data pattern is assumed to be *Missing Completely At Random* (MCAR) (see, for instance, Laird (1988)). Statistical inferences based on the GEE approach under the presence of missing values remain valid in such a scenario. According to Lipsitz and Fitzmaurice (2008), the data are said to be MCAR when the probability that responses are missing is unrelated to either the specific values that, in principle, should have been obtained (the missing responses) or the set of observed responses. In Section 2.4.2, the weighted GEE method to handle dropout-type missing data under the MAR (*Missing At Random*) assumption is approached. The MAR assumption is weaker than MCAR.

3.5 Variance estimation

The vcov-type extraction method associated with the objects generated by the function glmgee() allows the user to compute five different estimates of $\mathrm{Var}(\hat{\boldsymbol{\beta}})$. The user may specify the estimate type through the argument type of the vcov-type method. The five types of estimates for $\mathrm{Var}(\hat{\boldsymbol{\beta}})$ are described as follows:

- type="model":

$$\hat{\mathrm{Var}}_{\mathrm{M}}(\hat{\boldsymbol{\beta}}) = [\mathbf{K}(\hat{\boldsymbol{\beta}})]^{-1} = \hat{\phi}\left(\mathbf{X}^{\top}\hat{\mathbf{W}}\mathbf{X}\right)^{-1}$$

- type="robust" (Liang and Zeger, 1986):

$$\hat{\mathrm{Var}}_{\mathrm{R}}(\hat{\boldsymbol{\beta}}) = \left(\mathbf{X}^{\top}\hat{\mathbf{W}}\mathbf{X}\right)^{-1}\left(\sum_{i=1}^{n}\mathbf{X}_i^{\top}\hat{\mathbf{W}}_i\hat{\mathbf{K}}_i^{-1}\mathbf{e}_i\mathbf{e}_i^{\top}\hat{\mathbf{K}}_i^{-1}\hat{\mathbf{W}}_i\mathbf{X}_i\right)\left(\mathbf{X}^{\top}\hat{\mathbf{W}}\mathbf{X}\right)^{-1},$$

  where $\mathbf{e}_i = \mathbf{y}_i - \hat{\boldsymbol{\mu}}_i$. This estimator is robust to misspecification of the working correlation matrix. That is, it is a consistent estimator of $\mathrm{Var}(\hat{\boldsymbol{\beta}})$ provided that the mean model (μ) is correctly specified.

- type="df-adjusted":

$$\hat{\mathrm{Var}}_{\mathrm{A}}(\hat{\boldsymbol{\beta}}) = \frac{n}{n-p-1}\hat{\mathrm{Var}}_{\mathrm{R}}(\hat{\boldsymbol{\beta}})$$

- type="bias-corrected" (Mancl and DeRouen, 2001):

$$\hat{\mathrm{Var}}_{\mathrm{B}}(\hat{\boldsymbol{\beta}}) = \left(\mathbf{X}^{\top}\hat{\mathbf{W}}\mathbf{X}\right)^{-1}\left(\sum_{i=1}^{n}\mathbf{X}_i^{\top}\hat{\mathbf{W}}_i\hat{\mathbf{K}}_i^{-1}\widetilde{\mathbf{e}}_i\widetilde{\mathbf{e}}_i^{\top}\hat{\mathbf{K}}_i^{-1}\hat{\mathbf{W}}_i\mathbf{X}_i\right)\left(\mathbf{X}^{\top}\hat{\mathbf{W}}\mathbf{X}\right)^{-1},$$

  where $\widetilde{\mathbf{e}}_i = (\mathbf{I} - \hat{\mathbf{H}}_i)^{-1}\mathbf{e}_i$ and $\hat{\mathbf{H}}_i = \hat{\mathbf{K}}_i\mathbf{X}_i\left(\mathbf{X}^{\top}\hat{\mathbf{W}}\mathbf{X}\right)^{-1}\mathbf{X}_i^{\top}\hat{\mathbf{K}}_i\hat{\mathbf{V}}_i^{-1}$. The "bias-corrected" estimator for $\mathrm{Var}(\hat{\boldsymbol{\beta}})$ is also robust to the misspecification of the working correlation matrix, and is very useful in cases where the sample size is "small" due to its improved finite sample properties.

- type="jackknife" (Lipsitz et al., 1990):

$$\hat{\mathrm{Var}}_{\mathrm{J}}(\hat{\boldsymbol{\beta}}) = \left(\sum_{i=1}^{n}\hat{\boldsymbol{\beta}}_{(i)}^{1} - \overline{\hat{\boldsymbol{\beta}}^{1}}\right)\left(\sum_{i=1}^{n}\hat{\boldsymbol{\beta}}_{(i)}^{1} - \overline{\hat{\boldsymbol{\beta}}^{1}}\right)^{\top} = \hat{\mathrm{Var}}_{\mathrm{B}}(\hat{\boldsymbol{\beta}}) - \frac{1}{n}\left(\sum_{i=1}^{n}\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}_{(i)}^{1}\right)\left(\sum_{i=1}^{n}\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}_{(i)}^{1}\right)^{\top},$$

where $\hat{\boldsymbol{\beta}}^1_{(i)}$ is the "one-step approximation" of $\hat{\boldsymbol{\beta}}_{(i)}$ given in Section 2.3.8, with $\hat{\boldsymbol{\beta}}_{(i)}$ representing the estimate of $\boldsymbol{\beta}$ obtained from the dataset in which the ith cluster or subject is excluded, and $\overline{\hat{\boldsymbol{\beta}}^1} = n^{-1}(\hat{\boldsymbol{\beta}}^1_{(1)} + \ldots + \hat{\boldsymbol{\beta}}^1_{(n)})$.

The summary-type method associated with the objects generated by the function glmgee() also allows the user to choose among the five different types of estimates for $\mathrm{Var}(\hat{\boldsymbol{\beta}})$ through its argument varest.

3.6 Comparison of nested models

The package glmtoolbox includes an anova-type method associated with the objects generated by the function glmgee(), which allows the user to compare nested GEE models (that is, it allows the user to assess the hypothesis system $\mathrm{H}_0 : \boldsymbol{\beta}^* = \mathbf{0}$ versus $\mathrm{H}_1 : \boldsymbol{\beta}^* \neq \mathbf{0}$, where the elements of $\boldsymbol{\beta}^*$ are a subset of those of $\boldsymbol{\beta}$, as $\boldsymbol{\beta}^*$ may be written as $\boldsymbol{\beta}^* = \mathbf{L}^\top \boldsymbol{\beta}$, in which $\mathbf{L}$ is a $r \times (p+1)$ contrast matrix) by using not just the Wald test but also the generalized score test (Rotnitzky and Jewell, 1990; Boos, 1992)). The following decision rule may be used to assess the hypothesis system $\mathrm{H}_0 : \boldsymbol{\beta}^* = \mathbf{0}$ versus $\mathrm{H}_1 : \boldsymbol{\beta}^* \neq \mathbf{0}$:

"*Reject* H_0 *at the approximate* $100(\alpha)\%$ *significance level if* $\xi > \chi^2_{1-\alpha}(r)$",

where $\alpha \in (0,1)$, $\chi^2_{1-\alpha}(r)$ is the $100(1-\alpha)$th percentile of the chi-square distribution with r degrees-of-freedom, and ξ is one of the following statistics:

- test="wald". Computes the Wald test, which is based on the following statistic:

$$\xi_W = \left(\mathbf{L}^\top \hat{\boldsymbol{\beta}}\right)^\top \left(\mathbf{L}^\top \hat{\mathrm{V}}\mathrm{ar}_{\mathrm{R}}(\hat{\boldsymbol{\beta}})\mathbf{L}\right)^{-1} \left(\mathbf{L}^\top \hat{\boldsymbol{\beta}}\right).$$

- test="score". Computes the generalized score test, whose statistic, denoted here by ξ_S, reduces to the following expression evaluated at the estimate of $\boldsymbol{\beta}$ obtained under the restriction given by H_0 (that is, the estimate of $\boldsymbol{\beta}$ restricted to $\boldsymbol{\beta}^* = \mathbf{0}$):

$$\mathbf{U}^\top(\boldsymbol{\beta})\left[\hat{\mathrm{V}}\mathrm{ar}_{\mathrm{M}}(\hat{\boldsymbol{\beta}})\mathbf{L}\left(\mathbf{L}^\top \hat{\mathrm{V}}\mathrm{ar}_{\mathrm{R}}(\hat{\boldsymbol{\beta}})\mathbf{L}\right)^{-1}\mathbf{L}^\top \hat{\mathrm{V}}\mathrm{ar}_{\mathrm{M}}(\hat{\boldsymbol{\beta}})\right]\mathbf{U}(\boldsymbol{\beta}).$$

The packages gee and geeM do not include an anova-type method. On the other hand, the anova-type method available in geepack() allows the user to compare nested models using just the Wald test.

3.7 Criteria for choosing a working correlation structure

The selection criteria available in glmtoolbox are the following:

- Quasi-likelihood under Independence model Criterion (Pan, 2001):

$$\mathrm{QIC} = -2\sum_{i=1}^{n}\sum_{j=1}^{n_i}\int_{y_{ij}}^{\hat{\mu}_{ij}} \omega_{ij}\frac{(y_{ij}-\mu_{ij})}{\hat{\phi}\,\mathrm{V}(\mu_{ij})}d\mu_{ij} + 2\,\mathrm{trace}\left\{\hat{\phi}^{-1}(\mathbf{X}^\top \hat{\mathbf{K}}\hat{\mathbf{A}}^{-1}\hat{\mathbf{K}}\mathbf{X})\hat{\mathrm{V}}\mathrm{ar}_{\mathrm{R}}(\hat{\boldsymbol{\beta}})\right\}$$

The expressions for $\int_{y_{ij}}^{\hat{\mu}_{ij}} \omega_{ij}\frac{(y_{ij}-\mu_{ij})}{\mathrm{V}(\mu_{ij})}d\mu_{ij}$ for the variance functions in Table 2.3.1 are listed in Table 9.1 of McCullagh and Nelder (1989).

- Correlation Information Criterion (Hin and Wang, 2009):

$$\mathrm{CIC} = \mathrm{trace}\left\{\hat{\phi}^{-1}(\mathbf{X}^\top \hat{\mathbf{K}}\hat{\mathbf{A}}^{-1}\hat{\mathbf{K}}\mathbf{X})\hat{\mathrm{V}}\mathrm{ar}_{\mathrm{R}}(\hat{\boldsymbol{\beta}})\right\}$$

- Gosho-Hamada-Yoshimura's Criterion (Gosho et al., 2011; Gosho, 2014):

$$\mathrm{GHYC} = \mathrm{trace}\left\{\left[\left(\frac{1}{n}\sum_{i=1}^{n}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i)(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i)^\top\right)\left(\frac{1}{n}\sum_{i=1}^{n}\hat{\phi}\hat{\mathbf{V}}_i\right)^{-1} - \mathbf{I}\right]^2\right\}$$

- Pardo-Alonso's Criterion ([Pardo and Alonso](), 2019):

$$\text{PAC} = \left| \frac{\det\left(\frac{1}{n}\sum_{i=1}^{n}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i)(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i)^{\top}\right)}{\det\left(\frac{1}{n}\sum_{i=1}^{n}\hat{\phi}\hat{\mathbf{V}}_i\right)} - 1 \right|$$

- Rotnitzky-Jewell's Criterion ([Hin et al.](), 2007):

$$\text{RJC} = \left(\left[1 - \frac{\text{trace}(\overline{\text{RJC}})}{p+1}\right]^2 + \left[1 - \frac{\text{trace}(\overline{\text{RJC}}^2)}{p+1}\right]^2\right)^{\frac{1}{2}},$$

where $\overline{\text{RJC}} = \hat{\text{V}}\text{ar}_{\text{R}}(\hat{\boldsymbol{\beta}})[\hat{\text{V}}\text{ar}_{\text{M}}(\hat{\boldsymbol{\beta}})]^{-1}$.

- Akaike-type penalized Gaussian Pseudo-likelihood Criterion ([Carey and Wang](), 2011; [Zhu and Zhu](), 2013; [Fu et al.](), 2018):

$$\text{AGPC} = \sum_{i=1}^{n}\left[n_i\log(2\pi) + \frac{1}{\hat{\phi}}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i)^{\top}\hat{\mathbf{V}}_i^{-1}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i) + \log(\hat{\phi}|\hat{\mathbf{V}}_i|)\right] + 2(p+1+q)$$

- Schwarz-type penalized Gaussian Pseudo-likelihood Criterion ([Carey and Wang](), 2011; [Zhu and Zhu](), 2013; [Fu et al.](), 2018):

$$\text{SGPC} = \sum_{i=1}^{n}\left[n_i\log(2\pi) + \frac{1}{\hat{\phi}}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i)^{\top}\hat{\mathbf{V}}_i^{-1}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i) + \log(\hat{\phi}|\hat{\mathbf{V}}_i|)\right] + \log(n)(p+1+q)$$

The above criteria may be computed for one or more GEE models using the extraction methods `QIC()`, `CIC()`, `GHYC()`, `PAC()`, `RJC()`, `AGPC()` and `SGPC()`.

3.8 Global influence

The dfbeta- and cooks.distance-type extraction methods associated with the objects generated by the function `glmgee()` compute and, optionally, display plots of the statistics dfbeta and Cook's distance, which are "leave-one-out" statistics computed to quantify the effect on the estimates of the parameters of interest of deleting each subject/cluster or observation. If the ith cluster is excluded, these statistics may be expressed as follows:

$$\text{Dfbeta}_{(i)} = \hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}_{(i)} \quad \text{and} \quad \text{CD}_{(i)} = \frac{1}{(p+1)}(\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}_{(i)})^{\top}[\hat{\text{V}}\text{ar}(\hat{\boldsymbol{\beta}})]^{-1}(\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}_{(i)}),$$

respectively, where $\hat{\boldsymbol{\beta}}_{(i)}$ represents the estimate of $\boldsymbol{\beta}$ computed from the dataset in which the ith cluster has been excluded. Similar to the extraction function `vcov()`, the estimate of $\text{Var}(\hat{\boldsymbol{\beta}})$ to be used in the computation of the Cook's distance can be chosen by using the argument `varest` of the function `cooks.distance()`, whose options are `varest="model"`, `varest="robust"`, `varest="df-adjusted'`, `varest="bias-corrected'` and `varest="jackknife"`. To avoid computational burden, the values of $\hat{\boldsymbol{\beta}}_{(i)}$ are replaced by their "one-step approximations", denoted here by $\hat{\boldsymbol{\beta}}_{(i)}^{1}$. Next, the two methods for the computation of the "one-step approximations" available in the dfbeta- and cooks.distance-type extraction methods (through their arguments `method`) are described:

- `method="full"`. $\hat{\boldsymbol{\beta}}_{(i)}$ is replaced by the result of the first iteration of the estimating algorithm of the GEE when it is performed using: (i) the dataset in which the ith cluster has been excluded; and (ii) a starting value which is the solution to the same GEE but computed from the dataset including all clusters (that is, the current $\hat{\boldsymbol{\beta}}$).

- `method="Preisser-Qaqish"`. $\hat{\boldsymbol{\beta}}_{(i)}$ is replaced by the result of the first iteration of the estimating algorithm of the GEE when it is performed using: (i) the dataset in which the ith cluster has been excluded; (ii) a starting value which is the solution to the same GEE but computed from the dataset including all clusters (that is, the current $\hat{\boldsymbol{\beta}}$); and (iii) the working correlation matrix is assumed to be known and equal to its current estimate. According to [Preisser and Qaqish]() (1996); [Hammill and Preisser]() (2006), $\hat{\boldsymbol{\beta}}_{(i)}^{1}$ reduces to

$$\hat{\boldsymbol{\beta}}_{(i)}^{1} = \hat{\boldsymbol{\beta}} - \left(\mathbf{X}^{\top}\hat{\mathbf{W}}\mathbf{X}\right)^{-1}\mathbf{X}_i^{\top}\hat{\mathbf{W}}_i\hat{\mathbf{K}}_i^{-1}\widetilde{\mathbf{e}}_i.$$

Thus, $\hat{\mathrm{Var}}_{\mathrm{B}}(\hat{\boldsymbol{\beta}}) = \sum_{i=1}^{n} (\mathrm{Dfbeta}_{(i)})(\mathrm{Dfbeta}_{(i)})^{\top}$. Similar, but more complicated closed-form expressions for $\hat{\boldsymbol{\beta}}_{(ij)}^{1}$ are given in Preisser and Qaqish (1996) and Hammill and Preisser (2006) when observations instead of clusters/subjects have been excluded from the dataset. The statistics based on those values of $\hat{\boldsymbol{\beta}}_{(ij)}$ may be obtained by specifying level="observations" when using the extraction methods dfbeta() and cooks.distance().

3.9 Local influence

The localInfluence-type extraction method associated with the objects generated by the function glmgee() computes and, optionally, displays plots of some local influence measures based on the approach proposed by Cook (1986). Let $\mathbf{u}$ be a set of perturbations applied to the model and/or the data. The resulting estimating equations are denoted by $\mathbf{U}(\boldsymbol{\beta}|\mathbf{u})$. Similar to Jung (2008), the following is assumed: (*i*) the working correlation matrix is known and equal to its current estimate; (*ii*) the quasi-likelihood function $Q(\boldsymbol{\beta})$ exists such that $\mathbf{U}(\boldsymbol{\beta})$ is its gradient; and (*iii*) $\mathbf{u}_0$ exists, such that $\mathbf{U}(\boldsymbol{\beta}|\mathbf{u}_0)$ and $\mathbf{U}(\boldsymbol{\beta})$ coincide. So, the influence of the set of perturbations $\mathbf{u}$ on the estimate of $\boldsymbol{\beta}$ may be assessed by studying the conformal normal curvature, $C_{\mathbf{d}}$, around $\mathbf{u}_0$, along a unitary direction $\mathbf{d}$, in which $C_{\mathbf{d}} = 2|\mathbf{d}^{\top} \mathbf{C} \, \mathbf{d}|$ and

$$\mathbf{C} = \hat{\boldsymbol{\Delta}}^{\top}[-\ddot{\mathbf{Q}}(\hat{\boldsymbol{\beta}})]^{-1}\hat{\boldsymbol{\Delta}},$$

where $\hat{\boldsymbol{\Delta}}$ corresponds to the $(p+1) \times \dim(\mathbf{u})$ matrix given by $\dfrac{\partial \mathbf{U}(\boldsymbol{\beta}|\mathbf{u})}{\partial \mathbf{u}^{\top}} = (\Delta_1, \ldots, \Delta_n)$ evaluated at $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}$ and $\mathbf{u} = \mathbf{u}_0$, and $\ddot{\mathbf{Q}}(\hat{\boldsymbol{\beta}})$ corresponds to the $(p+1) \times (p+1)$ matrix given by $\dfrac{\partial \mathbf{U}(\boldsymbol{\beta}|\mathbf{u})}{\partial \boldsymbol{\beta}^{\top}}$ evaluated at $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}$ and $\mathbf{u} = \mathbf{u}_0$. The matrix $\ddot{\mathbf{Q}}(\boldsymbol{\beta})$ may be written as follows

$$\ddot{\mathbf{Q}}(\boldsymbol{\beta}) = \sum_{i=1}^{n} \mathbf{X}_i^{\top} \left[\mathrm{diag}\{\mathbf{D}_{(\mathbf{a}_i)} \mathbf{V}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i)\} + \mathbf{K}_i \mathbf{V}_i^{-1} \mathbf{D}_{(\mathbf{b}_i)} \right] \mathbf{X}_i$$

in which $\mathbf{D}_{(\mathbf{a}_i)}$ and $\mathbf{D}_{(\mathbf{b}_i)}$ are diagonal matrices with diagonal elements given by $a_{i1}, \ldots, a_{in_i}$ and $b_{i1}, \ldots, b_{in_i}$, respectively, where

$$a_{ij} = -\frac{1}{[g'(\mu_{ij})]^2} \left[\frac{g''(\mu_{ij})}{g'(\mu_{ij})} + \frac{\mathrm{V}'(\mu_{ij})}{2\mathrm{V}(\mu_{ij})} \right] \quad \text{and} \quad b_{ij} = -\frac{1}{g'(\mu_{ij})} \left[1 + \frac{(y_{ij} - \mu_{ij})\mathrm{V}'(\mu_{ij})}{2\mathrm{V}(\mu_{ij})} \right].$$

Next, the options for the perturbation schemes available in the extraction method localInfluence() (through its argument perturbation) are described:

- Case weight perturbations

    (1) Clusters (perturbation="cw-clusters"):

    $$\mathbf{U}(\boldsymbol{\beta}|\mathbf{u}) = \sum_{i=1}^{n} u_i \mathbf{X}_i^{\top} \mathbf{K}_i \mathbf{V}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i).$$

    Therefore, $\dim(\mathbf{u}) = n$, $\mathbf{u}_0 = (1, \ldots, 1)^{\top}$ and $\hat{\Delta}_i = \mathbf{X}_i \hat{\mathbf{K}}_i \hat{\mathbf{V}}_i^{-1}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i)$ is a $(p+1) \times 1$ matrix.

    (2) Observations (perturbation="cw-observations")

    $$\mathbf{U}(\boldsymbol{\beta}|\mathbf{u}) = \sum_{i=1}^{n} \mathbf{X}_i^{\top} \mathbf{K}_i \mathrm{diag}(\mathbf{u}_i) \mathbf{V}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i),$$

    in which $\mathbf{u}_i = (u_{i1}, \ldots, u_{in_i})^{\top}$. Therefore, $\dim(\mathbf{u}) = N$, $\mathbf{u}_0 = (1, \ldots, 1)^{\top}$ and $\hat{\Delta}_i = \mathbf{X}_i^{\top} \mathrm{diag}\{\hat{\mathbf{K}}_i \hat{\mathbf{V}}_i^{-1}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i)\}$ is a $(p+1) \times n_i$ matrix.

- Response perturbation (perturbation="response"):

    If the response is continuous, then the value of y_{ij} is perturbed by adding a quantity which is proportional to the standard deviation of Y_{ij}, thus,

    $$\mathbf{U}(\boldsymbol{\beta}|\mathbf{u}) = \sum_{i=1}^{n} \mathbf{X}_i^{\top} \mathbf{K}_i \mathbf{V}_i^{-1}(\mathbf{y}_i + \sqrt{\phi}[\mathrm{diag}(\mathbf{V}_i)]^{\frac{1}{2}} \mathbf{u}_i - \boldsymbol{\mu}_i),$$

    in which $\mathbf{u}_i = (u_{i1}, \ldots, u_{in_i})^{\top}$. Therefore, $\dim(\mathbf{u}) = N$, $\mathbf{u}_0 = (0, \ldots, 0)^{\top}$ and $\hat{\Delta}_i = \hat{\phi}^{\frac{1}{2}} \mathbf{X}_i^{\top} \hat{\mathbf{K}}_i \hat{\mathbf{V}}_i^{-1}[\mathrm{diag}(\hat{\mathbf{V}}_i)]^{\frac{1}{2}}$ is a $(p+1) \times n_i$ matrix.

- Perturbation of covariates (perturbation= "covariate"):
  If the r-column of $\mathbf{X}$ corresponds to a continuous covariate, then the values of that covariate are perturbed by adding a quantity which is proportional to its standard deviation, c_r, thus,

$$\mathbf{U}(\boldsymbol{\beta}|\mathbf{u}) = \sum_{i=1}^{n} \left(\mathbf{X}_i + \mathbf{u}_i \boldsymbol{\delta}_r^{\top}\right) \mathbf{K}_i \mathbf{V}_i^{-1} \left(\mathbf{y}_i - g^{-1}\left[\mathbf{z}_i + (\mathbf{X}_i + \mathbf{u}_i \boldsymbol{\delta}_r^{\top})\boldsymbol{\beta}\right]\right),$$

in which $\mathbf{u}_i = (u_{i1}, \ldots, u_{in_i})^{\top}$ and $\boldsymbol{\delta}_r$ is a $(p+1)$-dimensional column vector of zeros with the known constant c_r in the r-th position. Therefore, $\dim(\mathbf{u}) = N$, $\mathbf{u}_0 = (0, \ldots, 0)^{\top}$ and $\Delta_i = c_r \boldsymbol{\delta}_r \left\{ (\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i)^{\top} \hat{\mathbf{V}}_i^{-1} \hat{\mathbf{K}}_i + (\mathbf{X}_i \hat{\boldsymbol{\beta}})^{\top} \left[\hat{\mathbf{M}}_i \hat{\mathbf{V}}_i^{-1} (\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i) - \hat{\mathbf{K}}_i \hat{\mathbf{V}}_i^{-1} \hat{\mathbf{K}}_i \mathbf{1}_{n_i} \right] \mathbf{1}_{n_i}^{\top} \right\}$ is a $(p+1) \times n_i$ matrix, where $\mathbf{M}_i$ is a diagonal matrix with diagonal elements $m_{i1}, \ldots, m_{in_i}$, with $m_{ij} = -g''(\mu_{ij})/[g'(\mu_{ij})]^3$.

The options to study the local influence from $C_{\mathbf{d}}$ available in the extraction method localInfluence() (through its argument type) are described as follows:

- type="local". Computes and, optionally, displays an index plot of the elements of $\mathbf{d}_{\max}$ (that is, the eigenvector which correspond to the maximum absolute eigenvalue of $\mathbf{C}$). The vector $\mathbf{d}_{\max}$ is computed using the power iteration algorithm.

- type="total". Computes and, optionally, displays an index plot of the elements of the main diagonal of $\mathbf{C}$.

3.10 Variable selection

The function stepCriterion() associated with the objects generated by the function glmgee() allows the user to iteratively choose the more "relevant" and/or " significant" variables and/or effects in the model fit by using either of the following "hybrid stepwise" strategies (see James et al. (2013), page 212):

- direction="forward". The "hybrid forward stepwise" strategy starts with the simplest model (which may be set at the argument scope, and by default, is a model whose parameters in the linear predictor, except the intercept, if there is, are set to be 0), and then the candidate models are built by hierarchically including effects in the linear predictor, whose "relevance" and/or "importance" in the model fit is assessed by comparing nested models (that is, by comparing the models with and without the included effect) using a criterion previously specified. If an effect is included in the model, this strategy may also remove any effect which, according to the previously specified criterion, no longer contributes to an improvement in the model fit.

- direction="backward". The "hybrid backward stepwise" strategy starts with the more complex model (which may be specified at the argument scope), and then the candidate models are built by hierarchically removing effects in the linear predictor, whose "relevance" and/or "importance" in the model fit is assessed by comparing nested models (that is, by comparing the models with and without the excluded effect) using a criterion previously chosen. If an effect is excluded from the model, then this strategy may also add any effect which, according to the criterion previously specified, provides an improvement in the model fit.

The available comparison criteria are the following:

- criterion="qic". QIC

- criterion="qicu". According to Pan (2001), the QICu may be written as

$$\text{QICu} = -2\sum_{i=1}^{n} \sum_{j=1}^{n_i} \int_{y_{ij}}^{\hat{\mu}_{ij}} \omega_{ij} \frac{(y_{ij} - \mu_{ij})}{\hat{\phi} \, \mathrm{V}(\mu_{ij})} d\mu_{ij} + 2(p+1)$$

- criterion="agpc". AGPC

- criterion="sgpc". SGPC

- criterion="p-value". p-values of the Wald (test="wald") or generalized score (test="score") tests.

According to Xu et al. (2019), the AGPC and SGPC outperform other existing methods in selecting variables, and they perform well regardless of whether the working correlation structure is correctly specified or not.

3.11 Leverage

The leverage-type extraction method associated with the objects generated by the function `glmgee()` computes and, optionally, displays a plot of the leverage measures at the cluster- and observation-level. According to Preisser and Qaqish (1996); Hammill and Preisser (2006), the observation leverage for the jth observation of the ith cluster is the value of the jth diagonal element of the matrix $\hat{\mathbf{H}}_i$. Cluster leverage for cluster i is the mean of the observation leverages. This, unlike the sum of observation leverages, makes the leverage measures comparable when there are clusters of different sizes. The leverage at cluster- and observation-level may be obtained from the leverage-type extraction method (`leverage()`) by specifying `level="clusters"` and `level="observations"`, respectively.

3.12 Residuals

The residuals-type extraction method associated with the objects generated by the function `glmgee()` computes and, optionally, displays a plot of three different types of residuals. The user may specify the residual type through the argument `type` of the residuals-type method. The residuals are computed to quantify the goodness-of-fit of the model at the cluster-level (Mahalanobis-type residuals) and at the observation-level (Pearson- and deviance-type residuals). Indeed, a plot of the Pearson- or deviance-type residuals versus the fitted values may be a useful tool to assess if, for instance, the specified variance function provides a suitable description of the dispersion present in the data. As follows, three types of residuals are described:

- `type="pearson"`. Computes the Pearson-type residuals, given by

$$r_{ij}^P = \frac{y_{ij} - \hat{\mu}_{ij}}{\sqrt{\hat{\phi}\,\mathrm{V}(\hat{\mu}_{ij})/\omega_{ij}}} \qquad \text{for} \quad i = 1,\ldots,n \quad \text{and} \quad j = 1,\ldots,n_i.$$

- `type="deviance"`. Computes the deviance-type residuals, given by

$$r_{ij}^D = \mathrm{sign}(y_{ij} - \hat{\mu}_{ij})\sqrt{d(y_{ij}, \hat{\mu}_{ij}, \omega_{ij})/\hat{\phi}} \qquad \text{for} \quad i = 1,\ldots,n \quad \text{and} \quad j = 1,\ldots,n_i,$$

  where $d(y_{ij}, \mu_{ij}, \omega_{ij})$ represents the contribution to the non-scaled deviance of the jth measurement performed on the ith subject or cluster.

- `type="mahalanobis"`. Computes the Mahalanobis-type residuals, given by

$$r_i^M = n_i^{-1}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i)^\top [\hat{\mathrm{V}}\mathrm{ar}(Y_i)]^{-1}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i) = \frac{n_i^{-1}}{\hat{\phi}}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i)^\top \hat{\mathbf{V}}_i^{-1}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i) \qquad \text{for} \quad i = 1,\ldots,n.$$

The residuals-type extraction method in geepack provides neither deviance- nor Mahalanobis-type residuals computation, whereas gee and geeM do not include a residuals-type extraction method.

4 Extensions

4.1 Nonlinear predictors

Unlike that described in expression (1), where $g(\mu_{ij})$ is restricted to being a linear combination of the elements of the parameter vector $\boldsymbol{\beta}$, in the new model formulation described here, $g(\mu_{ij})$ may be expressed using a more general family of functions of $\boldsymbol{\beta} = (\beta_1,\ldots,\beta_p)^\top$. That is,

$$g(\mu_{ij}) = \eta(\mathbf{x}_{ij}, \boldsymbol{\beta}),$$

where $\eta_{ij}(\boldsymbol{\beta}) \equiv \eta(\mathbf{x}_{ij}, \boldsymbol{\beta})$ is a continuous and twice differentiable function of $\boldsymbol{\beta}$, with $\mathbf{x}_{ij}$ being a vector of continuous and/or discrete regressors. The estimate of $\boldsymbol{\beta}$ can be obtained by solving the p non-linear equations given by $\mathbf{U}(\hat{\boldsymbol{\beta}}) = \mathbf{0}$, where

$$\mathbf{U}(\boldsymbol{\beta}) = \phi^{-1}\sum_{i=1}^n \mathbf{D}_i^\top \mathbf{K}_i \mathbf{V}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i) = \phi^{-1}\sum_{i=1}^n \mathbf{D}_i^\top \mathbf{W}_i \mathbf{K}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i) = \phi^{-1}\mathbf{D}^\top \mathbf{W}\mathbf{K}^{-1}(\mathbf{y} - \boldsymbol{\mu}),$$

in which $\mathbf{D} = (\mathbf{D}_1^\top,\ldots,\mathbf{D}_n^\top)^\top$, $\mathbf{D}_i = (\mathbf{d}_{i1},\ldots,\mathbf{d}_{in_i})^\top$ and $\mathbf{d}_{ij} = (\partial\eta_{ij}(\boldsymbol{\beta})/\partial\beta_1,\ldots,\partial\eta_{ij}(\boldsymbol{\beta})/\partial\beta_p)^\top$. This type of GEE model is implemented in the function `gnmgee()` of the package glmtoolbox. The arguments of the function `gnmgee()` are very similar to those of `glmgee()`. Nevertheless, the form of the non-linear function $\eta_{ij}(\boldsymbol{\beta})$ and the starting value for $\boldsymbol{\beta}$ in the estimation algorithm must be set by the user

via the arguments `formula` and `start` of `gnmgee()`. But, the argument `formula` also accepts built-in non-linear functions such as `SSasymp()`, `SSasympOff()`, `SSasympOrig()`, `SSbiexp()`, `SSfol()`, `SSfpl()`, `SSgompertz()`, `SSlogis()`, `SSmicmen()` and `SSweibull()`, which do not require user-supplied starting values.

4.2 Weighted GEE methods

In longitudinal studies, in which the response variable is planned to be measured at J time points on each subject/cluster, the weighted GEE methods provide consistent estimates under the MAR assumption when the missing data pattern is dropout and its mechanism is correctly specified (Robins et al., 1995). Let $t_{ij} = 1$ if the response is observed on the ith subject/cluster at time j, and 0 otherwise, which is assumed to be a realization of a random variable denoted here by T_{ij}. In addition, let $t_i = 1 + t_{i1} + t_{i2} + \ldots + t_{iJ}$ the time of dropout for the ith subject/cluster, so, $t_i \in \{2, \ldots, J+1\}$. The probability of observing y_{ij} may be expressed using the following logistic model:

$$\text{logit}(\pi_{ij}) = \text{logit}\Big(\Pr\Big[T_{ij} = 1 \mid T_{i,j-1} = 1, \mathbf{x}_{i1}, \ldots, \mathbf{x}_{ij}, Y_{i1}, \ldots, Y_{i,j-1}\Big]\Big) = \mathbf{z}_{ij}^\top \boldsymbol{\tau},$$

where $\boldsymbol{\tau} = (\tau_0, \tau_1, \ldots, \tau_s)^\top$ is an unknown parameter vector and $\mathbf{z}_{ij} = (1, z_{ij1}, \ldots, z_{ijs})$ is a vector of regressors which may include visit indicator variables ($I(j = 2), \ldots, I(j = J-1)$, where $I(\cdot)$ is the indicator function), covariates ($\mathbf{x}_{i1}, \ldots, \mathbf{x}_{ij}$) and past responses ($y_{i1}, \ldots, y_{i,j-1}$). The maximum likelihood estimate of $\boldsymbol{\tau}$ is $\hat{\boldsymbol{\tau}} = \underset{\boldsymbol{\tau} \in \mathbb{R}^{s+1}}{\text{argmax}}\, \ell(\boldsymbol{\tau})$, where (Robins et al., 1995; Preisser et al., 2002)

$$\ell(\boldsymbol{\tau}) = \sum_{i=1}^{n} \sum_{j=1}^{t_i^*} t_{i,j-1}\Big[t_{ij} \log(\pi_{ij}) + (1 - t_{ij}) \log(1 - \pi_{ij})\Big]$$

is the log-likelihood function of $\boldsymbol{\tau}$, in which $t_i^* = \min(t_i, J)$ and $t_{i0} = 0$. If the missing data pattern is dropout, then $t_{i1} = 1$ is assumed for all $i = 1, \ldots, n$. Therefore,

$$\hat{\pi}_{ij} = \begin{cases} 1 & \text{if } j = 1 \\[2mm] \dfrac{\exp(\mathbf{z}_{ij}^\top \hat{\boldsymbol{\tau}})}{1 + \exp(\mathbf{z}_{ij}^\top \hat{\boldsymbol{\tau}})} & \text{if } j > 1 \end{cases}$$

The estimate of $\boldsymbol{\tau}$ satisfies $\dfrac{\partial \ell(\boldsymbol{\tau})}{\partial \boldsymbol{\tau}}\Big|_{\boldsymbol{\tau}=\hat{\boldsymbol{\tau}}} = \mathbf{S}_1 + \ldots + \mathbf{S}_n = \mathbf{0}$, where $\mathbf{S}_i = \mathbf{Z}_i^\top(\mathbf{t} - \hat{\boldsymbol{\pi}})$, $\mathbf{Z}_i = (\mathbf{z}_{i1}, \ldots, \mathbf{z}_{it_i^*})^\top$, $\mathbf{t} = (t_{i1}, \ldots, t_{it_i^*})^\top$ and $\hat{\boldsymbol{\pi}} = (\hat{\pi}_{i1}, \ldots, \hat{\pi}_{it_i^*})^\top$.

Observation-specific weights

According to Fitzmaurice et al. (2011), the underlying idea of this weighting method is to base the estimation on the observed responses but weight them to account for the probability of remaining in the study. When using the observation-specific weighted GEE method, the covariates for all occasions for a subject/cluster must be observed, regardless of whether the response is missing. That is, the input data set must contain J observations for each subject/cluster. The estimate of $\boldsymbol{\beta}$ is the solution to the $(p+1)$ nonlinear equations given by $\mathbf{U}^*(\hat{\boldsymbol{\beta}}) = \mathbf{0}$, in which $\mathbf{U}^*(\boldsymbol{\beta})$ may be expressed as follows

$$\mathbf{U}^*(\boldsymbol{\beta}) = \phi^{-1} \sum_{i=1}^{n} \mathbf{X}_i^\top \mathbf{K}_i \mathbf{V}_i^{-1} \boldsymbol{\Lambda}_i (\mathbf{y}_i - \boldsymbol{\mu}_i),$$

where $\boldsymbol{\Lambda}_i = \text{diag}\{t_{i1}\lambda_{i1}, \ldots, t_{iJ}\lambda_{iJ}\}$ and $\lambda_{ij} = 1/(\hat{\pi}_{i1} \times \hat{\pi}_{i2} \times \ldots \times \hat{\pi}_{ij})$. The estimator of $\boldsymbol{\beta}$ has an asymptotic normal distribution with consistent estimator of its asymptotic variance given by (Robins et al., 1995; Preisser et al., 2002)

$$\hat{\mathbb{V}}\text{ar}(\hat{\boldsymbol{\beta}}) = \left(\sum_{i=1}^{n} \mathbf{X}_i^\top \hat{\mathbf{K}}_i \hat{\mathbf{V}}_i^{-1} \boldsymbol{\Lambda}_i \hat{\mathbf{K}}_i \mathbf{X}_i\right)^{-1} \left(\sum_{i=1}^{n} \mathbf{E}_i \mathbf{E}_i^\top\right) \left(\sum_{i=1}^{n} \mathbf{X}_i^\top \hat{\mathbf{K}}_i \hat{\mathbf{V}}_i^{-1} \boldsymbol{\Lambda}_i \hat{\mathbf{K}}_i \mathbf{X}_i\right)^{-1},$$

where $\mathbf{E}_i = \mathbf{X}_i^\top \hat{\mathbf{K}}_i \hat{\mathbf{V}}_i^{-1} \boldsymbol{\Lambda}_i (\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i) - \left(\sum_{i=1}^{n} \mathbf{X}_i^\top \hat{\mathbf{K}}_i \hat{\mathbf{V}}_i^{-1} \boldsymbol{\Lambda}_i (\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i)\mathbf{S}_i^\top\right)\left(\sum_{i=1}^{n} \mathbf{S}_i \mathbf{S}_i^\top\right)^{-1} \mathbf{S}_i$.

Subject/Cluster-specific weights

In the cluster-specific weighted GEE method, covariates for a cluster who dropout at time k must be observed for occasions up to and including time k. That is, each subject must have at least k observations in the input data set. The estimate of $\boldsymbol{\beta}$ is the solution to the $(p+1)$ nonlinear equations given by $\mathbf{U}(\hat{\boldsymbol{\beta}}) = \mathbf{0}$, in which $\mathbf{U}(\boldsymbol{\beta})$ is given by (2), but where the weights in the matrices $\mathbf{A}_1, \ldots, \mathbf{A}_n$ are set to be $\omega_{ij}^* = \omega_{ij} \times \lambda_i$ for $i = 1, \ldots, n$ and $j = 1, \ldots, t_i - 1$. The value of λ_i may be computed as follows

$$\lambda_i^{-1} = \left(\prod_{j=1}^{t_i-1} \hat{\pi}_{ij}\right)(1 - \hat{\pi}_{it_i})^{I(t_i \leq J)}.$$

The estimator of $\boldsymbol{\beta}$ has an asymptotic normal distribution with a consistent estimator of its asymptotic variance given by (Robins et al., 1995; Preisser et al., 2002)

$$\hat{\mathrm{Var}}(\hat{\boldsymbol{\beta}}) = \left(\sum_{i=1}^{n} \mathbf{X}_i^{\top} \hat{\mathbf{K}}_i \hat{\mathbf{V}}_i^{-1} \hat{\mathbf{K}}_i \mathbf{X}_i\right)^{-1}\left(\sum_{i=1}^{n} \mathbf{E}_i \mathbf{E}_i^{\top}\right)\left(\sum_{i=1}^{n} \mathbf{X}_i^{\top} \hat{\mathbf{K}}_i \hat{\mathbf{V}}_i^{-1} \hat{\mathbf{K}}_i \mathbf{X}_i\right)^{-1},$$

where $\mathbf{E}_i = \mathbf{X}_i^{\top} \hat{\mathbf{K}}_i \hat{\mathbf{V}}_i^{-1}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i) - \left(\sum_{i=1}^{n} \mathbf{X}_i^{\top} \hat{\mathbf{K}}_i \hat{\mathbf{V}}_i^{-1}(\mathbf{y}_i - \hat{\boldsymbol{\mu}}_i)\mathbf{S}_i^{\top}\right)\left(\sum_{i=1}^{n} \mathbf{S}_i \mathbf{S}_i^{\top}\right)^{-1} \mathbf{S}_i$.

The weighted GEE methods based on observation-specific weights and cluster-specific weights are implemented in the function `wglmgee()` of the package `glmtoolbox`, whose arguments are very similar to those of the function `glmgee()`. In that function, the user sets the GEE and missingness models at the same argument of type `Formula` (Zeileis and Croissant, 2010). In addition, the user sets the type of weighting method: observation-specific weights (`level="observations"`) or cluster-specific weights (`level="clusters"`). The `wglmgee()` function estimates the parameters of the missingness model and uses them to compute the required weights. Then, `wglmgee()` introduces the weights in the estimation process of the GEE model parameters as well as in the estimation of their asymptotic variance matrix.

5 Examples

5.1 Growth patterns of trees under two types of atmosphere

The dataset of this example, described by Diggle et al. (2002) and available in the spruces object of `glmtoolbox`, is composed of the columns `tree`, `days`, `size` and `treat` (see Table 1). The analysis of this dataset aims to assess the effect of ozone pollution on tree growth. Ozone pollution is common in urban areas, thus the impact of increased ozone concentrations on tree growth is of considerable interest. The response variable is tree size (`size`), where size is conventionally measured by the product of tree height and stem diameter squared. A total of 79 trees, identified in the dataset by the column `tree`, were considered in this experiment. In the first group a total of 54 trees were grown under an ozone-enriched atmosphere (`treat="ozone-enriched"`), that is, ozone exposure at 70 parts per billion, whereas in the second group, a total of 25 trees were grown under a normal atmosphere (`treat="normal"`). The size of each tree was observed and recorded exactly 13 times across the time since the experiment began (`days`), so the data are balanced and the number of rows in the dataset is 1027. The main objective of the analysis is to compare the growth patterns of trees under two types of atmosphere: normal and ozone-enriched.

Column	Role	Description
`tree`	Cluster/subject identifier	Identifier of the tree
`days`	Explanatory variable	Number of days after the treatment began
`treat`	Explanatory variable	Treatment: "normal" or "ozone-enriched"
`size`	Response variable	Tree size

Table 1: Columns in the object `spruces` of the package `glmtoolbox`.

An adjusted for skewness box-plot of the data (Figure 1), obtained using the function `adjbox()` in the package `robustbase` (Maechler et al. (2022)), shows that ozone suppresses tree growth. The plot also indicates that under the two types of atmosphere the location as well as the dispersion of the tree size are non-decreasing and non-linear functions of the time since the experiment began, which suggests the data should be analyzed using a GEE with the following features: (*i*) a linear predictor which includes a polynomial effect of the time, as well as a dummy variable to indicate the type of atmosphere under which the trees grew; and (*ii*) an increasing variance function such as $\mathrm{V}(\mu) = \mu$,

$V(\mu) = \mu^2$ or $V(\mu) = \mu^3$, which is aimed to include in the model the heteroscedasticity observed in the data.
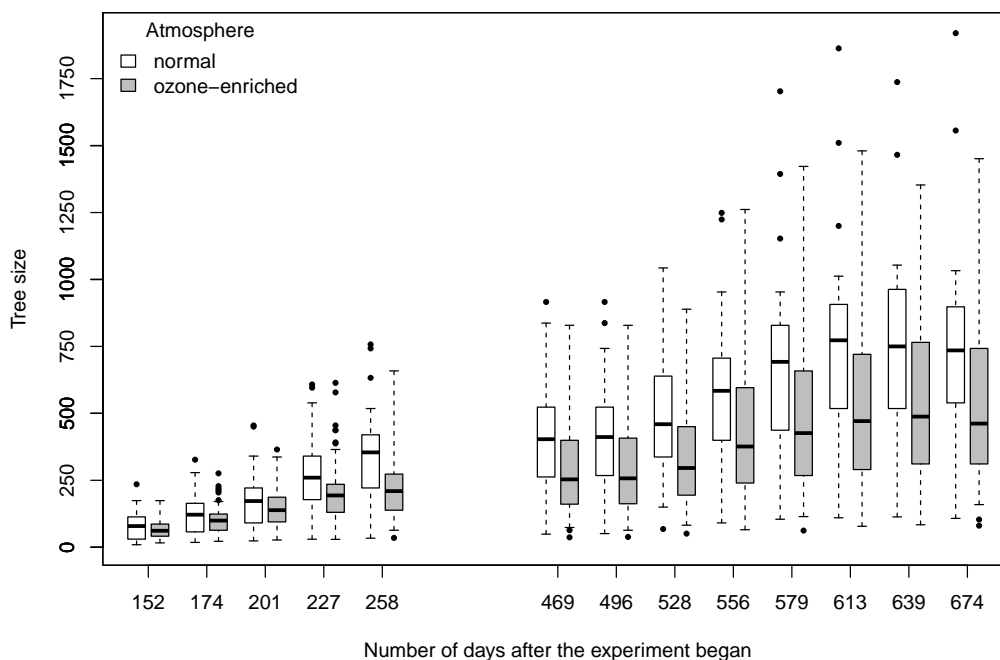

Figure 1: Growth patterns of trees under normal and ozone-enriched atmospheres.

Plots (not shown here) of the deviance-type residuals versus the fitted values for two GEE with logarithmic link, working correlation matrix specified to be the identity matrix, and variance functions $V(\mu) = \mu$ and $V(\mu) = \mu^3$, reveal megaphone shaped and inverted megaphone shaped patterns, respectively, indicating thus how inappropriate such variance functions are for describing the heteroscedasticity present in the data. On the other hand, the same plot but using the variance function $V(\mu) = \mu^2$ (Figure 2(a)) does not reveal any trend, indicating that the data could be suitably analyzed under the assumption of a constant coefficient of variation. So, several GEE with the variance function $V(\mu) = \mu^2$, logarithmic link, and different structures for the working correlation matrix are fitted to the data. Then, the selection criteria available in `glmtoolbox` are used to choose the more suitable structure for the correlation matrix.

```
> data(spruces)
> m1 <- glmgee(size ~ poly(days,4) + treat, id=tree, family=Gamma(log), data=spruces)
> m2 <- update(m1, corstr="Exchangeable")
> m3 <- update(m1, corstr="AR-M-dependent(1)")
> m4 <- update(m1, corstr="AR-M-dependent(2)")
> m5 <- update(m1, corstr="AR-M-dependent(3)")

> a <- CIC(m1, m2, m3, m4, m5, verbose=FALSE)
> b <- QIC(m1, m2, m3, m4, m5, verbose=FALSE)
> c <- GHYC(m1, m2, m3, m4, m5, verbose=FALSE)
> d <- RJC(m1, m2, m3, m4, m5, verbose=FALSE)
> e <- AGPC(m1, m2, m3, m4, m5, verbose=FALSE)
> f <- SGPC(m1, m2, m3, m4, m5, verbose=FALSE)
> cbind(a,QIC=b[,"QIC"],GHYC=c[,"GHYC"],RJC=d[,"RJC"],AGPC=e[,"AGPC"],SGPC=f[,"SGPC"])

  Object      Correlation   CIC   QIC   GHYC    RJC  AGPC  SGPC
1     m1     Independence 23.43 42068 116.42 41.303 13539 13554
2     m2     Exchangeable 23.43 42068  40.96  7.639 11689 11706
3     m3 AR-M-dependent(1) 23.66 42086  11.26  0.129 10941 10957
4     m4 AR-M-dependent(2) 23.56 42158  13.72  0.489 10981 11000
```

```
5     m5 AR-M-dependent(3) 23.56 42201  12.45  0.914 10994 11016
```

Most of the selection criteria (that is, Gosho-Hamada-Yoshimura's criterion, Rotnitzky-Jewell's criterion, Akaike-type penalized Gaussian pseudo-likelihood criterion, and Schwarz-type penalized Gaussian pseudo-likelihood criterion) suggest the first-order autoregressive (AR-1) and Independence as the more and less adequate structures for the correlation matrix, respectively. GEE with the AR-1 structure for the correlation matrix is summarized as follows:

```
> summary(m3)
Sample size
   Number of observations:  1027
       Number of clusters:  79
             Cluster size:  13
**********************************************************************
Model
        Variance function:  Gamma
            Link function:  log
     Correlation structure:  AR-M-dependent(1)
**********************************************************************
Coefficients
                    Estimate Std.Error   z-value Pr(>|z|)
(Intercept)          5.90378   0.10486  56.30321 < 2e-16
poly(days, 4)1      19.20015   0.51848  37.03159 < 2e-16
poly(days, 4)2      -2.85755   0.20585 -13.88147 < 2e-16
poly(days, 4)3       5.41639   0.18246  29.68549 < 2e-16
poly(days, 4)4      -3.57407   0.12478 -28.64405 < 2e-16
treatozone-enriched -0.25861   0.12835  -2.01486 0.043919

Dispersion           0.32866
**********************************************************************
Working correlation
     [1]  [2]  [3]  [4]  [5]  [6]  [7]  [8]  [9]  [10] [11] [12] [13]
[1]  1.00 0.97 0.93 0.90 0.87 0.84 0.81 0.78 0.76 0.73 0.70 0.68 0.66
[2]  0.97 1.00 0.97 0.93 0.90 0.87 0.84 0.81 0.78 0.76 0.73 0.70 0.68
[3]  0.93 0.97 1.00 0.97 0.93 0.90 0.87 0.84 0.81 0.78 0.76 0.73 0.70
[4]  0.90 0.93 0.97 1.00 0.97 0.93 0.90 0.87 0.84 0.81 0.78 0.76 0.73
[5]  0.87 0.90 0.93 0.97 1.00 0.97 0.93 0.90 0.87 0.84 0.81 0.78 0.76
[6]  0.84 0.87 0.90 0.93 0.97 1.00 0.97 0.93 0.90 0.87 0.84 0.81 0.78
[7]  0.81 0.84 0.87 0.90 0.93 0.97 1.00 0.97 0.93 0.90 0.87 0.84 0.81
[8]  0.78 0.81 0.84 0.87 0.90 0.93 0.97 1.00 0.97 0.93 0.90 0.87 0.84
[9]  0.76 0.78 0.81 0.84 0.87 0.90 0.93 0.97 1.00 0.97 0.93 0.90 0.87
[10] 0.73 0.76 0.78 0.81 0.84 0.87 0.90 0.93 0.97 1.00 0.97 0.93 0.90
[11] 0.70 0.73 0.76 0.78 0.81 0.84 0.87 0.90 0.93 0.97 1.00 0.97 0.93
[12] 0.68 0.70 0.73 0.76 0.78 0.81 0.84 0.87 0.90 0.93 0.97 1.00 0.97
[13] 0.66 0.68 0.70 0.73 0.76 0.78 0.81 0.84 0.87 0.90 0.93 0.97 1.00
```

The Wald test and the generalized score test suggest that there is no interaction between time and the type of atmosphere. This is because, as shown below, the p-values associated with that effect are "large".

```
> m3a <- update(m3, . ~ . + poly(days,4):treat)
> anova(m3a, test="wald")
Wald test

Model 1 :  size ~ 1
Model 2 :  size ~ poly(days, 4)
Model 3 :  size ~ poly(days, 4) + treat
Model 4 :  size ~ poly(days, 4) + treat + poly(days, 4):treat

           Chi    df   Pr(>Chi)
1 vs 2 1931.9813   4    < 2e-16 ***
2 vs 3    4.0597   1    0.04392 *
3 vs 4    3.6641   4    0.45336

> anova(m3a, test="score")
```

```
Generalized score test

Model 1 :  size ~ 1
Model 2 :  size ~ poly(days, 4)
Model 3 :  size ~ poly(days, 4) + treat
Model 4 :  size ~ poly(days, 4) + treat + poly(days, 4):treat


          Chi   df   Pr(>Chi)
1 vs 2 61.3028    4   1.544e-12 ***
2 vs 3  3.3687    1     0.06645 .
3 vs 4  3.4665    4     0.48300
```

Variance estimation

Next, the estimates of $\text{Var}(\hat{\beta}_0), \text{Var}(\hat{\beta}_1), \ldots, \text{Var}(\hat{\beta}_5)$ are obtained using four different estimators.

```
> cbind(model=diag(vcov(m3, type="model")),
+       robust=diag(vcov(m3, type="robust")),
+       bias.corrected=diag(vcov(m3, type="bias-corrected")),
+       jackknife=diag(vcov(m3, type="jackknife")))
                    model robust bias.corrected jackknife
(Intercept)        0.0110 0.0110         0.0119    0.0119
poly(days, 4)1     0.2564 0.2688         0.2758    0.2758
poly(days, 4)2     0.0922 0.0424         0.0435    0.0435
poly(days, 4)3     0.0352 0.0333         0.0342    0.0342
poly(days, 4)4     0.0283 0.0156         0.0160    0.0160
treatozone-enriched 0.0159 0.0165        0.0176    0.0176
```

Parameter interpretation

Across time, the expected size of the trees that grew under the ozone-enriched atmosphere is approximately $22.79\% = 100 \times [1 - \exp(\hat{\beta}^{\text{treat}})]$ lower than that of the trees that grew under the normal atmosphere, where β^{treat} represents the parameter associated with the dummy variable indicating the type of atmosphere under which the trees grew. According to the Wald test, the hypothesis $H_0 : \beta^{\text{treat}} \geq 0$ versus $H_1 : \beta^{\text{treat}} < 0$ is rejected at the approximate level of 5%, indicating thus that the ozone-enriched atmosphere suppresses tree growth.

Variable selection

As an illustration, the procedure of variable selection is applied using the strategy "hybrid forward stepwise" with the p-value of the generalized score test as the comparison criterion, and where the thresholds for add and drop effects are set at 10% and 5%, respectively. In addition, the simplest and most complex models are specified to be 1 and 1+poly(days,4)+treat+poly(days,4):treat, respectively. As shown below, the best linear predictor according to the chosen strategy incorporates both time and atmosphere, but not the interaction between them. The same results are obtained when the strategy of variable selection is changed as follows: (i) the generalized score test is replaced by the Wald test; (ii) and the "hybrid forward stepwise" procedure is replaced by the "hybrid backward stepwise".

```
stepCriterion(m3, direction="forward", criterion="p-value", test="score",
+                scope=list(lower=~1, upper=~poly(days,4)*treat), levels=c(0.10,0.05))

    Variance function:  Gamma
        Link function:  log
Correlation structure:  AR-M-dependent(1)
  Comparison criteria:  P(Chisq>)(*)


Initial model:
~ 1


Step 0 :
                  df   QIC  QICu  AGPC  SGPC P(Chisq>)(*)
+ poly(days, 4)    4 39944 39928 10927 10941   1.544e-12
```

```
+ treat                      1 19534 19517 12353 12360      0.03618
<none>                         18998 18989 12362 12367


Step 1 : + poly(days, 4)

                       df   QIC  QICu  AGPC  SGPC P(Chisq>)(*)
+ treat                 1 42086 42051 10941 10957      0.06645
<none>                    39944 39928 10927 10941


Step 2 : + treat

                       df   QIC  QICu  AGPC  SGPC P(Chisq>)(*)
+ poly(days, 4):treat   4 41815 41787 10954 10980       0.483
<none>                    42086 42051 10941 10957


Final model:
~ poly(days, 4) + treat
****************************************************************************
(*) p-values of the generalized score test
 Effects are added when their p-values are lower than 0.1
 Effects are excluded when their p-values are higher than 0.05
```
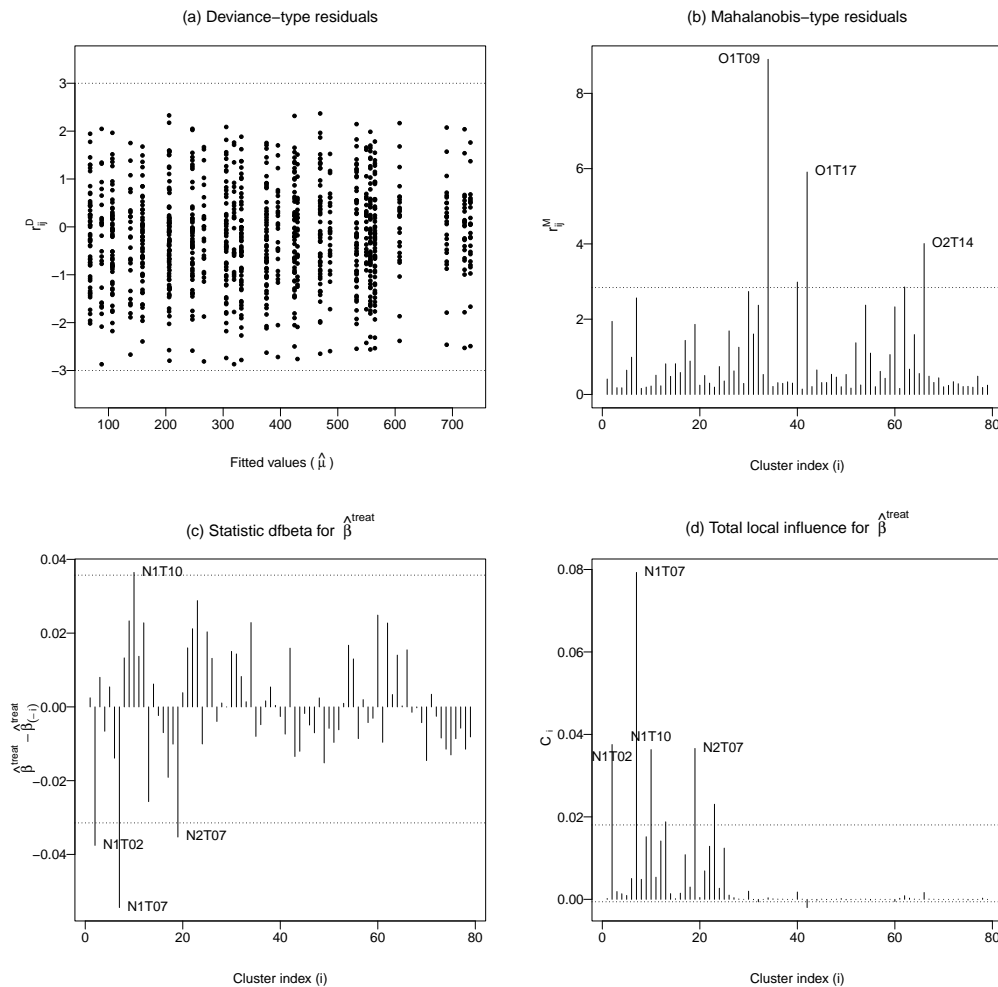

Figure 2: Some diagnostic plots for the GEE with correlation structure AR-1 fitted to the data on growth patterns of trees.

Residual analysis

```
> residuals(m3, type="deviance", plot.it=TRUE, pch=16)
> residuals(m3, type="mahalanobis", plot.it=TRUE, identify=3)
```

According to the Mahalanobis-type residuals (Figure 2(b)) the trees for which the model has the lowest goodness-of-fit are those identified as O1T09, O1T17 and O2T14. Although those trees grew under an ozone-enriched atmosphere, which seems to be associated with expected sizes lower than those of the trees which grew under the normal atmosphere, their observed sizes across the time are even greater than those observed for 70% of the trees which grew under a normal atmosphere.

Global influence

```
> dfbeta(m3, method="full", coefs="treat", identify=4)
```

According to the statistic Dfbeta at the cluster-level for $\hat{\beta}^{\text{treat}}$ (Figure 2(c)), the trees known as N1T02, N1T07 and N2T07 are those providing the greatest evidence which supports the negative effect of the ozone-enriched atmosphere on the growth pattern of the trees, as their exclusion from the dataset leads the estimate of β^{treat} closer to zero. Those trees grew under a normal atmosphere, and their sizes across time are higher than those observed for 90% of the trees growing under such an atmosphere. On the other hand, the tree identified as N1T10 is that providing the greatest evidence against the negative effect of the ozone-enriched atmosphere on the growth pattern of the trees, because its exclusion from the dataset decreases the estimate of β^{treat}. The tree identified as N1T10 grew under a normal atmosphere, however, its size across time is lower than that observed for 90% of the trees growing under the ozone-enriched atmosphere.

Local influence

```
> localInfluence(m3, type="total", perturbation="cw-clusters", coefs="treat",
+                plot.it=TRUE, identify=4)
```

The plot of the total local influence under the case weight perturbation scheme at the cluster-level for $\hat{\beta}^{\text{treat}}$ (Figure 2(d)) highlights the trees identified as N1T07, N1T02, N2T07 and N1T10 as suspected to be influential on the estimate of β^{treat}, which confirms the results of the global influence analysis above.

5.2 Comparison with other GEE solvers

The parameter estimates and the associated standard errors provided by the function `glmgee()` are compared with those generated by the GEE solvers available in the packages gee, geepack and geeM. The results are presented in Table 2. The values provided by the other GEE solvers are very similar to those generated by the function `glmgee()`.

	glmtoolbox	geepack	gee	geeM
(Intercept)	5.904(0.105)	5.903(0.105)	5.904(0.105)	5.904(0.105)
poly(days, 4)1	19.200(0.518)	19.186(0.519)	19.201(0.518)	19.200(0.518)
poly(days, 4)2	-2.858(0.206)	-2.860(0.206)	-2.857(0.206)	-2.858(0.206)
poly(days, 4)3	5.416(0.182)	5.414(0.182)	5.417(0.182)	5.416(0.182)
poly(days, 4)4	-3.574(0.125)	-3.572(0.125)	-3.574(0.125)	-3.574(0.125)
treatozone-enriched	-0.259(0.128)	-0.257(0.128)	-0.259(0.128)	-0.259(0.128)
ρ	0.97	0.97	0.97	0.97

Table 2: Parameter estimates (standard errors) of the GEE model with correlation structure AR-1 fitted to the data on growth patterns of trees.

5.3 Treatment of severe postnatal depression

The dataset of this example, available in the object `depression` of `glmtoolbox` and composed of columns named `subj`, `group`, `visit`, `dep` and `depressd` (see Table 3), arose from a double-blind placebo-controlled study on the efficacy of oestrogen given transdermally for treatment of severe postnatal depression (Gregoire et al., 1996). A total of 61 women with severe depression (identified in the dataset

by the column subj), which began within 3 months of childbirth and persisted for up to 18 months postnatal, were randomly assigned to either a control group (group="placebo") of size 27, which were treated with a placebo patch, or an active treatment group (group="oestrogen") of size 34, which were treated with an oestrogen patch. Prior to therapy, all women were assessed by self-ratings of depressive symptoms on the Edinburgh Postnatal Depression Scale (EPDS), where higher scores are indicative of higher levels of depression. A monthly EPDS (dep) was collected for six months once treatment began (visit). The binary response (depressd) was 1 to indicate severe depression when the EDPS value was greater than or equal to 11, and 0 in other cases. There are missing values in the data, because for some women, just two measurements of the response variable are available. However, those missing values are positioned at the last time positions, so, there are no intermixed missing values, and the argument waves of the function glmgee() is not needed.

Column	Role	Description
subj	Cluster/subject identifier	Identifier of the woman
group	Explanatory variable	Treatment: "placebo" or "oestrogen"
visit	Explanatory variable	Number of months after the treatment began
depressd	Response variable	1 if EDPS $\geq$ 11 and 0 otherwise

Table 3: Columns in the object depression of the package glmtoolbox.

A plot of the data (Figure 3) suggests that oestrogen patches are an effective treatment for postnatal severe depression, as across time, the proportion of women with severe depression is lower in the group treated with oestrogen patches than in the group treated with placebo patches. The plot also indicates that the (logit of the) proportion of women with severe depression decreases linearly as a function of the time since the therapy began, but the rate of decreasing seems to be independent of the type of patch (placebo or oestrogen), which suggests that the data could be analyzed by using a GEE with the logit link function and a linear predictor including the effects of time and type of patch, but without the interaction between them.
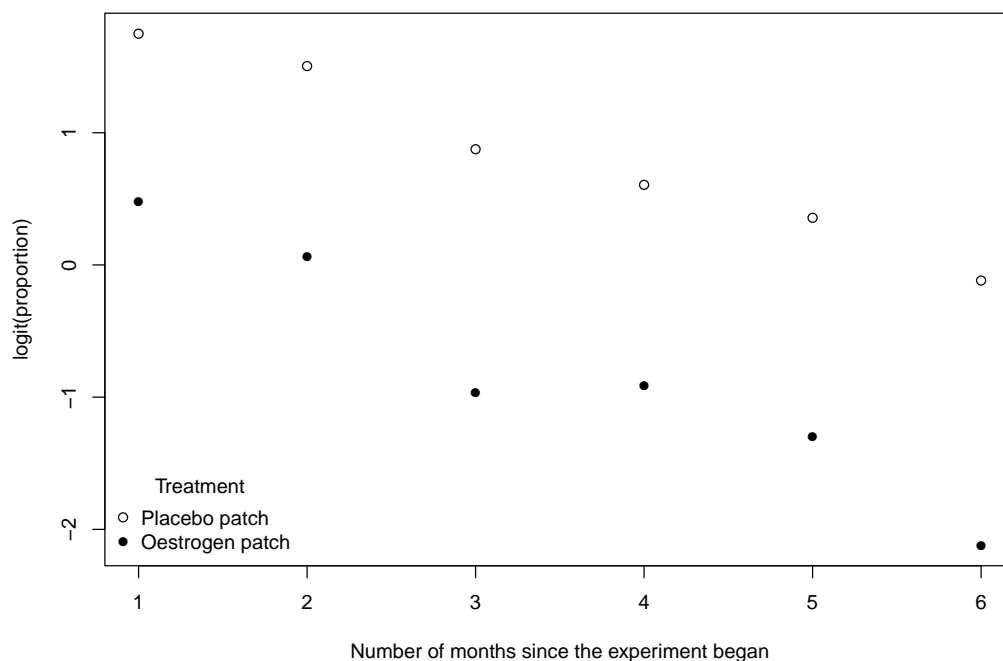
Figure 3: Evolution of the (logit of the) proportion of women with severe depression.

Several GEE with variance function $V(\mu) = \mu(1 - \mu)$, logit link function, and different structures for the working correlation matrix are fitted to the data. Then, some selection criteria available in glmtoolbox are used to choose the more suitable structure for the correlation matrix.

```
> data(depression)
```

```
> m1 <- glmgee(depressd ~ visit + group, id=subj, family=binomial(logit), data=depression)
> m2 <- update(m1, corstr="Exchangeable")
> m3 <- update(m1, corstr="AR-M-dependent(1)")
> m4 <- update(m1, corstr="AR-M-dependent(2)")
> m5 <- update(m1, corstr="AR-M-dependent(3)")

> a <- CIC(m1, m2, m3, m4, m5, verbose=FALSE)
> b <- QIC(m1, m2, m3, m4, m5, verbose=FALSE)
> c <- AGPC(m1, m2, m3, m4, m5, verbose=FALSE)
> d <- SGPC(m1, m2, m3, m4, m5, verbose=FALSE)
> cbind(a,QIC=b[,"QIC"],AGPC=c[,"AGPC"],SGPC=d[,"SGPC"])

  Object        Correlation  CIC    QIC      AGPC      SGPC
1    m1        Independence 7.708 383.555 304.2907 310.6233
2    m2        Exchangeable 8.048 377.815 247.9647 256.4082
3    m3 AR-M-dependent(1) 6.971 358.244 234.4696 242.9131
4    m4 AR-M-dependent(2) 7.031 363.784 234.9438 245.4982
5    m5 AR-M-dependent(3) 7.230 366.387 231.5530 244.2183
```

According to most of the selection criteria, the AR-1 structure is more suitable. Here is the summary of the GEE with that structure for the working correlation matrix:

```
> summary(m3)
Sample size
   Number of observations:  356
       Number of clusters:  61
                              Min 25% 50% 75% Max
           Cluster sizes:    2    4    7    7    7
**********************************************************************
Model
        Variance function:  binomial
            Link function:  logit
    Correlation structure:  AR-M-dependent(1)
**********************************************************************
Coefficients
               Estimate Std.Error  z-value   Pr(>|z|)
(Intercept)     3.23604   0.51842  6.24218 4.3152e-10
visit          -0.62632   0.07477 -8.37681 < 2.22e-16
groupoestrogen -1.77723   0.54578 -3.25631  0.0011287

Dispersion    1.02842
**********************************************************************
Working correlation
     [1]   [2]   [3]   [4]   [5]   [6]   [7]
[1] 1.000 0.513 0.263 0.135 0.069 0.036 0.018
[2] 0.513 1.000 0.513 0.263 0.135 0.069 0.036
[3] 0.263 0.513 1.000 0.513 0.263 0.135 0.069
[4] 0.135 0.263 0.513 1.000 0.513 0.263 0.135
[5] 0.069 0.135 0.263 0.513 1.000 0.513 0.263
[6] 0.036 0.069 0.135 0.263 0.513 1.000 0.513
[7] 0.018 0.036 0.069 0.135 0.263 0.513 1.000
```

The Wald test and the generalized score test indicate that there is no interaction between time and the type of patch. As shown below, the p-values associated with that effect are "large".

```
> m3a <- update(m3, . ~ . + visit:group)
> anova(m3a, test="wald")
Wald test

Model 1 :  depressd ~ 1
Model 2 :  depressd ~ visit
Model 3 :  depressd ~ visit + group
Model 4 :  depressd ~ visit + group + visit:group

        Chi    df   Pr(>Chi)
```

```
1 vs 2 88.1275   1  < 2.2e-16 ***
2 vs 3 10.6036   1   0.001129 **
3 vs 4  2.2104   1   0.137082


> anova(m3a, test="score")
Generalized score test

Model 1 :  depressd ~ 1
Model 2 :  depressd ~ visit
Model 3 :  depressd ~ visit + group
Model 4 :  depressd ~ visit + group + visit:group


        Chi   df   Pr(>Chi)
1 vs 2 39.9226   1  2.642e-10 ***
2 vs 3 10.9208   1  0.0009509 ***
3 vs 4  2.3977   1  0.1215150
```

Variance estimation

Next, the estimates of $\mathrm{Var}(\hat{\beta}_0), \mathrm{Var}(\hat{\beta}_1), \mathrm{Var}(\hat{\beta}_2)$ are obtained using four different estimators.

```
> cbind(model=diag(vcov(m3, type="model")),
+       robust=diag(vcov(m3, type="robust")),
+       bias.corrected=diag(vcov(m3, type="bias-corrected")),
+       jackknife=diag(vcov(m3, type="jackknife")))
                 model  robust bias.corrected jackknife
(Intercept)     0.26219 0.26875        0.29023   0.29023
visit           0.00844 0.00559        0.00583   0.00583
groupoestrogen  0.21114 0.29788        0.32441   0.32441
```

Parameter interpretation

Regardless of the type of patch (placebo or oestrogen), the probability of severe depression decreases across time. However, the odds of severe depression of women treated with oestrogen patches is approximately $83.09\% = 100 \times [1 - \exp(\hat{\beta}^{\mathrm{group}})]$ lower than that of women treated with placebo patches, where β^{group} represents the parameter associated with the dummy variable indicating the type of patch the women were treated with.

Variable selection

As an illustration, the procedure of variable selection is applied using the strategy "hybrid forward stepwise" with the QIC as the comparison criterion. In addition, the simplest and most complex models are specified to be 1 and 1 + visit + group + visit*group, respectively. According to this strategy, the "best" linear predictor consists of the effects of time and type of patch, but without the interaction between them. The same results are obtained in the following scenarios: (*i*) the "hybrid forward stepwise" is replaced by the "hybrid backward stepwise"; and (*ii*) the comparison criterion based on the QIC is replaced by the AGPC.

```
> stepCriterion(m3a, direction="forward", criterion="qic")

    Variance function:  binomial
        Link function:  logit
Correlation structure:  AR-M-dependent(1)
  Comparison criteria:  QIC

Initial model:
~ 1


Step 0 :
            df    QIC   QICu    AGPC   SGPC P(Chisq>)(*)
+ visit      1 413.89 409.23  273.13 279.46    < 2.2e-16
+ group      1 443.77 440.13  366.77 373.10     0.000424
<none>         465.70 463.63  385.09 389.31
```
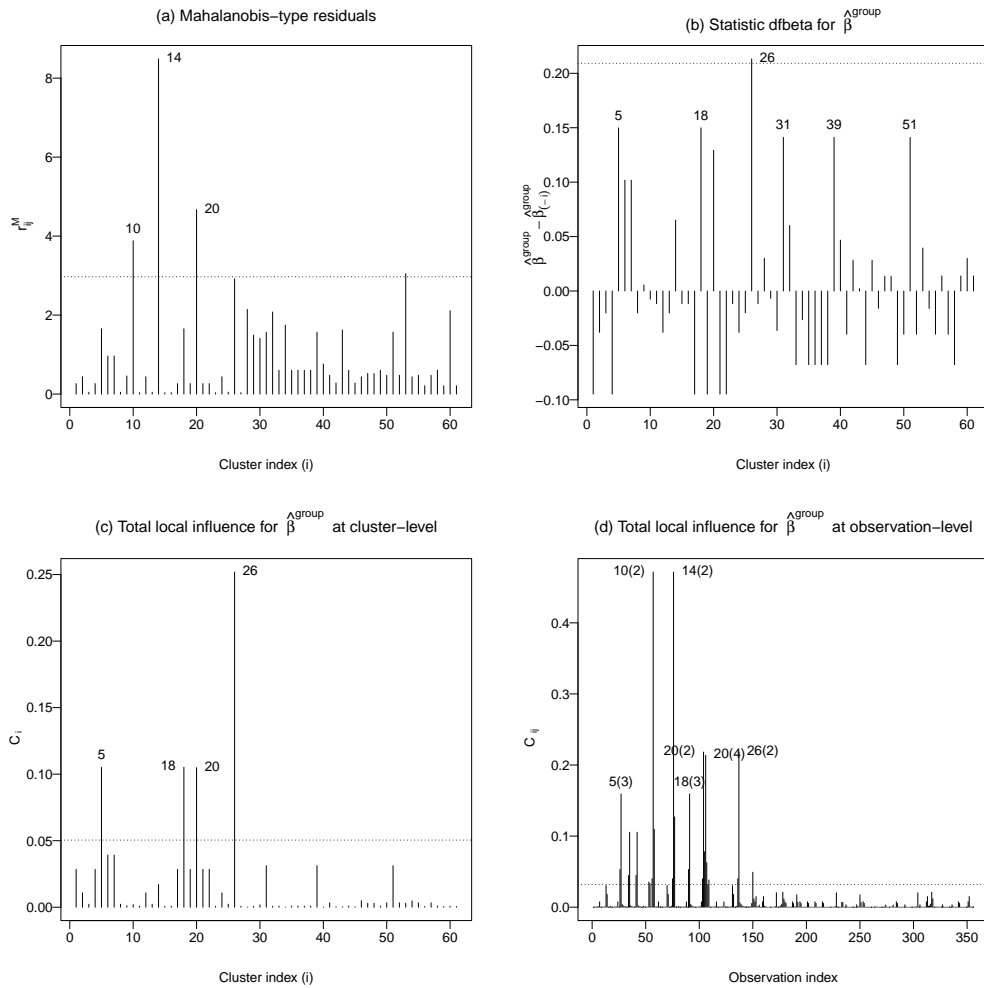
Figure 4: Some diagnostic plots for the GEE with correlation structure AR-1 fitted to the data on postnatal depression.

```
Step 1 : + visit

                df    QIC   QICu   AGPC   SGPC P(Chisq>)(*)
+ group          1 358.24 350.30 234.47 242.91    0.001129
<none>             413.89 409.23 273.13 279.46

Step 2 : + group

                df    QIC   QICu   AGPC   SGPC P(Chisq>)(*)
<none>             358.24 350.30 234.47 242.91
+ group:visit    1 358.63 351.24 234.60 245.16      0.1371
- visit          1 443.77 440.13 366.77 373.10      <2e-16

Final model:
~ visit + group
*************************************************************************
(*) p-values of the Wald test
```

Residual analysis

```
> residuals(m3, type="mahalanobis", plot.it=TRUE, identify=3)
```

Mahalanobis-type residuals suggest that the women for whom the model has the lowest goodness-

of-fit are those identified as 10, 14 and 20 (Figure 3(a)). Those women were treated with placebo patches and just one month after therapy began their EDPS decreased until reaching values lower than 11. However, one or two months later their EDPS values increased to 11 or higher.

Global influence

```
> dfbeta(m3, method="full", coefs="group", identify=6)
```

The plot of the dfbeta statistic for $\hat{\beta}^{\text{group}}$ at the cluster-level (Figure 3(b)) highlights the women identified as 5, 18, 26, 31, 39 and 51. The EDPS values of the women identified as 31, 39 and 51 remained higher or equal to 11 even after they were treated with oestrogen patches. Their exclusion from the dataset decreases the estimate of β^{group}, that is, increases the evidence on the effectiveness of oestrogen patches for treatment of postnatal severe depression. On the other hand, the values on the EDPS of the women identified as 5, 18 and 26 remained lower than 11 since the first or second month since therapy began, although they were treated with placebo patches, so their exclusion from the dataset also increases the evidence on the effectiveness of the oestrogen patches for treatment of postnatal severe depression.

Local influence

```
> localInfluence(m3, type="total", perturbation="cw-clusters", coefs="group",
+                plot.it=TRUE, identify=4)
> localInfluence(m3, type="total", perturbation="cw-observations", coefs="group",
+                plot.it=TRUE, identify=7)
```

According to the plot of the total local influence for $\hat{\beta}^{\text{group}}$ under the case weight perturbation scheme at the cluster-level (Figure 3(c)), the women identified as 5, 18, 20 and 26 are suspected to be influential on $\hat{\beta}^{\text{group}}$. At least 4/7 of the EDPS measurements carried out on those women were smaller than 11, although they were supplied with placebo patches. The plot of the total local influence for $\hat{\beta}^{\text{group}}$ under the case weight perturbation scheme at the observation-level (Figure 3(d)) highlights mainly two kinds of observations: (1) measurements of the EDPS in which, unlike the others measurements made on the same women, the values were lower than 11, although they were treated with placebo patches (second measurement performed on women identified as 10 and 14); (2) measurements of the EDPS performed on women treated with placebo patches and in which, for the first time for those women since the treatment began, the reported value was lower than 11, thus indicating absent severe depression, which remains until the end of the observation period (second, third and fourth measurements performed on women identified as 26, 18 and 5, respectively).

Comparison with other GEE solvers

The parameter estimates and the associated standard errors provided by the function glmgee() are compared with those calculated by the GEE solvers available in the packages gee, geepack and geeM. The results are presented in Table 4. The values obtained with the other GEE solvers are very similar to those obtained with the function glmgee().

	glmtoolbox	geepack	gee	geeM
(Intercept)	3.236(0.518)	3.276(0.531)	3.214(0.514)	3.199(0.543)
visit	-0.626(0.075)	-0.630(0.077)	-0.624(0.074)	-0.633(0.077)
groupestrogen	-1.777(0.546)	-1.847(0.556)	-1.754(0.543)	-1.781(0.572)
ρ	0.51	0.48	0.47	0.51

Table 4: Parameter estimates (standard errors) of the GEE model with correlation structure AR-1 fitted to the data on severe postnatal depression.

5.4 Growth patterns of two soybean genotypes

This dataset, analyzed in Davidian and Giltinan (1995) and Pinheiro and Bates (2000) and available in the object Soybean of the package nlme (Pinheiro et al., 2022), arose from an experiment aimed at comparing growth patterns of two genotypes of soybeans: Plant Introduction (Variety="P"), an experimental strain, and Forrest (Variety="F"), a commercial variety. The average leaf weight per

plant (`weight`), in grams, was measured at 14, 20, 27, 34, 41, 55, 69 and 84 days after planting (`Time`) in each plot (`Plot`). As an illustration, only plots planted in 1989 (`Year`="1989") are analyzed here. Table 5 describes the roles played in the analysis of the variables in the `Soybean` dataset. The graph of the data (Figure 5) indicates that the location of the response (average leaf weight per plant) increases non-linearly over time. In addition, there is an approximately proportional relationship between the mean and the standard deviation, as the variance of the response variable (in the log scale) seems to be constant. As a result, the data may be analyzed assuming that the coefficient of variation is constant, that is, using a quadratic variance function. Moreover, the graph of the data also indicates that at each time point, the location of the response is larger for the experimental strain (`Variety`="P") than for the commercial variety (`Variety`="F") of soybean.

Column	Role	Description
`Plot`	Cluster/subject identifier	Identifier of the plot
`Variety`	Explanatory variable	Treatment: "F" (experimental) or "P" (commercial)
`Year`	Explanatory variable	Year the plot was planted
`Time`	Explanatory variable	Days after planting
`weight`	Response variable	Average leaf weight per plant

Table 5: Columns in the object `Soybean` of the package `nlme`.
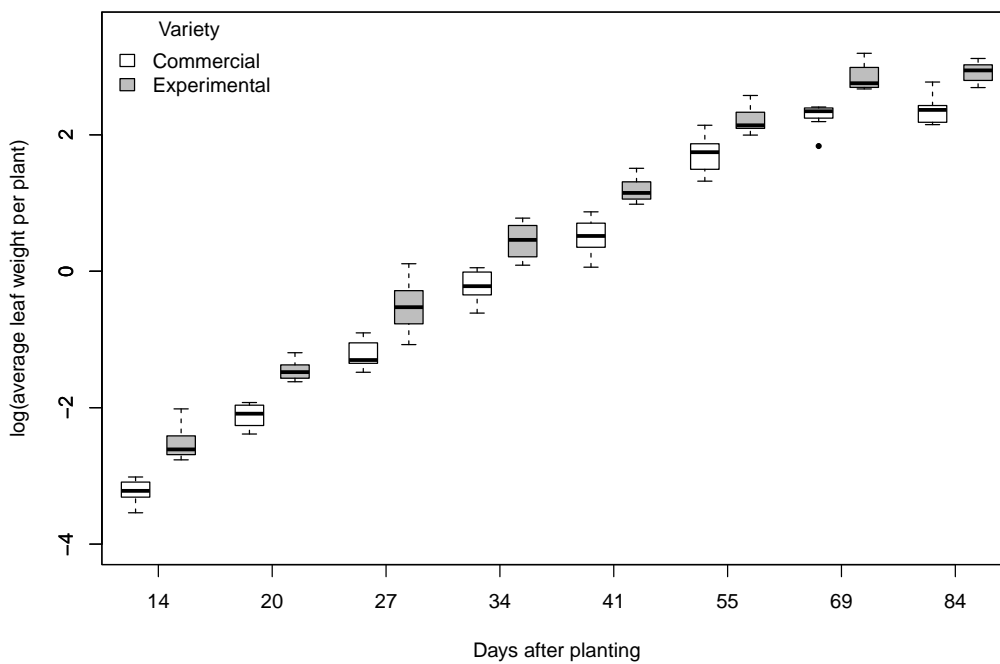
Figure 5: Average leaf weight per plant over time.

The data are analyzed using a model where the mean of the random variable Y_{ij} (jth measurement of the average leaf weight per plant performed on the ith plot) is given by the following logistic-type curve:

$$\mu_{ij} = \frac{\beta_1 + \beta_4 \, \texttt{Variety}_{ij}}{1 + \exp\left(-(\texttt{Time}_{ij} - \beta_2 - \beta_5 \, \texttt{Variety}_{ij})/(\beta_3 + \beta_6 \, \texttt{Variety}_{ij})\right)}, \quad i = 1, \dots, 16; j = 1, \dots, 8,$$

where $\texttt{Variety}_{ij} = 1$ if the soybean genotype is the experimental strain and $\texttt{Variety}_{ij} = 0$ otherwise. Therefore, the horizontal asymptote as $\texttt{Time} \to \infty$ (also known as the carrying capacity), the inflection point and the scale parameter of μ for commercial varieties of soybean are β_1, β_2 and β_3; and $(\beta_1 + \beta_4)$, $(\beta_2 + \beta_5)$ and $(\beta_3 + \beta_6)$ for experimental strains.

The starting value for β in the algorithm of parameter estimation is obtained through the built-in function `SSlogis()`.

```
> data(Soybean,package="nlme")
> Soybean2 <- subset(Soybean,Year=="1989")
> Soybean2 <- within(Soybean2,x <- ifelse(Variety=="P",1,0))
>
> m0 <- gnmgee(weight ~ SSlogis(Time,b1,b2,b3), id=Plot, family=Gamma(identity),
+              data=Soybean2)
> start <- c(coef(m0),rep(0,3))
> names(start) <- paste0("b",1:6)
> start
        b1        b2        b3        b4        b5        b6
14.185637 51.453724  7.086697  0.000000  0.000000  0.000000
```

Then, GEE models with quadratic variance function and different correlation matrix structures are fitted to the data.

```
> m1 <- gnmgee(weight ~ (b1 + b4*x)/(1 + exp(-(Time - b2 - b5*x)/(b3 + b6*x))),
+              start=start, id=Plot, family=Gamma(identity), data=Soybean2)
> m2 <- update(m1, corstr="Exchangeable")
> m3 <- update(m1, corstr="AR-M-dependent(1)")
> m4 <- update(m1, corstr="AR-M-dependent(2)")
> m5 <- update(m1, corstr="AR-M-dependent(3)")
> m6 <- update(m1, corstr="AR-M-dependent(4)")
```

As shown below, the correlation matrix structure chosen by the most of the criteria (that is, CIC, QIC, GHYC and PAC) is `AR-M-dependent(3)`.

```
> a <- CIC(m1, m2, m3, m4, m5, m6, verbose=FALSE)
> b <- QIC(m1, m2, m3, m4, m5, m6, verbose=FALSE)
> c <- GHYC(m1, m2, m3, m4, m5, m6, verbose=FALSE)
> d <- PAC(m1, m2, m3, m4, m5, m6, verbose=FALSE)
> e <- AGPC(m1, m2, m3, m4, m5, m6, verbose=FALSE)
> f <- SGPC(m1, m2, m3, m4, m5, m6, verbose=FALSE)
> cbind(a,QIC=b[,"QIC"],GHYC=c[,"GHYC"],PAC=d[,"PAC"],AGPC=e[,"AGPC"],SGPC=f[,"SGPC"])

  Object       Correlation  CIC      QIC  GHYC    PAC    AGPC    SGPC
1     m1      Independence 6.951 6163.648 8.126 0.9847 90.5844 95.2200
2     m2      Exchangeable 6.951 6163.648 7.552 0.9785 86.8152 92.2233
3     m3 AR-M-dependent(1) 6.795 6098.876 6.640 0.9753 86.1055 91.5136
4     m4 AR-M-dependent(2) 6.713 6095.808 6.622 0.9737 87.7812 93.9619
5     m5 AR-M-dependent(3) 6.708 6094.956 6.621 0.9736 89.7920 96.7453
6     m6 AR-M-dependent(4) 6.752 6115.573 6.673 0.9741 91.3912 99.1171
```

The chosen model is summarized as follows:

```
> summary(m5)

Sample size
   Number of observations:  128
       Number of clusters:  16
             Cluster size:  8
**************************************************************
Model
        Variance function:  Gamma
            Link function:  identity
    Correlation structure:  AR-M-dependent(3)
**************************************************************
Coefficients
          Estimate Std.Error  z-value Pr(>|z|)
b1        10.58794   0.54866 19.29779 < 2e-16
b2        52.08512   0.99860 52.15828 < 2e-16
b3         7.01786   0.19565 35.87033 < 2e-16
b4         7.48960   0.88795  8.43475 < 2e-16
b5        -0.77453   1.29528 -0.59797  0.54986
b6         0.09913   0.24511  0.40441  0.68591


Dispersion  0.05686
```

```
*************************************************************
Working correlation
      [1]   [2]   [3]   [4]   [5]   [6]   [7]   [8]
[1] 1.000 0.253 0.151 0.053 0.025 0.010 0.004 0.002
[2] 0.253 1.000 0.253 0.151 0.053 0.025 0.010 0.004
[3] 0.151 0.253 1.000 0.253 0.151 0.053 0.025 0.010
[4] 0.053 0.151 0.253 1.000 0.253 0.151 0.053 0.025
[5] 0.025 0.053 0.151 0.253 1.000 0.253 0.151 0.053
[6] 0.010 0.025 0.053 0.151 0.253 1.000 0.253 0.151
[7] 0.004 0.010 0.025 0.053 0.151 0.253 1.000 0.253
[8] 0.002 0.004 0.010 0.025 0.053 0.151 0.253 1.000
```

These results suggest that only the horizontal asymptote as `Time` $\to \infty$ depends on the soybean variety. Their estimates are 10.588 grams and 18.078 grams for commercial varieties and experimental soybean strains, respectively.

5.5 Amenorrhea rates over time

The dataset of this example, available in the object `amenorrhea` of `glmtoolbox` and comprised of the columns named `ID`, `Dose`, `Time`, and `amenorrhea` (see Table 6), arose from a longitudinal clinical trial of contracepting women (Machin et al., 1988; Fitzmaurice et al., 2011). A total of 1151 women completed menstrual diaries. The diary data were used to generate a binary sequence for each woman, indicating whether she had experienced amenorrhea (the absence of menstrual bleeding for a specified number of days) on the day of randomization and three additional 90-day intervals. This trial compared the two treatments (injections of 100 mg or 150 mg of depot-medroxyprogesterone acetate (DMPA)) in terms of how amenorrhea rates change over time with continued use of the contraceptive method. Figure 6 shows that amenorrhea rates increase across treatments, but that it appears that women treated with 150 mg of DMPA are more likely to experience amenorrhea than those treated with 100 mg of DMPA at each time point. Moreover, Figure 6 shows that the proportion of women experiencing amenorrhea (on the logit scale) increases non-linearly over time. A feature of this clinical trial is that there was substantial dropout. This is when a woman skips a particular injection and never returns for subsequent injections. Indeed, 38% of the women dropped out before the trial ended; 17.2% dropped out after receiving only one injection of DMPA, 13.5% dropped out after receiving only two injections of DMPA, and 7.3% dropped out after receiving three injections of DMPA. The subsequent statistical analysis is performed using the weighted GEE method, as it is assumed that the missing data pattern is better described by MAR than MCAR.

Column	Role	Description
ID	Cluster/subject identifier	Identifier of the woman
Dose	Explanatory variable	Treatment: "100mg" or "150mg" of DMPA
Time	Explanatory variable	Number of 90-day intervals since the trial began
amenorrhea	Response variable	1 if experienced amenorrhea; 0 otherwise

Table 6: Columns in the object `amenorrhea` of the package `glmtoolbox`.

The data are analyzed using a model in which the probability of the ith woman experienced amenorrhea at time j, denoted here by μ_{ij}, is such that

$$\text{logit}(\mu_{ij}) = 1 + \texttt{Time} + \texttt{Time}^2 + \texttt{Dose}.$$

For the missingness model the following systematic component is considered

$$\text{logit}(\pi_{ij}) = 1 + \texttt{CTime} + \texttt{Dose} + \texttt{ylag1},$$

where `CTime` is a categorical version of the explanatory variable `Time` and `ylag1` is defined to be $y_{i,j-1}$ if $j > 1$ and 0 if $j = 1$. In addition, the structure of the working correlation matrix is set to be `AR-M-dependent(1)`. The observation-specified weighted GEE method results are the following:

```
> data(amenorrhea)
> amenorrhea2 <- within(amenorrhea,{Ctime <- factor(Time)
+                                   Ctime <- relevel(Ctime,ref="1")
+                                   ylag1 <- c(0,amenorrhea[-length(ID)])
+                                   ylag1 <- ifelse(Time==0,0,ylag1)})
>
```
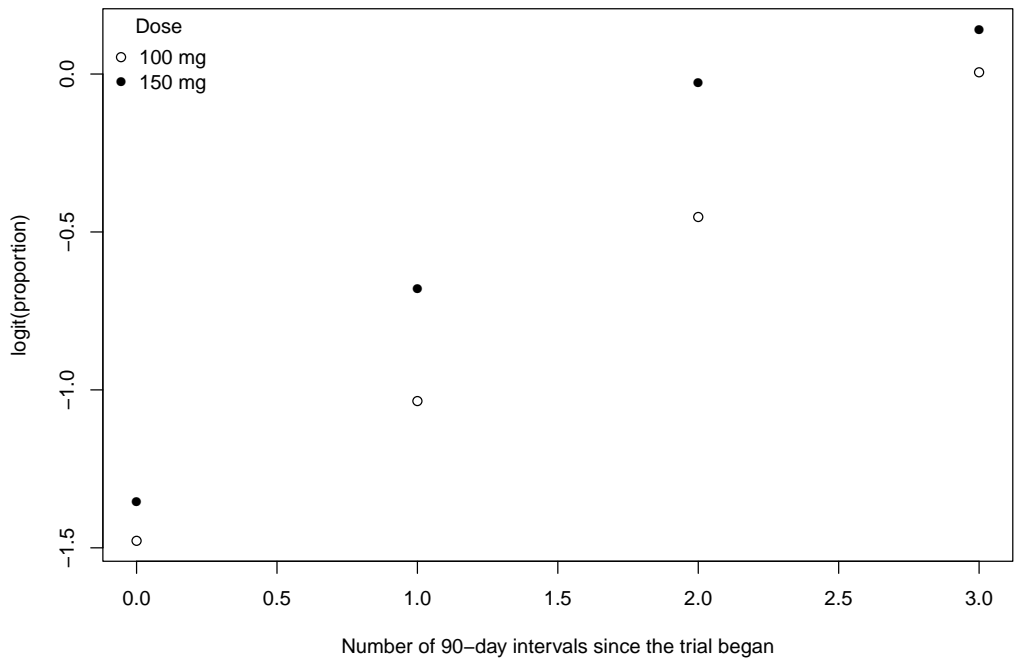
Figure 6: Amenorrhea rates over time.

```
> fit1 <- wglmgee(amenorrhea ~ poly(Time,2) + Dose | Ctime + Dose + ylag1,
+                 family=binomial, data=amenorrhea2, id=ID, corstr="AR-M-dependent(1)",
+                 scale.fix=TRUE, scale.value=1, level="observations")
> summary(fit1)

Clusters by dropout time
 Time 1    2    3    4    Freq   %
   ---- ---- ---- ---- | ---- ----
     X    .    .    . |  198 17.2
     X    X    .    . |  155 13.5
     X    X    X    . |   84  7.3
     X    X    X    X |  714   62
   ---- ---- ---- ---- | ---- ----
                      | 1151  100
**************************************************************
Coefficients of missingness model
           Estimate Std.Error z-value   Pr(>|z|)
(Intercept)  2.4349    0.1401 17.3845 < 2.2e-16 ***
Ctime1      -0.7247    0.1438 -5.0399 4.659e-07 ***
Ctime2      -0.5911    0.1469 -4.0250 5.698e-05 ***
Dose150mg   -0.0174    0.1049 -0.1663    0.8679
ylag1       -0.5765    0.1122 -5.1369 2.793e-07 ***
**************************************************************
Observation-specific Weighted GEE

        Variance function:  binomial
            Link function:  logit
    Correlation structure:  AR-M-dependent(1)
**************************************************************
Coefficients
              Estimate Std.Error z-value Pr(>|z|)
(Intercept)    -0.6835    0.0750 -9.1136 < 2e-16 ***
poly(Time, 2)1 40.7447    2.2598 18.0301 < 2e-16 ***
poly(Time, 2)2 -4.6883    1.9528 -2.4008 0.01636 *
```

```
Dose150mg        0.2437    0.1061  2.2975  0.02159 *

Dispersion       1.0000
*************************************************************
Working correlation
     [1]   [2]   [3]   [4]
[1] 1.000 0.414 0.171 0.071
[2] 0.414 1.000 0.414 0.171
[3] 0.171 0.414 1.000 0.414
[4] 0.071 0.171 0.414 1.000
```

According to the missingness model, the probability of remaining in the trial increases over time, regardless of the amenorrhea status reported by the women in the previous measurement. In addition, the missingness model indicates that the probability of remaining in the trial is higher for women without amenorrhea in their previous measurement, regardless of how many DMPA injections they have received. Therefore, the MAR assumption seems more appropriate than MCAR. Moreover, according to the model for μ, the odds of experiencing amenorrhea is approximately $27.6\% = 100 \times [\exp(\hat{\beta}^{\text{Dose}}) - 1]$ higher in women treated with 150 mg of DMPA than those treated with 100 mg of DMPA, where β^{Dose} represents the parameter associated with the explanatory variable `Dose`. This is irrespective of the number of 90-day intervals since the experiment began. The results of the cluster-specific weighted GEE approach (do not show here) are very similar.

Acknowledgments

References

D. Boos. On generalized score tests. *The American Statistician*, 46(4):327–33, 1992. [p110]

V. Carey. *gee: Generalized Estimation Equation Solver*, 2022. URL https://CRAN.R-project.org/package=gee. R package version 4.13-23. [p105]

V. Carey and Y.-G. Wang. Working covariance model selection for generalized estimating equations. *Statistics in Medicine*, 30(26):3117–3124, 2011. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.4300. [p111]

R. Cook. Assessment of local influence. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(2):133–169, 1986. [p112]

M. Davidian and D. M. Giltinan. *Nonlinear models for repeated measurement data*, volume 62. CRC press, 1995. [p126]

P. Diggle, P. J. Diggle, P. Heagerty, K.-Y. Liang, S. Zeger, et al. *Analysis of longitudinal data*. Oxford university press, 2002. [p116]

G. Fitzmaurice, N. Laird, and J. Ware. *Applied Longitudinal Analysis. 2nd ed.* John Wiley & Sons, 2011. [p115, 129]

L. Fu, Y. Hao, and Y.-G. Wang. Working correlation structure selection in generalized estimating equations. *Computational Statistics*, 33:983–96, 2018. [p111]

M. Gosho. Criteria to select a working correlation structure in sas. *Journal of Statistical Software, Code Snippets*, 57(1):1–10, 2014. doi: 10.18637/jss.v057.c01. URL https://www.jstatsoft.org/index.php/jss/article/view/v057c01. [p110]

M. Gosho, C. Hamada, and I. Yoshimura. Criterion for the selection of a working correlation structure in the generalized estimating equation approach for longitudinal balanced data. *Communications in Statistics - Theory and Methods*, 40(21):3839–3856, 2011. [p110]

A. Gregoire, R. Kumar, B. Everitt, and J. Studd. Transdermal oestrogen for treatment of severe postnatal depression. *The Lancet*, 347(9006):930–933, 1996. [p121]

B. Hammill and J. Preisser. A sas/iml software program for gee and regression diagnostics. *Computational Statistics & Data Analysis*, 51(2):1197–1212, 2006. [p111, 112, 114]

L. Hin, V. Carey, and Y.-G. Wang. Criteria for working-correlation-structure selection in gee: Assessment via simulation. *The American Statistician*, 61(4):360–364, 2007. URL http://www.jstor.org/stable/27643940. [p111]

L.-Y. Hin and Y.-G. Wang. Working-correlation-structure identification in generalized estimating equations. *Statistics in Medicine*, 28(4):642–658, 2009. [p110]

S. Højsgaard, U. Halekoh, and J. Yan. The R package geepack for generalized estimating equations. *Journal of Statistical Software*, 15(2):1–11, 2005. [p105]

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. [p113]

K.-M. Jung. Local influence in generalized estimating equations. *Scandinavian Journal of Statistics*, 35 (2):286–294, 2008. [p112]

N. Laird. Missing data in longitudinal studies. *Statistics in medicine*, 7(1-2):305–315, 1988. [p109]

K. Liang and S. Zeger. Longitudinal data analysis using generalized linear models. *Biometrika*, 73: 13–22, 1986. [p105, 106, 109]

S. Lipsitz and G. Fitzmaurice. *Generalized estimating equations for longitudinal data analysis*, chapter chapter3. CRC Press, 2008. [p105, 109]

S. Lipsitz, N. Laird, and D. Harrington. Using the jackknife to estimate the variance of regression estimators from repeated measures studies. *Communications in Statistics - Theory and Methods*, 19(3): 821–845, 1990. [p109]

D. Machin, T. Farley, B. Busca, M. Campbell, and C. d'Arcangues. Assessing changes in vaginal bleeding patterns in contracepting women. *Contraception*, 38:165–179, 1988. [p129]

M. Maechler, P. Rousseeuw, C. Croux, V. Todorov, A. Ruckstuhl, M. Salibian-Barrera, T. Verbeke, M. Koller, E. L. T. Conceicao, and M. Anna di Palma. *robustbase: Basic Robust Statistics*, 2022. R package version 0.95-0. [p116]

L. Mancl and T. DeRouen. A covariance estimator for gee with improved small-sample properties. *Biometrics*, 57(1):126–134, 2001. doi: doi:10.1111/j.0006-341x.2001.00126.x. [p109]

P. McCullagh and J. Nelder. *Generalized Linear Models, Second Edition*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. Chapman & Hall, 1989. URL http://books.google.com/books?id=h9kFH2_FfBkC. [p105, 110]

L. McDaniel, N. Henderson, and P. Rathouz. Fast pure R implementation of GEE: application of the Matrix package. *The R Journal*, 5:181–187, 2013. URL https://journal.r-project.org/archive/2013-1/mcdaniel-henderson-rathouz.pdf. [p105]

W. Pan. Akaike's information criterion in generalized estimating equations. *Biometrics*, 57(1):120–125, 2001. URL http://www.jstor.org/stable/2676849. [p110, 113]

M. C. Pardo and R. Alonso. Working correlation structure selection in gee analysis. *Statistical Papers*, 60(5):1447–1467, 2019. URL https://doi.org/10.1007/s00362-017-0881-0. [p111]

J. Pinheiro and D. Bates. *Mixed-Effects Models in S and S-PLUS*. Statistics and Computing. Springer New York, 2000. ISBN 9780387989570. URL https://books.google.com.co/books?id=N3WeyHFbHLQC. [p126]

J. Pinheiro, D. Bates, and R Core Team. *nlme: Linear and Nonlinear Mixed Effects Models*, 2022. URL https://CRAN.R-project.org/package=nlme. R package version 3.1-160. [p126]

J. Preisser and B. Qaqish. Deletion diagnostics for generalised estimating equations. *Biometrika*, 83(3): 551–562, 1996. [p111, 112, 114]

J. Preisser, K. Lohman, and P. Rathouz. Performance of weighted estimating equations for longitudinal binary data with drop-outs missing at random. *Statistics in Medicine*, 21:3035–3054, 2002. [p115, 116]

J. Robins, A. Rotnitzky, and L. Zhao. Analysis of semiparametric regression models for repeated outcomes in the presence of missing data. *Journal of the American Statistical Association*, 90:122–129, 1995. [p115, 116]

A. Rotnitzky and N. Jewell. Hypothesis testing of regression parameters in semiparametric generalized linear models for cluster correlated data. *Biometrika*, 77(3):485–497, 1990. URL http://www.jstor.org/stable/2336986. [p110]

R. Wedderburn. Quasi-likelihood functions, generalized linear models, and the gauss—newton method. *Biometrika*, 61(3):439–447, 1974. [p105]

J. Xu, J. Zhang, and L. Fu. Variable selection in generalized estimating equations via empirical likelihood and gaussian pseudo-likelihood. *Communications in Statistics - Simulation and Computation*, 48(4):1239–1250, 2019. [p113]

J. Yan. geepack: Yet another package for generalized estimating equations. *R-News*, 2/3:12–14, 2002. [p105]

A. Zeileis and Y. Croissant. Extended model formulas in R: Multiple parts and multiple responses. *Journal of Statistical Software*, 34(1):1–13, 2010. doi: 10.18637/jss.v034.i01. [p116]

X. Zhu and Z. Zhu. Comparison of criteria to select working correlation matrix in generalized estimating equations. *Chinese Journal of applied probability and statistics*, 5:515–30, 2013. [p111]

L.H. Vanegas
Departamento de Estadística, Universidad Nacional
Ciudad Universitaria, Bogotá
Colombia
lhvanegasp@unal.edu.co

L.M. Rondón
Departamento de Estadística, Universidad Nacional
Ciudad Universitaria, Bogotá
Colombia
lmrondonp@unal.edu.co

G.A. Paula
Instituto de Matemática e Estatística, Universidade de São Paulo
Rua do Matão, 1010, São Paulo
Brazil
giapaula@ime.usp.br

clustAnalytics: An R Package for Assessing Stability and Significance of Communities in Networks

by Martí Renedo-Mirambell and Argimiro Arratia

Abstract This paper introduces the R package clustAnalytics, which comprises a set of criteria for assessing the significance and stability of communities in networks found by any clustering algorithm. clustAnalytics works with graphs of class igraph from the R-package igraph, extended to handle weighted and/or directed graphs. clustAnalytics provides a set of community scoring functions, and methods to systematically compare their values to those of a suitable null model, which are of use when testing for cluster significance. It also provides a non parametric bootstrap method combined with similarity metrics derived from information theory and combinatorics, useful when testing for cluster stability, as well as a method to synthetically generate a weighted network with a ground truth community structure based on the preferential attachment model construction, producing networks with communities and scale-free degree distribution.

1 Introduction

The segmentation of a network into communities such that individuals in the same community share similar features, whilst individuals across communities are dissimilar is a technique known as clustering, and is a principal task in the analysis of networks. This can be achieved by various algorithms that differ significantly in their understanding of what constitutes a community (or cluster) and in the way to find them. Once a clustering algorithm is selected, the user is faced with the problem of determining how meaningful the clusters obtained are. Here is where clustAnalytics comes into play.

clustAnalytics contains a suite of novel methods to validate the partitions of networks obtained by any given clustering algorithm. In particular, its clustering validation methods focus on two of the most important aspects of cluster assessment: the *significance* and the *stability* of the resulting clusters. Clusters produced by a clustering algorithm are considered to be *significant* if there are strong connections within each cluster, and weaker connections (or fewer edges) between different clusters. On the other hand, *stability* measures how much a clustering remains unchanged under small perturbations of the network. In the case of weighted networks, these could include the addition and removal of vertices, as well as the perturbation of edge weights.

clustAnalytics handles weighted networks (that is, those in which the connections between nodes have an assigned numerical value representing some property of the data), as well as unweighted, and contains several other functionalities for producing different statistics on a network, and measures of similarity of partitions produced by two different clustering algorithms, most notably an enhanced version of the Reduced Mutual Information (RMI) of Newman et al. (2020). In the following sections we introduce some examples of usage and highlight the principal tasks resolved by **clustAnalytics**.

2 Background concepts

Community detection on networks, which are represented by graphs, is a very active topic of research with many applications. The **igraph** (Csardi and Nepusz, 2006) package contains a collection of the most popular algorithms for this task, such as the Louvain (Blondel et al., 2008), walktrap (Pons and Latapy, 2005) and label propagation Raghavan et al. (2007) algorithms.

Evaluating the significance of the community structure of a network is no simple task, because there is not an authoritative definition of what a significant community is. However, there is some agreement in the literature (see the survey by Fortunato (2010)) in that communities should have high internal connectivity (presence of edges connecting nodes in the community) while being well separated from each other. These notion can be quantified and formalized by applying several community *scoring functions* (also known as quality functions in (Fortunato, 2010)), that gauge either the intra-cluster or inter-cluster density. **clustAnalytics** implements the most relevant, or representative, community scoring functions following the taxonomy of these quality measures done by Yang and Leskovec (2015) and the further discussion on how to adapt them to weighted networks in (Arratia and Renedo-Mirambell, 2021).

significance	scoring functions	`scoring_functions(g, com, type , weighted, w_max)`
		`average_degree, average_odf, conductance, coverage,` `cut_ratio, density_ratio, edges_inside, expansion, FOMD,` `internal_density, max_odf, normalized_cut,` `weighted_clustering_coefficient, weighted_transitivity`
	graph rewiring	`rewireCpp(g, Q=100, weight_sel="const_var", lower_bound=0,` `          upper_bound=NULL)`
	evaluation	`evaluate_significance(g, alg_list, gt_clustering, w_max)` `evaluate_significance_r(g, alg_list, gt_clustering,` `                        Q=100, lower_bound=0, w_max=NULL,` `                        table_style = "default")`
stability		`boot_alg_list(g, alg_list, R=999, return_data=FALSE,` `              type="global")` `reduced_mutual_information(c1, c2, base=2, normalized=FALSE,` `                           method="approximation2")`
other functions		`apply_subgraphs(g, com, f, ...)` `barabasi_albert_blocks(m, p, B, t_max, G0=NULL, t0=NULL,` `                       G0_labels=NULL, type="block_first")` `                       sample_with_replacement=FALSE` `sort_matrix(M)`

Table 1: *clustAnalytics* list of functions split by category.

However, to evaluate the significance of clusters on a given network, one needs reference values of the scoring functions to determine whether they are actually higher than those of a comparable network with no community structure. For this, we use a method described in (Arratia and Renedo-Mirambell, 2021) that rewires edges (or transfers some of their weight, in the case of weighted networks) while keeping the degree distribution constant. Then, it can be determined that the partition of a network contains significant clusters if it obtains sufficiently better scores than those for a comparable network with uniformly distributed edges.

On the other hand, stability measures how much the partition of a network into communities remains unchanged under small perturbations. In the case of weighted networks, these could include the addition and removal of vertices, as well as the perturbation of edge weights. This is consistent with the idea that meaningful clusters should capture an inherent structure in the data and not be overly sensitive to small or local variations, or the particularities of the clustering algorithm. To measure the variation that such perturbations present in the clusters, there are multiple similarity metrics available. We have selected for inclusion the Variation of Information (Meilă, 2007), the Reduced Mutual Information (Newman et al., 2020), and the Rand Index (both in its original and adjusted forms) (Hubert and Arabie, 1985). The first two are based on information theory, while the second one counts agreements and disagreements in the membership of pairs of elements. Then, it is possible to evaluate the network using resampling methods such as nonparametric bootstrap, as described for clustering on Euclidean data by Hennig (2007), and later for networks by Arratia and Renedo-Mirambell (2021), and quantify the deviations from the initial partition with the similarity measures.

3 The *clustAnalytics* package

The *clustAnalytics* package (version 0.5.4) contains 23 functions for assessing clustering significance and stability, and other useful utilites. These are listed in Table 1 grouped by category. It also contains some auxiliary functions to support package management and provide useful baseline graphs with communities. Check these in the reference manual. In what follows we detail the usage of the main functions.

3.1 Cluster significance

The scoring functions are formally defined in (Arratia and Renedo-Mirambell, 2021), and were selected and programmed based on the analysis of appropriate scoring functions for unweighted graphs made in (Yang and Leskovec, 2015). They take into account the weights of the edges if the graph is weighted. They take as arguments the graph as an **igraph** object, and a membership vector: a vector of the same length as the graph order for which each element is an integer that indicates the cluster that its corresponding vertex belongs to.

A general call to all the scoring functions is made with `scoring_functions()`, which computes all the scores and returns a dataframe containing a row for each community (if type = "local") and a column for each score, or alternatively (if type = "global") returns a single row with the weighted average scores. Additionally, an individual function is available for each of the scores, as listed in Table 1. The package includes efficient implementations of the clustering coefficient and transitivity for weighted networks introduced by McAssey and Bijma (2015).

The main functions for significance evaluation are

`evaluate_significance_r()` and `evaluate_significance()`.

The first one takes an **igraph** graph and a list of clustering algorithms, and computes the scoring functions of the resulting communities, both on the original graph and on rewired versions of it for comparison. The second version does the same while skipping the rewired graphs. By default the clustering algorithms used by these functions are Louvain, label propagation and Walktrap, but they can take any list of clustering algorithms for **igraph** graphs. Both functions allow for comparison against ground-truth in cases where this is known.

The edge rewiring method (including its versions for weighted networks) is available separately as `rewireCpp`. This differs from the **igraph** function `rewire`, in that it is capable of rewiring weighted as well as directed graphs while keeping the weighted degrees constant.

Rewiring algorithm

The function `rewireCpp` provided by the package is an implementation of the switching algorithm that rewires edges while keeping the degree distribution constant described in (Milo et al., 2003; Rao et al., 1996) (conceived originally for unweighted graphs). The function has been extended to work with weighted and/or directed graphs.

The directed version works very similarly to the undirected one. In the unweighted case, at each step of the algorithm, two directed edges AC and BD are selected randomly, and replaced with the new edges AD, BC (as in the original algorithm, any steps that would produce self-edges or multi-edges are skipped). For vertices A and B, we add and remove 1 to the out-degree, so it remains constant (as well as the in-degree, since no incoming edges are modified). Analogously, we add and remove one incoming edge to both the C and D vertices, so their in-degrees remain constant as well.

We do the same for the directed weighted case, extending the undirected unweighted algorithm. This time, when edges AC and BD are selected, there is a transfer of a certain amount $\bar{w}$ of weight from both AC and BD to AD and BC. This means that the only effects on the in and out-degrees are adding and removing $\bar{w}$ to out-degrees of vertices A and B, and the same to out-degrees of vertices C and D, which means that they all remain constant.

If the graph is directed, the `rewireCpp` function automatically detects it and internally runs the implementation for directed graphs, so there is no need to specify direction as a parameter. The following example is a food network (where edges indicate predator-prey relationships) from the **igraphdata** package:

```
> data(foodwebs, package="igraphdata")
> rewired_ChesLower <- rewireCpp(foodwebs$ChesLower, weight_sel = "max_weight")
```

In the weighted case, the rewiring algorithm transfers a certain amount w of weight from some edges to others. The package provides two settings, which are chosen according to what type of weighted graph is provided as input:

- **Complete graphs with a fixed upper bound:** These graphs have an edge between every pair of vertices, which will usually be the result of applying some function to each pair. For example, networks resulting from computing correlations of time series (where each series corresponds to a vertex, and the edge weights are the correlations between series) fall into this category.

- **More sparse graphs with weights that are non-negative but not necessarily upper bounded:** This describes most commonly found weighted graphs, where the weights quantify some characteristic of the edges. Multigraphs also fit here, if we reinterpret them as weighted graphs where the edge weight is the number of parallel edges between each pair of vertices.

Of the first type, we show an example built from correlations of currency exchange time series (from Arratia and Renedo-Mirambell (2021)). In this network (g_forex included in the package) vertices are pairs of exchange rates, and the edge weights are the correlations of their corresponding time series, scaled to the interval $[0, 1]$. In this case, the appropriate setting is the one that keeps the variance of the edge weights constant.

```
> data(g_forex, package="clustAnalytics")
> rewireCpp(g=g_forex, weight_sel="const_var", lower_bound=0, upper_bound=1)
```

As for the second type, this includes most of the well known examples of weighted graphs, such as Zachary's karate club graph:

```
> data(karate, package="igraphdata")
> rewired_karate <- rewireCpp(karate, weight_sel="max_weight")
```

The number of iterations, which is computed as $Q \cdot \#edges$ can be controlled with the parameter Q, but we recommend leaving it on the default value ($Q = 100$), which has been shown to provide more than enough shuffling, while still being very fast (Arratia and Renedo-Mirambell, 2021, p.10).

3.2 Cluster stability

As for the study of cluster stability, the function used to perform the evaluation is boot_alg_list(). This performs a bootstrap resampling (i.e. uniform sampling of the vertices with replacement) of the input graph, applies a given list of clustering algorithms, and measures the variation of the communities obtained in the resampled graphs with respect to the original communities. In more detail, for each input graph and a list of clustering algorithms, the set of vertices in the input graph is resampled many times, the induced graph is obtained by taking the new set of vertices with the induced edges from the original graph (two vertices are joined with an edge on the resampled graph if they were on the original graph), and the clustering algorithms are applied to it. Then, the resulting clusterings (in each of the resampled graphs) are compared to the clustering of the original graph using several metrics: the variation of information (vi.dist from package **mcclust**), normalized reduced mutual information (NRMI) and both adjusted and regular Rand index (rand.index from package **fossil** and adjustedRandIndex from package **mclust**). If return_data is set to TRUE, the output is a list of objects of class boot (from package **boot**); otherwise, returns a table with the mean distances from the clusters in the original graph to the resampled ones, for each of the algorithms.

The Reduced Mutual Information is provided as a separate function:

$$\text{reduced\_mutual\_information}().$$

This is an implementation of Newman's Reduced Mutual Information (RMI) Newman et al. (2020), a version of the mutual information that is corrected for chance. The exact computation of this metric cannot be reasonably achieved for even moderately sized graphs, so it must be approximated. We provide two analytical methods for this approximation, and other that combines a Monte Carlo method with the analytical formula (method="hybrid").

```
> data(karate, package="igraphdata")
> c1 <- membership(cluster_louvain(karate))
> c2 <- V(karate)$Faction
> reduced_mutual_information(c1, c2, method="approximation2")
[1] 0.5135699
```

Just as with the standard mutual information, the RMI can be normalized as well:

```
> reduced_mutual_information(c1, c2, method="approximation2", normalized=TRUE)
[1] 0.6621045
```

3.3 Graph generators and other utilities

In the analysis of clustering algorithms it is useful to generate controlled examples of networks with communities. The package **igraph** provides the function sample_sbm which builds random graphs with communities from the stochastic block model, and hence these networks have binomial degree distribution.

We provide in **clustAnalytics** the barabasi_albert_blocks() function, which produces scale-free graphs using extended versions of the Barabási-Albert model that include a community structure. This function generates the graph by iteratively adding vertices to an initial graph and joining them to the existing vertices using preferential attachment (existing higher degree vertices are more likely to receive new edges). Additionally, vertices are assigned labels indicating community membership, and the probability of one vertex connecting to another is affected by their community memberships according to a fitness matrix B (if a new vertex belongs to community i, the probability of connecting to a vertex of community j is proportional to B_{ij}).

The parameters that need to be set are m the number of new edges per step, the vector p of label probabilities, the fitness matrix B (with the same dimensions as the length of p), and t_max the final graph order. The initial graph $G0$ can be set manually, but if not, an appropriate graph will be generated with m edges per vertex, labels sampled from p, and edge probabilities proportional to B.

There are two variants of the model. If `type="Hajek"`, new edges are connected with preferential attachment to any existing vertex but using the appropriate values of B as weights (see (Hajek and Sankagiri, 2019)). If `type="block_first"`, new edges are connected first to a community with probability proportional to the values of B, and then a vertex is chosen within that community with regular preferential attachment. In this case, the resulting degree distribution is scale-free (see (Renedo-Mirambell and Arratia, 2023) for a proof of this fact).

This is a simple example with just two communities and a graph of order 100 and size 400:

```
> B <- matrix(c(1, 0.2, 0.2, 1), ncol=2)
> G <- barabasi_albert_blocks(m=4, p=c(0.5, 0.5), B=B, t_max=100, type="Hajek",
                              sample_with_replacement = FALSE)
> plot(G, vertex.color=(V(G)$label), vertex.label=NA, vertex.size=10)
```
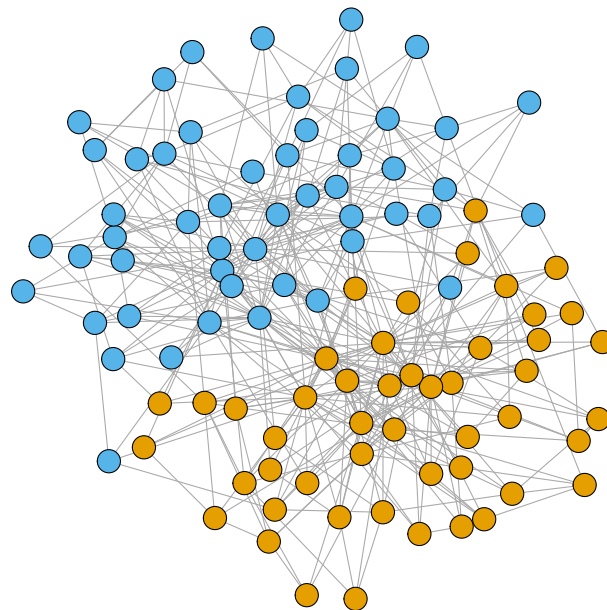

Figure 1: Example of the `barabasi_albert_communities` function with the community labels as vertex colors.

Finally, it is worth mentioning the `apply_subgraphs()` function, which is used internally in the package, but has also been made available to the user because it can be very convenient. It simply calls a function `f` on each of the communities of a graph (treated as it's own **igraph** object), acting as a wrapper for the `vapply` function. The communities are given as a membership vector `com`. For a very simple example, we call it to obtain the order of each of the factions of the karate club graph:

```
> apply_subgraphs(g=karate, com=V(karate)$Faction, f=gorder)
[1] 16 18
```

4   An introductory example

As a toy example we consider a famous benchmark social network: the Zachary's karate club graph (Zachary, 1977). First to showcase the graph randomization procedure `rewireCpp`, we apply it to the Zachary's karate club graph with the default settings (positive weights with no upper bound, which suits this graph):

```
> library(clustAnalytics)
> data(karate, package="igraphdata")
```

```
> rewired_karate <- rewireCpp(karate, weight_sel = "max_weight")
> par(mfrow=c(1,2), mai=c(0,0.1,0.3,0.1))
> plot(karate, main="karate")
> plot(rewired_karate, main="rewired_karate")
```

The resulting plots are shown in Figure 2.
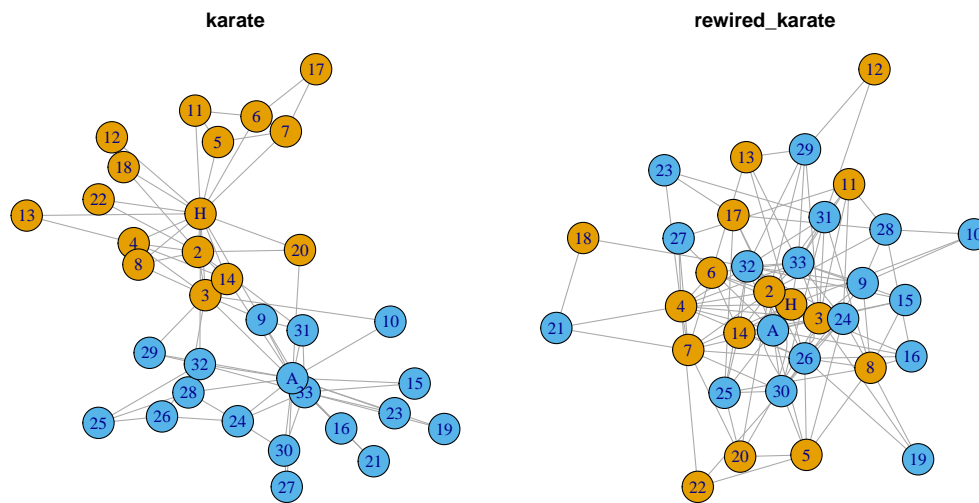
Figure 2: Karate club graph before and after the edge randomization process. Colors represent the faction of each participant, the *ground truth* clustering in this network.

The resulting rewired graph has lost its original communities centered around nodes *A* and *H*, and if any alternative community structure appears, it is only due to chance. Now we continue with an analysis of significance and stability of some known clustering algorithms on the Zachary's karate club graph.

4.1 Evaluating cluster significance

The function evaluate_significance takes the graph and a list of clustering functions as arguments. If the graph has a known *ground truth* community structure (such as the factions in the karate club), we can set ground_truth=TRUE and set gt_clustering as the membership vector to evaluate it and compare it to the results of the clustering algorithms. In our karate graph the ground truth is available with V(karate)$Faction.

```
> evaluate_significance(karate, ground_truth=TRUE,
+                   alg_list=list(Louvain=cluster_louvain,
+                                 "label prop"= cluster_label_prop,
+                                 walktrap=cluster_walktrap),
+                   gt_clustering=V(karate)$Faction)
```

	Louvain	label prop	walktrap	ground truth
size	9.58823529	10.52941176	10.00000000	17.05882353
internal density	1.29491979	1.29766214	1.32254902	0.76885813
edges inside	50.35294118	59.17647059	51.82352941	104.82352941
av degree	5.05882353	5.29411765	5.05882353	6.14705882
FOMD	0.26470588	0.29411765	0.26470588	0.41176471
expansion	3.47058824	3.00000000	3.47058824	1.29411765
cut ratio	0.14311885	0.12786548	0.14540629	0.07638889
conductance	0.25696234	0.22578022	0.25484480	0.09518717
norm cut	0.37937069	0.34206059	0.38131607	0.19090909
max ODF	0.43576990	0.42077566	0.51172969	0.38911607
average ODF	0.18336040	0.17884402	0.18493603	0.07498851
flake ODF	0.05882353	0.02941176	0.08823529	0.00000000
density ratio	0.87751142	0.88955342	0.86846364	0.90017702
modularity	0.41978961	0.41510519	0.41116042	0.37146614
graph_order	34.00000000	34.00000000	34.00000000	34.00000000
n_clusters	4.00000000	4.00000000	4.00000000	2.00000000

```
mean_cluster_size      8.50000000  8.50000000  8.50000000  17.00000000
coverage               0.74458874  0.77922078  0.74458874   0.90476190
global density ratio   0.75864388  0.76992481  0.74273256   0.80043860
VIdist_to_GT           0.90782167  0.82624391  0.87293838   0.00000000
```

If a *ground truth* clustering has been provided, the row VIdist_to_GT indicates the variation of information distance (Meilă, 2007) between that and each of the partitions. In this case the label propagation algorithm obtains the partition closest to the ground truth, while the Louvain algorithm is the furthest.

With the function evaluate_significance_r we compute the scoring functions as above, and we compare the results to those of a distribution of randomized graphs obtained with the rewiring method. The parameters of the rewiring method can be selected as shown in Table 1, in this case we specify weight_sel="max_weight", but we could also set an upper bound if appropriate to the graph. The resulting (default) table is shown below. This is a table with three columns per algorithm: the scores for the original graph, the mean of the corresponding scores for the rewired graphs and its percentile rank within the distribution of scores for rewired graphs. If parameter table_style = "string", the function instead returns a table with a column per algorithm where each element is of the format "original|rewired(percentile)".

```
> evaluate_significance_r(karate,
+                     alg_list=list(Lv=cluster_louvain,
+                                 "WT"= cluster_walktrap),
+                     weight_sel="max_weight", n_reps=100)
                        Lv          WT        Lv_r        WT_r  Lv_perc WT_perc
size                  9.58823529 10.00000000  8.1670588  8.3705882    0.86    0.81
internal density      1.29491979  1.32254902  1.3911098  1.3728923    0.45    0.55
edges inside         50.35294118 51.82352941 29.1011765 39.1491176    0.99    0.82
av degree             5.05882353  5.05882353  3.6779412  3.8764706    1.00    0.98
FOMD                  0.26470588  0.26470588  0.1723529  0.1955882    0.98    0.87
expansion             3.47058824  3.47058824  6.2323529  5.8352941    0.00    0.02
cut ratio             0.14311885  0.14540629  0.2383912  0.2325316    0.00    0.00
conductance           0.25696234  0.25484480  0.4755745  0.4825704    0.00    0.01
norm cut              0.37937069  0.38131607  0.6760032  0.7073753    0.00    0.00
max ODF               0.43576990  0.51172969  0.6456208  0.6610584    0.00    0.03
average ODF           0.18336040  0.18493603  0.4450276  0.4626576    0.00    0.01
flake ODF             0.05882353  0.08823529  0.3423529  0.4088235    0.00    0.01
density ratio         0.87751142  0.86846364  0.7441942  0.7591046    1.00    1.00
modularity            0.41978961  0.41116042  0.1797852  0.1545682    1.00    1.00
clustering coef       0.54148426  0.61381521  0.4306771  0.4253075    0.76    0.87
graph_order          34.00000000 34.00000000 34.0000000 34.0000000    0.00    0.00
n_clusters            4.00000000  4.00000000  4.8200000  6.4800000    0.04    0.05
mean_cluster_size     8.50000000  8.50000000  7.2306667  5.7914928    0.72    0.85
coverage              0.74458874  0.74458874  0.5413420  0.5705628    1.00    0.98
global density ratio  0.75864388  0.74273256  0.5296269  0.5789977    1.00    0.98
```

Depending on the application, more emphasis might be given to some of the metrics over others, but in most cases, those that take into account both internal and external connectivity (such as the modularity, conductance, or density ratio) will be the most relevant. Networks with significant clusters should result in values that are on the extremes of the distribution of rewired scores (so percentile rank one or close to one for the scores where higher values are better, and zero or close to zero for those where lower is better). The interpretation of the results is discussed in more detail in (Arratia and Renedo-Mirambell, 2021).

If one simply wishes to compare clustering algorithms against each other, though, the distribution of rewired scores is not necessary and the evaluate_significance function should be used instead.

4.2 Applying scoring functions

If it is the case that we already have some explicit community partition, but not the algorithm that produced it, we can assess its significance by applying the scoring functions directly to the network and the partition. To apply all scoring functions at once use scoring_functions with either type local or global:

```
> scoring_functions(karate, V(karate)$Faction, type="local")
  size internal density edges inside av degree      FOMD expansion  cut ratio
```

```
1   16        0.8250000              99  6.187500 0.5000000  1.375000 0.07638889
2   18        0.7189542             110  6.111111 0.3333333  1.222222 0.07638889
  conductance   norm cut   max ODF average ODF flake ODF density ratio modularity
1  0.10000000 0.1909091 0.3636364  0.05651941         0    0.9074074         NA
2  0.09090909 0.1909091 0.4117647  0.09140548         0    0.8937500         NA

> scoring_functions(karate, V(karate)$Faction, type="global")
        size internal density edges inside av degree      FOMD expansion  cut ratio
[1,] 17.05882        0.7688581    104.8235  6.147059 0.4117647  1.294118 0.07638889
     conductance   norm cut   max ODF average ODF flake ODF density ratio modularity
[1,]  0.09518717 0.1909091 0.3891161  0.07498851         0    0.900177  0.3714661
     graph_order n_clusters mean_cluster_size  coverage global density ratio
[1,]          34          2                17 0.9047619              0.8004386
```

Alternatively, we can apply the scoring functions individually. Each is called with the graph and the membership vector as arguments, and return a vector with the scores for each community:

```
> cut_ratio(karate, V(karate)$Faction)
[1] 0.07638889 0.07638889
```

```
> conductance(karate, V(karate)$Faction)
[1] 0.10000000 0.09090909
```

A case in point are the clustering coefficient and transitivity. As they can be applied to weighted graphs in general and not only to their partition into communities, they are simply called with the graph as the only argument:

```
> weighted_clustering_coefficient(karate)
[1] 0.8127164
```

To be able to obtain the result for every community in the graph, we provide the function `apply_subgraphs`; which given a graph, a membership vector and a scalar function that takes a graph as input, applies the function to every community and returns the vector of results. In this case it works as follows:

```
> apply_subgraphs(karate, V(karate)$Faction, weighted_clustering_coefficient)
[1] 0.9514233 0.7783815
```

4.3 Evaluating cluster stability

Here we perform a nonparametric bootstrap to the karate club graph and the same selection of algorithms. For each instance, the set of vertices is resampled, the induced graph is obtained by taking the new set of vertices with the induced edges from the original graph, and the clustering algorithms are applied. Then, these results are compared to the induced original clusterings using the metrics mentioned above: the variation of information (VI), the normalized reduced mutual information (NRMI), and both adjusted and regular Rand index (Rand and adRand). Their exact definitions can be found in (Arratia and Renedo-Mirambell, 2021).

```
> boot_alg_list(g=karate, return_data=FALSE, R=99,
+              alg_list=list(Louvain=cluster_louvain,
+                            "label prop"= cluster_label_prop,
+                            walktrap=cluster_walktrap))
             Louvain label prop  walktrap
VI         0.2657555  0.3623330 0.2608622
NRMI       0.7024417  0.3415649 0.6959898
Rand       0.8584598  0.5969139 0.8609266
AdRand     0.6457648  0.2574423 0.6645099
n_clusters 5.9191919  5.1313131 6.3030303
```

Note that in this table the variation of information is a distance, so lower values indicate similar partitions, while for the NRMI, Rand, and adRand, higher values mean the partitions are more similar (1 means they are the same partition). Therefore, algorithms that produce stable clusters should result in low values of VI, and high values of the rest of metrics. In this example network we can see how the Louvain and walktrap algorithms have similar stability, while the label propagation algorithm performs much worse, and this is reflected in all metrics.

4.4 Clustering assessment on synthetic ground truth networks

We can evaluate the significance and stability of clusters produced by a set of clustering algorithms on a network with known community synthetically created with the stochastic block model (with function `sample_sbm`) or the preferential attachment model (with `barabasi_albert_blocks`). The former produces a network with binomial degree distribution, and the latter produces networks with scale-free degree distribution.

Let us generate a graph from a stochastic block model in which we set very strong clusters: the elements in the diagonal of the matrix are much larger than the rest, so the probability of intra-cluster edges is much higher than that of inter-cluster edges.

```
> pm <- matrix (c(.3, .001, .001, .003,
                  .001, .2, .005, .002,
                  .001, .005, .2, .001,
                  .003, .002, .001, .3), nrow=4, ncol=4)
> g_sbm <- igraph::sample_sbm(100, pref.matrix=pm, block.sizes=c(25,25,25,25))
> E(g_sbm)$weight <- 1
> memb <- c(rep(1,25), rep(2,25), rep(3,25), rep(4,25))
> significance_table_sbm <- evaluate_significance(g_sbm, gt_clustering=memb)
> significance_table_sbm
                         Louvain    label prop      walktrap ground truth
size                25.00000000  2.144000e+01   25.00000000   25.00000000
internal density     0.26500000  3.228846e-01    0.26500000    0.26500000
edges inside        79.50000000  6.968000e+01   79.50000000   79.50000000
av degree            3.18000000  3.010000e+00    3.18000000    3.18000000
FOMD                 0.43000000  4.000000e-01    0.43000000    0.43000000
expansion            0.24000000  5.800000e-01    0.24000000    0.24000000
cut ratio            0.00320000  6.907324e-03    0.00320000    0.00320000
conductance          0.03812704  1.091380e-01    0.03812704    0.03812704
norm cut             0.05064474  1.272639e-01    0.05064474    0.05064474
max ODF              0.29047619  3.554762e-01    0.29047619    0.29047619
average ODF          0.03789358  1.068777e-01    0.03789358    0.03789358
flake ODF            0.00000000  1.000000e-02    0.00000000    0.00000000
density ratio        0.98728802  9.778080e-01    0.98728802    0.98728802
modularity           0.69994949  6.663131e-01    0.69994949    0.69994949
clustering coef      0.28604270  3.302172e-01    0.28604270    0.28604270
graph_order        100.00000000  1.000000e+02  100.00000000  100.00000000
n_clusters           4.00000000  6.000000e+00    4.00000000    4.00000000
mean_cluster_size   25.00000000  1.666667e+01   25.00000000   25.00000000
coverage             0.96363636  9.121212e-01    0.96363636    0.96363636
global density ratio 0.97584906  9.498650e-01    0.97584906    0.97584906
VIdist_to_GT         0.00000000  3.403855e-01    0.00000000    0.00000000
```

In this case, `memb` is the membership vector of the ground truth clusters of the model. The clusters in the network are so strong that both the Louvain and walktrap algorithms manage to identify and match them exactly (their VI distance to the ground truth clustering is zero).

We now assess for stability of the clustering algorithms on this sbm graph:

```
> b_sbm <- boot_alg_list(g=g_sbm, return_data=FALSE, R=99)
> b_sbm

            Louvain label prop  Walktrap
VI        0.1234341  0.1769217 0.1178832
NRMI      0.8536997  0.7841236 0.8656356
Rand      0.9411244  0.9230160 0.9472768
AdRand    0.8306925  0.7651778 0.8476909
n_clusters 6.9797980  7.7070707 7.4646465
```

We can clearly see that for all metrics, the results are much more stable than in the previous example, which makes sense because we purposefully created the SBM graph with very strong clusters.

5 The clustAnalytics package in context of related R packages

The outstanding quality of clustAnalytics is that it is a set of robust and efficient measures for assessing significance and stability of clustering algorithms on graphs with the convenience of working with igraph objects, which makes it a valuable complement to the igraph package (Csardi and Nepusz, 2006). A revision of the *CRAN Task View: Cluster Analysis & Finite Mixture Models* shows that there are very few packages devoted to assessing quality of clusters in general, and none for igraph graphs as input. One could use in a limited manner some of the existing packages by converting igraph graphs to their adjacency matrices, but then quality evaluation follows different paradigms not quite pertaining to networks. For instance, the package ClustAssess (Shahsavari et al., 2022) conceived for evaluating robustness of clustering of single-cell RNA sequences data using proportion of ambiguously clustered pairs, as well as similarity across methods and method stability using element-centric clustering comparison; sigclust (Huang et al., 2014) which provides a single function to assess the statistical significance of splitting a data set into two clusters; clValid (Brock et al., 2021) implements Dunn Index, Silhouette, Connectivity, Stability, BHI and BSI, for a statistical and biological-based validation of clustering results. None of these apply directly to igraph objects, and were not conceived for the analysis of clustering in social networks.

References

A. Arratia and M. Renedo-Mirambell. Clustering assessment in weighted networks. *PeerJ Computer Science*, 7(e600):1–27, 2021. [p134, 135, 136, 137, 140, 141]

V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008. doi: http://dx.doi.org/10.1088/1742-5468/2008/10/P10008. [p134]

G. Brock, V. Pihur, S. Datta, and S. Datta. *clValid: Validation of Clustering Results*, 2021. URL https://CRAN.R-project.org/package=clValid. R package version 0.7. [p143]

G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5):1–9, 2006. [p134, 143]

S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75 – 174, 2010. doi: https://doi.org/10.1016/j.physrep.2009.11.002. [p134]

B. Hajek and S. Sankagiri. Community recovery in a preferential attachment graph. *IEEE Transactions on Information Theory*, 65(11):6853–6874, 2019. ISSN 0018-9448. doi: 10.1109/TIT.2019.2927624. [p138]

C. Hennig. Cluster-wise assessment of cluster stability. *Computational Statistics & Data Analysis*, 52(1): 258–271, 2007. doi: https://doi.org/10.1016/j.csda.2006.11.025. [p135]

H. Huang, Y. Liu, and J. S. Marron. *sigclust: Statistical Significance of Clustering*, 2014. URL https://CRAN.R-project.org/package=sigclust. R package version 1.1.0. [p143]

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. doi: https://doi.org/10.1007/BF01908075. [p135]

M. P. McAssey and F. Bijma. A clustering coefficient for complete weighted networks. *Network Science*, 3(2):183–195, 2015. doi: http://dx.doi.org/10.1017/nws.2014.26. [p136]

M. Meilă. Comparing clusterings - an information based distance. *Journal of Multivariate Analysis*, 98 (5):873 – 895, 2007. doi: http://dx.doi.org/10.1016/j.jmva.2006.11.013. [p135, 140]

R. Milo, N. Kashtan, S. Itzkovitz, M. E. J. Newman, and U. Alon. On the uniform generation of random graphs with prescribed degree sequences. *Arxiv preprint cond-mat/0312028*, 2003. [p136]

M. E. J. Newman, G. T. Cantwell, and J.-G. Young. Improved mutual information measure for clustering, classification, and community detection. *Phys. Rev. E*, 101:042304, Apr 2020. doi: 10.1103/PhysRevE.101.042304. URL https://link.aps.org/doi/10.1103/PhysRevE.101.042304. [p134, 135, 137]

P. Pons and M. Latapy. Computing communities in large networks using random walks. In *International symposium on computer and information sciences*, pages 284–293. Springer, 2005. doi: https://doi.org/10.1007/11569596_31. [p134]

U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3), Sep 2007. [p134]

A. R. Rao, R. Jana, and S. Bandyopadhyay. A markov chain monte carlo method for generating random (0, 1)-matrices with given marginals. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, 58 (2):225–242, 1996. ISSN 0581572X. [p136]

M. Renedo-Mirambell and A. Arratia. Identifying bias in network clustering quality metrics. *PeerJ Computer Science*, 9:e1523, 2023. doi: https://doi.org/10.7717/peerj-cs.1523. [p138]

A. Shahsavari, A. Munteanu, and I. Mohorianu. *ClustAssess: Tools for Assessing Clustering*, 2022. URL https://CRAN.R-project.org/package=ClustAssess. R package version 0.3.0. [p143]

J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015. doi: http://dx.doi.org/10.1007/s10115-013-0693-z. [p134, 135]

W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977. [p138]

Martí Renedo-Mirambell
Department of Computer Sciences,
Polytechnical University of Catalonia,
Barcelona, SPAIN
marti.renedo@upc.edu

Argimiro Arratia
Soft Computing Research Group (SOCO)
Intelligent Data Science and Artificial Intelligence Research Center,
Department of Computer Sciences,
Polytechnical University of Catalonia,
Barcelona, SPAIN
(ORCID ID: 0000-0003-1551-420X)
argimiro@cs.upc.edu

PINstimation: An R Package for Estimating Probability of Informed Trading Models

by Montasser Ghachem and Oguz Ersan

Abstract The purpose of this paper is to introduce the R package PINstimation. The package is designed for fast and accurate estimation of the probability of informed trading models through the implementation of well-established estimation methods. The models covered are the original PIN model (Easley and O'Hara 1992; Easley et al. 1996), the multilayer PIN model (Ersan 2016), the adjusted PIN model (Duarte and Young 2009), and the volume- synchronized PIN (Easley, De Prado, and O'Hara 2011; Easley, López De Prado, and O'Hara 2012). These core functionalities of the package are supplemented with utilities for data simulation, aggregation and classification tools. In addition to a detailed overview of the package functions, we provide a brief theoretical review of the main methods implemented in the package. Further, we provide examples of use of the package on trade-level data for 58 Swedish stocks, and report straightforward, comparative and intriguing findings on informed trading. These examples aim to highlight the capabilities of the package in tackling relevant research questions and illustrate the wide usage possibilities of PINstimation for both academics and practitioners.

1 Introduction

Informed trading indicates the presence of information asymmetry in a given market, and is usually attributed to trading with better-quality information and/or more sophisticated tools for analyzing available information (see Ahn et al., 2008). Given its impact on prices and liquidity, researchers have dedicated considerable effort to the measurement of informed trading, and to the characterization of its relevant aspects (Berkman et al., 2014; Chang et al., 2014; Bongaerts et al., 2014; Hsieh and He, 2014; Yin and Zhao, 2015; Guo and Qiu, 2016). The growth in informed trading measures has been made possible thanks to the availability of rich datasets, and as a response to the continuously evolving nature of trading in financial markets.

Despite the plethora of alternative and more recent measures, "fundamental" measures developed by the pioneering works are still widely used in academic research. Some prominent measures include: relative trade informativeness measure (Hasbrouck, 1991), percentage-price-impact measure (Huang and Stoll, 1996), adverse selection component (Huang and Stoll, 1997) and the adverse information parameter (Madhavan et al., 1997). Above the rest, the probability of informed trading (PIN; Easley and O'Hara, 1992; Easley et al., 1996) has probably been the most widely used measure of informed trading in the literature. Easley and O'Hara, beginning with their foundational work in 1987 and continuing through subsequent studies in the 1990s and 2000s, developed, tested and refined the PIN measure to quantify informed trading in financial markets. A major factor behind the persistent wide use (prominence) of the PIN model lies in the branch of studies addressing the limitations of the model, remedying to the challenges of its estimation; and extending the original model. Due to the rapid evolution of trading in financial markets, the estimation of the original PIN model has become vulnerable to errors; and the model – and its assumptions – as it was first suggested has faced difficulties in matching the real world. Over the years, many extensions and improvements to the PIN model have been developed, addressing various shortcomings of the original model and estimation challenges. However, because of their more complex theoretical underpinnings and implementation details, most of these models have not been adopted by the wider academic and practitioner audience. To address these issues, the **PINstimation** package seeks to provide easy and convenient access to these extensions of the PIN model. To this end, the package is designed in a compact structure allowing users to directly obtain informed-trading estimates solely by the use of an intraday trading data. The package includes easy-to-use functions, that accurately implement preexisting, and novel remedial solutions to estimation challenges as suggested in the literature. Besides, it provides a rich toolbox for simulating datasets, something that can help researchers conduct robust, and reliable comparative analyses. By the introduction of the package, we hope to contribute to further use of the PIN models in academic research, to improve the validity, and quality of scientific findings within the field, and eventually to heighten the interest of practitioners in these models.

To our knowledge, there are two packages available for the estimation of PIN models: **pinbasic** (Recktenwald, 2018, 2019) and **InfoTrad** (Celik and Tiniç, 2017, 2018). Both packages have limited scope as they solely focus on the original PIN model (Easley et al., 1996). In addition to scope differences,

other motivations for users to shift to **PINstimation** are that (1) the package **pinbasic** has recently been placed in the archive by CRAN (2) the package **InfoTrad** in its current version (V.1.2) is not error-free.[1]

    PINstimation contains functions to estimate probability of informed trading (PIN) as introduced by Easley and O'Hara (1992), and Easley et al. (1996). The estimation procedures implemented in these functions help to avoid floating point errors, boundary solutions, and convergence to local maxima. Besides, the package provides a comprehensive treatment of two important extensions of the PIN model. The multilayer probability of informed trading model (MPIN model; Ersan, 2016), in contrast to the original PIN model, allows for multiple information types, and assumes that information events cluster in layers with uniform informed trading intensity. Relaxing the assumption of a unique information type allows for a more realistic, and accurate treatment of informed trading. However, it poses, at least, two additional challenges: (1) the larger parameter space of the MPIN model makes it more likely that the maximum likelihood estimate may lie on the parameter boundary, and (2) An accurate determination of the number of information layers is crucial to produce reliable estimations of the probability of informed trading. **PINstimation** tackles these two issues by including a function to generate strategic[2] initial parameter sets, and three functions for estimating the number of information layers in datasets. The second extension is the adjusted probability of informed trading model (AdjPIN model; Duarte and Young, 2009). This model challenges the assumption that trading is only performed by uninformed liquidity traders and informed traders, and accounts for the possibility of liquidity shocks to both the buy and sell side. **PINstimation** provides functions to estimate the AdjPIN measure and the PSOS (probability of a symmetric order flow shock), as well as three functions to generate initial sets of parameters for maximum-likelihood estimation. In addition to the standard maximum-likelihood method, the package provides a novel implementation of the estimation of PIN models via the expectation-conditional maximization algorithm. The speed, and accuracy of this algorithm has been recently documented in Ghachem and Ersan (2022). As for informed trading in high-frequency settings, **PINstimation** enables users to estimate the volume-synchronized probability of informed trading (VPIN; Easley et al., 2011, 2012). This measure is an adaptation of the PIN measure to the high-frequency trading, and aims to capture the order flow toxicity in a trading data. Finally, the package offers two supporting utilities: (1) a rich simulation toolbox to simulate data according to the assumptions of the different PIN models and, thereby, test the accuracy of estimation algorithms, and (2) a fast implementation of the prominent trade classification algorithms that allow users to generate daily sequences of buyer-initiated, and seller-initiated trades from raw trading level data. Such sequences are to be used later as inputs for the estimations of PIN, MPIN, and AdjPIN models.

    The remainder of this paper is organized as follows. Next section provides a brief introduction to the theoretical background of PIN models. Third section presents a detailed description of the package and illustrates its applications through several examples. Fourth section reports and discusses the results of two empirical investigations conducted using the package. The last section concludes with a summary of the package capabilities and a discussion of its potential extensions.

2 Theoretical background

2.1 PIN model

Easley and O'Hara (1992) developed a model where the change in the order imbalance is associated to the presence of informed trading. The information can be either positive, leading to excess trading on the buy side, or negative, leading to excess trading on the sell side. On days with no information event, there are only uninformed traders in the market. On the days with a good-information (bad-information) event, informed buyers (sellers) join uninformed buyers and sellers to trade on the information. Statistically, Easley et al. (1996) model total trades by a finite Poisson mixture model, where the numbers of buyer-initiated and seller-initiated trades; follow each a Poisson distribution.

---

[1]In fact, two of the five functions suffer from implementation errors. The function EA() implements the algorithm of Ersan and Alıcı (2016), but performs the clustering process inaccurately: the days within the information-event cluster are distributed into good-event and bad-event days via a clustering step based on order imbalance rather than the actual step of grouping them into two based on the sign of order imbalance. The function YZ() of the same package implements the algorithm of Yan and Zhang (2012). It, however, contains an error in the denominator of the PIN formula. The correct formula should be $PIN = \alpha\mu/(\alpha\mu + \varepsilon_b + \varepsilon_s)$. This error might impact the results in research papers using the package (See, for instance, Figure 12 in Griffin et al. (2021) – very poor performance of PIN estimates using YZ()). Our comparative tests confirm those observations, as the mean absolute errors in PIN estimates of **InfoTrad** and **PINstimation** implementations are 0.02476, and 0.00014 respectively for Yan and Zhang (2012); and 0.00777 and 0.00014 respectively for Ersan and Alıcı (2016).

[2]Strategic initial parameter sets stand in contrast to those obtained through random selection or grid-search methods, as they are derived from the characteristics of the dataset used for the estimation. They are typically limited in number and meticulously selected to cover relevant areas in the parameter space, ensuring a more accurate and efficient optimization process.

The likelihood of observing B buyer-initiated trades (or buys) and S seller-initiated trades (or sells) on a trading day is stated as:

$$L(B,S|\Theta) = \alpha(1-\delta)e^{-(\mu+\varepsilon_b)}\frac{(\mu+\varepsilon_b)^B}{B!}e^{-\varepsilon_s}\frac{\varepsilon_s^S}{S!} + \alpha\delta e^{-\varepsilon_b}\frac{\varepsilon_b^B}{B!}e^{-(\mu+\varepsilon_s)}\frac{(\mu+\varepsilon_s)^S}{S!} + (1-\alpha)e^{-\varepsilon_b}\frac{\varepsilon_b^B}{B!}e^{-\varepsilon_s}\frac{\varepsilon_s^S}{S!} \quad (1)$$

where $\Theta = (\alpha, \delta, \mu, \varepsilon_b, \varepsilon_s)$ is the set of parameters to be estimated: α is the probability of occurrence of information events, δ is the conditional probability that the information event is a bad event, μ is the informed trading intensity, and ε_b and ε_s are uninformed trading intensities on the buy and sell sides, respectively. For a time period of N days, the joint likelihood of observing a set of daily buys and sells, $M = (B_i, S_i)_{i=1}^N$ is presented as:

$$\mathcal{L}(M|\Theta) = \prod_{i=1}^{N} L(B_i, S_i|\Theta) \quad (2)$$

Typically, the estimation of the five parameters is performed via maximum likelihood estimation (MLE). Once the parameter set Θ is estimated, the probability of informed trading (PIN) is calculated as:

$$PIN = \frac{\alpha\mu}{\alpha\mu + \varepsilon_b + \varepsilon_s} \quad (3)$$

The PIN model relies on several assumptions. First, trading days are assumed to be independent of each other, an assumption that leads to the joint likelihood in Eq.(2). Tests on the validity of independence assumption provide supportive evidence and sample results are reported in Easley et al. (1997). Second, information events are assumed to occur outside trading hours. Third, at most one information event can occur in any given trading day. Finally, information events are assumed to be of a single type, i.e., leading to the same magnitude of informed trading μ, whenever they occur.

2.2 MPIN model

The MPIN model (Ersan, 2016) is a generalization of PIN model that allows for multiple information event types (information layers). When the number of layers J equals to 1, then the model is simplified to the original PIN model. The model relaxes several assumptions of the PIN model. First, information events can be of different types, i.e., generate different magnitudes of informed trading. Second, more than one information event can occur at any given day. Third, the model allows for the occurrence of information events within trading hours. The model's ability to handle multiple information types enables these two last features. It can aggregate the effects of multiple events or identify instances of partially disseminated informed trading on any given day by introducing an additional layer.

The parameter set of an MPIN model with J layers $\Theta_m = (\alpha_1, \ldots, \alpha_J, \delta_1, \ldots, \delta_J, \mu_1, \ldots, \mu_J, \varepsilon_b, \varepsilon_s)$ has length $3J + 2$, where $(\alpha_j)_{j=1\ldots J}$ is the probability of occurrence for an information event in layer j, $(\delta_j)_{j=1\ldots J}$ is the (conditional) probability the event in layer j is a bad-information event, $(\mu_j)_{j=1\ldots J}$ is the informed trading intensity in layer j, ε_b and ε_s are the uninformed trading intensities. Similar to the PIN model, the multilayer probability of informed trading (MPIN) is the ratio of expected informed trading intensity to the expected total trading intensity as:

$$MPIN = \frac{\sum_{j=1}^{J} \alpha_j \mu_j}{\sum_{j=1}^{J} \alpha_j \mu_j + \varepsilon_b + \varepsilon_s} \quad (4)$$

The estimation of the MPIN model using the standard maximum-likelihood estimation requires a prior estimation of the number of information layers in the data. An algorithm for detecting the number of layers in a dataset has already been suggested by Ersan (2016). Ersan and Ghachem (2022a) improved this algorithm by refining the correction for the order imbalance.

2.3 AdjPIN model

Duarte and Young (2009) suggest an alternative, extended informed trading model, to address two main concerns. First, for many stocks, there is a well-documented positive correlation between the numbers of buyer- and seller-initiated trades (Duarte and Young, 2009). This fact cannot be modelled by the original PIN model. Second, it is difficult to capture the large variance of buys and sells by the use of PIN model, if investors are restricted to be of two types: informed and liquidity traders. Accordingly, the authors introduce an extended model, in which a symmetric order-flow shock to both buy and sell sides is introduced. On any given day, in addition to information events, a positive liquidity shock, symmetric in buys and sells, can occur. In addition to the adjusted PIN measure (AdjPIN) capturing

the probability of informed trading, the model introduces the probability of symmetric order flow shock (PSOS) that measures the probability of a trade to occur due to a symmetric liquidity shock. The parameter set of the original AdjPIN model $\Theta_a = (\alpha, \delta, \theta, \theta', \mu_b, \mu_s, \varepsilon_b, \varepsilon_s, \Delta_b, \Delta_s)$ has 10 elements: α is the probability of occurrence of an information event; δ is the probability that the information event is a bad event; μ_b (μ_s) is the informed trading intensity on the buy (sell) side; ε_b (ε_s) is the uninformed trading intensity on the buy (sell) side. θ (θ') is the probability of a symmetric order flow shock occurrence in the absence (presence) of an information event. Δ_b (Δ_s) is the additional arrival rate of buys (sells) caused by symmetric liquidity shocks. Once the parameter set Θ_a is estimated, typically through MLE, AdjPIN and PSOS are calculated as follows:

$$\text{AdjPIN} = \frac{\alpha\left(\delta\mu_s + (1-\delta)\,\mu_b\right)}{\alpha\left(\delta\mu_s + (1-\delta)\,\mu_b\right) + (\Delta_b + \Delta_s)\left(\alpha\theta' + (1-\alpha)\,\theta\right) + \varepsilon_b + \varepsilon_s} \tag{5}$$

$$\text{PSOS} = \frac{(\Delta_b + \Delta_s)\left(\alpha\theta' + (1-\alpha)\,\theta\right)}{\alpha\left(\delta\mu_s + (1-\delta)\,\mu_b\right) + (\Delta_b + \Delta_s)\left(\alpha\theta' + (1-\alpha)\,\theta\right) + \varepsilon_b + \varepsilon_s} \tag{6}$$

2.4 Computation issues for PIN, MPIN, and AdjPIN estimations

PIN estimation is prone to two main sources of numerical errors. First, large numbers of trades (buys and sells) in the power terms (Eq 2) can lead to floating point exception problem.[3] While this was not a problem in 1990's, most stocks in developed markets today are traded tens of thousands of times a day, rendering the likelihood function in Eq (2) numerically intractable. Consequently, several logarithmic transformations (factorizations) of the likelihood function have been suggested to address this problem. Easley et al. (2008) were the first authors to suggest a factorization of the likelihood function, however their transformation is shown to generate non-negligible biases (Lin and Ke, 2011; Yan and Zhang, 2012). Lin and Ke (2011) provide another factorization leading to more accurate estimates. Finally, Ersan (2016) suggests a similar, yet simpler factorization that leads to the same results as with Lin and Ke (2011), yet in shorter estimation times. More importantly, the Ersan (2016) factorization is easily generalized for the MPIN model. In line with previous efforts, Ersan and Ghachem (2022b) have suggested a factorization of the likelihood function of the AdjPIN model.

Second issue related to the estimation of PIN, MPIN, and AdjPIN models is that the estimation procedure may not reach the global maximum of the (factorized) likelihood function. Several papers document that the ML estimation of the PIN model frequently yields boundary solutions, not the global maxima (Yan and Zhang, 2012; Gan et al., 2015; Ersan and Alıcı, 2016). As a remedial solution, Gan et al. (2015) suggests the use of a single strategic parameter set generated by their hierarchical clustering algorithm. In contrast, Yan and Zhang (2012) recommend that the MLE procedure is started up to $125\,(5 \times 5 \times 5)$ times using the initial parameter sets from their grid search algorithm and that the highest likelihood estimates are picked. Similarly, Ersan and Alıcı (2016) settle for multiple MLE runs, but recommend five sets of parameters determined by their clustering algorithm, and show them to be sufficient to reach the global maxima. When compared to PIN model, achieving global maxima in MPIN model is harder given the larger dimension of the parameter set ($3J + 2$ parameters). The generalization of Yan and Zhang (2012) grid search algorithm would require up to 5^9 runs of MLE. In contrast, the clustering algorithm of Ersan and Alıcı (2016) is easily generalized, and in its basic setting, produces $\binom{J+5}{J}$ initial parameter sets. As for the AdjPIN model, generating initial parameter sets turned out to be challenging, given its large parameter set, and that preexisting generation algorithms do not allow a straightforward adaptation to the model. Therefore, a large number of studies relied on a limited number of randomly generated initial parameter sets (see e.g. Duarte and Young, 2009). Recently, Cheng and Lai (2021) suggested an extension of the grid-search algorithm of Yan and Zhang (2012), while Ersan and Ghachem (2022b) suggested a novel method loosely based on the algorithm of Ersan and Alıcı (2016).

2.5 The expectation-maximization algorithm

The estimation of PIN models has typically been performed through a direct maximization of the corresponding factorization of the likelihood function. The use of alternative estimation methods such as the Gibbs sampler has also been recently suggested (Griffin et al., 2021). More recently, Ghachem and Ersan (2022) have suggested the use of a variant of the expectation-maximization (EM) algorithm to estimate PIN models. In statistics, the EM algorithm is an iterative method for finding maximum likelihood estimates of parameters in finite-mixture models, where the model may depend on unobserved latent variables (Ng et al., 2012). In finite mixture models, each data observation is

---

[3]Statistical software make calculations in limited ranges. R calculates, e.g., between $\exp(-745)$ and $\exp(709)$.

associated with an unobserved cluster label, i.e. a reference to the cluster it belongs to. In this respect, PIN models can be considered as a Poisson mixture model with a finite number of clusters (Lin and Lee, 2015; Ghachem and Ersan, 2022). Ghachem and Ersan (2022) considered a variant of the EM algorithm, the Expectation-Conditional Maximization algorithm (ECM algorithm), for the estimation of the PIN models, and provided a detailed implementation and an empirical assessment of it. They show that the ECM algorithm yields faster and more accurate estimates than alternative methods.

2.6 VPIN measure

Volume-synchronized probability of informed trading (VPIN) metric is introduced by Easley et al. (2011), and Easley et al. (2012). VPIN aims at detecting order flow toxicity in high-frequency financial markets. As Easley et al. (2012) define, "*order flow is toxic when it adversely selects market makers, who may be unaware they are providing liquidity at a loss*". It is shown that order flow becomes toxic prior to intraday shocks, such as the 2010 Flash Crash (Easley et al., 2011). VPIN metric proceeds with the volume of trades that arrive to the market, rather than number of trades. In a high-frequency framework, VPIN uses volume clock rather than time clock, forming equal sized volume buckets intraday. A new trade classification algorithm - bulk volume classification - is suggested by the authors. Accordingly, trades are aggregated in short time intervals (e.g., 1 minute) and standardized price changes are used in distributing trades into buys and sells. As shown in Easley et al. (2008), informed trading probability from the PIN model can be proxied by the ratio of expected trade imbalance to the expected total volume of trades. In line with that, VPIN is calculated as follows:

$$\text{VPIN} = \frac{E\left[\left|V_\tau^{Sell} - V_\tau^{Buy}\right|\right]}{E\left[V_\tau^{Sell} + V_\tau^{Buy}\right]} = \frac{\sum_{\tau=1}^n OI_\tau}{n \times V} \tag{7}$$

where V is the predetermined volume bucket size and equals to $V_\tau^{Sell} + V_\tau^{Buy}$ in that bucket. OI is the order imbalance. In Easley et al. (2012), volume bucket size is determined by dividing the average daily volume by 50. Each volume is filled by aggregating the short time bars. In addition to the time bar (t) and volume bucket size (V), third parameter in VPIN calculation is the sample length (n) that determines how many volume buckets to be included. Thus, VPIN at any time is calculated based on the last n volume buckets. It is updated with each new volume bucket in a rolling window process.

3 The PINstimation package

The R package **PINstimation** provides utilities for the estimation of PIN models, partitioned into six categories:

- The standard PIN model (Easley and O'Hara, 1992; Easley et al., 1996), including various tools that remedy to floating-point exception, provide efficient algorithms for initial parameter sets and treat boundary solutions (Lin and Ke, 2011; Yan and Zhang, 2012; Gan et al., 2015; Ersan and Alıcı, 2016; Ke et al., 2019);

- Multilayer probability of informed trading or MPIN (Ersan, 2016) and tools for respective computational issues;

- Adjusted probability of informed trading or AdjPIN (Duarte and Young, 2009) and tools for respective computational issues;

- Volume-synchronized probability of informed trading or VPIN (Easley et al., 2011, 2012);

- Simulation utilities that generate datasets for testing and benchmarking the different PIN model estimation methods;

- Trade classification via commonly used algorithms and daily aggregation of buyer-initiated and seller-initiated trades.

3.1 Standard PIN model functions

The different factorizations of the likelihood function can be specified using the family of functions of the form fact_pin_*, where the suffix (*) can be one of ("eho", "lk", "e"), corresponding to the factorization of Easley et al. (2010), Lin and Ke (2011), and Ersan (2016) respectively. The different algorithms for the generation of initial parameter sets are implemented in the family of functions of the form initials_pin_*, where the suffix (*) can be one of ("yz", "gwj", "ea"), corresponding to the algorithm of Yan and Zhang (2012), Gan et al. (2015), and Ersan and Alıcı (2016) respectively.

The family of functions of the form `pin_*` allows the estimation of the PIN model using the afore-mentioned algorithms for the generation of initial parameter sets, where the suffix (`*`) can be one of ("yz","gwj","ea"). The function `pin()` estimates the PIN model using custom initial parameter sets.

These functions take the two arguments: `data`, and `factorization`. The `data` argument is a data frame that contains daily data of buyer-initiated trades or buys in the first column, and seller-initiated trades or sells in the second column. The argument `data` is usually a dataset with around 60 (250) rows as representative of a quarterly (yearly) data while any custom length can be determined by the user. The `factorization` argument referring to the likelihood function factorization used for the maximum likelihood maximization. It can be one of ("none","EHO","LK","E").

Estimation output

The output of the estimation functions `pin()`, `pin_yz()`, `pin_gwj()` and `pin_ea()` is an S4 object of class `estimate.pin`. The slots of this object are presented in Table S1.

Examples

We estimate the PIN model using a preloaded dataset called `dailytrades`.

```
# [1] Estimate the PIN model using the function pin_ea()

library("PINstimation")
model_ea <- pin_ea(dailytrades)
show(model_ea)
## ---------------------------------
## PIN estimation completed successfully
## ---------------------------------
## Initial parameter sets   : Ersan and Alici (2016)
## Likelihood factorization : Ersan (2016)
## ---------------------------------
## 5 initial set(s) are used for estimation
## Type object@initialsets to see the initial parameter sets used
##
##  PIN model
##
## =========== ===========
## Variables    Estimates
## =========== ===========
## alpha        0.749997
## delta        0.133334
## mu           1193.52
## eps.b        357.27
## eps.s        328.63
## ----
## Likelihood   (3226.469)
## PIN          0.566172
## =========== ===========

# [2] Display the optimal parameter estimates and the PIN value

model_ea@parameters
##     alpha       delta          mu       eps.b        eps.s
##   0.7499975   0.1333342 1193.5179655  357.2659099  328.6291793

model_ea@pin
## [1] 0.5661721
```

3.2 MPIN model functions

The factorization of the likelihood function of the MPIN model can be evaluated using the function `fact_mpin()`. The initial sets of parameters can be obtained using a generalization of the clustering algorithm developed by Ersan (2016) via the function `initials_mpin()`. The number of layers in

datasets can be detected using the family of functions of the form detectlayers_*, where the suffix (*) can be one of ("e","eg","ecm"), corresponding to the layer detection algorithm of Ersan (2016), Ersan and Ghachem (2022a), and Ghachem and Ersan (2022) respectively.

The function mpin_ml() estimates this probability using the standard maximum likelihood estimation method, the factorization of Ersan (2016), and the initial parameter sets in Ersan and Alıcı (2016). The function mpin_ml() takes as an argument layers that specifies the number of information layers assumed to be present in the data. If the user omits this argument, the number of layers is detected using the algorithm referred to in the argument detectlayers. This number is then used to generate the initial parameter sets, before proceeding to compute the maximum likelihood estimates of the MPIN model. The function mpin_ecm() estimates the MPIN model via the ECM algorithm. The function mpin_ecm() takes as an argument the number of information layers layers assumed to be present in the data. If this number is provided by the user, the function finds the optimal estimates for each of the initial parameter sets, and then selects the parameter estimates that give the highest likelihood. If the argument layers is omitted, then the function performs the aforementioned estimation for each number of layers in the integer set from 1 to 8, and then select the optimal model having the lowest model selection criterion. The default criterion is the Bayesian Information Criterion or BIC. The function selectModel() allows to change the selection criterion.

Estimation output

The outputs of the functions mpin_ml() and mpin_ecm() are two S4 objects of class estimate.mpin, and estimate.mpin.ecm respectively. The latter object inherits all slots of the former, with a few additional slots: Three slots for information criteria (@AIC, @BIC, and @AWE), one slot for the hyperparameters (@hyperparams), one slot stating whether the information criterion is used (@optimal), and one slot for the active information criterion (@criterion). Common slots of both objects are presented in Table S2. Additional slots of estimate.mpin.ecm objects are described in Table S3.

Examples

We estimate the MPIN model using the preloaded dataset dailytrades.

```
# [1] Estimate the MPIN model using the function 'mpin_ml()'

model_mpin <- mpin_ml(dailytrades, verbose = FALSE)
show(model_mpin)
## ---------------------------------
## MPIN estimation completed successfully
## ---------------------------------
## Likelihood factorization : Ersan (2016)
## Estimation Algorithm     : Maximum Likelihood Estimation
## Initial parameter sets   : Ersan (2016), Ersan and Alici (2016)
## Info. layers detected    : using Ersan and Ghachem (2022a)
## ---------------------------------
## 35 initial set(s) are used in the estimation
##
## =========   ==========================
## Variables   Estimates
## =========   ==========================
## alpha       0.216664, 0.050001, 0.483339
## delta       0.230769, 0.666673, 0.034481
## mu          602.86, 986.44, 1506.81
## eps.b       336.91
## eps.s       335.89
## ----
## Likelihood  (643.458)
## mpin(j)     0.082615, 0.031196, 0.460647
## mpin        0.574458
## =========   ==========================

# [2] Estimate the MPIN model using the function 'mpin_ecm()'

model_empin <- mpin_ecm(dailytrades, verbose = FALSE)
show(model_empin)
```

```
## ----------------------------------
## MPIN estimation completed successfully
## ----------------------------------
## Likelihood factorization : Ersan (2016)
## Estimation Algorithm     : Expectation Conditional Maximization
## Initial parameter sets   : Ersan (2016), Ersan and Alici (2016)
## Info. layers detected    : using Ghachem and Ersan (2022) [ECM]
## Selection criterion      : Bayes Information Criterion (BIC)
## ----------------------------------
## 525 initial set(s) are used for all 8 estimations
##
## ==============   ===========================
## Variables        Estimates
## ==============   ===========================
## alpha            0.216667, 0.050000, 0.483333
## delta            0.230769, 0.666667, 0.034483
## mu               602.88, 986.45, 1506.84
## eps.b            336.91
## eps.s            335.89
## ----
## Likelihood       (643.458)
## mpin(j)          0.082619, 0.031196, 0.460648
## mpin             0.574463
## ----
## AIC | BIC | AWE  1308.92, 1331.95, 1409.99
## ==============   ===========================
##
## Table: Summary of 8 MPIN estimations by ECM algorithm
##
##              BIC      AIC      AWE    layers  #Sets  time
## ---------  -------  -------  -------  ------  -----  ----
## model.1    6473.41  6462.94  6508.88      1       5  0.06
## model.2    1633.51  1616.76  1690.27      2      15  0.49
## model.3    1331.95  1308.92  1409.99      3      35  0.98
## model.4**  1331.95  1308.92  1409.99      3      70  1.78
## model.5    1331.95  1308.92  1409.99      3     100  2.55
## model.6    1331.95  1308.92  1409.99      3     100  2.62
## model.7    1342.58  1313.26  1441.9       4     100  3.31
## model.8    1342.58  1313.26  1441.9       4     100  2.83

model_empin@mpinJ
##   layer.1    layer.2    layer.3
## 0.08261897 0.03119604 0.46064817

model_empin@parameters\$alpha
##  layer.1   layer.2   layer.3
## 0.2166667 0.0500000 0.4833333
```

3.3 AdjPIN model functions

The factorization of the likelihood function of the AdjPIN model can be specified using the function `fact_adjpin()`. Three functions are provided to generate initial parameter sets for the estimation of the AdjPIN model. First, `initials_adjpin()` implements the algorithm suggested in Ersan and Ghachem (2022b). Second, `initials_adjpin_rnd()` randomly generates initial parameter sets as follows: The buy rate parameters $\{\varepsilon_b, \mu_b, \Delta_b\}$ are randomly generated from the interval (minB,maxB), where minB (maxB) is the smallest (largest) value of buys in the dataset, under the condition that $\varepsilon_b + \mu_b + \Delta_b < $ maxB. Analogously, the sell rate parameters $\{\varepsilon_s, \mu_s, \Delta_s\}$ are randomly generated from the interval (minS,maxS), where minS (maxS) is the smallest(largest) value of sells in the dataset, under the condition that $\varepsilon_s + \mu_s + \Delta_s < $ maxS. Third, `initials_adjpin_cl()` generates initial parameter sets using an extension of the algorithm derived in Cheng and Lai (2021). In their paper, the authors assume that the probability of liquidity shock is the same in no-information, and information days, i.e., $\theta = \theta'$, and use a procedure similar to that of Yan and Zhang (2012) to generate 64 initial parameter sets. The function implements an extension of their algorithm, by relaxing the assumption of equality

of liquidity shock probabilities, and generates thereby 256 initial parameter sets for the unrestricted AdjPIN model.

The estimation of the AdjPIN model is performed using the function adjpin(). The argument method specifies the estimation method used: "ML" for the standard maximum-likelihood estimation, and "ECM" for the ECM algorithm. The standard maximum-likelihood method writes a factorization of the likelihood function and find its maxima using Nelder–Mead method. The expectation-conditional maximization (ECM) algorithm is suggested and detailed in Ghachem and Ersan (2022). The function allows for the estimation of the AdjPIN model (Duarte and Young, 2009), as well as related restricted models. Restricted models are models where pairs of parameters are assumed to be equal. The choice of a restricted model can be specified via the argument restricted. For instance, calling the function adjpin() with the argument restricted = list(mu = TRUE) correspond the estimation of the restricted AdjPIN model where $\mu_b = \mu_s$.

Estimation output

The output of the estimation function adjpin() is an S4 object of class estimate.adjpin. The slots of the estimate.adjpin object are presented in Table S4.

Examples

We estimate unrestricted, and restricted AdjPIN models using a preloaded dataset called dailytrades.

```
# [1] Generate initial parameter sets for the estimation of the AdjPIN model and use it
# to estimate the model using the ECM algorithm (default)

init.sets <- initials_adjpin(dailytrades)
model <- adjpin(data = dailytrades, initialsets = init.sets)
show(model)
## ---------------------------------
## AdjPIN estimation completed successfully
## ---------------------------------
## Likelihood factorization : Ersan and Ghachem (2022b)
## Estimation Algorithm     : Expectation-Conditional Maximization
## Initial parameter sets   : Custom initial sets
## Model Restrictions       : Unrestricted model
## ---------------------------------
## 49 initial set(s) are used in the estimation
## Type object@initialsets to see the initial parameter sets used
##
## AdjPIN model
##
## =========== ==============
## Variables    Estimates
## =========== ==============
## alpha        0.733333
## delta        0.136364
## theta        0.0625
## theta'       0.636364
## ----
## eps.b        337.17
## eps.s        336.19
## mu.b         599.12
## mu.s         870.98
## d.b          912.75
## d.s          0
## ----
## Likelihood   (893.025)
## adjPIN       0.295083
## PSOS         0.27903
## =========== ==============

# [2] Display probability estimates, trading intensity estimates, adjpin, and psos.
```

```
model@parameters[1:4]
##     alpha     delta     theta    thetap
## 0.7333333 0.1363636 0.0625000 0.6363636

model@parameters[5:10]
##     eps.b     eps.s      mu.b      mu.s       d.b       d.s
## 337.161195 334.770146 599.144502 872.396521 912.749207   2.671429

model@adjpin
## [1] 0.2951761

model@psos
## [1] 0.279842


# [3] Estimate a restricted AdjPIN model where the liquidity shock rates are assumed equal on
# the buy and sell side, i.e., d.b = d.s.

model <- adjpin(data = dailytrades, method = "ML", restricted = list(d = TRUE))
```

3.4 Volume-synchronized probability of informed trading - VPIN

The Volume-Synchronized Probability of Informed Trading or VPIN is developed by Easley et al. (2011) and Easley et al. (2012), and refers to the adaptation of the original PIN model to the high frequency environment.

The function vpin()

The package provides the function vpin() that computes VPIN using a dataset of high-frequency transactions containing three variables timestamp, price, volume. The three essential arguments of the function are: (1) timebarsize, the size of timebars in seconds with a default value of 60, (2) buckets, the number of buckets per volume of bucket size (VBS) with a default value of 50, (3) samplength, the sample length or window of buckets to calculate VPIN, with a default value of 50. Following Easley et al. (2011, 2012), the default value for the argument timebarsize is 1 minute (60 seconds). Recall that the unit of the argument timebarsize is in seconds, enabling the user to use shorter time bar sizes as well.

Estimation output

The output of the estimation function vpin() is an S4 object of class estimate.vpin. The slots of the estimate.vpin object are presented in Table S5.

Examples

We use a dataset called hfdata included in the package, which is a simulated dataset containing sample timestamp, price, volume, bid and ask for 100.000 high-frequency transactions. The function automatically selects the first 3 columns of the provided data, thus ignores the last two columns (bid and ask). When the function vpin() is run without arguments, it uses the default parameters: a time bar size of 60 seconds, 50 buckets per daily average volume, and a sample length of 50 buckets.

```
# [1] Estimate the volume-synchronized probability of informed trading (vpin)

model_vpin <- vpin(hfdata)

# [2] Show selected information details about buckets

tail(model_vpin@bucketdata[, c(1,4:7)], 3)
##       bucket       aoi          starttime           endtime      vpin
## 3596    3596 2240.3600 2019-01-11 05:11:33 2019-01-11 05:14:33 0.2512113
## 3597    3597 1386.5201 2019-01-11 05:14:33 2019-01-11 05:17:33 0.2521648
## 3598    3598  655.7131 2019-01-11 05:17:33 2019-01-11 05:21:33 0.2554926
```

```
# [3] Display summary statistics of and plot the daily vpin vector

summary(model_vpin@dailyvpin$dvpin)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.1364  0.1776  0.1952  0.2050  0.2336  0.4041
```

3.5 Data simulation functions

We provide utilities to generate simulated data for the PIN, MPIN and AdjPIN models, via the functions generatedata_mpin() and generatedata_adjpin().

The function generatedata_mpin() generates datasets according to the assumptions of the generalized PIN model of Easley and O'Hara (1992), and Easley et al. (1996) as derived by Ersan (2016). The main arguments of the function are as follows: series, which represents the number of datasets to be generated; days, specifying the number of days in each dataset; layers, denoting the number of information layers to be generated in the data; parameters, defining the parameters $\Theta = (\alpha, \delta, \mu, \varepsilon_b, \varepsilon_s)$ used in data generation; ranges, a list containing the ranges for some or all parameters; and maxlayers, representing the maximum number of layers in the generated datasets. If the user omits the argument parameters, the function checks the ranges of simulation parameters as present in the argument ranges. If the user provides a range for a given parameter, it is used in simulating the parameter value. Otherwise, a default range is used. The function generatedata_mpin() has three additional arguments that control the relationship between the theoretical values of the simulation parameters: eps_ratio, mu_ratio, and confidence. For more information about these arguments, and default parameter ranges, we refer the reader to the package documentation.

The function generatedata_adjpin() generates datasets according to the assumptions of the Adjusted PIN model (Duarte and Young, 2009). The arguments of the function are as follows: series, representing the number of datasets; days, specifying the number of days in each dataset; parameters, defining the parameters $\Theta = (\alpha, \delta, \theta, \theta', \varepsilon_b, \varepsilon_s, \mu_b, \mu_s, \Delta_b, \Delta_s)$ used in data generation; restricted, a list of binary variables specifying whether two analogous model parameters are assumed equal; and ranges, an alternative to parameters, determining the range for each parameter. The argument restricted can be specified as a vector with four elements: (theta,eps,mu,d). Each of the four elements, when set to TRUE, corresponds to a given restriction on the AdjPIN model. For instance, theta = TRUE corresponds to the AdjPIN model where $\theta = \theta'$. If the user omits the argument restricted, then no restrictions are applied, and the simulated data is generated to fit the unrestricted model. If the user omits the argument parameters, the function checks the ranges of the different simulation parameters contained in the argument ranges. If the user provides a range for a given parameter, it is considered in simulating the value of that parameter. Otherwise, a default range is used. For more information about the function, and the default parameter ranges, we refer the reader to the package documentation.

Simulation output

The output of the data generation functions generatedata_*(), where the suffix (*) can be one of ("mpin","adjpin"), depends on the value of the argument series. If series=1, the output is of class dataset; otherwise the output is of class dataseries. The slot @datasets of the latter object contains the simulated data in the form of a list of dataset objects. The slots of the objects dataset, and dataseries are presented in Table S6, and Table S7 respectively.

Examples

We generate several data series using the functions generatedata_mpin() and generatedata_adjpin() by using different values for the arguments. Note that your results might differ from ours as the data is randomly generated.

```
# [1] Generate a series of 100 simulated semi-annually datasets, having 3 layers
# and 125 days each

dataseries <- generatedata_mpin(series = 100, days = 125, layers = 3)

# [2] Generate, in two ways, a single MPIN dataset with one information layer and the
# simulation parameters (alpha ,delta ,mu ,eb, es) = (0.3, 0.7, 8000, 1500, 2000).
```

```
# (1) Using the argument 'parameters'
sdata <- generatedata_mpin(parameters = c(0.3,0.7,8000,1500,2000))

# (2) Using the argument 'ranges'
sdata <- generatedata_mpin(layers = 1,
ranges = list(alpha=0.3, delta=0.7, eps.b=1500, eps.s=1800, mu=8000))

# [3] Generate a series of 500 datasets with 2 layers where each layer has a minimum
# share of 0.1, eps.b is equal to 5000; and mu is between 5000 and 25000.

dataseries <- generatedata_mpin(series = 500, layers = 2,
ranges = list(alpha = c(0.1,1), eps.b = 5000, mu = c(5000, 25000)))

# [4] Generate a collection of 100 datasets, whose data sequences span 60 days, and
# contain 3 layers, and use it to check the accuracy of the MPIN estimation.

collection <- generatedata_mpin(series = 100, layers = 3)
accuracy <- devmpin <- 0
for (i in 1:100) {
    sdata <- collection@datasets[[i]]
    model <- mpin_ml(sdata@data, xtraclusters = 3, verbose=FALSE)
    accuracy <- accuracy + (sdata@layers == model@layers)
    devmpin <- devmpin + abs(sdata@emp.pin - model@mpin)
}
cat("The accuracy of layer detection: ", paste0(accuracy,"%.\n"), sep="")
cat("The average error in MPIN estimates: ", devmpin/100, ".\n", sep="")

## The accuracy of layer detection: 96%.
## The average error in MPIN estimates: 0.00234024.

# [5] Generate a dataset of 60 days for the adjusted PIN model (default settings).

sdata <- generatedata_adjpin()

# [6] Using a dataset of 10 000 000 days, check that the empirical parameters indeed
# converge to the theoretical parameters — in virtue of the weak law of large numbers.

simdata <- generatedata_mpin(days = 10000000, layers = 1)
## ...
## =========== ============== =============== ===============
## Variables   Theoretical.   Empirical.      Aggregates.
## =========== ============== =============== ===============
## alpha       0.750919       0.750961        0.750961
## delta       0.730749       0.730886        0.730886
## mu          215            214.99          214.99
## eps.b       446            446             446
## eps.s       461            460.99          460.99
## ----
## Likelihood  -              (100973934.705) (100973934.705)
## mpin        -              0.151106        0.151106
## =========== ============== =============== ===============
```

3.6 Trade aggregation function

The PIN model and its extensions use daily numbers of buyer-initiated and seller-initiated trades. Thus, the estimation of the probability of informed trading requires two initial tasks. First is the determination of trade initiator in each trade (trade classification), and second is the aggregation of buys and sells on daily basis.[4] The function aggregate_trades() performs both tasks. Among the trade classification algorithms, **PINstimation** implements four algorithms, which are "Tick", "Quote",

---

[4]In case the data already attaches buy, and sell labels to the individual trades, there is no need to use the algorithms. Besides, when the detailed order book reflecting the arrival times of each electronic message is accessible, high-precision Odders-White (2000) chronological method is preferred. For other kinds of data, trade classification algorithms remain in use, despite non-negligible errors (see e.g., Lee and Ready, 1991; Piwowar and Wei, 2006; Aktas and Kryzanowski, 2014).

"LR", and "EMO". Table 1 gives the definition of each of these algorithms, as taken from Aktas and Kryzanowski (2014). "LR" refers to the Lee and Ready (1991) algorithm, and "EMO" refers to the Ellis et al. (2000) algorithm.

The trade classification algorithms are implemented in a single function `aggregate_trades()` that takes four main arguments: (1) `data`, a dataframe with four variables in the following order (`timestamp`, `price`, `bid`, `ask`), (2) `algorithm`, specifying the algorithm used to determine the trade initiator, accepting one of four possible values: ("Tick","Quote","LR","EMO"), (3) `timelag`, representing the time lag in milliseconds used to calculate the lagged mid-quote for the methods "Quote", "EMO", and "LR", with a default value of `0` milliseconds, and (4) `fullreport`, determining whether the day variable is returned. The default value is `FALSE`. The default value for the time lag to be used in the algorithms is set to `0` – chosen mainly for speed considerations. There are studies also suggesting the better performance of 5-seconds time-lag (Lee and Ready, 1991) and 1-second time-lag (Piwowar and Wei, 2006; Aktas and Kryzanowski, 2014). Given today's high-speed financial markets, a much shorter time-lag of, for example, `100` milliseconds can also be considered.

Table 1: Definition of trade classification algorithms

Tick	A trade is classified as a buy (sell) if the price of the trade to be classified is above (below) the closest different price of a previous trade
Quote	Classifies a trade as a buy (sell) if the trade price of the trade to be classified is above (below) the mid-point of the bid and ask spread. Trades executed at the mid –spread are not classified.
LR	Classifies a trade as a buy (sell) if its price is above (below) the mid-spread (quote algorithm), and uses the tick algorithm if the trade price is at the mid-spread.
EMO	Classifies trades at the bid (ask) as sells (buys) and uses the tick algorithm to classify trades within the then prevailing bid-ask spread.

Estimation output

The output of the function `aggregate_trades()` is a dataframe of two (or three) variables. If the argument `fullreport` is omitted, or set to `FALSE`, the output is a dataframe composed of two variables (`b`,`s`). Otherwise, the dataframe consists of 3 variables (`day`,`b`,`s`).

Examples

We use the preloaded dataset `hfdata` to create a raw high-frequency dataset to aggregate.

```
# [1] Create a high-frequency dataset 'xdata'
xdata <- hfdata
xdata[, "volume"] <- NULL

# [2] Aggregate data using the EMO algorithm with 'timelag' of 50 milliseconds.

aggtrades <- aggregate_trades(xdata, algorithm = "EMO", timelag = 50)

# [3] Aggregate all observations using the 'LR' algorithm with timelag set to 1 second

aggtrades <- aggregate_trades(xdata, algorithm = "LR", timelag = 1000)
```

3.7 More on the PINstimation package

Optimization algorithms: The maximum-likelihood estimation relies on the maximization of the factorized likelihood function over a feasible parameter space. For all instances of MLE throughout the package, this constrained maximization is performed using the Nelder-Mead Simplex algorithm (Nelder and Mead, 1965), as implemented in the function `neldermead()` of the package **nloptr** (Johnson, 2022). In contrast, the expectation-conditional maximization (ECM) algorithm does not require multi-dimensional non-linear optimization. Thanks to the use of conditional maximization in the maximization step, the search for the optimal parameters in the maximization step of the complete-data log-likelihood is reduced to the search for the roots of polynomials using the algorithm of Jenkins

and Traub (1972), which can be implemented, for example, via the function polyroot(). In the documentation of the function, it is stated that "numerical stability may be an issue for all but low-degree polynomials." Luckily, the highest degree of maximands (polynomials) for the AdjPIN (MPIN) model estimation via the ECM algorithm is 4 $(J + 1)$, where J — the number of information layers in the MPIN model — often takes a low value, usually less than 5 (Ersan, 2016).

Parallel processing: The search for global maxima of the log-likelihood function, either through standard MLE, or via ECM algorithm, is performed through running the method for several initial parameter sets to obtain, for each dataset, an optimal estimate, then out of these estimates, the one with the highest log-likelihood is selected. Since the search for local optimum for any given initial set is independent of the search for other initial sets, then parallel processing can be used to speed up the execution. Similarly, the trade aggregation — as implemented in the function aggregate_trades() — takes an argument timelag, and if this argument is positive, it assigns for each high-frequency trade a lagged mid-quote computed using bid and ask registered a timelag earlier. The computation of lagged midquote can be independently performed for all trades, and therefore can be parallelized. Consequently, the package supports parallel processing for these two main tasks, in particular when these tasks take considerably long time. This concerns namely: (1) estimation of the MPIN model when the number of initial parameter sets is large, (2) data aggregation of high-frequency data when a time-lag is used. The parallel processing is enabled using the argument is_parallel available for the functions mpin_ml(), mpin_ecm(), and aggregate_trades(). The default value for this argument is TRUE for the data aggregation, and FALSE for the MPIN model estimation. The parallel processing depends on two additional options: (1) the number of cores used by the functions, (2) the threshold of initial parameter sets needed to activate parallel processing for MPIN estimations. By default, the number of CPU cores used in the parallel processing is 2. The option is stored in, and accessed through the R option pinstimation.parallel.cores. As for the MPIN estimation, parallel processing will not be activated unless the number of initial sets exceeds a threshold, by default 100 sets. The option is stored in, and accessed through the R option pinstimation.parallel.threshold. Information on how to change these options are available on the package website or the package vignette "parallel processing". The parallel processing feature in the package relies on the future framework available through the R package **future** (Bengtsson, 2021). The actual mapping of functions via futures is performed through the function future_map() of the package **furrr** (Vaughan and Dancho, 2022).

Empirical time complexity We have performed an empirical investigation into the time complexity of the algorithms associated with the PIN, MPIN, and ADJPIN models, but chose not to report the results. This decision is motivated by theoretical considerations, as these algorithms are designed to be used with small datasets, typically consisting of 60 to 250 observations[5]. For such small datasets, the algorithms typically execute quite efficiently on a fairly average computer. In contrast, the package contains two functions that can be used with larger datasets, namely the data aggregation function aggregate_trades() and the function vpin(). To inspect the empirical time complexity of these functions, we obtain a real dataset containing two millions high-frequency trades (sampledata), run the functions on subsets of increasing size and inspect at what rate the execution time grows with the size. For the function aggregate_trades(), we perform the procedure for both the sequential and parallel processing. For each value of size in the set (100000,200000,. . . ,2000000), we run the following lines of code (1) aggregate_trades(sampledata[1:size,],algorithm = "LR",timelag = 1000) ,(2) aggregate_trades(sampledata[1:size,],algorithm = "LR",timelag = 1000,is_parallel=FALSE), and (3) vpin(sampledata[1:size,]). For each run, we record the pair consisting of the dataset size, and the execution time. Figure S1 displays the behavior of execution time as a function of the number of high-frequency trades in the dataset for the functions aggregate_trades() and vpin() respectively. Figure S1(a) shows clearly that the function aggregate_trades() displays a linear time complexity, both for sequential, and parallel processing. Similarly, Figure S1(b) shows that the function vpin() does also have a linear time complexity.

Convergence of the ECM algorithm: In theory, the ECM algorithm may fail to converge, and if it does, it may do so slowly (large number of iterations), or converge to a local optimum. To avoid long running times due to non-convergence, Ghachem and Ersan (2022) set an upper bound of 100 iterations per initial set, and report that between 93% and 99% of initial sets lead to convergence in fewer than 100 iterations. To avoid local optima, they use limited number of strategic initial sets, and show that the average bias of AdjPIN(PSOS) is as low as of 0.07% (0.101%); while it is roughly 0.01% for MPIN. Raising the bound on iterations and/or the number of initial sets may further enhance convergence and reduce estimation bias, while keeping running times reasonably low thanks to the fast ECM estimation. Users

---

[5]A 60-day dataset corresponds to approximately three months of trading days, and typically captures the quarterly information flow, such as earnings announcements and other periodic disclosures. A 250-day dataset approximates a year of trading days, and captures annual cycles of information flow, including yearly financial disclosures and seasonal market variations. Using datasets with more than 250 daily observations in the PIN model estimation risks (1) overfitting, (2) incorporating outdated or less relevant information, and (3) compromising model accuracy due to the influence of multiple seasonal and cyclical factors.

may adjust these parameters using the arguments `hyperparams` and `xtraclusters` of `mpin_ecm()`, or `hyperparams` and `num_init` of `adjpin(...,method = "ECM")`.

Sample datasets: The functions included in the package accept datasets in two different formats. Therefore, and for the sake of compactness, we have only included two sample datasets. This is justified by the fact that package enables users to easily generate simulated datasets that fit their preferences and needs (e.g. number of days, any feasible combination of model parameters) using the functions `generatedata_mpin()`, and `generatedata_adjpin()`. More information on these functions, and their arguments can be found in the package documentation.

Clustering algorithm: A large number of algorithms implemented in the package, namely those for layer detection (Ersan, 2016; Ersan and Ghachem, 2022a), or for generating initial parameter sets (Gan et al., 2015; Ersan and Alıcı, 2016; Ersan, 2016; Ersan and Ghachem, 2022b), rely on the hierarchical agglomerative clustering (HAC) in one or more of its steps. The function used in the implementation of HAC throughout the package is `hclust()`.

Custom initial parameter sets: The package provides several functions for generating initial parameter sets for the different PIN models, to be fed in the different estimation functions. These latter functions also allow for the use of custom initial parameter sets. This enables researchers to develop, and experiment with eventually more efficient algorithms for generating initial parameter sets. Therefore, an argument `initialsets` is included in the estimation functions of the PIN models (`pin()`, `mpin_ml()`, `mpin_ecm()`, and `adjpin()`) that allows researchers/users to use the estimation method while providing their own initial parameter sets.

4 Applications

In this section, we showcase the different capabilities of the package by describing in sufficient detail two usage examples analyzing real-world datasets. The purpose of these examples is to show that the package can be used to answer typical research questions, and also to serve as a complementary check – our empirical results corroborate well-established findings in the literature, mainly that small stocks have higher informed trading than large stocks, and VPIN values vary around firm-specific announcements.

In the first example, we use different measures of informed trading (implemented in the package) to conduct descriptive and comparative analyses of informed trading activity in large and small stocks. More specifically, we collect and compare the probability of informed trading obtained by estimating the three major models using a sequence of daily buyer-initiated and seller-initiated trades. These models are PIN (Easley and O'Hara, 1992; Easley et al., 1996), MPIN (Ersan, 2016), and AdjPIN (Duarte and Young, 2009). The research strategy consists of three steps. First, we aggregate the high-frequency transaction datasets into datasets of daily trades using the function `aggregate_trades()` using Lee and Ready (1991) algorithm (`algorithm="LR"`) with zero-second time lag (`timelag=0`). Second, we estimate each of the three models with various methods and specifications suggested in the literature. Finally, we compare the estimates of informed trading in large and small cap stocks, and test the well-established hypothesis that small stocks experience larger probability of informed trading (see e.g. Easley et al., 2002; Aslan et al., 2011).

In the second example, we conduct an intraday analysis of informed trading, using the same dataset, but different variations of the volume-synchronized probability of informed trading or VPIN (Easley et al., 2011, 2012). First, we provide summary statistics for the different VPIN estimates. Next, we provide modified versions of the two tables in Easley et al. (2011) showing the distribution of VPIN and absolute post-returns conditional on each other. Additionally, we investigate the distributions of positive and negative returns separately. Finally, we examine whether order-flow toxicity changes around firm-specific announcements for the examined stocks and during the study period.[6]

4.1 Data

Our main dataset is a stock-level intraday dataset, consisting of all trading transactions for 58 Swedish stocks listed in NASDAQ Stockholm, which took place within the last quarter of 2020 (59 days). The data is a collection of reconstructed order books, based on the NASDAQ OMX Historical ITCH files, and obtained from the website of Swedish House of Finance, National Research Data Center. Reconstructed order books contain extensive information about the different order book entries, such as the instrument symbol, date and timestamp in nanoseconds, first and second-best prices and

---

[6]Few studies examine VPIN around announcements. For example, Bjursell et al. (2017) examine VPIN around inventory announcements and price jumps in crude oil and natural gas futures markets. Bugeja et al. (2015) study VPIN around takeover announcements.

associated volume at both bid and ask sides, transaction price and volume. The main motivation behind the selection of 58 stocks in the sample is to conduct comparative analyses of informed trading between large and small stocks. The 29 large cap stocks are selected in a straightforward manner from among the 30 large-cap stocks listed in OMX Stockholm 30 Index (OMXS30). Of these 30 stocks, one stock (ATCO A) is excluded because of data unavailability. As for the 30 small-cap stocks, we consider the stocks listed in NASDAQ OMX Small Cap Sweden GI (NOMXSCSEGI), which are not listed in neither the mid-cap, nor the large-cap indices (OMXSMCGI, OMXSLCGI). At the time of the study, 219 stocks are listed in the Small Cap index, among which, 39 are not listed in neither of the aforementioned indices. We select the first 30 stocks of these 39 stocks, chronologically. Of these 30 stocks, one small stock (EGTX) is excluded as it only has six days with any trading records. In sum, we have 29 large and 29 small stocks with $5,410,411$ associated transactions.

Our second dataset consists of firm-level announcements pertaining to the selected 58 stocks and occurring within the 59 trading days of the first dataset. The announcements' data were manually collected from company news, available on the website of NASDAQ NORDIC and amount to a total of 546 announcements. We apply several filtering steps on the collected raw data before obtaining the final sample of announcements. For instance, we exclude 353 announcements occurring outside of the trading hours, or within the first and last 10 minutes of the trading day. To avoid ambiguity from combined effects of multiple announcements, we exclude all announcements for any stock-day pair having more than one announcement. The final sample consists of 96 announcements, out of which 41 concern large stocks and 55 concern small stocks.

4.2 Example 1 – PIN estimation

We estimate the standard PIN model (Easley et al., 1996), the MPIN model (Ersan, 2016), and the AdjPIN model, (Duarte and Young, 2009) using a dataset of high frequency trades on a sample of 58 stocks (29 large and 29 small stocks) during the last quarter of 2020. We perform a comparative study of the estimates of different specifications for each of these models, and provide evidence for the existence of significant differences in informed trading between small and large stocks. Technically, we estimate the original PIN model using 8 different specifications, MPIN model using 5 specifications, and ADJPIN model and its restricted versions using 7 specifications. We, however, report a selection of these specifications. Unreported specifications are variations of the reported models with different factorizations, initial sets, and/or restrictions on parameters. Table 2 defines the ten specifications we report and provides the corresponding code to implement each of them.

Table 2: Definition, and implementation code for a selection of model specifications

The factorization, and initial sets for MPIN and AdjPIN models are presented in Ersan (2016), and Ersan and Ghachem (2022b) respectively. Estimations using the ECM algorithm are detailed in Ghachem and Ersan (2022).

Models	Name	Code
PIN Models	PIN_EA	`pin_ea(data)`
	EA initial sets (Ersan and Alıcı, 2016) and E factorization (Ersan, 2016)	
	PIN_GWJ	`pin_gwj(data)`
	GWJ initial sets (Gan et al., 2015) and E factorization (Ersan, 2016)	
	PIN_YZ	`pin_yz(data)`
	YZ initial sets (Yan and Zhang, 2012), and E factorization (Ersan, 2016)	
MPIN Models	MPIN.ML_EG	`mpin_ml(data)`
	ML estimation method, Layer detection algorithm in Ersan and Ghachem (2022a).	
	MPIN.ML_E	`mpin_ml(data,detectlayers = "E")`
	ML estimation method, Layer detection algorithm in Ersan (2016).	
	MPIN.ECM	`mpin_ecm(data,hyperparams = list(maxinit=100))`
	ECM algorithm with up to 100 initial sets per model.	
ADJPIN Models	ADJPIN_GE	`adjpin(data,method = "ML")`
	ML estimation method with GE initial sets (Ersan and Ghachem, 2022b)	
	ADJPIN_RND	`adjpin(data,method = "ML",initialsets = "random")`
	ML estimation method with random initial sets.	
	ADJPIN.ECM_GE	`adjpin(data,method = "ECM")`
	ECM algorithm with GE initial sets (Ersan and Ghachem, 2022b).	
	ADJPIN.ECM_RND	`adjpin(data,method = "ECM",initialsets = "random")`
	ECM algorithm with random initial sets.	

Table 3 presents the mean estimates of the probability of informed trading (PIN) as well as five parameters for the 58 examined stocks. In summary, Table 3 suggest that variation of estimates from different specifications of the same model is of limited scope, while the variation of estimates across models might be quite significant. The MPIN model yields the highest PIN estimates, mainly due to higher probability of information events. Interestingly, the PIN and ADJPIN models produce very similar PIN estimates, even though all their model parameters differ significantly from each other. These results are in line with the assumptions of the different models.

Table 3: Mean estimates of PIN and five parameters in PIN, MPIN, and ADJPIN models

Probability terms, PIN, α, and δ are in percentage. The average running time (*Time*) is in seconds.

Models	Name	PIN	α	δ	μ	ε_b	ε_s	Time
PIN Models	PIN_EA	13.316	17.871	29.45	984.833	727.163	709.545	1.344
	PIN_GWJ	13.385	18.352	30.171	960.139	731.366	706.103	0.291
	PIN_YZ	13.316	17.871	29.45	984.833	727.163	709.545	25.098
MPIN Models	MPIN.ML_EG	23.910	58.537	23.081	534.789	581.911	684.098	49.641
	MPIN.ML_E	20.972	47.633	21.701	528.856	619.009	695.322	23.84
	MPIN.ECM	22.461	54.513	24.563	515.325	665.626	689.832	67.179
ADJPIN Models	ADJPIN_GE	12.658	40.836	48.91	642.788	610.051	554.048	12.449
	ADJPIN_RND	13.484	42.805	50.232	661.256	610.924	549.872	12.49
	ADJPIN.ECM_GE	12.282	40.641	47.496	627.301	632.203	564.216	2.589
	ADJPIN.ECM_RND	12.506	41.023	51.562	601.549	636.203	555.151	2.836

Table 4 reports the mean estimates on the probability of informed trading for large and small stocks separately, their difference, and its statistical significance. For all specifications, the mean PIN estimate is significantly larger for small stocks in comparison to large stocks. For instance, the PIN model mean estimate is around 8.7% for large stocks, while it is almost 18% for small stocks. Similarly, MPIN mean model estimates are larger than those of the PIN model, both for large and small stocks, and can reach up to 30% for small stocks. This finding is in line with previous findings in the market microstructure literature that document larger probabilities of informed trading for small stocks (Easley et al., 2002; Aslan et al., 2011; Chen and Zhao, 2012). In the bottom row of Table 4, mean number of layers detected using the different specifications of the MPIN model are reported. The average number of layers for large stocks is consistently higher than that for small stocks. For instance, for the MPIN.ML_EG, mean number of layers detected in the 2020 last-quarter datasets of large stocks is 4.172. This number is significantly higher than its counterpart for small stocks (around 2.9) for the same period, suggesting that large stocks are more likely to witness different types of information events.

Table 4: Mean PIN estimates and number of layers for large and small stocks

***, **, and * represent significance from a one-sided t-test at 1%, 5% and 10% levels, respectively. PIN values and their differences are in percentages.

Models	Name	PIN - Large	PIN - Small	Difference
PIN Models	PIN_EA	8.658	17.975	9.317***
	PIN_GWJ	8.679	18.091	9.412***
	PIN_YZ	8.658	17.975	9.317***
MPIN Models	MPIN.ML_EG	19.931	27.889	7.958***
	MPIN.ML_E	16.749	25.196	8.447***
	MPIN.ECM	14.574	30.348	15.775***
ADJPIN Models	ADJPIN_GE	10.211	15.105	4.894***
	ADJPIN_RND	10.35	16.618	6.269***
	ADJPIN.ECM_GE	9.591	14.973	5.381***
	ADJPIN.ECM_RND	9.637	15.375	5.737***
Layers (MPIN)	MPIN.ML_EG_layer	4.172	2.931	-1.241***
	MPIN.ML_E_layer	2.897	2.207	-0.69***
	MPIN.ECM_layer	4.207	3.724	-0.483

Next, we focus on one selected implementation for each of three models (PIN_EA, MPIN_ML_EG, ADJPIN_GE). Figure 1 shows stock-level PIN and alpha estimates for each of the three selected specifications. Left (right) hand side of each panel reports estimates for 29 large (small) stocks. Figure 1a displays the PIN estimates for each of the PIN, MPIN and ADJPIN models. While PIN and ADJPIN models produce relatively close PIN estimates, MPIN model estimates are consistently higher. In particular, the difference between MPIN, and PIN estimates is positive, and can reach up to 25%. In contrast, the difference between the estimates from ADJPIN and PIN models does not have a stable sign, and tends to fluctuate around 0.

(a) PIN for large and small stocks   **(b)** Alpha for large and small stocks
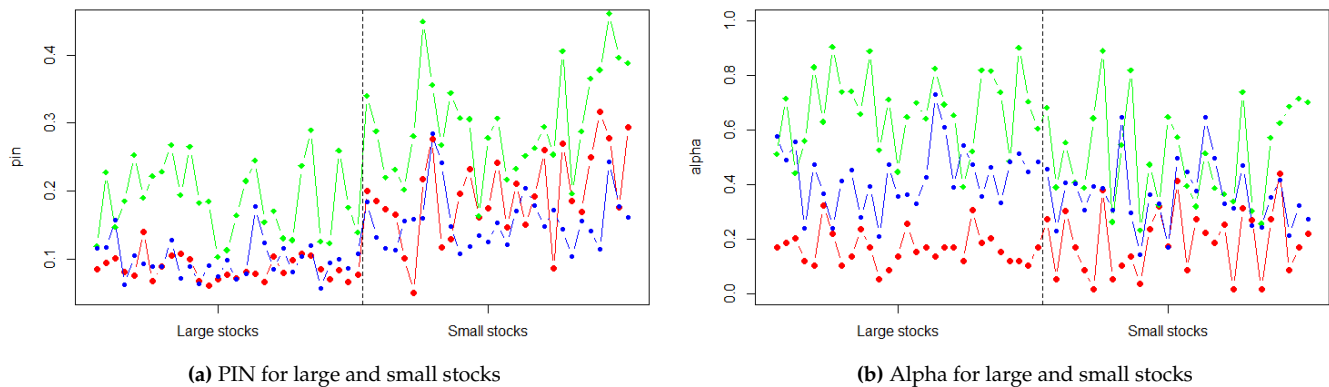
Figure 1: Stock-level model comparisons for PIN and Alpha for the different models: PIN (red), MPIN (green), ADJPIN (blue)

Figure 1a also shows relatively higher estimates in the right side of the panel (small stocks), as well as high stock-based variations, e.g., for the MPIN model PIN estimates range from 10% to around 40% for the examined stocks. Figure 1b replicates Figure 1a for alpha parameter estimates (information event occurrence probability). It shows that PIN model consistently has lower alpha estimates than MPIN and ADJPIN models, but with higher variability, ranging from 2% to 42%. Significant differences in alpha estimates are observed among the three models and across stocks. Therefore, a careful analysis of each model's assumptions is necessary to draw any conclusions.

4.3 Example 2 – VPIN and announcements

Using over 5.4 million trades on 58 Swedish stocks spanning 59 trading days during the last quarter of 2020, we estimate VPIN with three different parameter sets, i.e., 1-50-50, 1-1-5, and 5-1-5 [7].

In each parameter set 'a-b-c', a represents the length of time bars in minutes, b stands for the number of buckets per a day with average trading volume, c is the number of previous buckets used in the calculation of VPIN at any bucket. In line with Easley et al. (2011, 2012), we select the parameter set 1-50-50 as our main setting.

Table 5 presents the summary statistics for VPIN estimates for the three settings, and this for both the whole sample, and for the large and small stocks separately. Mean (median) VPIN with 1-50-50 is 27.6% (25.3%) for the whole sample. Number of VPIN observations is 166, 875, almost equally composed of observations on small and large cap stocks. Mean VPIN is slightly larger for the small stocks (28.1% and 27.2%, respectively).

Under the basic setting, the difference between mean VPIN measures of small and large stocks, while in line with our expectations, it is not as large as previous studies suggest. For instance, Abad and Yagüe (2012) report mean VPIN values of 25% and 53% for the Spanish large and small stocks, respectively. We too obtain positive difference between the mean VPIN values for small and large stocks for all parameter sets. The VPIN value for small stocks is substantially larger than for large stocks (almost twofold) for settings, for which an average trading day contains a single bucket, and five buckets are used in calculating the VPIN (parameter sets 1-1-5 and 5-1-5). The excess informed trading of small stocks is not restricted to average values. For instance, under our basic setting, first and third quartiles of VPIN for the whole sample are around 20% and 34%. This range as well as the standard deviation for small stocks are relatively larger than those of large stocks.

---

[7]The first parameter set 1-50-50 is the main setting used in several studies (see e.g. Easley et al., 2011, 2012; Abad and Yagüe, 2012). The parameter sets 1-1-5, and 5-1-5 are two of the several sets previously used for comparative purposes (see e.g. Abad and Yagüe, 2012).

Table 5: Descriptive statistics for three settings of VPIN - for large, small, and all stocks.

N refers to the number of observations; min and max refer to the minimum, and maximum values respectively. SD corresponds to the standard deviation, while Qx is the xth quantile.

Setting	Sample	N	mean	min	Q25	Q50	Q75	max	SD
1-50-50	Large	84131	27.2	10.7	21.4	25.1	30	92.9	9.2
1-50-50	Small	82744	28.1	0	14.1	25.8	39	100	18.2
1-50-50	All	166875	27.6	0	19.5	25.3	33.6	100	14.4
1-1-5	Large	1595	6.9	1.2	4.7	6.3	8.5	22.8	3.1
1-1-5	Small	1568	13.5	0.7	8.2	12.5	17.1	62.5	7.5
1-1-5	All	3163	10.1	0.7	5.6	8.3	13	62.5	6.6
5-1-5	Large	1595	9	1.2	6.2	8.2	10.6	33.2	4.3
5-1-5	Small	1568	18.3	0.6	11.3	16.2	23.2	89.1	10.3
5-1-5	All	3163	13.6	0.6	7.5	10.8	17.2	89.1	9.1

We turn now to investigate whether the correlation observed between the VPIN distribution and the absolute post returns distribution for the S&P 500 E-mini index, as reported by Easley et al. (2011), can be generalized to (1) individual stocks, (2) another (non-US) market, i.e. NASDAQ Stockholm, (3) more recent data, (4) positive and negative post-returns. To do this, we replicate the two tables (Exhibit 7 and 8) as they appear in Easley et al. (2011) for individual stocks, for absolute post-returns initially, before differentiating between positive and negative post-returns. Table 6 reports, in Panel A, the distribution of the absolute post-returns conditional on VPIN. Each of the 3 rows represents the distribution in percentage for the $0 - 5^{th}, 45^{th} - 50^{th}, 95^{th} - 100^{th}$ quantiles of the VPIN values. Respective quantile values are given in the first column (e.g., 0.062 is the 5th quantile of VPINs in our data).

The results in Table 6 (Panel A) are significantly similar to the results in Easley et al. (2011), both qualitatively, and even quantitatively. For instance, the share of large absolute post-returns is highest in the highest VPIN quantile, and substantially higher than the same share in other quantiles. The share of large absolute post-returns (exceeding 2%) associated with the highest VPIN quantile is 2.16%, while it is below 0.44% for the 45th to 50th VPIN quantiles. The highest levels of VPIN (in the highest quantile) have 4.5 times higher likelihood to be followed by large absolute post-returns than intermediate levels of VPIN (in the median quantile) (2.16% and 0.44%). This ratio is strikingly similar to the one found in the referenced paper (0.22% and 0.05%). However, the likelihood of large absolute post-returns is higher in our study (2.16% vs 0.22%), which is likely due to our use of individual stocks rather than an index. For each of the absolute return intervals larger than 0.5%, the share of VPIN values in the highest quantile is at least twice as large as the ones in lower quantiles. The share of VPINs within the highest quantile (last row of Table 6 - Panel B) is noteworthy: Absolute returns larger than 1% are highly likely to be preceded by a high VPIN value. In our unreported results, for over 40% of intraday periods with absolute returns larger than 2%, the (preceding) VPIN is at its highest quantile.

Table 6: Conditional distributions of VPIN and absolute post-returns

Panel A provides the distribution of absolute post returns (leading VPIN bucket return) conditional on VPIN values, while Panel B provides the distribution of VPIN values conditional on the absolute post returns. For brevity, only the $5^{th}, 50^{th}$ and 100th quantiles are reported in each panel. Numbers are given in percentages.

Panel A: Absolute post-returns conditional on VPIN

	0.25	0.5	0.75	1	1.25	1.5	1.75	2	>2.00
0.062	80.67	10.1	4.68	2.33	0.97	0.53	0.2	0.2	0.32
0.253	80.65	13.11	3.15	1.14	0.71	0.43	0.25	0.12	0.44
1	74.94	10.09	5.03	2.7	1.82	1.5	1.08	0.68	2.16

Panel B: VPIN conditional on absolute post-returns

	0.25	0.5	0.75	1	1.25	1.5	1.75	2	>2.00
0.062	5.19	3.83	5.54	6.21	4.82	4.11	2.62	4.06	2.28
0.253	5.19	4.97	3.74	3.04	3.51	3.36	3.24	2.39	3.12
1	4.82	3.83	5.97	7.2	9.05	11.68	13.87	13.6	15.19

We now turn to investigate whether the distribution patterns for absolute returns hold true when returns are split into positive and negative and analyzed separately. Table S8 summarizes the results across four panels showing only the lowest, median, and highest 5th VPIN quantiles. The distribution patterns of preceding VPINs remain consistent for positive and negative returns, except for the return interval $(-0.5\%, 0.5\%)$. When VPIN values are within the highest quantile and post return is positive (negative), the likelihood of return in the next volume bucket exceeding 2% (-2%) is as high as 3.87% (4.31%). These probabilities are more than seven times that of the median quantile. Note that we excluded zero-return observations before analyzing positive and negative returns separately. This might explain why the findings in Table S8 are more pronounced than those in Table 6.

Finally, we investigate VPIN around firm-specific announcements. Using 96 firm-specific announcements taking place within the last quarter of 2020, and pertaining to the selected stocks, we investigate whether VPIN values change prior to, and following the announcements, and whether the behavior of VPIN around announcements is similar for the large and smalls stocks. Figure 2 plot the mean VPIN for the $(-100, +100)$ volume buckets where 0 refers to the announcement bucket, i.e., the bucket, during which the announcement took place. It represents VPIN values around announcements for the whole sample, and for both large, and small stocks separately. The main finding of our analysis on the whole sample is that, mean VPIN starts to increase shortly prior to announcements, and continues to increase post-announcement, reaches a maximum, before starting to decrease to pre-announcement levels.

As shown in Figure 2a, mean VPIN starts to increase at bucket (-13) from a level 25.7%, monotonically increases for around 50 buckets, reaching a level of 30.81%, before reverting gradually to around its pre-announcement levels. Mean VPIN of small stocks, in Figure 2b, starts rising at bucket (-13) from a level of 25.6%, and keeps increasing until bucket $(+29)$ reaching a level of 32.4% before starting to gradually decrease. It, then, reaches its lowest post-announcement level at bucket $(+81)$, before starting to rise again. As for large stocks in Figure 2b, mean VPIN starts rising at bucket (-7) from a level of 25.2%, and keep increasing until reaching a level of 30.3% at bucket $(+50)$, before gradually decreasing afterwards.

(a) Whole sample of stocks

(b) Large and small stocks
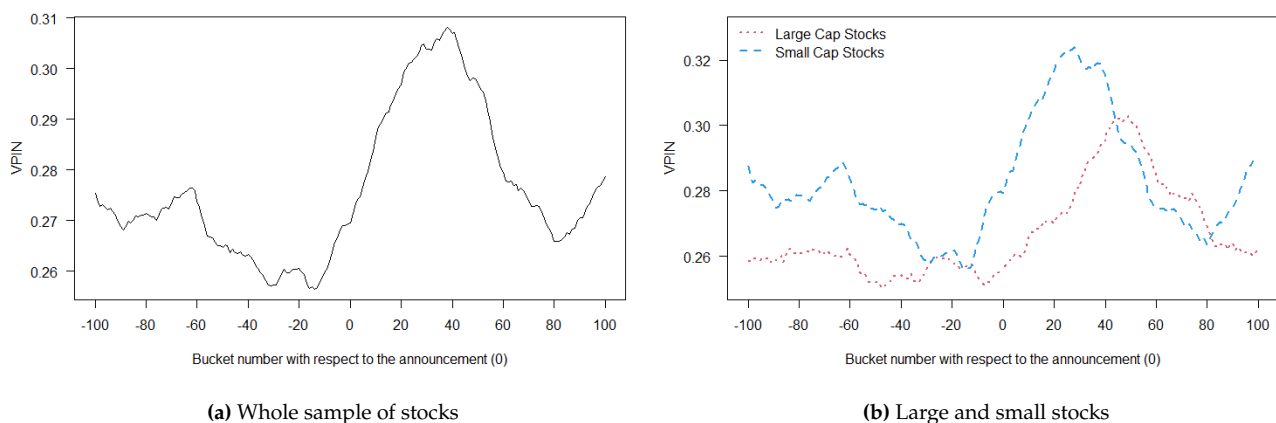
Figure 2: Average VPIN around announcements for small, large, and all stocks

Interestingly, VPIN starts to react relatively earlier for small stocks than for large stocks. Nevertheless, the presence of early warning property of VPIN is evident for both small and large Swedish stocks. This corroborates with the findings of Easley et al. (2011, 2012), where they suggest VPIN as a metric providing an early warning signal for intraday events, such as crashes. Bjursell et al. (2017) document an increase in VPIN prior to news events, and price jumps in the crude oil market. Similarly, Bugeja et al. (2015) examining takeover announcements in the Australian markets, find out that VPIN significantly increases for target firms in the four days prior to the takeover announcements. Our findings suggest the potential of VPIN as an early warning signal might well extend to regular firm-specific events. These VPIN patterns could be further investigated, in light of recent findings on price discovery around announcements in today's financial markets with large HFT prevalence (Beschwitz et al., 2020; Ersan et al., 2021).

5 Conclusion

PINstimation is an attempt to centralize, and implement in a rigorous manner, the main estimation methods suggested in the literature. In addition to efficiency, we aim that **PINstimation** be (1) all-

encompassing, i.e. it includes the main model treating the probability of informed trading and its most relevant extensions, (2) complete, i.e it includes not only the tools required to estimate PIN models, but also algorithms to generate initial parameter sets, tools to simulate datasets, and algorithms to aggregate high-frequency trades into daily trading data, and (3) up-to-date, as the current version of **PINstimation** package is highly up to date including several methods suggested in 2020-2022.

Future work on the package aims at continuous extension of the package with the most up-to-date estimation methods available. For instance, we have recently added function pin_bayes() which implements a Bayesian approach for the estimation of the original PIN model as suggested by Griffin et al. (2021). Even though the **PINstimation** package aims to be all-encompassing, it remains primarily dedicated to the estimation of probability of informed trading (PIN) models. Thus, other informed trading measures suggested in the literature are, and shall remain, beyond the scope of the package. By the introduction of the package, we hope to contribute to widen the user base of PIN models both in academic circles, and among practitioners; as well as improve the validity, and the comparability of scientific findings within the field.

References

D. Abad and J. Yagüe. From pin to vpin: An introduction to order flow toxicity. *Spanish Review of Financial Economics*, 10(2):74–83, 7 2012. ISSN 21731268. doi: 10.1016/j.srfe.2012.10.002. [p162]

H.-J. Ahn, J. Kang, and D. Ryu. Informed trading in the index option market: The case of kospi 200 options. *Journal of Futures Markets*, 28(12):1118–1146, 12 2008. ISSN 1096-9934. doi: 10.1002/FUT. 20369. [p145]

O. U. Aktas and L. Kryzanowski. Trade classification accuracy for the bist. *Journal of International Financial Markets, Institutions and Money*, 33:259–282, 8 2014. ISSN 10424431. doi: 10.1016/j.intfin. 2014.08.003. [p156, 157]

H. Aslan, D. Easley, S. Hvidkjaer, and M. O'Hara. The characteristics of informed trading: Implications for asset pricing. *Journal of Empirical Finance*, 18(5):782–801, 12 2011. ISSN 0927-5398. doi: 10.1016/J. JEMPFIN.2011.08.001. [p159, 161]

H. Bengtsson. A unifying framework for parallel and distributed processing in r using futures. *The R Journal*, 13(2):208–227, 2021. doi: 10.32614/RJ-2021-048. URL https://doi.org/10.32614/RJ-2021-048. [p158]

H. Berkman, P. D. Koch, and P. J. Westerholm. Informed trading through the accounts of children. *Journal of Finance*, 69(1):363–404, 2 2014. ISSN 00221082. doi: 10.1111/jofi.12043. [p145]

V. B. Beschwitz, D. B. Keim, and M. Massa. First to "read" the news: News analytics and algorithmic trading. *Review of Asset Pricing Studies*, 10(1):122–178, 2 2020. ISSN 20459939. doi: 10.1093/RAPSTU/ RAZ007. [p164]

J. Bjursell, G. H. Wang, and H. Zheng. Vpin, jump dynamics and inventory announcements in energy futures markets. *Journal of Futures Markets*, 37(6):542–577, 6 2017. ISSN 10969934. doi: 10.1002/fut.21839. [p159, 164]

D. Bongaerts, D. Rösch, and M. A. Van Dijk. Cross-sectional identification of informed trading. *SSRN Electronic Journal*, 12 2014. ISSN 1556-5068. doi: 10.2139/ssrn.2532128. [p145]

M. Bugeja, V. Patel, and T. Walter. The microstructure of australian takeover announcements. *Australian Journal of Management*, 40(1):161–188, 2 2015. ISSN 13272020. doi: 10.1177/0312896213517247. [p159, 164]

D. Celik and M. Tiniç. *InfoTrad: Calculates the Probability of Informed Trading (PIN)*, 2017. URL https://CRAN.R-project.org/package=InfoTrad. R package version 1.2. [p145]

D. Celik and M. Tiniç. Infotrad: An r package for estimating the probability of informed trading. *R Journal*, 10(1):31–42, 2018. [p145]

S. S. Chang, V. L. Chang, and F. A. Wang. A dynamic intraday measure of the probability of informed trading and firm-specific return variation. *Journal of Empirical Finance*, 29:80–94, 12 2014. ISSN 09275398. doi: 10.1016/j.jempfin.2014.02.003. [p145]

Y. Chen and H. Zhao. Informed trading, information uncertainty, and price momentum. *Journal of Banking and Finance*, 36(7):2095–2109, 7 2012. ISSN 0378-4266. doi: 10.1016/J.JBANKFIN.2012.03.016. [p161]

T. C. Cheng and H. N. Lai. Improvements in estimating the probability of informed trading models. *Quantitative Finance*, 21(5):771–796, 2021. ISSN 14697696. doi: 10.1080/14697688.2020.1800805. [p148, 152]

J. Duarte and L. Young. Why is pin priced? *Journal of Financial Economics*, 91(2):119–138, 2009. ISSN 0304405X. doi: 10.1016/j.jfineco.2007.10.008. [p146, 147, 148, 149, 153, 155, 159, 160]

D. Easley and M. O'Hara. Time and the process of security price adjustment. *Journal of Finance*, 47(2): 577–605, 1992. [p145, 146, 149, 155, 159]

D. Easley, N. M. Kiefer, M. O'Hara, and J. B. Paperman. Liquidity, information, and infrequently traded stocks. *The Journal of Finance*, 51(4):1405, 9 1996. ISSN 00221082. doi: 10.2307/2329399. [p145, 146, 149, 155, 159, 160]

D. Easley, N. M. Kiefer, and M. O'Hara. The information content of the trading process. *Journal of Empirical Finance*, 4(2-3):159–186, 6 1997. ISSN 09275398. doi: 10.1016/S0927-5398(97)00005-4. [p147]

D. Easley, S. Hvidkjaer, and M. O'Hara. Is information risk a determinant of asset returns? *The Journal of Finance*, 57(5):2185–2221, 10 2002. ISSN 1540-6261. doi: 10.1111/1540-6261.00493. [p159, 161]

D. Easley, R. F. Engle, M. O'hara, and L. Wu. Time-varying arrival rates of informed and uninformed trades. *Journal of Financial Econometrics*, 6(2):171–207, 3 2008. ISSN 14798409. doi: 10.1093/jjfinec/ nbn003. [p148, 149]

D. Easley, S. Hvidkjaer, and M. O'Hara. Factoring information into returns. *Journal of Financial and Quantitative Analysis*, 45(2):293–309, 4 2010. ISSN 00221090. doi: 10.1017/S0022109010000074. [p149]

D. Easley, M. M. De Prado, and M. O'Hara. The microstructure of the "flash crash": Flow toxicity, liquidity crashes, and the probability of informed trading. *Journal of Portfolio Management*, 37(2): 118–128, 12 2011. ISSN 00954918. doi: 10.3905/jpm.2011.37.2.118. [p146, 149, 154, 159, 162, 163, 164]

D. Easley, M. M. López De Prado, and M. O'Hara. Flow toxicity and liquidity in a high-frequency world. *Review of Financial Studies*, 25(5):1457–1493, 5 2012. ISSN 08939454. doi: 10.1093/rfs/hhs053. [p146, 149, 154, 159, 162, 164]

K. Ellis, R. Michaely, and M. O'Hara. The accuracy of trade classification rules: Evidence from nasdaq. *The Journal of Financial and Quantitative Analysis*, 35(4):529, 12 2000. ISSN 00221090. doi: 10.2307/2676254. [p157]

O. Ersan. Multilayer probability of informed trading. *SSRN Electronic Journal*, 11 2016. ISSN 1556-5068. doi: 10.2139/ssrn.2874420. [p146, 147, 148, 149, 150, 151, 155, 158, 159, 160]

O. Ersan and A. Alıcı. An unbiased computation methodology for estimating the probability of informed trading (pin). *Journal of International Financial Markets, Institutions and Money*, 43:74–94, 2016. ISSN 10424431. doi: 10.1016/j.intfin.2016.04.001. [p146, 148, 149, 151, 159, 160]

O. Ersan and M. Ghachem. Identifying information types in probability of informed trading (pin) models: An improved algorithm. *SSRN Electronic Journal*, 2022a. [p147, 151, 159, 160]

O. Ersan and M. Ghachem. A methodological approach to the computational problems in the estimation of adjusted pin model. *SSRN Electronic Journal*, 2022b. [p148, 152, 159, 160]

O. Ersan, S. A. Simsir, K. D. Simsek, and A. Hasan. The speed of stock price adjustment to corporate announcements: Insights from turkey. *Emerging Markets Review*, 47:100778, 6 2021. ISSN 18736173. doi: 10.1016/j.ememar.2020.100778. [p164]

Q. Gan, W. C. Wei, and D. Johnstone. A faster estimation method for the probability of informed trading using hierarchical agglomerative clustering. *Quantitative Finance*, 15(11):1805–1821, 2015. ISSN 14697696. doi: 10.1080/14697688.2015.1023336. [p148, 149, 159, 160]

M. Ghachem and O. Ersan. Estimation of the probability of informed trading models via an expectation maximization algorithm. *SSRN Electronic Journal*, 2022. [p146, 148, 149, 151, 153, 158, 160]

J. Griffin, J. Oberoi, and S. D. Oduro. Estimating the probability of informed trading: A bayesian approach. *Journal of Banking and Finance*, 125, 2021. ISSN 0378-4266. doi: https://doi.org/10.1016/j. jbankfin.2021.106045. [p146, 148, 165]

H. Guo and B. Qiu. A better measure of institutional informed trading. *Contemporary Accounting Research*, 33(2):815–850, 6 2016. ISSN 19113846. doi: 10.1111/1911-3846.12160. [p145]

J. Hasbrouck. Measuring the information content of stock trades. *The Journal of Finance*, 46(1):179–207, 3 1991. ISSN 15406261. doi: 10.1111/j.1540-6261.1991.tb03749.x. [p145]

W. l. G. Hsieh and H. R. He. Informed trading, trading strategies and the information content of trading volume: Evidence from the taiwan index options market. *Journal of International Financial Markets, Institutions and Money*, 31(1):187–215, 7 2014. ISSN 10424431. doi: 10.1016/j.intfin.2014.03.012. [p145]

R. D. Huang and H. R. Stoll. Dealer versus auction markets: A paired comparison of execution costs on nasdaq and the nyse. *Journal of Financial Economics*, 41(3):313–357, 7 1996. ISSN 0304405X. doi: 10.1016/0304-405X(95)00867-E. [p145]

R. D. Huang and H. R. Stoll. The components of the bid-ask spread: A general approach. *Review of Financial Studies*, 10(4):995–1034, 10 1997. ISSN 08939454. doi: 10.1093/rfs/10.4.995. [p145]

M. A. Jenkins and J. F. Traub. Algorithm 419: zeros of a complex polynomial [c2]. *Communications of the ACM*, 15(2):97–99, 1972. [p157]

S. G. Johnson. The nlopt nonlinear-optimization package, 2022. [p157]

W. C. Ke, H. Chen, and H. W. W. Lin. A note of techniques that mitigate floating-point errors in pin estimation. *Finance Research Letters*, 31(December 2018):458–464, 12 2019. ISSN 15446123. doi: 10.1016/j.frl.2018.12.017. [p149]

C. M. Lee and M. J. Ready. Inferring trade direction from intraday data. *The Journal of Finance*, 46(2): 733–746, 6 1991. ISSN 15406261. doi: 10.1111/j.1540-6261.1991.tb02683.x. [p156, 157, 159]

E. Lin and C.-F. Lee. Application of poisson mixtures in the estimation of probability of informed trading. In *Handbook of Financial Econometrics and Statistics*, pages 2601–2619. Springer, 2015. [p149]

W. Lin and W. Ke. A computing bias in estimating the probability of informed trading. *Journal of Financial Markets*, 14(4):625–640, 2011. doi: 10.1016/j.finmar.2011.03.001. [p148, 149]

A. Madhavan, M. Richardson, and M. Roomans. Why do security prices change? a transaction-level analysis of nyse stocks. *Review of Financial Studies*, 10(4):1035–1064, 2 1997. ISSN 08939454. doi: 10.1093/rfs/10.4.1035. [p145]

J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4): 308–313, 1965. [p157]

S. K. Ng, T. Krishnan, and G. J. McLachlan. The em algorithm. In *Handbook of computational statistics*, pages 139–172. Springer, 2012. [p148]

E. R. Odders-White. On the occurrence and consequences of inaccurate trade classification. *Journal of Financial Markets*, 3(3):259–286, 8 2000. ISSN 13864181. doi: 10.1016/S1386-4181(00)00006-9. [p156]

M. Piwowar and L. Wei. The sensitivity of effective spread estimates to trade–quote matching algorithms. *Electronic Markets*, 16(2):112–129, 5 2006. doi: 10.1080/10196780600643803. [p156, 157]

A. Recktenwald. *pinbasic: Fast and Stable Estimation of the Probability of Informed Trading (PIN)*, 2018. URL https://CRAN.R-project.org/package=pinbasic. R package version 1.2.2. [p145]

A. Recktenwald. Advanced methods for estimating the probability of informed trading. *Saarländische Universitäts-und Landesbibliothek*, 2019. doi: http://dx.doi.org/10.22028/D291-31254. [p145]

D. Vaughan and M. Dancho. *furrr: Apply Mapping Functions in Parallel using Futures*, 2022. https://github.com/DavisVaughan/furrr, https://furrr.futureverse.org/. [p158]

Y. Yan and S. Zhang. An improved estimation method and empirical properties of the probability of informed trading. *Journal of Banking and Finance*, 36(2):454–467, 2 2012. ISSN 03784266. doi: 10.1016/j.jbankfin.2011.08.003. [p146, 148, 149, 152, 160]

X. Yin and J. Zhao. A hidden markov model approach to information-based trading: Theory and applications. *Journal of Applied Econometrics*, 30(7):1210–1234, 11 2015. ISSN 10991255. doi: 10.1002/jae.2412. [p145]

Montasser Ghachem
Department of Economics, Stockholm University
Stockholm, 106 91, Sweden
Sweden

(0000-0001-6991-3316)
montassar.ghachem@su.se

Oguz Ersan
International Trade and Finance Department, Kadir Has University
Istanbul, 34083
Turkey
(0000-0003-3135-5317)
oguzersan@khas.edu.tr

EviewsR: An R Package for Dynamic and Reproducible Research Using EViews, R, R Markdown and Quarto

by Sagiru Mati, Irfan Civcir, and S. I. Abba

Abstract EViews is a software designed for conducting econometric data analysis. There exists a one-way communication between EViews and R, as the former can run the code of the latter, but the reverse is not the case. We describe EviewsR, an R package which allows users of R, R Markdown and Quarto to execute EViews code. In essence, EviewsR does not only provide functions for base R, but also adds EViews to the existing knitr's knit-engines. We also show how EViews equation, graph, series, and table objects can be imported and customised dynamically and reproducibly in R, R Markdown and Quarto document. Therefore, EviewsR seeks to improve the accuracy, transparency and reproducibility of research conducted with EViews and R.

1 Introduction

EViews, which stands for Econometric Views, is software designed for econometric analysis. EViews provides powerful statistical, time series, forecasting, and modelling tools through an innovative, easy-to-use object-oriented interface (Startz 2019). The software can be operated via its built-in menu, code or scripts saved in a file with the `.prg` extension. Although EViews is programmable, it lacks some routines to estimate new econometric techniques. For this reason, the users of EViews have to resort to using R for routines that are available in R but not yet implemented in EViews. This process involves having to switch between R and EViews, which hinders reproducibility and can lead to errors. EViews can open and run R code, but the converse is not true. An R package **hexView** (Murrell 2019) has been created to facilitate importing EViews workfile data to R. The package is limited as it cannot run EViews code from within the R environment. Another R package **gets** (Pretis, Reade, and Sucarrat 2018) allows exporting R object to EViews program via the package's function `eviews(object, file=NULL, print=TRUE, return=FALSE)`. Still, **gets** package does not allow embedding EViews code in R Markdown and Quarto. Xie (2019) had created the R package **knitr**, which allows communication between R and other statistical applications such as Stata and Octave, or programming languages such as Julia and Python. The statistical application and programming languages supported by the **knitr** package are called knit-engines, and EViews is not one of them. Users can view the existing **knitr** languages by running the following code in R console: `names(knitr::knit_engines$get())`. Another major challenge is that EViews does not support markdown syntax. To address these problems, we created an R package **EviewsR** (Mati 2019b) which integrates EViews and R. Basically, **EviewsR** provides base R functions and additional **knitr**'s knit-engine for EViews.

The **EviewsR** package is designed to be useful for all users of EViews, R, R Markdown and Quarto. The package allows EViews users to comfortably work in R, R Markdown or Quarto and have access to the ecosystems of all the applications. For example, they can make their document dynamic. On the other hand, the users of R, R Markdown and Quarto can easily benefit from the unique EViews's statistical or econometric routines and appealing graphics. For example, they can easily plot a line graph from an undated R dataframe (`dataFrame`) by `EviewsR::eviews_graph(dataFrame,start_date=1990)`, which may need several lines of code in base R. EViews has some advantages over R: 1. code stability. EViews has standard syntax and documentation and offers backward compatibility of code so that code does not break; 2. it also generates corresponding code for each menu action; 3. better support for econometric models; 4. easier to customise graphs. EViews graphs can be edited using the graphical user interface (GUI) and get the corresponding code generated by EViews; 5. stronger support for timeseries and panel data. Thus, teachers of Econometrics can spend more time on teaching Econometrics than on teaching software. EviewsR package helps EViews users to use R Markdown or Quarto without having to learn R functions for estimation and graphing.

Research is reproducible if its scientific computations can be replicated by an independent researcher (Stodden, Leisch, and Peng 2014). On the other hand, interactive reports entail the ability of a computational output to reactively change with the changes in input(s). For example, the output of an input 2+2 is 4. This output is expected to change automatically to 8 when the input changes to 3+5. Modification of figures, tables, bibliography, captions and other objects becomes very easy in dynamic documents. Xie (2014) provides a detailed explanation and implementation of dynamic documents with R.

Sandve (2013) discuss ten simple rules that will ensure reproducibility of computational research.

Some of these rules include avoiding manual data manipulation steps, use of version control and providing public access to scripts and results. Christensen and Miguel (2018) examines the transparency, reproducibility, and credibility of Economics research, revealing evidence of result non-replicability within the field. The interest in reproducibility of research has traversed various fields of STEM and social sciences (see for example Franco, Malhotra, and Simonovits 2014; Simmons, Nelson, and Simonsohn 2011; Gerber et al. 2014; Harvey, Liu, and Zhu 2015; Ioannidis 2005). The aforementioned studies emphasise on the need for guidelines and solid criteria to ensure reproducibility of research. Therefore this article can help ensure replicability and reproducibility of research in the fields that employ EViews and R for their computations.

We categorise the reproducibility of research into three: 1. sharing the data and providing an easy guide on how to implement the computations 2. sharing the data, text and software code in separate files 3. sharing the data, the text and code in a single file. This paper aims to implement the third aspect of reproducibility using EViews, R, R Markdown and Quarto.

We intend to contribute to the current theme of dynamic and reproducible research as follows. We have created an R package **EviewsR**, which does not only integrate EViews and R, but also adds `eviews` as a new knit-engine for the **knitr** package. We also show how to create and modify EViews `equation`, `graph`, `series` and `table` objects dynamically and reproducibly. EViews code can now be embedded in R Markdown and Quarto documents so that both R and EViews users can collaborate on a single document. The package also provides R functions that could be used to 1. graph EViews series objects 2. graph an R dataframe using EViews 3. import data from external sources such as csv, xlsx as a new EViews workfile or into an existing workfile 4. create an EViews workfile from an R dataframe 5. save an EViews workfile or page as a workfile or another file format 6. execute EViews code 7. export an R dataframe as a new EViews workfile or to an existing EViews workfile 8. import EViews table object as `kable` 9. import EViews series objects as a dataframe or xts object 10. import EViews equation data members, `graph`, `series` and `table` objects 11. simulate a random walk process using EViews. We finally show how to use existing EViews workfiles in a dynamic document in order to avoid repeating time-consuming computations.

The rest of the article is structured as follows. We provide an overview of EViews, R, R Markdown and Quarto in Section 2.2. The description of the **EviewsR** package is in Section 2.3. We briefly explain how to use the package along with R, R Markdown and Quarto in Sections 2.4 and 2.6. Section 2.5 is dedicated to the implementation of dynamic document, Section 2.7 to the package implementation, while Section 2.8 covers the summary and conclusion.

2 EViews, R, R Markdown and Quarto

EViews (Econometric Views) is a statistical tool that facilitates both time-series and panel data analyses[1]. It can be operated using GUI, command or a program containing a set of commands.

R, on the other hand, is a free and open-source statistical programming language developed and maintained by R Core Team (2019)[2]. Unlike the EViews, the R software is command-based, implying that every output is generated by executing a command or a set of commands. Thus, reproducing any outputs is as easy as running the code in the R console. Base R's functionality can be extended via custom-made functions and objects, that can be organized into R packages. The R packages are available for free to download at Comprehensive R Archive Network (CRAN)[3].

RStudio is an Integrated Development Environment (IDE) for the R. It simplifies the use of the R as some of the R code can be executed via the GUI drop-down menus in RStudio[4]. In addition to that, RStudio works as an efficient plain text editor; it is easy and straightforward to edit text files with extensions such as `bib`, `tex`, `Rmd`, `Rmarkdown`, `md`, `yaml` and several other extensions.

R Markdown provides an easy way to write a markdown document (Allaire et al. 2020). It is available in RStudio with two alternative extensions: Rmd and Rmarkdown. It facilitates the ability to combine Markdown syntax with the syntax of R and other programming languages supported by the **knitr** package. Users can easily create R Markdown documents in RStudio by clicking `File-> New File-> R Markdown`. R Markdown documents consist of three components: metadata, text and code (Xie 2015). Metadata, also known as YAML metadata or YAML frontmatter, is written in-between a pair of three dashes. It can contain the author name, output format, title and so on (see Xie 2014, 2015, 2019)

---

[1]Please visit https://eviews.com for details

[2]Please visit https://www.r-project.org/ for details.

[3]The CRAN's homepage is https://cran.r-project.org/. The R software and its packages can be downloaded from the homepage.

[4]RStudio can be downloaded for free from https://rstudio.com/

```
---
title: 'An Example of YAML metadata'
author: "Author Name"
output: bookdown::pdf_document2
---
```

After the above YAML metadata, text and code follow until the end of the R Markdown document. Text syntax is just like plain Markdown syntax, but code has to be placed inside blocks delimited by backticks. Note that we use `bookdown::pdf_document2` as the output because it allows for cross-reference in an R Markdown document.

Quarto is the "next generation version of R Markdown" developed by RStudio team. It is an open-source scientific and technical publishing system built on Pandoc. Unlike R Markdown, Quarto can be used to create dynamic content with Python, R, Julia, and Observable and work with IDEs such as VS code, RStudio, Jupyter and Text Editor[5]. The extension of Quarto file is `.qmd`.

2.1 Code chunks

The ability to embed R code is the major difference between an R Markdown/Quarto document and a Markdown document. R code can easily be embedded in R Markdown and Quarto documents. The R **knitr** package extends this capability to allow users to embed the code of other programming languages such as Python and Go, or other statistical packages such as Stata and Octave. A minimal example of R code chunk looks like the following

```
```{r chunkLabel,eval=TRUE,echo=FALSE}
y=runif(100)
```
```

Or in YAML format:

```
```{r}
#| label: chunkLabel
#| eval: true
#| echo: false
y=runif(100)
```
```

R chunk starts with three back-ticks, followed by curly braces containing the knit-engine, chunk label, chunk options, R code and ends with three back-ticks. In the chunk above, `r` is the engine name (knit-engine), `chunkLabel` is an arbitrary but unique text that gives the chunk a name, `eval=TRUE` and `echo=FALSE` are some examples of chunk options, `y=runif(100)` is a typical R code. If `r` is replaced with `eviews` in the chunk above, we will refer to the chunk as an EViews code chunk, not an R code chunk. We will continue to use the YAML format since both R Markdown and Quarto accept it.

We use prompts like "EViews >" and "R >" to represent code written in EViews and R respectively. We use "-" and "+" to signify continuation of EViews and R code respectively. Therefore "+ -" stands for continuation of EViews code written within R function. Code chunk indicates code written in an R Markdown or Quarto document.

3 About EviewsR

EviewsR is an R package to integrate EViews and base R, and also built on top of **knitr** package (Xie 2014, 2015, 2019) to add new knit-engine. It allows users of base R to communicate with EViews via R functions and users of R Markdown to embed EViews code chunks in an R Markdown document.

3.1 How to configure EViews before using EviewsR

To run the package successfully, users need to do one of the following:

1. Do not change anything if the name of the EViews executable is one of the following: `EViews12_x64`, `EViews12_x86`, `EViews11_x64`, `EViews11_x86`, `EViews10_x64`, `EViews10_x86`, `EViews9_x64`, `EViews9_x86`. The package will find the executable automatically.

---

[5]Please visit https://quarto.org/ for details

Figure 1: Setting up EviewsR, EViews, R, R Markdown and Quarto

2. Rename the EViews executable to `eviews` or one of the names above.

3. Alternatively, you can use `set_eviews_path()` function to set the path to the EViews executable as follows:

```
R> set_eviews_path("C:/Program Files (x86)/EViews 10/EViews10.exe")
```

Or

```
R> set_eviews_path("C:\\Program Files (x86)\\EViews 10\\EViews10.exe")
```

The backslash symbol (\) is used as a special escape symbol in R strings, so in order to write a literal backslash we need to precede it with another backslash.

3.2 How to use EviewsR

The package **EviewsR** can be used along with base R, R Markdown or Quarto document. Users should start by loading the **EviewsR** package via R console or by creating an R chunk in an R Markdown or Quarto document as shown below:

```
```{r EviewsRPackage,echo=FALSE}
if(!require("EviewsR")) install.packages("EviewsR")
library(EviewsR)
```
```

Figure 1 presents a chart showing the steps and requirements to use **EviewsR** package.

4 EviewsR: R Markdown and Quarto document

As mentioned earlier, the **EviewsR** package adds `eviews` as a knit-engine to **knitr** package. Therefore, it allows users to embed EViews code in an R Markdown or Quarto document. After loading the package, then create an EViews chunk as shown below:

```
```{eviews}
#| label: fig-EviewsR
#| graph: ""
#| eval: true
#| echo: false
```

```
  'This is some comment in EViews program, feel free to write anything

  wfcreate(wf=EviewsR,page=EviewsR) m 2000 2022

  for %y EviewsR1 EviewsR2
  pagecreate(page={%y}) m 2000 2022
  next

  for %y EviewsR EviewsR1 EviewsR2

  pageselect {%y}

  genr y=@cumsum(nrnd)
  genr x=@cumsum(nrnd)

  graph x_graph.line(o=eviews5) x
  graph y_graph.dot(o=bokeh) y

  table EviewsRTable

  for !j=1 to 7
  EviewsRTable(1,{!j})="Header"+" "+@str({!j})
  next

  for !i=1 to 10
  for !j=1 to 7
  EviewsRTable({!i}+1,{!j})=@str({!i})+","+@str({!j})
  next
  next

  next

  wfsave EviewsR_files/EviewsR
```

The above EViews chunk creates an EViews program with the chunk's content, then automatically opens EViews and runs the program, which will create an EViews workfile with pages (EviewsR, EviewsR1, and EviewsR2) each containing random walk series x and y from January, 2000 to December, 2022. The program will also save an EViews workfile named EviewsR.wf1 in the current directory. We believe that elaboration on the chunk header and options is in order. The word eviews tells the chunk to execute using EViews code, fig-EviewsR is the chunk's label, "eval: true" asks the chunk to evaluate the content (code) of the chunk, "echo: true" allows the content of the chunk to appear in the final document output, comment: NULL eliminates the comment prefix in the chunk output. We set 'graph: ""' in order not show any EViews graph objects. Please refer to **knitr**'s documentation for details. Table 1 lists the specific chunk options for **EviewsR** package:

5 EviewsR: Dynamic Document

This section is about working with EViews's equation, graph, series and table objects dynamically in an R Markdown or Quarto document.

5.1 Accessing EViews objects from EViews chunk

The EViews chunk below (label: fig-EviewsR1) contains EViews code which generates a workfile named EviewsR_workfile along with a page EviewsR_page with monthly frequency from 2000 2022. The chunk also creates additional three EViews pages (page1, page2, and EviewsR) and generates three series objects x, y and z. It then runs the ordinary least square (OLS) method with y as the dependent variable and the rest as independent variables on each of the new pages. On each of the new pages, an equation, a table and three graph objects are created. The equation object is defined as OLS, the table object as OLS_TABLE and three graph objects as graph1, graph2 and graph3.

Table 1: The chunk options available for EviewsR package in R Markdown and Quarto documents

Chunk option	Default value	Possible values	Example	Meaning
equation	*	Any valid name pattern for EViews @wlookup function	eq*, ??e, *e*	EViews equation object
graph	*	asis, first, last, asc, desc, or any valid name pattern for EViews @wlookup function	gr*, g??, *t*	EViews graph object
series	*	Any valid name pattern for EViews '@wlookup' function	se*, ?e?, *y*	EViews series object
table	*	Any valid name pattern for EViews @wlookup function	ta*, ?t, *s*	EViews table object
page	*	Any valid name pattern for EViews @wlookup function	page1, page2, page3, page4, page5	EViews page
graph_procs		Any valid EViews graph's procedure	align(2,1,1)	EViews graph's procedure
save_options		Any valid options for EViews save command	t=pdf, -c	EViews options for graph save command
class	df	xts		Class of R's object for imported EViews series objects
save_path		Any valid path	C:/Users/EviewsR	Path to save EViews graphs

The chunk automatically creates a new environment with the chunk's label as the environment's name, then returns all the EViews's equation, series and table objects into R as a dataframe saved within the new environment. Note that the chunk label (fig-EviewsR1) is not a valid R object, so **EviewsR** removes the prefix `fig-` before creating the new environment as `EviewsR1`. The prefix (`fig-`) is added to the chunk label because it is the Quarto's default way to wrap plots contained in a chunk in a figure environment for cross-reference. If you are working with R Markdown or the chunk does not contain EViews graph objects, you do not need to add the prefix.

The contents of the EViews `table` objects are imported into R exactly as they are on the EViews workfile, but the table name is changed to small letters. You can use the following format to get the imported table:

- chunkLabel$pageName_tableName

The `chunkLabel` stands for the chunk label and is the new environment, `pageName` for the EViews page name and `tableName` for the EViews table object. Therefore, `EviewsR1$eviewsr_ols_table` accesses the table `ols_table`, which is imported from `EviewsR` page of chunk `EviewsR1` into R as a dataframe.

````
```{eviews}
#| label: fig-EviewsR1
#| fig.cap: EViews graphs automatically imported by EViews chunk (fig-EviewsR1: default chunk options)
#| out.width: 32%
#| out.height: 15%
#| fig.ncol: 3
#| echo: false
#| eval: true

'This is some comment in EViews program, feel free to write anything

wfcreate(page=EviewsR_page,wf=EviewsR_workfile) m 2000 2022

!n=123

for %y page2 EviewsR  page1
pagecreate(page={%y}) q 2000 2022
pageselect {%y}
rndseed !n
!n=!n+100
````
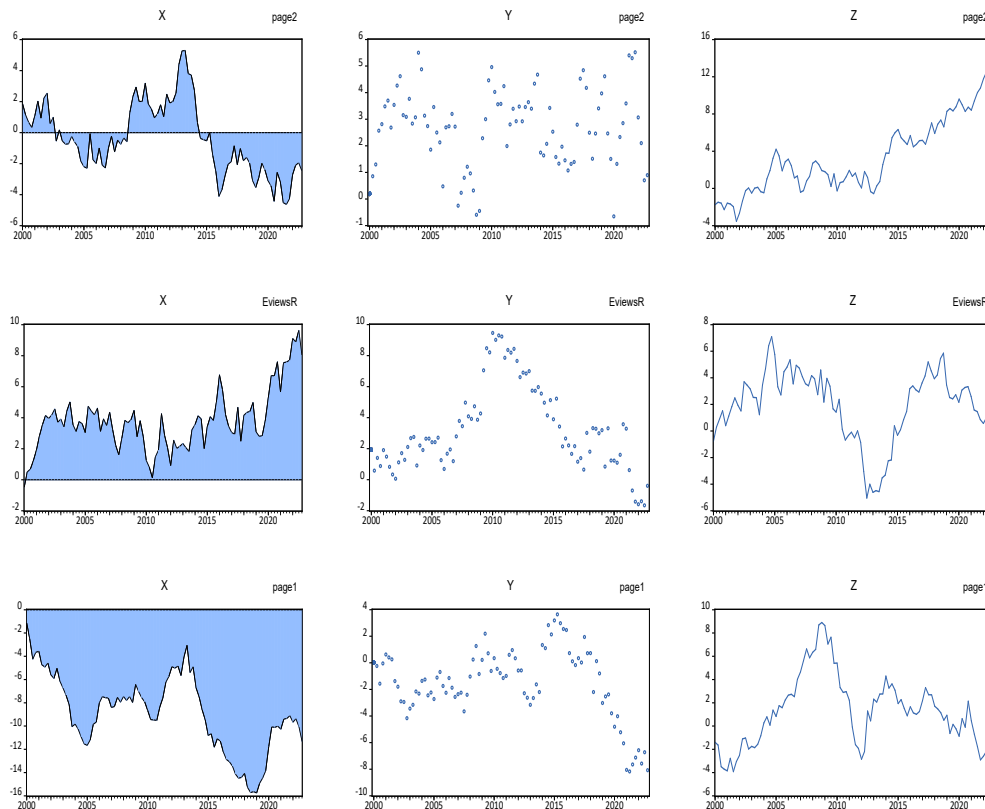
Figure 2: EViews graphs automatically imported by EViews chunk (fig-EviewsR1: default chunk options)

```
genr x=@cumsum(nrnd)
genr y=@cumsum(nrnd)
genr z=@cumsum(nrnd)
equation OLS.ls y c x z
freeze(OLS_TABLE,mode=overwrite) OLS
delete(noerr) GRAPH*
freeze(GRAPH3,mode=overwrite) z.line
graph GRAPH2.dot y
graph GRAPH1.area x
graph3.addtext(ar) %y
graph2.addtext(ar) %y
graph1.addtext(ar) %y
next
```
```

On the other hand, some data members of the EViews `equation` objects are extracted and imported into the new environment as a dataframe. Data member' is an EViews term for statistics, summaries, and information criteria related to EViews `equation` objects. The code below shows the general way to access the imported equation object, where `equationName` is the EViews equation object. The page and equation names are separated by underscore (_). The data members of the equation objects can be accessed via standard list element access syntax:

- chunkLabel$pageName_equationName$dataMember

For example, the $R^2$ value of the OLS equation object on `EviewsR` page is 0.402377, which can be accessed using the inline expression `` `r EviewsR1$eviewsr_ols$r2` ``.

Note that `@coefs`, `@pval`, `@stderrs` and `@tstats` return vectors with length equal to the number of estimated coefficients. The second value of each of these vectors can be accessed with typical square bracket notation, i.e. by appending `[2]`. The rest of the data members return scalar values. The number of data members reported by the EviewsR package is greater than that provided by `lm()`, `glm()` and `summary()` R functions. For example, R users need to use more functions or lines of code to get the

values of Schwarz and Hannan-Quinn Information Criteria of an estimated model. These values are automatically imported into R, R Markdown or Quarto environment by the EviewsR package.

The graph objects are saved on disk to the path defined by the chunk option `fig.path` and can be captioned and sub-captioned via the chunk options `fig.cap` (fig-cap) and `fig.subcap` (fig-subcap) in an R Markdown (Quarto) document. The graph file on the disk are named in the following format:

- `chunkLabel-pageName-graphName`

For example, `eviewsr1-page2-graph1` is the name of `graph1` object of `page2` created in `fig-EviewsR1` chunk.

The series objects of each page can be fetched through

- `chunkLabel$pageName`.

For example, `EviewsR1$page2` provides the dataframe of all the series objects on `page2`. Even if the page is not defined in the created or imported workfile, EViews names both the workfile and page as `untitled` by default. So `EviewsR1$untitled` is the right way to access the series object, if no page name is defined in the `fig-EviewsR1` chunk.

By default, **EviewsR** imports the series objects as a dataframe with date column formatted as `POSIXct`. We can import the series objects as `xts` object by setting chunk option `class="xts"`.

Note that the chunk label is case-sensitive, while the EViews equation, series and table objects are not. They can be in capital letters in the chunk, but they are in lowercase in the R object. This is because R variable names are case-sensitive, while EViews objects are not. For consistency, **EviewsR** is designed to convert EViews object names to lowercase. Therefore, we recommend naming EViews objects with lowercase names.

The graph objects in Figure 2 are imported with the default chunk options. We can use chunk options `page`, `graph` and `graph_procs` to change the behaviour of the graphical outputs. The chunk option page can take `page="*"`, or a space-delimited (or a vector of) string wtih the names of EViews pages. Note that we ask EViews to write the page name on the top right corner of each graph. The chunk option graph can have the values such as `*`, `first`, `last`, `asis`, `asc`, `desc` or a numeric vector.

## 5.2 Importing EViews graph asis

We set these chunk options to get Figure 3.

- `graph: "@asis"`
- `graph_procs: template newspaper`

The option `graph: "@asis"` makes sure that the graph objects are imported in the order they appear in the EViews chunk, that is `graph3` followed by `graph2` and `graph1`. This option is only useful in an R Markdown or Quarto document:

## 5.3 Use of graph_procs chunk option

To plot Figure 4, include the following arguments in the chunk options:

- `graph_procs: [template bokeh, setelem(1) lcolor(green) fillcolor(green)]`
- `graph: "@asis"`
- `page: eviewsr page1`

Note that only the graph objects from pages `EviewsR` and `page1` are included.

## 5.4 Importing only the first graph

To plot only the first graph object from each of the EViews pages, after all the graph objects on each page are arranged in ascending order. Therefore only a combination of `graph1` object from each of the pages is plotted as shown in Figure 5:

- `graph_procs: [template modern,datelabel format("YYYY"),setelem(1) fillcolor(red)]`
- `graph: "@first"`

**Figure 3:** EViews graphs automatically imported by EViews chunk (fig-EviewsR2)



**Figure 4:** EViews graphs automatically imported by EViews chunk (fig-EviewsR3)

**Figure 5:** Importing only the first EViews graphs by EViews chunk (fig-EviewsR4)

## 5.5 Importing only the last graph

The chunk option `graph: "@last"` is the direct opposite of the `graph: " @first"` as it considers the last graph object only.

- graph: "@last"

## 5.6 Importing only graphs selected by index

We modify the chunk options as follows to ensure that only the second, fifth and eighth graph objects across all pages, are included in the output. Therefore only `graph2` object from each of the pages is considered.

- graph: [2,5,8]

## 5.7 Importing graphs in ascending/descending order

We arranges all the graph objects across all the pages in ascending order by setting:

- graph: "@asc"

We can also import the graphs in descending order by:

- graph: "@desc"

## 6 EviewsR: base R functions

We understand that not everyone uses R Markdown or Quarto. Moreover, some EViews computations may take long time to complete, in which case it will be better to work with the existing workfile. For these reasons, we create a number of R functions that can be used to communicate with EViews from R. The functions include:

To work with EViews, a workfile and a workfile page are required, therefore we add `wf` (workfile) and `page` (workfile page) arguments for each function. These functions are explained as follows:

### 6.1 The create_object() function.

The function `create_object()` can be used to create an EViews object in the existing EViews workfile. We can use the EViews workfile (`EviewsR.wf1`) created by the EViews chunk in Section 2.4. The complete EViews syntax for an object command has the form:

```
EViews> action(action_opt) object_name.view_or_proc(options_list) arg_list
```

```
EViews> object_type(options) object_name[=expression]
```

Where `action` can be one of the four EViews commands (`do`, `freeze`, `print`, `show`); `action_opt` modifies the default behaviour of the `action`; `object_name` is any arbitrary character string to represent the name of the EViews object to be acted upon; `view_or_proc` stands for the EViews object's view or procedure to be performed; `options_list` is the option for the `view_or_proc`; `arg_list` stands for the EViews view or procedure arguments.

To create an equation object `eviews_equation` as an `ar(1)` process on `EviewR1` page of `EviewsR.wf1` workfile, we can use the following EViews code:

```
EViews> wfopen EviewsR_files/EviewsR
EViews> pageselect EviewsR1
EViews> equation eviews_equation.ls y ar(1)
EViews> wfsave EviewsR_files/EviewsR
```

The above can be translated into base R function as:

```
R> create_object(wf = "EviewsR_files/EviewsR", page = "EviewsR1",
+    action = "equation", object_name = "eviews_equation", view_or_proc = "ls",
+    arg_list = "y ar(1)")
```

To create a series object `series1` and assign it to the square of y series on `EviewsR2` page of `EviewsR.wf1` workfile:

```
EViews> wfopen EviewsR_files/EviewsR
EViews> pageselect EviewsR2
EViews> series series1=y^2
EViews> wfsave EviewsR_files/EviewsR
```

```
R> create_object(wf = "EviewsR_files/EviewsR", page = "EviewsR2",
+    object_type = "series", object_name = "series1", expression = "y^2")
```

6.2 The eviews_graph() function

EViews graph can be included in an R Markdown or Quarto document by `eviews_graph()` function. This function is a blend of EViews `freeze`, `graph` and `save` commands. The following are the EViews syntaxes:

```
EViews> freeze(options, name) object_name.view_command
EViews> graph graph_name.graph_command(options) arg1 [arg2 arg3 ...]
EViews> graph_name.save(options) [path\]file_name
```

The first and second syntaxes can be merged to produce another syntax which we use to create the `eviews_graph()` function. So the new syntax is:

```
EViews> freeze(options, name) object_name.graph_command(options)
```

Each of the `freeze`, `graph_command` and `save` commands has an `options` keyword. We use `mode` for `freeze` command options, `graph_options` for `graph_command` command options, and `save_options` for `save` command options. Check EViews manual for all the available options for these commands.

The `series` argument can be the names of EViews series objects or an R dataframe. If `series` is a set of EViews series objects, `wf` and `page` need to be specified. If `series` is an R dataframe, `wf` and `page` are optional.

To create figures of series objects x and y saved on `EviewsR` page of the `EviewsR.wf1` workfile:

```
EViews> wfopen EviewsR_files/eviewsr
EViews> pageselect eviewsr
EViews> freeze(eviewsGraph_x,mode=overwrite) x.line
EViews> freeze(eviewsGraph_y,mode=overwrite) y.line
EViews> for %y eviewsGraph_x eviewsGraph_y
EViews> {%y}.axis(l) font(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.axis(r) font(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.axis(b) font(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.axis(t) font(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.legend columns(5) inbox position(BOTCENTER) font(Calibri,12,-b,-i,-u,-s)
EViews> {%y}.options antialias(on)
EViews> {%y}.options size(6,3)
EViews> {%y}.options -background frameaxes(all) framewidth(0.5)
EViews> {%y}.setelem(1) linecolor(@rgb(57,106,177)) linewidth(1.5)
```

Figure 6: EviewsR example figure using LaTeX subfig package

```
EViews> {%y}.setfont legend(Calibri,12,-b,-i,-u,-s) text(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.setfont obs(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.textdefault font(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.save(t=pdf) {%y}
EViews> next
```

Note that the EViews `for loop` in the code above contains the default values of EViews graph procedures provided by **EviewsR** package. It can be found in the `eviews_graph()` function's skeleton defined as `graphicsDefault`. We will avoid repeating the default values and use 'insert default values' instead. These default values can be modified or replaced entirely by using `graph_procs` argument. For example the line colour of the first graph element is blue by default. This can be changed to red by setting `graph_procs="setelem(1) linecolor(red)"`, which overwrites the default "setelem(1) linecolor(@rgb(57,106,177))". We can also overwrite the default behaviour by setting any available EViews templates, for example `graph_grocs="template defaults"`, which resets the graph objects to EViews global graphics defaults.

The EViews code above can be easily written as an R chunk to produce Figure 6 as follows:

```
```{r}
#| label: fig-eviewsGraph
#| fig.cap: EviewsR example figure using \LaTeX{} subfig package
#| fig.show: hold
#| out.width: 45%
#| out.height: 15%
#| eval: TRUE

eviews_graph(series="x y",wf="EviewsR_files/EviewsR",page = "EviewsR",save_options="t=pdf")

```
```

To produce EViews graphs aligned in two columns, as in Figure 7, from an R dataframe named `EviewsRDataFrame`:

- Use R's `write.csv()` function to write the dataframe as a CSV file.

```
R> write.csv(EviewsRDataFrame, "csvFile.csv", row.names = FALSE)
```

- Use EViews to import the `csvFile.csv`, then create and save the graphs on disk:

```
EViews> import csvFile.csv @freq m start_date=1990
EViews> group some_group x y
EViews> freeze(eviewsGraph1_xy,mode=overwrite) some_group.line(m)
                    insert default values
EViews> eviewsGraph1_xy.axis(b) angle(45) font(b)
EViews> eviewsGraph1_xy.save(t=png,d=300) eviewsGraph1_xy
```

- Use **knitr**'s `include_graphics()` function to import the graph into the R Markdown or Quarto document.

The values provided by `graph_procs` argument are appended to the graph's default values. Therefore, `graph_procs` overwrites the existing default value of the EViews graph procedures.

We use the following options to modify the figure:

Figure 7: EViews graph from dataframe (chunk: fig-eviewsGraph1)

- group=TRUE
- graph_options="m"
- graph_procs="axis(b) angle(45) font(b)"

````{r}
#| label: fig-eviewsGraph1
#| fig.cap: "EViews graph from dataframe (chunk: fig-eviewsGraph1)"
#| out.width: 90%
#| out.height: 15%
#| eval: true

eviews_graph(series=EviewsRDataFrame,start_date = "1990",group=TRUE,
graph_options="m",graph_procs="axis(b) angle(45) font(b)")
````
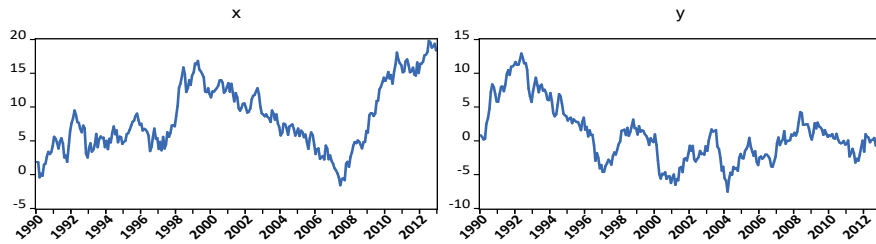
To create a scatterplot along with histogram on each axis border using green colour (#008753 hex code):

```
EViews> wfopen EviewsR_files/eviewsr
EViews> pageselect eviewsr
EViews> group some_group x y
EViews> freeze(eviewsGraph2_xy,mode=overwrite) some_group.scat(ab=histogram) linefit()
                   insert default values
EViews> eviewsGraph2_xy.setelem(1) lcolor(@hex(008753))
EViews> eviewsGraph2_xy.save(t=png,d=300) eviewsGraph2_xy
```

The equivalent R chunk to produce Figure 8:

````{r}
#| label: fig-eviewsGraph2
#| fig.cap: EViews graph from dataframe
#| out.width: 90%
#| out.height: 40%
#| eval: TRUE

eviews_graph(series="x y",wf="EviewsR_files/EviewsR",page="EviewsR",
graph_command="scat(ab=histogram) linefit()",group=TRUE,
graph_procs='setelem(1) lcolor(@hex(008753))')
````

If we want to plot all the series objects contained in `EviewsR2` page of `EviewsR.wf1` workfile that lives in `EviewsR_files/` folder, we can simply use:

```
R> eviews_graph(wf = "EviewsR_files/EviewsR", page = "EviewsR2")
```

Note that we have not specified the `series` argument as `eviews_graph()` function is designed to plot all series objects by default. Similarly, we do not need to provide the page argument if we intend to include the graphs of all the series objects from all the pages of the workfile. We can also use any valid EViews wildcard expressions or pattern, such as `series="x*"`, `series="???x"` and so on.

To plot two or more line graphs with daily frequency on one frame from a dataframe:
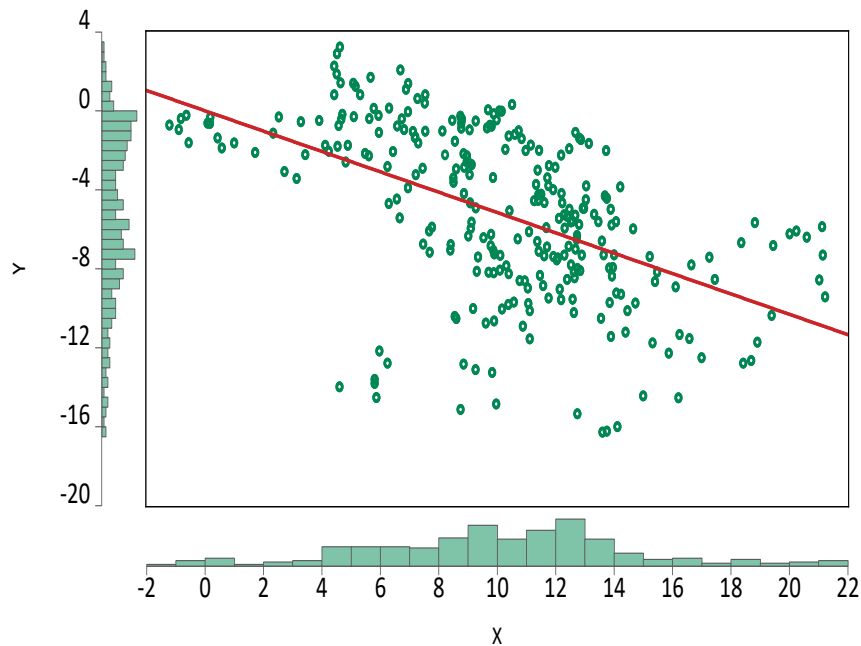
Figure 8: EViews graph from dataframe (chunk: fig-eviewsGraph2)

```
EViews> import csvFile.csv @freq 7 start_date=2000
EViews> group some_group x y
EViews> freeze(eviewsGraph3_xy,mode=overwrite) some_group.line
                    insert default values
EViews> eviewsGraph3_xy.datelabel format("dd Mon, yyyy")  interval(month,1)
EViews> eviewsGraph3_xy.save(t=png,d=300) eviewsGraph3_xy
```

Figure 9 uses the R equivalent of the above EViews code as follows:

```{r}
#| label: fig-eviewsGraph3
#| fig.cap: "EViews graph from dataframe (chunk: fig-eviewsGraph3)"
#| out.width: 90%
#| out.height: 40%
#| eval: TRUE

eviews_graph(series=EviewsRDataFrame,frequency="7",start_date = "2000",group = TRUE,
graph_procs='datelabel format("dd Mon, yyyy")  interval(month,1)')
```

To import EViews line graph of stacked values of the series objects as in Figure 10:

```
EViews> import csvFile.csv @freq a start_date=2010
EViews> group some_group x y
EViews> freeze(eviewsGraph4_xy,mode=overwrite) some_group.line(s)
                    insert default values
EViews> eviewsGraph4_xy.save(t=png,d=300) eviewsGraph4_xy
```

The R's syntax:

```{r}
#| label: fig-eviewsGraph4
#| fig.cap: "EViews graph from dataframe (chunk: fig-eviewsGraph4)"
#| out.width: 90%
```
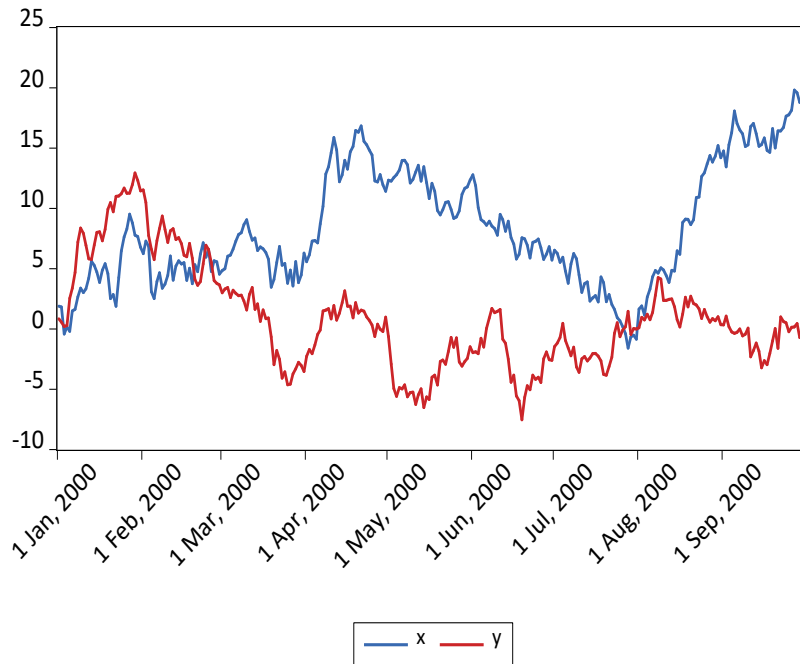
Figure 9: EViews graph from dataframe (chunk: fig-eviewsGraph3)

```
#| out.height: 40%
#| eval: TRUE
#| graph_procs: [template magazine, datelabel format("YYYY")]

eviews_graph(series=EviewsRDataFrame,frequency="m",start_date = "2010",
group = TRUE,graph_options="s")
```

Some journals require contributors to submit figures in black-and-white (see for example Mati, Civcir, and Ozdeser 2023; Mati, Civcir, and Ozdeser 2019). Setting `graph_procs='template monochrome'` or `save_options="t=png,-c"` gives the greyscale image as in Figure 11. The difference is that the former overwrites all the graph modifications to match the monochrome template, while the latter only saves the graphs without any modifications.

6.3 The eviews_import() function

Data can be imported from external sources by `eviews_import()` function. This function is a wrapper for EViews's `import` command. The EViews syntax is:

```
EViews> import([type=], options) source_description import_specification [@smpl smpl_string]
+    [@genr genr_string] [@rename rename_string]
```

To create a new workfile `eviews_import.wf1` from `eviews_import.csv`, which contains columns of x and y variables, using monthly series starting from 1990:

```
EViews> import EviewsR_files/eviews_import.csv @freq m 1990
EViews> wfsave EviewsR_files/eviews_import
```

The R's syntax:

```
R> eviews_import(source_description = "EviewsR_Files/eviews_import.csv",
+    frequency = "m", start_date = "1990")
```

To import x and y, with the former renamed to x2, from `eviews_import.csv` into existing workfile `eviews_import.wf1` within the sample of 1990m10 to 1992m11
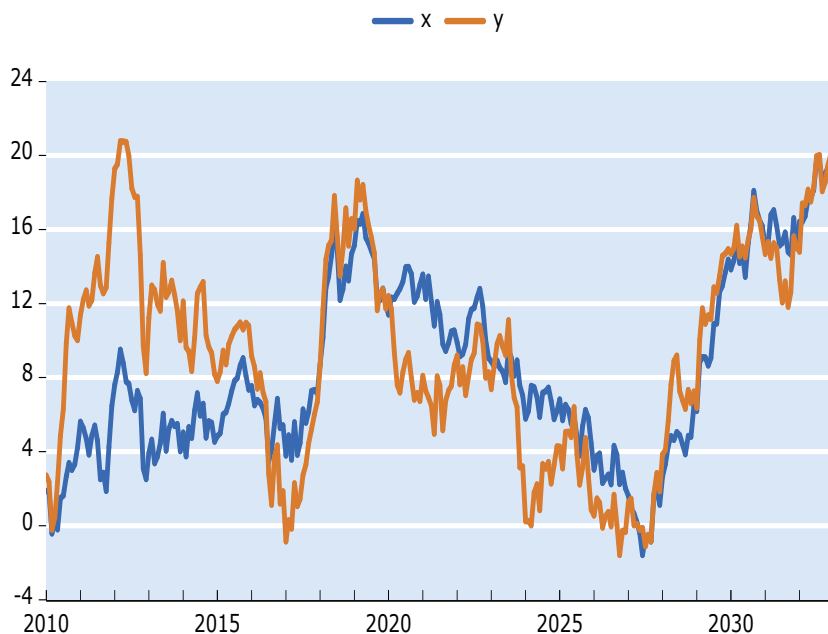
Figure 10: EViews graph from dataframe (chunk: fig-eviewsGraph4)
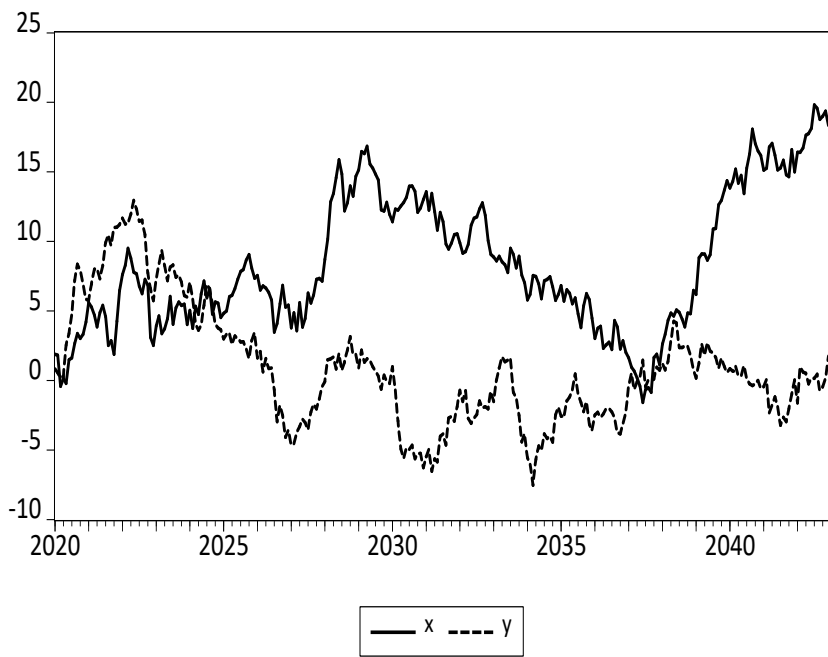
Figure 11: EViews greyscale figure (chunk: fig-eviewsGraph5)

```
EViews> wfopen EviewsR_files/eviews_import.wf1
EViews> import EviewsR_files/eviews_import.csv @smpl 1990m10 1992m11 @rename x x2
EViews> wfsave EviewsR_files/eviews_import
```

```
R> eviews_import(source_description = "EviewsR_files/eviews_import.csv",
+     wf = "EviewsR_files/eviews_import", smpl_string = "1990m10 1992m11",
+     rename_string = "x x2")
```

To import an R dataframe EviewsRDataFrame into eviews_import.wf1 workfile and simultaneously generate another series object z as the sum of x and y:

The dataframe needs to be written as eviews_import.csv file using write.csv() function before executing the following code.

```
EViews> wfopen EviewsR_files/eviews_import.wf1
EViews> import EviewsR_files/eviews_import.csv @smpl 1990m10 1992m11 @genr z=x+y
EViews> wfsave EviewsR_files/eviews_import
```

The R's syntax below makes the process easier:

```
R> eviews_import(source_description = EviewsRDataFrame, wf = "EviewsR_files/eviews_import",
+     genr_string = "z =x+y")
```

To create a new workfile eviews_import1.wf1 from an R dataframe EviewsRDataFrame with a quarterly series starting from September, 2000:

```
R> eviews_import(source_description = EviewsRDataFrame, wf = "EviewsR_Files/eviews_import1",
+     frequency = "m", start_date = "2000m9")
```

It is easier to plot a line graph of an undated dataframe with EviewsR than with the base R. For example, eviews_graph(dataFrame,start_date=1990) will graph a line plot labelled with dates starting from January 1990 on the x-axis. In base R, this requires creating a column for date in the dataframe or converting the dataframe to a timeseries object before plotting.

6.4 The eviews_pagesave() function

An EViews page can be saved in various formats by eviews_pagesave() function.

```
EViews> pagesave(options) source_description table_description [@keep keep_list] [@drop drop_list]
+ [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]
```

To save EviewsR page from EviewsR.wf1 workfile as a workfile named eviews_pagesave.wf1:

```
EViews> wfopen EviewsR_files/EviewsR
EViews> pageselect eviewsr
EViews> pagesave eviews_pagesave
```

The R's syntax:

```
R> eviews_pagesave(wf = "EviewsR_files/EviewsR", page = "EviewsR",
+     source_description = "EviewsR_files/eviews_pagesave")
```

To save the first ten observations of series x only from EviewsR page of EviewsR.wf1 workfile in a CSV file named eviews_pagesave.csv:

```
EViews> wfopen EviewsR_files/EviewsR
EViews> pageselect eviewsr
EViews> pagesave eviews_pagesave.csv @keep x @smpl @first @first+9
```

```
R> eviews_pagesave(wf = "EviewsR_files/EviewsR", page = "EviewsR",
+     source_description = "EviewsR_files/eviews_pagesave.csv",
+     keep_list = "x", smpl_spec = "@first @first+9")
```

Similarly, the values of keepmap_list and dropmap_list can be set as keepmap_list="y*" and dropmap_list="x?" respectively.

6.5 The eviews_wfcreate() function

An EViews workfile can be created using eviews_wfcreate() function in R.

```
EViews> wfcreate(options) frequency[(subperiod_opts)] start_date end_date [num_cross_sections]

EViews> wfcreate(options) frequency[(subperiod_opts)] start_date +num_observations

EViews> wfcreate(options) u num_observations
```

To create an EViews workfile eviews_wfcreate.wf1 along with a page named EviewsR_page in EviewsR_files/ folder:

```
EViews> cd EviewsR_files
EViews> wfcreate(wf=eviews_wfcreate,page=EviewsR_page) m 2000 2022
EViews> wfsave eviews_wfcreate
```

The R's syntax:

```
R> eviews_wfcreate(wf = "eviews_wfcreate", page = "EviewsR_page",
+      frequency = "m", start_date = "2000", end_date = "2022",
+      save_path = "EviewsR_files")
```

6.6 The eviews_wfsave() function

An EViews workfile can be saved in various output formats using eviews_wfsave() function in R.

```
EViews> wfsave(options) [path\]filename

EViews> wfsave(options) source_description [@keep keep_list] [@drop drop_list]
+ [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]

EViews> wfsave(options) source_description table_description [@keep keep_list]
+ [@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]
```

To save all series objects in EviewsR page of EviewsR.wf1 workfile except x in eviews_wfsave.csv file:

```
EViews> wfopen EviewsR_files/eviewsr
EViews> pageselect eviewsr
EViews> wfsave eviews_wfsave.csv @drop x

R> eviews_wfsave(wf = "EviewsR_files/EviewsR", page = "eviewsr",
+      source_description = "EviewsR_files/eviews_wfsave.csv", drop_list = "x")
```

6.7 The exec_commands() function

A set of EViews commands can be executed with the help of exec_commands() function in R.

The EViews chunk in Section 2.4 can be translated using this function as follows:

```
R> exec_commands(c("'This is some comment in EViews program, feel free to write anything",
+
+   'wfcreate(wf=EviewsR,page=EviewsR) m 2000 2022',
+
+   'for %y EviewsR1 EviewsR2',
+   'pagecreate(page={%y}) m 2000 2022',
+   'next',
+
+   'for %y EviewsR EviewsR1 EviewsR2',
+
+   'pageselect {%y}',
```

```
+
+    'genr y=@cumsum(nrnd)',
+    'genr x=@cumsum(nrnd)',
+
+    'graph x_graph.line(o=eviews5) x',
+    'graph y_graph.dot(o=bokeh) y',
+
+    'table EviewsRTable',
+
+    'for !j=1 to 7',
+    'EviewsRTable(1,{!j})="Header"+" "+@str({!j})',
+    'next',
+
+    'for !i=1 to 10',
+    'for !j=1 to 7',
+    'EviewsRTable({!i}+1,{!j})=@str({!i})+","+@str({!j})',
+    'next',
+    'next ',
+
+    'next',
+
+    'wfsave EviewsR_files/EviewsR'))
```

Alternatively, we can assign the EViews commands to an object before calling the function:

```
R> commands=r'('This is some comment in EViews program, feel free to write anything
+
+    wfcreate(wf=EviewsR,page=EviewsR) m 2000 2022
+
+    for %y EviewsR1 EviewsR2
+    pagecreate(page={%y}) m 2000 2022
+    next
+
+    for %y EviewsR EviewsR1 EviewsR2
+
+    pageselect {%y}
+
+    genr y=@cumsum(nrnd)
+    genr x=@cumsum(nrnd)
+
+    graph x_graph.line(o=eviews5) x
+    graph y_graph.dot(o=bokeh) y
+
+    table EviewsRTable
+
+    for !j=1 to 7
+    EviewsRTable(1,{!j})="Header"+" "+@str({!j})
+    next
+
+    for !i=1 to 10
+    for !j=1 to 7
+    EviewsRTable({!i}+1,{!j})=@str({!i})+","+@str({!j})
+    next
+    next
+
+    next
+
+    wfsave EviewsR_files/EviewsR)'
R> exec_commands(commands)
```

To create a workfile `exec_commands1.wf1` using a monthly frequency from November, 2000 to January 2022:

```
EViews> cd EviewsR_files
```

```
EViews> wfcreate(wf=exec_commands1,page=Page) m 2000m11 2022m1
EViews> %wf=@wfname
EViews> wfsave {%wf}
EViews> exit
```

The base R's syntax:

```
R> exec_commands(c("cd EviewsR_files", "wfcreate(wf=exec_commands1,page=Page) m 2000m11 2022m1"))
```

To execute EViews commands on an existing EViews workfile like the one created above:

```
EViews> wfopen EviewsR_files/exec_commands1
EViews> pageselect EviewsR
EViews> genr x=@cumsum(nrnd)
EViews> genr y=@cumsum(nrnd)
EViews> genr z=x+y
EViews> delete(noerr) grap
EViews> graph grap.line x y z
EViews> %wf=@wfname
EViews> wfsave {%wf}
EViews> exit
```

The R's syntax:

```
R> exec_commands(commands = c("genr x=@cumsum(nrnd)", "genr y=@cumsum(nrnd)",
+      "genr z=x+y", "delete(noerr) grap", "graph grap.line x y z"),
+      wf = "EviewsR_files/exec_commands1", page = "page")
```

6.8 The export_dataframe() function

Use `export_dataframe()` function to export dataframe object to EViews as a workfile.

To export the dataframe `EviewsRDataFrame` as a workfile `export_dataframe.wf1` with monthly frequency starting from January 1990:

```
R> export_dataframe(source_description = EviewsRDataFrame, wf = "EviewsR_files/export_dataframe",
+      start_date = "1990", frequency = "m")
```

To export the dataframe as a workfile `export_dataframe1.wf1` with undated frequency. However, if the dataframe contains a column with a regular dated frequency, EViews will automatically detect the date series and create a dated workfile.

```
R> export_dataframe(source_description = EviewsRDataFrame, wf = "EviewsR_files/export_dataframe1")
```

6.9 The import_equation() function

The data members of the EViews equation objects are imported in R as a dataframe.

The data members are accessible via:

- `eviews$pageName_equationName$dataMember` in base R or
- `chunkLabel$pageName_equationName$dataMember` in an R Markdown or Quarto document.

To get the value of Akaike Information Criterion (AIC) of an equation object named `OLS` on page `EviewsR` into base R:

```
R> eviews$eviewsr_ols$aic
```

To obtain the same value from the same equation object in an R Markdown or Quarto document, if the `import_equation()` function is called from a chunk label `importEquation`:

```
R> importEquation$eviewsr_ols$aic
```

Note that the `equation`, `graph`, `series` and `table` objects do not need the chunk label in base R. However, where an R environment is required, we use `eviews` instead of the chunk label.
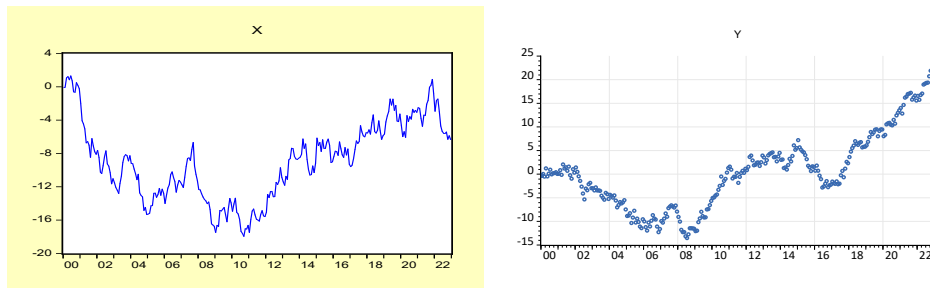
Figure 12: Existing EViews graph imported (chunk: fig-importGraph)

Table 2: EViews table imported as kable

Header 1	Header 2	Header 3	Header 4	Header 5	Header 6	Header 7
1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,1	5,2	5,3	5,4	5,5	5,6	5,7
6,1	6,2	6,3	6,4	6,5	6,6	6,7
7,1	7,2	7,3	7,4	7,5	7,6	7,7
8,1	8,2	8,3	8,4	8,5	8,6	8,7
9,1	9,2	9,3	9,4	9,5	9,6	9,7
10,1	10,2	10,3	10,4	10,5	10,6	10,7

6.10 The import_graph() function

Importing existing graph objects from EViews workfile is easy with `import_graph()` function.

The R chunk below imports EViews graph objects `x_graph` and `y_graph` into an R Markdown or Quarto document as Figure 12. Use `graph="*"` and `page="*"` to import all EViews graph objects across all the workfile pages. The `graph` argument accepts any valid EViews pattern or wildcard expressions, such as `graph="x_*"`, `graph="?_graph"`}. Both the `graph` and `page` arguments can be used to choose specific graph object(s) and workfile page(s) respectively.

To import all graph objects from `EviewsR2` of `EviewsR.wf1` workfile that lives in `EviewsR_files/` folder:

````
```{r}
#| label: fig-importGraph
#| fig.cap: "Existing EViews graph imported (chunk: fig-importGraph)"
#| out.width: 45%
#| out.height: 15%

import_graph(wf="EviewsR_files/EviewsR",page="eviewsr2")
```
````

6.11 The import_kable() function

EViews table objects can be imported as kable object by `import_kable()` function.

To import the entire table object `EviewsRTable` from `EviewsR` page of `EviewsR.wf1` workfile as Table 2:

```
R> import_kable(wf = "EviewsR_files/EviewsR", page = "EviewsR",
+     table = "EviewsRTable", caption = "EViews table imported as kable",
+     format = ifelse(is_html_output(), "html", "latex"), linesep = "")
```

To import certain range of the table object `EviewsRTable` as Table 3:

```
R> import_kable(wf = "EviewsR_files/EviewsR", page = "EviewsR",
```

Table 3: Selected cells of EViews table imported as kable

Header 2	Header 3	Header 4	Header 5	Header 6
1,2	1,3	1,4	1,5	1,6
2,2	2,3	2,4	2,5	2,6
3,2	3,3	3,4	3,5	3,6
4,2	4,3	4,4	4,5	4,6
5,2	5,3	5,4	5,5	5,6

```
+    table = "EviewsRTable", range = "r1c2:r5c6", digits = 3,
+    caption = "Selected cells of EViews table imported as kable",
+    format = ifelse(is_html_output(), "html", "latex"))
```

6.12 The import_series() function

Use `import_series()` function to import EViews series objects into R as a dataframe or `xts` object. The function creates a new environment `eviews`, whose objects can be accessed via `eviews$pageName` in base R or `chunkLabel$pageName` in R Markdown or Quarto.

To import EViews series objects from `EviewsR` page of `EviewsR.wf1` workfile into R as a dataframe:

```
R> import_series(wf = "EviewsR_files/EviewsR", page = "EviewsR")
```

To access the imported series of `EviewsR` page in base R:

```
R> eviews$eviewsr
```

To access the imported series of `EviewsR` page in an R Markdown or Quarto document, if the chunk label is `importSeries`:

```
R> importSeries$eviewsr
```

The `series` argument can take any valid EViews wildcard expressions. The series objects are imported to EViews as a dataframe with date column as `POSIXct` by default. We can import the series as `xts` object by setting `class="xts"`.

```
R> import_series(wf = "EviewsR_files/EviewsR", page = "EviewsR",
+    class = "xts")
```

6.13 The import_table() function

Use `import_table()` function to import EViews table objects into R as a dataframe. The imported table objects can be accessed via `eviews$pageName_tableName`.

```
R> import_table(wf = "", page = "*", table = "*")
```

The `table` argument accepts valid EViews wildcard expressions.

To import all the table objects in `EviewsR.wf1`:

```
R> import_table(wf = "EviewsR_files/EviewsR")
```

To import `eviewsrtable` from the same workfile but from a page named `EviewsR` only:

```
R> import_table(wf = "EviewsR_files/EviewsR", page = "EviewsR",
+    table = "eviewsrtable")
```

To get the table imported from `EviewsR` in an R Markdown or Quarto document, if the chunk label is `importTable1`:

```
R> importTable1$eviewsr_eviewsrtable
```

6.14 The import_workfile() function

To import EViews equation, graph, series and table objects at once from an existing workfile, use the `import_workfile()` function. This function is a combination of `import_equation()`, `import_graph()`, `import_series()` and `import_table()` functions.

To import all the EViews equation, graph, series and table objects from the `EviewsR.wf1` workfile.

```{r}
#| label: fig-importWorkfile
#| fig.cap: Graphs imported by EViews chunk (fig-importWorkfile)
#| out.width: 45%
#| out.height: 15%

import_workfile("EviewsR_files/EviewsR")
```

We can also obtain the imported EViews equation, series and table objects in the same way as `import_equation()`, `import_series()` and `import_table()`.

6.15 The rwalk() function

A set of random walk series can be simulated in R using EViews engine, thanks to `rwalk()` function. The random walk series objects are returned as R dataframe, which can be accessed as `eviews$seriesNames` in base R and as `chunkLabel$seriesNames` in an R Markdown or Quarto document. Removing space from the value(s) of the `series` argument provides the `seriesNames`. Setting argument `class="xts"` returns `xts` object instead of dataframe. This function can be used to replicate examples provided in Econometrics textbooks that use EViews, as random numbers generated in R and EViews differ.

To generate random walk series X Y and Z using EViews and imports them into R as a dataframe `eviews$XYZ`}:

```
R> rwalk(series = "X Y Z", rndseed = 12345, start_date = "1990",
+       frequency = "M", num_observations = 276)
```

To generate random series rw1, rw2 and rw3 each with a drift of 10, on the existing workfile `EviewsR.wf1`:

```
R> rwalk(wf = "EviewsR_files/EviewsR", series = "rw1 rw2 rw3", rndseed = 12345,
+       drift = 10)
```

To obtain the head in base R:

```
R> head(eviews$rw1rw2rw3)
```

To get the head in an R Markdown or Quarto document, if the chunk label is `rwalk1`:

```
R> head(rwalk1$rw1rw2rw3)
```

To plot the random walk dataframe with both **ggplot2** package (Wickham 2016) and `eviews_graph()` function as shown in Figure 13:

```{r}
#| label: fig-rwalk
#| out.width: 45%
#| out.height: 15%
#| dim: [7,4]
#| fig.cap: "Random walk generated by EViews"
#| fig.subcap: ["Graph from ggplot package","Graph from EviewsR package"]

ggplot(rwalk$XYZ,aes(x=date)) +
    geom_line(aes(y=x,color="x"))+ geom_line(aes(y=y,color="y"))+
    geom_line(aes(y=z,color="z"))+
```

(a) Graph from ggplot2 package   **(b)** Graph from EviewsR package
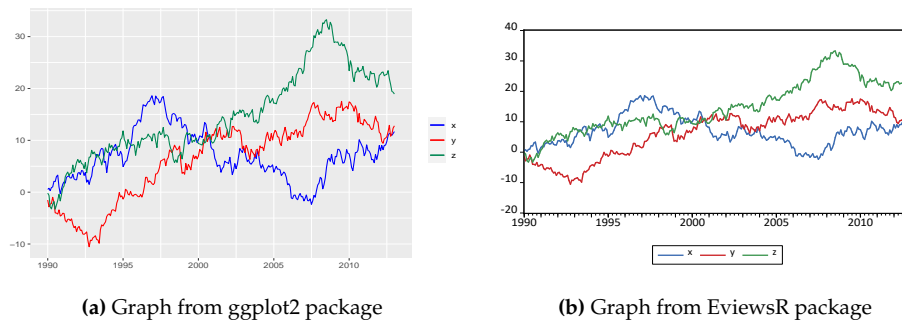
Figure 13: Random walk generated by EViews

```
xlab('')+ylab('')+
labs(colour = "")+
scale_color_manual(values=c(x="blue",y="red",z="#008753"))
```

```
eviews_graph(series=rwalk$XYZ,group = TRUE,graph_procs=c('template reverse','legend position(right)'))
```

```
```
```

6.16 The Set_eviews_path() function

In case of non-standard EViews installation or presence of more than one EViews executable and we do not want to use the latest, we can use this function to set the path to the EViews executable:

To use EViews executable `EViews10`:

```
R> set_eviews_path("C:/Program Files (x86)/EViews 10/EViews10.exe")
```

or

```
R> set_eviews_path("EViews10")
```

7 Package implementation

The purpose of this section to explain how the package is implemented, so that contributions for further development of the package can be easy and straightforward. We show how the `eviews` knit-engine is added and how the base R functions are created.

7.1 Adding eviews as knit-engine

As mentioned earlier, **EviewsR** (Mati 2019b) package adds a knit-engine for Econometric Views (EViews), which is a proprietary econometric software package. The behaviour of EViews is listed below:

1. The file extension of `eviews` code file is `.prg`.
2. The command line execution of the `eviews` code file requires running "eviewsExecutable exec EviewsFileName.prg" or opening the `.prg` file.
3. The `eviews` code file can be opened by quoting the path to the EViews program file. If the EViews code file lives in the current working directory (CWD), writing its full file name (together with the `.prg` extension), with or without the quotes, will run the `eviews` code file.

The first step is to create a character string with the EViews's `.prg` file extension. For example, a character string "MyEviewsFile.prg" can be assigned to an object `fileName` using the code `fileName <-"MyEviewsFile.prg"`. The second step is to create another R object (`eviewsCode`) and use `writeLines()` function to write the EViews code to the file name object created in step 1. The code `writeLines(eviewsCode,fileName)` creates a file in the EViews's CWD and names it

Figure 14: Architecture of EviewsR

`MyEviewsFile.prg`. The third step is to execute the file `MyEviewsFile.prg` using `system2()` function. The code `system2('EViewsExecutable','exec path/to/EViewsFileName.prg')` opens and executes the EViews code file `MyEviewsFile.prg`. The last step is to use `.onLoad()` function along with a function from **knitr** package `knit_engines$set(eviews=eng_eviews)` to set `eviews` as the knit-engine for EViews. Figure 14 presents four easy steps for adding `eviews` to the existing **knitr** engines and creating the R functions.

7.2 Creating base R functions

The base R functions are created in a way similar to adding the knit-engine, but they do not depend on **knitr** package. For the sake of demonstration, we use `eviews_wfsave()` function, which saves EViews workfile in various formats. The EViews syntax for `wfsave` command is as follows:

```
EViews> wfsave(options) [path\]filename
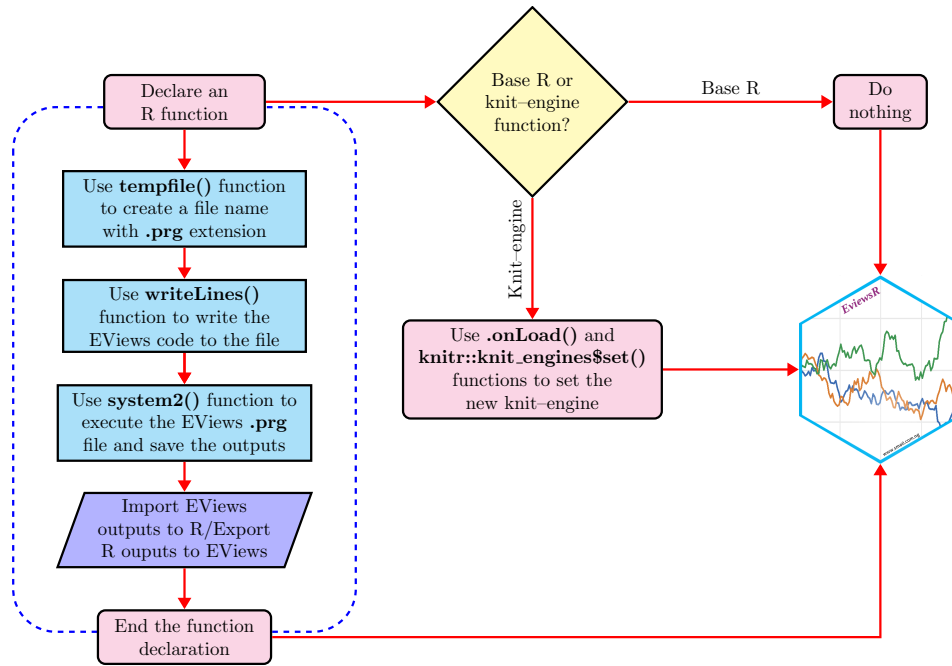

EViews> wfsave(options) source_description [@keep keep_list] [@drop drop_list]
+ [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]


EViews> wfsave(options) source_description table_description [@keep keep_list]
+[@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]
```

The last line of the code provides a more general syntax than the first two. Therefore, we use it to create the equivalent base R function. The syntax contains optional arguments such as `@keep keep_list`, `@drop drop_list`, `@drop drop_list`, `@dropmap dropmap_list`, `@smpl smpl_spec`, which are enclosed in square brackets. The arguments `options`, `source_description` and `table_description` are essential. In addition to this, we need to distinguish constant arguments from variable arguments. The constants include `wfsave`, `@keep`, `@drop`, `@keepmap` and `@dropmap`. On the other hand, the variables, which users can change, include `options`, `source_description`, `table_description`, `keep_list`, `drop_list`, `drop_list`, `dropmap_list`, `smpl_spec`. So, we include these variables as the function arguments for the base R function `eviews_wfsave()`. The suffix `eviews_` indicates that the function is based on the EViews command. For details on `wfsave` command, please visit https://eviews.com/help/helpintro.html#page/content%2Fcommandcmd-pagesave.html.

Since we need a workfile and a workfile page to work with `wfsave` command, the base R function arguments also include `wf` and `page` for workfile and workfile page respectively. The `wf` argument is necessary, but the `page` is not, due to the fact that active EViews page can be used to execute the function.

8 Summary and Conclusion

We have provided an overview of the **EviewsR** package, which allows users to run EViews code in R, R Markdown, and Quarto via R functions or knit-engine. The package provides a convenient and efficient way to integrate EViews and R, two of the most popular software packages for econometrics and statistics. The package offers the ability to run EViews code directly from within R, R Markdown, and Quarto, without having to switch between software packages. It also allows users to access and manipulate EViews objects from within R, making it easy to combine EViews analysis with other R-based workflows. In addition, it can be used to generate high-quality reports and presentations using R Markdown and Quarto, with EViews results seamlessly embedded. The package is a valuable tool for economists, statisticians, and other researchers who need to use both software packages.

The **EviewsR** package is still under development, and there are a number of potential future directions. One possibility is to incorporate graph templates similar to **ggplot2**'s themes.

We encourage users to provide feedback and suggestions for the **EviewsR** package, so that it can continue to be improved and meet the needs of the community.

Similar packages include **URooTab** (Mati 2023) , **DynareR** (Mati 2019a) and **gretlR** (Mati 2019c).

Acknowledgements

APPENDIX

1   EViews examples using base R functions

We provide the base R equivalent operations of examples provided by EViews user manual (https://www.eviews.com/help/helpintro.html).

1.1 EViews graph command

https://eviews.com/help/helpintro.html#page/content%2Fgraphs-Creating_a_Graph.html

Example 1

```
EViews> line income

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "income")
```

Example 2

```
EViews> bar cons

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "income",
+     graph_command = "bar")
```

Example 3

```
EViews> scat x y z

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "x y z",
+     group = TRUE, graph_command = "scat")
```

Example 4

```
EViews> bar(rotate) cons

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "cons", graph_command = "bar",
+     graph_options = "rotate")
```

   Or

```
R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "cons", graph_command = "bar(rotate)")
```

Example 5

```
EViews> scat(ab=boxplot) x y z

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "x y z",
+     group = TRUE, graph_command = "scat", graph_options = "ab=boxplot")
```

   Or

```
R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "x y z",
+     group = TRUE, graph_command = "scat(ab=boxplot)")
```

Example 6

```
EViews> ser2.area(n)

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "ser2", graph_command = "area",
+     graph_options = "n")
```

Or

```
R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "ser2", graph_command = "area(n)")
```

Example 7

```
EViews> grp6.xypair

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "series1 series2",
+     group = TRUE, graph_command = "xypair")
```

Example 8

```
EViews> group g1 x y z
EViews> g1.scat

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "x y z",
+     group = TRUE, graph_command = "scat")
```

Example 9

Since EViews graph and `freeze` commands create a graph object on the current EViews workfile, we have use `exec_commands()` function instead of `eviews_graph()` function. However, if we want to include the graph in an R Markdown or Quarto document, we have to use `import_graph()` function.

```
EViews> freeze grp6.xypair(m)

R> exec_commands(commands = "freeze grp6.xypair(m)", wf = "EviewsR_files/EviewsR")
```

Example 10

```
EViews> freeze(graph1) grp6.line

R> exec_commands(commands = "freeze(graph1) grp6.line", wf = "EviewsR_files/EviewsR")
```

Example 11

```
EViews> graph gr1 ser1 ser2

R> exec_commands(commands = "graph gr1 ser1 ser2", wf = "EviewsR_files/EviewsR")
```

Example 12

```
EViews> graph gr2.line ser1 ser2

R> exec_commands(commands = "graph gr2.line ser1 ser2", wf = "EviewsR_files/EviewsR")
```

Example 13

```
EViews> graph gr3.xyline group3

R> exec_commands(commands = "graph gr3.xyline group3", wf = "EviewsR_files/EviewsR")
```

Example 14

```
EViews> graph g1.xyline(d) unemp gdp inv

R> exec_commands(commands = "graph g1.xyline(d) unemp gdp inv",
+       wf = "EviewsR_files/EviewsR")
```

Example 15

```
EViews> group grp1 sales1 sales2
EViews> graph grsales.bar(s) grp1
EViews> show grsales

R> commands = "
+ group grp1 sales1 sales2
+ graph grsales.bar(s) grp1
+ show grsales
+ "
R>
R> exec_commands(commands = commands, wf = "EviewsR_files/EviewsR")
```

Example 16

```
EViews> graph gr2.merge gr1 grsales

R> exec_commands(commands = "graph gr2.merge gr1 grsales", wf = "EviewsR_files/EviewsR")
```

1.2 EViews import command

https://www.eviews.com/help/helpintro.html#page/content%2Fcommandcmd-import.html

Example 1

```
EViews> import c:\temp\quarterly.xls @freq q 1990

R> eviews_import(source_description = "c:\\temp\\quarterly.xls",
+       frequency = "q", start_date = 1990)
```

Example 2

```
EViews> import(c=s) c:\temp\quarterly.xls range="GDP_SHEET" @freq q 1990 @rename gdp_per_capita gdp

R> eviews_import(source_description = "c:\\temp\\quarterly.xls range=\"GDP_SHEET\"",
+    options = "c=s", frequency = "q", start_date = 1990, rename_string = "gdp_per_capita gdp")
```

Example 3

```
EViews> import(mode=p) c:\temp\annual.txt @freq a 1990 @smpl 1994 1996

R> eviews_import(source_description = "c:\\temp\\annual.txt", options = "mode=p",
+       frequency = "a", start_date = 1990, smpl_string = "1994 1996")
```

Example 4

```
EViews> import(c=max, type=excel) c:\data\stateunemp.xls @id states @destid states

R> eviews_import(source_description = "c:\\data\\stateunemp.xls",
+    type = "excel", options = "c=max", id = "states", destid = "states")
```

Example 5

```
EViews> import c:\data\stategdp.txt colhead=3 delim=comma @id states @date(year) @destid states @date

R> eviews_import(source_description = "c:\\data\\stategdp.txt colhead=3 delim=comma",
+     id = "states @date(year)", destid = "states @date")
```

Example 6

```
EViews> import c:\data\cagdp.xls @id states @date(year) @destid states @date @genr states="CA"

R> eviews_import(source_description = " c:\\data\\cagdp.xls", id = "states @date(year)",
+     destid = "states @date", genr_string = "states=\"CA\"")
```

Example 7

```
EViews> import(resize) sales.dta @smpl @all

R> eviews_import(wf = "EviewsR_files/EviewsR", source_description = "sale.dta")
```

Example 8

```
EViews> import(page=demand) demand.txt @append

R> eviews_import(wf = "EviewsR_files/EviewsR", source_description = "demand.txt",
+     options = "page=demand", append = TRUE)
```

1.3 EViews wfcreate command

https://www.eviews.com/help/helpintro.html#page/content%2Fcommandcmd-wfcreate.html

Example 1

```
EViews> wfcreate(wf=storehours) 30MIN(1-6, 8:00-17:00) 1/3/2000 12/30/2000

R> eviews_wfcreate(wf = "storehours", frequency = "30MIN", subperiod_opts = "1-6, 8:00-17:00",
+     start_date = "1/3/2000", end_date = "12/30/2000")
```

Or

```
R> eviews_wfcreate(wf = "storehours", frequency = "30MIN(1-6, 8:00-17:00)",
+     start_date = "1/3/2000", end_date = "12/30/2000")
```

Example 2

```
EViews> wfcreate(wf=storehours) 30MIN(1-6, 8AM-5PM) 1/3/2000 12/30/2000

R> eviews_wfcreate(wf = "storehours", frequency = "30MIN", subperiod_opts = "1-6, 8AM-5PM",
+     start_date = "1/3/2000", end_date = "12/30/2000")
```

Or

```
R> eviews_wfcreate(wf = "storehours", frequency = "30MIN(1-6, 8AM-5PM)",
+     start_date = "1/3/2000", end_date = "12/30/2000")
```

Example 3

```
EViews> wfcreate(wf=storehours) 30MIN(1-7, 10AM-3PM) 1/3/2000 12/30/2000

R> eviews_wfcreate(wf = "storehours", frequency = "30MIN", subperiod_opts = "1-7, 10AM-3PM",
+     start_date = "1/3/2000", end_date = "12/30/2000")
```

Or

```
R> eviews_wfcreate(wf = "storehours", frequency = "30MIN(1-7, 10AM-3PM)",
+     start_date = "1/3/2000", end_date = "12/30/2000")
```

Example 4

```
EViews> wfcreate(wf=storehours) 30MIN(1-6, 8AM-5PM) 1/3/2000 10AM 12/30/2000 2PM

R> eviews_wfcreate(wf = "storehours", frequency = "30MIN", subperiod_opts = "1-6, 8AM-5PM",
+     start_date = "1/3/2000 10AM", end_date = "12/30/2000 2PM")
```

Or

```
R> eviews_wfcreate(wf = "storehours", frequency = "30MIN(1-6, 8AM-5PM)",
+     start_date = "1/3/2000 10AM", end_date = "12/30/2000 2PM")
```

Example 5

```
EViews> wfcreate w(monday) 2000 2010

R> eviews_wfcreate(frequency = "W", subperiod_opts = "monday", start_date = "2000",
+     end_date = "2010")
```

Or

```
R> eviews_wfcreate(frequency = "W(monday)", start_date = "2000",
+     end_date = "2010")
```

Example 6

```
EViews> wfcreate a(july) 2001 2007

R> eviews_wfcreate(frequency = "a", subperiod_opts = "july", start_date = "2001",
+     end_date = "2007")
```

Or

```
R> eviews_wfcreate(frequency = "a(july)", start_date = "2001", end_date = "2007")
```

Example 7

```
EViews> wfcreate w 2000 2010

R> eviews_wfcreate(frequency = "W", start_date = "2000", end_date = "2010")
```

Example 8

```
EViews> wfcreate(wf=annual, page=myproject) a 1950 2005

R> eviews_wfcreate(wf = "annual", page = "myproject", frequency = "a",
+     start_date = "1950", end_date = "2005")
```

Example 9

```
EViews> wfcreate(wf=unstruct, page=undated) u 1000

R> eviews_wfcreate(wf = "unstruct", page = "undated", frequency = "u",
+     num_observations = 1000)
```

Example 10

```
EViews> wfcreate(wf=griliches_grunfeld, page=annual) a 1935 1954 10

R> eviews_wfcreate(wf = "griliches_grunfeld", page = "annual", frequency = "a",
+     start_date = 1935, end_date = 1954, num_cross_sections = 10)
```

Example 11

```
EViews> wfcreate(wf=fourday) D(1,4) 1/3/2000 12/31/2000

R> eviews_wfcreate(wf = "fourday", frequency = "D", subperiod_opts = "1,4",
+     start_date = "1/3/2000", end_date = "12/31/2000")
```

    Or

```
R> eviews_wfcreate(wf = "fourday", frequency = "D(1,4)", start_date = "1/3/2000",
+     end_date = "12/31/2000")
```

Example 12

```
EViews> wfcreate(wf=fourday) D(1-4) 1/3/2000 12/31/2000

R> eviews_wfcreate(wf = "fourday", frequency = "D", subperiod_opts = "1-4",
+     start_date = "1/3/2000", end_date = "12/31/2000")
```

    Or

```
R> eviews_wfcreate(wf = "fourday", frequency = "D(1-4)", start_date = "1/3/2000",
+     end_date = "12/31/2000")
```

Example 13

```
EViews> wfcreate(wf=captimes) 15SEC(2-4) 1/3/2000 12/30/2000

R> eviews_wfcreate(wf = "captimes", frequency = "15SEC", subperiod_opts = "2-4",
+     start_date = "1/3/2000", end_date = "12/31/2000")
```

    Or

```
R> eviews_wfcreate(wf = "captimes", frequency = "15SEC(2-4)", start_date = "1/3/2000",
+     end_date = "12/31/2000")
```

Example 14

```
EViews> wfcreate m 1995 +30

R> eviews_wfcreate(frequency = "m", start_date = "1995", end_date = "+30")
```

1.4 EViews pagesave command

https://eviews.com/help/helpintro.html#page/content%2Fcommandcmd-pagesave.html

Example 1

```
EViews> pagesave new_wf

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", source_description = "new_wf")
```

Example 2

```
EViews> pagesave "c:\documents and settings\my data\consump"

R> eviews_pagesave(wf="EviewsR_files/EviewsR",
+               source_description = "c:\\documents and settings\\my data\\consump")
```

Example 3

```
EViews> pagesave macro @keep gdp unemp

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", source_description = "macro",
+      keep_list = "gdp unemp")
```

Example 4

```
EViews> pagesave macro @dropmap gdp*

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", source_description = "macro",
+      dropmap_list = "gdp*")
```

Example 5

```
EViews> pagesave(type=excelxml, mode=update) macro.xlsx

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", options = "type=excelxml,mode=update",
+      source_description = "macro.xlsx")
```

Example 6

```
EViews> pagesave(type=excelxml, mode=update) macro.xlsx range="Sheet2!a1" byrow @keep gdp unemp

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", options = "type=excelxml,mode=update",
+      source_description = "macro.xlsx", table_description = "range=\"sheet2!a1\" byrow",
+      keep_list = "gdp unemp")
```

Example 7

```
EViews> pagesave(type=excelxml, mode=update) macro.xlsm range="Sheet2!a1" byrow @keep gdp unemp

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", options = "type=excelxml,mode=update",
+      source_description = "macro.xlsm", table_description = "range=\"sheet2!a1\"",
+      keep_list = "gdp unemp")
```

Example 8

```
EViews> pagesave(type=excelxml, noid) macro.xlsx range="Sheet2!a1"

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", options = "type=excelxml,noid",
+      source_description = "macro.xlsx", table_description = "range=\"sheet2!a1\"")
```

1.5 EViews wfsave command

https://eviews.com/help/helpintro.html#page/content%2Fcommandcmd-wfsave.html

The EViews commands pagesave and wfsave are similar. Therefore, the above examples about pagesave can be adopted by replacing pagesave with wfsave.

2 Notes for EViews users

This section explains some peculiarities of R's syntax. It is important for working with **EviewsR**'s functions.

Each of the following is valid way to write a character string containing double quote ("). All of them assign the same string (range="sheet2!a1" byrow) to table_description.

```
R> table_description='range="sheet2!a1" byrow'
R> table_descriptione="range=\"sheet2!a1\" byrow"
R> table_description=r'(range="sheet2!a1" byrow)'
R> table_description=r"(range="sheet2!a1" byrow)"
```

Similarly, we can write string with single quote (') as follows:

```
R> table_description='range=\'sheet2!a1\' byrow'
R> table_descriptione="range='sheet2!a1' byrow"
R> table_description=r'(range='sheet2!a1' byrow)'
R> table_description=r"(range='sheet2!a1' byrow)"
```

```
R> set_eviews_path("C:/Program Files (x86)/EViews 10/EViews10.exe")
R> set_eviews_path("C:\\Program Files (x86)\\EViews 10\\EViews10.exe")
R> set_eviews_path(r'(C:\Program Files (x86)\EViews 10\EViews10.exe)')
R> set_eviews_path(r"(C:\Program Files (x86)\EViews 10\EViews10.exe)")
```

Therefore, we recommend using r'()' or r"()" to write complex strings, as both return the strings written between the braces exactly as they are.

3 Current knitr knit-engines

This section provides the list of knit-engines that are currently available as of November 10, 2023. Please note that eviews is included among the list because **EviewsR** has added it as a knit-engine.

```
#>  [1] "awk"        "bash"       "coffee"     "gawk"        "groovy"
#>  [6] "haskell"    "lein"       "mysql"      "node"        "octave"
#> [11] "perl"       "php"        "psql"       "Rscript"     "ruby"
#> [16] "sas"        "scala"      "sed"        "sh"          "stata"
#> [21] "zsh"        "asis"       "asy"        "block"       "block2"
#> [26] "bslib"      "c"          "cat"        "cc"          "comment"
#> [31] "css"        "ditaa"      "dot"        "embed"       "eviews"
#> [36] "exec"       "fortran"    "fortran95"  "go"          "highlight"
#> [41] "js"         "julia"      "python"     "R"           "Rcpp"
#> [46] "sass"       "scss"       "sql"        "stan"        "targets"
#> [51] "tikz"       "verbatim"   "glue"       "glue_sql"    "gluesql"
#> [56] "theorem"    "lemma"      "corollary"  "proposition" "conjecture"
#> [61] "definition" "example"    "exercise"   "hypothesis"  "proof"
#> [66] "remark"     "solution"
```

4 Session information

This section provides detailed information on the Operating System, R packages and their versions used in this document.

```
#> R version 4.2.2 (2022-10-31)
#> Platform: aarch64-apple-darwin20 (64-bit)
#> Running under: macOS 14.0
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
#>
#> locale:
#> [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
#>
#> attached base packages:
#> [1] grid       stats      graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
#> [1] magrittr_2.0.3   ggplot2_3.4.3    Gmisc_3.0.3      htmlTable_2.4.1
#> [5] Rcpp_1.0.11      kableExtra_1.3.4 knitr_1.45       EviewsR_0.1.5
#>
#> loaded via a namespace (and not attached):
#>  [1] lubridate_1.9.3     svglite_2.1.1       lattice_0.21-9
#>  [4] zoo_1.8-12          digest_0.6.33       utf8_1.2.3
#>  [7] R6_2.5.1            backports_1.4.1     evaluate_0.22
#> [10] httr_1.4.7          pillar_1.9.0        rlang_1.1.1
#> [13] data.table_1.14.8   rstudioapi_0.15.0   rpart_4.1.19
#> [16] checkmate_2.2.0     rmarkdown_2.25      webshot_0.5.5
#> [19] stringr_1.5.0       foreign_0.8-85      htmlwidgets_1.6.2
#> [22] tinytex_0.47        munsell_0.5.0       hunspell_3.0.3
#> [25] compiler_4.2.2      xfun_0.40           pkgconfig_2.0.3
#> [28] systemfonts_1.0.4   base64enc_0.1-3     htmltools_0.5.6.1
#> [31] nnet_7.3-19         forestplot_3.1.3    tidyselect_1.2.0
#> [34] gridExtra_2.3       tibble_3.2.1        bookdown_0.35
#> [37] rjtools_1.0.12      Hmisc_5.1-1         XML_3.99-0.14
#> [40] fansi_1.0.5         viridisLite_0.4.2   withr_2.5.1
#> [43] dplyr_1.1.3         gtable_0.3.4        lifecycle_1.0.3
#> [46] formatR_1.14        scales_1.2.1        cli_3.6.1
#> [49] stringi_1.7.12      fs_1.6.3            xml2_1.3.5
#> [52] xts_0.13.1          yesno_0.1.2         vctrs_0.6.3
#> [55] generics_0.1.3      Formula_1.2-5       tools_4.2.2
#> [58] glue_1.6.2          purrr_1.0.2         abind_1.4-5
#> [61] fastmap_1.1.1       yaml_2.3.7          timechange_0.2.0
#> [64] colorspace_2.1-0    cluster_2.1.4       BiocManager_1.30.22
#> [67] rvest_1.0.3
```

References

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2020. *Rmarkdown: Dynamic Documents for r*. https://github.com/rstudio/rmarkdown.

Christensen, Garret, and Edward Miguel. 2018. "Transparency, Reproducibility, and the Credibility of Economics Research." *Journal of Economic Literature* 56 (3): 920–80. https://doi.org/10.1257/jel.20171350.

Franco, Annie, Neil Malhotra, and Gabor Simonovits. 2014. "Publication Bias in the Social Sciences: Unlocking the File Drawer." *Science* 345 (6203): 1502–5. https://doi.org/10.1126/science.1255484.

Gerber, Alan, Kevin Arceneaux, Cheryl Boudreau, Conor Dowling, Sunshine Hillygus, Thomas Palfrey, Daniel R. Biggers, and David J. Hendry. 2014. "Reporting Guidelines for Experimental Research: A Report from the Experimental Research Section Standards Committee." *Journal of Experimental Political Science* 1 (1): 81–98. https://doi.org/10.1017/xps.2014.11.

Harvey, Campbell R., Yan Liu, and Heqing Zhu. 2015. "... and the Cross-Section of Expected Returns." *The Review of Financial Studies* 29 (1): 5–68. https://doi.org/10.1093/rfs/hhv059.

Ioannidis, John P. A. 2005. "Why Most Published Research Findings Are False." *PLOS Medicine* 2 (8). https://doi.org/10.1371/journal.pmed.0020124.

Mati, Sagiru. 2019a. *DynareR: Bringing the Power of Dynare to R, R Markdown, and Quarto*. https://CRAN.R-project.org/package=DynareR.

———. 2019b. *EviewsR: A Seamless Integration of EViews and R*. https://CRAN.R-project.org/package=EviewsR.

———. 2019c. *gretlR: A Seamless Integration of Gretl and R*. https://CRAN.R-project.org/package=gretlR.

———. 2023. *URooTab: Tabular Reporting of EViews Unit Root Tests*. CRAN. https://CRAN.R-project.org/package=EviewsR.

Mati, Sagiru, Irfan Civcir, and Huseyin Ozdeser. 2023. "ECOWAS Common Currency, a Mirage or Possibility?" *Panoeconomicus*, 1–25. https://doi.org/10.2298/PAN191119015M.

Mati, Sagiru, Irfan Civcir, and Hüseyin Ozdeser. 2019. "ECOWAS Common Currency: How Prepared Are Its Members?" *Investigación Económica* 78 (May): 89. https://doi.org/10.22201/fe.01851667p.2019.308.69625.

Murrell, Paul. 2019. **hexView**: *Viewing Binary Files*. https://CRAN.R-project.org/package=hexView.

Pretis, Felix, J. James Reade, and Genaro Sucarrat. 2018. "Automated General-to-Specific (**GETS**) Regression Modeling and Indicator Saturation for Outliers and Structural Breaks." *Journal of Statistical Software* 86 (3): 1–44. https://doi.org/10.18637/jss.v086.i03.

R Core Team. 2019. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Sandve, Anton AND Taylor, Geir Kjetil AND Nekrutenko. 2013. "Ten Simple Rules for Reproducible Computational Research." *PLOS Computational Biology* 9 (10): 1–4. https://doi.org/10.1371/journal.pcbi.1003285.

Simmons, Joseph P., Leif D. Nelson, and Uri Simonsohn. 2011. "False-Positive Psychology: Undisclosed Flexibility in Data Collection and Analysis Allows Presenting Anything as Significant." *Psychological Science* 22 (11): 1359–66. https://doi.org/10.1177/0956797611417632.

Startz, Richard. 2019. "EViews Illustrated." *University of California: Santa Barbara, CA, USA*.

Stodden, Victoria, Friedrich Leisch, and Roger D Peng. 2014. *Implementing Reproducible Research*. Crc Press.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Xie, Yihui. 2014. "**knitr**: A Comprehensive Tool for Reproducible Research in r." In *Implementing Reproducible Computational Research*, edited by Victoria Stodden, Friedrich Leisch, and Roger D. Peng. Chapman; Hall/CRC. http://www.crcpress.com/product/isbn/9781466561595.

———. 2015. *Dynamic Documents with r and* **knitr**. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.name/knitr/.

———. 2019. **knitr**: *A General-Purpose Package for Dynamic Report Generation in r*. https://yihui.name/knitr/.

Sagiru Mati
Near East University
[1]Operational Research Center in Healthcare
Near East University, Nicosia, North Cyprus
[2]Department of Economics
Yusuf Maitama Sule University, Nigeria
https://www.smati.com.ng
ORCiD: 0000-0003-1413-3974
sagirumati@gmail.com

Irfan Civcir
Ankara University
Department of Economics
Ankara, Turkey
http://cv.ankara.edu.tr/kisi.php?id=civcir@politics.ankara.edu.tr&deger=2
ORCiD: 0000-0002-2557-2625
civcir@politics.ankara.edu.tr

S. I. Abba
King Fahd University of Petroleum and Minerals
Interdisciplinary Research Center for Membranes and Water Security
Dhahran 31261, Saudi Arabia
https://scholar.google.com/citations?user=4h2JX7YAAAAJ&hl=en

ORCiD: 0000-0001-9356-2798

sani.abba@kfupm.edu.sa

langevitour: Smooth Interactive Touring of High Dimensions, Demonstrated with scRNA-Seq Data

by Paul Harrison

Abstract langevitour displays interactive animated 2D projections of high-dimensional datasets. Langevin Dynamics is used to produce a smooth path of projections. Projections are initially explored at random. A "guide" can be activated to look for an informative projection, or variables can be manually positioned. After a projection of particular interest has been found, continuing small motions provide a channel of visual information not present in a static scatter plot. langevitour is implemented in Javascript, allowing for a high frame rate and responsive interaction, and can be used directly from the R environment or embedded in HTML documents produced using R. Single cell RNA-sequencing (scRNA-Seq) data is used to demonstrate the widget. langevitour's linear projections provide a less distorted view of this data than commonly used non-linear dimensionality reductions such as UMAP.

1 Introduction

Understanding high-dimensional data is difficult. High-dimensional data is data where many variables have been measured at once. There may be complex relationships between variables, and the data may contain clusters and other features with complex shapes. This article introduces a new interactive tool that may be helpful for visualizing and understanding high-dimensional data using animated 2D projections.

High-dimensional data is produced in fields across the breadth of science. This article will focus on a motivating example from biology. Single cell RNA-sequencing (scRNA-Seq) typically measures the expression levels of thousands of genes in tens of thousands of biological cells. We can think of cells as points in a gene-expression space with thousands of dimensions. There is a complex high-dimensional geometry due to differences between biological cell types, variation in expression within cell types, cell developmental trajectories, and treatment responses. Principal Components Analysis (PCA) can find a set of directions in which the data is most variable, allowing scRNA-Seq data to be summarized down to perhaps tens of dimensions while still capturing most of the important geometry. However even a ten-dimensional space is difficult to comprehend.

One way to explore high-dimensional data is using a "tour." A tour is a sequence of projections of the dataset, most commonly into two dimensions. A Grand Tour (Asimov 1985) is a tour that will eventually visit as close as we like to every possible projection of the data, typically using a sequence of random projections. A Guided Tour, on the other hand, seeks an "interesting" projection by moving toward the maximum of some index function (Cook et al. 1995). The sequence of projections is animated, with smooth interpolation between each successive pair of projections. The software XGobi and GGobi (Swayne, Cook, and Buja 1998) provide an interactive graphical application incorporating tours for exploring high-dimensional data. The more recent R package tourr (Wickham et al. 2011) provides a framework for creating and displaying tours in the R language. Displaying animations directly in R usually does not achieve a high frame rate. It is also not possible to interact with the display as with GGobi. To get around these problems, a recent R package called detourr (Hart and Wang 2022) computes a tour path in R using tourr and then displays it using a Javascript widget (using htmlwidgets) (Vaidyanathan et al. 2021). The widget then provides a high frame rate display and interactive features. However, the projection path itself can not be modified interactively.

This article introduces a new R package, langevitour, that differs from previous tour software by using Langevin Dynamics, a method from physics, to produce a continuous path of projections. This path can be directly used for animation, eliminating the need to interpolate between distinct projections to animate the tour. The package is htmlwidgets-based, with interaction, calculations, and animation performed in Javascript. The projection can be controlled interactively, with the user able to switch between Grand and Guided Tours while also interactively focusing in on particular dimensions of interest.

The methods section describes Langevin Dynamics mathematically, but I will outline its important features here using two physical examples. First, consider modelling the position and velocity of a large particle over time. Many small particles continuously jostle the large particle. This is the original Brownian motion scenario studied by Langevin in 1908 (see translation by Lemons and Gythiel 1997). Rather than modelling every particle, Langevin Dynamics simulates the jostling as small random

forces. Langevin's model includes these random forces and damping of momentum, and we can also add a force field acting on the particle. The particle explores the space it is in, and the force field may cause the particle to spend more time near certain locations.

langevitour applies Langevin Dynamics to an orthonormal projection matrix rather than to a particle's position. As a second physical example, imagine a two-dimensional disk in a high-dimensional space. The disk represents a projection plane for a high-dimensional dataset. It has a fixed center but can rotate freely. Tiny unseen particles continuously jostle the disk, causing it to spin first one way and then another. The motion of the disk provides the path for a Grand Tour of the dataset. A force field may also draw it toward particular orientations. The force field is specified using a potential energy function. It is used to seek interesting data projections, similar to the index functions used in previous tour software, providing a Guided Tour.

This article begins by demonstrating the widget using data from the palmerpenguins package. The method and implementation are then described in detail. Finally, an extended demonstration using scRNA-Seq data is presented.

2 Palmer Penguins example

The R data package palmerpenguins (Horst, Hill, and Gorman 2020) provides body measurements of penguins of three different species from the Palmer Archipelago, Antarctica. The langevitour based visualization is shown in Figure 2. R code to produce this figure is:

```
library(langevitour)
library(palmerpenguins)

completePenguins <- na.omit(penguins[,c(1,3,4,5,6)])
scale <- apply(completePenguins[,-1], 2, sd)*4

langevitour(completePenguins[,-1], completePenguins$species,
            scale=scale, pointSize=2)
```

The widget displays a moving projection of high-dimensional points onto a two-dimensional plane. The current projection is indicated using a collection of axis lines, with the axes labelled by their respective variable names. A second set of variable and group labels appear to the right of the plot area when interacting with the plot. These can be dragged on to the plot to control the projection. Let us now step through some manipulations the langevitour widget allows.

- Setting a "guide" using the drop-down list. This causes langevitour to pursue projections near the minimum of an energy function. For example, the PCA guide seeks projections with large variance in both the x and y directions.
- Hiding particular groups by unchecking their checkbox in order to focus on other groups. For example by hiding Gentoo penguins, we can focus on the difference between Adelie and Chinstrap penguins. The guide is also only applied to the visible groups.
- Hiding particular axes by unchecking their checkbox. For example, without bill length Adelie and Chinstrap penguins can no longer be distinguished.
- Dragging labels onto the plot to concentrate on particular axes or try to separate a particular group. The projection may not exactly match the label positions since it must still be orthonormal.
- Adjusting the damping slider. High damping produces jerky Brownian motion. Less damping produces smoother, less random motion. An intermediate damping level is the fastest way to explore the space of projections thoroughly.
- Adjusting the heat slider. More heat makes the projection move faster, and stray further from the optimum projection defined by a guide or any labels dragged onto the plot.
- When the mouse is over a group label the group is highlighted.
- When the mouse is over an axis label a scale and rug are displayed, and points are colored according to their position on that axis.

Unchecking all but two or three axes can make the relationship between those particular axes clear. With three axes checked, the eye interprets the display as three-dimensional. A systematic way to examine a dataset is to proceed through all possible combinations of two or three axes.

Figure 1: palmerpenguins data visualized using **langevitour**. Each dot is a penguin, and the axes are four different penguin measurements. An interactive version of this figure is available in the supplemental file `figures-page.html`. (A) The widget initially spins at random. (B) The user has selected the PCA guide, and the widget has rotated to an informative projection using this guide. (C) The user has moused over a label, causing points to be colored by that variable. (D) The user has dragged a label onto the plot to concentrate on a particular variable.

If a particular interesting projection is found using **langevitour** it can be brought back into R by pressing the "?" button and copying and pasting R code that is shown. The "?" button also shows a JSON record of the current settings of the widget, including form inputs and label positions. All or some of these settings can be specified to a future call to langevitour(), or applied to a running widget using Javascript code. Example code to control the widget using HTML buttons and Javascript is given in the supplemental file figures-page.Rmd.

3 Method

Say we have a set of n p-dimensional data points. A $2 \times p$ projection matrix from p to 2 dimensions will be denoted **X**. The two rows of **X** are called a 2-frame. These rows must be unit vectors and orthogonal to each other. The set of all 2-frames in p dimensions is a Stiefel manifold.

It will often be necessary to consider all of the elements of the projection matrix concatenated together into a single vector ("melted"), which will be denoted **x**. We will think of **x** as a simulated physical system's "position" vector. With **x** in use as the position of the system, the data points will be called the vectors $\mathbf{y}_i$. The projection of point i into two dimensions is calculated with the matrix product $\mathbf{X}\mathbf{y}_i$.

3.1 Langevin Dynamics overview

Projections are generated using a numerical simulation of Langevin Dynamics but with projections constrained to lie on the Stiefel manifold. This section *briefly* summarizes Langevin Dynamics. The next section will describe the numerical simulation method and how the constraint is applied.

The description of Langevin Dynamics given here follows Leimkuhler and Matthews (2015), but for simplicity I set the Boltzmann constant to 1 and all masses to 1. We define a system with a position vector **x** and a velocity vector **v**. We must specify a temperature T, a damping rate γ, and a potential energy function $U(\mathbf{x})$. The behavior of the system is then defined by a pair of Stochastic Differential Equations (SDEs):

$$\mathrm{d}\mathbf{v} = -\gamma\mathbf{v}\mathrm{d}t + \sqrt{2\gamma T}\mathrm{d}\mathbf{W} - \nabla U(\mathbf{x})\mathrm{d}t \tag{1}$$
$$\mathrm{d}\mathbf{x} = \mathbf{v}\mathrm{d}t \tag{2}$$

Here **W** is a vector of Wiener processes. For any positive time-step Δt:

$$\mathbf{W}(t + \Delta t) - \mathbf{W}(t) \sim \sqrt{\Delta t}\mathcal{N}(0, \mathbf{I})$$

The total energy of the system, kinetic energy plus potential energy, is called the Hamiltonian:

$$H(\mathbf{x}, \mathbf{v}) = \frac{1}{2}|\mathbf{v}|^2 + U(\mathbf{x})$$

In Equation (1) in a physical system, the first two terms would describe the exchange of kinetic energy with the surrounding environment. In the first term, kinetic energy is lost (damping), while the second term adds randomness to the velocity, increasing the kinetic energy again. The third term applies acceleration according to the gradient of the potential energy function.

If we were to set $\gamma = 0$, we would be doing Hamiltonian Dynamics, and the system's total energy would remain constant. If $\gamma > 0$ the total energy can fluctuate, and in the long run the process is ergodic (Leimkuhler and Matthews 2015 in section 6.4.4), producing samples with the Gibbs-Boltzmann probability density:

$$\rho(\mathbf{x}, \mathbf{v}) \propto e^{-H(\mathbf{x}, \mathbf{v})/T}$$

From this density, it can be seen that each component of the velocity is normally distributed with variance T and that the position has probability density

$$\rho(\mathbf{x}) \propto e^{-U(\mathbf{x})/T}$$

The potential energy function completely controls the distribution of positions being produced, providing a great deal of freedom. Here, we will use this to craft suitable potential energy functions to allow the user to control the explored projections.

3.2 Langevin Tour numerical simulation

We are to generate a sequence of animation frames $i = 1, 2, ...$, each with a projection matrix $\mathbf{X}_i$ (written $\mathbf{x}_i$ when viewed as a vector). Each frame will also have a velocity vector $\mathbf{v}_i$. The time-step between frames can vary depending on the computational load from langevitour and other things happening in the web browser. Call the time-step from frame $i - 1$ to frame i Δt_i.

The Position Based Dynamics method (PBD, Müller et al. 2007) is used to perform the numerical simulation while constraining the system to produce orthonormal projection matrices. PBD is simple to implement and emphasizes stability over accuracy when enforcing constraints, which is appropriate and adequate for this application. Using PBD, in each iteration we will:

1. Update the velocity.
2. Update the position based on the velocity.
3. Fix the updated position to be an orthonormal projection matrix.
4. Fix the velocity to be consistent with the fixed position.

Step 1. Update the velocity

We will write $\mathbf{v}'_i$ and $\mathbf{x}'_i$ for the initially proposed velocity and position of the current frame. These will be adjusted in steps 3 and 4 to produce the final position and velocity, $\mathbf{x}_i$ and $\mathbf{v}_i$. The first step is to calculate

$$\mathbf{v}'_i = e^{-\gamma \Delta t_i} \, \mathbf{v}_{i-1} + \sqrt{T \left(1 - e^{-2\gamma \Delta t_i}\right)} \, \mathbf{r}_i - \Delta t_i \, \nabla U(\mathbf{x}_{i-1}) \tag{3}$$

where the components of $\mathbf{r}_i$ follow a standard normal distribution.

In the limit for $\Delta t_i \to 0$, Equation (3) matches the rate of change of the mean and rate of added variance in equation (1). Equation (3) has also been carefully chosen to have stable behavior for large Δt or γ or both. The first term decays the existing velocity by a factor of $e^{-\gamma \Delta t_i}$. If Equation (1) only contained the first term, this would be the exact solution. This decay reduces the variance of the velocity by a factor of $\left(e^{-\gamma \Delta t_i}\right)^2$. The second term re-injects variance sufficient to restore the variance of the velocity in every direction (orthogonal to constraints) as T.

A small refinement is made to avoid random rotation in the plane of projection, as this can be unsettling to view. Any part of the random noise $\mathbf{r}_i$ within the plane of the projection is subtracted out before the noise is added to the velocity. More precisely, considering the noise in matrix form $\mathbf{R}_i$ in the same way as the projection matrix $\mathbf{X}_{i-1}$, the projection of each row of $\mathbf{R}_i$ onto each row of $\mathbf{X}_{i-1}$ is subtracted from that row of $\mathbf{R}_i$. Previous tour software has also avoided this type of rotation, but in a different way, by using a "geodesic interpolation" method that operates between planes rather than frames (see Buja et al. 2005).

Step 2. Update the position based on the velocity

The position is advanced according to the velocity and the size of the time-step.

$$\mathbf{x}'_i = \mathbf{x}_{i-1} + \Delta t_i \mathbf{v}'_i$$

Step 3. Fix the updated position to be an orthonormal projection matrix

Position Based Dynamics requires the proposed position $\mathbf{x}'_i$ be projected back to a constraint-satisfying position $\mathbf{x}_i$. Here, the constraint is $\mathbf{x}_i$ represents an orthonormal projection matrix. Doing this arbitrarily might cause unexpected spinning in the projection plane. We must find the *nearest* valid $\mathbf{x}_i$ to $\mathbf{x}'_i$.

Considering the proposed position vector as a projection matrix, we take the singular value decomposition and set the singular values to 1.

$$\mathbf{U}\mathbf{S}\mathbf{V}^\top = \mathbf{X}'_i \tag{4}$$

$$\mathbf{X}_i = \mathbf{U}\mathbf{V}^\top \tag{5}$$

Here, $\mathbf{U}$ is a 2×2 orthonormal matrix, $\mathbf{S}$ is a 2×2 diagonal matrix, and $\mathbf{V}$ is a $p \times 2$ matrix with orthonormal columns. Let s_j be the values along the diagonal of $\mathbf{S}$, the singular values, all of which are non-negative.

Equation (5) chooses the closest orthonormal projection matrix in terms of Euclidean distance to $\mathbf{x}'_i$. Stated another way, this is the matrix $\mathbf{X}$ that minimizes the Frobenius norm $\|\mathbf{X}' - \mathbf{X}\|$ with the proposed projection matrix $\mathbf{X}'$. To see this, consider first the problem of finding the nearest orthonormal projection matrix to $\mathbf{U}^\top \mathbf{X}'$.

$$\mathbf{U}^\top \mathbf{X}' = \mathbf{U}^\top \mathbf{U} \mathbf{S} \mathbf{V}^\top = \mathbf{S} \mathbf{V}^\top$$

For each row in $\mathbf{U}^\top \mathbf{X}'$, $\left(\mathbf{U}^\top \mathbf{X}'\right)_{j,\cdot} = s_j \mathbf{V}^\top_{j,\cdot}$, the nearest unit vector will be parallel to this vector, namely $\mathbf{V}^\top_{j,\cdot}$. We know that the rows of $\mathbf{V}^\top$ are orthogonal, so $\mathbf{V}^\top$ is the nearest orthonormal projection matrix to $\mathbf{U}^\top \mathbf{X}'$. Multiplying both matrices by an orthonormal matrix does not change the Frobenius norm of their difference, so the nearest orthonormal projection matrix to $\mathbf{U}\mathbf{U}^\top \mathbf{X}' = \mathbf{X}'$ is $\mathbf{U}\mathbf{V}^\top$.

Step 4. Fix the velocity to be consistent with the fixed position

Position Based Dynamics requires the velocity to match the actual update made to the position, rather than the initially proposed update.

$$\mathbf{v}_i = \frac{\mathbf{x}_i - \mathbf{x}_{i-1}}{\Delta t_i}$$

3.3 Guiding projections using the potential energy function

We can use any function we like for the potential energy $U(\mathbf{x})$, so long as we can calculate its gradient. This is used in **langevitour** to provide a set of automatic guides and also as a method of interaction.

When an energy function is being used, the temperature T plays a role analogous to variance in the normal distribution. When the temperature is very low, the system seeks the minimum of the energy function. As the temperature is raised, projections further and further from the minimum are produced.

Linear energy function for interaction

For some choice of vector $\mathbf{a}$, we can set the energy function to be the dot product

$$U(\mathbf{x}) = -\mathbf{a}\mathbf{x}$$

This encourages the projection to have a large component parallel to $\mathbf{a}$. In **langevitour** this is used when labels are dragged onto the plot area to control the placement of particular axes of the high-dimensional space or to control the position of the mean of a group of points.

Central force energy function

The Box-Cox power transformation (Box and Cox 1964) provides a useful building block for energy functions.

$$f(x; \lambda_1, \lambda_2) = \begin{cases} \dfrac{(x + \lambda_2)^{\lambda_1} - 1}{\lambda_1} & \text{if } \lambda_1 \neq 0, \\ \ln(x + \lambda_2) & \text{if } \lambda_1 = 0 \end{cases}$$

An energy function creating forces away from or toward the center can be defined using:

$$U_{\text{central}}(\mathbf{x}; c, \lambda_1, \lambda_2) = \frac{c}{n} \sum_{i=1}^{n} f\left(|\mathbf{X}\mathbf{y}_i|^2; \lambda_1, \lambda_2\right)$$

langevitour offers a central "push" guide ($c < 0, \lambda_1 = 0.5, \lambda_2 = 0.0001$) and a central "pull" guide ($c > 0, \lambda_1 = 0.5, \lambda_2 = 0.0001$). These cone-shaped energy functions result in a nearly constant magnitude outward or inward force on points, except for a small region close to the center.

Layout by point-point repulsion

It was found that repulsion forces between pairs of points can be used to produce an informative layout. Let S be a set of pairs of points (i, j). Ideally we would make this the set of all possible pairs of points but, for computational efficiency, **langevitour** uses a random mini-batch of 5,000 pairs of points per iteration, with a different mini-batch used each time. Using random mini-batches to approximate the gradient injects extra noise into the system (see Mandt, Hoffman, and Blei 2017). The effect is similar to increasing the temperature slightly.

$$U_{\text{layout}}(\mathbf{x}; \lambda_1, \lambda_2) = \frac{c}{|S|} \sum_{(i,j) \in S} f\left(|\mathbf{X}\mathbf{y}_j - \mathbf{X}\mathbf{y}_i|^2; \lambda_1, \lambda_2\right)$$

The power parameter λ_1 determines whether the layout is governed by long-range or short-range forces. **langevitour** offers "ultra-local" ($c < 0, \lambda_1 = -1, \lambda_2 = 0.0025$), "local" ($c < 0, \lambda_1 = 0, \lambda_2 = 0.0001$), "PCA" ($c < 0, \lambda_1 = 1, \lambda_2 = 0$), and "outlier" ($c < 0, \lambda_1 = 2, \lambda_2 = 0$) guides. The "local" guide is the preferred default. With this guide, pairs of points that are near to each other exert more force than pairs of points that are far apart. The "ultra-local" guide potentially produces better layouts but is somewhat unstable. The "PCA" guide is equivalent to PCA. The "outlier" guide seeks projections where there are some points that are very far from other points.

Blending energy functions

A sum of energy functions such as the above can be used to produce behavior that blends the behaviors from the individual functions. For example, there could be an active guide and also one or more labels dragged onto the plot.

4 Implementation

langevitour uses the **htmlwidgets** framework. It was an important design goal that using **langevitour** be no more difficult than any other plotting function in R. **htmlwidgets** allows Javascript widgets to be used in most places that conventional R graphical output can be used. The widget may be displayed during an interactive R session or included in a knitted document with a call to the langevitour function. The only required argument is a matrix (or data frame) of numerical data. A grouping of rows is often also given, allowing points to be distinguished by color. There are further optional arguments providing adjustments to the scaling, appearance, and further optional features.

The **htmlwidgets** scaffoldWidget function was used to scaffold the package, including functions to create the widget (langevitour) and to use the widget in **shiny** applications (langevitourOutput, renderLangevitour).

langevitour operates without a server, so the R portion of **langevitour** is limited to sanity-checking all the inputs and ensuring **htmlwidgets** will translate the data to JSON consistently. In particular, vectors convert to lists, which ensures vectors of length 1 are not unboxed. A Javascript class performs calculation, plotting, and interaction. The D3 Javascript package is used to perform drag-and-drop interaction, color operations, scale operations, and some DOM element manipulation. The SVD-JS Javascript package is used for the singular value decomposition calculation. The jStat Javascript package is used to produce normally distributed random numbers. Besides these packages, calculations are performed using plain Javascript code, following the steps in the previous section.

Gradients need to be calculated in order to use potential energy functions as a guides or for interaction. The necessary partial derivatives were found by hand, and used to implement a collection of gradient functions. To add a new guide, a function to calculate the required gradient can be written, and the source code edited to make it available in the widget interface.

In Javascript, animation frames are scheduled using requestAnimationFrame, allowing the browser to manage the frame rate, co-ordinate multiple animations within a document, and pause animation when the document is not on screen. A typical frame rate for the browser to aim for is 60 frames per second. The frame rate may drop if there is too much calculation and drawing required, such as when there are many points to display. Multiple widgets may be active in a document at once, and even if the document is visible, not all widgets may be visible. To minimize CPU usage the animation is paused if a widget scrolls off-screen or is otherwise hidden. Canvas-based rendering was used.

5   scRNA-Seq example

A dataset by Kang et al. (2018) demonstrates many of the complex high-dimensional features that are found in scRNA-Seq data. In this dataset, peripheral blood mononuclear cells (PBMCs) from eight patients with lupus were pooled. PBMCs are cells from the immune system that circulate in blood, including monocytes, B cells, T cells, and natural killer (NK) cells. These cells were then stimulated with a cytokine, recombinant interferon beta, causing a change in the gene expression of the cells. The dataset contains a sample of unstimulated cells (U), and a separate sample of stimulated cells (S).

Single cell sequencing produces a small proportion of doublets, where two cells end up in a single micro-droplet and appear in the final data as a single cell. A nice feature of this dataset is that doublets containing cells from two different individuals can be identified with certainty due to genetic differences between the individuals.

5.1 Processing steps

Sequencing data was produced using a 10x Chromium Single Cell instrument and an Illumina HiSeq 2500 sequencer. Kang et al. (2018) then processed sequencing reads using the 10x Genomics CellRanger software and provided the resulting RNA molecule count data in the Gene Expression Omnibus (GEO) database as accession number GSE96583. They also provided their annotation of the cells into different types, and doublet detection based on genetic differences between individuals. Slightly simplified annotations are shown in this article. There are 29,065 cells in total. 3,169 of these are identified as doublets.

In the processed data for each of the two samples, there is a matrix giving the number of molecules of RNA associated with each gene within each cell. Normalization by total count per cell, log transformation, and PCA were carried out using the **Seurat** package (Hoffman 2022). As per Seurat defaults, only the top 2,000 highly variable genes are used. Each resulting Principal Component (PC) has a score for each cell and a loading for each gene.

The top PCs capture as much variation in the data as possible but are not necessarily individually interpretable. To aid biological interpretation, it would be better if each component represented changes in the expression of a distinct set of genes. Each differentially expressed gene should have loadings that are mostly concentrated in a single component, and we prefer the loadings to be positive if possible. With these goals in mind, the varimax rotation of the gene loadings was found using the `varimax()` function in the built-in stats package, with Kaiser normalization disabled. Both the gene loadings and the cell scores are rotated. Then, for each component, if the loadings have negative skew both the loadings and scores are negated.

Genetic differences can not identify doublets containing cells from the same individual, so Bioconductor package **scDblFinder** (Germain et al. 2022) was used to impute further doublets using the `recoverDoublets()` function. This only works between cells of different types, but doublets containing cells of the same type are not a problem. A further 595 doublets were identified this way.

The dataset was sub-sampled down to 10,000 cells to allow a smooth frame rate in **langevitour**,

The R code used to process the scRNA-Seq data is given in the supplemental file `processing.R`. Code for figure generation from the processed data is given in `figures.R`.

5.2 Cell scores

Results from analysis with **Seurat** are shown in Figure 2. The scree plot has a fat tail with no clear elbow. A large number of PCs potentially contain useful information. The top 10 PCs will be used simply as a manageable number with which to interact. Common practice is to visualize cells using a 2D UMAP layout computed from the PCs, as shown in Figure 2B, to see what clusters exist in the data and try to understand their relationships.

UMAP (McInnes, Healy, and Melville 2018) is a non-linear dimensionality reduction technique. Ironically, UMAP may give a curvy biological appearance to linear structures in the original data! Problems with UMAP are discussed by Coenen and Pearce (2019). They are similar to the problems with t-SNE (Wattenberg, Viégas, and Johnson 2016), an earlier method that UMAP has largely supplanted for scRNA-Seq analysis. Problems include that UMAP may arbitrarily change the distances between clusters and that UMAP will hide whether clusters are more or less spread-out by design.

Figure 3 shows the **langevitour** visualization of the cell scores. Components have been varimax rotated to improve interpretability (see previous section). With **langevitour**, we only ever see linear projections of the data. Straight lines remain straight, and parallelograms remain parallelograms. Distances may be decreased but will not be increased by the projection.

Figure 2: scRNA-Seq analysis using Seurat. (A) A scree plot showing variance accounted for by each PC. The scree plot has a fat tail, indicating that variation in the data can not be summarized with only a few PCs. (B) UMAP layout based on the cell scores of the first 10 PCs. U are unstimulated and S are stimulated cells.

Figure 3: scRNA-Seq cell scores visualized using langevitour. An interactive version of this figure is available in the supplemental file `figures-page.html`. The "local" guide is active. (A) Unstimulated cells. (B) All cells. (C) Colored by component 3. (D) Doublets hidden.

Figure 4: Noise reduction of scRNA-Seq cell scores. Original and denoised positions of 1,000 cells are shown, joined by lines. An interactive version of this figure is available in the supplemental file `figures-page.html`.

This is a moderately complex dataset. We will step through some manipulations of the widget in Figure 3 to try to understand it better.

- To simplify the dataset, hide the stimulated cells by unticking the checkbox on all the "S" labels.
- To find a projection giving a good overview of the relationships between unstimulated PBMC cells, activate the "local" guide from the drop-down list. The guided layout may not be able to perfectly separate all the clusters as UMAP can. However, small relative motions make clear where clusters are overlapping by accident rather than real proximity. For example, B cells and T cells are distinct.
- Examine the scale by mousing over the labels for particular axes. The scale for each component is meaningful, representing distance along a certain direction in scaled gene expression space. The direction is specified by the gene loadings, which are examined in a section below.
- Mouse over the "U Doublet" and "S Doublet" labels to highlight doublets. Doublets located between two clusters may be interpreted as a mixture of a pair of cells with different cell types. Hiding the doublets by unchecking their checkboxes makes the clusters more distinct.

Let us compare this data view to the UMAP layout (Figure 2B). In the linear view provided by langevitour, the monocytes are more spread out than other cell clusters. This isn't visible in UMAP which, as a deliberate feature, erases differences in scale. The whiskers extending from various clusters in the UMAP correspond to components at right angles to other components in the data, i.e., a subset of cells in which certain genes are active. For example, the thin whiskers extending from the unstimulated and stimulated monocytes extend in the same direction, along C7, but in the UMAP they extend in different directions. Doublets in the UMAP tend to form clumps near the edges of clusters or between clusters. In the linear data view, they are spread out between clusters.

k-nearest neighbor denoising

The fuzziness of the clusters in Figure 3 is an honest depiction of the data. However, to interpret the geometry of the data, it may be helpful to reduce the amount of noise. We want to do so with minimal distortion. A suggested method is implemented in **langevitour** in the function `knnDenoise()`, based on the k-nearest neighbor graph. The k-nearest neighbors to each point are first found. Then, each

Figure 5: Noise reduced scRNA-Seq cell scores.

point is updated to be the average of the set of points reachable within a certain number of steps along the directed k-nearest neighbors graph. Here $k = 30$ and two steps were used. This method is loosely inspired by the k-nearest neighbors smoothing method of Wagner, Yan, and Yanai (2018) and the use of the nearest neighbor graph in UMAP.

A comparison of the original and denoised cell positions is shown in Figure 4 for a sample of 1,000 cells. The effect has been to make clusters thinner and smaller but not to move them in space. The full result is shown in Figure 5.

We will again step through some widget manipulations, to understand the relationships between clusters and components.

- To provide an overview of the data, the "local" guide has already been activated.

- Doublets still lie between clusters in this denoised version. Compare this to the UMAP layout, where the doublets tend to be attached to one or other clusters. To make the clusters cleaner, hide the doublets by unticking the checkboxes on the doublet labels.

- To examine component C1, drag the C1 label on to the plot. This pulls out the monocyte cluster. It appears monocytes are associated with C1.

- To undo this action, drag the C1 label to the right to remove it from the plot area.

Similarly C5 pulls out CD8 T cells and NK cells, and C6 pulls out B cells. The response to the cytokine is mostly contained in C3 with a further monocyte specific response in C2.

5.3 Gene loadings

Each component examined in the preceding sections represents a certain pattern of gene expression. These patterns of gene expression are represented by the gene loadings. Within each component, each gene has a loading. As will be seen, the tendency is for large loadings to be concentrated in a relatively small set of genes for each component. Genes with large loadings may provide insights into the biology represented by each component based on their known function or involvement in known biological pathways.

The gene loadings may also be examined using **langevitour**, as shown in Figure 6. In this figure, the points are genes, and the variables are the loading for each component.

Figure 6: scRNA-Seq gene loadings visualized using langevitour. The points are genes and the variables are the loading for each component. An interactive version of this figure is available in the supplemental file `figures-page.html`. (A) Genes involved in component 3. (B) Genes involved in components 1 and 2.

- Observe the widget spinning freely for a while. The overall geometry is of a spiky ball. The spikes generally represent sets of genes with large loadings in one component.

- Drag particular component labels onto the plot to examine them. Biological phenomena such as different cell types (C5, C6) or responses to the cytokine (C3) often involve distinct sets of genes. Furthermore, cell types are defined more by genes' activation than genes' deactivation. Varimax rotation was able to align these distinct sets of genes into specific components. An exception is genes used by monocytes and the monocyte response to stimulation (C1 and C2). These two components show a broad fan of genes, which can be interpreted as the genes involved in being a monocyte also being involved to varying degrees in the monocyte response to stimulation.

- Mouse near points to see the specific genes they represent.

6 Discussion

Langevin Dynamics produces samples from a specified probability distribution and produces a path suitable for animation. Besides tours, a possible future application would be to use this to visualize uncertainty in the posterior distribution of a Bayesian statistical model while placing details of the model directly under interactive control. Visualization of samples from a distribution has been previously investigated by Hullman, Resnick, and Adar (2015), as the Hypothetical Outcome Plots (HOPs) method. HOPs displays a graphical representation of samples from a multivariate distribution one after the other. An appealing feature of HOPs is that any visualization may be used. The viewer can perceive the uncertainty in the distribution, and how the variables relate to each other in the values they might take. However the display of HOPs simply hops from one sample to the next. There is also earlier work in geospatial applications, such as by Ehlschlaeger, Shortridge, and Goodchild (1997) in which a smooth animation was produced by interpolating between a sequence of samples. They note that interpolated frames may no longer follow the original distribution, and propose a method that preserves the correct variance in interpolated frames for their specific application. By using Langevin Dynamics, a smooth animation could be produced in which each individual frame is naturally drawn from the correct distribution.

Langevin Dynamics is similar to the Hamiltonian Monte Carlo method (HMC, see Neal 2011) used for example by the Stan software package (Carpenter et al. 2017). Usually HMC alternates steps of completely randomizing the velocity and relatively long runs of Hamiltonian Dynamics simulation. This would produce a continuous path, but with occasional sharp turns. Neal (2011) describes a version of HMC with frequent partial velocity refreshment that more closely resembles Langevin Dynamics. An accept/reject step can make the sampling precisely accurate even with discrete time-steps. Another possibility, for large datasets, is to use mini-batch gradients for computational efficiency (Mandt, Hoffman, and Blei 2017). Mini-batch gradients provide a noisy estimate of the full gradient and, with careful tuning, the level of noise required for Langevin Dynamics will be injected into the

velocity. It may be possible to animate sampling from complex models in real-time smoothly.

Motion provides a channel of visual information not possible in static images. We are not accustomed to visualizing objects in more than three dimensions, but things that move together in the natural environment are usually physically connected, and this seems to be how our eyes interpret the small rotations in more than three dimensions displayed by langevitour.

A Javascript widget has been introduced for interactively exploring high-dimensional data. It is readily usable from the R environment, Shiny websites, or HTML documents generated using R Markdown, including static HTML reports, slideshows, and journal articles.

7 Acknowledgements

References

Asimov, Daniel. 1985. "The Grand Tour: A Tool for Viewing Multidimensional Data." *SIAM Journal on Scientific and Statistical Computing* 6 (1): 128–43. https://doi.org/10.1137/0906011.

Box, G E P, and D R Cox. 1964. "An Analysis of Transformations." *Journal of the Royal Statistical Society. Series B (Methodological)* 26: 211–52. http://www.jstor.org/stable/2984418.

Buja, Andreas, Dianne Cook, Daniel Asimov, and Catherine Hurley. 2005. "Computational Methods for High-Dimensional Rotations in Data Visualization." In *Data Mining and Data Visualization*, edited by C R Rao, E J Wegman, and J L Solka, 24:391–413. Elsevier. https://doi.org/10.1016/S0169-7161(04)24014-7.

Carpenter, Bob, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. "Stan: A Probabilistic Programming Language." *Journal of Statistical Software* 76. https://doi.org/10.18637/jss.v076.i01.

Coenen, Andy, and Adam Pearce. 2019. "Understanding UMAP." https://pair-code.github.io/understanding-umap/.

Cook, Dianne, Andreas Buja, Javier Cabrera, and Catherine Hurley. 1995. "Grand Tour and Projection Pursuit." *Journal of Computational and Graphical Statistics* 4 (September): 155–72. https://doi.org/10.1080/10618600.1995.10474674.

Ehlschlaeger, Charles R., Ashton M. Shortridge, and Michael F. Goodchild. 1997. "Visualizing Spatial Data Uncertainty Using Animation." *Computers & Geosciences* 23 (May): 387–95. https://doi.org/10.1016/S0098-3004(97)00005-8.

Germain, Pierre-Luc, Aaron Lun, Will Macnair, and Mark D. Robinson. 2022. "Doublet Identification in Single-Cell Sequencing Data Using scDblFinder." *F1000Research* 10 (May): 979. https://doi.org/10.12688/f1000research.73600.2.

Hart, Casper, and Earo Wang. 2022. *Detourr: Portable and Performant Tour Animations*. https://CRAN.R-project.org/package=detourr.

Hoffman, Paul. 2022. *Seurat: Tools for Single Cell Genomics*. https://CRAN.R-project.org/package=Seurat.

Horst, Allison Marie, Alison Presmanes Hill, and Kristen B Gorman. 2020. *Palmerpenguins: Palmer Archipelago (Antarctica) Penguin Data*. https://allisonhorst.github.io/palmerpenguins/.

Hullman, Jessica, Paul Resnick, and Eytan Adar. 2015. "Hypothetical Outcome Plots Outperform Error Bars and Violin Plots for Inferences about Reliability of Variable Ordering." *PLOS ONE* 10 (November): e0142444. https://doi.org/10.1371/journal.pone.0142444.

Kang, Hyun Min, Meena Subramaniam, Sasha Targ, Michelle Nguyen, Lenka Maliskova, Elizabeth McCarthy, Eunice Wan, et al. 2018. "Multiplexed Droplet Single-Cell RNA-sequencing Using Natural Genetic Variation." *Nature Biotechnology* 36 (1): 89–94. https://doi.org/10.1038/nbt.4042.

Leimkuhler, Ben, and Charles Matthews. 2015. *Molecular Dynamics with Deterministic and Stochastic Numerical Methods*. 1st ed. Interdisciplinary Applied Mathematics, 39.

Lemons, Don S., and Anthony Gythiel. 1997. "Paul Langevin's 1908 Paper 'On the Theory of Brownian Motion' ['Sur La Théorie Du Mouvement Brownien,' C. R. Acad. Sci. (Paris) 146, 530-533 (1908)]." *American Journal of Physics* 65 (November): 1079–81. https://doi.org/10.1119/1.18725.

Mandt, Stephan, Matthew D Hoffman, and David M Blei. 2017. "Stochastic Gradient Descent as Approximate Bayesian Inference." *Journal of Machine Learning Research* 18: 1–35. http://jmlr.org/papers/v18/17-214.html.

McInnes, Leland, John Healy, and James Melville. 2018. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction." *arXiv*. https://doi.org/10.48550/ARXIV.1802.03426.

Müller, Matthias, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. "Position Based Dynamics." *Journal of Visual Communication and Image Representation* 18 (2): 109–18. https://doi.org/10.1016/j.jvcir.2007.01.005.

Neal, Radford M. 2011. "MCMC Using Hamiltonian Dynamics." In *Handbook of Markov Chain Monte Carlo*, edited by Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng, 113–62. CRC Press.

Swayne, Deborah F., Dianne Cook, and Andreas Buja. 1998. "XGobi: Interactive Dynamic Data Visualization in the X Window System." *Journal of Computational and Graphical Statistics* 7 (1): 113–30. https://doi.org/10.1080/10618600.1998.10474764.

Vaidyanathan, Ramnath, Yihui Xie, JJ Allaire, Joe Cheng, Carson Sievert, and Kenton Russell. 2021. *Htmlwidgets: HTML Widgets for R*. https://CRAN.R-project.org/package=htmlwidgets.

Wagner, Florian, Yun Yan, and Itai Yanai. 2018. "K-Nearest Neighbor Smoothing for High-Throughput Single-Cell RNA-Seq Data." *bioRxiv*, April, 217737. https://doi.org/10.1101/217737.

Wattenberg, Martin, Fernanda Viégas, and Ian Johnson. 2016. "How to Use t-SNE Effectively." *Distill* 1 (October). https://doi.org/10.23915/distill.00002.

Wickham, Hadley, Dianne Cook, Heike Hofmann, and Andreas Buja. 2011. "Tourr: An R Package for Exploring Multivariate Data with Projections." *Journal of Statistical Software* 40. https://doi.org/10.18637/jss.v040.i02.

Paul Harrison
Monash Genomics and Bioinformatics Platform, Monash University
15 Innovation Walk
Monash University, Clayton Campus
Clayton VIC, Australia, 3800
https://logarithmic.net/pfh/
ORCiD: 0000-0002-3980-268X
paul.harrison@monash.edu

ggdensity: Improved Bivariate Density Visualization in R

by James Otto and David Kahle

Abstract The ggdensity R package extends the functionality of ggplot2 by providing more interpretable visualizations of bivariate density estimates using highest density regions (HDRs). The visualizations are created via drop-in replacements for the standard ggplot2 functions used for this purpose: geom_hdr() for geom_density_2d_filled() and geom_hdr_lines() for geom_density_2d(). These new geoms improve on those of ggplot2 by communicating the probabilities associated with the displayed regions. Various statistically rigorous estimators are available, as well as convenience functions geom_hdr_fun() and geom_hdr_fun_lines() for plotting HDRs of user-specified probability density functions. Associated geoms for rug plots and pointdensity scatterplots are also presented.

1 Introduction

Density estimation is foundational to modern statistics. Not only does it provide a theoretical basis for maximum likelihood estimation (Scott, 1992), it is also an important tool in exploratory data analysis. This is especially true for univariate data: histograms, frequency polygons, and kernel density estimates (KDEs) all visualize an estimated density.

With bivariate data, the situation is more complicated as the estimated density is a 3D surface, and there is a tendency to avoid 3D visualization in static graphics due to visual perception biases. A more common strategy is to represent the surface using other geometric objects or aesthetics in a 2D plot, most commonly via contours of the density's level sets. Typically the density is estimated with a KDE and the contours correspond to the level sets of an equally spaced mesh over $(0, M]$, where M is the maximum of the estimated density's height (usually rounded to the closest "pretty" value). We will refer to these contours as ordinate mesh density contours (OMDCs), and the corresponding graphics as traditional density contour plots or OMDC plots. By ordinate, we mean the variable $z = f(x, y)$, or in general the last element of the graph of a function $f(\mathbf{x})$, which in this context represents density.

For example, the **MASS** package documentation suggests using `MASS::kde2d()` with `graphics::contour()`, which selects its level sets by calling `base::pretty()` on 10 such breaks over the $(0, M]$ ordinate range, and **ggplot2**'s geom_density_2d() and geom_density_2d_filled() do the same (Venables and Ripley, 2002; Wickham, 2009; Wilkinson, 2005). Unfortunately, the resulting regions—those bounded by the OMDCs—cannot be immediately identified with corresponding probabilities, and are challenging to interpret in the best of cases.

Following Hyndman (1996), we propose the use of highest density regions (HDRs) as replacements for density visualization based on OMDCs. In a sense made rigorous in the next section, an HDR is the smallest region containing a certain percentage of the estimated distribution, e.g. 90%. An HDR contour is the boundary of this region. HDRs are constructed by determining "good" cutoff values for the density, whereby cutoff values of the density we mean the ordinate values corresponding to the HDR contours (i.e. the HDR contours are the level sets of these "good" cutoff values). Unlike OMDCs, HDR contours' ordinate values are almost never equally spaced in the ordinate range, and computing them presents a number of practical and technical challenges.

In this article we introduce **ggdensity**, a new R package intended to address these challenges in facilitating the visualization of bivariate HDRs and related topics in the **ggplot2** framework. **ggdensity** extends **ggplot2** with a tight integration: instead of wrapping **ggplot2** calls to return ggplot objects that are hard to modify, **ggdensity** uses **ggplot2**'s API to provide new extensible (geom, stat) pairs that behave in the way **ggplot2** users have come to expect. These new stats provide a range of density estimation options, as we describe in the next sections.

2 Motivating example

We begin with a motivating example to show how traditional density contour plots can be misleading when exploring bivariate distributions. The top left plot in Figure 1 is a scatterplot of simulated bivariate standard normal data whose distribution we want to visualize. On the bottom left, we present the traditional way of visualizing the data's 2d distribution: a contour plot of slices of its estimated density (OMDCs). The function that created this graphic, geom_density2d() (alternatively geom_density_2d()), has been in **ggplot2** since its inception. It was modeled after a similar graphic made with base graphics using `MASS::kde2d()` with `contour()`. In the top right is the filled contour

version of the same plot, made with `geom_density_2d_filled()`. This type of plot was introduced in 2020 with **ggplot2** version 3.3.2, which leveraged **ggplot2**'s new dependency on **isoband** that came in **ggplot2** version 3.3.0 (Wilke and Pedersen, 2021).

In the bottom right is our proposed alternative, `ggdensity::geom_hdr()`. Each of the three contour plots show contours from the same estimated density surface, but the contours plotted by `geom_hdr()` are HDR contours and are chosen to be inferentially relevant. By default these are the smallest regions containing 50%, 80%, 95%, and 99% of the estimated density.

Plotting the HDRs results in a significantly more interpretable graphic that conveys more information than equally spaced density contours. To make a more direct comparison, in Figure 2 we superimpose the HDR contours onto the filled traditional density contour plot in the top right of Figure 1. The result reveals that nearly 20% of the estimated distribution is outside the lowest OMDC. Consequently, we would expect almost 1 out of every 5 observations to fall outside the traditional density contour plot.

This is somewhat of a cautionary tale: while the contours seen in the bottom left and top right plots of Figure 1 do seem to communicate some information to the viewer, it's hard to say exactly what that information is. And worse: it seems surprisingly easy to draw wrong conclusions. Upon scrutiny, the overall OMDC strategy seems suspect as a general purpose tool for visualizing where the probability mass of a bivariate distribution resides as it focuses exclusively on the density and ignores the region over which that density extends. On the other hand, with HDRs one immediately understands where the majority of the observed data lie and roughly how much data lies in each region. It is not possible to achieve these insights with traditional density contour plots, since equivalent interpretations require double integrals of the estimated density.

Figure 1: Comparing various geoms on a bivariate standard normal sample of size $n = 2500$.

Figure 2: `geom_density_2d_filled()` and `geom_hdr()` (white) from Figure 1, showing that the traditional density contour plots can mislead: nearly 20% of the estimated distribution falls outside the lowest OMDC. Labels generated with **geomtextpath** (Cameron and van den Brand, 2022).

3 Highest density regions

More formally, following Hyndman (1996) we adopt the following definition of highest density regions (HDRs):

Definition 1 *Let $f(x)$ be the probability density function (PDF) of a random vector $\boldsymbol{X} \in \mathbb{R}^p$ and $\alpha \in (0,1)$. For any constant $c \in \mathbb{R}$ define $\mathcal{R}_f(c) = \{x \in \mathbb{R}^p : f(x) \geq c\}$ to be the subset of the sample space of $\boldsymbol{X}$ with density at least c. The $100(1-\alpha)\%$ HDR of $\boldsymbol{X}$ is the subset $\mathcal{R}_f(f_\alpha)$, where f_α is the largest constant c such that $p_f(c) := P_f\left[\boldsymbol{X} \in \mathcal{R}_f(c)\right] \geq 1-\alpha$. When $f(x)$ is clear, we will simplify this to $\mathcal{R}_\alpha = \mathcal{R}_f(f_\alpha)$.*

Note that $p_f(c)$ is non-increasing in c. Intuitively, we see that as c gets bigger the region where $f(\mathbf{x}) \geq c$ shrinks (or possibly remains unchanged), and so the probability of the region gets correspondingly smaller (or at least can't get bigger). f_α is the largest point at which this probability is at least $1-\alpha$. We revisit this in the next section.

Of course, in practice we are given data $\mathbf{x}_1, \ldots, \mathbf{x}_n$, ideally a random sample from $f(\mathbf{x})$, and we need to estimate the population HDRs. The problem of estimating HDRs, and more generally density contour estimation, has been widely studied and several estimators have been proposed. Estimators generally fall in one of three classes: plug-in estimators (Rigollet and Vert, 2009; Cadre, 2006), excess mass estimators (Muller and Sawitzki, 1991; Polonik, 1995), and convex contour estimators (Hartigan, 1987). In this work we focus exclusively on plug-in estimators of HDRs due to their straightforward interpretation and implementation. These estimators are of the form $\widehat{\mathcal{R}}_\alpha = \widehat{\mathcal{R}}_f(f_\alpha) := \mathcal{R}_{\hat{f}}(\hat{f}_\alpha)$ for some PDF estimate $\hat{f}(\mathbf{x})$ of $f(\mathbf{x})$, where $\hat{f}_\alpha$ is the largest value c such that $\hat{p}_f(c) := p_{\hat{f}}(c) = P_{\hat{f}}\left[\mathbf{X} \in \mathcal{R}_{\hat{f}}(c)\right] \geq 1-\alpha$.

Thus, plug-in HDR estimators estimate population HDRs with the HDRs of estimated densities. As there are many ways to estimate a density, both parametric and nonparametric, one can arrive at many different HDR estimates with the same data and probability mass $1-\alpha$, and different choices confer advantages and disadvantages. We revisit this notion in HDRs using different density estimators using **ggdensity** after explaining how such estimates can be computed.

Before addressing that topic, it is worth reflecting on another aspect that makes HDRs so special: their size. While HDRs are primarily of interest because their corresponding probabilities are immediately interpretable, they are also important because they are the smallest such sets that contain their probabilities.

For any continuous distribution, there are an infinite number of different regions in its support that contain probability at least $1-\alpha$. The standard normal distribution provides a simple univariate example. If $\Phi^{-1}(x)$ is the quantile function of the standard normal distribution and $0 \leq l < u \leq 1$ with $u - l = 1 - \alpha$, any interval of the form $[\Phi^{-1}(l), \Phi^{-1}(u)]$ contains probability exactly $1-\alpha$, and indeed these are all such intervals that do. This is because $P\left[\Phi^{-1}(l) \leq Z \leq \Phi^{-1}(u)\right] = P\left[Z \leq \Phi^{-1}(u)\right] - P\left[\Phi^{-1}(l) \leq Z\right] = \Phi(\Phi^{-1}(u)) - \Phi(\Phi^{-1}(l)) = u - l = 1 - \alpha$. Setting $1 - \alpha = .95$, we can choose $l = 0, u = .95$ yielding the interval $(-\infty, 1.645]$. We can find similar intervals with other choices of l, u:

$[-2.326, 1.751]$; $[-2.054, 1.881]$; $[-1.881, 2.054]$; $[-1.751, 2.326]$; and $[-1.645, \infty)$, all of which contain 95% probability.

However, all intervals are typically not equally preferable: if we are interested in summarizing the normal distribution with a set of probability $1 - \alpha$, we typically want to provide the smallest such set. The intervals above have lengths ∞, 4.08, 3.93, 3.93, 4.08, and ∞. As is well-known, if we want the smallest interval, we use $[\Phi^{-1}(\alpha/2), \Phi^{-1}(1 - \alpha/2)] = [-1.96, 1.96]$ as the interval bounds, with the interval length of 3.92. This interval corresponds to the region where the density exceeds $f_\alpha = \phi(\Phi^{-1}(\alpha/2))$; and as such meets the definition of an HDR.

This is a general feature of HDRs: whether in one or many dimensions, they constitute the smallest regions containing their corresponding probabilities, a fact seen from the following measure-theoretic argument. For some $f(\mathbf{x})$, $\alpha \in [0, 1]$, and an associated HDR $\mathcal{R}_\alpha$, let $\eta = P[\mathbf{X} \in \mathcal{R}_\alpha]$. While η may be equal to $1 - \alpha$, in some cases it may be more, for instance in the case of the uniform distribution. Suppose $\mathcal{A}$ is a subset of the sample space of $\mathbf{X}$ with probability $\xi \geq \eta$ such that $\mathcal{A} \setminus \mathcal{R}_\alpha$ is non-null, i.e. $\mathcal{A}$ and $\mathcal{R}_\alpha$ are not the same set. Then

$$\eta = P[\mathbf{X} \in \mathcal{R}_\alpha] = \int_{\mathcal{R}_\alpha} f(\mathbf{x})d\mathbf{x} = \int_{\mathcal{R}_\alpha \cap \mathcal{A}} f(\mathbf{x})d\mathbf{x} + \int_{\mathcal{R}_\alpha \setminus \mathcal{A}} f(\mathbf{x})d\mathbf{x}.$$

Similarly,

$$\xi = P[\mathbf{X} \in \mathcal{A}] = \int_{\mathcal{R}_\alpha \cap \mathcal{A}} f(\mathbf{x})d\mathbf{x} + \int_{\mathcal{A} \setminus \mathcal{R}_\alpha} f(\mathbf{x})d\mathbf{x}.$$

Since $\xi \geq \eta$,

$$\int_{\mathcal{A} \setminus \mathcal{R}_\alpha} f(\mathbf{x})d\mathbf{x} \geq \int_{\mathcal{R}_\alpha \setminus \mathcal{A}} f(\mathbf{x})d\mathbf{x}$$

By the definition of $\mathcal{R}_\alpha$, $f_\alpha > f(\mathbf{x})$ over $\mathcal{A} \setminus \mathcal{R}_\alpha$ and $f(\mathbf{x}) \geq f_\alpha$ over $\mathcal{R}_\alpha$. Thus,

$$f_\alpha m(\mathcal{A} \setminus \mathcal{R}_\alpha) = \int_{\mathcal{A} \setminus \mathcal{R}_\alpha} f_\alpha d\mathbf{x} > \int_{\mathcal{A} \setminus \mathcal{R}_\alpha} f(\mathbf{x})d\mathbf{x} \geq \int_{\mathcal{R}_\alpha \setminus \mathcal{A}} f(\mathbf{x})d\mathbf{x} \geq \int_{\mathcal{R}_\alpha \setminus \mathcal{A}} f_\alpha d\mathbf{x} = f_\alpha m(\mathcal{R}_\alpha \setminus \mathcal{A}),$$

where m denotes the Lebesgue measure of the given set, its size. Thus $m(\mathcal{A} \setminus \mathcal{R}_\alpha) > m(\mathcal{R}_\alpha \setminus \mathcal{A})$ and consequently $m(\mathcal{A}) > m(\mathcal{R}_\alpha)$.

The minimality of HDRs comes with a few trade-offs. First, there may be other regions of the sample space with probability between $1 - \alpha$ and η, the nominal and actual probabilities of the HDRs respectively, that are smaller than the HDR. This can happen in cases where the PDF is constant on some set of positive measure so that $p_f(c)$ jumps discontinuously over $1 - \alpha$. The $100(1 - \alpha)\%$ quantile interval of the Unif$(0, 1)$ distribution, $[\alpha/2, 1 - \alpha/2]$, illustrates this, for instance; it contains probability exactly $1 - \alpha$ and is of length $1 - \alpha$. By contrast, the $100(1 - \alpha)\%$ HDR for the uniform distribution is $\mathcal{R}_\alpha = [0, 1]$ for any α, contains 100% of the distribution, and is of length 1. While this is a rare circumstance when using density estimators, it does occur when constructing HDRs from histogram density estimates – they always contain more than nominal probability. The second trade-off is that HDRs are only connected sets for all α if the distribution is unimodal. This is rarely the case for density estimators, so it is common for the lowest probability HDRs, the smallest sets, to be disconnected: two or more intervals in 1D and unions of blobs in 2D. A third challenge is that HDRs are non-trivial to compute. It is to this challenge that we now turn.

4 Computing highest density regions

ggdensity enables the computation and visualization of 2D HDRs based on various plug-in estimators, extending the functionality of **ggplot2**. In order to understand how HDRs are computed in two dimensions, we find it helpful to first illustrate the process in one dimension.

4.1 Computing HDRs in one dimension

Consider the challenge of computing the 95% HDR of the standard normal distribution with PDF $f(x) = \phi(x)$. One way to do this might be to fix a c, determine the interval $[l, u]$ over which $\phi(x) \geq c$, integrate $\phi(x)$ over $[l, u]$, and move c up or down depending on whether the interval contains too much or too little probability. l and u, the boundaries of the interval where $\phi(x) \geq c$, can be determined numerically or algebraically from $\phi(x) = c$, and the integral can be done via numerical integration. f_α, the special c that provides $1 - \alpha$ probability, can also be determined numerically in many ways.

Unfortunately, this method doesn't scale well. In general, the set $\mathcal{R}_f(c)$ will not be a simple interval as in the normal case. It will instead be a union of an unknown number of intervals whose boundaries are unknown and hard to determine. Once known, numerical integration can be relied

upon to determine the integral, but only easily in one dimension. In two or more dimensions, the situation is significantly more complex. The region $\mathcal{R}_f(c)$ is implicitly described, and therefore some form of grid-based approximation would be required before numerical integration could be applied. Notice that there are essentially two hard problems here: computing $\mathcal{R}_f(c)$, in the sense of determining a useful description of it, and computing $p_f(c)$.

The basic idea used by **ggdensity** is simple: discretize and compute. In the univariate case, the fundamental algorithm is this:

1. Evaluate $f(x)$ on a regular mesh $\{x_i : i = 1, \ldots N\}$ to obtain $f_i = f(x_i)$ over some interval nominally larger than the support of the data to create a table with rows (x_i, f_i) for $i = 1, \ldots, N$.

2. Normalize the points into a discrete distribution $p_i = f_i / \sum_i f_i$ to create the table (x_i, f_i, p_i).

3. Sort the N rows of the table (x_i, f_i, p_i) by p_i in decreasing order to obtain $(x_{(i)}, f_{(i)}, p_{(i)})$.[1]

4. Compute the cumulative sum $a_{(k)} = \sum_{i=1}^{k} p_{(i)}$ to create the table $(x_{(i)}, f_{(i)}, p_{(i)}, a_{(i)})$.

5. Estimate f_α with the first $f_{(i)}$ such that $a_{(i)} \geq 1 - \alpha$.

The point masses p_i of the discrete approximation are in fact the areas of the rectangles of a Riemann (i.e. piecewise constant) approximation to $f(x)$ collapsed to individual points: they approximate the probability over a range such as $[x_i - \delta/2, x_i + \delta/2]$, where δ is the resolution of the mesh. Just as Riemann approximations converge to the true value of the integral under very minor conditions on $f(x)$, arbitrarily accurate approximations to f_α can be obtained by setting N suitably large. Because of this, we will refer to approximating f_α as "computing" f_α instead of estimating it, and our notation will not reflect the fact that it is an approximated quantity.

With f_α in hand, the HDR can be approximated in any of a number of ways that are practically equivalent for sufficiently large N. The easiest is to simply union of intervals $[x_i - \delta/2, x_i + \delta/2]$ such that $f_i \geq f_\alpha$. Figure 3 provides an illustration of the process using a very coarse mesh to emphasize the process.

Figure 3: $p_f(c)$ and $\mathcal{R}_f(c)$ are computed by discretizing the density $f(x)$, determining the probabilities with densities above c, and constructing HDRs as unions of intervals. In this illustration, $c = .23$, yielding $p_f(c) = .763$.

---

[1]This is actually the opposite of order statistics notation: here $p_{(1)}$ is the *largest* probability of the discretized distribution, and $x_{(1)}$ is the corresponding x_i value.

Figure 4: As the mesh size N grows, the HDR approximation improves for any $f(x)$. Here $N = 100$ and the 90% HDR is illustrated.

N governs the accuracy of the approximation of $p_f(c)$ and the accuracy and resolution of $\mathcal{R}_f(c)$ in the resulting plot. Consequently, a reasonably large number is desired: **ggdensity** defaults this parameter to $N = 512$ in `geom_hdr_rug()`. Figure 4 illustrates a more complicated example with a bimodal f where $N = 100$ is used so that it is still possible to see the approximation.

Two adjustments are needed to fully operationalize 1D HDRs for data: using an approximation $\hat{f}(x)$ in place of $f(x)$ and plotting several HDRs. Fortunately, both of these are easy. In practice density estimator implementations usually return their estimates as values of the function $\hat{f}_i = \hat{f}(x_i)$ on a mesh, not as some analytic expression. Such is the case for `stats::density()`, for example, which accepts a univariate vector and returns its estimated density at $N = 512$ points on a regular mesh a little larger than the data. Similarly, computing several HDRs at once requires virtually no added computational expense: HDRs are nested regions, so to find several HDRs simply continue looking down the list of accumulated probabilities $a_{(i)}$ until the desired probabilities are reached. This is illustrated in Figure 5.

Figure 5: Computing several HDRs can be done with no added computational complexity, since the regions are nested. Here the 50%, 80%, and 90% HDRs $\widehat{\mathcal{R}}_\alpha$ are illustrated using an estimated density based on $n = 1000$ draws.

4.2 Finding HDRs in two dimensions

The procedure for finding $\mathcal{R}_\alpha$ for bivariate data is very similar to the univariate case. The basic idea behind the computation of HDRs in **ggdensity** is again to discretize $f(\mathbf{x}) = f(x, y)$ and compute. The fundamental algorithm is this:

1. Evaluate $f(x, y)$ on a fine mesh $\{(x_i, y_j) : i = 1, \ldots N_x, j = 1, \ldots N_y\}$ to obtain $f_{ij} = f(x_i, y_j)$ over some rectangular region nominally larger than the support of the data to create a table (x_i, y_j, f_{ij}) for $i = 1, \ldots, N_x$ and $j = 1, \ldots, N_y$. In practice, we usually use $N = N_x = N_y$.

2. Normalize the points into a discrete distribution $p_{ij} = f_{ij} / \sum_{i,j} f_{ij}$ to obtain the table $(x_i, y_j, f_{ij}, p_{ij})$.

3. Sort the rows of the table by p_{ij} in decreasing order to obtain $(x_{(k)}, y_{(k)}, f_{(k)}, p_{(k)})$, where $k = 1, \ldots, N_x \times N_y$ and the parenthetical notation is used for consistency with the univariate case. The original order is immaterial to the algorithm.

4. Compute and append the cumulative sum $a_{(k)} = \sum_{i=1}^{k} p_{(i)}$ to the table to obtain the table $(x_{(k)}, y_{(k)}, f_{(k)}, p_{(k)}, a_{(k)})$.

5. Estimate f_α with the first $f_{(k)}$ such that $a_{(k)} \geq 1 - \alpha$.

Similar to the univariate case, the point masses p_{ij} of the discrete approximation can be thought of as the volumes of the rectangular prisms representing a Riemann approximation to $f(x, y)$ collapsed to individual points. And again, just as Riemann approximations converge to the true value of the integral, arbitrarily accurate approximations to f_α can be obtained by setting N_x and N_y suitably large for any reasonable $f(x, y)$. We illustrate this in Figure 6. In practice, $N = N_x = N_y$ governs the resolution of the HDRs in the resulting plot and is set to a suitably large number, **ggdensity** defaults this parameter to 100. Note, too, that this same approach will work in three dimensions and more, however, the computational complexity does not scale into higher dimensions well, and **ggdensity** does not support 3D graphics, so we do not pursue this further here.

In one dimension, the HDR corresponding to the points for which $f(x) \geq f_\alpha$ is naturally described by union of the corresponding rectangular regions of the Riemann approximation. The same could be done in two dimensions, too, as illustrated in Figure 6, however, any contour generating algorithm would work. **ggdensity** uses an implementation of marching squares provided by the **isoband** package (Wilke and Pedersen, 2021). Given a rectangular array of zeros and ones, the basic function implementing the algorithm results in parametrically-described polygonal regions whose interior contains only the points corresponding to the 1's. In the present setting, such an array is provided by the table with $f_{ij} \geq f_\alpha$. This is illustrated in Figure 7.

Figure 6: In two dimensions and by analogy with Figure 3, f_α is computed by discretizing the density $f(x, y)$, and HDRs are constructed from points (x_i, y_j) where $f(x_i, y_j) \geq f_\alpha$. Here $\alpha = .05$ was used.

Figure 7: Figure 6 from the plane perspective. **ggdensity** constructs HDRS by applying the marching squares contouring algorithm to binary grid where $f_{ij} \geq f_\alpha$ provided by `isolines()` and `isobands()`.

An alternative approach is worth mentioning at this point. The method described above is essentially what Hyndman (1996) refers to as the "numerical integration approach". However, alternative approaches exist. Hyndman (1996) suggests using the simple, consistent quantile estimate $\hat{f}_\alpha = \hat{f}_{(j)}$, where $\hat{f}_{(j)}$ is the (j/n) sample quantile of $\{\hat{f}(x_i, y_i)\}$ and $j = \lfloor \alpha n \rfloor$. Presented with data $(x_1, y_1), \dots, (x_n, y_n)$, if $f(x, y)$ is known, any estimate of the $1 - \alpha$ quantile of $f(X, Y)$ is an estimate of f_α; this is referred to as the "density quantile approach". Notice that this requires contours intersect at least one data point and forces a certain proportion of observed values outside of the HDRs, regardless of sample size. Unavailable in **ggdensity**, this method[2] is implemented in both **hdrcde** (Hyndman et al., 2021) and **gghdr** (O'Hara-Wild et al., 2022).

---

[2]There are many valid choices of $\hat{f}_\alpha$, **hdrcde** and **gghdr** make use of `stats::quantile()` with `type = 7` which estimates a continuous sample quantile function with linear interpolation, a slight modification of the strategy outlined in Hyndman (1996).

4.3 HDRs using different density estimators

As we noted previously, sample HDRs can be computed using many density estimation methods. Figure 8 illustrates how 95% HDRs are calculated for histogram estimators and KDEs on a sample of size $n = 1,000$ from the standard bivariate normal distribution. These use the exact same method as previously described.

Figure 8: ggdensity facilitates using different density estimators to determine HDRs, including histograms (top, 11 bins in each dimension) and kernel density estimators (bottom), the default. Here 95% HDRs are illustrated using $N = 25$ and $n = 1,000$ draws from the standard bivariate normal distribution. The illustration refects the method of construction, not output of geom_hdr().

Figure 9 displays the output of geom_hdr() using the full range of methods available for three different simulated data sets with different features. By default, geom_hdr() and geom_hdr_lines() plot the 50%, 80%, 90% and 95% HDRs. The methods are available in both geom_hdr() and geom_hdr_lines() through the method argument, which allows for the specification of various nonparametric and parametric estimators, each offering advantages in certain contexts. For example, histogram estimators result in HDRs that obey constrained supports. Normal estimators, i.e. the best-fit bivariate normal estimator, can be helpful in providing simplified visuals that give the viewer a sense of where the distributions are, potentially at the expense of over-simplifying and removing important features of how the variables co-vary.

Figure 9: Comparing HDRs obtained with different `method` arguments to `geom_hdr()`.

4.4 HDRs from known density functions

The same method can be used to determine the HDRs of a given density function. This is implemented for bivariate densities in **ggdensity** as geom_hdr_fun() and geom_hdr_lines_fun(), both accepting a PDF via the fun argument. Figure 10 illustrates this by plotting the HDRs of bivariate random vector $\mathbf{X} = (X_1, X_2)$ with $X_1 \perp X_2$, $X_1 \sim \mathcal{N}(0,1)$ and $X_2 \sim \text{Gamma}(5,3)$ (left), and $\mathbf{Y} \sim f_{\mathbf{Y}}(y_1, y_2) \propto \exp\left\{-\frac{1}{2(.20)^2}(y_1^2 + y_2^2 - 1)^2\right\}$ (right), which concentrates its probability along the unit circle $\mathcal{S}^1$. In this case we make use of the fact that geom_hdr_fun() can find the HDRs of unnormalized PDFs. It does so by leveraging the fact that over a given window, the discretization is not affected by whether or not the density is normalized.

```
f_X <- function(x1, x2) dnorm(x1) * dgamma(x2, 5, 3)
ggplot() + geom_hdr_fun(fun = f_X, xlim = c(-4, 4), ylim = c(0, 5))
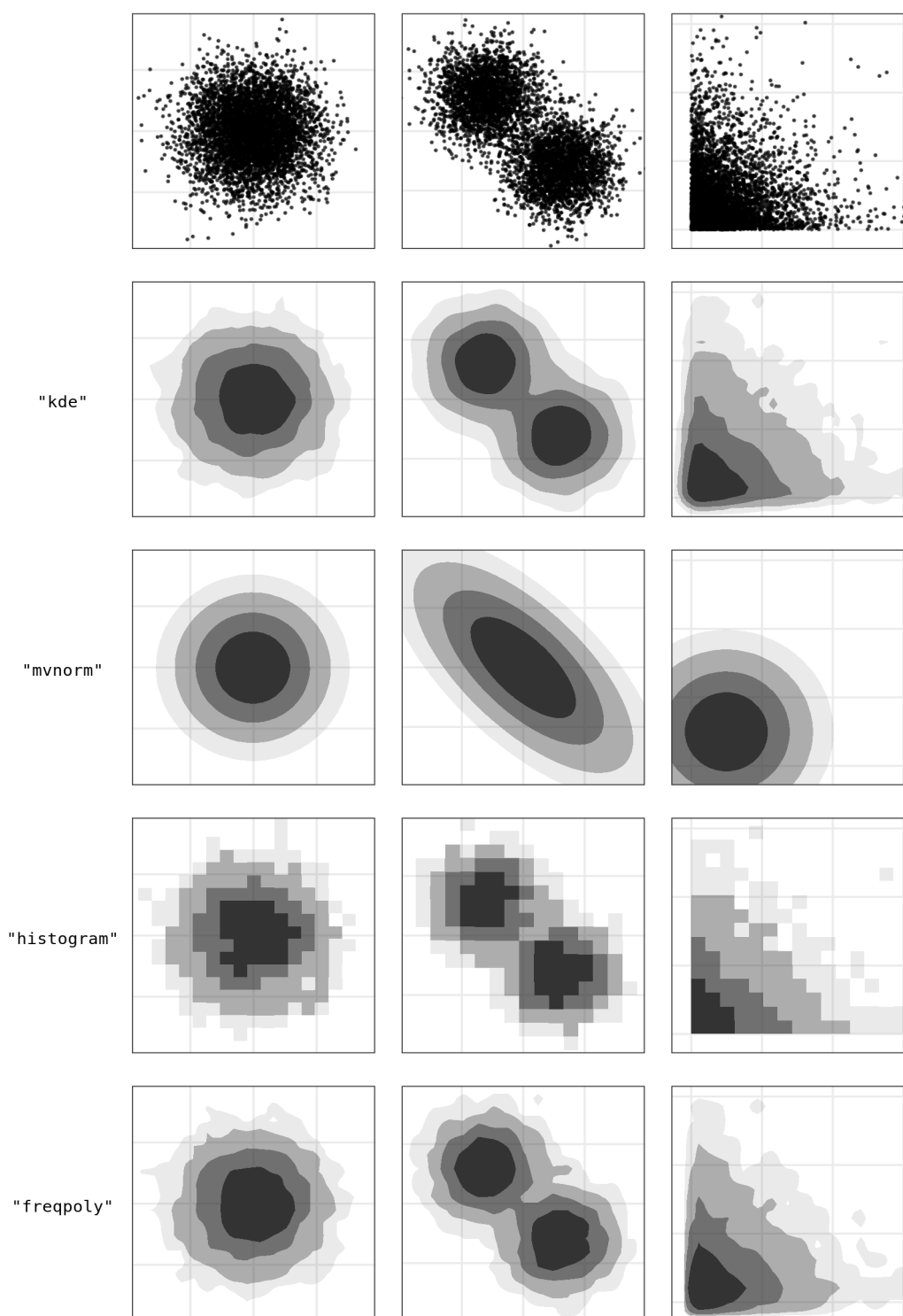

f_Y <- function(y1, y2) exp(-1/(2 * .20^2) * (y1^2 + y2^2 - 1)^2)
ggplot() +
  geom_hdr_fun(fun = f_Y, normalized = FALSE, xlim = c(-4, 4), ylim = c(-4, 4)) +
  coord_equal()
```


Figure 10: geom_hdr_fun() can be used to plot HDRs of normalized and un-normalized known PDFs.

In both of these examples, determining the exact contours is a nontrivial proposition. However, we have not had to derive any results regarding the distributions of $\mathbf{X}$ or $\mathbf{Y}$ to plot them or find the values of f_α – this is all been numerically approximated by **ggdensity** via the previously discussed "numerical integration" method. This represents a simple, powerful tool for visualizing and understanding the probabilistic behavior of arbitrary densities, so long as their support is roughly known.

Beyond the utility of visualizing HDRs of theoretical densities, geom_hdr_fun() and geom_hdr_lines_fun() can be used to plot HDRs for arbitrary parametric estimates of f. We discuss this at the end of the following section.

5 Further examples

We conclude with a series of more advanced examples that illustrate the flexibility and power of **ggdensity** through more complicated use-cases.

5.1 Comparing populations

Since geom_hdr() and geom_hdr_lines() use transparency (the alpha aesthetic) to communicate probability, color remains available to communicate group membership in the context of more than one population via either the fill or color aesthetics. This allows for easy comparison of multiple bivariate populations via their HDRs. In Figure 11, we use this strategy to compare the relationship between flipper length and bill length for different species of penguins using the popular Palmer penguins dataset (Horst et al., 2020). In this case geom_hdr_lines() is used to reduce overplotting.

As discussed previously, **ggdensity** provides several nonparametric and parametric estimators to compute the HDRs. Figure 11 assumes a bivariate normal distribution, expressed by setting method = "mvnorm" in geom_hdr_lines(). This implies that each group's HDRs are elliptical and the resulting visualization is a useful approximation of the true distributions. With it we can easily see the general location of each of the groups and that all have similar covariance structures. These details can be obscured when more flexible non-parametric HDR estimators are used, especially when sample sizes are small.

```
ggplot(penguins, aes(flipper_length_mm, bill_length_mm, fill = species)) +
  geom_hdr_lines(aes(color = species), method = "mvnorm") +
  geom_jitter(shape = 21)
```


Figure 11: Using geom_hdr_lines() with a color aesthetic can be used to reduce overplotting when visualizing the HDRs for different subgroups of data.

5.2 HDRs and goodness of fit

It can be useful to combine geom_hdr() and geom_hdr_lines() to compare different estimators of f. A powerful example is plotting elliptical contour lines corresponding to an estimated normal model on top of filled contours of the KDE, facilitating a visual exploration of goodness of fit. We have included two examples of this strategy in Figure 12. The left graphic illustrates bill length versus flipper length for the Chinstrap penguins from Figure 11. Notice that the filled contours generally match the contour lines, providing visual evidence towards the validity of an assumption of normality. By contrast, the right graphic explores the relationship between two measurements from a dataset comparing 178 wines from Forina et al. (1986) exported as wines in **sn** (Azzalini, 2022). The filled contours do not coincide with the contour lines – the nonparametric estimate of the density is visibly more skewed – indicating that a normal approximation might not be appropriate for this data.

```
penguins |>
  filter(species == "Chinstrap") |>
  ggplot(aes(flipper_length_mm, bill_length_mm)) +
  geom_hdr() +
  geom_hdr_lines(color = "red", method = "mvnorm") +
  geom_jitter(color = "red")


ggplot(wines, aes(uronic, malic)) +
  geom_hdr() +
  geom_hdr_lines(method = "mvnorm", color = "red") +
  geom_jitter(color = "red")
```


Figure 12: Normality can be visually assessed by layering the HDRs of a KDE (black) with that of a parametrically estimated bivariate normal (red), here illustrated with Palmer penguin data (left) and wines data (right). The points in each plot have been jittered due to rounding in the data, notice that this leads to small inconsistencies between the plotted data and HDRs.

This strategy can be extended to evaluating goodness of fit for arbitrary parametric models via combining `geom_hdr()` and `geom_hdr_lines_fun()`, following the strategy outlined in HDRs for arbitrary parametric models.

5.3 Other related geoms

ggdensity also includes functions `geom_hdr_points()` and `geom_hdr_rug()` for alternative methods of visualizing HDRs[3]. These are illustrated in Figure 13, in which we visualize the old faithful dataset (Azzalini and Bowman, 1990). The left image displays the standard visualization of HDRs from `geom_hdr()`. The graphic in the middle, created by `geom_hdr_points()`, displays the data itself with points colored by their HDR membership–this can be useful in situations where overplotting is a concern. The plot on the right presents the original data with the estimated marginal HDRs via `geom_hdr_rug()`. Note that the scale is the same across all of the plots, with the 50%, 80%, 95%, and 99% HDRs being visualized by default.

```
p <- ggplot(faithful, aes(eruptions, waiting))


p + geom_hdr()
p + geom_hdr_points()
p + geom_hdr_rug()
```

---

[3]The previously mentioned **gghdr** includes similar tools, we discuss this further in Discussion and future directions.

Figure 13: `geom_hdr_points()` and `geom_hdr_rug()` can provide more insight into bivariate scatter-plots, as seen here with the faithful dataset.

It is important to note that `geom_hdr_rug()` can also be used when only an x or y aesthetic is provided. This is illustrated in Figure 14 where the KDE of eruption duration is visualized alongside its estimated HDRs. In this graphic we have chosen to communicate the HDRs via colors – in some cases we have found this to be preferable when using `geom_hdr_rug()`.

```
ggplot(faithful, aes(eruptions)) +
  geom_density() +
  geom_hdr_rug(aes(fill = after_stat(probs)), length = unit(.05, "npc"), alpha = 1) +
  scale_fill_viridis_d(option = "magma", begin = .8, end = 0)
```


Figure 14: `geom_hdr_rug()` can also improve the visualization of univariate densities.

5.4 HDRs for arbitrary parametric models

Historically, there has been much focus on contour estimation based on non-parametric estimates of f, typically KDEs. To our knowledge, there has been relatively little focus on parametric estimation of HDRs. If a probability model is specified, estimated HDRs are simple to derive from $\hat{f}_{\text{MLE}}$, the density's maximum likelihood estimator (MLE). This allows for the visualization of a much larger class of HDR estimators than those built into `geom_hdr()`; users can specify and estimate arbitrary parametric models and provide the resulting density estimate to `geom_hdr_fun()`.

We include an example of HDRs corresponding to a custom estimated parametric density in Figure 15. Here we generated $n = 100$ draws from a bivariate exponential distribution $(X, Y) \sim f(x, y|\boldsymbol{\theta}) = \text{Exp}(\boldsymbol{\theta})$, estimated $\boldsymbol{\theta}$ with its MLE, and passed the resulting estimate $\hat{f}(x, y) = f(x, y|\hat{\boldsymbol{\theta}})$ to `geom_hdr_fun()` via the `fun` argument.

```
set.seed(1)
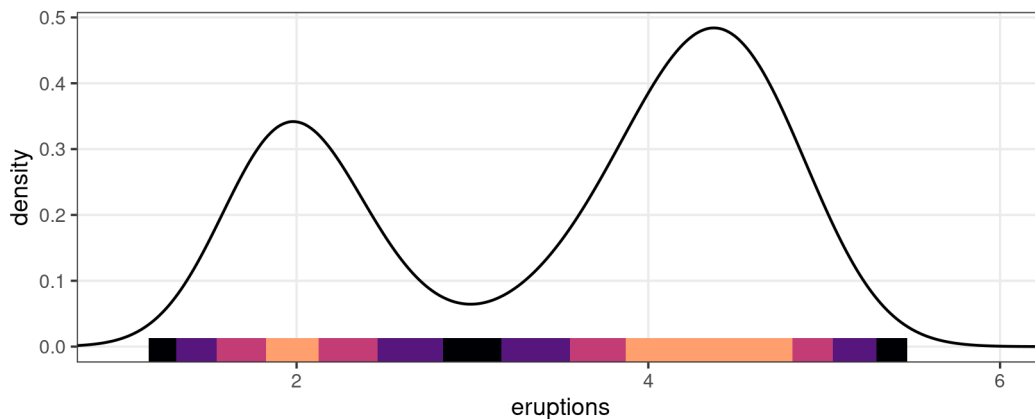
df <- data.frame(x = rexp(100, 1), y = rexp(100, 1))

# pdf for parametric density estimate
f <- function(x, y, lambda) dexp(x, lambda[1]) * dexp(y, lambda[2])

# estimate parameters governing joint pdf
lambda_hat <- apply(df, 2, mean)

ggplot(df, aes(x, y)) +
  geom_hdr_fun(fun = f, args = list(lambda = lambda_hat)) +
  geom_point(fill = "lightgreen", shape = 21) +
  coord_fixed() +
  scale_x_continuous(limits = c(0, 7)) +
  scale_y_continuous(limits = c(0, 7))
```


Figure 15: Plotting HDRs of specified distributions can be achieved with `geom_hdr_fun()`.

6 Discussion and future directions

As described in Finding HDRs in two dimensions, **ggdensity** approximates sample HDRs via the numerical integration approach, unlike other software such as **hdrcde** and **gghdr** that both rely on the quantile approach (Hyndman, 1996). In practice the two approaches perform similarly, although there are several technical ways in which they differ. One example is that HDRs estimated via sample quantiles are not guaranteed to contain a certain proportion of estimated density, even if $p_{\hat{f}}(c)$ is strictly decreasing. In other words, there is no guarantee that $\int_{\hat{\mathcal{R}}_\alpha} \hat{f}(\mathbf{x}) \, d\mathbf{x} = 1 - \alpha$ when $\hat{\mathcal{R}}_\alpha$ is determined using $\hat{f}_\alpha$ computed via the quantile method. Additionally, as the quantile method requires a sample from f it is not possible to calculate HDRs from arbitrary densities as in `geom_hdr_fun()`. Historically, the density quantile approach has been favored due to computational limitations associated with numerical integration (Hyndman, 1996). However, with modern computing power, this is not a concern anymore – we have found **ggdensity** to be very performant.

With both **ggdensity** and **gghdr** being extensions to **ggplot2** for visualizing HDRs there is overlap in their capabilities. There are analogs to `geom_hdr_rug()` and `geom_hdr_points()` implemented as `gghdr::geom_hdr_rug()` and the helper function `gghdr::hdr_bin()`, respectively. From the user's perspective these implementations are similar, with **ggdensity** offering HDRs estimated via different methods. A more serious distinction between the two is that **gghdr** does not provide a way to plot *bivariate* HDRs in a way similar to `geom_hdr()` or `geom_hdr_lines()`. At present, **ggdensity** is the only package that facilitates the visualization of bivariate HDR contours with **ggplot2**.

Another important difference is that both **gghdr** and **hdrcde** implement visualizations of conditional HDRs, something we plan on implementing in **ggdensity** in the future. These allow users to make visuals similar to regression-style modeling bands. We also plan on extending **ggdensity**'s

capabilities to plot univariate HDRs, implementing something similar to `ggdensity::geom_hdr_rug()` for the main plotting window; this will result in a tool similar to `gghdr::geom_hdr_boxplot()`. This future feature also bears resemblance to the `stat_slabinterval()` family from **ggdist**, another **ggplot2** extension for visualizing densities and their estimates (Kay, 2023). Finally, we also look to implement more density estimators available via the `method` argument, for example skew-normal and mixture models.

References

A. Azzalini. sn: The Skew-Normal and Related Distributions Such as the Skew-t and the SUN, Mar. 2022. URL https://CRAN.R-project.org/package=sn. [p231]

A. Azzalini and A. W. Bowman. A Look at Some Data on the Old Faithful Geyser. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 39(3):357–365, 1990. ISSN 0035-9254. doi: 10.2307/2347385. URL https://www.jstor.org/stable/2347385. Publisher: [Wiley, Royal Statistical Society]. [p232]

B. Cadre. Kernel estimation of density level sets. *Journal of Multivariate Analysis*, 97(4):999–1023, Apr. 2006. ISSN 0047-259X. doi: 10.1016/j.jmva.2005.05.004. URL https://www.sciencedirect.com/science/article/pii/S0047259X05000825. [p222]

A. Cameron and T. van den Brand. *geomtextpath: Curved Text in 'ggplot2'*, 2022. URL https://CRAN.R-project.org/package=geomtextpath. R package version 0.1.0. [p222]

M. Forina, C. Armanino, M. Castino, and M. Ubigli. Multivariate data analysis as a discriminating method of the origin of wines. *VITIS - Journal of Grapevine Research*, 25(3):189–189, 1986. ISSN 2367-4156. doi: 10.5073/vitis.1986.25.189-201. URL https://ojs.openagrar.de/index.php/VITIS/article/view/5950. Number: 3. [p231]

J. A. Hartigan. Estimation of a Convex Density Contour in Two Dimensions. *Journal of the American Statistical Association*, 82(397):267–270, Mar. 1987. ISSN 0162-1459. doi: 10.1080/01621459.1987.10478428. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1987.10478428. Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1987.10478428. [p222]

A. Horst, A. Hill, and K. Gorman. palmerpenguins: Palmer Archipelago (Antarctica) Penguin Data, July 2020. URL https://CRAN.R-project.org/package=palmerpenguins. [p231]

R. Hyndman, J. Einbeck, M. Wand, S. Carrignon, and F. Cheng. hdrcde: Highest Density Regions and Conditional Density Estimation, Jan. 2021. URL https://CRAN.R-project.org/package=hdrcde. [p227]

R. J. Hyndman. Computing and Graphing Highest Density Regions. *The American Statistician*, 50(2):120–126, 1996. ISSN 0003-1305. doi: 10.2307/2684423. URL https://www.jstor.org/stable/2684423. Publisher: [American Statistical Association, Taylor & Francis, Ltd.]. [p220, 222, 227, 234]

M. Kay. *ggdist: Visualizations of Distributions and Uncertainty*, 2023. URL https://mjskay.github.io/ggdist/. R package version 3.2.1. [p235]

D. W. Muller and G. Sawitzki. Excess Mass Estimates and Tests for Multimodality. *Journal of the American Statistical Association*, 86(415):738–746, 1991. ISSN 0162-1459. doi: 10.2307/2290406. URL https://www.jstor.org/stable/2290406. Publisher: [American Statistical Association, Taylor & Francis, Ltd.]. [p222]

M. O'Hara-Wild, S. Pearce, R. Nakagawara, S. Gupta, D. Vanichkina, E. Tanaka, T. Fung, and R. Hyndman. gghdr: Visualisation of Highest Density Regions in 'ggplot2', Feb. 2022. URL https://CRAN.R-project.org/package=gghdr. [p227]

W. Polonik. Measuring Mass Concentrations and Estimating Density Contour Clusters-An Excess Mass Approach. *The Annals of Statistics*, 23(3):855–881, June 1995. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1176324626. URL https://projecteuclid.org/journals/annals-of-statistics/volume-23/issue-3/Measuring-Mass-Concentrations-and-Estimating-Density-Contour-Clusters-An-Excess/10.1214/aos/1176324626.full. Publisher: Institute of Mathematical Statistics. [p222]

P. Rigollet and R. Vert. Optimal rates for plug-in estimators of density level sets. *Bernoulli*, 15(4):1154–1178, Nov. 2009. ISSN 1350-7265. doi: 10.3150/09-BEJ184. URL https://projecteuclid.org/journals/bernoulli/volume-15/issue-4/Optimal-rates-for-plug-in-estimators-of-density-level-sets/10.3150/09-BEJ184.full. Publisher: Bernoulli Society for Mathematical Statistics and Probability. [p222]

D. W. Scott. *Multivariate density estimation: theory, practice, and visualization / David W. Scott.* Wiley series in probability and mathematical statistics. Wiley, New York, 1992. ISBN 978-0-471-54770-9. [p220]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S.* Springer, New York, fourth edition, 2002. URL https://www.stats.ox.ac.uk/pub/MASS4/. ISBN 0-387-95457-0. [p220]

H. Wickham. *ggplot2*. Springer, New York, NY, 2009. ISBN 978-0-387-98140-6 978-0-387-98141-3. doi: 10.1007/978-0-387-98141-3. URL http://link.springer.com/10.1007/978-0-387-98141-3. [p220]

C. O. Wilke and T. L. Pedersen. *isoband: Generate Isolines and Isobands from Regularly Spaced Elevation Grids*, 2021. URL https://CRAN.R-project.org/package=isoband. R package version 0.2.5. [p221, 226]

L. Wilkinson. *The Grammar of Graphics.* Statistics and Computing. Springer-Verlag, New York, 2005. ISBN 978-0-387-24544-7. doi: 10.1007/0-387-28695-0. URL http://link.springer.com/10.1007/0-387-28695-0. [p220]

James Otto
Baylor University
One Bear Place #97140
Waco, TX 77005
https://orcid.org/0000-0002-0665-2515
jamesotto852@gmail.com

David Kahle
Baylor University
One Bear Place #97140
Waco, TX 77005
https://orcid.org/0000-0002-9999-1558
david_kahle@baylor.edu

Three-Way Correspondence Analysis in R

by Rosaria Lombardo, Michel van de Velden, and Eric J. Beh

Abstract Three-way correspondence analysis is a suitable multivariate method for visualising the association in three-way categorical data, modelling the global dependence, or reducing dimensionality. This paper provides a description of an R package for performing three-way correspondence analysis: CA3variants. The functions in this package allow the analyst to perform several variations of this analysis, depending on the research question being posed and/or the properties underlying the data. Users can opt for the classical (symmetrical) approach or the non-symmetric variant - the latter is particularly useful if one of the three categorical variables is treated as a response variable. In addition, to perform the necessary three-way decompositions, a Tucker3 and a trivariate moment decomposition (using orthogonal polynomials) can be utilized. The Tucker3 method of decomposition can be used when one or more of the categorical variables is nominal while for ordinal variables the trivariate moment decomposition can be used. The package also provides a function that can be used to choose the model dimensionality.

1 Introduction

In many applications, one encounters problems where detecting and describing the association between three categorical variables is of interest. For example, one may wish to analyse animal counts stratified by species-by-site-by-time, treatment success stratified by cure-by-therapy-by-hospital, customer satisfaction-by-service's quality-by-country, or two interacting genes in expression under the genotypes of another gene. One method specifically designed for analysing such data is three-way correspondence analysis (Carlier and Kroonenberg, 1996). For this method of analysis, a three-way contingency table is decomposed in such a way that the maximum amount of association is reflected in a low-dimensional display. Depending on the underlying data, and the research questions being asked, there are various ways to quantify and decompose the association in the table, generate a visual display of the association and calculate the accompanying numerical summaries. Hence, several variants of three-way correspondence analysis exist. Common among all the variants that we describe below is the emphasis that is placed on data exploration through the visualization of the associations.

There exists a sizable body of literature that examines the various theoretical properties and extensions of three-way correspondence analysis. For example, Kroonenberg (1989), Carlier and Kroonenberg (1996), Kroonenberg (2008, Chap. 17), Beh and Lombardo (2014, Chap. 11) and Lombardo et al. (2021) discuss a wide range of issues concerned with this technique. However, there also appears to be only a few applications that use these techniques (Carlier and Kroonenberg, 1998; van Herk and van de Velden, 2007; Lombardo et al., 2019). One reason for the lack of applications could be the absence of R software packages to perform three-way correspondence analysis.

In this paper, we introduce **CA3variants**, a comprehensive R package that allows researchers to apply variants of three-way correspondence analysis. In Section 2.2, we introduce the notation that we adopt as well as two key measures of association - Pearson's three-way phi-squared statistic and Marcotorchino's three-way index. These measures lie at the core of the three-way correspondence analysis variants that we describe below. In Section 2.3 we present three methods for decomposing a three-way contingency table, with a particular focus on the appropriateness of the different variants. In Section 2.4 we show how the two association measures above, can be partitioned in bivariate and trivariate association terms, and how can be used to define variants of three-way correspondence analysis, and we consider specific issues concerned with the visualization and selection of the dimensionality of the three-way correspondence analysis solution. In Section 2.5, we briefly review the software that is currently available for three-way analyses. In Section 2.6, we introduce our three-way correspondence analysis package, **CA3variants**, and illustrate its features and application through some illustrative examples. Some final comments are left for Section 2.7.

2 Measures of three-way association

Three-way correspondence analysis provides a numerical and graphical summary of how categories and variables are related to one another. Rather than only considering the bivariate associations between pairs of variables, three-way correspondence analysis also considers the trivariate associations (Lombardo et al., 2021).

When performing three way correspondence analysis, the dependence structure of the three categorical variables that are cross-classified to form a contingency table is analysed by considering

an appropriate measure of association. This measure can then be partitioned to reveal more detail about the nature of the association that exists between the variables. Two measures of association are implemented in the **CA3variants** package: Pearson's phi-squared statistic and Marcotorchino's index. Pearson's three-way statistic is appropriate when studying deviations from three-way independence and when the variables are symmetrically associated, while Marcotorchino's three-way index is a more suitable choice when the variables are not symmetrically associated. Depending on the choice of the measure of association used, the appropriately scaled three-way table can be decomposed into (low-dimensional) components for each of the variables. Before discussing these three-way decomposition methods, we first introduce the notation used throughout this paper. We then provide a brief description of Pearson's phi-squared statistic and Marcotorchino's index.

2.1 Notation

Suppose we have data from a sample of n subjects on three categorical variables. Such data can be represented by a three-way contingency table consisting of I rows, J columns and K tubes, where each cell value represents the count within an intersection of the levels of each of the three variables.

Denote $\underline{\mathbf{N}}$ to be the contingency table of order $I \times J \times K$ belonging to the space $\Re^{I \times J \times K}$, subscripted by i, j and k for $i = 1, \ldots, I$, $j = 1, \ldots, J$, $k = 1, \ldots, K$, whose (i, j, k)th term is n_{ijk}, while $\underline{\mathbf{P}}$ is the table of joint relative frequencies of $\underline{\mathbf{N}}$ whose (i, j, k)th term is $p_{ijk} = n_{ijk}/n$, such that $\sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} p_{ijk} = 1$. Define $p_{i\bullet\bullet} = \sum_{j=1}^{J} \sum_{k=1}^{K} p_{ijk}$, $p_{\bullet j \bullet} = \sum_{i=1}^{I} \sum_{k=1}^{K} p_{ijk}$, $p_{\bullet\bullet k} = \sum_{i=1}^{I} \sum_{j=1}^{J} p_{ijk}$, $p_{ij\bullet} = \sum_{k=1}^{K} p_{ijk}$, $p_{i\bullet k} = \sum_{j=1}^{J} p_{ijk}$ and $p_{\bullet jk} = \sum_{i=1}^{I} p_{ijk}$ to be the univariate and bivariate marginal relative frequencies of the three-way contingency table. In addition, define $\mathbf{I}_I$ to be the identity matrix of order $I \times I$ in the space $\Re^I$, and let $\mathbf{D}_I$, $\mathbf{D}_J$, $\mathbf{D}_K$ be the diagonal matrices containing the univariate marginal relative frequencies in $\Re^I$, $\Re^J$ and $\Re^K$ whose general term is $p_{i\bullet\bullet}$, $p_{\bullet j \bullet}$ and $p_{\bullet\bullet k}$, respectively.

2.2 Pearson's three-way statistic

When the association between the categorical variables of a three-way contingency table, $\underline{\mathbf{N}}$, is considered to be symmetric, we can analyse the strength of this association using Pearson's three-way phi-squared statistic

$$
\begin{aligned}
\Phi^2 &= \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} p_{i\bullet\bullet} p_{\bullet j \bullet} p_{\bullet\bullet k} \left(\frac{p_{ijk} - p_{i\bullet\bullet} p_{\bullet j \bullet} p_{\bullet\bullet k}}{p_{i\bullet\bullet} p_{\bullet j \bullet} p_{\bullet\bullet k}} \right)^2 \\
&= \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} p_{i\bullet\bullet} p_{\bullet j \bullet} p_{\bullet\bullet k} \left(\frac{p_{ijk}}{p_{i\bullet\bullet} p_{\bullet j \bullet} p_{\bullet\bullet k}} - 1 \right)^2 \\
&= \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} p_{i\bullet\bullet} p_{\bullet j \bullet} p_{\bullet\bullet k} \left(\pi_{P_{ijk}} \right)^2 .
\end{aligned}
\tag{1}
$$

The symmetric nature of this measure implies that the three variables are all treated as predictor variables. That is, none are deemed to be dependent on the outcome of any other variable being studied. It can be shown that, under the independence assumption, Φ^2 can be partitioned as

$$
\begin{aligned}
\Phi^2 &= \sum_{i=1}^{I} \sum_{j=1}^{J} p_{i\bullet\bullet} p_{\bullet j \bullet} \left(\frac{p_{ij\bullet} - p_{i\bullet\bullet} p_{\bullet j \bullet}}{p_{i\bullet\bullet} p_{\bullet j \bullet}} \right)^2 + \sum_{i=1}^{I} \sum_{k=1}^{K} p_{i\bullet\bullet} p_{\bullet\bullet k} \left(\frac{p_{i\bullet k} - p_{i\bullet\bullet} p_{\bullet\bullet k}}{p_{i\bullet\bullet} p_{\bullet\bullet k}} \right)^2 \\
&+ \sum_{j=1}^{J} \sum_{k=1}^{K} p_{\bullet j \bullet} p_{\bullet\bullet k} \left(\frac{p_{\bullet jk} - p_{\bullet j \bullet} p_{\bullet\bullet k}}{p_{\bullet j \bullet} p_{\bullet\bullet k}} \right)^2 + \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} p_{i\bullet\bullet} p_{\bullet j \bullet} p_{\bullet\bullet k} \left(\frac{p_{ijk} -_{\alpha} p_{ijk}}{p_{i\bullet\bullet} p_{\bullet j \bullet} p_{\bullet\bullet k}} \right)^2 ,
\end{aligned}
\tag{2}
$$

where

$$
{}_\alpha \hat{p}_{ijk} = \hat{p}_{ij\bullet} \hat{p}_{\bullet\bullet k} + \hat{p}_{i\bullet k} \hat{p}_{\bullet j \bullet} + \hat{p}_{\bullet jk} \hat{p}_{i\bullet\bullet} - 2\hat{p}_{i\bullet\bullet} \hat{p}_{\bullet j \bullet} \hat{p}_{\bullet\bullet k} .
\tag{3}
$$

For further details see Carlier and Kroonenberg (1996) and Lombardo et al. (2020). Briefly, we get

$$
\Phi^2 = \Phi_{IJ}^2 + \Phi_{IK}^2 + \Phi_{JK}^2 + \Phi_{IJK}^2 .
\tag{4}
$$

Observe that this partition also concerns Pearson's chi-squared statistic, X^2, (Lancaster, 1951; Lombardo et al., 2020) obtained by multiplying each of the terms of phi-squared in equation (4) by the sample size, n. Indeed, Pearson's chi-squared statistic is well established for testing association

between variables in contingency tables. Hence, deviations from three-way independence can be orthogonally partitioned into three deviations from independence (for each of the two-way tables formed by summing over each variable of the three-way contingency table) and a three-way association term, as it will be shown in Section 2.6.1. This partition has been extensively discussed by Carlier and Kroonenberg (1996) and more recently by Kroonenberg (2008, Chap. 17), Loisel and Takane (2016) and Lombardo et al. (2020).

2.3 Marcotorchino's three-way index

If the three categorical variables are non-symmetrically associated, or if one is interested in exploring an non-symmetric association between the variables, a more appropriate measure is the three-way Marcotorchino index. This index is defined by

$$\tau_M \quad = \quad \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} p_{\bullet j\bullet}p_{\bullet\bullet k}\left(\frac{p_{ijk}}{p_{\bullet j\bullet}p_{\bullet\bullet k}} - p_{i\bullet\bullet}\right)^2}{1 - \sum_{i=1}^{I} p_{i\bullet\bullet}^2}. \tag{5}$$

See, for example, Marcotorchino (1984a,b), Lombardo et al. (1996), Beh et al. (2007), Beh and Lombardo (2014, Section 11.4.2) and Beh and Lombardo (2021b, Section 7.5). Since the denominator of equation (5) is independent on the cell values of **N**, the numerator of the Marcotorchino index suffices as a measure of association when performing three-way correspondence analysis. This numerator measures the absolute increase in predictability of the response variable, given the predictor variables (Marcotorchino, 1985; Lombardo et al., 1996). Like Pearson's three-way phi-squared statistic, Marcortorchino's index is based on deviations from the three-way independence model. Without loss of generality, assume that the row variable is considered to be dependent on the column and tube variables. In doing so, the numerator of equation (5), which we shall simply refer to as Marcotorchino's $\tau_{M_{num}}$ statistic, is equal to

$$\tau_{M_{num}} \quad = \quad \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} p_{\bullet j\bullet}p_{\bullet\bullet k}\left(\frac{p_{ijk}}{p_{\bullet j\bullet}p_{\bullet\bullet k}} - p_{i\bullet\bullet}\right)^2 \tag{6}$$

$$= \quad \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} p_{\bullet j\bullet}p_{\bullet\bullet k}\left(\pi_{M_{ijk}}\right)^2.$$

As in the symmetric case, an additive orthogonal partition of $\tau_{M_{num}}$ exists and is given by

$$\tau_{M_{num}} \quad = \quad \sum_{i=1}^{I}\sum_{j=1}^{J} p_{\bullet j\bullet}\left(\frac{p_{ij\bullet}}{p_{\bullet j\bullet}} - p_{i\bullet\bullet}\right)^2 + \sum_{i=1}^{I}\sum_{k=1}^{K} p_{\bullet\bullet k}\left(\frac{p_{i\bullet k}}{p_{\bullet\bullet k}} - p_{i\bullet\bullet}\right)^2$$

$$+ \frac{1}{I}\sum_{j=1}^{J}\sum_{k=1}^{K} p_{\bullet j\bullet}p_{\bullet\bullet k}\left(\frac{p_{\bullet jk} - p_{\bullet j\bullet}p_{\bullet\bullet k}}{p_{\bullet j\bullet}p_{\bullet\bullet k}}\right)^2$$

$$+ \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} p_{\bullet j\bullet}p_{\bullet\bullet k}\left(\frac{p_{ijk} - _\alpha p_{ijk}}{p_{\bullet j\bullet}p_{\bullet\bullet k}}\right)^2, \tag{7}$$

where

$$_\alpha p_{ijk} = \hat{p}_{ij\bullet}\hat{p}_{\bullet\bullet k} + \hat{p}_{i\bullet k}\hat{p}_{\bullet j\bullet} + \frac{\hat{p}_{\bullet jk}}{I} - \hat{p}_{i\bullet\bullet}\hat{p}_{\bullet j\bullet}\hat{p}_{\bullet\bullet k} - \hat{p}_{\bullet j\bullet}\frac{\hat{p}_{\bullet\bullet k}}{I}.$$

The partition of $\tau_{M_{num}}$ may be more simply expressed as

$$\tau_{M_{num}} = \tau_{IJ} + \tau_{IK} + \tau_{JK} + \tau_{IJK}. \tag{8}$$

Hence, like Pearson's three-way phi-squared statistic, $\tau_{M_{num}}$ (and hence the total predictability measure τ_M) is partitioned into four additive terms. The first three of these terms reflect the two-way associations and the fourth term reflects the three-way association. The first two bivariate terms of equation (8) are equal to the numerators of the Goodman-Kruskal indices (Goodman and Kruskal, 1954) between the response (row) variable and each of the two predictor (column and tube) variables, respectively. These terms are also equal to the inertias of the marginal two-way tables in classical two-way non-symmetric correspondence analysis (Lauro and D'Ambra, 1984; D'Ambra and Lauro, 1989; Kroonenberg and Lombardo, 1999; Takane and Jung, 2008). The third bivariate term of (8) is (up

to the constant $1/I$) equal to Φ^2_{JK}, which is Pearson's phi-squared statistic for the $J \times K$ contingency table formed by aggregating over the row categories. This term can be seen as a measure of the symmetric association between the two predictor variables. Finally, the last term of equation (8) is a measure of the trivariate association between the variables. Beh et al. (2007) showed that the test statistic associated with Marcotorchino's three way index is the generalization of the C-statistic (Light and Margolin, 1971), referred to here as the C_M-statistic, and is defined by

$$C_M = (n-1)(I-1)\tau_M \sim \chi^2_{\alpha, df}. \tag{9}$$

Therefore, for both Pearson's three-way chi-squared statistic and Marcotorchino's three-way τ_M statistic, under the null hypothesis of complete independence, each term of the partition is a chi-squared random variable. For further details see Light and Margolin (1971), Beh et al. (2007), Beh and Lombardo (2014, Section 11.5.2) and Beh and Lombardo (2021b, Section 7.5.2).

3 Decomposing three-way tables

The choice of which measure of association to use should be made based on the data at hand and the research question under investigation. Depending on the choice, an appropriately scaled matrix can be constructed. Three-way correspondence analysis can then be performed and involves fitting a model to the data. In particular, low-dimensional component matrices as well as a core matrix that links the different components, are fitted to the data in such a way that the sum-of-squares of the deviations between the low-dimensional approximation and the original table is as small as possible.

Several decomposition models have been proposed in the literature for three-way contingency tables. In the **CA3variants** package three types of decomposition are implemented. They are the Tucker3 model (Tucker, 1963; Kroonenberg, 1983, 2008; Kiers et al., 1992) for when all three variables are nominal, the trivariate moment decomposition (Lombardo et al., 2016b, 2021) for when all three variables are ordinal, and a hybrid decomposition for a mix of nominal and ordinal categorical variables (Lombardo and Beh, 2017). In the following subsections, we briefly review these decomposition methods and how they apply to the different variants of three-way correspondence analysis.

3.1 Tucker3 decomposition for three-way tables

For the Tucker3 decomposition, a three-way matrix $\underline{\mathbf{X}}$ with elements x_{ijk} is decomposed such that

$$x_{ijk} = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr} + e_{ijk},$$

where P, Q and R ($P \leq I, Q \leq J, R \leq K$) are the fixed number of the components corresponding to the row, column and tube variables, respectively. The a_{ip}, b_{jp} and c_{kp} values are elements of the column matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, respectively, and give component loadings for the row, column and tube variables, while g_{pqr} is an element of the $P \times Q \times R$ core array. The term e_{ijk} is the error of approximation. By "flattening" the three-way matrix $\underline{\mathbf{X}}$ – for example, by concatenating the K tubes of $\underline{\mathbf{X}}$ – we can write the Tucker3 decomposition in matrix form by

$$\text{Tucker3}(\mathbf{X}) = \mathbf{AG}\left(\mathbf{B}^T \otimes \mathbf{C}^T\right) + \mathbf{E}, \tag{10}$$

where $\mathbf{X}$ and $\mathbf{G}$ are, respectively, the $I \times JK$ matrix of (flattened) data values and the $P \times QR$ matrix of core elements.

The solution to $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{G}$ is obtained by minimizing the sum-of-squares of the elements of $\mathbf{E}$ (matrix of the errors of approximation) using an alternating least-squares algorithm. The general framework of the algorithm that **CA3variants** uses is based on the Tuckals3 alternating least squares algorithm discusssed by Kroonenberg and Leeuw (1980) and Kroonenberg (1983, 1994).

Symmetric three-way correspondence analysis

For symmetric three-way correspondence analysis, the elements of Pearson's three-way phi-squared statistic are decomposed using a Tucker3 decomposition. In particular, the Tucker3 decomposition is

applied to the appropriately scaled three-way array $\underline{\mathbf{\Pi}}_P$ with elements

$$\pi_{P_{ijk}} = \frac{p_{ijk}}{p_{i\bullet\bullet}p_{\bullet j\bullet}p_{\bullet\bullet k}} - 1 \, , \tag{11}$$

where the component matrices, $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are constrained to be orthonormal with respect to the diagonal matrices of univariate marginal relative frequencies such that

$$\mathbf{A}^T\mathbf{D}_I\mathbf{A} = \mathbf{I}_P \, , \quad \mathbf{B}^T\mathbf{D}_J\mathbf{B} = \mathbf{I}_Q \, , \quad \text{and} \quad \mathbf{C}^T\mathbf{D}_K\mathbf{C} = \mathbf{I}_R \, . \tag{12}$$

Note that the weighted sum-of-squares of the elements of $\pi_{P_{ijk}}$ is equal to Pearson's three-way phi-squared statistic; see equation (1). In other words, the symmetric variant of three-way correspondence analysis amounts to minimizing the weighted squared differences between the standardized deviations of independence in the three-way table with the approximated values using the Tucker3 model. That is:

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} p_{i\bullet\bullet}p_{\bullet j\bullet}p_{\bullet\bullet k}\left(\pi_{P_{ijk}} - \widehat{\pi}_{P_{ijk}}\right)^2 \, ,$$

is minimized where, for some value of P, Q and R

$$\widehat{\pi}_{P_{ijk}} = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr}a_{ip}b_{jq}c_{kr} \, .$$

The constraints of equation (12) are similar to the constraints used in the traditional approach to simple (two-way) correspondence analysis. Consequently, symmetric three-way correspondence analysis can be seen as a direct extension of the traditional two-way correspondence analysis approach. For more details see, for example, Carlier and Kroonenberg (1996).

Non-symmetric three-way correspondence analysis

For non-symmetric three-way correspondence analysis, one variable needs to be selected as the response variable. In the following discussion we choose, without loss of generality, the first (row) variable to serve as the response variable. When performing non-symmetric three-way correspondence analysis, we use the Tucker3 decomposition to decompose Marcotorchino's three-way $\tau_{M_{num}}$ statistic defined by equation (6). Let $\underline{\mathbf{\Pi}}_M$ represents the three-way matrix with elements

$$\pi_{M_{ijk}} = \frac{p_{ijk}}{p_{\bullet j\bullet}p_{\bullet\bullet k}} - p_{i\bullet\bullet} \, . \tag{13}$$

Non-symmetric three-way correspondence analysis is then performed by applying the Tucker3 decomposition to $\underline{\mathbf{\Pi}}_M$ where the components contained in the row, column and tube matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, are constrained to be orthornormal with respect to the weight matrices $\mathbf{I}_I$, $\mathbf{D}_J$, $\mathbf{D}_K$. That is,

$$\mathbf{A}^T\mathbf{A} = \mathbf{I}_P \, , \quad \mathbf{B}^T\mathbf{D}_J\mathbf{B} = \mathbf{I}_Q \, , \quad \text{and} \quad \mathbf{C}^T\mathbf{D}_K\mathbf{C} = \mathbf{I}_R \, .$$

Note that, for the decomposition of $\underline{\mathbf{\Pi}}_M$, these constraints ensure that the weighted quadratic norm of the low-dimensional approximation $\underline{\widehat{\mathbf{\Pi}}}_M$, can be written as

$$\|\underline{\widehat{\mathbf{\Pi}}}_M\|^2 = \tau_{M_{num}} = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} g_{pqr}^2 \, .$$

3.2 Trivariate moment decomposition

Rather than considering component matrices as the Tucker3 decomposition does, the trivariate moment decomposition is based on column matrices consisting of orthogonal polynomials. The decomposition was first proposed by Beh (1998b, Chap. 7) and has since been described by, for example, Beh and Davy (1998), Lombardo et al. (2016b) and Lombardo et al. (2021, eq. 10), as an alternative method of three-way decomposition. It is particularly useful when a variable consists of ordered categories, either increasing or decreasing. The decomposition can be applied to either $\underline{\mathbf{\Pi}}_P$ or $\underline{\mathbf{\Pi}}_M$ and allows the researcher to incorporate the ordinality by replacing the Tucker3 components with the orthogonal polynomials for the ordinal variable. These polynomials are typically generated using the three-term recurrence formulae of Emerson (1968) who demonstrated their computational efficiency when compared with

the Gram-Schmidt orthogonalization process. Refer to Beh (1997, 1998a,b) and Beh and Lombardo (2021a) for a definition and properties of these polynomials when performing correspondence analysis.

To form the polynomial basis space we generate as many orthogonal polynomials as there are ordered categories. The matrix of row, column and tube orthogonal polynomials is denoted by $\mathcal{A} = \{\alpha_{iu}\}$, (for $i = 1, \ldots, I$ and $u = 0, \ldots, I - 1$), $\mathcal{B} = \{\beta_{jv}\}$ (for $j = 1, \ldots, J$ and $v = 0, \ldots, J - 1$) and $\mathcal{C} = \{\gamma_{kw}\}$ (for $k = 1, \ldots, K$ and $w = 0, \ldots, K - 1$), respectively. Like the Tucker3 components, when a symmetric variant of three-way correspondence analysis is performed, the row polynomials are orthogonal with respect to the marginal relative frequencies $p_{i\bullet\bullet}$, while the column and tube polynomials are orthogonal with respect to $p_{\bullet j\bullet}$ and $p_{\bullet\bullet k}$, respectively. In general, the first polynomial that is computed for each ordered variable of the three-way table represents the *zeroth-order* polynomial and is equal to 1 when in its normalized state. The second polynomial is the *first-order* polynomial and reflects the variation in the linearity of the categories. The third polynomial is the *second-order* orthogonal polynomial and reflects the variation in the dispersion of the categories. Higher-order polynomials represent higher-order moments of the ordered categories. These polynomials have been used extensively in the correspondence analysis literature. For more information, see, for example, Beh (1997, 1998a), Beh and Lombardo (2014, p. 94), Lombardo et al. (2016a) and Beh and Lombardo (2021b, Chap. 4).

When using the trivariate moment decomposition for symmetric and non-symmetric three-way correspondence analysis, the decomposition of the arrays $\underline{\mathbf{\Pi}}_P$ and $\underline{\mathbf{\Pi}}_M$ is defined by replacing the matrices of components $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ (see equation (10)) with their orthogonal polynomial equivalents. In particular, for the non-symmetric case and given the different row weights, we consider $\alpha_u^* = p_{i\bullet\bullet}^{1/2}\alpha_u$, β_v and γ_w such that

$$\pi_{M_{ijk}} = \sum_{u=0}^{U}\sum_{v=0}^{V}\sum_{w=0}^{W}\tilde{z}_{uvw}\alpha_{iu}^*\beta_{jv}\gamma_{kw}. \tag{14}$$

For the decomposition given by equation (14), the row polynomials are weighted such that $\sum_{i=1}^{I}\alpha_{iu}^{*2} = 1$ while the column and tube polynomials are weighted so that $\sum_{j=1}^{J}p_{\bullet j\bullet}\beta_{jv}^2 = 1$ and $\sum_{k=1}^{K}p_{\bullet\bullet k}\gamma_{kw}^2 = 1$, respectively. Note that the indices u, v, w are from 0 to U, V and W (where $U \leq I - 1$, $V \leq J - 1$, $W \leq K - 1$), respectively, and correspond to the orders of the polynomials. The $\tilde{z}_{uvw}$ value in equation (14) is analogous to the core element g_{pqr} in the nominal case and is therefore referred to as the *polynomial core element* and is defined by

$$\tilde{z}_{uvw} = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\pi_{M_{ijk}}p_{\bullet j\bullet}p_{\bullet\bullet k}\alpha_{iu}^*\beta_{jv}\gamma_{kw},$$

and is of order (u, v, w). Such a term has also been referred to as a *generalized correlation*. See, for example, Rayner and Beh (2009), Beh and Lombardo (2014, Chap. 6) and Beh and Lombardo (2021b, Chap. 5). Observe that, unlike the Tucker3 decomposition given by equation (10), the trivariate moment decomposition has a closed form that justifies the absence of the error of approximation in equation (14).

3.3 Hybrid decomposition for nominal and ordinal variables

The hybrid decomposition involves computing Tucker3 components for the nominal variables, and orthogonal polynomials for the ordinal variables (Lombardo and Beh, 2017; Lombardo et al., 2021). Generally for the analysis of three-way contingency tables, we distinguish the following two cases: 1) there are two ordinal variables and one nominal variable, and 2) there are two nominal variables and only one ordinal variable. Suppose we consider the case where we have a three-way contingency table in which the row and column variables are ordinal and the tube variable is nominal. Then the *hybrid decomposition*, for case 1, involves calculating the polynomials for the row and column variables and the Tucker3 components for the nominal tube variable. When the row variable is treated as a response variable, three-way non-symmetric correspondence analysis can be performed using the hybrid decomposition of $\pi_{M_{ijk}}$ such that

$$\pi_{M_{ijk}} = \hat{\pi}_{M_{ijk}} + e_{ijk}$$
$$= \sum_{u=0}^{U}\sum_{v=0}^{V}\sum_{r=1}^{R}z_{uvr}\alpha_{iu}^*\beta_{jv}c_{kr} + e_{ijk}. \tag{15}$$

Here $\boldsymbol{\alpha}_u^*$ and $\boldsymbol{\beta}_v$ are the uth order row and vth order column polynomials, respectively, while c_r is the rth tube (Tucker3) component. The value of z_{uvr} in equation (15) is defined by

$$z_{uvr} = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} \pi_{M_{ijk}} p_{\bullet j \bullet} p_{\bullet \bullet k} \alpha_{iu}^* \beta_{jv} c_{kw} \,,$$

and is referred to as the *hybrid core element* of order (u, v, r). While the number of orthogonal polynomials for the rows and columns should always be equal to the number of categories that define the variable (see Section 2.3.2), the number of Tucker3 components for the tube variable can be smaller ($R \leq K$). A complete orthogonal decomposition is always used when all the three variables are ordered, as it is for equation (14), but is seldom used in practice when the variables are not all ordered. Like the Tucker3 decomposition (see equation (10)) and unlike the trivariate moment decomposition (see equation (14)), the hybrid decomposition given by equation (15) includes the error of approximation, e_{ijk}, because the decomposition no longer has a closed form solution because of the presence of the Tucker3 components.

4 Three-way correspondence analysis variants

Combining the two measures of three-way association described in Section 2.2 with the three methods for decomposing three-way tables outlined in Section 2.3 gives four variants of three-way correspondence analysis:

- Symmetric three-way correspondence analysis: this analysis is based on the partition of Pearson's three-way phi-squared statistic and the Tucker3 decomposition of $\boldsymbol{\Pi}_P$. It executes three-way correspondence analysis by treating all variables symmetrically and corresponds to the analysis described by Carlier and Kroonenberg (1996).

- Non-symmetric three-way correspondence analysis: this corresponds to partitioning Marcotorchino's three-way statistic and applies a Tucker3 decomposition to $\boldsymbol{\Pi}_M$. In this analysis, one of the three variables is treated as a response variable and the other two are treated as predictor variables (Lombardo et al., 1996).

- Ordered symmetric three-way correspondence analysis: for this analysis, either the trivariate moment decomposition (if all variables are ordinal) or the hybrid decomposition (if one or two of the three variables are ordinal) is applied to $\boldsymbol{\Pi}_P$ leading to the partition of Pearson's three-way phi-squared statistic (Lombardo et al., 2021).

- Ordered non-symmetric three-way correspondence analysis: this analysis is based on the trivariate moment decomposition (if all variables are ordinal) or the hybrid decomposition (if one or two of the three variables are ordinal) of $\boldsymbol{\Pi}_M$ and leads to the partition of Marcotorchino's three-way index. Hence, in this analysis one variable is treated as the response variable and the other two are treated as predictor variables.

These four variants of three-way correspondence analysis are incorporated into the **CA3variants** package.

4.1 Visualizing three-way correspondence analysis solutions

Like the traditional approach to two-way correspondence analysis, visualization in three-way correspondence analysis is an important feature and helps to provide a descriptive analysis of the data. To visually display the (symmetric or non-symmetric) association that exists among the variables we consider the *interactive biplot* (Carlier and Kroonenberg, 1996), also called as *nested biplot* by Kroonenberg (2008, p. 441). In the interactive biplot, the categories of one variable, referred to as a *reference variable*, are jointly visualized with all pair-wise combinations of the categories of the other two variables. Hence, depending on the choice of reference variable, we can distinguish three different *interactive coordinates*: row-column, row-tube and column-tube interactive coordinates.

To see how this works, and why the resulting visualizations are indeed biplots, note that all four three-way correspondence analysis variants described in Section 2.4 yield three sets of "coordinates" (one for each variable), as well as an array of core elements that describe the strength of association between these values. Differences between variants can be described in terms of the different measures of association under consideration (i.e., Pearson's three-way phi-squared statistic or Marcotorchino's index), the orthogonalization constraints adopted, or the type of decomposition (i.e., Tucker3, trivariate moment decomposition or hybrid decomposition) used.

Recall that the general form of the Tucker3 decomposition is given by equation (10). As we described above, this matrix formulation is based on a "flattened" version of the three-way matrices

that involves the concatenation of the categories of a variable. In fact, the concatenation of a variable that leads to the $P \times Q \times R$ approximation can be seen in the following three ways

$$\begin{aligned}
\mathbf{X}_{JK,I} &= (\mathbf{B} \otimes \mathbf{C})\, \mathbf{G}_1 \mathbf{A}^T \\
\mathbf{X}_{IK,J} &= (\mathbf{A} \otimes \mathbf{C})\, \mathbf{G}_2 \mathbf{B}^T \\
\mathbf{X}_{IJ,K} &= (\mathbf{A} \otimes \mathbf{B})\, \mathbf{G}_3 \mathbf{C}^T .
\end{aligned} \tag{16}$$

Note that these arrangements have no influence on the approximated values of $\underline{\mathbf{X}}$. The subscripted and flattened $\mathbf{G}$'s indicate that, although their elements are the same, the organization differs between them. Each of the formulations in equation (16) constitutes a biplot. To show this, suppose we consider the decomposition of $\mathbf{X}_{JK,I}$. Then the rows of $(\mathbf{B} \otimes \mathbf{C})\, \mathbf{G}_1$ are the principal coordinates of the pairwise combinations of columns and tubes categories of $\underline{\mathbf{X}}$. Hence, plotting these jointly with the row standard coordinates contained in the rows of $\mathbf{A}$ provides the analyst with a biplot interpretation of the association. For an extensive discussion of biplot interpretations in the context of correspondence analysis see, for example, Greenacre (2010) and Gower et al. (2011, Chapters 7 and 8). For a more general treatment of data visualizations in dimension reduction methods, see Gower et al. (2014).

For each approximation in equation (16), the interactive coordinates can be expressed in either their standard or principal form (whose features are the same of those derived for biplots in the classical approach to correspondence analysis) and so leads to two types of *interactive biplots*:

- For the first type of interactive biplot, we can factorize each equation in such a way that the categories for the non-interactive variable are displayed using standard coordinates so that they are orthonormal with respect to the appropriate metric. Therefore, observing the combination of categories from the other two variables (which constitutes the "interactive" structure of the categories) are defined using principal coordinates (Kroonenberg, 2008, p. 273). Algebraically, this choice simply means that the interactive coordinates are a form of principal coordinates. They are calculated from the Kronecker product of two component matrices (for example, $\mathbf{B}$ and $\mathbf{C}$) multiplied by the appropriate $\mathbf{G}$ matrix (for example, $\mathbf{G}_1$). When displaying the standard coordinates of the non-interactive variable, the points are often displayed as a projection from the origin to their position defined by their standard coordinate.
- For the second type of interactive biplot, the $\mathbf{G}$ matrix is applied to the non-interactive variable. Hence, the categories for this variable are displayed in terms of their principal coordinates while the coordinates corresponding to the combination of categories from the other two variables (i.e., the interactive coordinates) are depicted as standard coordinates (Lombardo et al., 2021).

4.2 Selecting the number of components

The three-way decompositions described in Section 2.3 require a chosen number of components (P, Q and R) for each of the variables of $\underline{\mathbf{N}}$. A common approach is to consider various solutions for the components, resulting in different values of dimensionality (i.e., values of P, Q and R) and then inspect their appropriateness using a goodness-of-fit (or a lack-of-fit) measure with respect to the degrees-of-freedom of the approximation obtained from these solutions. By increasing the number of components the model becomes more complex but the goodness-of-fit of the model improves. Hence, by considering a goodness- (or lack-) of-fit measure for different model complexities, the trade-off between model fit and model complexity can be assessed.

Unfortunately, there is no "best" way to determine the optimal trade-off between model fit and model complexity. Often, the choice of what dimensionality to select is made by visually inspecting a plot of the goodness-of-fit against the degrees-of-freedom of the model. One such plot is a scree-like plot and selecting the desired dimensionality is made by using a variety of strategies including simply looking for an "elbow". One may also select the dimensionality by observing where the "elbow" lies in the lower boundary of the convex hull (Kroonenberg and Oort, 2003; Murakami and Kroonenberg, 2003; Kroonenberg, 2008). Scree-like plots can also be considered by using a measure of goodness- (or lack-) of-fit on the y-axis and the degrees of freedom (or the number of free parameters) on the x-axis. In this case, the analyst selects a model on or close to the "elbow" near the upper boundary of the convex hull (Timmerman and Kiers, 2000; Ceulemans and Kiers, 2006).

To aid in the visual detection of an "elbow" in the convex hull, Ceulemans and Kiers (2006) introduce the *st*-criterion which looks at the smallest angle on the convex hull and allows one to choose a model on the higher (lower) boundary of the convex hull, with the best balance of goodness-(or lack-) of-fit and df (or free parameters). Given the goodness-fit-value, f, and the model complexity-value, df, the *st* criterion for a model of dimensionality l can be written as

$$st(l) = \left(\frac{f(l) - f(l-1)}{df(l) - df(l-1)} \right) \Big/ \left(\frac{f(l+1) - f(l)}{df(l+1) - df(l)} \right) . \tag{17}$$

The number of models to consider when constructing the convex hull depends on the choice of dimensionality, l, made, since there are as many models available to consider as there are combinations of the three dimensions.

In addition to evaluating the goodness-of-fit for the different models, it may also be insightful to asses how stable models of certain dimensionalities are. This can be done by first applying re-sampling procedures to the three-way tables and then considering the resulting convex hulls. Several ways to facilitate such an assessment have been implemented in the **CA3variants** package. More details of the relevant functions and options can be found in Section 2.6.

5 Related software

Currently, there are no packages available in R devoted to three-way correspondence analysis. However, the R packages **PTAk** (Leibovici, 2010), **ThreeWay** (Giordano et al., 2014), **rTensor** (Li et al., 2018), **multiway** (Eilers, 2019), **psych** (Revelle, 2018), **tensorA** (Statnikov, 2018), **mvoutlier** (Zhou, 2019) and **irlba** (Hoffman, 2017) can be used to perform several different three-way decompositions, including the Tucker3 decomposition. An overview of the areas of data analysis that these packages cover is summarised in Table 1.

A complete three-way methods program is also available in Pieter Kroonenberg's Fortran package 3WayPack and includes functionality to perform a multi-way correspondence analysis; see http://three-mode.leidenuniv.nl/ of the *The Three-Mode Company*. Similarly, an extensive collection of three-way methods and decomposition tools are available for MATLAB through the N-Way Toolbox (Bro, 2020). However, while these packages can be used to calculate solutions for the three-way correspondence analysis variants based on the Tucker3 decomposition, doing so requires some non-trivial data preparation and output processing steps.

Table 1: R packages for three-way data analysis. CA3: symmetric three-way correspondence analysis; NSCA3: non-symmetric three-way correspondence analysis; OCA3: ordered symmetric three-way correspondence analysis; ONSCA3: ordered non-symmetric three-way correspondence analysis; PCA3: three-way principal component analysis

	Three-way Data Analysis				
package	CA3	NSCA3	OCA3	ONSCA3	PCA3
CA3variants	x	x	x	x	
ThreeWay					x
PTAk	x				x
rTensor					x
multiway					x
psych					x
tensorA					x
mvoutlier					x
irlba					x

The R package **CA3variants** provides a straightforward way to perform the different variants of three-way correspondence analysis described above on a three-way contingency table. Moreover, in addition to the Tucker3 variants of three-way correspondence analysis, the package also allows for the application of trivariate moment decomposition and hybrid decomposition methods, suitable when variable categories are ordered.

6 CA3variants: Package description and examples

In this section, we introduce the main functions, arguments and options available in the **CA3variants** package. These functions are tunelocal() and CA3variants().

The tunelocal() function can be used to determine an appropriate number of dimensions in the approximation of $\mathbf{\Pi}_P$ or $\mathbf{\Pi}_M$, while the function CA3variants() can be used to perform all four methods described in Section 2.4. Some similarities and differences of these four methods are summarized in Table 2.

The CA3variants() and tunelocal() functions return S3 objects from which the plot(), print()

and summary() functions are available. Note that both functions require the input arguments Xdata, ca3type, resp and norder. Respectively, these arguments specify the three-way data, the type of analysis being performed (which can be chosen from those outlined in Section 2.4), the response variable (in the case of a non-symmetric variant) and the number of ordinal variables (when an ordered variant is performed). Xdata can be a three-way table, or an ($n \times 3$) data matrix where the rows represent the n observations/objects and the 3 columns correspond to three categorical variables, i.e. the row, column and tube variables (the levels/categories of each variable are given by integer numbers).

The tunelocal() function can help the user to choose an appropriate number of dimensions for any variant of three-way correspondence analysis. A list detailing the fit of all of the models considered can be obtained using print(tune.out); here tune.out is the output object produced using the tunelocal() function. This function considers the decompositions of the original data for all triplets of dimensions. The stability of the fit of the solutions for different dimensionalities can also be assessed by adding arguments related to the implementation of three resampling schemes (when 'boots = TRUE' and 'nboots = 100'). The available schemes are a non-parametric bootstrap resampling method or a parametric bootstrap method using one of two distributions (multinomial or Poisson). The parametric bootstrap can be considered as a *simple* parametric bootstrap ('boottype = "bootpsimple"') when the row, column and tube marginals are fixed to equal those of the original three-way table. Alternatively, it can be performed using a stratified parametric bootstrap method ('boottype = "bootpstrat"') where the row and column marginals are fixed for each tube (for $k = 1, ..., K$) of the original three-way table.

Differently from tunelocal(), another important argument of CA3variants() is dims. The argument dims defines the dimensionality of the solution which can be driven by first using tunelocal(). The available variants for ca3type are:

- ca3type = "CA3" for symmetric three-way correspondence analysis. This option is appropriate when all variables are assumed, or known, to be nominal and symmetrically associated. This is also the default analysis that is performed.

- ca3type = "NSCA3" for non-symmetric three-way correspondence analysis. This option is appropriate when one of the variables is defined as the response variable which can be chosen by specifying resp = "row" (the default choice), resp = "column" or resp = "tube". All three variables are treated as being nominal.

- ca3type = "OCA3" for three-way ordered symmetric correspondence analysis. This option is appropriate when at least one of the three variables consists of ordered categories.

- ca3type = "ONSCA3" for three-way ordered non-symmetric correspondence analysis. This option is appropriate when at least one of three variables consists of ordered categories and one of the variables is defined as the response variable. The analyst can specify the response variable in the same way that the response variable is defined for non-symmetric three-way correspondence analysis (see ca3type = "NSCA3").

Method	Variables	Association	Decomposition method
ca3type = "CA3"	nominal	symmetric	Tucker3
ca3type = "NSCA3"	nominal	non-symmetric	Tucker3
ca3type = "OCA3"	ordinal	symmetric	Trivariate moment
ca3type = "ONSCA3"	ordinal	non-symmetric	Trivariate moment
ca3type = "OCA3"	one or two variables are ordinal	symmetric	Hybrid
ca3type = "ONSCA3"	one or two variables are ordinal	non-symmetric	Hybrid

Table 2: Similarities and differences of three-way correspondence analysis methods in CA3variants().

Finally, the package contains four example data sets that can be used to test and benchmark the different methods, all with varying features and variable structures. They are: happy - a $4 \times 6 \times 4$ contingency table with $n = 40323$ - (Davis, 1977), happyNL - a $4 \times 5 \times 4$ contingency table with $n = 1669$ - (from the European Social Survey of 2016, http://www.europeansocialsurvey.org/), museum - a 253×3 data matrix with $n = 253$ - (from a 2019 survery promoted by the University "Luigi Vanvitelli", Italy), and ratrank - a $9 \times 9 \times 5$ contingency table with $n = 44568$ - (van Herk and van de Velden, 2007). In Section 2.6.2 we illustrate the package by performing a NSCA3 on the data set happyNL, while Sections 2.6.1 and 2.6.3 consider two symmetric analyses of the ratrank data set (a nominal three-way correspondence analysis and a hybrid three-way analysis).

6.1 Three-way symmetric correspondence analysis: Ranking and rating data

The dataset `ratrank` is one of the four datasets included in the **CA3variants** package. It is a data array of size $9 \times 9 \times 5$ that is formed from the cross-classification of the *Rating* (row), *Ranking* (column), and *Country* (tube) variables, and was analyzed by van Herk and van de Velden (2007).

Participants from five European countries were asked to rate and rank the same nine values taken from the list of values (LOV) described by Kahle (1983). For each of these European countries, a contingency table was constructed with counts of co-occurrences of rating numbers and rankings. The ranking task required participants to provide a strict ranking of the items. In the rating task, participants are asked to provide ratings (on a 9 point scale) to the same items. It gives the participants the freedom to rank the items in any way they desire, however, it is also open to response tendencies. Such tendencies can be referred to as response styles. For example, some individuals may be more inclined to use extreme ratings (lowest or highest) where others only use middle ratings to express their preferences. The observed correspondence between the ratings and rankings could then be used to inspect response tendencies and, in this study, to relate such tendencies to nationalities. For more details on the data and the theory underlying the response tendencies, we refer to the van Herk and van de Velden (2007). Our objective here is to illustrate the application of the **CA3variants** package by reproducing some of the results published in their paper. After downloading and installing **CA3variants** from the Comprehensive R Archive Network (CRAN), we load the package:

```
library("CA3variants")
```

Dimensionality of the solution

We use the `tunelocal()` function to determine an appropriate triplet of dimensions:

```
tune.ca3.out <- tunelocal(ratrank, ca3type = "CA3")
print(tune.ca3.out)
plot(tune.ca3.out)
```

The function `tunelocal()` yields an object containing goodness-of-fit measures, model complexity and, when `boots = TRUE`, the bootstrap samples used. However, using `print(tune.ca3.out)` we show the following numerical results for the models on the boundary:

```
#> # Convex hull (upper bound)

#> # Selected model(s):
#>          complexity      fit
#> c(2, 2, 1)          1 17726.27


#> All models on upper bound:
#>          complexity       fit        st
#> c(1, 6, 1)           0 14265.56        NA
#> c(2, 2, 1)           1 17726.27 16.097440
#> c(3, 3, 1)           4 18371.23  3.116993
#> c(3, 3, 2)          12 18923.00  1.233506
#> c(3, 4, 2)          17 19202.58  1.846595
#> c(3, 4, 3)          28 19535.67  1.650064
#> c(4, 4, 3)          39 19737.53  1.291062
#> c(4, 4, 4)          54 19950.73  1.538958
#> c(5, 5, 4)          88 20264.76  1.368834
#> c(6, 6, 4)         130 20548.15  1.497295
#> c(7, 7, 4)         180 20773.47        NA
```

The numerical and graphical output of `tune.ca3.out` show that an appropriate triplet of dimensions is (2, 2, 1). Note that Figure 1 is generated when using `plot(tune.ca3.out)`. While cluttered, as the result of the large number of triplets that can be considered in the analysis of `ratrank`, Figure 1 also shows that an appropriate dimensionality of the solution is (2, 2, 1). Each point in Figure 1 corresponds to a combination of dimensions. The y-axis gives the goodness-of-fit measure for each model which we use as the criterion for choosing the most appropriate dimensionality for the solution. More complex models that involve higher dimensionalities (or, equivalently, higher degrees of freedom) have a better fit. The red line outlines the convex hull where models on this line are superior to higher dimensional options with a similar fit. For example, in Figure 1, for the models that lie below the red line there is typically an alternative, less complex, model that achieves the same fit. Alternatively,

Figure 1: Model fit versus complexity for three-way nominal CA of the `ratrank` data.

there is also an equally complex model having a better fit. In three-way correspondence analysis, like other dimension reduction methods, users can factor in subjective criteria (such as interpretability) when selecting the dimensionality of a model. Here, in accordance with Ceulemans and Kiers (2006), we follow the *st* criterion and select the model, marked green in Figure 1, with two dimensions for the row (*Rating*) and column (*Ranking*) variables and one for tube (*Country*) variable.

Numerical summary of the association

By following the analysis described in van Herk and van de Velden (2007) we perform a symmetric three-way correspondence analysis, the default method, on the `ratrank` data. Following the output of the tunelocal function and in accordance with van Herk and van de Velden (2007), we specify two dimensions for the row (*Rating*) and column (*Ranking*) categories, and a single dimension for the tube (*Country*) categories. This can be achieved by:

```
ca3.out <- CA3variants(ratrank, dims = c(2, 2, 1))
```

The `print()` function returns several key measures of association that are included in this output. These include the percentage of explained inertia along each dimension, the partition of Pearson's three-way chi-squared and phi-squared statistics into four terms (see equation (4)), the corresponding degrees of freedom, the p-value, and the relative sizes of each term of the partition (allowing for comparisons between chi-squared values from different, asymptotic chi-squared distributions):

```
print(ca3.out)

#> # Percentage contributions of the components to the total inertia for column-tube
biplots

#>     p1      p2
#> 67.008  16.347

#> # Percentage contributions of the components to the total inertia for row-tube
biplots
```

```
#>      q1      q2
#> 67.008 16.347

#> # Percentage contributions of the components to the total inertia for row-column
biplots

#>      r1
#> 83.355

#> #  Index partition

#>              Term-IJ Term-IK Term-JK Term-IJK Term-total
#> Chi-squared 18359.272 589.605 254.629 2062.404  21265.910
#> Phi-squared     0.412   0.013   0.006    0.046      0.477
#> % of Inertia   86.332   2.773   1.197    9.698    100.000
#> df             64.000  32.000  32.000  256.000    384.000
#> p-value         0.000   0.000   0.000    0.000      0.000
#> X2/df         286.864  18.425   7.957    8.056     55.380
```

This output shows that the Pearson chi-squared statistic of `ratrank` is 21265.91 and, with a p-value that is less than 0.0001, there is a statistically significant association between at least two of the variables of the data set. Further insight into the nature of the association can be obtained from the terms of the partition of the overall chi-squared. The output shows that the most dominant source of association exists between the *Rating* and *Ranking* variables (`Term-IJ`), and contributes to 18359.27, or 86.33%, of the total association among the three variables. The association between the *Rating-Country* variables (`Term-IK`) and *Ranking-Country* variables (`Term-JK`) accounts for relatively little in comparison (2.77% and 1.20%, respectively), but are still statistically significant sources of association. The association among all three variables (`Term-IJK`) contributes to the remainder (or nearly 10%) of the association between the variables. Further information about the nature of the association can be obtained visually by performing a correspondence analysis.

Visual summary of the association

To reproduce the results from van Herk and van de Velden (2007), we consider here the row-tube (*Rating - Country*) interactive biplot, so that the interactive row-tube points are plotted using principal coordinates. This biplot is given by Figure 2 and is produced from the command:

```
plot(ca3.out, biptype = "row-tube", addlines = F)
```

By default, the `plot()` function uses a straight line from the origin to each standard coordinate to depict the non-interactive variable. However, with so many points in Figure 2, adding projection lines for each of the nine *Ranking* categories leads to a cluttered plot. Hence, and in accordance with van Herk and van de Velden (2007), we use `addlines = F` to remove the lines. Furthermore, to control the size of the points and their labels, the `plot()` function uses two arguments `size1` and `size2` (for the points and labels, respectively); by default `size1 = 1` and `size2 = 3`.

Finally, to avoid any further clutter of points close to the origin, a scaling argument can be used that helps to reveal important features of the association without impacting the approximation. The default for this scaling argument, which was applied here, is set such that the average sum of squares for the two sets of points is the same, and is thus in accordance with the recommendations given by Gower et al. (2010) and van de Velden et al. (2017). This default can be overwritten by specifying a value for the `scaleplot` argument in the `plot()` function. Note that, except for this scaling, the biplot given by Figure 2 is identical to Figure 1 in van Herk and van de Velden (2007).

Figure 2 shows that the highest value rank ("rank9") generally receives the highest possible rating ("9") across all five countires. However, for the second highest value rank ("rank8") the ratings tend to vary from 4 to 8, showing some heterogeneity in how "rank8" is perceived in terms of the *Rating* categories. For the lowest valued rank ("rank1"), we see a clear association with the lowest rating ("1"). However, this level of rating is also often linked to items that received a rank of "2". Moreover, for the items that receive a rank of "1" up to "7", we see that individuals tend to assign to them a rating of between "1" and "3" (inclusive). Finally, each of the ratings appear rather homogeneous across all five countries. However, with ratings from Germany being consistently furthest from the origin, and those from the United Kingdom being closest to the origin, these *Country* categories provide, relatively speaking, the strongest and weakest (respectively), contribution to the association. See van Herk and van de Velden (2007) for a more in-depth analysis and explanation of this analysis.

6.2 Three-way non-symmetric correspondence analysis: Happiness data

Since 1977, the study of the relationship between happiness, household characteristics and education using data obtained from social survey data has received a great deal of attention. For an analysis of this data set, see, for example, Davis (1977), Clogg (1982), Beh and Davy (1998) and Kroonenberg (2008, Chap.17). Davis' data set Davis (1977) examines the association between happiness, number of siblings and years of schooling completed of 1517 individuals and is included in **CA3variants** as happy. Kroonenberg (2008, Chap.17) studied Davis' data by performing a symmetric three-way correspondence analysis.

Following on from those studies mentioned above, we analyze a three-way contingency table, obtained from the 2016 European Social Survey (http://www.europeansocialsurvey.org/). It involves a sample of 1669 respondents from the Netherlands and investigates the association between their reported level of *Happiness*, the level of *Education* and the number of people in their *Household*. As an illustration of one of the three-way variants implemented in the **CA3variants** package, we now turn our attention to performing the non-symmetric variant of three-way correspondence analysis.

Defining the variables

To assess the level of *Happiness* of a respondent, people were asked to reply to the question:

"Taking all things together, how happy would you say you are?"

Responses were made on a scale from 1 ("extremely unhappy") to 10 ("extremely happy"). After observing the distribution of counts in the data, we re-coded these scales into the following four

Figure 2: Interactive row-tube biplot for `ratrank`. The column points (*Ranking* categories) are depicted using standard coordinates and are labeled as "rank1" to "rank9". The interactive row-tube points (*Rating-Country* categories) are depicted using principal coordinates and are labelled by the rating number (1 to 9) followed by the first letter of the country: France (F), Germany (G), Italy (I), Spain (S) and the United Kingdom (U).

categories of *Happiness*: "low" (for ratings < 6), "middle" (for ratings between $6 - 7$), "high" (for a rating equal to 8), and "very-high" (for ratings > 8).

The *Education* variable is defined using four categories. These are "Less than lower secondary education" (coded "ED1"), "Lower secondary education completed" (coded "ED2"), "Upper secondary education completed" (coded "ED3") and "Post-secondary and/or tertiary education completed" (coded "ED45").

Finally, for the *Household* variable, the respondents were asked to reply to the question:

> *"Including yourself, how many people - including children - live here regularly as members of this household?"*

The four categories from this question were defined as follows: a one person household is coded "HS1", a two person household is coded "HS2", a three person household is coded "HS3", a four person household is coded "HS4", a five person household is coded "HS5' and a household containing more than five people is coded ">HS5".

The cross-classification of the *Happiness*, *Education* and *Household* variables forms a three-way contingency table which has been included in the package with the object name happyNL.

For our analysis of this contingency table, we consider the non-symmetric three-way correspondence analysis variant with the row variable (*Happiness*) treated as the response variable, and the column (*Education*) and tube (*Household*) variables defined as the predictor variables.

Dimensionality of the solution

Before performing a three-way NSCA on happyNL we first need to determine the dimensionality of the solution. This can be done by comparing the fit and complexity of models of different dimensionality using the tunelocal() function. For this example, we consider decompositions applied to 100 resampled data tables (using the parametric bootstrap; the default), and calculate, for each triplet of dimensions, the mean goodness of fit over the bootstrap samples. Note that, by doing so, the overall number of estimated models equals $I \times J \times K \times nboots = 80 \times 100 = 8000$. All resampled data tables are collected in the object named 'XG' of the output of the tunelocal() function:

```
tune.nsca3.out <- tunelocal(happyNL, ca3type = "NSCA3", resp = "row", boots = T)
plot(tune.nsca3.out)
```

The resulting plot is given as Figure 3. Each point in this plot corresponds to a combination of dimensions. The y-axis gives the goodness-of-fit measure for each model. More complex models, that is those involving higher dimensionalities have a better fit. The red line denotes the convex hull where models on this line are superior to higher dimensional options with a similar fit. For example, in Figure 3, for the models which are below the red line there is an alternative, less complex, model achieving the same (or similar) fit, or there is an equally complex model having a better fit. In three-way correspondence analysis, as with other dimension reduction techniques, users can factor in subjective criteria such as interpretability when selecting the dimensionality of a model. Here, in accordance with Ceulemans and Kiers (2006), we follow the *st* criterion and select the model, marked green in Figure 3, with two dimensions for the row (*Education*) column (*Household*) and tube (*Happiness*) variables.

Using print(tune.nsca3.out) we obtain the numerical results (i.e. fit) for the models that lie along the red line in Figure 3. The output from this command is:
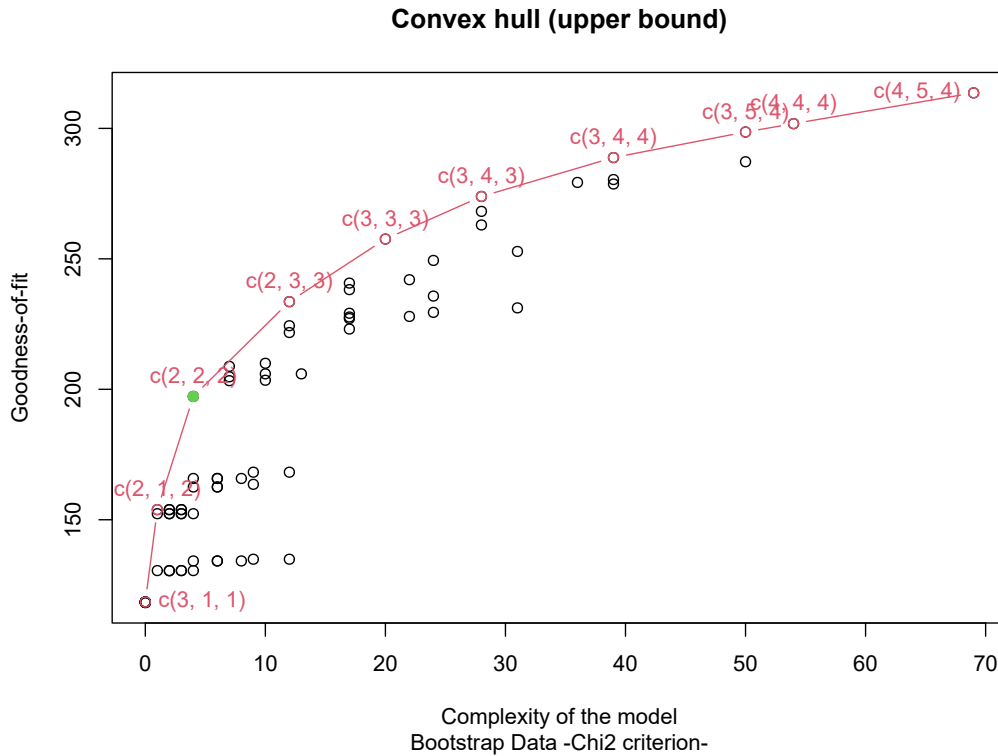
```
#> # Note that when boots = T,  the data samples generated
#> # are given in the object named 'XG'

#> # Results for choosing the optimal model dimension

#> # Convex hull (upper bound)

#> # Selected model(s):
#>             complexity      fit
#> c(2, 2, 2)          4 187.6015


#> # All models on upper bound:
#>             complexity      fit        st
#> c(1, 1, 4)          0 111.6055        NA
#> c(1, 2, 2)          1 145.6444 2.433839
```

Convex hull (upper bound)

Figure 3: Model fit versus complexity for three-way NSCA of the happiness bootstrapped data.

```
#> c(2, 2, 2)      4 187.6015 2.965930
#> c(2, 3, 3)     12 225.3251 1.766711
#> c(3, 3, 3)     20 246.6776 1.382302
#> c(3, 4, 3)     28 262.1246 1.246718
#> c(3, 4, 4)     39 279.1611 1.734889
#> c(3, 5, 4)     50 288.9810 1.190973
#> c(4, 5, 4)     69 303.2229       NA
```

Numerical summary of the association

The `CA3variants()` function can be used to perform a three-way non-symmetric correspondence analysis on happyNL by specifying the arguments of the function so that they define the data table, the dimensionality of the solution, the type of analysis and the response variable. Here, using the suggested dimensions, the analysis is performed so that:

```
nsca3.out <- CA3variants(happyNL, ca3type = "NSCA3", resp = "row",
dims = c(2, 2, 2))
```

The numerical output from this analysis is obtained using `print(nsca3.out)`:

```
print(nsca3.out)

#> # Percentage contributions of the components to the total inertia for pred biplots

#>  p1     p2
#> 47.407 22.696

#> #  Index partition

#>              Term-IJ Term-IK Term-JK Term-IJK Term-total
#> Tau Numerator  0.014   0.005   0.008    0.006      0.032
#> Tau            0.021   0.007   0.012    0.008      0.047
#> % of Inertia  43.162  14.719  24.749   17.371    100.000
```

```
#> CM-Statistic    102.583   34.981   58.819   41.285    237.669
#> df               12.000    9.000   12.000   36.000     69.000
#> p-value           0.000    0.000    0.000    0.251      0.000
#> CM-Statistic/df   8.549    3.887    4.902    1.147      3.444
```

By reducing the dimensionality of the solution to (2, 2, 2), the first set of values (p1 and p2) are the percentages of the total association which is explained by the two axes of a biplot; we will speak more on these two values shortly. The summary of values that follow `Index partition` gives the four terms of the partition of the Marcotorchino index, its numerator and its associated test statistic, C_M-statistic (see Section 2.2.3). Note that the last column, labeled `Term-total` corresponds to the three-way index being partitioned. Consequently, the seven rows of this output summarize the elements of each term of this partition, including their p-value and their relative sizes (allowing for comparisons between C_M-statistic values from different, asymptotic chi-squared distributions).

The data set happyNL has a C_M-statistic of 237.669. Its small p-value (< 0.0001, df = 69) confirms that there is very strong evidence to conclude that the *Household* and *Education* variables are statistically significant predictors of *Happiness*. By partitioning the C_M-statistic associated with the Marcotorchino index, we can examine the sources of non-symmetric association that exists in the three-way table. We see that all the bivariate association terms are statistically significant, but not the trivariate association term (p-value < 0.251, df = 36) which assesses the increase in predictability of *Happiness* given the number of people in a *Household* and the highest level of *Education* of the participants.

Visual summary of the association

While the trivariate term from the partition of the C_M-statistic is not statistically significant, we shall nonetheless visually explore how people's level of *Happiness* is influenced by the number of people in their *Household* and their highest level of *Education*. This shall be done by generating an interactive biplot with the interaction of each combination of categories of the predictor variables depicted using principal coordinates and, therefore, setting `biptype = "pred"`. Note that there is indeed an "interaction" (via a symmetric association) between the two predictor variables since the `Term-JK` p-value is less than < 0.0001. The categories of the response variable are depicted in the biplot using standard coordinates when `biptype = "pred"`.

Applying the `plot()` function to the **CA3variants** object can be used to generate different biplots. A description of some of all available plotting arguments can be found in Table 3. However, when a non-symmetric variant is applied to the **CA3variants** object, a suitable interactive biplot that portrays the non-symmetric association can be obtained using the command:
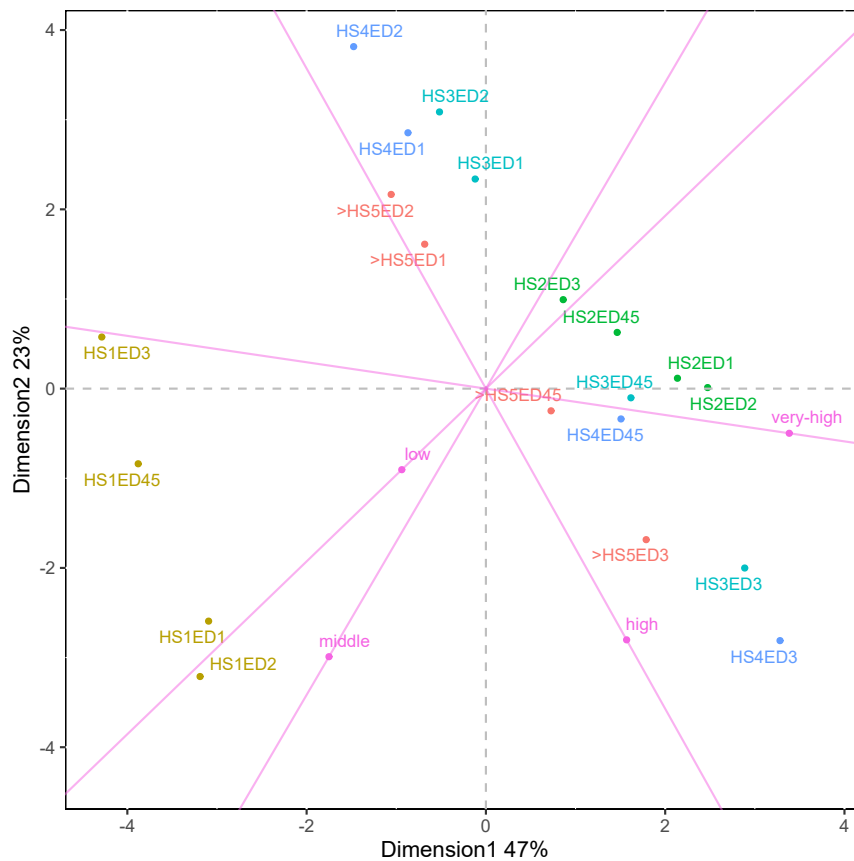
```
plot(nsca3.out, biptype = "pred")
```

which produces the interactive biplot of Figure 4. Figure 4 displays straight lines from the origin to each standard coordinate to depict the non-interactive variable for the four levels of the *Happiness* variable. Such lines are convenient for visualizing how the interactive points relate to the non-interactive points. This is because the proximity of the points from the origin reflect deviations from independence.

In Figure 4, we see that the first dimension accounts for 47% (rounded to the nearest integer) of the association between the variables while the second dimension accounts for 23%. Thus, Figure 4 captures approximately 70% of the association between the three categorical variables (when treated non-symmetrically) of happyNL. These two percentages are also included as p1 and p2, respectively, from the numerical summaries included in `print(nsca3.out)`. Since Figure 4 provides a good visual summary of the non-symmetric association of the variables of happyNL, we now turn our attention to describing the nature of this association. The left side of Figure 4 shows a group of points corresponding to HS1 (a single person household) combined with all levels of education. It shows that respondents tend to exhibit lower levels of happiness when they live alone, regardless of education level. Due to the non-symmetric nature of the association we can also infer that for these single households, the groups with lower levels of education (HS1ED1 and HS1ED2) lead (or help predict) a low, or middle, level of happiness. For those with a higher education, Figure 4 also suggests that having a higher level of education does not necessarily lead to (or help to predict) a very-high happiness level. Furthermore, respondents in a two person household (HS2) tend to be very happy (HS2 is a good predictor of very-high levels of happiness), especially for those with a lower level of education (HS2ED1 and HS2ED2). The interactive biplot shows that those with higher levels of education in a two person household are still more associated with a very-high level of happiness (HS2ED3 and HS2ED45) but less than those with less of an education.

For the large households (HS4 and >HS5), we observe that the effect of education level on happiness appears to be stronger. That is, for these larger households, respondents with a higher (ED45) or a middle-high (ED3) level of education tend to be more happy (high and very-high) than people with a

Arguments	Description
Xout	The output of CA3variants().
biptype	Specifies the type of interactive biplot being produced. When ca3type = "CA3" or = "OCA3" there are six options: biptype = "row", "column", "tube", "row-column", "row-tube" and "column-tube". Each option refers to what is depicted using principal coordinates. For instance, "row" specifies that the row points are depicted using principal coordinates and, consequently, the interactive column-tube points are depicted using standard coordinates. When ca3type = "NSCA3" or "ONSCA3", there are only two biplot options: biptype = "resp" or "pred". The option "resp" specifies that the response categories are depicted using principal coordinates, while the option "pred" indicates that the interactive predictor points are in principal coordinates.
scaleplot	A biplot scaling argument used to avoid spatial cluttering by pulling points away from the origin. See the description of the "gamma scaling" in Gower et al. (2011, Section 2.3.1). By default, scaleplot is the overall average of the sum-of-squares of the two sets of coordinates (principal and standard ones), so that the average sum-of-squares for the two sets of points is the same (van de Velden et al., 2017).
addlines	Specifies whether the points in standard coordinates are represented using axes. By default, addlines = TRUE.

Table 3: Summary of important plotting options available in plot.CA3variants(). For all options use ?plot.CA3variants

Figure 4: Interactive biplot from the NSCA3 of happyNL with *Happiness* and *Education* the interactive variables

lower level of education (ED1 and ED2). Respondents that live in large households and have a low level of education (HS4ED2, HS4ED1, >HS5ED2 and >HS5ED1) are not highly happy individuals. Indeed, these interative points are on the opposite side of Figure 4 to the high level of happiness.

6.3 Ordered three-way correspondence analysis: Ranking and rating data

In the analysis of `ratrank` in Section 2.6.1 we treated the *Ranking* and *Rating* variables as nominal when they are, in fact, ordinal variables. Using the `CA3variants` package we can incorporate this ordinality in the decomposition. In particular, we perform the analysis by treating *Country* as a nominal variable and *Ranking* and *Rating* as ordinal by using the hybrid decomposition described in Section 2.3.3.

Dimensionality of the solution

Before we construct a low-dimensional display of the association between the ordinal variables (*Rating* and *Ranking*) and the nominal variable (*Country*), we determine the appropriate dimension of the solution. The $9 \times 9 \times 5$ data set `ratrank` has a $8 \times 8 \times 4$ sized matrix of hybrid core elements that reflect the trivariate sources of association between the three variables. Not all these sources are important for describing the analysis, or are even practically relevant. In most practical cases the linear and quadratic sources of association are sufficient and provide a meaningful description of the association. We use the `tunelocal()` function to determine the appropriate number of hybrid core elements to define the dimensionality of the solution. When using the `tunelocal()` function for analysing ordinal variables, one needs to specify the number of them; this is done by setting the argument `norder = 2`. The numerical and graphical summaries from using this function are obtained using the commands:

```
tune.oca3.out <- tunelocal(ratrank, ca3type = "OCA3", norder = 2)
print(tune.oca3.out)
```

The numerical and graphical output of a similar form to those seen in the previous examples. The visual and numerical outputs (not given here) show that the highest order hybrid core element is of order (2, 2, 1) (i.e. the quadratic-by-quadratic-by-first order component association) so that all terms, up to the (2, 2, 1) term, together account for most of the association between the three variables.

Numerical summary of the association

We perform OCA3 on `ratrank` using the dimensionalities as suggested by the output of the `tunelocal()` function. Hence, we confine our attention to sources of association no higher than the quadratic-by-quadratic-by-first hybrid core so that:

```
oca3.out <- CA3variants(ratrank, ca3type = "OCA3", dims = c(2, 2, 1), norder = 2)
```

We note that from such an analysis, only four of the 256 hybrid core elements are required to account for most of the association that exists between the variables. We can gain more insight into the structure of the association by inspecting the core elements from the hybrid decomposition. When using the function `summary()`, the elements of the core and squared core arrays (Lombardo et al., 2021), respectively, can be obtained:

```
summary(oca3.out)

#> Core table
#> , , r1

#>        q1     q2
#> p1 -0.527 -0.143
#> p2  0.198 -0.246

#> Squared core table
#> , , r1

#>       q1     q2
#> p1 0.278 0.020
#> p2 0.039 0.061

#> Explained inertia (reduced dimensions)
#> [1] 0.398

#> Total inertia (complete dimensions)
#> [1] 0.477
```

```
#> Proportion of explained inertia (when reducing dimensions)
#> [1] 0.834
```

Note that by confining the solution to include terms no higher than (2, 2, 1), the sum of squares of these four squared core elements is 83.4% of the association that exists between the three variables of the contingency table.

The four terms from this output are all adequately described using the linear and quadratic polynomials for *Rating* and *Ranking* and just one Tucker3 component for *Country*, and are:

- the linear-by-linear polynomial component term (0.278) which describes the association between the ordered variables in terms of any differences that exist in the linearity of each ordered set of categories that form the *Rating* and *Ranking* variables,
- the linear-by-quadratic polynomial component term (0.020) which describes the association between the ordered variables in terms of any linear differences in the *Rating* variable and dispersion differences in the *Ranking* variable,
- the quadratic-by-linear polynomial component (0.039) which describes the association between the ordered variables in terms of any dispersion differences in the *Rating* variable and any linear differences that exist in the *Ranking* variable, and
- the quadratic-by-quadratic polynomial component (0.061) which describes the association between the ordered variables in terms of any dispersion differences that exist in the *Rating* and *Ranking* variables.
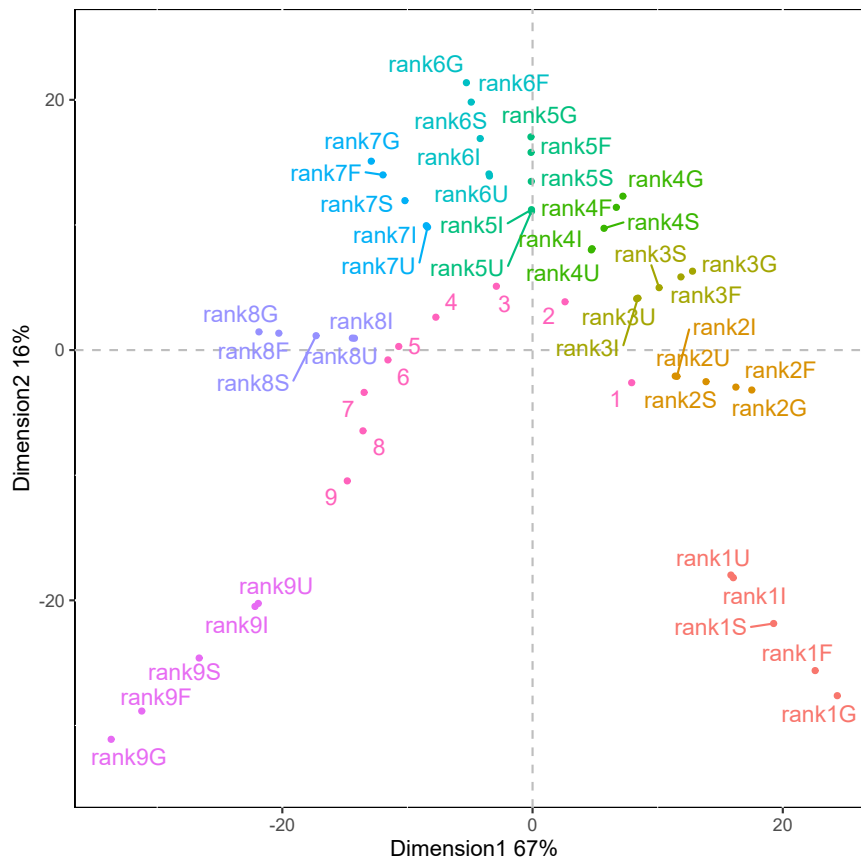
Using the `print()` function we obtain the percentage contributions of the components to the total inertia for different biplots, the overall decomposition information, as well as the partitionings of the four terms of the Pearson three-way chi-squared statistic into their polynomial components. For example, suppose we focus on the pair-wise association between the *Rating* and *Ranking* variables. The partial output corresponding to the row and column (linear and non-linear) components of the `Chi2-IJ` term can be shown:

```
print(oca3.out)
#> # ...
#> # Partition of the Term-IJ using polynomials

#>          Term-IJ-poly %inertia df p-value
#> poly-row1   12701.275   69.182 8       0
#> poly-row2    3882.996   21.150 8       0
#> poly-row3     833.867    4.542 8       0
#> poly-row4     346.473    1.887 8       0
#> poly-row5     177.262    0.966 8       0
#> poly-row6     167.495    0.912 8       0
#> poly-row7     122.977    0.670 8       0
#> poly-row8     126.927    0.691 8       0
#> Chi2-IJ     18359.272  100.000 64      0
#> poly-col1   13819.761   75.274 8       0
#> poly-col2    3177.816   17.309 8       0
#> poly-col3     605.183    3.296 8       0
#> poly-col4     189.747    1.034 8       0
#> poly-col5     149.127    0.812 8       0
#> poly-col6     160.638    0.875 8       0
#> poly-col7     124.849    0.680 8       0
#> poly-col8     132.150    0.720 8       0
#> Chi2-IJ     18359.272  100.000 64      0
#> # ...
```

This output shows that all components are statistically significant (with p-values smaller than 0.0001). It also shows that the variation in the *Rating* variable (row variable) is dominated by the difference in the linearity of its categories - the linear component accounts for $100 \times 12701.28/18359.27 = 69.18\%$ of the variation in this variable. The linear component also accounts for the largest source of variation in the *Ranking* variable (column variable), contributing to $100 \times 13819.76/18359.27 = 75.27\%$ of the variables' variation. Thus, if we were to confine our attention to just exploring further the association between the *Rating* and *Ranking* variables by generating a visual summary of the association this can be done using the correspondence analysis approach introduced in Beh (1997) and described by Beh and Lombardo (2014, Chap. 6) and Beh and Lombardo (2021b, Chap. 4). Since both variables are dominated by differences in the linearity of their categories, such an analysis will produce a correspondence plot that is dominated more by the first axis than any of the other axis in the optimal plot.

Figure 5: The row biplot from the classical three-way correspondence analysis of ratrank.

Visual summary of the association

Since the three-way association term is statistically significant ($X^2 = 21265.91$, p-value < 0.0001), we can examine the nature of this association term more closely. Visually summarizing this three-way association can be done by considering the coordinate systems that generate the biplots described in Section 2.4.1. Recall that in Section 2.6.1, we performed the classical approach to three-way correspondence analysis and visualized the results using the row-tube (interactive) biplot. In doing so, the *Ranking-Country* association – which is comparatively weak (contributing to 1.2% of the association) but is statistically significant (p-value < 0.0001) – is depicted using standard coordinates while the *Rating* categories are depicted using principal coordinates that are akin to $\mathbf{X}_{IK,J}$ in (16).

To highlight differences between the classical and ordered three-way correspondence analysis, we construct the row biplot of Figure 5 using the command:
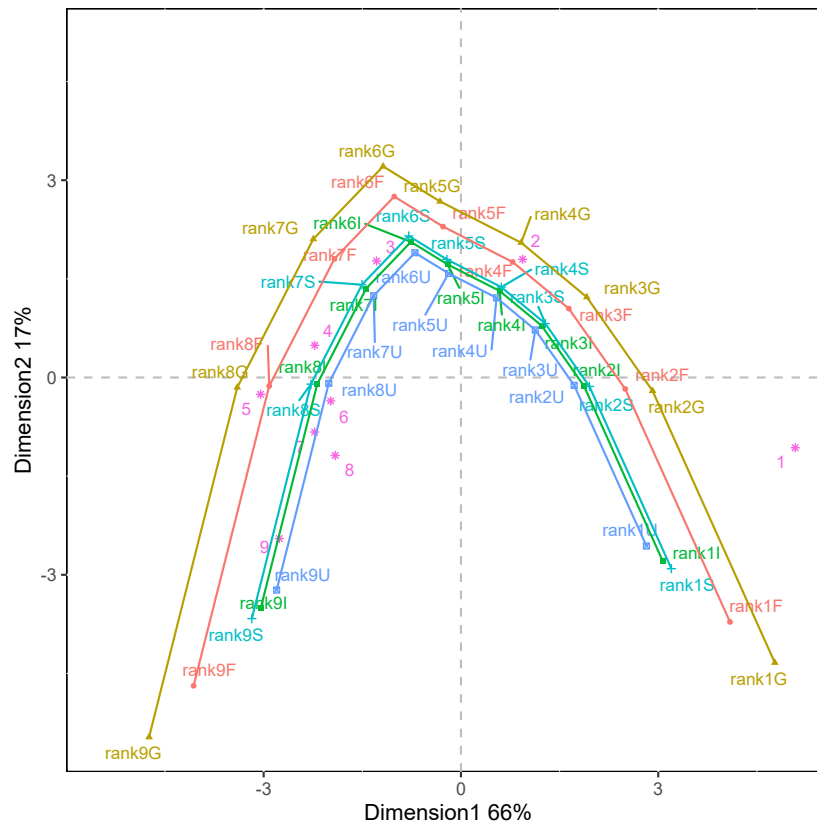
```
plot(ca3.out, biptype = "row", addlines = F, scaleplot = 15)
```

note that ca3.out is the output from the classical analysis performed in Section 2.6.1. When *Rating* and *Ranking* are treated as ordinal variables, Figure 6 gives the row biplot that can be obtained from the command:

```
plot(oca3.out, biptype = "row", scaleplot = 15)
```

where the value for scaleplot = 15 was chosen by trial and error to ensure a reasonable separation of the points in the biplot, without afffecting the approximation of the association between the variables. While Figure 5 and Figure 6 both give parabolic configurations of the points, these configurations are quite different since the former treats the variables as nominal and uses the components from a Tucker3 decomposition while the latter is constructed using orthogonal polynomials for the ordered row and column (*Ratings and Rankings*) variables and a Tucker3 component for the tube (*Country*) variable. Observe that the parabolic shape of *Rating* in Figure 5 is more pronounced than the parabolic configuration of *Rating* in Figure 6.

In addition to the visual differences between the two configurations of points, there are also some features that make Figures 5 and 6 distinct. Indeed, the first axis of Figure 6 is constructed using the

Figure 6: The row biplot from the ordered (hybrid) three-way correspondence analysis of `ratrank`.

linear orthogonal polynomial while the second axis is constructed using the quadratic orthogonal polynomial. The linear and dispersion components contribute to 66% and 17%, respectively, of the total inertia of the data; these percentages can be obtained from the output of the `print(oca3.out)` command.

When considering all variables as nominal, as was done in our analysis in Section 2.6.1, the ratings appear closely associated with the rankings across the five countries. However, treating the two variables (*Rating* and *Ranking*) as ordinal provides additional information on some aspects of the variable distribution (mean and variability). For example, Figure 6 shows that the configuration of the *Rating* categories along the first (linear polynomial) axis is different to the configuration along the first axis of Figure 5. This is because the variation of the *Rating* variabe is dominated more by differences in the linearity of its categories than by its dispersion differences. This dominant linear component affects the variable association and is captured by the configuration of points in Figure 6.

7 Conclusion

The **CA3variants** package described in this paper is, to the best of our knowledge, the only package that allows practitioners and researchers to directly perform four variants of three-way correspondence analysis, including the classical three-way correspondence analysis (Carlier and Kroonenberg, 1996), the non-symmetric variant and the two ordered versions of three-way correspondence analysis (Lombardo et al., 2021). Subsequent versions of the package may allow for additional flexibility by providing the user more tools to numerically and visually explore the association structure between categorical variables. These include, but are not confined to, the decomposition of the generalised Cressie-Read family of divergence statistics (Pardo, 1996). Indeed, Pearson's statistic is one of many measures of symmetric association that can be considered. Alternatives include the Freeman-Tukey statistic, log-likelihood ratio statistic, Neyman's chi-squared statistic, and the Cressie-Read statistic, which were originally developed to study two variables (Cressie and Read, 1984; Beh and Lombardo, 2023). These measures are all special cases of the Cressie-Read family of divergence statistics and have been adapted for three-way and multi-way contingency tables (Pardo, 1996; Pardo and Pardo, 2003;

Lombardo and Beh, 2022). Thus, this family of statistics may be incorporated into the **CA3variants** package, thereby providing the user with greater flexibility for the choice of symmetric association they wish to consider. Furthermore, next version of the package might consider the construction of confidence regions that determine those categories (and interactions) that provide a statistically significant contribution to the association between the variables (Beh, 2010; Ringrose, 1996, 2012). Numerical summaries that accompany such regions, including p-values (Beh and Lombardo, 2015) can certainly be incorporated and would provide similar functionality that is available in the **CAvariants** package used for the correspondence analysis of two cross-classified categorical variables (Lombardo and Beh, 2016).

References

E. J. Beh. Simple correspondence analysis of ordinal cross-classifications using orthogonal polynomials. *Biometrical Journal*, 39:589 – 613, 1997. URL https://doi.org/10.1002/bimj.4710390507. [p242, 256]

E. J. Beh. A comparative study of scores for correspondence analysis with ordered categories. *Biometrical Journal*, 40:413 – 429, 1998a. URL https://doi.org/10.1002/(SICI)1521-4036(199808)40:4<413::AID-BIMJ413>3.0.CO;2-V. [p242]

E. J. Beh. *Correspondence Analysis using Orthogonal Polynomials*. Unpublished PhD Thesis, University of Wollongong, Australia, 1998b. [p241, 242]

E. J. Beh. Elliptical confidence regions for simple correspondence analysis. *Journal of Statistical Planning and Inference*, 140:2582 – 2588, 2010. URL http://dx.doi.org/10.1016/j.jspi.2010.03.018. [p259]

E. J. Beh and P. J. Davy. Partitioning Pearson's chi-squared statistic for a completely ordered three-way contingency table. *The Australian and New Zealand Journal of Statistics*, 40:465 – 477, 1998. URL https://doi.org/10.1111/1467-842X.00050. [p241, 250]

E. J. Beh and R. Lombardo. *Correspondence Analysis, Theory, Practice and New Strategies*. John Wiley & Sons, Chichester, UK, 2014. [p237, 239, 240, 242, 256]

E. J. Beh and R. Lombardo. Confidence regions and approximate p-values for classical and non symmetric correspondence analysis. *Communications in Statistics - Theory and Methods*, 44:95 – 114, 2015. URL https://doi.org/10.1080/03610926.2013.768665. [p259]

E. J. Beh and R. Lombardo. Features of the polynomial biplot for ordered contingency tables. *Journal of Computational and Graphical Statistics*, 31:403 – 412, 2021a. URL https://doi.org/10.1080/10618600.2021.1990773. [p242]

E. J. Beh and R. Lombardo. *An Introduction to Correspondence Analysis*. John Wiley & Sons, Chichester, UK, 2021b. [p239, 240, 242, 256]

E. J. Beh and R. Lombardo. Correspondence analysis using the cressie–read family of divergence statistics. *International Statistical Review*, 2023. URL https://doi.org/10.1111/insr.12541. [p258]

E. J. Beh, B. Simonetti, and L. D'Ambra. Partitioning a non-symmetric measure of association for three-way contingency tables. *Journal of Multivariate Analysis*, 98:1391 – 1411, 2007. URL https://doi.org/10.1016/j.jmva.2007.01.011. [p239, 240]

R. Bro. The N-Way Toolbox. 2020. URL https://www.mathworks.com/matlabcentral/fileexchange/1088-the-n-way-toolbox. [p245]

A. Carlier and P. M. Kroonenberg. Decompositions and biplots in three-way correspondence analysis. *Psychometrika*, 61:355 – 373, 1996. URL https://doi.org/10.1007/BF02294344. [p237, 238, 239, 241, 243, 258]

A. Carlier and P. M. Kroonenberg. The case of the French cantons: An application of three-way correspondence analysis. In J. Blasius and M. Greenacre, editors, *Visualization of Categorical Data*, pages 253 – 275. Academic Press, San Diego, 1998. [p237]

E. Ceulemans and H. A. L. Kiers. Selecting among three-mode principal component models of different types and complexities: A numerical convex hull based method. *British Journal of Mathematical & Statistical Psychology*, 59:133 – 150, 2006. URL https://doi.org/10.1348/000711005X64817. [p244, 248, 251]

C. C. Clogg. Some models for the analysis of association in multiway cross-classifications having ordered categories. *Journal American Statistical Association*, 77:803 – 815, 1982. URL https://doi.org/10.2307/2287311. [p250]

N. A. C. Cressie and T. R. C. Read. Multinomial goodness-of-fit tests. *Journal of the Royal Statistical Society, Series B*, 46:440 – 464, 1984. URL http://www.jstor.org/stable/2345686. [p258]

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. [p239]

J. A. Davis. Codebook for the 1977 General Social Survey, 1977. National Opinion Research Centre, Chicago. [p246, 250]

P. H. C. Eilers. **multiway**: Analysis of multi-way arrays, 2019. URL https://CRAN.R-project.org/package=multiway.Rpackageversion1.0-6. [p245]

P. Emerson. Numerical construction of orthogonal polynomials from a general recurrence formula. *Biometrics*, 24:696 – 701, 1968. URL https://doi.org/10.2307/2528328. [p241]

P. Giordano, H. A. Kiers, and M. A. D. Ferraro. Three-way component analysis using the R package threeway. *Journal of Statistical Software*, 57:1 – 23, 2014. URL 10.18637/jss.v057.i07. [p245]

L. A. Goodman and W. H. Kruskal. Measures of association for cross classifications. *Journal of the American Statistical Association*, 49:732 – 764, 1954. URL https://doi.org/10.2307/2281536. [p239]

J. C. Gower, P. J. F. Groenen, and M. van de Velden. Area biplots. *Journal of Computational and Graphical Statistics*, 19:46 – 61, 2010. URL https://www.jstor.org/stable/25651299. [p249]

J. C. Gower, S. Lubbe, and N. le Roux. *Understanding Biplots*. Wiley, Chichester, 2011. [p244, 254]

J. C. Gower, P. J. F. Groenen, M. van de Velden, and K. Vines. Better perceptual maps: Introducing explanatory icons to facilitate interpretation. *Food Quality and Preference*, 36:61 – 69, 2014. URL https://doi.org/10.1016/j.foodqual.2014.01.004. [p244]

M. Greenacre. *Biplots in Practice*. Fundación BBVA, Barcelona, 2010. [p244]

M. J. Hoffman. **irlba**: Fast truncated singular value decomposition and principal components analysis for large dense and sparse matrices, 2017. URL https://CRAN.R-project.org/package=irlba.Rpackageversion2.3.5.1. [p245]

L. R. Kahle. *Social Values and Social Change: Adaptation to Life in America*. Praeger, New York, 1983. [p247]

H. A. L. Kiers, P. M. Kroonenberg, and J. M. F. T. Berge. An efficient algorithm for TUCKALS3 on data with large numbers of observation units. *Psychometrika*, 57:415 – 422, 1992. URL https://doi.org/10.1007/BF02295429. [p240]

P. 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. URL https://doi.org/10.1207/S15327906MBR3403_4. [p239]

P. Kroonenberg and F. J. Oort. Three-mode analysis of multi-mode covariance matrices. *British Journal of Mathematical and Statistical Psychology*, 56:305 – 336, 2003. URL https://doi.org/10.1348/000711003770480066. [p244]

P. M. Kroonenberg. *Three Mode Principal Component Analysis*. DSWO Press, Leiden, 1983. [p240]

P. M. Kroonenberg. Singular value decompositions of interactions in three-way contigency tables. In R. Coppi and S. Bolasco, editors, *Multiway Data Analysis*, pages 169 – 184. Elsevier, Amsterdam, 1989. [p237]

P. M. Kroonenberg. The TUCKALS line: A suite of programs for three-way data analysis. *Computational Statistics and Data Analysis*, 18:73 – 96, 1994. URL https://doi.org/10.1016/0167-9473(94)90133-3. [p240]

P. M. Kroonenberg. *Applied Multiway Data Analysis*. John Wiley & Sons, Hoboken, NJ, 2008. [p237, 239, 240, 243, 244, 250]

P. M. Kroonenberg and J. D. Leeuw. Principal component analysis of three mode data by means of alternating least squares algorithms. *Psychometrika*, 45:69 – 97, 1980. URL https://doi.org/10.1007/BF02293599. [p240]

H. O. Lancaster. Complex contingency tables treated by the partition of the chi-square. *Journal of Royal Statistical Society, Series B*, 13:242 – 249, 1951. URL https://www.jstor.org/stable/2984066. [p238]

N. C. Lauro and L. D'Ambra. L'analyse non symétrique des correspondances. In E. Diday and et al, editors, *Data Analysis and Informatics III*, pages 433 – 446. Elsevier, Amsterdam, 1984. [p239]

D. Leibovici. Spatio-temporal multiway data decomposition using principal tensor analysis on k-modes: The r package **PTAk**. *Journal of Statistical Software*, 34(10):34 pages, 2010. URL 10.18637/jss.v034.i10. [p245]

J. Li, J. Bien, and M. T. Wells. **rTensor**: An R package for multidimensional array (tensor) unfolding, multiplication, and decomposition. *Journal of Statistical Software*, 87(10):31 pages, 2018. URL DOI:10.18637/jss.v087.i10. [p245]

R. J. Light and H. B. Margolin. An analysis of variance for categorical data. *Journal of the American Statistical Association*, 66:534 – 544, 1971. URL https://doi.org/10.2307/2283520. [p240]

S. Loisel and Y. Takane. Partitions of Pearson's chi-square statistic for frequency tables: A comprehensive account. *Computational Statistics*, 31:1429 – 1452, 2016. URL https://doi.org/10.1007/s00180-015-0619-1. [p239]

R. Lombardo and E. J. Beh. Variants of simple correspondence analysis. *The R Journal*, 8/2:167 – 184, 2016. [p259]

R. Lombardo and E. J. Beh. Three–way correspondence analysis for ordinal–nominal variables. In A. Petrucci and R. Verde, editors, *SIS 2017 Statistics and Data Science: New Challenges, New Generations, 28–30 June 2017, Florence (Italy) Proceedings of the Conference of the Italian Statistical Society*, pages 613 – 620. Firenze Press, 2017. [p240, 242]

R. Lombardo and E. J. Beh. Partitioning the Cressie-Read divergence statistic for three-way contingency tables: a study on environmental sustainability data. In R. Lombardo, I. Camminatiello, and V. Simonacci, editors, *IES 2022 Innovation & Society 5.0: Statistical and Economic Methodologies for Quality Assessment. Book of Short Papers*, pages 491–497. PKE Press, 2022. [p259]

R. Lombardo, A. Carlier, and L. D'Ambra. Nonsymmetric correspondence analysis for three-way contingency tables. *Methodologica*, 4:59 – 80, 1996. [p239, 243]

R. Lombardo, E. J. Beh, and P. M. Kroonenberg. Modelling trends in ordered correspondence analysis using orthogonal polynomials. *Psychometrika*, 81:325 – 349, 2016a. URL https://doi.org/10.1007/s11336-015-9448-y. [p242]

R. Lombardo, P. Kroonenberg, and E. J. Beh. Modelling trends in ordered three-way non-symmetrical correspondence analysis. In M. Pratesi and C. Perna, editors, *Proceedings of the 48th Scientific Meeting of the Italian Statistical Society, June 8-10, 2016*, page 14 pages. Springer, 2016b. [p240, 241]

R. Lombardo, E. J. Beh, and L. Guerrero. Analysis of three-way non-symmetrical association of food concepts in cross-cultural marketing. *Quality & Quality*, 53:2323 – 2337, 2019. URL https://doi.org/10.1007/s11135-018-0733-6. [p237]

R. Lombardo, Y. Takane, and E. J. Beh. Familywise decompositions of Pearson's chi-square statistic in the analysis of contingency tables. *Advances in Data Analysis and Classification*, 14 (3):629 – 649, 2020. URL https://doi.org/10.1007/s11634-019-00374-7. [p238, 239]

R. Lombardo, E. J. Beh, and P. M. Kroonenberg. Symmetrical and non-symmetrical variants of three-way correspondence analysis for ordered variables. *Statistical Science*, 36 (4):542 – 561, 2021. URL https://doi.org/10.1214/20-STS814. [p237, 240, 241, 242, 243, 244, 255, 258]

M. Marcotorchino. Utilisation des comparaisons par paires en statistique des contingencies: Partie i. Etude du Centre Scientifique, IBM, France, No F 069, 1984a. [p239]

M. Marcotorchino. Utilisation des comparaisons par paires en statistique des contingencies: Partie ii. Etude du Centre Scientifique, IBM, France, No F 071, 1984b. [p239]

M. Marcotorchino. Utilisation des comparaisons par paires en statistique des contingencies: Partie iii. Etude du Centre Scientifique, IBM, France, No F 081, 1985. [p239]

T. Murakami and P. M. Kroonenberg. Three-mode models and individual differences in semantic differential data. *Multivariate Behavioral Research*, 38:247 – 283, 2003. URL https://doi.org/10.1207/S15327906MBR3802_5. [p244]

L. Pardo and M. C. Pardo. Minimum power-divergence estimator in three-way contingency tables. *Journal of Statistical Computation and Simulation*, 73:819 – 831, 2003. URL https://doi.org/10.1080/0094965031000097782. [p258]

M. C. Pardo. An empirical investigation of cressie and read tests for the hypothesis of independence in three-way contingency tables. *Kybernetika*, 32:175 – 183, 1996. URL http://hdl.handle.net/10338.dmlcz/124180. [p258]

J. C. W. Rayner and E. J. Beh. Towards a better understanding of correlation. *Statistica Neerlandica*, 63: 324 – 333, 2009. URL https://doi.org/10.1111/j.1467-9574.2009.00425.x. [p242]

W. Revelle. **psych**: Procedures for psychological, psychometric, and personality research, 2018. URL https://CRAN.R-project.org/package=psych. [p245]

T. J. Ringrose. Alternative confidence regions for canonical variate analysis. *Biometrika*, 83:575 – 587, 1996. URL https://doi.org/10.1093/biomet/83.3.575. [p259]

T. J. Ringrose. Bootstrap confidence regions for correspondence analysis. *Journal of Statistical Computation and Simulation*, 82:1397 – 1413, 2012. URL https://doi.org/10.1080/00949655.2011.579968. [p259]

A. Statnikov. **tensorA**: Algebra for tensors, 2018. URL https://CRAN.R-project.org/package=tensorA.Rpackageversion0.36.2. [p245]

Y. Takane and S. Jung. Regularized partial and/or constrained redundancy analysis. *Psychometrika*, 73: 671 – 690, 2008. URL https://doi.org/10.1007/s11336-008-9067-y. [p239]

M. Timmerman and H. A. L. Kiers. Three-mode principal component analysis: Choosing the numbers of components and sensitivity to local optima. *British Journal of Mathematical and Statistical Psychology*, 53:1 – 16, 2000. URL https://doi.org/10.1348/000711000159132. [p244]

L. R. Tucker. Implications of factor analysis of three-way matrices for measurement of change. In C. W. Harris, editor, *Problems in Measuring Change*, pages 122 – 137. University of Wisconsin Press, 1963. [p240]

M. van de Velden, A. I. D'Enza, and F. Palumbo. Cluster correspondence analysis. *Psychometrika*, 82: 158 – 185, 2017. URL https://doi.org/10.1007/s11336-016-9514-0. [p249, 254]

H. van Herk and M. van de Velden. Insight into the relative merits of rating and ranking in a cross-national context using three-way correspondence analysis. *Food Quality and Preference*, 18:1096 – 1105, 2007. URL https://doi.org/10.1016/j.foodqual.2007.05.006. [p237, 246, 247, 248, 249]

X. Zhou. **mvoutlier**: Multivariate outlier detection, 2019. URL https://CRAN.R-project.org/package=mvoutlier.Rpackageversion2.1.1. [p245]

Rosaria Lombardo
Department of Economics, University of Campania "Luigi Vanvitelli"
Capua (CE), Italy
rosaria.lombardo@unicampania.it

Michel van de Velden
Econometric Institute, Erasmus University
Rotterdam, The Netherlands
vandevelden@ese.eur.nl

Eric J. Beh
National Institute for Applied Statistics Research Australia (NIASRA), University of Wollongong
Wollongong, Australia
and
Centre for Multi-Dimensional Data Visualisation (MuViSU)
Stellenbosch University
Stellenbosch, South Africa
ericb@uow.edu.au

Estimating Heteroskedastic and Instrumental Variable Models for Binary Outcome Variables in R

by Mauricio Sarrias

Abstract The objective of this article is to introduce the package Rchoice which provides functionality for estimating heteroskedastic and instrumental variable models for binary outcomes, whith emphasis on the calculation of the average marginal effects. To do so, I introduce two new functions of the Rchoice package using widely known applied examples. I also show how users can generate publication-ready tables of regression model estimates.

1 Introduction

Often, applied researchers in different fields deal with binary (probit and logit) models that exhibit heteroskedasticity (the error variance is not homogeneous across individuals), or with endogenous variables.[1] In both cases, the standard binary logit and probit estimator will be inconsistent, which can lead to misleading conclusions (Yatchew and Griliches 1985; Wooldridge 2010).[2]

One widely used estimator to address heteroskedastic disturbances in the realm of binary outcomes is the fully parametric multiplicative heteroskedastic binary model (Keele and Park 2006). This model assumes that the error term's variance depends on specific known covariates. For example, Alvarez and Brehm (1995) use a heteroskedastic probit model to show that policy choices about abortion are heterogeneous due to unequal variances.[3]

If some of the regressor is endogenous, approaches such as the control function (CF, Wooldridge 2015) or the maximum likelihood estimator (MLE, Newey 1987; Rivers and Vuong 1988) allow to remediate the inconsistent estimates using an instrumental variables (IV) approach.

Routines for heteroskedastic and IV models exist in commercial software such as Stata (StataCorp 2019) and LIMDEP (Greene 2002). One advantage of Stata is that its command `margins` allows such models to quickly and flexibly compute marginal effects. This is very attractive for users who need to produce and export tables of estimates in Latex or other formats.

In this article, I review the main approaches and functions in R to estimate heteroskedastic and IV models for binary outcomes, with a special focus on applied examples and the computation of the marginal effects. Additionally, this article introduces two new functions of the Rchoice package (Sarrias 2016) that allow estimating both types of models. The first function, `hetprob()`, estimates binary dependent variable models assuming a parametric form for the heteroskedasticity. The model can be either the probit or logit model and the parameters are estimated by Maximum Likelihood (ML), which find the parameter values that make the observed data most probable under the assumptions of the statistical model.

The second function, `ivpml()`, estimates binary probit models with endogenous continuous variables using also the ML approach. As an additional feature, Rchoice also provides functions to compute the average marginal effects for both models under different modelling approaches: categorical variables, interactions terms, and quadratic variables. The package can also be used in concert with the **memisc** package (Elff 2012), which produces publication-ready tables of regression model estimates. Finally, I show that both functions produce the same estimates as the corresponding Stata commands.[4]

The function `hetprob()` is intended to complement other related packages in R. For example, the packages **glmx** (Zeileis, Koenker, and Doebler 2015) and **oglmx** (Carroll 2018) also allow to estimate heteroskedastic binary models using MLE. The latter has the advantage of being able to compute the marginal effects. However, the current version does not allow to identify functions of variables that enter the equations for the mean and standard equations, interaction terms, or polynomials. The `ivpml()` function provides the MLE for the probit model and hence complements the R package **ivprobit** (Zaghdoudi 2018) which provides a two-step procedure. Another is the **LARF** package (An

---

[1]In econometrics, endogeneity refers to situations in which an explanatory variable is correlated with the error term. The common sources of endogeneity are omitted variables, simultaneity, and measurement error.

[2]Inconsistency means that the estimator will not converge in probability to the true parameter.

[3]For other applications see Knapp and Seaks (1992) and Williams (2009).

[4]Stata codes for replicating the main results of this article are presented in **Appendix C** and **Appendix D**. Do files are available in the supplemental material.

and Wang 2016), which estimates local averages response functions for binary treatments and binary instruments.

2 Models

2.1 Heterokedastic binary model

The multiplicative heterokedastic binary model (also known as the location-scale binary model) for cross-sectional data has the following structure (Williams 2009):[5]

$$y_i^* = \mathbf{x}_i^\top \boldsymbol{\beta} + \epsilon_i, \tag{1}$$

$$\text{Var}(\epsilon_i|\mathbf{z}_i) = \sigma_i^2 = \sigma_\epsilon^2 \left[\exp\left(\mathbf{z}_i^\top \delta\right)\right]^2, \tag{2}$$

where y_i^* is the latent (unobserved) response variable for individual $i = 1, ..., n$, $\mathbf{x}_i$ is a k-dimensional vector of explanatory variables determining the latent variable y_i^*, $\boldsymbol{\beta}$ is the vector of parameters, and ϵ_i is the error term distributed either normally or logistically with $\mathbf{E}(\epsilon_i|\mathbf{z}_i, \mathbf{x}_i) = 0$ and multiplicative heterokedastic variance $\text{Var}(\epsilon_i|\mathbf{z}_i) = \sigma_i^2, \forall i = 1, ..., n$ (Harvey 1976). The variance for each individual is modeled parametrically assuming that it depends on a p-dimensional vector of observed variables $\mathbf{z}_i$, whereas δ is the vector of coefficients associated with each variable. It is important to emphasize that $\mathbf{z}_i$ does not include a constant, otherwise the parameters are not identified (Greene and Hensher 2010).

Since we do not observe y_i^*, we need a rule that relates the binary variable that we actually observe, y_i, to the latent variable. As it is standard, we use the following rule:

$$y_i = \begin{cases} 1 & \text{if } y_i^* > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Using Equations (1), (2) and (3), the probability of observing $y_i = 1$ is:

$$\Pr(y_i = 1|\mathbf{x}_i, \mathbf{z}_i) = F\left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta}}{\exp(\mathbf{z}_i^\top \delta)}\right), \tag{4}$$

where $F(\cdot)$ is either $\Phi(\cdot)$, that is, the cumulative distribution function (CDF) for the standard normal distribution, such that $\sigma_\epsilon^2 = 1$, or $\Lambda(\cdot) = \frac{\exp(\cdot)}{1+\exp(\cdot)}$, where $\Lambda(\cdot)$ represents the CDF for the standard logistic distribution, so that $\sigma_\epsilon^2 = \pi^2/3$.

Let $\boldsymbol{\theta}$ be the $(k + p)$-dimensional vector of all parameters. The vector $\boldsymbol{\theta}$ can be estimated using the Maximum Likelihood procedure. Using Equation (4), the MLE is the value of the parameters that maximizes the following log-likelihood function:[6]

$$\widehat{\boldsymbol{\theta}}_{ML} \equiv \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmax}} \sum_{i=1}^{n} \ln\left\{\left[1 - F\left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta}}{\exp(\mathbf{z}_i^\top \delta)}\right)\right]^{1-y_i} \left[F\left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta}}{\exp(\mathbf{z}_i^\top \delta)}\right)\right]^{y_i}\right\}.$$

As in any non-linear model, the estimated coefficients alone cannot be interpreted as marginal changes on $\Pr(y_i = 1|\mathbf{x}_i, \mathbf{z}_i)$. Let w_k be a continuous variable appearing in both $\mathbf{x}$ and $\mathbf{z}$, then the partial effect is (see Greene 2003):

$$\frac{\partial \Pr(y_i = 1|\mathbf{x}_i, \mathbf{z}_i)}{\partial w_{ik}} = f\left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta}}{\exp\left(\mathbf{z}_i^\top \delta\right)}\right)\left(\frac{\beta_k - (\mathbf{x}_i^\top \boldsymbol{\beta})\delta_k}{\exp\left(\mathbf{z}_i^\top \delta\right)}\right), \tag{5}$$

where $f(\cdot)$ is the probability density function (PDF) for the standard normal or standard logistic distribution. The average partial effect (APE) can be consistently estimated as follows:

$$\widehat{\text{APE}}_k = \frac{1}{n} \sum_{i=1}^{n} f\left(\frac{\mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}}{\exp\left(\mathbf{z}_i^\top \widehat{\delta}\right)}\right)\left(\frac{\widehat{\beta}_k - (\mathbf{x}_i^\top \widehat{\boldsymbol{\beta}})\widehat{\delta}_k}{\exp\left(\mathbf{z}_i^\top \widehat{\delta}\right)}\right), \tag{6}$$

and their standard error can be estimated either by delta method or bootstrap. The delta method

---

[5]Multiplicative exponential heteroskedasticity was first proposed by Harvey (1976) for linear models. For identification of the multiplicative heterokedastic binary model see Carlson (2019).

[6]The analytic gradient and Hessian for the multiplicative heterokedastic binary model used by **Rchoice** are presented in **Appendix A**.

provides an analytic approximation for the standard errors based on the asymptotic variance-covariance matrix of the MLE. The bootstrap is non-parametric resampling technique, which involves generating a large number of resampled datasets (bootstrap samples) and estimating (6) for each sample. For further details see Wooldridge (2010).

Finally, a likelihood-ratio (LR) or Wald test can be performed to test the null hypothesis of homoskedasticity: $H_0 : \delta = \mathbf{0}$.

2.2 Probit models with endogenous continous variable

Consider the following two-equation model:

$$
\begin{aligned}
y_{1i}^* &= \mathbf{x}_{1i}^\top \boldsymbol{\beta}_1 + \gamma y_{2i} + \epsilon_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \epsilon_i, & (7) \\
y_{2i} &= \mathbf{x}_{1i}^\top \boldsymbol{\delta}_1 + \mathbf{x}_{i2}^\top \boldsymbol{\delta}_2 + v_i = \mathbf{z}_i^\top \boldsymbol{\delta} + v_i, & (8) \\
y_{1i} &= \mathbf{1}\left[y_{1i}^* > 0\right], & (9)
\end{aligned}
$$

where $i = 1, ..., n$, y_{1i}^* is a latent (unobserved) response variable for individual i and we observe $y_{1i} = 1$ if and only if $\mathbf{1}\left[y_{1i}^* > 0\right]$, y_{2i} is the **continuous endogenous** variable, $\mathbf{x}_{i1}$ is a k_1-dimensional vector of predetermined (exogenous) variables, $\mathbf{x}_{i2}$ is a k_2-dimensional vector of additional (exogenous) instruments, $\mathbf{x}_i = \left(\mathbf{x}_{1i}^\top, y_{2i}\right)^\top$ is a $k \times 1$ column vector such that $k = k_1 + 1$, and $\mathbf{z}_i = \left(\mathbf{x}_{1i}^\top, \mathbf{x}_{2i}^\top\right)^\top$ is a $p \times 1$ vector where $p = k_1 + k_2$. Equation (7) is the structural equation, whereas Equation (8) is the first-stage equation. Further, assume that (ϵ, v) are distributed as bivariate normal with zero mean.

Two-step approach

The simplest approach for estimating the parameters of Equation (7) and (8) is using a two-step procedure (Rivers and Vuong 1988) also known as Control Function (CF) approach (Wooldridge 2015). Under joint normality of (ϵ, v), we can write ϵ as a function of v as follows:[7]

$$
\epsilon_i | v_i = \frac{\sigma_\epsilon}{\sigma_v} \rho v_i + \eta_i, \tag{10}
$$

where $\mathrm{Var}(\epsilon_i) = \sigma_\epsilon^2$, $\mathrm{Var}(v_i) = \sigma_v^2$, $\eta_i \sim N\left[0, (1 - \rho^2)\sigma_\epsilon^2\right]$ and $\rho = \mathrm{Cov}(\epsilon_i, v_i)/(\sigma_\epsilon \cdot \sigma_v)$. If $\rho = 0$, y_2 is exogenous and the traditional probit model will deliver consistent estimates. For identification, we need to set $\mathrm{Var}(\epsilon_i) = 1$. Then Equation (10) can be re-written as:

$$
\epsilon_i = \lambda v_i + \eta_i, \tag{11}
$$

where $\eta_i \sim N\left[0, (1 - \rho^2)\right]$ and $\lambda = \mathrm{Cov}(\epsilon_i, v_i)/\sigma_v^2$. Inserting Equation (11) in the latent Equation (7) yields:

$$
y_{1i}^* = \mathbf{x}_{1i}^\top \boldsymbol{\beta}_1 + \gamma y_{2i} + \lambda v_i + \eta_i,
$$

and the probability of observing $y_{1i} = 1$ is:

$$
\Pr(y_{1i} = 1 | y_{2i}, \mathbf{z}_i, v_i) = \Pr(y_{1i}^* > 0 | y_{2i}, \mathbf{z}_i, v_i) = \Phi\left(\mathbf{x}_{1i}^\top \boldsymbol{\beta}_1^* + \gamma^* y_{2i} + \lambda^* v_i\right). \tag{12}
$$

Thus, if we knew v_i, a probit of y_1 on $\mathbf{x}$ and v would consistently estimate the scaled parameters $\boldsymbol{\beta}_1^* = \boldsymbol{\beta}_1 / \sqrt{1 - \rho^2}$, $\gamma^* = \gamma / \sqrt{1 - \rho^2}$, and $\lambda^* = \lambda / \sqrt{1 - \rho^2}$. Using this idea, the estimation procedure is as follows (see Wooldridge 2010, sect. 15.7.2):

- Run an OLS regression of y_2 on $\mathbf{z}$ (Equation (8)) and compute the residuals $\widetilde{v}_i = y_{2i} - \mathbf{z}_i^\top \widetilde{\delta}$. Both $\widetilde{\delta}$ and $\widetilde{v}$ are consistently estimated.
- Run the probit y_1 on $\mathbf{x}_1$, y_2 and $\widetilde{v}$ to get consistent estimators of the scaled coefficients $\boldsymbol{\beta}^*$, γ^* and λ^*.

Note that the term control function comes from the fact that the inclusion of $\widetilde{v}$ in the second step controls for the correlation between ϵ_i and v_i.

Some of the structural parameters can be recovered after the two-step procedure. Since $\sigma_\epsilon = 1$, $\rho = \mathrm{Cov}(\epsilon_i, v_i)/\sigma_v = \lambda \cdot \sigma_v$. Thus, an estimate of ρ can be recovered from:

$$
\widehat{\rho} = \widehat{\lambda}^* \cdot \widetilde{\sigma}_v, \tag{13}
$$

---

[7]If $x \sim N(\mu, \sigma^2)$, then we can write $x_i = \mu + \sigma u_i$, where $u_i \sim N(0, 1)$.

where $\widehat{\lambda}^*$ is the probit estimate of λ^* and $\widetilde{\sigma}_v$ is the square root of the usual error variance estimator from the first-stage regression. The unscaled parameters can also be recovered using the two-stage estimates. For instance, since $\gamma^* = \gamma/\sqrt{(1-\rho^2)}$, and using our result in Equation (13), then $\widehat{\gamma} = \widehat{\gamma}^* \left[1 - \left(\widehat{\lambda}^* \cdot \widetilde{\sigma}_v\right)^2\right]^{1/2}$.

As explained by Wooldridge (2010), the usual probit z-statistic on $\widetilde{v}$ is a valid test of the null hypothesis that y_2 is exogenous: $H_0 : \lambda^* = 0$.[8] However, the estimated variance-covariance matrix of the probit model does not deliver correct standard errors for the rest of the parameters since it does not include the sampling variability of $\widehat{\delta}$ when $\lambda \neq 0$.

Following Wooldridge (2015), the APEs are obtained by taking either derivatives or differences (depending on whether the explanatory variable is continuous or discrete) of the Average Structural Function (ASF) given by:

$$\text{ASF}(\mathbf{x}_1, y_2) = \mathbf{E}_v \left[\Phi \left(\mathbf{x}_{1i}^\top \boldsymbol{\beta}_1^* + \gamma^* y_{2i} + \lambda^* v_i \right) \right]. \tag{14}$$

This function averages out the first-stage residuals v_i, purging the model of endogeneity. Under the weak law of large numbers, a consistent estimator for $\text{ASF}(\mathbf{x}_1, y_2)$ in Equation (14) is:

$$\widehat{\text{ASF}} = \frac{1}{n} \sum_{i=1}^n \Phi \left(\mathbf{x}_{1i}^\top \widehat{\boldsymbol{\beta}}_1^* + \widehat{\gamma}^* y_{2i} + \widehat{\lambda}^* v_i \right), \tag{15}$$

which incorporates the estimated unobservables from the first stage without perturbing them. Hence, to estimate the APE for y_2 we can compute:

$$\widehat{\text{APE}}_{y_2} = \widehat{\gamma}^* \frac{1}{n} \sum_{i=1}^n \phi \left(\mathbf{x}_{1i}^\top \widehat{\boldsymbol{\beta}}_1^* + \widehat{\gamma}^* y_{2i} + \widehat{\lambda}^* v_i \right). \tag{16}$$

where $\phi(\cdot)$ is the standard normal density function. A standard error for this $\widehat{\text{APE}}$ can be obtained via the delta method or bootstrap.

2.3 Maximum Likelihood approach

We can also estimate the parameters using the MLE. To derive the log-likelihood function, we need to find the joint distribution $f(y_{1i}, y_{2i}|\mathbf{z}) = f(y_{1i}|y_{2i}, \mathbf{z}_i) f(y_{2i}|\mathbf{z}_i)$. Under the joint normality, $y_{2i}|\mathbf{z}_i \sim N(\mathbf{z}_i^\top \delta, \sigma_v^2)$ and its conditional marginal density is (Wooldridge 2014):

$$f(y_{2i}|\mathbf{z}_i) = \frac{1}{\sigma_v} \phi \left(\frac{y_{2i} - \mathbf{z}_i^\top \delta}{\sqrt{1 - \rho^2}} \right). \tag{17}$$

Using the fact that the normal distribution is symmetric, the conditional density of y_{2i} given $(y_{2i}, \mathbf{z}_i)$ can be written as:

$$f(y_{1i}|y_{2i}, \mathbf{z}_i) = \Phi \left[q_i \cdot \left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta} + \frac{\rho}{\sigma_v} \left(y_{2i} - \mathbf{z}_i^\top \delta \right)}{\sqrt{1 - \rho^2}} \right) \right], \tag{18}$$

where $q_i = 2y_{2i} - 1$ (see Greene 2003). Using Equations (17) and (18), the joint probability for each individual i is:

$$f(y_{1i}, y_{2i}|\mathbf{z}_i; \boldsymbol{\theta}) = \Phi \left[q_i \cdot \left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta} + \frac{\rho}{\sigma_v} \left(y_{2i} - \mathbf{z}_i^\top \delta \right)}{\sqrt{1 - \rho^2}} \right) \right] \frac{1}{\sigma_v} \phi \left(\frac{y_{2i} - \mathbf{z}_i^\top \delta}{\sqrt{1 - \rho^2}} \right). \tag{19}$$

The MLE is a value of the parameter vector that maximizes the following expression:[9]

$$\widehat{\boldsymbol{\theta}}_{ML} \equiv \underset{\boldsymbol{\theta} \in \Theta}{\arg\max} \sum_{i=1}^n \ln \left\{ \Phi \left[q_i \cdot \left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta} + \frac{\rho}{\sigma_v} \left(y_{2i} - \mathbf{z}_i^\top \delta \right)}{\sqrt{1 - \rho^2}} \right) \right] \frac{1}{\sigma_v} \phi \left(\frac{y_{2i} - \mathbf{z}_i^\top \delta}{\sqrt{1 - \rho^2}} \right) \right\}.$$

[8]Under the null $H_0 : \lambda^* = 0$ it is true that $\epsilon = v$ and therefore the distribution of v does not play any role under the null.

[9]The analytic gradient and Hessian for the MLE used by **Rchoice** are presented in **Appendix B**.

After the parameters are estimated, the APE for the endogenous variable can be estimated as:

$$\widehat{\text{APE}}_{y_2} = \frac{\widehat{\gamma}}{\sqrt{1 - \widehat{\rho}^2}} \frac{1}{n} \sum_{i=1}^{n} \phi \left(\frac{\mathbf{x}_i^\top \widehat{\boldsymbol{\beta}} + \frac{\widehat{\rho}}{\widehat{\sigma}_v} \widehat{v}_i}{\sqrt{1 - \widehat{\rho}^2}} \right). \tag{20}$$

A second option would be to compute the effect for the structural model assuming that endogeneity does not exist (the values of the covariates are given and fixed). In this case, the APE for the endogenous variable is computed as:

$$\widehat{\text{APE}}_{y_2} = \widehat{\gamma} \frac{1}{n} \sum_{i=1}^{n} \phi \left(\mathbf{x}_i^\top \widehat{\boldsymbol{\beta}} \right). \tag{21}$$

3 Applications

3.1 Heteroskedastic binary models

Promotion of scientists

To show how R can be used to fit heteroskedastic binary response models, I first use Allison (1999)'s dataset called "tenure.cvs" (see also Williams 2010). The data consists of observations of the careers of university professors over time, tracking multiple cross-sectional and longitudinal indicators including gender, the number of published article, and quality of department, among others.

We can load the dataset into R as follows:

```
tenure_data <- read.csv(file = 'tenure.csv')
```

Following Allison (1999) and Williams (2009) I focus on whether women get a lower payoff from their published work than men. First, I estimate a binary logit model using the glm() function for men and women separately, where the structural model is given by

$$\text{tenure}^* = \beta_0 + \beta_1 \text{year} + \beta_2 \text{year}^2 + \beta_3 \text{select} + \beta_4 \text{articles} + \beta_5 \text{prestige} + \epsilon,$$
$$\text{tenure} = \mathbf{1} \left[\text{tenure}^* > 0 \right],$$

where ϵ is distributed logistically with mean 0 and variance $\pi^2/3$. The dependent variable, tenure, is whether an assistant professor was promoted in that year, and 0 otherwise, year is the number of years since the beginning of the assistant professorship, select is a measure of undergraduate selectivity of the colleges where scientists received their bachelor's degree, articles is the cumulative number of articles published by the end of each person-year, and prestige is a measure of prestige of the department in which scientist was employed. To obtain similar results as Allison (1999), I restrict the sample to year <= 10. Thus, each person has one record per year of service as an assistant professor, for as many as ten years.

```
sub_data <- subset(tenure_data, year <= 10)
logit_m <- glm(tenure ~  year + I(year^2) + select + articles + prestige,
               subset = (female == 0),
               data   = sub_data,
               family = binomial(link = "logit"))
logit_w <- glm(tenure ~  year + I(year^2) + select + articles + prestige,
               subset = (female == 1),
               data   = sub_data,
               family = binomial(link = "logit"))
```

To present the results I use the mtable() function from **memisc** package (Elff 2012).

```
library("memisc")
mtable("Logit for men" = logit_m,
       "Logit for women" = logit_w,
       summary.stats = c("Log-likelihood", "AIC", "BIC", "N"))

#>
#> Calls:
#> Logit for men: glm(formula = tenure ~ year + I(year^2) + select + articles +
```

```
#>      prestige, family = binomial(link = "logit"), data = sub_data,
#>      subset = (female == 0))
#> Logit for women: glm(formula = tenure ~ year + I(year^2) + select + articles +
#>      prestige, family = binomial(link = "logit"), data = sub_data,
#>      subset = (female == 1))
#>
#> ================================================
#>                    Logit for men   Logit for women
#> ------------------------------------------------
#>    (Intercept)        -7.680***         -5.842***
#>                        (0.681)           (0.866)
#>    year                1.909***          1.408***
#>                        (0.214)           (0.257)
#>    I(year^2)          -0.143***         -0.096***
#>                        (0.019)           (0.022)
#>    select              0.216***          0.055
#>                        (0.061)           (0.072)
#>    articles            0.074***          0.034**
#>                        (0.012)           (0.013)
#>    prestige           -0.431***         -0.371*
#>                        (0.109)           (0.156)
#> ------------------------------------------------
#>    Log-likelihood    -526.545          -306.191
#>    AIC               1065.090           624.382
#>    BIC               1097.863           654.155
#>    N                 1741               1056
#> ================================================
#>    Significance: *** = p < 0.001; ** = p < 0.01;
#>                  * = p < 0.05
```

From previous output, it can be noticed that the coefficient of articles for men is approximately twice as large as for women: 0.074 vs 0.034. One possible conclusion we could draw from this result is that women suffer from discrimination. That is, the return per additional article on the propensity to get a promotion is on average lower for women, holding other things constant. However, Allison (1999) notes that this result might be due to variance error term differences. For example, women might have more heterogeneous career patterns than men due to unobserved factors affecting promotion. In particular, assume that we have the following model for men (M) and women (W):

$$y_{iM}^* = \mathbf{x}_{iM}^\top \boldsymbol{\beta} + \epsilon_{iM},$$
$$y_{iW}^* = \mathbf{x}_{iW}^\top \boldsymbol{\beta} + \epsilon_{iW},$$
$$\epsilon_{iM} \sim \Lambda(0, \sigma_M^2),$$
$$\epsilon_{iW} \sim \Lambda(0, \sigma_W^2),$$

where $\Lambda(\cdot)$ is the logistic CDF. Both men and women have the same coefficients, $\boldsymbol{\beta}$, in the propensity to be promoted, but different scales, $\sigma_M^2 \neq \sigma_W^2$. Note that the logit model identifies $\beta = \frac{\alpha}{\sigma}$. Thus, if women have greater variance than men, $\sigma_W > \sigma_M$, their coefficient will be smaller, assuming similar return to productivity. To allow for such possibility, Williams (2009) suggests fitting a heteroskedastic logit (HET-Logit) model where the standard deviation of the error term is modeled as

$$\sigma_i = \exp(\delta \cdot \text{female}_i).$$

This model can be estimated in R using the hetglm() function from **glmx** package or hetprob() function from **Rchoice** package. The syntax to fit the model using hetprob() is the following

```
library("Rchoice")
het_logit <- hetprob(tenure ~ factor(female) + year + I(year^2) + select +
                     articles + prestige | factor(female),
                 data = sub_data,
                 link = "logit")
```

Similarly to hetglm() function, the formula argument of hetprob() has the form y ~ x | z, where y is the binary response variable, x are the explanatory covariates, and z are the covariates affecting the variance of the error term. The argument link indicates whether a logit (link = "logit") or probit (link = "probit") model should be fitted.

The output is the following:

```
summary(het_logit)

#> ------------------------------------------------------------------
#> Maximum Likelihood estimation of Heteroskedastic Binary model
#> Newton-Raphson maximisation, 4 iterations
#> Return code 8: successive function values within relative tolerance limit (reltol)
#> Log-Likelihood: -836.2824
#> 8  free parameters
#>
#> Estimates for the mean:
#>                  Estimate Std. error  z value   Pr(> z)
#> (Intercept)     -7.490505   0.659663 -11.3551 < 2.2e-16 ***
#> factor(female)1 -0.939190   0.370524  -2.5348 0.0112524 *
#> year             1.909544   0.199694   9.5624 < 2.2e-16 ***
#> I(year^2)       -0.139687   0.016943  -8.2448 < 2.2e-16 ***
#> select           0.181920   0.052657   3.4548 0.0005507 ***
#> articles         0.063534   0.010219   6.2173 5.059e-10 ***
#> prestige        -0.446207   0.096904  -4.6046 4.132e-06 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Estimates for lnsigma:
#>                    Estimate Std. error z value Pr(> z)
#> het.factor(female)1  0.30223    0.14618  2.0675 0.03868 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> LR test of lnsigma = 0: chi2 4.5 with 1 df. Prob > chi2 =  0.0339
#> ------------------------------------------------------------------
```

The results using hetglm() are the following

```
library("glmx")
het_glmx <- hetglm(tenure ~ factor(female) + year + I(year^2) + select +
                       articles + prestige | factor(female),
                     data = sub_data,
                   family = binomial(link = "logit"))
summary(het_glmx)

#>
#> Call:
#> hetglm(formula = tenure ~ factor(female) + year + I(year^2) + select +
#>    articles + prestige | factor(female), data = sub_data, family = binomial(link = "logit"))
#>
#> Deviance residuals:
#>    Min     1Q  Median     3Q    Max
#> -1.8473 -0.5666 -0.2926 -0.1149  3.3397
#>
#> Coefficients (binomial model with logit link):
#>                  Estimate Std. Error z value Pr(>|z|)
#> (Intercept)     -7.490489   0.648517 -11.550  < 2e-16 ***
#> factor(female)1 -0.939174   0.364357  -2.578 0.009948 **
#> year             1.909540   0.199095   9.591  < 2e-16 ***
#> I(year^2)       -0.139686   0.016762  -8.334  < 2e-16 ***
#> select           0.181919   0.051916   3.504 0.000458 ***
#> articles         0.063534   0.009884   6.428  1.3e-10 ***
#> prestige        -0.446207   0.097083  -4.596  4.3e-06 ***
#>
#> Latent scale model coefficients (with log link):
#>                  Estimate Std. Error z value Pr(>|z|)
#> factor(female)1   0.3022     0.1433   2.109   0.0349 *
#> ---
```

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Log-likelihood: -836.3 on 8 Df
#> LR test for homoscedasticity: 4.501 on 1 Df, p-value: 0.03387
#> Dispersion: 1
#> Number of iterations in nlminb optimization: 7
```

Although the coefficients estimated by both functions are very similar, their standard errors are somewhat different. One potential explanation for this difference is the optimization algorithm used by each function. hetprob() uses Newton-Raphson algorithm available in maxLik() function from **maxLik** package (Henningsen and Toomet 2011), whereas hetglm() uses nlminb algorithm as default.

Now, I compare the logit and Het-Logit estimates using mtable() function.[10] The following output presents the estimates.

```
mtable("Logit for men"   = logit_m,
       "Logit for women" = logit_w,
       "Heteroskedastic" = het_logit,
       summary.stats = c("Log-likelihood", "AIC", "BIC", "N"))

#>
#> Calls:
#> Logit for men: glm(formula = tenure ~ year + I(year^2) + select + articles +
#>     prestige, family = binomial(link = "logit"), data = sub_data,
#>     subset = (female == 0))
#> Logit for women: glm(formula = tenure ~ year + I(year^2) + select + articles +
#>     prestige, family = binomial(link = "logit"), data = sub_data,
#>     subset = (female == 1))
#> Heteroskedastic: hetprob(formula = tenure ~ factor(female) + year + I(year^2) +
#>     select + articles + prestige | factor(female), data = sub_data,
#>     link = "logit", method = "nr")
#>
#> ========================================================================
#>                 Logit for men  Logit for women   Heteroskedastic
#>                 -----------    -----------      -------------------
#>                   tenure         tenure          mean     lnsigma
#> ------------------------------------------------------------------------
#>   (Intercept)     -7.680***      -5.842***       -7.491***
#>                   (0.681)        (0.866)         (0.660)
#>   year             1.909***       1.408***        1.910***
#>                   (0.214)        (0.257)         (0.200)
#>   I(year^2)       -0.143***      -0.096***       -0.140***
#>                   (0.019)        (0.022)         (0.017)
#>   select           0.216***       0.055           0.182***
#>                   (0.061)        (0.072)         (0.053)
#>   articles         0.074***       0.034**         0.064***
#>                   (0.012)        (0.013)         (0.010)
#>   prestige        -0.431***      -0.371*         -0.446***
#>                   (0.109)        (0.156)         (0.097)
#>   factor(female)1                                -0.939*    0.302*
#>                                                  (0.371)   (0.146)
#> ------------------------------------------------------------------------
#>   Log-likelihood  -526.545       -306.191        -836.282
#>   AIC             1065.090        624.382        1688.565
#>   BIC             1097.863        654.155        1736.055
#>   N               1741           1056            2797
#> ========================================================================
#>   Significance: *** = p < 0.001; ** = p < 0.01; * = p < 0.05
```

The estimated coefficients for the HET-Logit model indicate that being a woman increases the variance of the error term ($\widehat{\delta} = 0.302$) and decreases the propensity to be promoted ($\widehat{\beta}_6 = -0.939$).

Using the estimate $\widehat{\delta}$, we can also compute how much the disturbance standard deviation differ by gender. Note that the standard deviation of the error term for women is $\sigma_W = \exp(0.302)$, whereas for men is $\sigma_M = \exp(0) = 1$. Then,

---

[10]mtable() does not support objects of class hetglm.

```
sigma_w <- exp(coef(het_logit)["het.factor(female)1"])
(1 - sigma_w) / sigma_w

#> het.factor(female)1
#>          -0.2608322
```

This result implies that the standard deviation of the disturbance for men is 26% lower than the standard deviation for women. Conversely, this also means that the standard deviation of the residuals is $\exp(0.302) = 1.35$ times larger for women compared to men (Williams 2009, 2010). The 95%-CI for this ratio can be computed using the delta method by `deltaMethod()` function from **car** package (Fox, Friendly, and Weisberg 2013):

```
library("car")
sharef <- "(1 - exp(`het.factor(female)1`)) / exp(`het.factor(female)1`)"
deltaMethod(het_logit, sharef)

#>                                                              Estimate      SE
#> (1 - exp(`het.factor(female)1`))/exp(`het.factor(female)1`) -0.26083  0.10805
#>                                                                2.5 %  97.5 %
#> (1 - exp(`het.factor(female)1`))/exp(`het.factor(female)1`) -0.47261 -0.0491
```

So far, the HET-Logit estimates suggest that there are gender differences in both the dependent variable and in the variance of the error term. However, the estimated coefficients do not allow us to conclude whether women have a lower return than men for productivity. To give some insights about this question, I estimate a HET-Logit model including the interaction between `female` and `articles` in the choice equation:

```
het_logit2 <- hetprob(tenure ~ factor(female) + year + I(year^2) + select +
                        articles + prestige + factor(female)*articles |
                        factor(female),
                    data = sub_data,
                    link = "logit")
print(summary(het_logit2), digits = 3)

#> ----------------------------------------------------------------
#> Maximum Likelihood estimation of Heteroskedastic Binary model
#> Newton-Raphson maximisation, 4 iterations
#> Return code 1: gradient close to zero (gradtol)
#> Log-Likelihood: -835.1335
#> 9  free parameters
#>
#> Estimates for the mean:
#>                         Estimate Std. error z value Pr(> z)
#> (Intercept)             -7.3653      0.6547  -11.25 < 2e-16 ***
#> factor(female)1         -0.3781      0.4500   -0.84   0.401
#> year                     1.8383      0.2029    9.06 < 2e-16 ***
#> I(year^2)               -0.1343      0.0170   -7.89 3.1e-15 ***
#> select                   0.1700      0.0517    3.29   0.001 **
#> articles                 0.0720      0.0114    6.31 2.8e-10 ***
#> prestige                -0.4205      0.0961   -4.37 1.2e-05 ***
#> factor(female)1:articles -0.0305     0.0187   -1.63   0.104
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Estimates for lnsigma:
#>                    Estimate Std. error z value Pr(> z)
#> het.factor(female)1    0.177      0.163    1.09    0.28
#>
#> LR test of lnsigma = 0: chi2 1.22 with 1 df. Prob > chi2 =  0.2684
#> ----------------------------------------------------------------
```

The coefficient for `female * articles` is not statistically significant when residual variation by gender is involved. As argued by Allison (1999), this result proposes dissimilarities in productivity
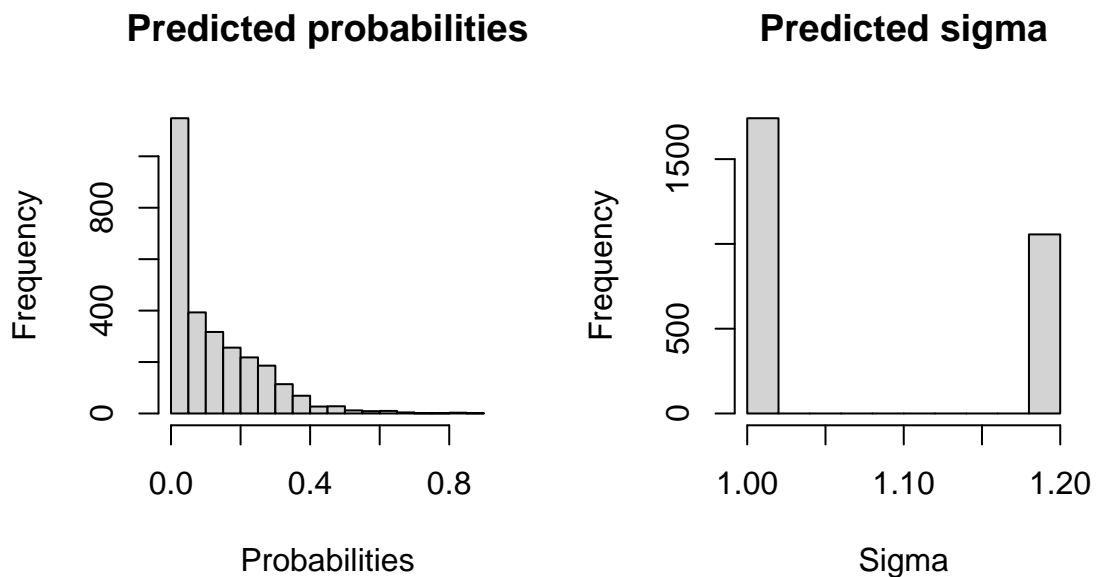
Predicted probabilities   **Predicted sigma**

Figure 1: Distribution of predicted probability and predicted sigma

returns between males and females resulting from variability in unobserved factors rather than discriminatory influences.

Once we have fitted a model, we can use the predict() command to obtain the predicted probability and the predicted scale factor, $\widehat{\sigma}_i$, which can be readily used for visualization as shown in Figure 1. The following lines plots the distribution of both measures:

```
par(mfrow = c(1, 2))
hist(predict(het_logit2, type = "pr"),
    main = "Predicted probabilities",
    xlab = "Probabilities")
hist(predict(het_logit2, type = "sigma"),
    main = "Predicted sigma",
    xlab = "Sigma")
```

An additional feature of **Rchoice** package is that it allows to estimate the APEs for heteroskedastic binary models, as in Equation (6), using effect() function. Similarly to command margins() from **margins** package (Leeper 2021) or avg_slopes() from **marginaleffects** package (Arel-Bundock 2023), this function takes into account whether the variables are continuous, categorical or both. The user must specify categorical variables using factor() in the formula argument; otherwise, the effect() function will assume that the variable is continuous, when the variable may already be a factor in the dataset. In the following lines, we compute the APEs for a HET-Probit and HET-Logit model.[11] The results are the following:

```
eff_logit  <- effect(het_logit2)
het_probit <- hetprob(tenure ~ factor(female) + year + I(year^2) + select +
                        articles + prestige + factor(female)*articles |
                        factor(female),
                    data = sub_data,
                    link = "probit")
eff_probit <- effect(het_probit)
mtable(eff_probit,
       eff_logit)


#>
#> Calls:
#> eff_probit: hetprob(formula = tenure ~ factor(female) + year + I(year^2) +
#>     select + articles + prestige + factor(female) * articles |
#>     factor(female), data = sub_data, link = "probit", method = "nr")
```

---

[11]The Jacobian matrix is computed numerically using jacobian() function from **numDeriv** package (Gilbert and Varadhan 2019).

```
#> eff_logit: hetprob(formula = tenure ~ factor(female) + year + I(year^2) +
#>     select + articles + prestige + factor(female) * articles |
#>     factor(female), data = sub_data, link = "logit", method = "nr")
#>
#> ==========================================
#>                 eff_probit    eff_logit
#> ------------------------------------------
#>   factor(female)1    -0.031**     -0.031**
#>                      (0.012)      (0.012)
#>   year                0.032***     0.032***
#>                      (0.002)      (0.002)
#>   select              0.014***     0.015***
#>                      (0.004)      (0.004)
#>   articles            0.006***     0.005***
#>                      (0.001)      (0.001)
#>   prestige           -0.035***    -0.036***
#>                      (0.008)      (0.008)
#> ------------------------------------------
#>   Log-likelihood  -832.478     -835.133
#>   N                2797         2797
#> ==========================================
#>   Significance: *** = p < 0.001;
#>                  ** = p < 0.01; * = p < 0.05
```

The APEs are very close to each other and statistically significant. According to the HET-Probit estimates, one additional published article increases the probability of being promoted by 0.6 percent points, whereas being a woman decreases the probability of promoted by 3.1%.

Labor participation

Our second example is a replication of Greene (2003)'s example 17.7 based on the dataset "mroz.cvs". This dataset is based on a cross-section data on the wages of 428 working, married women, originating from the 1976 Panel Study of Income Dynamics (PSID), which can be loaded as follows:

```
mroz <- read.csv(file = 'mroz.csv')
mroz$kids <- with(mroz, factor((kidslt6 + kidsge6) > 0,
                               levels = c(FALSE, TRUE),
                               labels = c("no", "yes")))
mroz$finc <-  mroz$faminc / 10000
```

Using this data, Greene (2003) estimates the following HET-Probit model for women labor participation:

$$\text{inlf}^* = \beta_0 + \beta_1\text{age} + \beta_2\text{age}^2 + \beta_3\text{finc} + \beta_4\text{educ} + \beta_5\text{kids} + \epsilon, \tag{22}$$

$$\epsilon \sim N(0, \sigma_i^2), \tag{23}$$

$$\sigma_i = \exp(\delta_1\text{kids} + \delta_2\text{finc}), \tag{24}$$

where inlf is a dummy variable indicating whether the woman participates in labor force, age is age in year, finc is family income in 1975 dollars divided by 10,000, educ is education in year and kids indicates whether children under 18 are present in the household. It is further assumed that kids and finc affect the variability of the error term.

The probit, Het-Probit and average marginal effects are estimated as follows:[12]

```
labor_hom <- glm(inlf ~  age + I(age^2) + finc + educ + factor(kids),
                 data = mroz,
                 family = binomial(link = "probit"))
labor_het <- hetprob(inlf ~  age + I(age^2) + finc + educ + factor(kids) |
                       factor(kids) + finc,
                     data = mroz,
                     link = "probit")
eff_labor_het <- effect(labor_het)
```

---

[12]Greene (2003) computes the marginal effects at the mean instead of the average marginal effects.

```
mtable(labor_hom,
       labor_het,
       eff_labor_het)
```

```
#>
#> Calls:
#> labor_hom: glm(formula = inlf ~ age + I(age^2) + finc + educ + factor(kids),
#>     family = binomial(link = "probit"), data = mroz)
#> labor_het: hetprob(formula = inlf ~ age + I(age^2) + finc + educ + factor(kids) |
#>     factor(kids) + finc, data = mroz, link = "probit", method = "nr")
#> eff_labor_het: hetprob(formula = inlf ~ age + I(age^2) + finc + educ + factor(kids) |
#>     factor(kids) + finc, data = mroz, link = "probit", method = "nr")
#>
#> ========================================================================
#>                          labor_hom     labor_het      eff_labor_het
#>                          -----------  ------------------  -----------
#>                             inlf        mean    lnsigma      inlf
#> ------------------------------------------------------------------------
#>    (Intercept)            -4.157**    -6.030*
#>                           (1.404)     (2.498)
#>    age                     0.185**     0.264*               -0.009***
#>                           (0.066)     (0.118)               (0.003)
#>    I(age^2)               -0.002**    -0.004*
#>                           (0.001)     (0.001)
#>    finc                    0.046       0.424    0.313*        0.069**
#>                           (0.043)     (0.222)  (0.123)       (0.024)
#>    educ                    0.098***    0.140**               0.030***
#>                           (0.023)     (0.052)               (0.009)
#>    factor(kids): yes/no   -0.449***   -0.879** -0.141        -0.161***
#>                           (0.130)     (0.303)  (0.324)       (0.043)
#> ------------------------------------------------------------------------
#>    Log-likelihood         -490.848    -487.636              -487.636
#>    N                       753         753                   753
#> ========================================================================
#>    Significance: *** = p < 0.001; ** = p < 0.01; * = p < 0.05
```

The results show that family income does not play any role in the choice equation. However, it increases the variability of the error term. APE indicates that an increase of $10,000 of family income increases the probability of labor force involvement by 6.9%. There is not enough statistical evidence that proves having children under 18 in the household produces heteroskedasticity.

We can also use the Wald test provided by linearHypothesis() function from **car** package to test the null hypothesis of homoskedasticity:

```
coefs <- names(coef(labor_het))
linearHypothesis(labor_het, coefs[grep("het", coefs)])
```

```
#> Linear hypothesis test
#>
#> Hypothesis:
#> het.factor(kids)yes = 0
#> het.finc = 0
#>
#> Model 1: restricted model
#> Model 2: inlf ~ age + I(age^2) + finc + educ + factor(kids) | factor(kids) +
#>     finc
#>
#>   Df  Chisq Pr(>Chisq)
#> 1
#> 2   2 6.5331    0.03814 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The null hypothesis of homoskedasticity is rejected at the 5% with a $\chi_2^2 = 6.533$.

Supplementary materials provide Stata code (version 16.1) to replicate all the results in this Section. The log file is presented in **Appendix C**. Overall, the results using Stata are exactly the same to those reported by `hetprob()` function from **Rchoice** package.

3.2 Instrumental variable probit model

Control function approach

In this example, and similar to Wooldridge (2010), we use the `mroz` sample and assume the following slightly modified model for married women's labor force participation from previous Section:

$$\text{inlf}^* = \beta_0 + \beta_1\text{educ} + \beta_2\text{exper} + \beta_3\text{exper}^2 + \beta_4\text{age} + \beta_5\text{kidslt6} +$$
$$\beta_6\text{kidsge6} + \beta_7\text{nwifeinc} + \epsilon,$$
$$\text{nwifeinc} = \delta_0 + \delta_1\text{educ} + \delta_2\text{exper} + \delta_3\text{exper}^2 + \delta_4\text{age} + \delta_5\text{kidslt6} +$$
$$\delta_6\text{kidsge6} + \delta_7\text{huseduc} + v,$$
$$\text{lfp} = \mathbf{1}\left[\text{lfp}^* > 0\right]$$

where `nwifeinc` is the other sources of income (divided by 1,000) and assumed to be endogenous. A just identified IV model is estimated by using husband's education, (`huseduc`), as an instrument for `nwifeinc`. The strong identification assumption here is that husband's schooling is unrelated to factors that affect a married woman's labor force decision once `nwifeinc` and the other variables are accounted for (Wooldridge 2010).

When interpreting the results from an IV model, it is important to compare its magnitude with a model that assumes exogeneity. In this example, our benchmark APE for `nwifeinc` is obtained by the standard probit model:

```
probit <- glm(inlf ~  educ + exper + I(exper^2) + age + kidslt6 + kidsge6 + nwifeinc,
              data = mroz,
              family = binomial(link = "probit"))
ape.probit <- mean(dnorm(predict(probit, type = "link"))) * coef(probit)["nwifeinc"]
ape.probit


#>     nwifeinc
#> -0.003616176
```

Accordingly, an increase of $1,000 in other sources of income reduces the labor force participation probability by 0.4%, holding all other factors constant. Note that the same APE, along with its standard error, can also be obtained using `avg_slopes()` command:

```
library("marginaleffects")
avg_slopes(probit, variables = "nwifeinc")


#>
#>      Term Estimate Std. Error    z Pr(>|z|)   S    2.5 %    97.5 %
#>  nwifeinc -0.00362    0.00147 -2.46   0.0139 6.2  -0.0065 -0.000736
#>
#> Columns: term, estimate, std.error, statistic, p.value, s.value, conf.low, conf.high
#> Type:  response
```

I proceed to estimate the model using the CF approach. First, I estimate the first-step equation, which is a linear model, and obtain the residuals $\tilde{v}$:

```
fstep   <- lm(nwifeinc ~ educ + exper + I(exper^2) + age + kidslt6 + kidsge6  + huseduc,
              data = mroz)
mroz$res.hat <- fstep$residuals
```

We can also test the power of the instrument using `linearHypothesis()` function:

```
linearHypothesis(fstep, "huseduc")
```

```
#> Linear hypothesis test
#>
#> Hypothesis:
#> huseduc = 0
#>
#> Model 1: restricted model
#> Model 2: nwifeinc ~ educ + exper + I(exper^2) + age + kidslt6 + kidsge6 +
#>     huseduc
#>
#>   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
#> 1    746  86955
#> 2    745  81120  1    5834.8 53.586 6.427e-13 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The first-stage F statistic on huseduc is substantially above the traditional cut-off of ten suggesting that the instrument is not weak.

The second-step is computed using glm() function and adding the residuals (res.hat) as an additional explanatory variable:

```
sstep  <- glm(inlf ~ educ + exper + I(exper^2) + age + kidslt6 + kidsge6 + nwifeinc + res.hat,
         data  = mroz,
         family = binomial(link = "probit"))
summary(sstep)
```

```
#>
#> Call:
#> glm(formula = inlf ~ educ + exper + I(exper^2) + age + kidslt6 +
#>     kidsge6 + nwifeinc + res.hat, family = binomial(link = "probit"),
#>     data = mroz)
#>
#> Deviance Residuals:
#>     Min      1Q   Median      3Q      Max
#> -2.2523  -0.9078   0.4204   0.8566   2.2803
#>
#> Coefficients:
#>              Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  0.0171183  0.5380339   0.032  0.97462
#> educ         0.1702142  0.0377615   4.508 6.56e-06 ***
#> exper        0.1163118  0.0193869   6.000 1.98e-09 ***
#> I(exper^2)  -0.0019458  0.0005999  -3.244  0.00118 **
#> age         -0.0449529  0.0101351  -4.435 9.19e-06 ***
#> kidslt6     -0.8444319  0.1197268  -7.053 1.75e-12 ***
#> kidsge6      0.0477912  0.0449431   1.063  0.28761
#> nwifeinc    -0.0368639  0.0183848  -2.005  0.04495 *
#> res.hat      0.0267092  0.0191539   1.394  0.16318
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 1029.75  on 752  degrees of freedom
#> Residual deviance:  800.61  on 744  degrees of freedom
#> AIC: 818.61
#>
#> Number of Fisher Scoring iterations: 4
```

Since the z-statistic for res.hat is 1.4, we cannot reject the null hypothesis that nwifeinc is exogenous: $H_0 : \lambda = 0$.

An estimate of ρ can be obtained using Equation (13) and the following syntax:

```
lambda.hat    <- coef(sstep)["res.hat"]
k             <- length(fstep$coefficients)
```

```
SSE           <- sum(fstep$residuals^2)
n             <- length(fstep$residuals)
sigma.upsilon <- sqrt(SSE/(n - k))
rho.hat       <- lambda.hat * sigma.upsilon
rho.hat

#>   res.hat
#> 0.2787068
```

Thus, the estimated correlation using the CF approach is $\hat{\rho} = 0.279$. It is important to recall that the estimated coefficients for the sstep model represent the coefficients scaled by a factor of $1/\sqrt{1 - \rho^2}$. Moreover, the standard errors from the sstep model are biased since they do not consider the sampling error of the first stage. However, we can use ivprobit() function from **ivprobit** package (Zaghdoudi 2018) to get the correct standard errors:[13]

```
library("ivprobit")
twostep.probit <- ivprobit(inlf ~  educ + exper + I(exper^2) + age + kidslt6 + kidsge6 |
                             nwifeinc | educ + exper + I(exper^2) + age + kidslt6 +
                             kidsge6 + huseduc,
                    data = mroz)
summary(twostep.probit)

#>                  Coef        S.E.  t-stat     p-val
#> Intercep    0.01711834  0.54865782  0.0312  0.975118
#> educ        0.17021419  0.03848938  4.4224 1.121e-05 ***
#> exper       0.11631183  0.01976301  5.8853 6.001e-09 ***
#> I(exper^2) -0.00194584  0.00061195 -3.1798  0.001535 **
#> age        -0.04495285  0.01032548 -4.3536 1.526e-05 ***
#> kidslt6    -0.84443188  0.12176581 -6.9349 8.818e-12 ***
#> kidsge6     0.04779117  0.04578807  1.0437  0.296940
#> nwifeinc   -0.03686390  0.01874338 -1.9668  0.049580 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimates of the sstep and twostep.probit models are the same, while their standard errors are slightly different.

The APE for nwifeinc—and any other continuous variable—can be computed using Equation (16) and its standard error via bootstrap method. Below, I use package **boot** (Canty 2002) to perform the simulation. First, a function called ape() is created which returns the APE. The first argument of this function is the dataset, whereas the second argument can be an index vector of the observations in the dataset.

```
ape <- function(data, indices){
  d <- data[indices, ]
  # Compute the first-stage regression
  fstep    <- lm(nwifeinc ~ educ + exper + I(exper^2) + age + kidslt6 + kidsge6 +
                   huseduc,
              data = d)
  # Obtain the residuals
  d$res.hat <- fstep$residuals
  # Compute the second-stage regression
  sstep    <- glm(inlf ~  educ + exper + I(exper^2) + age + kidslt6 + kidsge6 +
                   nwifeinc + res.hat,
              data   = d,
              family = binomial(link = "probit"))
  # Compute APE for nwincome
  out <- mean(dnorm(predict(sstep, type = "link"))) * coef(sstep)["nwifeinc"]
  return(out)
}
```

Once we have defined the function ape(), we can use the boot() function to perform the bootstrap procedure. In the following syntax, R = 500 resamplings are used and the 90%-CI interval is obtained using boot.ci() function.

---

[13]ivprobit() uses a minimum chi-squared estimator (Newey 1987).

```
library("boot")
set.seed(666)
results <- boot(data = mroz, statistic = ape, R = 500)
results

#>
#> ORDINARY NONPARAMETRIC BOOTSTRAP
#>
#>
#> Call:
#> boot(data = mroz, statistic = ape, R = 500)
#>
#>
#> Bootstrap Statistics :
#>       original        bias     std. error
#> t1* -0.0110576 -0.0005050597 0.005877061
```

```
boot.ci(results, type = "norm", conf = 0.90)

#> BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
#> Based on 500 bootstrap replicates
#>
#> CALL :
#> boot.ci(boot.out = results, conf = 0.9, type = "norm")
#>
#> Intervals :
#> Level      Normal
#> 90%   (-0.0202, -0.0009 )
#> Calculations and Intervals on Original Scale
```

The results show that another $1,000 in other sources of income reduces the labor force participation probability by 1.1 percent points with 90%-CI $[-2, -.09]$. This estimate, which is marginally statistically significant, is about three times larger than the probit estimate that treats `nwifeinc` as exogenous: -0.04.

Finally, we can recover the unscaled parameters by multiplying the coefficients by $\sqrt{(1 - \widehat{\rho}^2)}$ as follows:

```
coef(sstep) * sqrt(1 - rho.hat^2)

#>  (Intercept)         educ        exper    I(exper^2)         age       kidslt6
#>  0.016440046  0.163469667  0.111703116 -0.001868741 -0.043171653 -0.810972324
#>     kidsge6     nwifeinc      res.hat
#>  0.045897507 -0.035403215  0.025650873
```

Maximum likelihood estimator

In this Section I estimate the model from previous Section using the MLE. To do so, I use the `ivpml()` function from **Rchoice** package. The syntax is as follows:

```
fiml.probit <- ivpml(inlf ~  educ + exper + I(exper^2) + age + kidslt6 + kidsge6 +
                       nwifeinc | huseduc +  educ + exper + I(exper^2) + age +
                       kidslt6 + kidsge6,
                  data = mroz)

#>
#> Estimating a just identified model....
#>
#> Obtaining starting values from probit and linear model...
```

The syntax of `ivpml()` is similar to that of `ivreg()` function from **AER** package. The `formula` has two part in the right-hand side, that is, y ~ x | z where y is the binary response variable, x are the regressors (x in Equation (7)), and z are the exogenous variables ($\mathbf{x}_1$ and $\mathbf{x}_2$ in Equation (8)).

During the optimization procedure, `ivpml()` displays several messages which can be turned-off by setting `messages = FALSE`. The output indicates that the model is just-identified and that the initial values for the optimization procedure are obtained from the traditional probit and linear models for the structural and first-stage equation, respectively. Similarly to `hetprob()` function, the optimization algorithm can be managed using the argument `method`, which is passed on to the `maxLik()` function. Currently, the default algorithm is the Newton-Raphson, `method = "nr"`.

```
summary(fiml.probit)
```

```
#> -------------------------------------------
#> Maximum Likelihood estimation of IV Probit model
#> Newton-Raphson maximisation, 3 iterations
#> Return code 8: successive function values within relative tolerance limit (reltol)
#> Log-Likelihood: -3230.642
#> 18  free parameters
#> Estimates:
#>                        Estimate  Std. error z value   Pr(> z)
#> inlf:(Intercept)       1.6499e-02  5.3008e-01   0.0311 0.9751702
#> inlf:educ              1.6403e-01  3.1225e-02   5.2531 1.495e-07 ***
#> inlf:exper             1.1209e-01  2.1199e-02   5.2873 1.241e-07 ***
#> inlf:I(exper^2)       -1.8751e-03  5.9150e-04  -3.1701 0.0015237 **
#> inlf:age              -4.3319e-02  1.1331e-02  -3.8230 0.0001319 ***
#> inlf:kidslt6          -8.1375e-01  1.2994e-01  -6.2623 3.794e-10 ***
#> inlf:kidsge6           4.6054e-02  4.3139e-02   1.0676 0.2857141
#> inlf:nwifeinc         -3.5524e-02  1.6190e-02  -2.1941 0.0282247 *
#> nwifeinc:(Intercept) -1.4720e+01  3.7672e+00  -3.9076 9.322e-05 ***
#> nwifeinc:huseduc       1.1782e+00  1.6009e-01   7.3594 1.847e-13 ***
#> nwifeinc:educ          6.7469e-01  2.1254e-01   3.1744 0.0015016 **
#> nwifeinc:exper        -3.1299e-01  1.3752e-01  -2.2760 0.0228480 *
#> nwifeinc:I(exper^2)   -4.7756e-04  4.4955e-03  -0.1062 0.9153983
#> nwifeinc:age           3.4015e-01  5.9390e-02   5.7274 1.020e-08 ***
#> nwifeinc:kidslt6       8.2627e-01  8.1402e-01   1.0151 0.3100812
#> nwifeinc:kidsge6       4.3553e-01  3.2027e-01   1.3599 0.1738728
#> lnsigma                2.3398e+00  2.5768e-02  90.8016 < 2.2e-16 ***
#> atanhrho               2.7379e-01  1.9296e-01   1.4189 0.1559361
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Instrumented: nwifeinc
#> Instruments: (Intercept) huseduc educ exper I(exper^2) age kidslt6 kidsge6
#> Wald test of exogeneity (corr = 0): chi2 2.01 with 1 df. Prob > chi2 =  0.1559
#> -------------------------------------------
```

During the optimization procedure the parameters σ_v and ρ might tend to the boundary points of the parameter space, generating identifiability problems of the MLE. To avoid this issue, `ivpml()` re-parametrizes the parameters.[14] First, to ensure $\sigma_v > 0$, `ivpml()` instead estimates $\log \nu_v$ such that:

$$\sigma_v = \exp(\log \nu_v). \tag{25}$$

Second, `ivpml()` forces the correlation to remain in the $(-1, +1)$ range by using the inverse hyperbolic tangent:

$$\mathrm{atanh}(\rho) = \tau = \frac{1}{2} \log \left(\frac{1+\rho}{1-\rho} \right),$$

where τ is unrestricted, and ρ can be obtained using the inverse of τ:

$$\tau^{-1} = \rho = \tanh(\tau). \tag{26}$$

In the following syntax, we recover σ_v and ρ using Equations (25) and (26), respectively, and their standard errors are computed using delta method approach by `deltaMethod()` function:

```
deltaMethod(fiml.probit, "exp(lnsigma)")
```

---
[14]This re-parametrization is also used by `ivprobit` function in Stata.

```
#>              Estimate      SE    2.5 % 97.5 %
#> exp(lnsigma) 10.37928  0.26746  9.85508 10.903
```

```
deltaMethod(fiml.probit, "tanh(atanhrho)")
```

```
#>               Estimate      SE     2.5 % 97.5 %
#> tanh(atanhrho) 0.267145  0.179190 -0.084061 0.6184
```

Again, the FIML estimate of ρ is close to that found using the CF approach which was 0.279. If significant, a positive ρ would indicate that there is a positive correlation between ϵ and v. That is, the unobserved factors that make it more likely for a woman to have a higher income from other sources also make it more likely that the woman will be participating in the labor force.

For those users who are more familiar with Stata (see **Appendix D**), it is important to mention that its function ivprobit estimates the 95%-CI for $\hat{\rho}$ and $\hat{\sigma}_v$ as follows:

```
cbind(exp(coef(fiml.probit)["lnsigma"] - qnorm(0.975) * stdEr(fiml.probit)["lnsigma"]),
     exp(coef(fiml.probit)["lnsigma"] + qnorm(0.975) * stdEr(fiml.probit)["lnsigma"]))
```

```
#>            [,1]     [,2]
#> lnsigma 9.868094 10.91695
```

```
cbind(tanh(coef(fiml.probit)["atanhrho"] - qnorm(0.975) * stdEr(fiml.probit)["atanhrho"]),
    tanh(coef(fiml.probit)["atanhrho"] + qnorm(0.975) * stdEr(fiml.probit)["atanhrho"]))
```

```
#>             [,1]      [,2]
#> atanhrho -0.1040317 0.5730038
```

The APEs can be estimated using the function effect(). The main argument of this function is asf. If asf = TRUE (the default), then the APEs are computed using Equation (20). On the other hand, if asf = FALSE the APEs are computed using Equation (21).

```
summary(effect(fiml.probit))
```

```
#> -------------------------------------------------------
#> Marginal effects for the IV Probit model:
#> -------------------------------------------------------
#>              dydx Std. error z value  Pr(> z)
#> educ     0.051057  0.011101   4.599 4.24e-06 ***
#> exper    0.023071  0.002952   7.816 5.44e-15 ***
#> age     -0.013484  0.002986  -4.516 6.31e-06 ***
#> kidslt6 -0.253295  0.033077  -7.658 1.89e-14 ***
#> kidsge6  0.014335  0.013520   1.060   0.2890
#> nwifeinc -0.011058  0.005550  -1.992   0.0463 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Note: Marginal effects computed as the average for each individual
```

```
summary(effect(fiml.probit, asf = FALSE))
```

```
#> -------------------------------------------------------
#> Marginal effects for the IV Probit model:
#> -------------------------------------------------------
#>              dydx Std. error z value  Pr(> z)
#> educ     0.048777  0.008733   5.585 2.33e-08 ***
#> exper    0.021997  0.003723   5.908 3.46e-09 ***
#> age     -0.012882  0.003322  -3.878 0.000105 ***
#> kidslt6 -0.241982  0.036594  -6.613 3.78e-11 ***
#> kidsge6  0.013695  0.012792   1.071 0.284373
#> nwifeinc -0.010564  0.004736  -2.230 0.025724 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Note: Marginal effects computed as the average for each individual
```

The results show that both APEs are close to each other. Note also that the estimated APE for `nwifeinc` using the CF approach is very similar to that ones obtained by MLE. **Appendix D** also shows that the Stata function `ivprobit()` provides the same estimates and marginal effects as `ivpml()` function.

4 Summary

The aim of the article was to provide a primer on estimating heteroskedastic and IV model for binary outcomes in R. I also show that the current version of **Rchoice** package (available at https://cran.r-project.org/web/packages/Rchoice/index.html) allows to estimate such models in a flexible way and provides accurate average marginal effects that are very similar to those provided by Stata's `margins` command. **Rchoice** can be used in concert with other packages. For example, one can format the summary output from **Rchoice** with **memisc** to produce well-formatted tables for regression estimates

5 Appendix A: Gradient and Hessian for binary response models with heteroskedasticity

In this section, I provide the analytic gradient and Hessian used by `hetprob()` function in **Rchoice**. The log-likelihood function for the binary choice model with exponential heteroskedasticity can be written as:

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ln F(a_i),$$

where $F(\cdot)$ is either the CDF of the standard normal or standard logistic distribution, $\boldsymbol{\theta} = \left(\boldsymbol{\beta}^{\top}, \boldsymbol{\delta}^{\top} \right)^{\top}$ is the full $(k + p)$-dimensional vector of parameters, and:

$$a_i = q_i \left(\frac{\mathbf{x}_i^{\top} \boldsymbol{\beta}}{\exp(\mathbf{z}_i^{\top} \boldsymbol{\delta})} \right),$$

$$q_i = 2(y_i - 1).$$

The gradient is:

$$\frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{i=1}^{n} \left[\frac{f(a_i)}{F(a_i)} \frac{\partial a_i}{\partial \boldsymbol{\theta}} \right],$$

$$= \sum_{i=1}^{n} \left[m(a_i) \mathbf{g}_i \right],$$

where $m(\cdot) = f(\cdot)/F(\cdot) = \phi(\cdot)/\Phi(\cdot)$ for the probit model and $m(\cdot) = 1 - \Lambda(\cdot)$ for the logit model, and $\partial a_i / \partial \boldsymbol{\theta} = \mathbf{g}_i$ such that:

$$\mathbf{g}_i = \begin{pmatrix} \frac{\partial a_i}{\partial \boldsymbol{\beta}} \\ \frac{\partial a_i}{\partial \boldsymbol{\delta}} \end{pmatrix} = \frac{q_i}{\exp(\mathbf{z}_i^{\top} \boldsymbol{\delta})} \begin{pmatrix} \mathbf{x}_i \\ - \left(\mathbf{x}_i^{\top} \boldsymbol{\beta} \right) \mathbf{z}_i \end{pmatrix}.$$

The Hessian is given by:

$$\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^{\top}} = \frac{\partial}{\partial \boldsymbol{\theta}^{\top}} \left(\frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right),$$

$$= \sum_{i=1}^{n} \left[h(a_i) \mathbf{g}_i \mathbf{g}_i^{\top} + m(a_i) \mathbf{H}_i \right],$$

where $h(a_i) = \partial m(a_i)/\partial a_i = -a_i m(a_i) - m(a_i)^2$ and:

$$\mathbf{H}_i = \frac{\partial a_i}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^{\top}} = \begin{pmatrix} \frac{\partial a_i}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^{\top}} & \frac{\partial a_i}{\partial \boldsymbol{\beta} \partial \boldsymbol{\delta}^{\top}} \\ \frac{\partial a_i}{\partial \boldsymbol{\delta} \partial \boldsymbol{\beta}^{\top}} & \frac{\partial a_i}{\partial \boldsymbol{\delta} \partial \boldsymbol{\delta}^{\top}} \end{pmatrix} = \begin{pmatrix} \mathbf{O} & -\frac{q_i}{\exp(\mathbf{z}_i^{\top} \boldsymbol{\delta})} \mathbf{x}_i \mathbf{z}_i^{\top} \\ -\frac{q_i}{\exp(\mathbf{z}_i^{\top} \boldsymbol{\delta})} \mathbf{z}_i \mathbf{x}_i^{\top} & \frac{q_i (\mathbf{x}_i^{\top} \boldsymbol{\beta})}{\exp(\mathbf{z}_i^{\top} \boldsymbol{\delta})} \mathbf{z}_i \mathbf{z}_i^{\top} \end{pmatrix}.$$

6 Appendix B: Gradient and Hessian for binary response models with endogeneity

In this section, I provide the analytic gradient and Hessian used by ivpml function in **Rchoice**. The log-likelihood function can be written as:

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left[\ln\left(\Phi(a_i)\right) + \ln(1) - \ln\left(\sigma_v\right) + \ln\left[\phi(b_i)\right] \right],$$

where $\boldsymbol{\theta} = \left(\boldsymbol{\beta}^\top, \boldsymbol{\delta}^\top, \sigma_v, \rho\right)^\top$ is an $(k + p + 2)$-dimensional vector and:

$$a_i = q_i \left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta} + \frac{\rho}{\sigma_v}\left(y_{2i} - \mathbf{z}_i^\top \boldsymbol{\delta}\right)}{\sqrt{1 - \rho^2}} \right),$$

$$b_i = \frac{y_{2i} - \mathbf{z}_i^\top \boldsymbol{\delta}}{\sigma_v},$$

$$q_i = 2(y_i - 1),$$

$$\sigma_v = \exp(\ln v_v),$$

$$\rho = \tanh(\tau).$$

The first derivatives of the log-likelihood function are:

$$\frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} \left[m(a_i) \left(\frac{q_i}{\sqrt{1 - \rho^2}} \right) \mathbf{x}_i \right],$$

$$\frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\delta}} = \sum_{i=1}^{n} \left[-m(a_i) \left(\frac{q_i\left(\rho/\sigma_v\right)}{\sqrt{1 - \rho^2}} \right) + b_i \left(\frac{1}{\sigma_v} \right) \right] \mathbf{z}_i,$$

$$\frac{\partial \ell(\boldsymbol{\theta})}{\partial \ln v_v} = \sum_{i=1}^{n} \left[-m(a_i) \frac{q_i \rho}{\sqrt{1 - \rho^2}} b_i + b_i^2 - 1 \right],$$

$$\frac{\partial \ell(\boldsymbol{\theta})}{\partial \tau} = \sum_{i=1}^{n} \left[m(a_i) q_i \left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta} \rho + b_i}{\sqrt{\text{sech}^2(\tau)}} \right) \right],$$

where $m(a_i) = \phi(a_i)/\Phi(a_i)$, $d\tanh(\tau)/d\tau = \text{sech}^2(\tau)$, and we use the fact that $\phi'(b_i) = -b_i\phi(b_i)$ so that $\phi'(b_i)/\phi(b_i) = -b_i$.

The Hessian is given by:

$$\mathbf{H} = \begin{pmatrix} \frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} & \frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\delta}^\top} & \frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\beta} \partial \ln v_v} & \frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\beta} \partial \tau} \\ \cdot & \frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\delta} \partial \boldsymbol{\delta}^\top} & \frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\delta} \partial \ln v_v} & \frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\delta} \partial \tau} \\ \cdot & \cdot & \frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial (\ln v_v)^2} & \frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \ln v_v \partial \tau} \\ \cdot & \cdot & \cdot & \frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \tau^2} \end{pmatrix}.$$

The second derivatives are:

$$\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} = \sum_{i=1}^{n} \left[h(a_i) \left(\frac{q_i}{\sqrt{1-\rho^2}} \right)^2 \mathbf{x}_i \mathbf{x}_i^\top \right],$$

$$\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\delta}^\top} = \sum_{i=1}^{n} \left[-h(a_i) \left(\frac{q_i}{\sqrt{1-\rho^2}} \right)^2 \left(\frac{\rho}{\sigma_v} \right) \mathbf{x}_i \mathbf{z}_i^\top \right],$$

$$\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\beta} \partial \ln \nu_v} = \sum_{i=1}^{n} \left[-h(a_i) \left(\frac{q_i}{\sqrt{1-\rho^2}} \right)^2 (\rho b_i) \mathbf{x}_i \right],$$

$$\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\beta} \partial \tau} = \sum_{i=1}^{n} \left[h(a_i) \left(\frac{q_i}{\sqrt{1-\rho^2}} \right) \left(q_i \left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta} \rho + b_i}{\sqrt{\operatorname{sech}^2(\tau)}} \right) \right) \mathbf{x}_i \right],$$

$$\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\delta} \partial \boldsymbol{\delta}^\top} = \sum_{i=1}^{n} \left[h(a_i) \left(\frac{q_i (\rho/\sigma_v)}{\sqrt{1-\rho^2}} \right)^2 - \frac{1}{\sigma_v^2} \right] \mathbf{z}_i \mathbf{z}_i^\top,$$

$$\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\delta} \partial \ln \nu_v} = \sum_{i=1}^{n} \left(\frac{b_i}{\sigma_v} \right) \left[h(a_i) \left(\frac{q_i \rho}{\sqrt{1-\rho^2}} \right)^2 - 2 \right] \mathbf{z}_i,$$

$$\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\delta} \partial \tau} = \sum_{i=1}^{n} \left[-h(a_i) \left(\frac{q_i (\rho/\sigma_v)}{\sqrt{1-\rho^2}} \right) \left(q_i \left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta} \rho + b_i}{\sqrt{\operatorname{sech}^2(\tau)}} \right) \right) \right] \mathbf{z}_i,$$

$$\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial (\ln \nu_v)^2} = \sum_{i=1}^{n} \left[h(a_i) \left(\frac{q_i \rho b_i}{\sqrt{1-\rho^2}} \right)^2 + m(a_i) \left(\frac{q_i \rho b_i}{\sqrt{1-\rho^2}} \right) - 2b_i^2 \right],$$

$$\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \ln \nu_v \partial \tau} = \sum_{i=1}^{n} \left\{ -b_i \left[h(a_i) \frac{q_i \rho}{\sqrt{1-\rho^2}} q_i \left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta} \rho + b_i}{\sqrt{\operatorname{sech}^2(\tau)}} \right) + m(a_i) \frac{q_i}{\sqrt{\operatorname{sech}^2(\tau)}} \right] \right\},$$

$$\frac{\partial^2 \ell(\boldsymbol{\theta})}{\partial \tau^2} = \sum_{i=1}^{n} \left[h(a_i) \left(q_i \frac{\mathbf{x}_i^\top \boldsymbol{\beta} \rho + b_i}{\sqrt{\operatorname{sech}^2(\tau)}} \right)^2 + q_i m(a_i) \frac{\mathbf{x}_i^\top \boldsymbol{\beta} + b_i \rho}{\sqrt{\operatorname{sech}^2(\tau)}} \right],$$

where $h(a_i) = -a_i m(a_i) - m(a_i)^2$.

7 Appendix C: Stata code for heteroskedastic binary response models

```
 *=======================================
 *** Example 1: Promotion of scientists ***
 *=======================================
. import delimited "$dir/tenure.csv", clear
(23 vars, 2,945 obs)

. ** Logit models for men and women
. quietly eststo logit_m: logit tenure year c.year#c.year select ///
                  articles prestige if (year <= 10 & female == 0)

. quietly eststo logit_w: logit tenure year c.year#c.year select ///
                  articles prestige if (year <= 10 & female == 1)

. esttab logit_m logit_w, b(3) se(3)
------------------------------------------
                     (1)             (2)
                  tenure          tenure
------------------------------------------
tenure
year               1.909***        1.408***
                  (0.214)         (0.257)
c.year#c.y~r      -0.143***       -0.096***
                  (0.019)         (0.022)
```

```
select                0.216***          0.055
                     (0.061)          (0.072)
articles              0.074***          0.034**
                     (0.012)          (0.013)
prestige             -0.431***         -0.371*
                     (0.109)          (0.156)
_cons                -7.680***         -5.842***
                     (0.681)          (0.866)
----------------------------------------
N                      1741             1056
----------------------------------------
Standard errors in parentheses
* p<0.05, ** p<0.01, *** p<0.001

. ** Heterokedastic logit model
. quietly ssc install oglm

. quietly eststo het_logit: oglm tenure i.female year c.year#c.year select ///
                      articles prestige if (year <= 10), hetero(i.female) link(logit)

. esttab logit_m logit_w het_logit, b(3) se(3)
------------------------------------------------------------
                       (1)              (2)              (3)
                      tenure           tenure           tenure
------------------------------------------------------------
tenure
year                  1.909***          1.408***          1.910***
                     (0.214)          (0.257)          (0.200)
c.year#c.y~r         -0.143***         -0.096***         -0.140***
                     (0.019)          (0.022)          (0.017)
select                0.216***          0.055             0.182***
                     (0.061)          (0.072)          (0.053)
articles              0.074***          0.034**           0.064***
                     (0.012)          (0.013)          (0.010)
prestige             -0.431***         -0.371*           -0.446***
                     (0.109)          (0.156)          (0.097)
0.female                                                  0.000
                                                         (.)
1.female                                                 -0.939*
                                                         (0.371)
_cons                -7.680***         -5.842***
                     (0.681)          (0.866)
------------------------------------------------------------
lnsigma
0.female                                                  0.000
                                                         (.)
1.female                                                  0.302*
                                                         (0.146)
------------------------------------------------------------
cut1
_cons                                                     7.491***
                                                         (0.660)
------------------------------------------------------------
N                      1741             1056             2797
------------------------------------------------------------
Standard errors in parentheses
* p<0.05, ** p<0.01, *** p<0.001

. ** Testing how much the disturbance standard deviation differ by gender
. margins, expression((1 - exp([lnsigma]_b[1.female])) / exp([lnsigma]_b[1.female]))
Warning: expression() does not contain predict() or xb().
Warning: prediction constant over observations.

Predictive margins                           Number of obs    =       2,797
```

```
Model VCE    : OIM
Expression   : (1 - exp([lnsigma]_b[1.female])) / exp([lnsigma]_b[1.female])


         ------------------------------------------------------------------------------
                      |             Delta-method
                      |    Margin   Std. Err.      z    P>|z|    [95% Conf. Interval]
         -------------+----------------------------------------------------------------
              _cons |  -.2608323   .1080501    -2.41   0.016    -.4726065   -.0490581
         ------------------------------------------------------------------------------


. ** Heterokedastic logit model 2
. eststo het_logit2: oglm tenure i.female year c.year#c.year select ///
        articles prestige i.female#c.articles if (year <= 10), hetero(i.female) link(logit)

Heteroskedastic Ordered Logistic Regression      Number of obs    =       2,797
                                                 LR chi2(8)       =      415.39
                                                 Prob > chi2      =      0.0000
Log likelihood = -835.13347                      Pseudo R2        =      0.1992


         ------------------------------------------------------------------------------
             tenure |     Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
         -----------+------------------------------------------------------------------
tenure             |
         1.female |  -.3780598   .4500207    -0.84   0.401   -1.260084    .5039645
             year |   1.838257   .2029491     9.06   0.000    1.440484     2.23603
                   |
    c.year#c.year |  -.1342828    .017024    -7.89   0.000   -.1676492   -.1009165
                   |
           select |   .1699659   .0516643     3.29   0.001    .0687057    .2712261
         articles |   .0719821   .0114106     6.31   0.000    .0496178    .0943464
         prestige |  -.4204742   .0961206    -4.37   0.000    -.608867   -.2320813
                   |
female#c.articles |
                1 |  -.0304836   .0187427    -1.63   0.104   -.0672185    .0062514
         -----------+------------------------------------------------------------------
lnsigma            |
         1.female |   .1774193   .1627087     1.09   0.276   -.1414839    .4963226
         -----------+------------------------------------------------------------------
             /cut1 |   7.365286   .6547121    11.25   0.000    6.082074    8.648498
         ------------------------------------------------------------------------------


. ** Plot predicted probability and sigma
. predict phat, pr outcome(1)
. predict sigmahat, sigma

. hist phat
(bin=34, start=2.232e-12, width=.02503351)

. hist sigmahat
(bin=34, start=1, width=.00570976)


. ** Average Marginal Effects for logit and probit heterokedastic models
. quietly oglm tenure i.female year c.year#c.year select ///
        articles prestige i.female#c.articles if (year <= 10), hetero(i.female) link(probit)

. eststo eff_probit: margins, dydx(*) predict(outcome(1)) post
Average marginal effects                         Number of obs    =       2,797
Model VCE    : OIM
Expression   : Pr(tenure==1), predict(outcome(1))
dy/dx w.r.t. : 1.female year select articles prestige
         ------------------------------------------------------------------------------
                      |             Delta-method
                      |     dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
         -------------+----------------------------------------------------------------
```

```
   1.female |   -.031161   .0115614    -2.70   0.007    -.053821    -.0085011
       year |    .031839   .0019586    16.26   0.000     .0280002    .0356779
     select |   .0142546   .0041796     3.41   0.001     .0060626    .0224465
   articles |     .00559   .0007685     7.27   0.000     .0040838    .0070962
   prestige |  -.0350608   .0077056    -4.55   0.000    -.0501635   -.0199581
------------------------------------------------------------------------------
Note: dy/dx for factor levels is the discrete change from the base level.

. quietly oglm tenure i.female year c.year#c.year select ///
      articles prestige i.female#c.articles if (year <= 10), hetero(i.female) link(logit)

. eststo eff_logit: margins, dydx(*) predict(outcome(1)) post
Average marginal effects                        Number of obs     =      2,797
Model VCE     : OIM
Expression    : Pr(tenure==1), predict(outcome(1))
dy/dx w.r.t.  : 1.female year select articles prestige
------------------------------------------------------------------------------
            |            Delta-method
            |     dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
------------+-----------------------------------------------------------------
   1.female |  -.0312105   .0115836    -2.69   0.007     -.053914    -.008507
       year |   .0319057   .0019277    16.55   0.000     .0281275    .0356839
     select |   .0145808   .0042388     3.44   0.001      .006273    .0228886
   articles |   .0053378   .0007523     7.09   0.000     .0038632    .0068123
   prestige |  -.0360711    .007934    -4.55   0.000    -.0516215   -.0205207
------------------------------------------------------------------------------
Note: dy/dx for factor levels is the discrete change from the base level.

. esttab eff_probit eff_logit, b(3) se(3)
----------------------------------------
                      (1)          (2)

----------------------------------------
0.female            0.000        0.000
                      (.)          (.)
1.female           -0.031**     -0.031**
                    (0.012)      (0.012)
year                0.032***     0.032***
                    (0.002)      (0.002)
select              0.014***     0.015***
                    (0.004)      (0.004)
articles            0.006***     0.005***
                    (0.001)      (0.001)
prestige           -0.035***    -0.036***
                    (0.008)      (0.008)
----------------------------------------
N                    2797         2797
----------------------------------------
Standard errors in parentheses
* p<0.05, ** p<0.01, *** p<0.001

. *=======================================
. *** Example 2: Labor Participation ***
. *=======================================
.
. * Open dataset and create variables
. import delimited "$dir/mroz.csv", clear
. gen kids = (kidslt6 + kidsge6) > 0
. gen finc = faminc/10000

. * Hetekedastic binary probit model
. quietly eststo labor_hom: probit inlf age c.age#c.age finc educ kids
. quietly eststo labor_het: oglm inlf age c.age#c.age finc educ i.kids, ///
                      hetero(finc i.kids) link(probit)
```

```
. quietly eststo eff_labor_het: margins, dydx(*) predict(outcome(1)) post
. esttab labor_hom labor_het eff_labor_het, b(3) se(3)
```

	(1) inlf	(2) inlf	(3)
main			
age	0.185**	0.264*	-0.009***
	(0.066)	(0.118)	(0.003)
c.age#c.age	-0.002**	-0.004*	
	(0.001)	(0.001)	
finc	0.046	0.424	0.069**
	(0.042)	(0.222)	(0.024)
educ	0.098***	0.140**	0.030***
	(0.023)	(0.052)	(0.009)
kids	-0.449***		
	(0.131)		
0.kids		0.000	0.000
		(.)	(.)
1.kids		-0.879**	-0.161***
		(0.303)	(0.043)
_cons	-4.157**		
	(1.402)		
lnsigma			
finc		0.313*	
		(0.123)	
0.kids		0.000	
		(.)	
1.kids		-0.141	
		(0.324)	
cut1			
_cons		6.030*	
		(2.498)	
N	753	753	753

Standard errors in parentheses
* p<0.05, ** p<0.01, *** p<0.001

```
. * Wald test
. estimates restore labor_het
(results labor_het are active now)

. quietly oglm
. test [lnsigma]: finc 1.kids

 ( 1)  [lnsigma]finc = 0
 ( 2)  [lnsigma]1.kids = 0

        chi2(  2) =    6.53
      Prob > chi2 =    0.0381
```

8   Appendix D: Stata code for binary response models with endogeneity

```
. **********************************
. *** IV Probit ***
. **********************************


. *======================================
. *** Control function approach ***
```

```
. *=======================================

. import delimited "$dir/mroz.csv", clear
(22 vars, 753 obs)

. * Probit estimates and marginal effect
. probit inlf educ exper c.exper#c.exper age kidslt6 kidsge6 nwifeinc

Iteration 0:   log likelihood =  -514.8732
Iteration 1:   log likelihood = -402.06651
Iteration 2:   log likelihood = -401.30273
Iteration 3:   log likelihood = -401.30219
Iteration 4:   log likelihood = -401.30219

Probit regression                               Number of obs    =       753
                                                LR chi2(7)       =    227.14
                                                Prob > chi2      =    0.0000
Log likelihood = -401.30219                     Pseudo R2        =    0.2206
------------------------------------------------------------------------------
         inlf |     Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
--------------+---------------------------------------------------------------
         educ |  .1309047   .0252542     5.18   0.000     .0814074    .180402
        exper |  .1233476   .0187164     6.59   0.000     .0866641   .1600311
              |
c.exper#c.exper | -.0018871      .0006    -3.15   0.002     -.003063  -.0007111
              |
          age | -.0528527   .0084772    -6.23   0.000    -.0694678  -.0362376
      kidslt6 | -.8683285   .1185223    -7.33   0.000    -1.100628   -.636029
      kidsge6 |   .036005   .0434768     0.83   0.408     -.049208   .1212179
     nwifeinc | -.0120237   .0048398    -2.48   0.013    -.0215096  -.0025378
        _cons |  .2700768    .508593     0.53   0.595    -.7267473   1.266901
------------------------------------------------------------------------------

. margins, dydx(nwifeinc)
Average marginal effects                        Number of obs    =       753
Model VCE    : OIM
Expression   : Pr(inlf), predict()
dy/dx w.r.t. : nwifeinc


------------------------------------------------------------------------------
             |            Delta-method
             |     dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    nwifeinc | -.0036162   .0014414    -2.51   0.012    -.0064413  -.0007911
------------------------------------------------------------------------------

. * Control function approach
. ststo fstep: reg nwifeinc educ exper c.exper#c.exper age kidslt6 kidsge6 huseduc

      Source |       SS           df       MS      Number of obs   =       753
-------------+----------------------------------   F(7, 745)       =     27.13
       Model |  20676.7705         7   2953.82436   Prob > F        =    0.0000
    Residual |  81120.3451       745   108.886369   R-squared       =    0.2031
-------------+----------------------------------   Adj R-squared   =    0.1956
       Total |  101797.116       752   135.368505   Root MSE        =    10.435
------------------------------------------------------------------------------
     nwifeinc |     Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
--------------+---------------------------------------------------------------
         educ |  .6746951   .2136829     3.16   0.002     .2552029   1.094187
        exper | -.3129877   .1382549    -2.26   0.024    -.5844034  -.0415721
              |
c.exper#c.exper | -.0004776   .0045196    -0.11   0.916    -.0093501     .008395
              |
          age |  .3401521   .0597084     5.70   0.000     .2229354   .4573687
```

```
     kidslt6 |   .8262719    .8183785     1.01   0.313    -.7803305    2.432874
     kidsge6 |   .4355289    .3219888     1.35   0.177    -.1965845    1.067642
     huseduc |   1.178155    .1609449     7.32   0.000     .8621956    1.494115
       _cons |  -14.72048    3.787326    -3.89   0.000    -22.15559   -7.285383
-------------------------------------------------------------------------------


. predict res_hat, resi


. test huseduc
 ( 1)  huseduc = 0
       F(  1,    745) =   53.59
             Prob > F =    0.0000


. ststo sstep: probit inlf educ exper c.exper#c.exper age kidslt6 kidsge6 nwifeinc res_hat


Iteration 0:   log likelihood =  -514.8732
Iteration 1:   log likelihood = -401.13728
Iteration 2:   log likelihood = -400.30361
Iteration 3:   log likelihood = -400.30301
Iteration 4:   log likelihood = -400.30301


Probit regression                               Number of obs     =         753
                                                LR chi2(8)        =      229.14
                                                Prob > chi2       =      0.0000
Log likelihood = -400.30301                     Pseudo R2         =      0.2225
-------------------------------------------------------------------------------
        inlf |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+-----------------------------------------------------------------
        educ |   .1702153    .0376718     4.52   0.000     .0963798    .2440507
       exper |   .1163123    .0193312     6.02   0.000     .0784239    .1542007
             |
c.exper#c.exper |  -.0019459   .0006009    -3.24   0.001    -.0031235   -.0007682
             |
         age |   -.044953    .0101367    -4.43   0.000    -.0648206   -.0250855
     kidslt6 |  -.8444363    .1198154    -7.05   0.000     -1.07927   -.6096025
     kidsge6 |   .0477905    .0443204     1.08   0.281    -.0390758    .1346568
    nwifeinc |  -.0368641    .0182706    -2.02   0.044    -.0726738   -.0010543
     res_hat |   .0267093    .0189352     1.41   0.158    -.0104031    .0638217
       _cons |   .0171187    .5392914     0.03   0.975    -1.039873     1.07411
-------------------------------------------------------------------------------


. * Two-step IV-probit
. ivprobit inlf educ exper c.exper#c.exper age kidslt6 kidsge6 (nwifeinc = huseduc), twostep
Checking reduced-form model...


Two-step probit with endogenous regressors       Number of obs     =         753
                                                  Wald chi2(7)      =      173.79
                                                  Prob > chi2       =      0.0000
-------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+-----------------------------------------------------------------
    nwifeinc |  -.0368641    .0186314    -1.98   0.048    -.0733809   -.0003472
        educ |   .1702153    .0384014     4.43   0.000     .0949499    .2454806
       exper |   .1163123    .0197084     5.90   0.000     .0776846      .15494
             |
c.exper#c.exper |  -.0019459   .0006129    -3.17   0.001    -.0031471   -.0007446
             |
         age |   -.044953     .010327    -4.35   0.000    -.0651936   -.0247125
     kidslt6 |  -.8444363    .1218529    -6.93   0.000     -1.083264    -.605609
     kidsge6 |   .0477905     .045177     1.06   0.290    -.0407549    .1363359
       _cons |   .0171187    .5498911     0.03   0.975    -1.060648    1.094885
-------------------------------------------------------------------------------
Instrumented:  nwifeinc
Instruments:   educ exper c.exper#c.exper age kidslt6 kidsge6 huseduc
```

```
--------------------------------------------------------------------------------
Wald test of exogeneity: chi2(1) = 1.99                    Prob > chi2 = 0.1584


. *======================================
. *** MLE ***
. *======================================


. ivprobit inlf educ exper c.exper#c.exper age kidslt6 kidsge6 (nwifeinc = huseduc)

Fitting exogenous probit model

Iteration 0:   log likelihood =  -514.8732
Iteration 1:   log likelihood = -401.13728
Iteration 2:   log likelihood = -400.30361
Iteration 3:   log likelihood = -400.30301
Iteration 4:   log likelihood = -400.30301


Fitting full model

Iteration 0:   log likelihood = -3230.6635
Iteration 1:   log likelihood = -3230.6421
Iteration 2:   log likelihood = -3230.6421

Probit model with endogenous regressors      Number of obs    =        753
                                             Wald chi2(7)     =     200.50
Log likelihood = -3230.6421                  Prob > chi2      =     0.0000
--------------------------------------------------------------------------------
                     |    Coef.   Std. Err.     z    P>|z|    [95% Conf. Interval]
---------------------+----------------------------------------------------------
            nwifeinc | -.0355243   .0161904   -2.19   0.028   -.0672569   -.0037916
                educ |  .1640289   .0312249    5.25   0.000    .1028293    .2252285
               exper |   .112085   .0211991    5.29   0.000    .0705356    .1536344
                     |
     c.exper#c.exper | -.0018751   .0005915   -3.17   0.002   -.0030345   -.0007158
                     |
                 age | -.0433193   .0113314   -3.82   0.000   -.0655284   -.0211101
             kidslt6 | -.8137458   .1299442   -6.26   0.000   -1.068432   -.5590599
             kidsge6 |  .0460536   .0431386    1.07   0.286   -.0384966    .1306037
               _cons |  .0164965   .5300821    0.03   0.975   -1.022445    1.055438
---------------------+----------------------------------------------------------
 corr(e.nwifeinc,e.inlf)|  .2671475   .1791903                 -.1040303    .5730063
        sd(e.nwifeinc)|  10.37928   .2674576                  9.868095    10.91695
--------------------------------------------------------------------------------
Instrumented:  nwifeinc
Instruments:   educ exper c.exper#c.exper age kidslt6 kidsge6 huseduc
--------------------------------------------------------------------------------
Wald test of exogeneity (corr = 0): chi2(1) = 2.01        Prob > chi2 = 0.1559

. ststo me1: margins, dydx(*) predict(pr) post
Average marginal effects                      Number of obs    =        753
Model VCE    : OIM
Expression   : Average structural function probabilities, predict(pr)
dy/dx w.r.t. : nwifeinc educ exper age kidslt6 kidsge6
--------------------------------------------------------------------------------
            |           Delta-method
            |    dy/dx   Std. Err.     z    P>|z|    [95% Conf. Interval]
------------+-------------------------------------------------------------
   nwifeinc | -.0110576   .0055497   -1.99   0.046   -.0219348   -.0001805
       educ |  .0510572   .0111011    4.60   0.000    .0292994     .072815
      exper |  .0230711   .0029517    7.82   0.000    .0172858    .0288563
        age |  -.013484    .002986   -4.52   0.000   -.0193365   -.0076314
    kidslt6 | -.2532945   .0330766   -7.66   0.000   -.3181235   -.1884655
    kidsge6 |  .0143351   .0135204    1.06   0.289   -.0121644    .0408345
```

```
--------------------------------------------------------------------------------

. qui ivprobit inlf educ exper c.exper#c.exper age kidslt6 kidsge6 (nwifeinc = huseduc)
. eststo me2: margins, dydx(*) predict(pr fix(nwifeinc)) post

Average marginal effects                        Number of obs    =        753
Model VCE    : OIM

Expression   : Average structural function probabilities, predict(pr fix(nwifeinc))
dy/dx w.r.t. : nwifeinc educ exper age kidslt6 kidsge6
--------------------------------------------------------------------------------
             |            Delta-method
             |    dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+------------------------------------------------------------------
    nwifeinc | -.0105638   .0047364   -2.23   0.026   -.0198469   -.0012807
        educ |  .0487769   .0087333    5.59   0.000     .03166     .0658937
       exper |  .0219965   .0037232    5.91   0.000    .0146992    .0292939
         age | -.0128817   .0033216   -3.88   0.000    -.019392   -.0063714
     kidslt6 | -.2419815   .0365941   -6.61   0.000   -.3137047   -.1702583
     kidsge6 |  .0136948   .0127924    1.07   0.284   -.0113777    .0387674
--------------------------------------------------------------------------------
Warning: The chosen prediction can result in estimates of derivatives or
         contrasts that do not have a structural function interpretation.

. esttab me1 me2, b(3) se(3)
------------------------------------------
                    (1)            (2)

------------------------------------------
nwifeinc         -0.011*        -0.011*
                (0.006)        (0.005)
educ             0.051***       0.049***
                (0.011)        (0.009)
exper            0.023***       0.022***
                (0.003)        (0.004)
age             -0.013***      -0.013***
                (0.003)        (0.003)
kidslt6         -0.253***      -0.242***
                (0.033)        (0.037)
kidsge6          0.014          0.014
                (0.014)        (0.013)
------------------------------------------
N                   753            753
------------------------------------------
Standard errors in parentheses
* p<0.05, ** p<0.01, *** p<0.001
```

9   Acknowledgments

References

Allison, Paul D. 1999. "Comparing Logit and Probit Coefficients Across Groups." *Sociological Methods & Research* 28 (2): 186–208. https://doi.org/10.1177/0049124199028002003.

Alvarez, R Michael, and John Brehm. 1995. "American Ambivalence Towards Abortion Policy: Development of a Heteroskedastic Probit Model of Competing Values." *American Journal of Political Science*, 1055–82. https://doi.org/10.2307/2111669.

An, Weihua, and Xuefu Wang. 2016. "**LARF**: Instrumental Variable Estimation of Causal Effects Through Local Average Response Functions." *Journal of Statistical Software* 71 (1): 1–13. https://doi.org/10.18637/jss.v071.c01.

Arel-Bundock, Vincent. 2023. **marginaleffects**: *Predictions, Comparisons, Slopes, Marginal Means, and Hypothesis Tests*. https://vincentarelbundock.github.io/marginaleffects/.

Canty, Angelo J. 2002. "Resampling Methods in R: The **boot** Package." *The Newsletter of the R Project Volume* 2 (3). https://journal.r-project.org/articles/RN-2002-017/.

Carlson, Alyssa. 2019. "Parametric Identification of Multiplicative Exponential Heteroscedasticity." *Oxford Bulletin of Economics and Statistics* 81 (3): 686–96. https://doi.org/10.1111/obes.12280.

Carroll, Nathan. 2018. **oglmx**: *Estimation of Ordered Generalized Linear Models*. https://CRAN.R-project.org/package=oglmx.

Elff, Martin. 2012. **memisc**: *Tools for Management of Survey Data, Graphics, Programming, Statistics, and Simulation*. http://CRAN. R-project. org/package= memisc.

Fox, John, Michael Friendly, and Sanford Weisberg. 2013. "Hypothesis Tests for Multivariate Linear Models Using the **car** Package." *The R Journal* 5 (1): 39. https://doi.org/10.32614/RJ-2013-004.

Gilbert, Paul, and Ravi Varadhan. 2019. **numDeriv**: *Accurate Numerical Derivatives*. https://CRAN.R-project.org/package=numDeriv.

Greene, William H. 2002. *LIMDEP Econometric Modelling Guide: Version 8.0*. New York: Econometric Software. https://www.limdep.com/.

———. 2003. *Econometric Analysis*. 7th ed. Pearson Education India.

Greene, William H, and David A Hensher. 2010. *Modeling Ordered Choices: A Primer*. Cambridge University Press.

Harvey, Andrew C. 1976. "Estimating Regression Models with Multiplicative Heteroscedasticity." *Econometrica: Journal of the Econometric Society*, 461–65. https://doi.org/10.2307/1913974.

Henningsen, Arne, and Ott Toomet. 2011. "**maxLik**: A Package for Maximum Likelihood Estimation in R." *Computational Statistics* 26 (3): 443–58. https://doi.org/10.1007/s00180-010-0217-1.

Keele, Luke, and David K Park. 2006. "Difficult Choices: An Evaluation of Heterogeneous Choice Models." In *Paper for the 2004 Meeting of the American Political Science Association*, 2–5.

Knapp, Laura Greene, and Terry G Seaks. 1992. "An Analysis of the Probability of Default on Federally Guranteed Student Loans." *The Review of Economics and Statistics*, 404–11. https://doi.org/10.2307/2109484.

Leeper, Thomas J. 2021. **margins**: *Marginal Effects for Model Objects*. https://CRAN.R-project.org/package=margins.

Newey, Whitney K. 1987. "Efficient Estimation of Limited Dependent Variable Models with Endogenous Explanatory Variables." *Journal of Econometrics* 36 (3): 231–50. https://doi.org/10.1016/0304-4076(87)90001-7.

Rivers, Douglas, and Quang H Vuong. 1988. "Limited Information Estimators and Exogeneity Tests for Simultaneous Probit Models." *Journal of Econometrics* 39 (3): 347–66. https://doi.org/10.1016/0304-4076(88)90063-2.

Sarrias, Mauricio. 2016. "Discrete Choice Models with Random Parameters in R: The **Rchoice** Package." *Journal of Statistical Software* 74 (1): 1–31. https://doi.org/10.18637/jss.v074.i10.

StataCorp. 2019. *Stata Statistical Software, Release 16*. College Station, TX: StataCrop LLC. https://www.stata.com/.

Williams, Richard. 2009. "Using Heterogeneous Choice Models to Compare Logit and Probit Coefficients Across Groups." *Sociological Methods & Research* 37 (4): 531–59. https://doi.org/10.1177/0049124109335735.

———. 2010. "Fitting Heterogeneous Choice Models with oglm." *The Stata Journal* 10 (4): 540–67. https://doi.org/10.1177/1536867X1101000402.

Wooldridge, Jeffrey M. 2010. *Econometric Analysis of Cross Section and Panel Data*. MIT press.

———. 2014. "Quasi-Maximum Likelihood Estimation and Testing for Nonlinear Models with Endogenous Explanatory Variables." *Journal of Econometrics* 182 (1): 226–34. https://doi.org/10.1016/j.jeconom.2014.04.020.

———. 2015. "Control Function Methods in Applied Econometrics." *Journal of Human Resources* 50 (2): 420–45. https://doi.org/10.3368/jhr.50.2.420.

Yatchew, Adonis, and Zvi Griliches. 1985. "Specification Error in Probit Models." *The Review of Economics and Statistics*, 134–39. https://doi.org/10.2307/1928444.

Zaghdoudi, Taha. 2018. "**ivprobit**: An R Package to Estimate the Instrumental Variables Probit." *Journal of Open Source Software* 3 (22): 523. https://doi.org/10.21105/joss.00523.

Zeileis, Achim, Roger Koenker, and Philipp Doebler. 2015. **glmx**: *Generalized Linear Models Extended*. https://CRAN.R-project.org/package=glmx.

Mauricio Sarrias
Universidad de Talca
Facultad de Economía y Negocios (FEN)
Universidad de Talca, Talca, Chile
https://www.msarrias.com
ORCiD: 0000-0001-5932-4817

mauricio.sarrias@utalca.cl

genpathmox: An R Package to Tackle Numerous Categorical Variables and Heterogeneity in Partial Least Squares Structural Equation Modeling

by Giuseppe Lamberti,

Abstract Partial least squares structural equation modeling (PLS-SEM), combined with the analysis of the effects of categorical variables after estimating the model, is a well-established statistical approach to the study of complex relationships between variables. However, the statistical methods and software packages available are limited when we are interested in assessing the effects of several categorical variables and shaping different groups following different models. Following the framework established by Lamberti, Aluja, and Sanchez (2016), we have developed the genpathmox *R* package for handling a large number of categorical variables when faced with heterogeneity in PLS-SEM. The package has functions for various aspects of the analysis of heterogeneity in PLS-SEM models, including estimation, visualization, and hypothesis testing. In this paper, we describe the implementation of genpathmox in detail and demonstrate its usefulness by analyzing employee satisfaction data.

1 Introduction

Partial least squares structural equation modeling (PLS-SEM; Wold, 1985) is a method for estimating causal relationships between observed variables and hypothesized latent variables (Evermann and Rönkkö, 2021). PLS-SEM was developed initially as an alternative to the classical covariance based-structural equation modeling (CB-SEM) that estimates latent variables (LVs) as common factors that explain co-variation between the associated indicators (Hair Jr et al., 2017a). However, in the last fifteen years it has become a reference for estimating causal models, in particular in marketing and management research (Becker et al., 2022; Sarstedt et al., 2022a,b; Hair Jr et al., 2021; Henseler, 2020; Evermann and Rönkkö, 2021).

Concurrent with the affirmation of the PLS-SEM approach for estimating causal models, there has been a corresponding surge in research addressing the issue of heterogeneity in parameter estimation.This arises when the existence of different models is assumed for data characterized by differences in the coefficients that explain causal relationships between LVs. In this scenario, a single model could provide a biased view of those causal relationships. The literature describes several approaches to tackling heterogeneity that can be classified in terms of observed heterogeneity, and non-observed heterogeneity (for a detailed review of the PLS-SEM analysis in the presence of heterogeneity, see Klesel et al., 2022)

Observed heterogeneity is based on the hypothesis that the underlying relationship between latent variables vary by certain categorical variables (CVs), for example sociodemographic factors such as gender, education, or social status, such that the data can be separated into groups and a different model can be fit to each group. The presence of observed heterogeneity is then verified by testing whether the coefficients of the estimated models are significantly different between groups (Hair Jr et al., 2017b). Belonging to this category are the classical PLS multigroup tests, including the parametric (Keil et al., 2000), permutation (Chin and Dibbern, 2010), and Henseler (Henseler et al., 2009) tests, as well as the more recent approach proposed by Klesel et al. (2019). As for non-observed heterogeneity, this is present when differences are inherent to the data. In this scenario, a different approach is required, and different models are typically identified following a latent class analysis (Sarstedt et al., 2022c) or clustering (Esposito Vinzi et al., 2008) approach.

Methodological advances and the increase in applications have led to the development of numerous *R* packages, starting with **plspm** (Sanchez et al., 2015), subsequently followed by **cSEM** (Rademaker and Schuberth, 2020) and **SEMinR** (Ray et al., 2020), both of which incorporate recent developments in the PLS approach, including improved estimation (consistent PLS (PLSc) Dijkstra and Henseler, 2015), improved validation criteria (the PLSpredict approach to prediction Shmueli et al., 2016, 2019), new reliability measures (Dijkstra and Henseler, 2015; Hair Jr et al., 2019), and also **matrixpls** (Rönkkö, 2017), particularly used for simulation studies. Concerning heterogeneity, the **cSEM** package allows multigroup analysis with several embedded tests. Pathmox analysis (Sanchez and Aluja, 2006; Lamberti et al., 2016, 2017) was proposed as a useful method when observed heterogeneity is assumed, but several potential CVs exist that could define different groups and models. This

method explores and identifies, using an iterative algorithm, the most significantly different groups associated with significant differences in models. The algorithm follows a segmentation tree approach, where each node is a PLS-SEM model. Differences are compared and partitions are chosen that define the most significant divergences between coefficients (Lamberti et al., 2016). The algorithm has been further improved, first by including a statistical test capable of identifying, for each split, the model coefficient responsible for the partitions (Lamberti et al., 2017), and then by combining the pathmox algorithm with classical multigroup analysis in a new approach called hybrid multigroup analysis (Lamberti, 2021).

In this paper, we describe the **genpathmox** package (Lamberti, 2022) which implements the classical pathmox analysis and the more recently developed hybrid multigroup analysis (Lamberti, 2021). The package, initially developed in 2014, has been updated to include recent methodological advances (including PLSc; Dijkstra and Henseler, 2015) and has also been modified to work jointly with the **cSEM** package to increase analytical flexibility. Below we first review the framework formalized by Lamberti et al. (2016), then provide an overview of the **genpathmox** package, and finally, we demonstrate the use of the package on real-world analysis of employee satisfaction data and work climate drivers.

2   Overview of the pathmox methodology

Pathmox analysis (Lamberti et al., 2016, 2017) was introduced to handle observed heterogeneity in PLS-SEM when several CVs are present. Unlike classical methods for tackling observed heterogeneity in PLS-SEM, instead of testing whether a CV produces a significant difference in model coefficients (i.e., a confirmatory approach), an exploratory approach is adopted. That is, the aim of pathmox is to identify siginifcantly different groups associated with different PLS models, provided they exist. The algorithm applies a binary tree partitioning approach. It first estimates a single global model for the entire dataset to define the root node of the tree, and then explores all possible binary partitions for each CV. Differences in coefficients are statistically evaluated by the F-global test (Lamberti et al., 2016). This test provides a global measure of the degree of difference between partitions (i.e., a p-value). Comparisons are then sorted in descending order based on the p-values, and the partition with the smallest p-value (i.e. one which suggests the greatest difference from the root model) is then chosen as optimal.

The degree of difference between models (the split criterion) is determined by applying a test, inspired by Chow (1960) and Lebart et al. (1979) which compares differences between the coefficients of two linear regression models. In pathmox, the difference between two PLS-SEM models is determined by comparing structural model coefficients, i.e., by comparing, as in the case of Chow (1960) and Lebart et al. (1979), restricted deviance vs. unrestricted deviance, defined, respectively, as the deviance calculated for the whole sample considering a single model valid for all the observations, and the sum of the model deviances estimated for each group of observations.

From a graphical standpoint, pathmox is not much different from a classical segmentation tree – as the algorithm produces a tree with a root, intermediate nodes, and terminal nodes – other than that each node is associated with a PLS model.

2.1 The split criterion

The split criterion used to define the tree partitions is a critical aspect of the pathmox algorithm. Below we describe the F-global test, the formulation of the null and alternative hypotheses, and the statistic used to test the null hypothesis (further details are available in Lamberti et al., 2016).

Consider a simple structural model with one dependent LV, denoted by the Greek letter η, and explained by a generic set of independent LVs denoted by the matrix $\mathbf{X} = \{\xi_{ip}\}$, where $i = 1, \ldots, n$ refers to the observation, and where $p = 1, \ldots, P$ refers to the LV. Its generalization into a more complex model is straightforward.

Using the matrix form, the model can be expressed as:

$$\eta = \mathbf{X}\boldsymbol{\beta} + \varepsilon \tag{1}$$

where $\boldsymbol{\beta}$ is the vector of the regression coefficients of η, and where ε is the disturbance term. Let the data be partitioned by rows, where the partition is determined by a CV with m categories (i.e., segments or groups). The number of units in group g ($g = 1, \ldots, m$) is denoted by n_g, and the total sample size can be expressed as $n = \sum_{g=1}^{m} n_g$. The F-global test compares model coefficients only by considering binary partitions. This means that the number of comparisons depends on the nature of the CVs. With a dummy (binary) CV, there is just a single comparison. With a nominal CV, there are

$2^{m-1} - 1$ comparisons. Finally, with an ordinal CV, there are $m - 1$ comparisons.

The logic of the test is to compare the coefficients of two models, while considering two different scenarios. Under the null hypothesis, we assume that one model is valid for all observations. This implies that one coefficient for each independent LV is enough to explain the dependent LV. If we consider the simplest case of a dummy CV ($m=2$), denoting the two groups as A and B, the null and alternative hypotheses can be formulated as:

$$H_0 : \boldsymbol{\beta}_A = \boldsymbol{\beta}_B \tag{2}$$
$$H_1 : \boldsymbol{\beta}_A \neq \boldsymbol{\beta}_B \tag{3}$$

According to the null and alternative hypotheses, and following Lebart et al. (1979), we can rearrange Eq. 1 as follows:

$$\begin{bmatrix} \boldsymbol{\eta}_A \\ \boldsymbol{\eta}_B \end{bmatrix} \quad \begin{bmatrix} \mathbf{X}_A \\ \mathbf{X}_B \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\beta} \end{bmatrix} \quad + \begin{bmatrix} \boldsymbol{\varepsilon}_A \\ \boldsymbol{\varepsilon}_B \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} \boldsymbol{\eta}_A \\ \boldsymbol{\eta}_B \end{bmatrix} \quad \begin{bmatrix} \mathbf{X}_A & 0 \\ 0 & \mathbf{X}_B \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\beta}_A \\ \boldsymbol{\beta}_B \end{bmatrix} \quad + \begin{bmatrix} \boldsymbol{\varepsilon}_A \\ \boldsymbol{\varepsilon}_B \end{bmatrix} \tag{5}$$

We calculate the deviance (the sum of squared residuals, SSR) for both models: SSR_{H0} under the null hypothesis and SSR_{H1} under the alternative. Finally, we test the null hypothesis using the following statistic:

$$F = \frac{\left(SSR_{H_0} - SSR_{H_1} \right) \Big/ p}{SSR_{H_1} \Big/ n - 2p} \tag{6}$$

which follows an F distribution with p and $(n - 2p)$ degrees of freedom, where p is the number of explanatory LVs, and where $n = n_A + n_B$ is the total number of observations.

2.2 Improving tree partition interpretation: the F-coefficient test

The F-coefficient test was an important improvement in the algorithm introduced in Lamberti et al. (2017). The F-global test used in pathmox as a split criterion is a global criterion that establishes whether or not the CV reflects a significant difference. However, it does not provide information as to which coefficients are responsible for that difference. The F-coefficient test complements the split criterion in pathmox by providing information about which coefficients may be responsible for the significant difference.

Rearranging the model formulated by Eq. 1, we consider the particular case of one dependent LV denoted η, and two predictor LVs denoted ξ_1 and ξ_2:

$$\boldsymbol{\eta} = \boldsymbol{\xi}_1 \beta_1 + \boldsymbol{\xi}_2 \beta_2 + \boldsymbol{\varepsilon} \tag{7}$$

Let us assume that a significant difference exists between the models estimated for the two groups, A and B, as defined by a generic dummy variable. Applying the F-global test (Lamberti et al., 2016) we cannot determine whether the difference between the two models depends on ξ_1 or ξ_2, or depends on both. However the null hypotheses for β_1 and β_2, and the corresponding alternative hypotheses, can be reformulated to determine whether the coefficients estimated for the predictors are significantly different, as follows:[1]

$$H_0 : \beta_{iA} = \beta_{iB} \quad \text{with i } = 1, 2 \tag{8}$$
$$H_1 : \boldsymbol{\beta}_A \neq \boldsymbol{\beta}_B \tag{9}$$

According to the null and alternative hypotheses, and following Lebart et al. (1979), we can rearrange Eqs. 4 and 5 as:

---

[1]Note that the alternative hypothesis is the same for both ξ_1 and ξ_2.

$$\begin{bmatrix} \eta_A \\ \eta_B \end{bmatrix} \begin{bmatrix} \xi_{1A} & 0 & 0 \\ 0 & \xi_{2A} & 0 \\ \xi_{1B} & 0 & 0 \\ 0 & 0 & \xi_{2B} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_{2A} \\ \beta_{2A} \end{bmatrix} + \begin{bmatrix} \varepsilon_A \\ \varepsilon_B \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} \eta_A \\ \eta_B \end{bmatrix} \begin{bmatrix} \xi_{1A} & 0 & 0 \\ 0 & \xi_{2A} & 0 \\ 0 & 0 & \xi_{1B} \\ 0 & \xi_{2B} & 0 \end{bmatrix} \begin{bmatrix} \beta_{1A} \\ \beta_{1B} \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \varepsilon_A \\ \varepsilon_B \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} \eta_A \\ \eta_B \end{bmatrix} \begin{bmatrix} \xi_{1A} & 0 & 0 & 0 \\ 0 & \xi_{2A} & 0 & 0 \\ 0 & 0 & \xi_{1B} & 0 \\ 0 & 0 & 0 & \xi_{2B} \end{bmatrix} \begin{bmatrix} \beta_{1A} \\ \beta_{1B} \\ \beta_{2A} \\ \beta_{2B} \end{bmatrix} + \begin{bmatrix} \varepsilon_A \\ \varepsilon_B \end{bmatrix} \tag{12}$$

We calculate again the deviance for both models (SSR_{H0} and SSR_{H1}) and test the null hypothesis using the following statistic:

$$F_i = \frac{\left(SSR_{H0_{\beta_i}} - SSR_{H_1} \right) \Big/ 1}{SSR_{H_1} \Big/ 2\left(n - \sum p\right)} \quad \text{with i=1,2} \tag{13}$$

which follows an F distribution with 1 and $2\left(n - \sum p\right)$ degrees of freedom, and where p is the number of explanatory LVs, and $n = n_A + n_B$ is the total number of observations.

Note that both the F-global and the F-coefficient are implemented in the **genpathmox** package.

2.3 Stop criteria

Since pathmox is an iterative algorithm, its convergence depends on the specific stop criteria adopted by the user. Three criteria (all implemented in the **genpathmox** package) are proposed in Lamberti et al. (2016):

1. A more significant partition is not found. This means that the null hypothesis is not rejected in any of the candidate partitions, and as the obtained models are similar to each other, it makes no sense to continue splitting the data. This condition is also strictly related to the significance threshold of the p-value chosen by the user, usually set to 0.05 (a typical p-value threshold in PLS-SEM applications).

2. Maximum tree depth is achieved. This is related to the number of terminal nodes required by the user, a choice based on the complexity of the model and the number of CVs used. Generally speaking, trees of 2-3 levels (with a maximum of 4-8 associated terminal nodes) are preferred.[2]

3. A node has too few observations to be partitioned. PLS-SEM works well with a relatively small number of observations, but it is recommended to fix a threshold of a relatively large number of observations to ensure that nodes are representative. For exploratory purposes, the recommended number of observations in each node is between 50 and 100.

2.4 Pathmox to reduce the number of comparisons before running multigroup analysis: hybrid multigroup analysis

A criticism of the multigroup approach is that differences between coefficients could be difficult to interpret when the number of comparisons is high. This could happen when we have to simultaneously analyze more than one CV, or when the CV has more than 3 or 4 levels.

The pathmox algorithm does not perform an a posteriori statistical comparison of the coefficients of the models associated with the terminal nodes, nor does it establish the invariance between groups that is an important aspect of comparing PLS-SEM models (Henseler et al., 2016).[3] However, pathmox

---

[2] A greater tree depth results in a higher number of terminal nodes, with the direct consequence of having to make more comparisons between model coefficients, and with results that may not always be easy to interpret.

[3] Invariance ensures that a dissimilar group-specific model estimate does not depend on diverse LV meaning across groups. A specific procedure to verify measurement invariance in the PLS-PM framework – proposed by Henseler et al. (2016) – is measurement invariance of composite models (MICOM), consisting of three hierarchical steps: (1) configural invariance, which ensures the same LV specifications when LVs are equally parameterized

can be used just to reduce the number of groups to compare before running a classical multigroup analysis. Instead of using the original CV, the multigroup comparison uses a new intersection CV defined by the CV groups resulting from the tree partitions. This is called the hybrid segmentation variable (Lamberti, 2021), which is used for the hybrid multigroup analysis.

The hybrid multigroup analysis consists of sequential steps as follows:

1. Use pathmox to identify the most significantly different groups
2. Use multigroup analysis to compare the groups:

    (a) Test the invariance of the constructs among groups using the MICOM procedure (Henseler et al., 2016)

    (b) Test the statistical differences between models using a criterion proposed by the literature (Klesel et al., 2022).

Note that the **genpathmox** package does not include any function to automatically run the hybrid multigroup analysis. Rather, this analysis is done, as will be shown below, by combining the **genpathmox** and **cSEM** (using the functions `testMICOM()` to test invariance, and `testMGD()` to compare coefficients).

2.5 Pathmox advantages and limitations

A first advantage of pathmox is that, given a set of CVs, it yields the most significantly different groups associated with the most significantly different models. The algorithm reduces the number of groups to be compared and analyzed, with the direct consequence that the user merely has to interpret the differences. A second advantage is that it ranks CVs by their importance in the split process (as in other classical tree partitioning procedures). This is important because an analysis of differences in PLS-SEM with more CVs involves not only comparing groups, but also establishing the most significant sources of heterogeneity in defining differences.

The main limitation of pathmox is related to the split criteria. The fact that the algorithm realizes an exhaustive search over unadjusted p-values to determine the best partition could potentially produce biased results (Loh and Shih, 1997). A possible solution would be to apply a Bonferroni correction for multiple comparisons, but this is not yet available in the current version of the package. The F-global and the F-coefficients are parametric tests based on a classical parametric statistic: the F-statistic. This supposes the normality assumption of the perturbation terms with equal variance in all dependent constructs, even though the assumptions are rarely met in practice. Nevertheless, the sensitivity of the F-statistic is guaranteed by a larger sample size, lower levels of random perturbations, and clearer differences in the segments, as shown by the simulations performed by Lamberti et al. (2016, 2017). Another important limitation is that pathmox focuses only on the problem of detecting the path coefficients that are responsible for differences between PLS-SEM models, by adapting the measurement model to each segment. This leads to the problem of invariance, which greatly increases in importance when we analyze data with potential sources of heterogeneity by fitting one model to each segment. In this situation, it could become difficult to guarantee that each construct in each segment is measuring the same latent construct.

Finally, it is important to remark that the F-tests are determined by the sum of the squares of the residuals of the structural model in parent and children nodes and using the composite scores. Indeed, in the case of the common factor, the composites scores can just be used as common factor proxies since they are contaminated by measurement random error. Hence, the F-test ranking of the CVs may not be optimum when there is a small number of indicators per latent variable. Researchers who intend to apply pathmox when common factors are present in the model should take this limitation into account in performing the analysis; alternatively they should use the classical PLS algorithm modifying the options of the **genpathmox** functions accordingly.

---

and estimated across groups, (2) compositional invariance, which ensures that LV scores reflect the same construct across groups, and (3) equality of latent variables, which means that values and variances ensure that data can be pooled across groups. If all three steps are confirmed, full measurement invariance is established, while if only the first two steps are confirmed partial measurement invariance is established. Step one and two are necessary condition for performing multigroup analysis. A practical guideline on applying MICOM is provided by (Hair Jr et al., 2017b), while Henseler et al. (2016) provide more details on methodological aspects.

3 The *genpathmox* package

3.1 Overview

The **genpathmox** package is based on one main function called `pls.pathmox()`, which implements the pathmox algorithm and provides results for analysis. Four additional functions are a `summary()` function, and three plot functions (`plot()`, `bar_impvar()`, `bar_terminal()`) that help the user to interpret results. In practice, users should first apply the main function `pls.pathmox()` to generate a `"plstree"` object. The components of this object include tree partition results, fitted coefficients of the PLS models for each terminal node, and other results to be used for the analysis. The `"plstree"` object plays an instrumental role, as it is a necessary input for the other functions in the package. This design is convenient, as details of data, PLS model, and tree split rules need only be specified once in `pls.pathmox()`, and are passed to other functions.

The `summary()` function provides a complete output of all results, `plot()` provides the segmentation tree plot, `bar_impvar()` provides a bar plot of the ranking by importance of the CVs that participate in the split process, and `bar_terminal()` produces a bar plot of the coefficients of the PLS terminal nodes of the tree, enabling intuitive analysis of the differences between them.

The **genpathmox** package has been designed to interact with the **cSEM** package (Rademaker and Schuberth, 2020), one of the latest and most complete packages for PLS-SEM analysis. This package can be used to analyze each model associated with the terminal nodes identified by pathmox and to run the hybrid multigroup analysis. To that end, the `"plstree"` object also contains a list of datasets called `.hybrid`, corresponding to lists of datasets of observations belonging in the tree terminal nodes.

Using the `.hybrid` list combined with the **cSEM::csem()**, each terminal node can be easily and completely analyzed in terms of model validation, coefficient estimation, and inference. The resulting object generated by `csem()` can then be passed to `testMICOM()` to verify the invariance of the model constructs for the terminal nodes, and to `testMGD()` to compare the coefficients. The hybrid multigroup approach (Lamberti, 2021) can then be implemented. Further details on how to use the **cSEM** package are available in Rademaker and Schuberth (2020).

Figure 1 illustrates how to use the **genpathmox** package. On the left, the grey block contains the input elements, i.e., the data, the model, and the tree rules. Calling up `pls.pathmox()` generates the `"plstree"` object, as shown in the central orange block, which yields estimation and visualization results, as shown in the two orange blocks on the right. Finally, `plstree$hybrid` is used as the input parameter of the **cSEM** package, yielding full results for the terminal nodes and the multigroup analysis, as shown in the blue blocks.

Figure 1: Illustration of **genpathmox** package functions

3.2 Implementation of main functions

3.3 Estimation function: `pls.pathmox`

To apply the `pls.pathmox` function, users need to specify at least three arguments:

1. `.model`. A formula specifying the model described using syntax inspired by the lavaan package (Rosseel, 2012). Structural and measurement models are defined by enclosure between double quotes. The directional link between constructs is defined using the ("∼") operator. The dependent LV is on the left-hand side of the operator, and the explanatory LVs, separated by the ("+") operator, are on the right-hand side. As for the outer model, LVs are defined by listing the corresponding indicators after the operator ("=∼") if the LV is modelled as a common factor, or the operator("<∼") if the LV is modelled as a composite. On the left-hand side of the operator is the LV, and on the right-hand side are the indicators separated by the ("+") operator. Please note that variable labels cannot contain (".") (for details of the meaning of modes A and B, see Hair Jr et al., 2016).

2. `.data`. A matrix or data frame containing the indicators.

3. `.catvar`. A single factor or set of factors organized as a data frame containing the CVs used as sources of heterogeneity.

Other input parameters have default values. Table 1 reports the meaning of each parameter and the admissible and default values.

Parameter	Purpose	Possible values	Default values
`.scheme`	inner weighting scheme	`"centroid"`, `"factorial"`, or `"path"`	`"path"`
`.consistent`	consistent PLS estimation (Dijkstra and Henseler, 2015) is used instead of classical approach (Wold, 1985)	TRUE or FALSE	TRUE
`.alpha`	minimum threshold significance	values belonging the interval $[0, 1]$	0.05
`.deep`	maximum tree depth	an integer ≥ 1	2
`.size`	minimum proportion of total sample admissible for a node size	value belonging the interval $[0, 1]$	0.10
`.candidate size`	minimum admissible size for a candidate node	an integer ≥ 0	50
`.tree`	logic parameter to show the tree plot	TRUE or FALSE	TRUE

Table 1: Input parameters with default values

Once the split process is complete, results are saved in the object of class `"plstree"`, which contains all the results necessary to interpret the pathmox analysis (see Table 2)

Results	Use
MOX	provides information on the tree structure: node type (intermediate or terminal), node size, binary split
terminal_paths	allows visualization of path coefficients and R^2 for each terminal node
var_imp	provides a ranking of the CVs used in the split process
Fg.r	identifies which CV is responsible for the partition
Fc.r	identifies the path coefficient responsible for the partition
hybrid	subsets of data associated with each terminal node

Table 2: `"plstree"` results

3.4 Visualization functions: `plot`, `bar_impvar` **and** `bar_terminal`

Three types of plots are possible in the **genpathmox** package: a pathmox treeplot, a barplot which displays the ranking of the CVs, and a barplot of the PLS-SEM coefficients of the terminal nodes. The tree plot is obtained by applying the `plot()` function, which returns a tree structure with root, intermediate, and terminal nodes. For each partition, the *F*-global test *p*-value is reported with the associated CV, and the number of observations associated with each node. The plot is implemented using functions from the **diagram** package (Soetaert, 2020). The plot of the CV ranking is obtained using the `bar_impvar()` function. This function uses the `barplot()` function to visualize the importance of the CVs. The importance of each CV is based on the *F*-statistic of the *F*-global test calculated for each CV in each tree node. Finally, the plot of the coefficients of the PLS-SEM model for each terminal node is obtained using the `bar_terminal()` function, also based on the `barplot()` function, which allows a more intuitive comparison of the coefficients of the terminal nodes.

The user can choose between two bar plot visualizations: (1) a plot of all the coefficients of the same model in the same plot, which is useful for comparing the terminal nodes models, and (2) a plot of the same coefficients for all terminal nodes in the same plot (lines correspond to the coefficients and bars report the coefficient effects), useful for a more direct comparison of a specific coefficient between models. In the former, the bar plot depicted for each model also plots the associated R^2. Visualization options are selected by modifying the `.bycoef` parameter. By default, this is set to `FALSE`, meaning that the function implements the first option. We also need to specify for which dependent LV we want to visualize the predictor effect by fixing the parameter `.LV = " "`, which we do by indicating the dependent LV between quotation marks.

4 Application: analysis of employee satisfaction in terms of work climate drivers

The use of the **genpathmox** package is illustrated using real-world data on employee satisfaction in an international Spanish bank. In the financial sector, the impact of work climate on the relationship between strategic human resource management and organizational performance is crucial, in particular among younger employees (Kollmann et al., 2020). Another issue of relevance is that different groups of employees may respond in different ways to specific human resource management practices (Lamberti et al., 2020). The data of a sample of younger employees (≤30 years) of the Spanish bank contain measures regarding satisfaction (SAT), loyalty (LOY), and five work climate constructs: empowerment (EMP), company reputation (REP), leadership (LEAD), pay (PAY), and work conditions (WC). Our model relates the five work climate constructs with SAT, and SAT with LOY. Each construct is represented by a specific set of indicators. Information is also available on gender (female 53.36%), job level (intermediate, 52.01%), and seniority (length of service < 5 years, 66.81%).

Full details of indicators and LVs are available in the **genpathmox** manual, and details of the theoretical framework are provided in Lamberti et al. (2020).

Our objectives were: (1) to identify defining characteristics of different groups of employees, and (2) to analyze differences in the models for those groups.

4.1 Estimation

We used the `pls.pathmx()` function to partition the tree according to the CVs. We specified in order the parameter of the function `pls.pathmx()`: the model (`.model`), (2) the data (`.data`), and (3) the CVs (`.catvar`). The other parameters were left at the default values. We defined a structural model relating the five work climate constructs (EMP, REP, LEAD, PAY, WC) with SAT, and SAT with LOY, and we then related each construct to its own set of indicators (measurement model).

Note that, in this example, LVs are estimated as common factors. Indeed, by fixing the parameter `.consisten = TRUE`, consistent PLS estimation (Dijkstra and Henseler, 2015) will only have an effect on the final estimation of the path coefficients of the models of terminal nodes as identified by pathmox. Composite scores will be used to calculate the *F*-statistic, and to identify potential sources of heterogeneity.

```
# load genpathmox package
library(genpathmox)

# load data
data(climate)
```

```
# define del model
climate_model  =  "
        # structural model
        SAT ~ EMP + REP + PAY + WC + LEAD
        LOY ~   SAT
        # measurement model
        EMP =~ Empo1 + Empo2 + Empo3 + Empo4 + Empo5
        REP  =~ Imag1 + Imag2 + Imag3
        PAY  =~ Pay1+ Pay2 + Pay3 + Pay4
        WC =~ Work1 + Work2 + Work3
        LEAD =~ Lead1 + Lead2 + Lead3 + Lead4 + Lead5
        SAT =~ Sat1 + Sat2 + Sat3 + Sat4 + Sat5 + Sat6
        LOY =~ Loy1 + Loy2 + Loy3
        "
# define the set of categorical variables
climate_catvar = climate[,1:3]

# run the pls.pathmox() function
climate.pathmox = pls.pathmox(
        .model = climate_model,
        .data = climate,
        .catvar = climate_catvar)

PLS-SEM PATHMOX ANALYSIS

---------------------------------------------
Info parameters algorithm
  parameters algorithm value
1    threshold signif.  0.05
2   node size limit(\%) 0.10
3     tree depth level  2.00


---------------------------------------------
Info segmentation variables
         nlevels ordered treatment
Level          3    TRUE    ordinal
Seniority      2    TRUE    binary
Gender         2    FALSE   binary
```

As shown above, the default output of the `pls.pathmx()` function is a table containing the stop criteria and the list of CVs used in the split partitions. Below we use the `summary()` function to interpret the results.

```
summary(climate.pathmox)

PLS-SEM PATHMOX ANALYSIS

---------------------------------------------
Info parameters algorithm:
  parameters algorithm value
1    threshold signif  0.05
2   node size limit(%) 0.10
3     tree depth level 2.00
---------------------------------------------
Info tree:
        parameters tree value
1            deep tree      2
2 number terminal nodes     3
---------------------------------------------
Info nodes:
  node parent depth  type terminal size       % variable   category
1    1      0     0  root       no 669 100.00    <NA>        <NA>
2    2      1     1  node       no 476  71.15   Level low/medium
3    3      1     1 least      yes 193  28.85   Level       high
```

```
4    4    2    2 least    yes 258 38.57  Gender      Female
5    5    2    2 least    yes 218 32.59  Gender        Male
-------------------------------------------
Info splits:

Variable:
  node variable     g1.mod g2.mod
1    1    Level low/medium   high
2    2   Gender      Female   Male
```

```
Info F-global test results (global differences):
     node F value Pr(>F)
[1,]    1  6.9711 <2e-16 ***
[2,]    2  3.0647 0.0021 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Info F-coefficient test results (coefficient  differences) :

Node 1 :
           F value Pr(>F)
EMP -> SAT  2.5902 0.1078
REP -> SAT  0.4056 0.5243
PAY -> SAT  3.6390 0.0567 .
WC -> SAT   0.7342 0.3917
LEAD -> SAT 4.1333 0.0422 *
SAT -> LOY  0.1044 0.7467
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Node 2 :
           F value Pr(>F)
EMP -> SAT   0.0229 0.8797
REP -> SAT   0.6333 0.4263
PAY -> SAT   0.1874 0.6652
WC -> SAT    0.9907 0.3198
LEAD -> SAT  2.5447 0.1110
SAT -> LOY  17.9754 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
-------------------------------------------
Info variable importance ranking:
   variable   ranking
2     Level 0.3949974
3 Seniority 0.3101112
1    Gender 0.2948914
-------------------------------------------
Info terminal nodes PLS-SEM models (path coeff. & R^2):
         node 3 node 4 node 5
EMP->SAT  0.1233 0.2051 0.2725
REP->SAT  0.3037 0.1548 0.1238
PAY->SAT  0.0798 0.2863 0.1290
WC->SAT   0.4222 0.1333 0.3768
LEAD->SAT 0.2283 0.3283 0.1545
SAT->LOY  0.6934 0.7582 0.8806
R^2 SAT   0.6831 0.6863 0.6597
R^2 LOY   0.4808 0.5749 0.7754
```

We can interpret the summary() results as follows:

1. Pathmox indicates that different groups of employees exist that define SAT and LOY differently.

2. The variables that stratify the different groups of employees are, in order: job level (F-statistic = 6.971, p-value <0.001), gender (F-statistic = 3.065, p-value = 0.002). That this, the analysis suggests that employees are first partitioned into low/intermediate level employees versus high level employees, and low/intermediate level employees are then partitioned according to gender.

3. The coefficients responsible of the first split are LEAD–>SAT (F-statistic = 4.133, p-value = 0.042), and for the second split, SAT–>LOY (F-statistic = 17.975, p-value < 0.001).

4. Pathmox ultimately identifies three groups associated to the terminal tree nodes: high level employees (node 3), female low level employees (node 4), and male low level employees (node 5).

5. The CV ranking reveals that the most important differentiating characteristic for SAT and LOY is job level, followed by seniority, and finally gender.

6. In terms of work climate drivers defining SAT, the model comparisons indicate that:

    (a) High level employees (node 3) are least motivated by EMP ($\beta = 0.123$) and most motivated by WC $\beta = 0.422$) and REP ($\beta = 0.303$).

    (b) Female low level employees (node 4) are most motivated by LEAD ($\beta = 0.328$), PAY ($\beta = 0.286$), and EMP ($\beta = 0.205$).

    (c) Male low level employees (node 5) are most motivated by WC ($\beta = 0.377$) and least motivated by REP ($\beta = 0.124$), and also are the employees with the highest R^2 for LOY (0.775).

4.2 Visualization

The `summary()` output can be complemented by plots. First, the `plot()` function, as applied to the object of class `"plstree"`, produces the tree plot. The `bar_impvar()` and `bar_terminal()` functions allow graphical visualization of the ranking of CVs and a comparison of the coefficients (default value `bycoef = FALSE`, and `LV = "SAT"` to show the predictors of SAT most relevant for the analysis of work climate drivers). The three plots are shown in Figure 2.

```
# treeplot
plot(climate.pathmox)
# ranking of CVs
bar_impvar(climate.pathmox)
# coefficients comparison
bar_terminal(climate.pathmox, .LV = "SAT")
```

4.3 Terminal node outputs

Specific terminal nodes can be analyzed using the **cSEM** package function `csem()`. By default the `csem()` function needs two parameters: the datasets that include all indicators (`.data`), and the PLS-SEM model relationships (`.model`). As we are interested in the results of the terminal nodes, we pass the `hybrid` list in the `"plstree"` object to the `.data` parameter, and use the same formula object defined for the `pls.pathmox()` function. Below we reproduce the code, but not the output, as not directly related with the **genpathmox** package.

```
# load cSEM package
library(cSEM)

# identify terminal nodes
terminal_nodes_data = climate.pathmox$hybrid

# terminal nodes results
terminal_nodes_results = csem(.data = terminal_nodes_data,
                                        .model = climate_model)
```

Figure 2: Plot types available in the genpathmox package: the treeplot (top left), the variable-importance ranking plot (top right), and the barplot of the terminal nodes coefficients (bottom)

4.4 Hybrid multigroup approach (Lamberti, 2021): invariance and multigroup analysis

For the invariance and the multigroup comparison of the terminal nodes identified by pathmox, we pass the object generated by the csem() function to the testMICOM() and testMGD() functions. Note that, for the multigroup comparison, we need to indicate which coefficients to compare by fixing the parameter .parameters_to_compare. We generate the work climate model, but this time only indicating the causal relationship between the LVs. Finally, we indicate which statistical test to use for comparison using the .approach_mgd parameter (for our example, the permutation test, .approach_mgd = "Chin"). Below we reproduce the code to show how **genpathmox** interfaces with **cSEM**, omitting the results as there are not directly produced by the **genpathmox** package.

```
# MICOM procedure
climateMICOM = testMICOM(terminal_nodes_results)

# define the relationship between LVs
climateMICOM = testMICOM(terminal_nodes_results)

climate_innermodel = "
        # Structural model
        SAT ~ EMP + REP + PAY + WC + LEAD
        LOY ~   SAT
        "
# multigroup analysis
climateMGA = testMGD(terminal_nodes_results,
                                    .parameters_to_compare = climate_innermodel,
                                    .approach_mgd = "Chin")
```

5 Summary

The **genpathmox** *R* package handles observed heterogeneity in PLS-SEM models when the number of CVs is high and we do not know what the most significant groupings could be. Development of **genpathmox** reflects the statistical framework described in Lamberti (2021), and Lamberti et al. (2017, 2016), and the package has several functions that enable estimation and visualization of tree partitions. By using **genpathmox**, users can quickly explore the effects of heterogeneity on their PLS-SEM models and identify groups that may contribute to significant differences.

References

J. M. Becker, J. H. Cheah, R. Gholamzade, C. M. Ringle, and M. Sarstedt. Pls-sem's most wanted guidance. *International Journal of Contemporary Hospitality Management*, 2022. URL https://doi.org/10.1108/IJCHM-04-2022-0474. [p294]

W. W. Chin and J. Dibbern. An introduction to a permutation based procedure for multi-group pls analysis: Results of tests of differences on simulated data and a cross cultural analysis of the sourcing of information system services between germany and the usa. In *Handbook of partial least squares*, pages 171–193. Springer, Berlin Heidelberg, 2010. URL https://doi.org/10.1007/978-3-540-32827-8_8. ISBN 978-3-540-32825-4. [p294]

G. C. Chow. Tests of equality between sets of coefficients in two linear regressions. *Econometrica: Journal of the Econometric Society*, pages 591–605, 1960. URL https://doi.org/10.2307/1910133. [p295]

T. K. Dijkstra and J. Henseler. Consistent partial least squares path modeling. *MIS quarterly*, 39(2): 297–316, 2015. URL https://www.jstor.org/stable/26628355. [p294, 295, 300, 301]

V. Esposito Vinzi, L. Trinchera, S. Squillacciotti, and M. Tenenhaus. Rebus-pls: A response-based procedure for detecting unit segments in pls path modelling. *Applied Stochastic Models in Business and Industry*, 24(5):439–458, 2008. URL https://doi.org/10.1002/asmb.728. [p294]

J. Evermann and M. Rönkkö. Recent developments in pls. *Communications of the Association for Information Systems*, 44:123–132, 2021. URL https://doi.org/10.17705/1CAIS.044XX. [p294]

J. F. Hair Jr, G. T. M. Hult, C. Ringle, and M. Sarstedt. *A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM)*. saGe publications, Los Angeles, 2016. URL https://doi.org/10.1007/978-3-030-80519-7. ISBN 978-1-5443-9640-8. [p300]

J. F. Hair Jr, L. M. Matthews, R. L. Matthews, and M. Sarstedt. Pls-sem or cb-sem: updated guidelines on which method to use. *International Journal of Multivariate Data Analysis*, 1(2):107–123, 2017a. URL https://doi.org/10.1504/IJMDA.2017.087624. [p294]

J. F. Hair Jr, M. Sarstedt, C. M. Ringle, and S. P. Gudergan. *Advanced issues in partial least squares structural equation modeling*. saGe publications, Los Angeles, 2017b. ISBN 9781483377391. [p294, 298]

J. F. Hair Jr, J. J. Risher, M. Sarstedt, , and C. M. Ringle. When to use and how to report the results of pls-sem. *European business review*, 31(1):2–24, 2019. URL https://doi.org/10.1108/EBR-11-2018-0203. [p294]

J. F. Hair Jr, G. T. M. Hult, C. M. Ringle, M. Sarstedt, N. P. Danks, and S. Ray. *Partial least squares structural equation modeling (PLS-SEM) using R: A workbook*. Springer Nature, Los Angeles, 2021. URL https://doi.org/10.1007/978-3-030-80519-7. ISBN 9783030805197. [p294]

J. Henseler. *Composite-based structural equation modeling: Analyzing latent and emergent variables*. Guilford Publications, New York, 2020. ISBN 9781462545605. [p294]

J. Henseler, C. M. Ringle, and R. R. Sinkovics. The use of partial least squares path modeling in international marketing. In *New challenges to international marketing*, pages 277–319. Emerald Group Publishing Limited, Bingley, 2009. URL https://doi.org/10.1108/S1474-7979(2009)0000020014. ISBN 978-1-84855-468-9. [p294]

J. Henseler, C. M. Ringle, and M. Sarstedt. Testing measurement invariance of composites using partial least squares. *International marketing review*, 3(3):405–431, 2016. URL https://doi.org/10.1108/IMR-09-2014-0304. [p297, 298]

M. Keil, B. C. Tan, K. K. Wei, T. Saarinen, V. Tuunainen, and A. Wassenaar. A cross-cultural study on escalation of commitment behavior in software projects. *MIS quarterly*, pages 299–325, 2000. URL https://doi.org/10.2307/3250940. [p294]

M. Klesel, F. Schuberth, J. Henseler, and B. Niehaves. A test for multigroup comparison using partial least squares path modeling. *Internet research*, 29(3):464–477, 2019. URL https://doi.org/10.1108/IntR-11-2017-0418. [p294]

M. Klesel, F. Schuberth, B. Niehaves, and J. Henseler. Multigroup analysis in information systems research using pls-pm: A systematic investigation of approaches. *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, 53(3):26–48, 2022. URL https://doi.org/10.1145/3551783.3551787. [p294, 298]

T. Kollmann, C. Stöckmann, J. M. Kensbock, and A. Peschl. What satisfies younger versus older employees, and why? an aging perspective on equity theory to explain interactive effects of employee age, monetary rewards, and task contributions on job satisfaction. *Human Resource Management*, 59(1):101–115, 2020. URL https://doi.org/10.1002/hrm.21981. [p301]

G. Lamberti. Hybrid multigroup partial least squares structural equation modelling: an application to bank employee satisfaction and loyalty. *Quality & Quantity*, pages 1–23, 2021. URL https://doi.org/10.1007/s11135-021-01096-9. [p295, 298, 299, 305, 306]

G. Lamberti. *genpathmox: Pathmox Approach Segmentation Tree Analysis*, 2022. URL https://CRAN.R-project.org/package=genpathmox. R package version 0.9. [p295]

G. Lamberti, T. B. Aluja, and G. Sanchez. The pathmox approach for pls path modeling segmentation. *Applied Stochastic Models in Business and Industry*, 32(4):453–468, 2016. URL https://doi.org/10.1002/asmb.2168. [p294, 295, 296, 297, 298, 306]

G. Lamberti, T. Banet Aluja, and G. Sanchez. The pathmox approach for pls path modeling: Discovering which constructs differentiate segments. *Applied Stochastic Models in Business and Industry*, 33(6): 674–689, 2017. URL https://doi.org/10.1002/asmb.2270. [p294, 295, 296, 298, 306]

G. Lamberti, T. Aluja Banet, and J. Rialp Criado. Work climate drivers and employee heterogeneity. *The International Journal of Human Resource Management*, 33(3):472–504, 2020. URL https://doi.org/10.1080/09585192.2020.1711798. [p301]

L. Lebart, A. Morineau, and J. P. Fenelon. *Traitement des donnees statistiques*. Dunod, Paris, 1979. ISBN 10. 2040107878. [p295, 296]

W. Y. Loh and Y. S. Shih. Split selection methods for classification trees. *Statistica sinica*, pages 815–840, 1997. URL https://www.jstor.org/stable/24306157. [p298]

M. E. Rademaker and F. Schuberth. *cSEM: Composite-Based Structural Equation Modeling*, 2020. URL https://CRAN.R-project.org/package=cSEM. Package version: 0.4.0. [p294, 299]

S. Ray, N. P. Danks, and A. C. Valdez. *seminr: Building and Estimating Structural Equation Models*, 2020. URL https://CRAN.R-project.org/package=seminr. Package version: 0.4.0. [p294]

M. Rönkkö. *matrixpls: Matrix-based Partial Least Squares Estimation*, 2017. URL https://github.com/mronkko/matrixpls. R package version 1.0.5. [p294]

Y. Rosseel. lavaan: An r package for structural equation modeling. *Journal of Statistical Software*, 48(2): 1–36, 2012. URL https://doi.org/10.18637/jss.v048.i02. [p300]

G. Sanchez and T. Aluja. Pathmox: a pls-pm segmentation algorithm. *Proceedings of KNEMO*, 69, 2006. [p294]

G. Sanchez, L. Trinchera, and G. Russolillo. *plspm: Tools for Partial Least Squares Path Modeling (PLS-PM)*, 2015. URL https://github.com/gastonstat/plspm. R package version 0.4.9. [p294]

M. Sarstedt, J. F. Hair, M. Pick, B. D. Liengaard, L. Radomir, and C. M. Ringle. Progress in partial least squares structural equation modeling use in marketing research in the last decade. *Psychology & Marketing*, 39(5):1035–1064, 2022a. URL https://doi.org/10.1002/mar.21640. [p294]

M. Sarstedt, J. F. Hair Jr, and C. M. Ringle. Pls-sem: indeed a silver bullet–retrospective observations and recent advances. *Journal of Marketing Theory and Practice*, pages 1–15, 2022b. URL https://doi.org/10.1080/10696679.2022.2056488. [p294]

M. Sarstedt, L. Radomir, O. I. Moisescu, and C. M. Ringle. Latent class analysis in pls-sem: A review and recommendations for future applications. *Journal of Business Research*, 138:398–407, 2022c. URL https://doi.org/10.1016/j.jbusres.2021.08.051. [p294]

G. Shmueli, S. Ray, J. M. V. Estrada, and S. B. Chatla. The elephant in the room: Predictive performance of pls models. *Journal of Business Research*, 69(10):4552–4564, 2016. URL https://doi.org/10.1016/j.jbusres.2016.03.049. [p294]

G. Shmueli, M. Sarstedt, J. F. Hair, J. H. Cheah, H. Ting, S. Vaithilingam, and C. M. Ringle. Predictive model assessment in pls-sem: guidelines for using plspredict. *European journal of marketing*, 53(11): 2322–2347, 2019. URL https://doi.org/10.1108/EJM-02-2019-0189. [p294]

K. Soetaert. *diagram: Functions for Visualising Simple Graphs (Networks), Plotting Flow Diagrams*, 2020. URL https://CRAN.R-project.org/package=diagram. R package version 1.6.5. [p301]

H. Wold. *Partial Least Squares*, volume 6, pages 581–591. John Wiley and Sons, Ltd, 1985. URL https://doi.org/10.1002/0471667196.ess1914.pub2. [p294, 300]

Giuseppe Lamberti,
Department of Business,
Universitat Autonoma de Barcelona UAB,
Spain.
http://orcid.org/0000-0002-8666-796X
giuseppe.lamberti@uab.cat

Taking the Scenic Route: Interactive and Performant Tour Animations

by Casper Hart and Earo Wang

Abstract The tour provides a useful vehicle for exploring high dimensional datasets. It works by combining a sequence of projections—the tour path—in to an animation—the display method. Current display implementations in R are limited in their interactivity and portability, and give poor performance and jerky animations even for small datasets. We take a detour into web technologies, such as Three.js and WebGL, to support smooth and performant tour visualisations. The R package **detourr** implements a set of display tools that allow for rich interactions (including orbit controls, scrubbing, and brushing) and smooth animations for large datasets. It provides a declarative R interface which is accessible to new users, and it supports linked views using **crosstalk** and **shiny**. The resulting animations are portable across a wide range of browsers and devices. We also extend the radial transformation of the Sage Tour (Laa, Cook, and Lee (2021)) to 3 or more dimensions with an implementation in 3D, and provide a simplified implementation of the Slice Tour (Laa, Cook, and Valencia (2020)).

1 Introduction

An important first step in any data analysis task is to plot the data so that we can get an intuitive understanding of its structure, for example identifying the presence of clusters or outliers. When the data consists of one or two variables this is quite straightforward, but as the dimensionality of the data increases it becomes more difficult to visualise.

Several methods exist for high dimensional data visualisation. Given a data matrix $\mathbf{X}$ we can simply plot each variable $X_1 \dots X_p$ against the others in a pairwise fashion with the result being a scatterplot matrix (e.g. (Becker and Cleveland 1987)). We can also view projections of our data by calculating $\mathbf{Y} = \mathbf{XA}$ where $\mathbf{A}$ is a $p \times d$ projection matrix with d usually being 1 or 2. We can choose $\mathbf{A}$ in several different ways, some examples being Principal Component Analysis (PCA) which chooses the directions which explain the maximum variance, and Linear Descriminant Analysis (LDA) which maximises the ratio of between-group and within-group sums of squares. The scatterplot matrix can also be thought of as a projection method where the projections are parallel to each pair of coordinate axes. These are all examples of *linear dimension reduction* techniques, but non-linear techniques are also available such as t-SNE (Van der Maaten and Hinton 2008) and UMAP (McInnes, Healy, and Melville 2018) that aim to preserve both local and global structure of the data.

Rather than generating a single static visual, the *tour* (Asimov 1985); (Buja et al. 2005) works by combining a smooth sequence of projections in to an animation, which can then be viewed using a variety of different *display methods* (Wickham et al. 2011). This allows the viewer to explore the data from a number of different perspectives while being able to visually connect what would otherwise be disjointed views. However, existing display implementations for tours in R are limited in their interactivity, performance, and portability, and generally result in jerky animations even for small datasets with only tens or hundreds of observations.

In this paper we introduce a new R package called detourr, which provides portable and performant display methods for tours. In the first section we give a background of tours and review a few existing software implementations, and in the section following we describe how the software is used. We will then highlight some of the implementation decisions related to performance, and later provide a case study using embeddings created from the MNIST (LeCun 1998) dataset. In the final section we will discuss how this work might be extended in future.

2 Background and related works

At its core, the tour is a sequence of projections of a data set that are combined together to form an animation. If we denote an $n \times p$ data matrix $\mathbf{X}$ and a $p \times d$ projection matrix $\mathbf{A}$, then we can denote our $n \times d$ projected data set $\mathbf{Y}$ as $\mathbf{Y} = \mathbf{XA}$.

Each projection matrix $\mathbf{A}$ is often referred to as a *plane*, *frame*, or *basis*. (Note that in this paper, the term *frame* is avoided in this context to avoid ambiguity with *animation frames*). These bases are constrained to be orthonormal, so each column of $\mathbf{A}$ is a unit vector, and is orthogonal to each other column. In order to produce a smooth animation, a set of target bases are selected and interpolated between. Geodesic interpolation is generally used as described in (Buja et al. 2005).

2.1 Types of tours

Different tour paths arise from using different methods for selecting the target bases. For example, the *grand tour* was introduced by (Asimov 1985) and chooses a set of target projections at random. This can be thought of as a random walk around projections of the data.

The projection pursuit guided tour chooses bases to find a more interesting projection than the current one, where the interestingness is defined by some index function. Index functions such as *central mass*, *holes*, and *lda* are described in (Cook, Swayne, and Buja 2007).

Other types of tours include the *little tour*, which ensures bases parallel to the axes are visited; the *frozen tour* will fix some of the values in the projection matrix **A** between bases; the local tour chooses bases that are within some angular distance of the starting basis.

In all of these, the tour path is calculated in two steps; the target basis is calculated and then interpolation is done so that the transition to the next basis is smooth. Some methods generate the tour path in a single step, for example the langevitour package in R (Harrison 2022) produces tours as more of a physics simulation where points have position, velocity, momentum, and damping, and the position of points in subsequent animation frames is allowed to evolve while taking in to account user interactions.

2.2 Display methods

The typical display methods for tours include histograms or density plots for 1D projections, and x-y scatter plots for 2D ones. 3D data can be viewed by a 3D scatterplot with a virtual perspective camera to enable displaying on a monitor. This can be enhanced with fog to make closer points more prominent, and interactive rotation controls to give a more immersive 3D experience. Projections with 3 or more dimensions can be displayed using parallel coordinates plots. Other displays exist such as Andrew's plot (Andrews 1972), where each point is represented by a Fourier curve plotted between $-\pi$ and π.

These display methods can be enhanced to display additional information, for example the *slice tour* described in (Laa, Cook, and Valencia 2020) highlight points whose orthogonal distance to the projection plane is smaller than some threshold, and fade out points that are further away. This is good for finding hollowness in data, with an example shown in the case study.

Furthermore, the data may be transformed after being projected. One consequence of the curse of dimensionality is that when projecting from high to low dimensions, the points tend to crowd towards the centre. (Laa, Cook, and Lee 2021) describes the *sage tour*, and provides a radial transformation that ensures the relative volume at a radius r in the data space is preserved in the projected space. The effect of this is that the crowding is reduced, and uniformly distributed data in the original space will continue to be uniform in the projected space.

2.3 Software implementations

The tourr package (Wickham et al. 2011) is the most prevalent and comprehensive software in R (R Core Team 2021) for visualising tours. It implements many of the tour paths described previously including grand, little, guided, frozen, etc. and display methods including scatter plots (with variations for the sage and slice tour), parallel coordinates plots, depth displays, Andrew's plot. The package also allows exporting tours as GIF images via the gifski package (Ooms 2021), or exporting to GGobi (Swayne et al. 2003) to allow for interaction and linked brushing, etc. However, the **tourr** package uses the R graphics device as the primary display, which is quite limited in performance and interactivity.

The spinifex package (Spyrison and Cook 2020) provides manual tours built on **tourr** and using R shiny (Chang et al. 2021), and allows the user to manipulate the contribution of each variable one at a time. The liminal R package (Lee 2021) provides an interactive gadget for displaying tour visuals. Linked selection and brushing is implemented on both visuals, and play / pause / restart controls are provided.

The **langevitour** R package instead uses the htmlwidgets package (Vaidyanathan et al. 2021) to display the tour. The main calculations are performed in JavaScript and the points are displayed as a scatter plot using HTML5 Canvas. The displays have good performance so large numbers of data points can be plotted with the animation remaining smooth, and includes interactive features such as drag-and-drop of additional plot elements, and modifying parameters of the tour and having the changes reflected in real time. Once the tour visualisation is generated it then no longer relies on the R runtime so it can be easily exported and embedded on a website for example. But this package is developed with a particular focus of visualising physics dynamics, rather than the more classical tour methods like in **tourr**.

(Kipp, Laa, and Cook 2019) uses D3.js (Bostock, Ogievetsky, and Heer 2011) combined with the R **shiny** (Chang et al. 2021) package to display dynamic tour visualisations. However, this setup had limited performance; the client-server nature of **shiny** led to inconsistent frame rates, and the number of points that could be drawn was limited to <2000 because of the limitations of SVG when drawing many individual elements.

3 Usage and interactivity

When designing the user API for **detourr**, a data-oriented approach is taken to make it approachable, and the visuals are built in JavaScript to enable rich user interactions. **detourr** also supports the full suite of tour path generating functions from the **tourr** package. The result is an experience that is feature-rich, immersive, and accessible to newcomers.

This chapter is structured as follows. The first section describes the user API in R and supported features, the second describes how the user can interact with the resulting visual. Throughout the chapter, we use the pdfSense dataset (Wang et al. (2018); Lee (2021)) to provide a running example. This data set consists of 2808 observations and 56 input variables from CT14HERA parton distribution function fits. The first 6 principal components are used to create the tour, accounting for ~55% of the variance in the data.

3.1 User interface

detourr has a data-oriented user interface heavily influenced by the Tidy Data (Wickham (2014)) workflow, Grammar of Graphics (Wilkinson (2012); Wickham (2010)), and **ggplot2** (Wickham (2016)). The visualisation is built in a sequence of steps which follow the logical flow of data in the tour building process, which makes the API intuitive and accessible.

Instantiating the tour

To begin, we instantiate a tour using the detour() function:

```
p <- detour(pdf_df, tour_aes(
    projection = starts_with("PC"),
    colour = Type,
    label = ID
))
```

The first argument to detour() is a data frame in tidy format containing the tour data and aesthetics. Enforcing the use of data frames encourages data-centric statistical thinking. The second argument defines the aesthetic mapping of data variables through the tour_aes() function, similar to **ggplot2**. The currently supported aesthetics are:

- projection: (required) the numeric columns to be projected
- colour: point colour
- label: label text to be shown when the mouse is hovered over a point.

These mappings support tidy evaluation and **tidyselect** syntax (Henry and Wickham (2022)) such as starts_with(), where(is.numeric()), column ranges using col_1:col_n, negation -col_n, and others, for easier column selection.

Generating the tour path

Once the tour is initialised with the data and aesthetics, the tour path is defined by piping the output from detour() to the tour_path() function. Note that |> is the pipe operator introduced in R 4.1:

```
p <- p |> tour_path(grand_tour(3))
p

#> # A tibble: 472 x 2
#>   is_new_basis projection_matrix
#>   <lgl>        <list>
#> 1 TRUE         <dbl [6 x 3]>
#> 2 FALSE        <dbl [6 x 3]>
#> 3 FALSE        <dbl [6 x 3]>
#> 4 FALSE        <dbl [6 x 3]>
#> 5 FALSE        <dbl [6 x 3]>
#> # i 467 more rows
```

The tour_path() function defines parameters for the tour such as:

- tour_path: the tour path generator, e.g. grand_tour(), guided_tour(), or any other path generator compatible with the **tourr** package
- start: the starting basis or projection matrix
- fps: frames per second with which to display the animation. Defaults to 30 but can be increased for a smoother animation or decreased for very large data.
- max_bases: the number of basis frames to generate. A higher number will give a longer tour animation.

The resulting detour object is stored in a standard data frame for easy consumption and inspection. It contains the full details of the tour path, where the ith row corresponds to the ith animation frame of the tour, with the following columns:

1. is_new_basis: whether the projection matrix corresponds to a new basis (TRUE) or is interpolated (FALSE)
2. projection_matrix: the projection matrix.

This form gives the user full visibility of the tour path, and allows the projection matrices to be traced and extracted for further analyses.

Creating the animation

To display the tour animation, we simply pipe the output of tour_path() to any of the functions prefixed with show_ provided by the **detourr** package. The available display functions are:

- show_scatter(): the core 2D or 3D scatter plot display
- show_slice(): a slice tour display based on Laa, Cook, and Valencia (2020)
- show_sage(): a sage tour implementation based on Laa, Cook, and Lee (2021)

```
p |> show_scatter(axes = FALSE)
```

The output of tour_path() becomes the input of show_*(), forming a fluent pipeline. For the three display methods described above, the common parameters are:

- palette: the colour palette to use for the tour.
- center: whether the data should be centred before displaying.
- axes: whether to show axis / what the axis titles should be
- edges: a two-column numeric matrix defining indices of points where line segments should be drawn between
- paused: whether the animation should be initialised in a paused state.
- scale_factor: used to scale the points in or out so that they appear on a sensible range, similar to a zoom function. Defaults to the reciprocal of maximum distance from a point to the origin, so that the points fit inside a unit ball.

There are also parameters specific to each display method, such as slice_relative_volume for show_slice(), and gamma and R for show_sage(). These details will be described further in the Display Methods section.

Figure 1: Initial frame of the scatterplot display generated by the 'show_scatter' display function. Controls are on the left, and an interactive timeline is on the bottom of the plot.

Putting all of this together, we have:

```
detour(pdf_df, tour_aes(
    projection = starts_with("PC"),
    colour = Type,
    label = ID
)) |>
    tour_path(grand_tour(3)) |>
    show_scatter()
```

This chaining process allows us to construct the tour visualisation incrementally in a way that is intuitive and easy to follow. The user is able to inspect the result at each step in the chain, and it aligns well with the *grammar of graphics* and *tidy data* workflows. This makes **detourr** accessible to newcomers who may not have worked with tours previously.

3.2 Interactivity

Presently, several well-developed R packages allow the use of web technologies in R; **htmlwidgets** allows binding R code with HTML and JavaScript to create standalone widgets; **shiny** provides features for combining various elements in to interactive web applications powered by R; crosstalk (Cheng and Sievert (2022)) enables linked selection and brushing between different HTML Widgets; and rmarkdown (Allaire et al. (2022)) allows creating HTML documents with HTML Widgets embedded within. The use of web technologies such as JavaScript enable the resulting visuals to be portable and accessible, and enable the implementation of rich interactive features. In this section we will describe these interactive features and how they can be configured.

Label aesthetics

In the above example, labels are defined within the call to `tour_aes()`, which contains all of the aesthetic mappings for the tour. The `label` aesthetic produces a tooltip which is shown whenever the mouse is hovered over the data point. By default, the text in the tooltip will have the format `column_name: value`, with each specified column on a new line. If users want more control over what appears in the tooltip, one can use the `I()` function so that the values in the aesthetic column appear as-is. For example in 2, the left plot specifies the label aesthetic as `label = c(InFit, Type, ID, pt, x, mu)` and the right is specifies the label as-is by using `label = I(ID)` in the call to `tour_aes()`. When using the `I()` function for the label aesthetic, only one column can be specified at a time.

Figure 2: (Left) Tooltip showing data from the 6 columns specified in the 'label' aesthetic. Note that both the column names and values are present in the tooltip. (Right) The 'ID' column is provided as-is to the label aesthetic via the 'I()' function.

Controls

Table 1 shows a breakdown of the controls found on the left side of the visual. Note that the icon for the currently selected control will be highlighted blue; otherwise it will be black. When the icons are hovered over in the show_scatter() widget, alternative text will be shown.

Timeline controls

The timeline at the bottom of the widget controls play and pause, and allows for scrubbing to a specific point in the tour. The timeline can also be used to jump to a specific basis by clicking on any of the white basis markers, and hovering the mouse over the basis markers will display the index of that basis.

Linked selection and filtering

detourr supports linked selection and filtering by integrating with **crosstalk**. When a **crosstalk** SharedData object is provided to detour() in place of a data frame, selections made using the *box selection* tool will be reflected in all linked visuals. Likewise, any selection or filtering applied to a linked visual will be reflected by **detourr**. Compatible widgets include **plotly** (Sievert (2020)), **leaflet** (Cheng, Karambelkar, and Xie (2022)), and **DT** (Xie, Cheng, and Tan (2022)). An example of this is shown in the case study.

4   Web technologies for performance

One of the goals of this work is to improve upon the animation performance of existing tour displays. **detourr** uses several different web technologies to maximise performance so that smooth animations can be played with large data sets consisting of upwards of 100k data points. This performance also enables the animations to work with less powerful devices, making **detourr** accessible to a wider range of users.

The primary technology that allows for high-performance data visualisation is JavaScript itself. JavaScript engines in browsers such as Chrome and Firefox are highly optimised, leveraging methods such as Just-In-Time (JIT) compilation for improved runtime speed. However JavaScript is single-threaded, dynamically typed, and garbage collected, so despite these optimisations we can still run in to performance bottlenecks in some situations.

Figure 4 shows a simplified overview of the data flow in **detourr** when creating and viewing a widget. On the left are the operations that are performed by R, which only occur once when the visual is first created and have a minimal performance impact. On the right are the main operations

Table 1: An overview of the interactive controls available in the **detourr** displays

Control	Icon	Description
Orbit		When the 'show_scatter()' widget is generated, orbit controls will be enabled by default. This allows click and drag to rotate the visual, and scrolling/pinching to zoom. Note that orbit controls for the 2D variant work best if dragging from left to right, not up and down. Also note that the icon for the currently selected control will be highlighted blue; otherwise it will be black.
Pan		The pan control also allows scrolling to zoom, and click and drag to pan.
Box Selection		The selection control allows for transitory box selection by brushing. Holding the 'shift' key will allow for persistent selection, and points outside of the selection will be indicated by increased transparency. There is currently a limitation where only visible points can be selected. If a point is completely obscured by other points, it will not be selected.
Brush		The brush button will apply the current colour to the selected points.
Colour Selector		The colour selector will look slightly different depending on the browser being used. When the colour selection is changed, the selected points will be updated immediately.

Figure 3: An illustration of the box selection and brush tool being used together.

R   JavaScript

```
┌──────────────────────┐        ┌──────────────────────┐
│   Instantiate tour    │        │    Render points      │
│                       │        │  (Three.js / WebGL)   │
└──────────────────────┘        └──────────────────────┘
          │                                ↑↓
┌──────────────────────┐              Animation
│  Generate tour path   │                loop
└──────────────────────┘                  ↑↓
          │              JSON     ┌──────────────────────┐
┌──────────────────────┐  ───→    │ Calculate next frame  │
│ Create and display    │         │        Y=XA           │
│       widget          │         │ (TensorFlow.js / WASM)│
└──────────────────────┘         └──────────────────────┘
```

Figure 4: An overview of the data flow when creating a detourr visualisation. The full tour path is generated in R and then passed to JavaScript when the widget is created. The operations that occur in the animation loop in JavaScript are the most important to optimise.

performed by JavaScript when the widget is displayed in a browser or IDE. Linear algebra and rendering operations need to run 30 times per second, so the technology decisions surrounding them have a big impact on performance. These technology decisions and are discussed in this chapter.

4.1 Linear algebra operations

The single-threaded nature of JavaScript makes matrix multiplication a performance bottleneck. At each animation frame, we must calculate the product $\mathbf{XA}$ where $\mathbf{X}$ is our data matrix and $\mathbf{A}$ is our projection matrix. The slice tour generated by the show_slice() display function requires an additional step of calculating the distance from each point to the projection plane which involves several more matrix operations.

To address this, **detourr** uses TensorFlow.js (Abadi et al. (2016)) as the main library for storing data and projection matrices and performing matrix operations. TensorFlow.js requires the user chose between one of three available backends:

CPU is a single-threaded JavaScript implementation which carries with it the limitations of JavaScript dynamic typing and garbage collection causing non-deterministic slow-downs at runtime.

WebAssembly (WASM) is a binary format that is used as a compilation target allowing code written in other languages like C, C++, and Rust to be run in the browser. This circumvents the dynamic typing and garbage collection limitations of JavaScript and allows near-native execution speed. The TensorFlow WASM backend uses the XNNPACK library from Google to accelerate matrix operations, which can run operations in parallel using threads and SIMD (Single Instruction Multiple Data).

WebGL: uses WebGL shaders to perform matrix operations on the GPU. According to the documentation, the performance benefit is primarily seen with large and complex deep learning models, so is unlikely to provide much benefit over the WebAssembly backend for our use case, and so is not investigated further in this section.

4.2 Performance comparison

To compare these backend options a simple performance profile was run in Microsoft Edge (Chromium) on a Macbook Pro 2019 (i7, 32Gb RAM). The implementations that were compared were:

- *Hand Coded*: a manual JavaScript implementation coded using for loops, operating on nested arrays representing data and projection matrices.
- *TensorFlow CPU*: the vanilla single-threaded CPU backend for TensorFlow.js.
- *TensorFlow WASM*: the TensorFlow.js WASM backend.

These backends were compared across 3 datasets of different sizes and complexity, using a 2D Grand Tour:

Figure 5: Performance comparison across different data sets and backends. TensorFlow provides better performance than a hand-coded implementation across the board. For smaller datasets like pdfsense, there is little difference between CPU and WASM backends for TensorFlow.js, but for larger dataset WASM performs much better.

- *pdfsense*: The same data set used throughout this chapter; 2808 observations across 56 variables, taking the first 6 principal components for the tour.
- *mnist_embeddings_8d*: 8-dimensional embeddings of the MNIST dataset, with a total of 10k observations.
- *mnist_embeddings_32d*: 32-dimensional embeddings of the MNIST dataset, again with 10k observations.

Figure 5 shows the performance of the three backends across the example datasets. TensorFlow provides better performance across the board when compared to the hand-coded implementation, but the difference between the CPU and WASM backends only becomes apparent with the larger MNIST embeddings datasets. Note that the metric `% Scripting Time` is the time spent across *all* JavaScript scripting for the visual, and not just the time spend on linear algebra operations. This is why we see such diminishing returns with the smaller *pdfsense* dataset.

Another important comparison is the performance of the `show_slice()` display function between these datasets. The slice tour uses additional matrix operations to calculate the distance from each point to the projection plane, so the benefit of WASM backend is even more apparent. This is shown in Figure 6

4.3 Rendering

When displaying data visuals using JavaScript in a browser, there are three main technologies that can be used:

SVG is commonly used for web-based visuals, including in software such as D3.js (Bostock, Ogievetsky, and Heer (2011)) with good support for interaction and animation. Kipp, Laa, and Cook (2019) uses D3.js with SVG for rendering tours, but describes performance issues when the number of points gets close to 2,000. This is because while SVG is suitable for drawing large and complex shapes, performance can degrade when rendering many individual shapes.

HTML5 Canvas (2D) uses a canvas element with a 2D rendering context and provides good performance, allowing many thousand data points to be used with smooth animation. This is the rendering method used by the **langevitour** package, and provides much better performance over SVG for this use case.

HTML5 Canvas (WebGL) uses the WebGL rendering context with GPU acceleration to achieve high performance, and is used by a range of browser-based 3D animations and games. This typically provides higher performance than using the 2D canvas rendering context.

detourr implements **HTML5 Canvas** with the **WebGL** rendering context using the Three.js (Cabello (2010)) library. This is the same library that powers the TensorFlow Embedding Projector (Smilkov et al. (2016)), and allows for flexible and performant 2D and 3D data visuals.

Figure 6: The additional matrix operations required by the slice tour display function make the performance benefit of the WASM backend much more apparent.

One downside of using HTML5 Canvas elements is that custom logic is needed to determine where the mouse pointer is relative to visual elements when interactions occur. This issue is resolved is by rendering the image twice; the first pass renders to the screen and the second renders to an invisible "picking" scene. The colours of the points in this picking scene correspond to the ID of the point that was rendered. When a mouse is hovered over a pixel or a set of pixels are selected, we simply check their colour in the picking scene to determine which point IDs relate to the event. Rendering the scene twice at each frame makes performance all the more important.

Despite this extra step, a naive performance benchmark of the rendering performance of **detourr** using the `mnist_embeddings_8d` data set at 30 FPS shows only 3% of the time is devoted to rendering and painting points, which for our use case is negligible.

5 Display methods

There are three display functions implemented in the **detourr** package: `show_scatter()`, `show_sage()`, and `show_slice()`. All three support 2D and 3D tour paths, and are based on the core `show_scatter()` function. In this section, we will delve in to some of the implementation details of these functions and how the original and *sage* display has been extended to three dimensions.

5.1 Scatter display

The scatter display forms the core of the three display methods, and contains all of the features and interactions described in previous sections. It is implemented in TypeScript using the Three.js library (Cabello (2010)) for rendering and TensorFlow.js (Abadi et al. (2016)) for linear algebra operations.

5.2 Slice display

The slice display is implemented in the `show_slice()` function, and is based on the *slice tour* described in Laa, Cook, and Valencia (2020). At each animation frame, the distance from each point to the projection plane is computed. Those points closer than some threshold h to the projection plane are highlighted, and those further away are greyed out. Slices offset from the origin are also supported.

Despite the slice tour itself being equivalent to that in **tourr**, the implementation has been modified for a simpler implementation. Laa, Cook, and Valencia (2020) calculates the distance as:

$$\tilde{v}_i^2 = ||\mathbf{x}_i'||^2 \tag{1}$$

where

$$\mathbf{x}_i' = \mathbf{x}_i - (\mathbf{x}_i \cdot \mathbf{a}_1)\mathbf{a}_1 - (\mathbf{x}_i \cdot \mathbf{a}_2)\mathbf{a}_2 \tag{2}$$

Figure 7: Selected frames of a 2D slice tour of a hollow unit sphere. The anchor for the slice is set to (1, 0, 0). Initially the slice is near the origin, but moves closer to the edge of the sphere as v1 rotates to be near orthogonal to the projection plane.

and similar for the 3-dimensional case but with an additional term. With some rearranging, we can instead express this with the equivalent:

$$\tilde{v}^2 = (\mathbf{X} - \mathbf{X}\mathbf{A}\mathbf{A}^T)^2 \mathbf{1}_p \tag{3}$$

This requires fewer terms than the original, and is in a form that is more elegant to implement using TensorFlow.js. The implementation is also the same for both the 2D and 3D variants which keeps the code simple.

Offsetting the slice

Laa, Cook, and Valencia (2020) provides a generalisation of equations (1) and (2) for a projection plane passing through an arbitrary anchor point **c** as follows:

1. Calculate $\mathbf{x_i}'$ as per equation (2)
2. Calculate the component $\mathbf{c}'$ of **c** orthogonal to the projection plane as:

$$\mathbf{c}' = \mathbf{c} - (\mathbf{c} \cdot \mathbf{a}_1)\mathbf{a}_1 - (\mathbf{c} \cdot \mathbf{a}_2)\mathbf{a}_2$$

3. Calculate $v_i^2 = ||\mathbf{x}_i' - \mathbf{c}'||^2 = x_i'^2 + c'^2 - 2\mathbf{x}_i' \cdot \mathbf{c}'$ where the cross-term is expressed as:

$$\mathbf{x}_i' \cdot \mathbf{c}' = \mathbf{x_i} \cdot \mathbf{c} - (\mathbf{c} \cdot \mathbf{a}_1)(\mathbf{x}_i \cdot \mathbf{a}_1) - (\mathbf{c} \cdot \mathbf{a}_2)(\mathbf{x}_i \cdot \mathbf{a}_2)$$

With this method there are many terms to calculate, and it was found it was difficult to implement and test. To circumvent this issue we instead take a different approach. Rather than offsetting the **projection plane** to pass through the point **c** and then calculating the distances for each point, we instead offset the **data points** by **c** in the opposite direction. This gives a distance calculation between points and projection plane that is equivalent to the original implementation but is much simpler to calculate. First we calculate the offset points $\mathbf{X}'$:

$$\mathbf{X}' = \begin{bmatrix} \mathbf{x}_1 - \mathbf{c} \\ \mathbf{x}_2 - \mathbf{c} \\ \vdots \\ \mathbf{x}_n - \mathbf{c} \end{bmatrix} \tag{4}$$

And then calculate the distances to the projection plane similar to equation (3):

$$\tilde{v}^2 = (\mathbf{X}' - \mathbf{X}'\mathbf{A}\mathbf{A}^T)^2 \mathbf{1}_p \tag{5}$$

Figure 7 shows a slice tour implemented using equations (4) and (5) with an anchor of $(1, 0, 0)$. Initially v_1 is almost parallel to the projection plane, and so the slice runs close to the origin and only the points near the outside of the hollow sphere are highlighted. As the tour progresses, v_1 becomes nearly orthogonal to the projection plane, and so the slice runs close to the edge of the sphere and only a small number of points near the centre of the visual are highlighted.

5.3 Sage display

As the dimension of data increases, the volume of space that contains the data increases exponentially. One effect of this is that points tend to sit close to the edge of the space, with few points near the center. Hastie et al. (2001) gives a good illustration of this, where if we have N uniformly distributed points in a unit ball of dimension p centred at the origin, the median distance from the origin to the closest point is given by the equation

$$d(p, N) = \left(1 - \frac{1}{2}^{\frac{1}{N}}\right)^{\frac{1}{p}} \tag{6}$$

Counter-intuitively, when we project data from a high-dimensional space to low-dimensions, we see the opposite effect where points tend to crowd towards the center of the projected space. Laa, Cook, and Lee (2021) describes a method for correcting this distortion so that points are less crowded towards the center. It does this by ensuring the relative volume at a given radius r in the original space is preserved in the projected space. The relative volume for a 2-dimensional projection is given by the equation

$$v_2(r; p, R) = \frac{V_{2D}(r; p, R)}{V(R, p)} = 1 - \left(1 - \left(\frac{r}{R}\right)^2\right)^{\frac{p}{2}} \tag{7}$$

where p is the dimension of our original data, R is the radius of the p-ball that contains our data, and r is the projected radius within $[0, R]$.

The formula for the corrected radius r'_y is then given as

$$r'_y = R\sqrt{1 - \left(1 - \left(\frac{r}{R}\right)^2\right)^{\frac{p}{2}}} \tag{8}$$

detourr uses a slight variation of equation (8) to calculate the corrected radius, which omits the multiplier of R. This is because we always plot the data on the range $r'_y = [0, 1]$, so the multiplier is not needed:

$$r'_y = \sqrt{1 - \left(1 - \left(\frac{r}{R}\right)^2\right)^{\frac{p}{2}}} \tag{9}$$

The full implementation is as follows:

1. Calculate the projected data $\mathbf{Y} = \mathbf{XA}$, where $\mathbf{X}$ has already been scaled.
2. Calculate the trimmed radius of the projected points $r_y^{\text{trim}} = min(r_y, R)$ and apply the radial transformation described in equation (9) to get the corrected radius r'_y.
3. Scale the Euclidean point vectors by a factor of $\frac{r'_y}{r_y}$

This differs from the original implementation described in Laa, Cook, and Lee (2021) in that we don't convert the Euclidean vectors to polar form, and instead apply the scaling directly to the Euclidean vectors. This removal of the conversion step was primarily to improve performance.

Extension to 3D

Laa, Cook, and Lee (2021) provides the equation for the relative projected volume at radius r on to a two dimensional disk for the 2-dimensional sage display. In this paper, we extend and implement the scatter, sage, and slice displays in 3D, and to do this we needed to calculate the relative projected volume for the case of a 3-dimensional projection.

In the appendix we show that the relative projected volume for a **sphere** at radius r is given by:

$$v_3(r; p, R) = \text{BetaInc}\left(\left(\frac{r}{R}\right)^2, \frac{3}{2}, \frac{p-1}{2}\right) \tag{10}$$

Where $\text{BetaInc}(x, \alpha, \beta)$ is the regularised incomplete beta function. This is important because it represents the radial CDF of points projected to 3 dimensions, and suggests that the radial PDF of the projected points is $\text{Beta}\left(\frac{3}{2}, \frac{p-1}{2}\right)$ assuming the original data is a uniformly distributed ball of radius R.

So for the three dimensional case, the full radial transformation for the sage tour is given by

Figure 8: (Top) Initial frames of a 3D scatter tour of a 3, 10, and 50 dimensional ball respectively from left to right. (Bottom) Selected frames of a 3D sage tour of similar 3, 10, and 50 dimensional balls. As the dimensionality increases, the standard scatter display crowds the points near the center, whereas the sage display shows a consistent radial distribution of points. All screenshots are at the same zoom level.

$$r_y' = \sqrt[3]{\text{BetaInc}\left(\left(\frac{r}{R}\right)^2, \frac{3}{2}, \frac{p-1}{2}\right)} \tag{11}$$

We also show that this generalises to any projection from p to d dimensions with $p > d$ with the equation:

$$v(r; p, R, d) = \text{BetaInc}\left(\left(\frac{r}{R}\right)^2, \frac{d}{2}, \frac{p-d}{2}+1\right) \tag{12}$$

This also suggests that equation (7) is a special case of equation (12).

The 3D sage tour is currently implemented in the `show_sage()` function, and like the scatter and slice displays the correct variant is chosen automatically based on the dimension of the provided tour path. However, this is not implemented for $d > 3$ as we don't yet have a display method that can handle higher-dimensional projections. This will be implemented as an extension of a Parallel Coordinates Plot (PCP) or Andrew's plot in future.

An example of the 3D sage tour is shown in Figure 8.

6   Case study — MNIST embeddings

A common task when analysing wide or sparse data sets is to generate embeddings; finding a lower dimensional representation of high dimensional data, placing similar objects close together and dissimilar objects far apart in the embedding space. This is especially useful when dealing with text or image data.

An example of this is the algorithm used for facial recognition in FaceNet (Schroff, Kalenichenko, and Philbin (2015)). A neural network is trained which maps a vector representation of images of faces to a lower dimensional space. The network minimises the distance between examples of the same class and maximises distances between examples from different classes in the output space. The result is that the euclidean distance between faces can be used as a metric for face similarity, so an unknown face can be classified as belonging to a specific individual if the distance between the unknown face and one or more known faces is small.

Figure 9: Selected frames from the 8-dimensional MNIST embeddings data using show_scatter() as the display method. The colour corresponds to the handwritten digit 0, 1, ..., 9. Despite the large number of data points, the animation of the tour is smooth and interactions are responsive.

The datasets `mnist_embeddings_8d` and `mnist_embeddings_32d` in the **detourr** package are embeddings trained using a similar algorithm to FaceNet but using the MNIST (LeCun (1998)) handwritten digits dataset. The training set consists of 60,000 28x28 pixel training images and in the following examples we visualise the test set containing 10,000 examples.

6.1 Scatterplot display

Using the core `show_scatter()` function to display a `grand_tour()` tour path in Figure 9 we can see quite good separation between the clusters corresponding to each of the 10 digits. Despite the tour animation consisting of 10,000 data points, the animation runs smoothly at 30 FPS in Microsoft Edge on a Macbook Pro 2019. Running a performance profile of the animation indicates the CPU is idle 90% of the time while the animation is playing. The remaining time is divided between scripting (6%, including linear algebra operations), rendering (1.4%), painting (0.8%) and system (1.8%). When running the same tour on the `mnist_embeddings_32d` dataset, the animation is still quite smooth and CPU is 80% idle.

In the lower left of Figure 9 is an example of the `label` aesthetic at work. This allows the user to identify which group a set of points belongs to, as well as the precise ID of any outliers that may require further investigation.

Figure 10: Selected frames from the 8-dimensional MNIST embeddings data using show_sage() as the display method with a 2D grand tour path. The sage display shows the data points near the surface of the unit ball, which is due to the L2 normalisation of the original embeddings. This structure was not clear in the standard scatter display but is preserved with the sage display.

6.2 Sage and Slice display methods

The show_scatter() display method gives the viewer a fairly good sense of the data set, but there is some structure that may not be obvious. The embeddings in the mnist_embeddings_8d and mnist_embeddings_32d datasets are L2 normalised, so the points sit on the surface of a unit ball in the high-dimensional space. To reveal this structure, we can use the *sage* (Laa, Cook, and Lee (2021)) or *slice* (Laa, Cook, and Valencia (2020)) display methods, which are implemented as show_sage() and show_slice() respectively.

The *sage* display scales points outwards based on their radius so that the relative volume of the circle or sphere in the projected space is the same as in the original space. In the example shown in Figure 10, the show_sage() display method is used. The effect is that the projected points tend sit much closer to the surface of the unit circle, giving a much clearer view of the ball-like structure of the original data.

The *slice* display highlights points based on their proximity to the projection plane. Points that are close to the projection plane are highlighted and those further away are faded out by making them transparent. In the case of the MNIST embedding data in Figure 11 the ball structure of the data manifests as a clear circular void in the middle of the plot, with points highlighted only towards the edges.

6.3 Linked selection

Plot interactions such as selection and filtering can be helpful for identifying and exploring outliers, clusters, and other interesting features in a dataset. These are enhanced even further when multiple visuals are linked, and selections and filters are applied to all linked visuals. In this example, we compare the tour animation with the result of a T-SNE (Van der Maaten and Hinton (2008)) which was performed using the excellent **Rtsne** R package (Krijthe (2015)) and displayed using **plotly**. The visuals are linked using the R package **crosstalk** and a set of filter checkboxes is also added.

Figure 12 shows the linked visuals in their initial state with no filtering applied. We can then use the selection tool in either of the visuals to highlight points, and see the highlighting applied to both visuals as in Figure 13.

Figure 14 shows the result of filtering the visuals using the filter checkboxes on the left. In the filtered visual, outlying points are much easier to see, and they can be easily investigated using tooltips.

Figure 11: Selected frames of the 8-dimensional MNIST embeddings data using show_slice() as the display method. The slice display makes the hollowness of this data apparent.

Figure 12: Linked visuals of the tour using detourr (left) compared to a T-SNE dimension reduction (right)

Figure 13: Linked visuals with selection applied. Points can be selected in either visual via click-and-drag and the selection will be reflected in both.

Figure 14: Linked visuals with filtering applied. Viewing each digit individually makes outlying points much more apparent, and those points can be identified using tooltips.

The code used to produce figures 12, 13, and 14 is shown below. Here each plot is created using a **crosstalk** SharedData object in place of a standard data frame, and linked together using the bscols function:

```r
library(crosstalk)
library(Rtsne)
library(plotly)

data(mnist_embeddings_8d)

ts <- select(mnist_embeddings_8d, starts_with("X")) |>
    Rtsne(num_threads = 4)
Y <- as_tibble(ts$Y)
names(Y) <- c("Y1", "Y2")

plot_df <- bind_cols(mnist_embeddings_8d, Y)
shared_mnist <- SharedData$new(plot_df)

detour_plot <- detour(shared_mnist, tour_aes(
    projection = starts_with("X"), color = label,
    label = c(id, label),
)) |>
    tour_path(grand_tour(2)) |>
    show_sage(width = "100%", height = "450px")

tsne_plot <- plot_ly(shared_mnist,
    x = ~Y1,
    y = ~Y2,
    text = paste0("Label: ", plot_df$label, "<br>", "ID: ", plot_df$id),
    color = ~label,
    height = 450,
    colors = viridisLite::viridis(10)
) |>
    highlight(on = "plotly_selected", off = "plotly_doubleclick") |>
    add_trace(type = "scatter", mode = "markers")

bscols(
    list(
        filter_checkbox("label", "Label", shared_mnist, ~label)
    ),
    detour_plot, tsne_plot,
    widths = c(1, 5, 6)
)
```
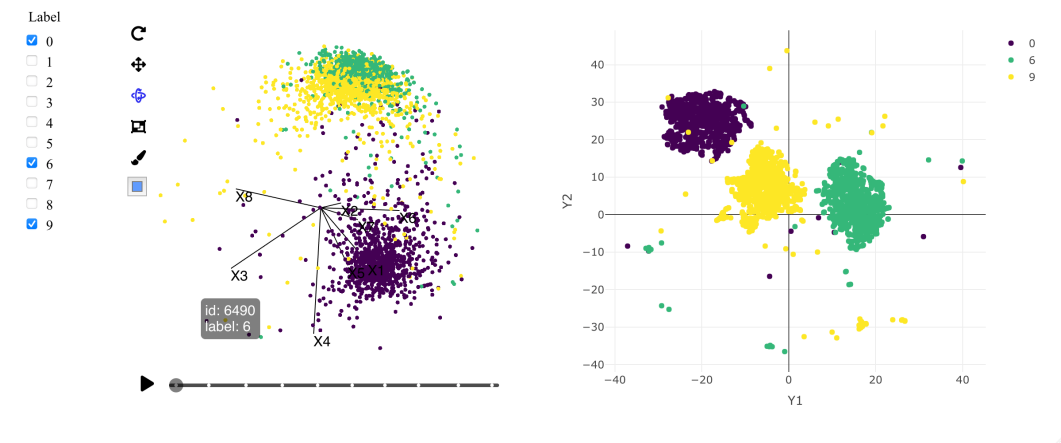
7 Conclusion and future work

In this paper we have introduced **detourr** which provides interactive, performant, and portable tour visualisations from R. We accomplish these things using web technologies; TensorFlow.js (Abadi et al. (2016)) provides fast linear algebra operations through WebAssembly, Three.js provides GPU rendering via WebGL, and JavaScript & HTML enable good performance and interactive features across the board. We also provide a simplified implementation of the Slice display (Laa, Cook, and Valencia (2020)), and have generalised the radial transformation from the Sage display (Laa, Cook, and Lee (2021)) to work with tours of 3 or more dimensions. All of this is done with an intuitive user interface which makes the software accessible to new users.

Looking ahead, the priority for the next stage of development is to leverage **detourr**'s extensible design to implement additional display methods such as density plots, histograms, parallel coordinates plots, and Andrew's plot. Additional changes could also be made to allow the radial transformation of the sage display and the highlighting of points from the slice display to be incorporated in to these other display methods, rather than being limited to only the scatter plot display. This would also allow the additional information from both the sage and slice tour applied to the same visual.

Further enhancements could be made by implementing facetting; allowing grouped data to be displayed across separate visuals with unified controls and timeline added. This could be taken further by allowing multiple *different* displays to use the same controls and timeline, for example displaying a scatter plot alongside one or more density plots.

To extend the existing scatter plot displays, the addition of an interactive legend would greatly enhance the user experience. As well as providing context for the point colour / fill, this would allow the user to be able to filter groups without needing to use a separate package like **shiny** or **crosstalk**. A shape aesthetic would also be beneficial, and the ability to export the projection matrix at the current frame would make it easier to perform analysis once an interesting projection is found.

As well as being able to display points and lines, support for plotting surfaces would allow for rich visualisations of regression model fits and classification boundaries. Three.js has good support for drawing surfaces, however it's not clear how a decision boundary can be projected down to a lower number of dimensions or whether this is actually feasible.

Support for displaying images or sprites directly on the tour visual or as an extension of the tooltip functionality is possible. A similar feature is implemented in the Tensorboard Embedding Visualiser (Smilkov et al. (2016)) which also uses Three.js under the hood.

What's more, Three.js has support for VR, which would be an interesting addition for exploring an immersive 3D tour visual.

8 Radial CDF of hyperspheres projected to 3 dimensions

In order to implement the 3D variant of the sage tour (Laa, Cook, and Lee (2021)), we need an expression for the relative projected volume of a **sphere** of radius R. This is then used as a scaling factor for point radii in the visualisation to prevent points from being crowded towards the centre.

First we denote the volume of a p-dimensional hypersphere by:

$$\frac{2\pi^{p/2}R^p}{p\Gamma(p/2)}$$

In the appendix of Laa et al. (2022) (equations 7–10) is a derivation for the relative projected volume of a ball of radius r.

$$F(r; p, R) = \frac{V_{inside}(r; p, R)}{V(p, r)} \tag{13}$$

$$= 1 - \frac{V_{outside}(r; p, R)}{V(p, r)} \tag{14}$$

And the formula for $V_{outside}(r; p, R)$ for a **circle** is given as:

$$V_{outside}(r; p, R) = \int_r^R V(p - 2, \sqrt{R^2 - x^2})2\pi x dx \tag{15}$$

To extend this to the 3-dimensional case, we can modify (15) to express the volume outside a **sphere** of radius r as:

$$V_{outside}(r; p, R) = \int_r^R V(p - 3, \sqrt{R^2 - x^2})4\pi x^2 dx \tag{16}$$

and it follows that the relative projected volume for a sphere is

$$F_3(r; p, R) = 1 - \frac{\int_r^R V(p - 3, \sqrt{R^2 - x^2})4\pi x^2 dx}{V(p, R)} \tag{17}$$

We know $2\Gamma(3/2) = \Gamma(1/2) = \sqrt{\pi}$ so with some rearranging this can be reduced to:

$$F_3(r; p, R) = 1 - \frac{2}{R^p}\frac{\Gamma(p/2 + 1)}{\Gamma(3/2)\Gamma((p-1)/2)}\int_r^R (R^2 - x^2)^{(p-3)/2}x^2 dx \tag{18}$$

Denoting $u = 1 - \left(\frac{x}{R}\right)^2$ and $dx = \frac{R^2}{-2x}du = \frac{R}{-2\sqrt{1-u}}du$ for a change of variable this becomes

$$F_3(r; p, R) = 1 - \frac{\Gamma(p/2 + 1)}{\Gamma(3/2)\Gamma((p-1)/2)}\int_0^{1-\frac{r^2}{R^2}} u^{(p-3)/2}(1 - u)^{1/2}du \tag{19}$$

$$= 1 - \text{BetaInc}\left(1 - \left(\frac{r}{R}\right)^2, \frac{p-1}{2}, \frac{3}{2}\right) \tag{20}$$

$$= \text{BetaInc}\left(\left(\frac{r}{R}\right)^2, \frac{3}{2}, \frac{p-1}{2}\right) \tag{21}$$

where *BetaInc* is the regularised incomplete beta function (the CDF of a Beta distribution).

We can generalise this to any projection from p to d dimensions using the same steps, but with

$$V_{outside}(r; p, R, d) = \int_r^R V(p - d, \sqrt{R^2 - x^2})\frac{2\pi^{d/2}}{\Gamma(d/2)}x^{d-1}dx \tag{22}$$

where $\frac{2\pi^{d/2}}{\Gamma(d/2)}x^{d-1}$ is the surface area of a d-ball.

This results in the relative projected volume of a projection from p to d dimensions being given by:

$$F(r; p, R, d) = \text{BetaInc}\left(\left(\frac{r}{R}\right)^2, \frac{d}{2}, \frac{p-d}{2} + 1\right) \tag{23}$$

Figures 15 and 16 compare the theoretical results from equations (21) and (23) respectively with
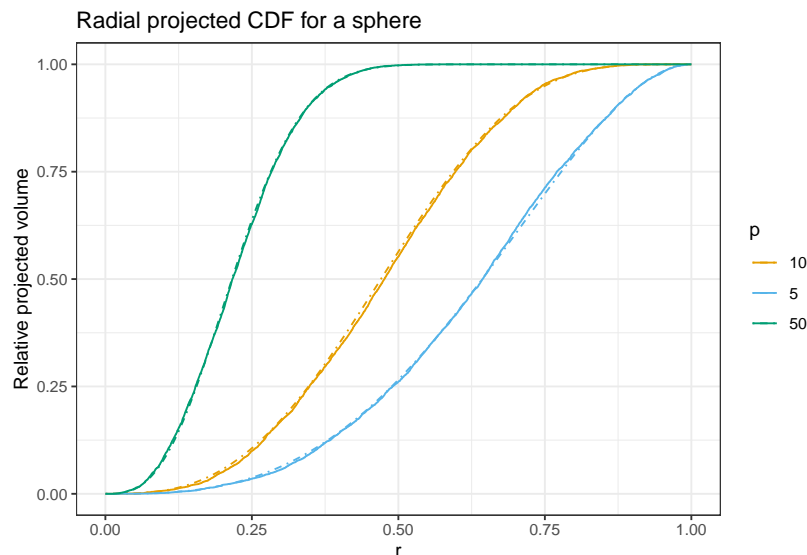
Figure 15: Relative projected volume for projections from p dimensions to d=3 dimensions. The solid line is simulated data, and the dashed line is the theoretical CDF

simulated values.

References

Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. "TensorFlow: A System for Large-Scale Machine Learning." In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 265–83.

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2022. *Rmarkdown: Dynamic Documents for r*. https://github.com/rstudio/rmarkdown.

Andrews, David F. 1972. "Plots of High-Dimensional Data." *Biometrics*, 125–36.

Asimov, Daniel. 1985. "The Grand Tour: A Tool for Viewing Multidimensional Data." *SIAM Journal on Scientific and Statistical Computing* 6 (1): 128–43.

Becker, Richard A, and William S Cleveland. 1987. "Brushing Scatterplots." *Technometrics* 29 (2): 127–42.

Bostock, Michael, Vadim Ogievetsky, and Jeffrey Heer. 2011. "D³ Data-Driven Documents." *IEEE Transactions on Visualization and Computer Graphics* 17 (12): 2301–9.

Buja, Andreas, Dianne Cook, Daniel Asimov, and Catherine Hurley. 2005. "Computational Methods for High-Dimensional Rotations in Data Visualization." *Handbook of Statistics* 24: 391–413.

Cabello, Ricardo. 2010. "Three.js." *GitHub Repository*. https://github.com/mrdoob/three.js.

Chang, Winston, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert, and Barbara Borges. 2021. *Shiny: Web Application Framework for r*. %7Bhttps://CRAN.R-project.org/package=shiny%7D.

Cheng, Joe, Bhaskar Karambelkar, and Yihui Xie. 2022. *Leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library*. %7Bhttps://CRAN.R-project.org/package=leaflet%7D.

Cheng, Joe, and Carson Sievert. 2022. *Crosstalk: Inter-Widget Interactivity for HTML Widgets*. %7Bhttps://rstudio.github.io/crosstalk/%7D.

Cook, Dianne, Deborah F Swayne, and Andreas Buja. 2007. *Interactive and Dynamic Graphics for Data Analysis: With r and GGobi*. Springer Science & Business Media.

Harrison, Paul. 2022. *Langevitour: Langevin Tour*. %7Bhttps://CRAN.R-project.org/package=langevitour%7D.

Hastie, Trevor, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Henry, Lionel, and Hadley Wickham. 2022. *Tidyselect: Select from a Set of Strings*. %7Bhttps://CRAN.R-project.org/package=tidyselect%7D.

Kipp, Michael, Ursula Laa, and Dianne Cook. 2019. "Connecting r with D3 for Dynamic Graphics, to Explore Multivariate Data with Tours." *The R Journal* 11 (1): 245.

Krijthe, Jesse H. 2015. *Rtsne: T-Distributed Stochastic Neighbor Embedding Using Barnes-Hut Implementation*. https://github.com/jkrijthe/Rtsne.

Laa, Ursula, Dianne Cook, Andreas Buja, and German Valencia. 2022. "Hole or Grain? A Section
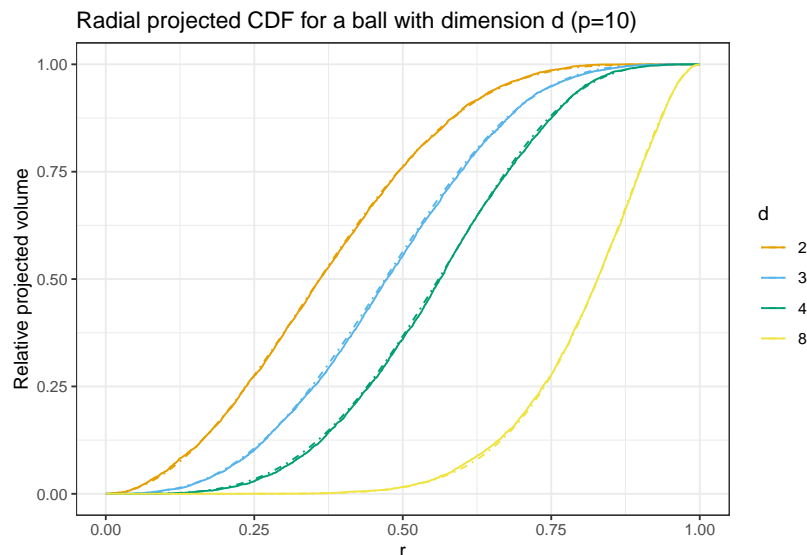
Figure 16: Relative projected volume for a projection of p=10 dimensions to d dimensions. The solid line is simulated data, and the dashed line is the theoretical CDF. This shows the generalisation to d > 3 dimensions

Pursuit Index for Finding Hidden Structure in Multiple Dimensions." *Journal of Computational and Graphical Statistics* 0 (0): 1–14. https://doi.org/10.1080/10618600.2022.2035230.

Laa, Ursula, Dianne Cook, and Stuart Lee. 2021. "Burning Sage: Reversing the Curse of Dimensionality in the Visualization of High-Dimensional Data." *Journal of Computational and Graphical Statistics*, 1–10.

Laa, Ursula, Dianne Cook, and German Valencia. 2020. "A Slice Tour for Finding Hollowness in High-Dimensional Data." *Journal of Computational and Graphical Statistics* 29 (3): 681–87.

LeCun, Yann. 1998. "The MNIST Database of Handwritten Digits." *Http://Yann. Lecun. Com/Exdb/Mnist/*.

Lee, Stuart. 2021. *Liminal: Multivariate Data Visualization with Tours and Embeddings*. %7Bhttps://CRAN.R-project.org/package=liminal%7D.

McInnes, Leland, John Healy, and James Melville. 2018. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction." arXiv. https://doi.org/10.48550/ARXIV.1802.03426.

Ooms, Jeroen. 2021. *Gifski: Highest Quality GIF Encoder*. %7Bhttps://CRAN.R-project.org/package=gifski%7D.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. %7Bhttps://www.R-project.org/%7D.

Schroff, Florian, Dmitry Kalenichenko, and James Philbin. 2015. "Facenet: A Unified Embedding for Face Recognition and Clustering." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 815–23.

Sievert, Carson. 2020. *Interactive Web-Based Data Visualization with r, Plotly, and Shiny*. Chapman; Hall/CRC. %7Bhttps://plotly-r.com%7D.

Smilkov, Daniel, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B. Viégas, and Martin Wattenberg. 2016. "Embedding Projector: Interactive Visualization and Interpretation of Embeddings." arXiv. https://doi.org/10.48550/ARXIV.1611.05469.

Spyrison, Nicholas, and Dianne Cook. 2020. "spinifex: An R Package for Creating a Manual Tour of Low- dimensional Projections of Multivariate Data." *The R Journal* 12 (1): 243–57. https://doi.org/10.32614/RJ-2020-027.

Swayne, Deborah F, Duncan Temple Lang, Andreas Buja, and Dianne Cook. 2003. "GGobi: Evolving from XGobi into an Extensible Framework for Interactive Data Visualization." *Computational Statistics & Data Analysis* 43 (4): 423–44.

Vaidyanathan, Ramnath, Yihui Xie, JJ Allaire, Joe Cheng, Carson Sievert, and Kenton Russell. 2021. *Htmlwidgets: HTML Widgets for r*. %7Bhttps://CRAN.R-project.org/package=htmlwidgets%7D.

Van der Maaten, Laurens, and Geoffrey Hinton. 2008. "Visualizing Data Using t-SNE." *Journal of Machine Learning Research* 9 (11).

Wang, Bo-Ting, TJ Hobbs, Sean Doyle, Jun Gao, Tie-Jiun Hou, Pavel M Nadolsky, and Fredrick I Olness. 2018. "Mapping the Sensitivity of Hadronic Experiments to Nucleon Structure." *Physical Review D* 98 (9): 094030.

Wickham, Hadley. 2010. "A Layered Grammar of Graphics." *Journal of Computational and Graphical Statistics* 19 (1): 3–28.

———. 2014. "Tidy Data." *The Journal of Statistical Software* 59. http://www.jstatsoft.org/v59/i10/.

———. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. springer.

Wickham, Hadley, Dianne Cook, Heike Hofmann, and Andreas Buja. 2011. "Tourr: An r Package for Exploring Multivariate Data with Projections." *Journal of Statistical Software* 40: 1–18.

Wilkinson, Leland. 2012. "The Grammar of Graphics." In *Handbook of Computational Statistics*, 375–414. Springer.

Xie, Yihui, Joe Cheng, and Xianying Tan. 2022. *DT: A Wrapper of the JavaScript Library 'DataTables'*. %7Bhttps://CRAN.R-project.org/package=DT%7D.

Casper Hart
University of Auckland
Department of Statistics
casperhart93@gmail.com

Earo Wang
The University of Auckland
Department of Statistics
earo.wang@gmail.com

Identifying Counterfactual Queries with the R Package cfid

by Santtu Tikka

Abstract In the framework of structural causal models, counterfactual queries describe events that concern multiple alternative states of the system under study. Counterfactual queries often take the form of "what if" type questions such as "would an applicant have been hired if they had over 10 years of experience, when in reality they only had 5 years of experience?" Such questions and counterfactual inference in general are crucial, for example when addressing the problem of fairness in decision-making. Because counterfactual events contain contradictory states of the world, it is impossible to conduct a randomized experiment to address them without making several restrictive assumptions. However, it is sometimes possible to identify such queries from observational and experimental data by representing the system under study as a causal model, and the available data as symbolic probability distributions. Shpitser and Pearl (2007) constructed two algorithms, called ID* and IDC*, for identifying counterfactual queries and conditional counterfactual queries, respectively. These two algorithms are analogous to the ID and IDC algorithms by Shpitser and Pearl (2006b, 2006a) for identification of interventional distributions, which were implemented in R by Tikka and Karvanen (2017) in the causaleffect package. We present the R package cfid that implements the ID* and IDC* algorithms. Identification of counterfactual queries and the features of cfid are demonstrated via examples.

1 Introduction

Pearl's ladder of causation (or causal hierarchy) consists of three levels: association, intervention, and counterfactual (Pearl, 2009). These levels describe a hierarchy of problems with increasing conceptual and formal difficulty. On the first and lowest level, inference on associations is based entirely on observed data in the form of questions such as "what is the probability that an event occurs?" or "what is the correlation between two variables". On the second level, the inference problems are related to manipulations of the system under study such as "what is the probability of an event if we change the value of one variable in the system". Questions on the intervention level cannot be answered using tools of the association level, because simply observing a change in a system is not the same as intervening on the system. Randomized controlled trials are the gold standard for studying the effects of interventions, because they enable the researcher to account for confounding factors between the treatment and the outcome and to carry out the intervention in practice. However, there are often practical limitations that make it difficult, expensive, or impossible to conduct a randomized experiment. The third and highest level is the counterfactual level. Typically, counterfactual statements compare the real world, where an action was taken or some event was observed, to an alternative hypothetical scenario, where a possibly different action was taken, or a different event was observed. Counterfactuals are often challenging to understand even conceptually due this notion of contradictory events in alternative worlds, and such alternatives need not be limited to only two. In general, questions on the counterfactual level cannot be answered by relying solely on the previous levels: no intervention or association is able to capture the notion of alternative hypothetical worlds.

While counterfactual statements can be challenging, they are a core part of our everyday thinking and discourse. Importantly, counterfactuals often consider retrospective questions about the state of the world, such as "would an applicant have been hired if they had more work experience". This kind of retrospection is crucial when fair treatment of individuals is considered in hiring, healthcare, receiving loans or insurance, etc., with regards to protected attributes, especially when the goal is automated decision-making. Statistical approaches to fairness are insufficient in most contexts, such as in scenarios analogous to the well-known Simpson's paradox, but routinely resolved using the framework of causal inference. In some cases, even interventional notions of fairness may be insufficient, necessitating counterfactual fairness (Kusner et al., 2017; Zhang and Bareinboim, 2018).

The structural causal model (SCM) framework of Pearl provides a formal approach to causal inference of interventional and counterfactual causal queries (Pearl, 2009). An SCM represents the system of interest in two ways, First, the causal relationships are depicted by a directed acyclic graph (DAG) whose vertices correspond to variables under study and whose edges depict the direct functional causal relationships between the variables. Typically, only some of these variables are observed and the remaining variables are considered latent, corresponding either to confounders between multiple variables or individual random errors of single variables. Second, the uncertainty

related to the variables in the system is captured by assuming a joint probability distribution over its latent variables. The functional relationships of the model induce a joint probability distribution over the observed variables. The SCM framework also incorporates the notion of external interventions symbolically via the do-operator, and a graphical representation of counterfactual scenarios via parallel worlds graphs (Avin et al., 2005; Shpitser and Pearl, 2007, 2008).

One of the fundamental problems of causal inference is the so-called identifiability problem, especially the identifiability of interventional distributions. Using the SCM framework and do-calculus, it is sometimes possible to uniquely represent an interventional distribution using only the observed joint probability distribution of the model before the intervention took place. Such interventional distributions are called *identifiable*. More generally, we say that a causal query is identifiable, if it can be uniquely represented using the available data. In most identifiability problems, the available data consists of causal quantities on levels below the query in the ladder of causation, but the levels also sometimes overlap, (e.g., Bareinboim and Pearl, 2012; Tikka and Karvanen, 2019; Lee et al., 2019). The identifiability problem of interventional distributions, and many other interventional identifiability problems have been solved by providing a sound and complete identification algorithm (e.g., Shpitser and Pearl, 2006; Huang and Valtorta, 2006; Lee et al., 2019; Kivva et al., 2022).

Software for causal inference is becoming increasingly prominent. For R, a comprehensive overview of the state-of-the-art is provided by the recently launched task view on *Causal Inference* on the Comprehensive R Archive Network (CRAN). Out of the packages listed in this task view, the **Counterfactual** (Chen et al., 2020) and **WhatIf** (Stoll et al., 2020) packages are directly linked to counterfactual inference, but the focus of these packages is estimation and they do not consider the identifiability of counterfactual queries. The **R6causal** (Karvanen, 2022) package can be used to simulate data from counterfactual scenarios in a causal model. R packages most closely related to causal identifiability problems are the **causaleffect** (Tikka and Karvanen, 2017), **dosearch** (Tikka et al., 2021), and **dagitty** (Textor et al., 2017).

We present the first implementation of the counterfactual identifiability algorithms of Shpitser and Pearl (2007) (see also Shpitser and Pearl, 2008) as the R package **cfid** (**c**ounter**f**actual **id**entification). The **cfid** package also provides a user-friendly interface for defining causal diagrams and the package is compatible with other major R packages for causal identifiability problems such as **causaleffect**, **dosearch** and **dagitty** by supporting graph formats used by these packages as inputs.

The paper is organized as follows. Section 2.2 introduces the notation, core concepts and definitions, and provides an example on manual identification of a counterfactual query without relying on the identifiability algorithms. Section 2.3 presents the algorithms implemented in **cfid** and demonstrates their functionality via examples by tracing their operation line by line. Section 2.4 demonstrates the usage of the **cfid** package in practice. Section 2.5 concludes the paper with a summary.

2 Notation and definitions

We follow the notation used by Shpitser and Pearl (2008) and we assume the reader to be familiar with standard graph theoretic concepts such as ancestral relations between vertices and d-separation. We use capital letters to denote random variables and lower-case letters to denote their value assignments. Bold letters are used to denote sets of random variables and counterfactual variables. We associate the vertices of graphs with their respective random variables and value assignments in the underlying causal models. In figures, observed variables of graphs are denoted by circles, variables fixed by interventions are denoted by squares, and latent unobserved variables are denoted by dashed circles when explicitly included and by bidirected edges when the corresponding latent variable has two observed children. Latent variables with only one child, which are called *error terms*, are not shown for clarity.

A *structural causal model* is a tuple $M = (\mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{u}))$ where $\mathbf{U}$ is a set of unobserved random variables, $\mathbf{V}$ is a set of n observed random variables, $\mathbf{F}$ is a set of n functions such that each function f_i is a mapping from $\mathbf{U} \cup \mathbf{V} \setminus \{V_i\}$ to V_i and such that it is possible to represent the set $\mathbf{V}$ as function of $\mathbf{U}$. $P(\mathbf{u})$ is a joint probability distribution over $\mathbf{U}$. The causal model also defines its causal diagram G. Each $V_i \in \mathbf{V}$ corresponds to a vertex in G, and there is a directed edge from each $V_j \in \mathbf{U} \cup \mathbf{V} \setminus \{V_i\}$ to V_i. We restrict our attention to *recursive* causal models in this paper, meaning models that induce an acyclic causal diagram.

A *counterfactual variable* $Y_\mathbf{x}$ denotes the variable Y in the submodel $M_\mathbf{x}$ obtained from M by forcing the random variables $\mathbf{X}$ to take the values $\mathbf{x}$ (often denoted by the do-operator as $do(\mathbf{X} = \mathbf{x})$ or simply $do(\mathbf{x})$). The distribution of $\mathbf{Y}_\mathbf{x}$ in the submodel $M_\mathbf{x}$ is called the *interventional distribution* of $\mathbf{Y}$ and it is denoted by $P_\mathbf{x}(\mathbf{y})$. However, if we wish to consider multiple counterfactual variables that originate from different interventions, we must extend our notation to counterfactual conjunctions. *Counterfactual conjunctions* are constructed from value assignments of counterfactual variables, and

individual assignments are separated by the $\wedge$ symbol. For example, $y_x \wedge z_x \wedge x'$ denotes the event that $Y_x = y$, $Z_x = z$ and $X = x'$. The probability $P(y_x \wedge z_x \wedge x')$ is the probability of the counterfactual event. Note that primes do not differentiate variables, instead they are used to differentiate between values i.e., x is a different value from x' and they are both different from x'' but all three are value assignments of the random variable X. If the subscript of each variable in the conjunction is the same, the counterfactual probability simply reduces to an interventional distribution.

Each counterfactual conjunction is associated with multiple *parallel worlds*, each induced by a unique combination of subscripts that appears in the conjunction. A *parallel worlds graph* of the conjunction is obtained by combining the graphs of the submodels induced by interventions such that the latent variables are shared. The simplest version of a parallel worlds graph is a twin network graph, contrasting two alternative worlds (Balke and Pearl, 1994a,b; Avin et al., 2005). As a more complicated example, consider the counterfactual conjunction $\gamma = y_x \wedge x' \wedge z_d \wedge d$. In simpler terms, this conjunction states that Y takes the value y under the intervention $\mathrm{do}(X = x)$, Z takes the value z under the intervention $\mathrm{do}(D = d)$, and X and D take the values x' and d, respectively, when no intervention took place. Importantly, this conjunction induces three distinct parallel worlds: the non-interventional (or observed) world, a world where X was intervened on, and a world where D was intervened on. For instance, if the graph in Figure 1(a) depicts the original causal model over the variables Y, X, Z, W and D, then Figure 1(b) shows the corresponding parallel worlds graph for γ, where each distinct world is represented by its own set of copies of the original variables. In Figure 1(b), U corresponds to the bidirected edge between X and Y in Figure 1(a), and the other U-variables are the individual error terms of each observed variable, that are not drawn when they have only one child in Figure 1(a).

Note that instead of random variables, some nodes in the parallel worlds graph now depict fixed values as assigned by the interventions in the conjunction. This is a crucial aspect when d-separation statements are considered between counterfactual variables in the parallel worlds graph, as a backdoor path through a fixed value is not open. Furthermore, not every variable is necessarily unique in a parallel worlds graph, making it possible to obtain misleading results if d-separation is used to infer conditional independence relations between counterfactual variables. For instance, if we consider the counterfactual variables Y_x, D_x and Z in a causal model whose diagram is the graph shown in Figure 1(a), then Y_x is independent of D_x given Z, even though Y_x is not d-separated from D_x in the corresponding parallel worlds graph of Figure 1(b). This conditional independence holds because Z and Z_x are in fact the same counterfactual variable. To overcome this problem, the parallel worlds graph must be further refined into the *counterfactual graph* where every variable is unique, which we will discuss in the following sections in more detail. For causal diagrams and counterfactual graphs, $V(G)$ denotes the set of observable random variables not fixed by interventions and $v(G)$ denotes the corresponding set of value assignments.

The following operations are defined for counterfactual conjunctions and sets of counterfactual variables: $\mathrm{sub}(\cdot)$ returns the set of subscripts, $\mathrm{var}(\cdot)$ returns the set of (non-counterfactual) variables, and $\mathrm{ev}(\cdot)$ returns the set of values (either fixed by intervention or observed). For example, consider again the conjunction $\gamma = y_x \wedge x' \wedge z_d \wedge d$. Now, $\mathrm{sub}(\gamma) = \{x, d\}$, $\mathrm{var}(\gamma) = \{Y, X, Z, D\}$ and $\mathrm{ev}(\gamma) = \{y, x, x', z, d\}$. Finally, $\mathrm{val}(\cdot)$ is the value assigned to a given counterfactual variable, e.g., $\mathrm{val}(y_x) = y$. The notation $y_{\mathbf{x}..}$ denotes a counterfactual variable derived from Y with the value assignment y in a submodel $M_{\mathbf{x} \cup \mathbf{z}}$ where $\mathbf{Z} \subseteq \mathbf{V} \setminus \mathbf{X}$ is arbitrary.

The symbol P_* is used to denote the set of all interventional distributions of a causal model M over a set of observed variables $\mathbf{V}$, i.e.,

$$P_* = \{P_{\mathbf{x}} \mid \mathbf{x} \text{ is any value assignment of } \mathbf{X} \subseteq \mathbf{V}\}$$

In the following sections, we consider identifiability of counterfactual queries in terms of P_*. In essence, this means that a counterfactual probability distribution $P(\gamma)$ is identifiable if it can be expressed using purely interventional and observational probabilities of the given causal model.

2.1 Example on identifiability of a counterfactual query

We consider the identifiability of the conditional counterfactual query $P(y_x | z_x \wedge x')$ from P_* in the graph depicted in Figure 2. This graph could for instance depict the effect of an applicant's education (X) on work experience (Z) and a potential hiring decision (Y) by a company. Our counterfactual query could then consider the statement "what is the probability to be hired if the applicant's education level was changed to x, given that their work experience under the same intervention was z and when in reality their education level was x'". In this example, we will not rely on any identifiability algorithms. Instead, we can derive a formula for the counterfactual query as follows:
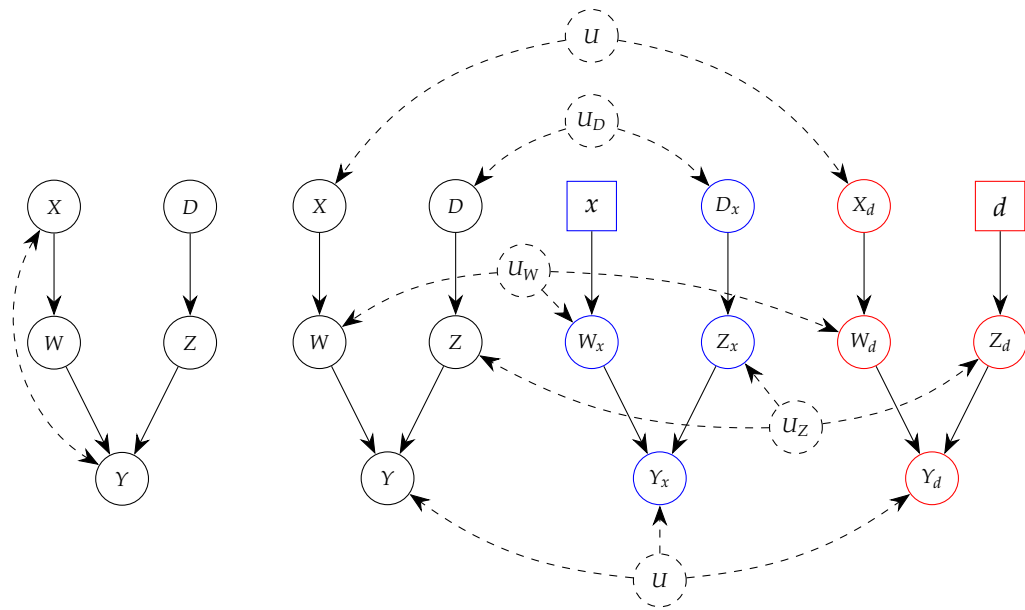
(a) A causal diagram.

(b) A parallel worlds graph of (a) for $y_x \wedge x' \wedge z_d \wedge d$. Colors are used here to distinguish the observed and fixed nodes that belong to different parallel worlds: black for the non-interventional world, blue for the world induced by $do(X = x)$, and red for the world induced by $do(D = d)$. Note that node U is drawn twice for clarity due to its many endpoints.

Figure 1: An example causal diagram and a corresponding parallel worlds graph.
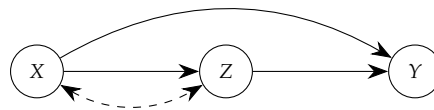
Figure 2: A graph for the example on identifiability of a conditional counterfactual query $P(y_x|z_x \wedge x')$.

$$
\begin{aligned}
P(y_x|z_x \wedge x') &= \frac{P(y_x \wedge z_x \wedge x')}{\sum_y P(y_x \wedge z_x \wedge x')} \\
&= \frac{P(y_{xz} \wedge z_x \wedge x')}{\sum_y P(y_{xz} \wedge z_x \wedge x')} && \text{(composition)} \\
&= \frac{P(y_{xz}|z_x \wedge x')P(z_x \wedge x')}{\sum_y P(y_{xz}|z_x \wedge x')P(z_x \wedge x')} \\
&= \frac{P(y_{xz})P(z_x \wedge x')}{P(z_x \wedge x')\sum_y P(y_{xz})} && \text{(independence restrictions)} \\
&= P(y_{xz}) \\
&= P_{xz}(y)
\end{aligned}
$$

Thus, we find that the answer to our initial question is simply the probability of hiring if the applicant's education level and work experience were changed to x and z, respectively. In the above derivation, we used the notions of composition and independence restrictions (Holland, 1986; Pearl, 1995; Halpern, 1998; Pearl, 2009). Composition is one of the axioms of counterfactuals stating that if a variable is forced to a value that it would have taken without the intervention, then the intervention will not affect other variables in the system. In this case, intervention setting Z_x to z has no effect on Y_x because we have observed $Z_x = z$, thus we can add Z to the intervention set of Y_x. Independence restrictions state if the observed parents of a variable are intervened on, then the counterfactual is independent of any other observed variable when their parents are also held fixed, if there are no paths between the variables via latent variables. In this case $Y_{x,z}$ is independent of Z_x and X because there is no path via latent variables connecting Y to Z or X in G.

In this example, the interventional distribution $P_{x,z}(y)$ can be further identified from the observed joint distribution $P(x, z, y)$ as $P(y|x, z)$ via the second rule of do-calculus by noting that Y is d-separated from X and Z in the graph when the outgoing edges of X and Z are removed. Thus, the answer to our initial question can be further refined into the probability of hiring among applicants with education level x and work experience z. The **cfid** package provides this kind of identification pipeline from the counterfactual level down to the lowest possible level in the causal hierarchy.

3 Algorithms for identifying counterfactual queries

Manual identification of counterfactuals is challenging and more nuanced than identification of interventional distributions due to fixed values and non-uniqueness of counterfactual variables in the parallel worlds graph. Therefore, identification of a counterfactual query can be achieved in several ways. First, we may find that the query is identifiable and thus we can express it in terms of purely interventional distributions. In contrast, we may find that the query is not identifiable, meaning that is not possible to represent it in terms of purely interventional distributions. Alternatively, we may find that the query is *inconsistent* meaning that the same counterfactual variable has been assigned at least two different values in the conjunction, and thus the query is identified as a zero-probability event. For example, suppose we are tasked with identifying $P(y_x, y'_z)$ but we find that Y_x and Y_z are actually the same variable, and thus cannot attain two different values y and y' simultaneously. For conditional counterfactual queries, there is also a fourth option where the query is undefined if the conditioning conjunction is inconsistent.

Algorithmic identification of interventional distributions takes advantage of the so-called *C-component factorization* (Tian and Pearl, 2002; Shpitser and Pearl, 2006) which also plays a key role in the identification of counterfactual queries. The *maximal C-components* of a causal diagram are obtained by partitioning the vertices $\mathbf{V}$ related to observed variables of the graph such that two vertices $A, B \in B$ in the same partition are connected by a path with edges into A and B where every node on the path in $\mathbf{V}$ except A and B is a collider, and A and B are not connected to any other partitions via such paths. Maximal C-components are defined analogously for parallel worlds graphs and counterfactual graphs with the restriction that we do not consider vertices that correspond to fixed values to belong to any C-component. The set of maximal C-components of a DAG G is denoted by $C(G)$. As an example, the maximal C-components of the graph of Figure 1(b) are $\{X, X_d, Y, Y_x, Y_d\}$, $\{D, D_x\}$, $\{Z, Z_x, Z_d\}$, and $\{W, W_x, W_d\}$.

We recall the ID* and IDC* algorithms of Shpitser and Pearl (2007) which are depicted in Figures 3 and 4 for identifying counterfactual queries and conditional counterfactual queries, respectively. Both algorithms are sound and complete (Shpitser and Pearl, 2008, Theorems 26 and 31), meaning that when they succeed in identifying the query, the expression returned is equal to $P(\gamma)$ or $P(\gamma|\delta)$, respectively, and when they fail, the query is not identifiable. We aim to characterize the operation of these algorithms on an intuitive level and provide line-by-line examples of their operation via examples.

> function **ID***(G, γ)
> INPUT: G a causal diagram, γ a conjunction of counterfactual events
> OUTPUT: an expression for $P(\gamma)$ in terms of P_*, or **FAIL**
>
> 1. if $\gamma = \varnothing$, return 1
>
> 2. if $(\exists x_{x'..} \in \gamma)$, return 0
>
> 3. if $(\exists x_{x..} \in \gamma)$, return **ID***$(G, \gamma \setminus \{x_{x..}\})$
>
> 4. $(G', \gamma') = $ **make-cg**(G, γ)
>
> 5. if $\gamma' = $ **INCONSISTENT**, return 0
>
> 6. if $C(G') = \{\mathbf{S}^1, \ldots, \mathbf{S}^k\}$, return $\sum_{V(G') \setminus \gamma'} \prod_{i=1}^{k}$ **ID***$(G, \mathbf{s}^i_{v(G') \setminus \mathbf{s}^i})$
>
> 7. if $C(G') = \{\mathbf{S}\}$, then
>
>> 8. if $(\exists \mathbf{x}, \mathbf{x}')$ s.t. $\mathbf{x} \neq \mathbf{x}', \mathbf{x} \in \mathrm{sub}(\mathbf{S}), \mathbf{x}' \in \mathrm{ev}(\mathbf{S})$, throw **FAIL**
>> 9. else, let $\mathbf{x} = \cup\, \mathrm{sub}(\mathbf{S})$, return $P_{\mathbf{x}}(\mathrm{var}(\mathbf{S}))$.

Figure 3: Counterfactual identification algorithm ID* by Shpitser and Pearl (2007).

We begin by describing the ID* algorithm. On line 1, we check for an empty conjunction, which by convention has probability 1. Line 2 checks for direct inconsistencies meaning counterfactual variables that are intervened on but simultaneously observed to have a different value than the intervention. Such counterfactuals violate the Axiom of Effectiveness (Pearl, 2009), and if found, we return probability 0. Line 3 removes tautological counterfactuals from the conjunction meaning counterfactuals where the variable was observed to have the value it was forced to take via intervention. Line 4 calls the make-cg algorithm to construct the counterfactual graph G' and the corresponding conjunction γ' where some counterfactual variables may have been relabeled due to equivalence between counterfactual variables. We leave the details of the make-cg algorithm and the related core results to the supplementary material. In summary, the output G' of make-cg is a refined version of the parallel worlds graph of G and γ, where each counterfactual variable is unique. Similarly, if some variables in γ were found to be equivalent, then those variables are replaced in γ' by their new representatives in G'. If as a result of this operation the refined conjunction γ' is now inconsistent, we again return probability 0. The next two lines take advantage of the C-component factorization of the counterfactual graph G', analogously to the ID algorithm. If there is more than one maximal C-component of G', then we proceed to line 6 where the original query is decomposed into a set of subproblems, each of which we again call ID* for. Note that the sets $\mathbf{S}^i$ are sets of counterfactual variables, but we may interpret them as counterfactual conjunctions in the subsequent recursive calls. Similarly, we may interpret γ' as a set of counterfactual variables when carrying out the outermost summation over the possible values of the counterfactual variables in $V(G') \setminus \gamma'$. In cases where a set $\mathbf{S}^i$ contains counterfactual variables, the intervention $do(v(G') \setminus \mathbf{s}^i)$ should be understood as merging of the subscripts, e.g., if $\mathbf{S}^i = \{Y_x\}$ and $V(G') \setminus \mathbf{S}^i = \{Z\}$, and Y_x has the value y in γ', then $\mathbf{s}^i_{v(G') \setminus \mathbf{s}^i} = y_{x,z}$.

If there is only one C-component, we enter line 7 that serves as the base case. There are now only two options. If there is an inconsistent value assignment on line 8 such that at least one of the values is in the subscript, then the query is not identifiable, and we fail. If there is no such conflict, we can take the union of all the subscripts in γ' and return their effect on the variables in γ' on line 9.

> function **IDC***(G, γ, δ)
> INPUT: G a causal diagram, γ, δ conjunctions of counterfactual events
> OUTPUT: an expression for $P(\gamma|\delta)$ in terms of P_*, or **FAIL**, or **UNDEFINED**
>
> 1. if **ID***$(G, \delta) = 0$, return **UNDEFINED**
>
> 2. $(G', \gamma' \wedge \delta') = $ **make-cg**$(G, \gamma \wedge \delta)$
>
> 3. if $\gamma' \wedge \delta' = $ **INCONSISTENT**, return 0
>
> 4. if $(\exists y_\mathbf{x} \in \delta')$ s.t. $(Y_\mathbf{x} \perp\!\!\!\perp \gamma')G'_{\underline{y_\mathbf{x}}}$, return **IDC***$(G, \gamma'_{y_\mathbf{x}}, \delta' \setminus \{y_\mathbf{x}\})$
>
> 5. else, let $P' = $ **ID***$(G, \gamma' \wedge \delta')$, return $P'/P'(\delta)$

Figure 4: Conditional counterfactual identification algorithm IDC* by Shpitser and Pearl (2007).

In contrast, the IDC* algorithm is simpler, as it leverages the ID* algorithm. The consistency of the conditioning conjunction δ is first confirmed on line 1, and if δ is found to be inconsistent, then the conditional probability $P(\gamma|\delta)$ is undefined, and we return. Line 2 applies the make-cg algorithm to the joint conjunction $\gamma \wedge \delta$ to construct the corresponding counterfactual graph G' and the restructured version of the conjunction, $\gamma' \wedge \delta'$. If $\gamma' \wedge \delta'$ was found to be inconsistent, we return probability 0 on line 3. Line 4 takes advantage of conditional independence relations implied by the counterfactual graph G' and the second rule of do-calculus to add variables as interventions to γ' by removing them from δ'. If the necessary d-separation holds, we initiate a recursive call to IDC* again. Finally on line 5, if no more variables can be removed from δ', we simply apply the ID* algorithm to the joint conjunction $\gamma' \wedge \delta'$ and obtain the identifying functional as a standard conditional probability from the distribution returned by ID*.

3.1 Examples on the identifiability algorithm

We recall the counterfactual conjunction $\gamma = y_x \wedge x' \wedge z_d \wedge d$ from Section 2.2 and describe how the ID* algorithm operates when applied to $P(\gamma)$ in the graph of Figure 1(a), which we will label as G in the context of this example. We start from line 1 and continue to line 2 as γ is not an empty conjunction. On line 2, we note that γ does not contain any inconsistencies, similarly on line 3 we see that γ does

not contain any tautological statements. Thus, we reach line 4 and apply the make-cg algorithm to obtain the counterfactual graph G' and the modified conjunction γ'.

We describe the operation of the make-cg algorithm in this instance. The goal is to determine which variables in the parallel worlds graph of Figure 1(b) represent the same variable. We consider all variable pairs in a topological order of G that originate from the same non-counterfactual variable in G. First, we can conclude that X and X_d are the same variable, as they have the same functional mechanisms and the same parent U. By the same argument, D and D_x are the same variable with the common parent U_D. The fixed variables x and d cannot be merged with the other X-derived variables and D-derived variables, respectively, as their functional mechanisms are different. Next, we merge W and W_d because their X-derived parents (X and X_d) were found to be the same and they have the same parent U_W. However, W_X cannot be merged with the other two W-derived variables, because X (and thus X_d) was observed to attain the value x' in γ, but x has the value x as fixed by the intervention. In contrast, we can merge the triplet Z, Z_x and Z_d, because their D-derived parents attain the same value, and they have the same parent U_Z. The intuition is that because the U-variables are shared between worlds, intervention and observation have the same effect if the observed values agree with the values fixed by intervention. This is a consequence of the Axiom of Composition as was considered in the example of Section 2.2.1. Finally, we consider the Y-derived variables and merge Y_x and Y_d because their Z-derived parents are the same, their W-derived parents are the same, and they have the same parent U. The variable Y_x cannot be merged with the other two, because its W-derived parent W_x was not the same variable as W and W_d.

Consequently, we must choose a name for each merged variable. This choice is arbitrary and plays no role in the correctness of the algorithm; the difference is purely notational. In this example, we pick the original name with the fewest subscripts to represent the merged variable, i.e., X represents the merged pair X, X_d, Z represents the merged triplet Z, Z_x, Z_d, W represents the merged pair W, W_d and finally Y represents the merged pair Y, Y_d. Note that because the Z-derived variables were all merged but d was not merged with D and D_x, we essentially have two D-derived parents for the merged Z. In such scenarios, we simply omit the fixed version of the parent variable from the graph, because this scenario may only arise if the parent variables were found to have the same value, thus their role in the functional mechanisms of their children is identical. Lastly, we may restrict our attention to those counterfactual variables that are ancestral to the query γ in this merged graph, which are x, W_x, Y_x, Z, D, X and U

Thus, we obtain the counterfactual graph G' for γ depicted in Figure 5 using once again the convention that unobserved variables with only one child are not drawn. As a result of the variable merges, we also update our original conjunction γ with references to the merged variables to obtain $\gamma' = y_x \wedge x' \wedge z \wedge d$. The new conjunction γ' is not inconsistent on line 5, and thus we continue.
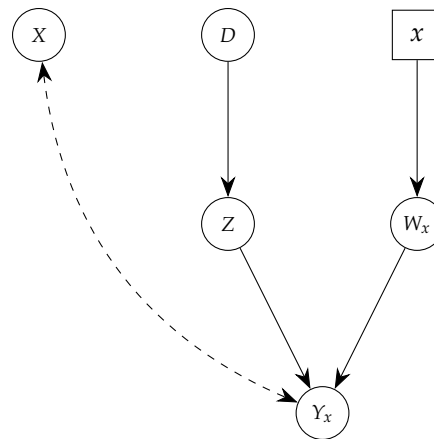
Figure 5: Counterfactual graph G' for $y_x \wedge x' \wedge z_d \wedge d$ of the graph of Figure 1(a).

On line 6 we first determine the maximal C-components of the counterfactual graph G' which are $\{X, Y_x\}$, $\{Z\}$, $\{W_x\}$ and $\{D\}$. By the C-component factorization we have that

$$P(y_x \wedge x' \wedge z \wedge d) = \sum_w P(y_{x,z,w,d} \wedge x'_{z,w,d}) P(z_{y,x,w,d}) P(w_{x,y,z,d}) P(d_{y,x,z,w}), \qquad (1)$$

which means that we launch four recursive calls to ID* to identify each of the terms in the right-hand side expression. We will consider the last three terms first as they result in a similar simple path through the algorithm. For each of these terms, the counterfactual graph will contain a single non-fixed vertex ($Z_{y,x,w,d}$, $W_{x,y,z,d}$ and $D_{y,x,z,w}$, respectively). Because the conjunctions are not empty, there are

no inconsistencies or tautologies, and only a single C-component, we end up on line 7 in each case. None of the terms contain value assignments that would conflict with the subscript and thus each term is identified as an interventional distribution on line 9. Note that when line 7 is reached, redundant subscripts should be removed, i.e., those subscript variables that are not ancestors of the counterfactual variables in γ' in the counterfactual graph G'. Otherwise, a conflict may be found erroneously on line 8. This operation was not formally included in the algorithm by Shpitser and Pearl (2007), but nonetheless carried out in a running example by Shpitser and Pearl (2008). Thus, $P(z_{y,x,w,d}) = P_d(z)$, $P(w_{x,y,z,d}) = P_x(w)$ and $P(d_{y,x,z,w}) = P(d)$. For the first term $P(y_{x,z,w,d} \wedge x'_{z,w,d})$, the only difference is that the counterfactual graph has two non-fixed vertices, but the outcome is the same and we end up on line 7 due to the single C-component containing $Y_{x,z,w,d}$ and $X_{z,w,d}$. There are no conflicts this time either, and we obtain $P(y_{x,z,w,d} \wedge x'_{z,w,d}) = P_{w,z}(y, x')$. Thus, we obtain the identifying functional of the counterfactual query:

$$P(y_x \wedge x' \wedge z_d \wedge d) = \sum_w P_{w,z}(y, x')P_d(z)P_x(w)P(d).$$

Next, we will consider an example that causes a conflict at line 7 resulting in a non-identifiable counterfactual query. Suppose that we also have an edge from X to Y in the graph of Figure 1(a) and we wish to identify the same counterfactual query $P(y_x \wedge x' \wedge z \wedge d)$ as in the previous example in this modified graph. The ID* algorithm proceeds similarly as in the previous example up to line 4 where we obtain a slightly different counterfactual graph, which is the graph of Figure 5, but with the corresponding extra edge from X to Y_x. Thus, the algorithm proceeds similarly to line 6, where the C-component factorization is the same as (1). The last three terms are still identifiable, but this time the first term $P(y_{x,z,w,d} \wedge x'_{z,w,d})$ is problematic. On line 7 after removing redundant interventions, the term takes the form $P(y_{x,z,w} \wedge x'_{z,w})$ which now contains a conflict, because x appears in the subscript but x' is observed at the same time, resulting in non-identification on line 8.

We return to the example presented in Section 2.2.1 and apply the IDC* algorithm to identify the counterfactual query $P(y_x | z_x \wedge x')$ in the graph of Figure 2, which we will again refer to as G in the context of this example. We trace the application of IDC*$(G, y_x, z_x \wedge x')$. On line 1, the ID* algorithm is applied to $z_x \wedge x'$, which is not identifiable, but also not inconsistent. Continuing to line 2, we apply the make-cg algorithm to construct the counterfactual graph G', which is shown in Figure 6(a). First, the parallel worlds graph is constructed and **make-cg** proceeds to determine which variable pairs can be merged (see the Supplementary Material for details on the **make-cg** algorithm).
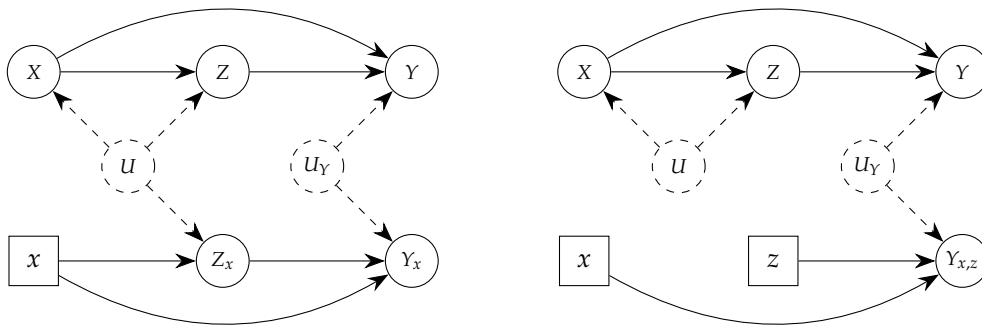
(a) Parallel worlds graph for $y_x \wedge z_x \wedge x'$ (the counterfactual graph).

(b) Parallel worlds graph for $y_{x,z} \wedge x'$ (the counterfactual graph).

Figure 6: Counterfactual graphs used during the derivation of $P(y_x | z_x \wedge x')$.

Because X was observed to have the value x', but the intervention for Z and Y has the value x, we cannot merge X and x. Similarly, the X-parent of Z in both worlds has a different value, meaning that Z and Z_x cannot be merged either. Finally, through the same reasoning, Y and Y_x will remain unmerged due to the difference in the Z-parent. Thus, the parallel worlds graph is the counterfactual graph G' in this instance. This also means that $\gamma' = \gamma$ and $\delta' = \delta$ in the output of make-cg.

On line 3, we check for inconsistencies in $y_x \wedge z_x \wedge x'$, but there are none. Next on line 4, we check whether either of the two variables in δ' are d-separated from γ' when outgoing edges of that variable have been removed. We can see that X is not d-separated from Y_x, because the path $X \leftarrow U_1 \rightarrow Z_x \rightarrow Y_x$ is open in $G'_{\underline{X}}$. However, Z_x is d-separated from Y_x in $G'_{\underline{Z_x}}$ (note that x is fixed by intervention, and thus the path $Z_x \leftarrow x \rightarrow Y_x$ is not an open backdoor path). Thus, line 4 adds an intervention on Z to Y_x because Y_x is a descendant of Z_x in G', and removes Z_x from δ', and we call IDC*$(G', y_{x,z}, x')$.

We now trace this new recursive call. Once again on line 1, ID* is not able to identify the effect, but is also not inconsistent. Next, we construct a new counterfactual graph G'' for $y_{x,z} \wedge x'$ as depicted in Figure 6(b). Using similar reasoning as before, the make-cg algorithm is not able to merge any nodes this time either and thus the parallel worlds graph is the counterfactual graph. Again, this means that $\gamma'' = \gamma'$ and $\delta'' = \delta'$ in the output of make-cg. Line 3 checks again for inconsistencies in $y_{x,z} \wedge x'$, but there are none. Thus, we arrive again on line 4, but this time X is d-separated from $Y_{x,z}$ in $G''_{\underline{X}}$. Now, $Y_{x,z}$ is not a descendant of X in G'' so no new intervention is added to $Y_{x,z}$, and x' is removed from δ''. Because the conditioning δ-argument of the next IDC* call is now empty, we can call ID* directly as ID*$(G, y_{x,z})$, but $P(y_{x,z})$ is no longer a counterfactual quantity, but an interventional distribution and thus directly identifiable from P_* as $P_{x,z}(y)$.

We note the difference compared to the manual identification strategy we used in Section 2.2.1 to obtain identifiability. Instead of using axioms of counterfactuals or independence restrictions explicitly, the ID* and IDC* algorithms take full advantage of the counterfactual graph and the conditional independence relations between the counterfactual variables implied by it.

4 Using the cfid package

The **cfid** package is available from CRAN at https://cran.r-project.org/package=cfid and can be obtained in R using the following commands:

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

Development of **cfid** takes place on GitHub https://github.com/santikka/cfid.

The main contributions of the **cfid** package are the implementations of the ID* and IDC* algorithms. The package also provides reimplementations of the ID and IDC algorithms for interventional distributions from the **causaleffect** package, but without relying on the **igraph** (Csardi and Nepusz, 2006) package. In fact, **cfid** has no mandatory package dependencies or installation requirements. The **cfid** package provides its own text-based interface for defining graphs, which closely follows the syntax of the **dagitty** package, and also supports other external graph formats directly. Installation of the **igraph** and **dagitty** packages is optional and required only if the user wishes to import or export graphs using the aforementioned packages.

The inclusion of the identifiability algorithms for interventional distributions enables a full identification pipeline. First, we determine the identifiability of a counterfactual query from the set of all interventional distributions, and then proceed to identify each interventional distribution that appears in the identifying functional of the counterfactual from the joint observed probability distribution of the causal model. The level of attempted identification can be specified by the user.

4.1 Defining causal diagrams

Causal diagrams (i.e., DAGs) in **cfid** are constructed via the function dag

```
dag(x, u = character(0L))
```

where x is a single character string in a syntax analogous to the DOT language for GraphViz (and the **dagitty** package), and u is an optional character vector of variable names that should be considered unobserved in the graph. Internally, a semi-Markovian representation is always used for DAGs where each latent variable has at most two children, which is obtained from the input via the latent projection (Verma and Pearl, 1990).

As an example, the graph of Figure 2 can be constructed as follows:

```
R> g <- dag("X -> Z -> Y; X -> Y; X <-> Z")
```

Above, individual statements are separated by a semicolon for additional clarity, but this is optional, and a space would suffice. More generally, the input of dag consists of statements of the form $n_1 e_1 n_2 e_2 \cdots e_k n_k$ where each e_i symbol must be a supported edge type, i.e., ->, <- or <->, and each n_i symbol must correspond to single node such as X or a subgraph such as {X,Y,Z} or {X -> Y}. Subgraphs are enclosed within curly braces, and they follow the same syntax as x. Subgraphs can also be nested arbitrarily. An edge of the form X -> {...} means that there is an edge from X to all vertices in the subgraph, and the interpretation for <- and <-> is analogous. Individual statements in the graph definition can be separated by a semicolon, a space, or a new line. Commas can be used within subgraphs to distinguish vertices, but a space is sufficient.

The same DAG can often be defined in many ways. For example, we could also define the graph of Figure 2 using a subgraph construct as follows:

```
R> g <- dag("X -> {Z, Y}; Z -> Y; X <-> Z")
```

We could also combine the outgoing edge of Z and the bidirected edge into a single statement:

```
R> g <- dag("X -> {Z, Y}; X <-> Z -> Y;")
```

The edge from Z to Y could be defined in the subgraph as well:

```
R> g <- dag("Z <-> X -> {Z -> Y}")
```

The output of dag is an object of class "dag" which is a square adjacency matrix of the graph, with additional attributes for the vertex labels and latent variables and a print method. Graph definitions that imply cycles or self-loops will raise an error. Examples of more complicated graph constructs can be found from the **cfid** package documentation for the dag function. Graphs using supported external formats can be converted to "dag" objects via the function import_graph. Conversely, "dag" objects can be exported in supported external formats using the function export_graph.

4.2 Defining counterfactual variables and conjunctions

Counterfactual variables are defined via the function counterfactual_variable or its shorthand alias cf

```
counterfactual_variable(var, obs = integer(0L), sub = integer(0L))
cf(var, obs = integer(0L), sub = integer(0L))
```

The first argument var is a single character string naming the variable, e.g., "Y". The second argument obs describes the value assignment as a single integer. The value of this argument does not describe the actual value taken by the variable, but simply the assignment level, meaning that obs = 1 is a different value assignment than obs = 0, but the actual values that the counterfactual variable takes need not necessarily be 1 and 0. The idea is similar to the internal type of factors in R. Finally, sub defines the set of interventions as a named integer vector, where the actual values correspond to the intervention levels, and not actual values, analogous to obs. The output of cf is an object of class "counterfactual_variable".

As an example, the counterfactual variables in $\gamma = y_x \wedge x' \wedge z_d \wedge d$ can be defined as follows:

```
R> v1 <- cf(var = "Y", obs = 0L, sub = c(X = 0L))
R> v2 <- cf(var = "X", obs = 1L)
R> v3 <- cf(var = "Z", obs = 0L, sub = c(D = 0L))
R> v4 <- cf(var = "D", obs = 0L)
R> list(v1, v2, v3, v4)

[[1]]
y_{x}

[[2]]
x'

[[3]]
z_{d}

[[4]]
d
```

The print method for "counterfactual_variable" objects mimics the notation used in this paper in LaTeX syntax.

Individual "counterfactual_variable" objects can be combined into a counterfactual conjunction via the function counterfactual_conjunction or its shorthand alias conj. This function takes arbitrarily many "counterfactual_variable" objects as input. The output of conj is an object of class "counterfactual_conjunction".

```
R> c1 <- conj(v1, v2, v3, v4)
R> c1

y_{x} /\ x' /\ z_{d} /\ d
```

Alternatively, the `+` operator can be used to build conjunctions from counterfactual variables or conjunctions.

```
R> c2 <- v1 + v2
R> c3 <- v3 + v4
R> c2
R> c3
R> c2 + c3

y_{x} /\ x'
z_{d} /\ d
y_{x} /\ x' /\ z_{d} /\ d
```

The subset operator `` `[` `` is supported for counterfactual conjunctions

```
R> c1[c(1, 3)]

y_{x} /\ z_{d}
```

Just as the cf function, the print method for "counterfactual_conjunction" objects mimics the formal notation of using the ∧ symbol to separate individual statements, but this symbol can also be changed by the user.

4.3 Identifying counterfactual queries

Identification of counterfactual queries is carried out by the function identifiable

```
identifiable(g, gamma, delta = NULL, data = "interventions")
```

where g is a causal diagram defined by the function dag, gamma is the counterfactual conjunction γ as a "counterfactual_conjunction" object describing the counterfactual query $P(\gamma)$ to be identified, delta is an optional argument also of class "counterfactual_conjunction" that should be provided if identification of a conditional counterfactual $P(\gamma|\delta)$ is desired instead. Finally, data defines the available probability distributions for identification. The default value "interventions" means that identification is carried out to the intervention level, i.e., by using only the set of all interventional distributions P_*. The alternatives are "observations", where only the joint observed probability distribution $P(\mathbf{v})$ is available, and "both" where both P_* and $P(\mathbf{v})$ are available, and identification in terms of $P(\mathbf{v})$ is prioritized.

We reassess the identifiability examples of Section 2.3.1 using the **cfid** package. The conjunction of the query γ for the first two examples is the same as c1 in the previous section. We define the graphs for the identifiable case in Figure 1(a) and the non-identifiable case with the additional edge from X to Y:

```
R> v1 <- cf(var = "Y", obs = 0L, sub = c(X = 0L))
R> v2 <- cf(var = "X", obs = 1L)
R> v3 <- cf(var = "Z", obs = 0L, sub = c(D = 0L))
R> v4 <- cf(var = "D", obs = 0L)
R> c1 <- conj(v1, v2, v3, v4)
R> g1 <- dag("Y <-> X -> W -> Y <- Z <- D")
R> g2 <- dag("Y <-> X -> W -> Y <- Z <- D; X -> Y")
R> out1 <- identifiable(g1, c1)
R> out2 <- identifiable(g2, c1)
R> out1
R> out2

The query P(y_{x} /\ x'/\ z_{d} /\ d) is identifiable from P_*.
Formula: \sum_{w} P_{w,z}(y,x')P_{x}(w)P_{d}(z)P(d)

The query P(y_{x} /\ x'/\ z_{d} /\ d) is not identifiable from P_*.
```

The identifiable function returns an object of class "query", whose print method provides a summary of the identification result. Objects of this class are lists with the following elements:

id A logical value that is TRUE if the counterfactual query is identifiable and FALSE otherwise.

formula An object of class "functional" representing the identifying functional. The format method for "functional" objects provides the formula of the counterfactual query in LaTeX syntax when the query is identifiable. Otherwise, formula is NULL.

undefined A logical value that is TRUE if the conditional counterfactual query is found to be undefined.

query The original query as a "counterfactual_conjunction" object.

data The data argument passed to identifiable.

By default, the notation of Shpitser and Pearl (2007) is used for interventional distributions, where interventions are denoted using the subscript, e.g. $P_x(y)$. If desired, the notation can be swapped to Pearl's notation with the explicit do-operator denoting interventions, e.g., $P(y|\mathrm{do}(x))$. This can be accomplished via the use_do argument of the format method for "functional" objects (passed here via the print method):

```
R> print(out1[["formula"]], use_do = TRUE)
```

```
\sum_{w} P(y,x'|do(w,z))P(w|do(x))P(z|do(d))P(d)
```

For the third example of Section 2.3.1, we have already defined the counterfactual variable Y_x of the query as v1 and the observation $X = x'$ in the condition as v2. We still need to define the graph of Figure 2 and the other conditioning variable Z_x:

```
R> g3 <- dag("Z <-> X -> {Z -> Y}")
R> v5 <- cf("Z", 0, c(X = 0))
R> identifiable(g3, v1, v5 + v2)
```

```
The query P(y_{x}|z_{x} /\ x') is identifiable from P_*.
Formula: P_{x,z}(y)
```

Recall from Section 2.2.1, that this interventional distribution can be further identified, which can be accomplished by setting the data argument to "observations" in identifiable (or to "both" in this case):

```
R> identifiable(g3, v1, v5 + v2, data = "observations")
```

```
The query P(y_{x}|z_{x} /\ x') is identifiable from P(v).
Formula: P(y|x,z)
```

4.4 Formatting output for reports

The LaTeX formatting of the formulas returned by the identifiable function enables them to be directly rendered as mathematics, for example in an R Markdown or a Sweave document. For instance, we can write an inline markdown code chunk within a mathematics environment, where we use the format method with the formula element of a "query" object.

```
\(`r format(out1$formula)`\)
```

This would render as $\sum_w P_{w,z}(y, x')P_x(w)P_d(z)P(d)$ in the document. Similarly, we could use the use_do argument to render the formula such that the do-operator is used instead to represent the interventions.

```
\(`r format(out1$formula, use_do = TRUE)`\)
```

This would render as $\sum_w P(y, x'|do(w,z))P(w|do(x))P(z|do(d))P(d)$.

Similarly, we can also directly render "counterfactual_query" objects into mathematics. We replace the default variable separator string, defined via the var_sep argument, with " \\wedge " to properly render the $\wedge$ symbol. The leading and trailing spaces are used so that the command name does not get mixed with the variable names.

```
\(`r format(c1, var_sep = " \\wedge ")`\)
```

When rendered, this produces $y_x \wedge x' \wedge z_d \wedge d$.

5 Summary

The **cfid** package provides an easy-to-use interface to identifiability analysis of counterfactual queries. The causal diagram of the causal model can be specified by the user via an intuitive interface, and a variety of commonly used external graph formats are supported. The results from the identifiability algorithms are wrapped neatly in LaTeX syntax to be readily used in publications or reports. This tutorial demonstrates the features of the package and provides insight into the core algorithms it implements.

Acknowledgments

References

C. Avin, I. Shpitser, and J. Pearl. Identifiability of path-specific effects. In *Proceedings of International Joint Conference on Artificial Intelligence*, volume 19, pages 357–363, 01 2005. [p333, 334]

A. Balke and J. Pearl. Counterfactual probabilities: Computational methods, bounds and applications. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 46–54, 1994a. [p334]

A. Balke and J. Pearl. Probabilistic evaluation of counterfactual queries. In *Proceedings of the 12th AAAI National Conference on Artificial Intelligence*, pages 230–237, 1994b. [p334]

E. Bareinboim and J. Pearl. Causal inference by surrogate experiments: z-identifiability. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, pages 113–120, 2012. [p333]

M. Chen, V. Chernozhukov, I. Fernandez-Val, and B. Melly. *Counterfactual: Estimation and Inference Methods for Counterfactual Analysis*, 2020. URL https://CRAN.R-project.org/package=Counterfactual. R package version 1.2. [p333]

G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL https://igraph.org. [p340]

J. Y. Halpern. Axiomatizing causal reasoning. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 202–210, 1998. [p335]

P. W. Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81(396): 945–960, 1986. URL https://doi.org/10.1080/01621459.1986.10478354. [p335]

Y. Huang and M. Valtorta. Pearl's calculus of intervention is complete. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 217–224. AUAI Press, 2006. [p333]

J. Karvanen. *R6causal: R6 Class for Structural Causal Models*, 2022. R package version 0.6.1. [p333]

Y. Kivva, E. Mokhtarian, J. Etesami, and N. Kiyavash. Revisiting the general identifiability problem. In *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence*, volume 180, pages 1022–1030. PMLR, 2022. [p333]

M. J. Kusner, J. Loftus, C. Russell, and R. Silva. Counterfactual fairness. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4069–4079, 2017. [p332]

S. Lee, J. D. Correa, and E. Bareinboim. General identifiability with arbitrary surrogate experiments. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence*, volume 115, pages 389–398. PMLR, 2019. [p333]

J. Pearl. Causal diagrams for empirical research. *Biometrika*, pages 669–710, 1995. URL https://doi.org/10.1093/biomet/82.4.669. [p335]

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition, 2009. [p332, 335, 337]

I. Shpitser and J. Pearl. Identification of joint interventional distributions in recursive semi-Markovian causal models. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, pages 1219–1226. AAAI Press, 2006. [p333, 336]

I. Shpitser and J. Pearl. What counterfactuals can be tested. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 352–359. AUAI Press, 2007. [p333, 336, 337, 339, 343]

I. Shpitser and J. Pearl. Complete identification methods for the causal hierarchy. *Journal of Machine Learning Research*, 9(64):1941–1979, 2008. [p333, 336, 339]

H. Stoll, G. King, L. Zeng, C. Gandrud, and B. Sabath. *WhatIf: Software for Evaluating Counterfactuals*, 2020. URL https://CRAN.R-project.org/package=WhatIf. R package version 1.5-10. [p333]

J. Textor, B. van der Zander, M. S. Gilthorpe, M. Liśkiewicz, and G. T. Ellison. Robust causal inference using directed acyclic graphs: The R package dagitty. *International Journal of Epidemiology*, 45(6): 1887–1894, 2017. URL https://doi.org/10.1093/ije/dyw341. [p333]

J. Tian and J. Pearl. A general identification condition for causal effects. In *Proceedings of the 19th AAAI National Conference on Artificial Intelligence*, pages 567–573, 2002. [p336]

S. Tikka and J. Karvanen. Identifying causal effects with the R package causaleffect. *Journal of Statistical Software*, 76(12):1–30, 2017. URL https://doi.org/10.18637/jss.v076.i12. [p333]

S. Tikka and J. Karvanen. Surrogate outcomes and transportability. *International Journal of Approximate Reasoning*, 108:21–37, 2019. [p333]

S. Tikka, A. Hyttinen, and J. Karvanen. Causal effect identification from multiple incomplete data sources: A general search-based approach. *Journal of Statistical Software*, 99(5):1–40, 2021. URL https://doi.org/10.18637/jss.v099.i05. [p333]

T. S. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*, pages 255–270, 1990. [p340]

J. Zhang and E. Bareinboim. Fairness in decision-making — the causal explanation formula. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 2037–2045, 2018. [p332]

Santtu Tikka
Department of Mathematics and Statistics, University of Jyväskylä
P.O. Box 35, FI-40014, Finland
santtu.tikka@jyu.fi

vivid: An R package for Variable Importance and Variable Interactions Displays for Machine Learning Models

by Alan Inglis, Andrew Parnell, and Catherine Hurley

Abstract We present vivid, an R package for visualizing variable importance and variable interactions in machine learning models. The package provides heatmap and graph-based displays for viewing variable importance and interaction jointly, and partial dependence plots in both a matrix layout and an alternative layout emphasizing important variable subsets. With the intention of increasing machine learning models' interpretability and making the work applicable to a wider readership, we discuss the design choices behind our implementation by focusing on the package structure and providing an in-depth look at the package functions and key features. We also provide a practical illustration of the software in use on a data set.

1 Introduction

Our motivation behind the creation of the vivid package is to investigate machine learning models in a way that is simple to understand while also offering helpful insights into how variables affect the fit. We do this through the use of heatmaps, network graphs, and both a generalized pairs plot style partial dependence plot (PDP) (Friedman 2000) and a space saving PDP based on key variable subsets. While the techniques and fundamental goals of these visualizations have been discussed in Inglis, Parnell, and Hurley (2022a), we focus here on the implementation details of the package by providing a complete listing of the functions and arguments included in the vivid package with further examples indicating advanced usage beyond that previously shown. In this work we examine the decisions made when designing the package and provide an in-depth look at the package functions and features with the intention of making the work applicable to a larger readership. This article outlines the general architectural principles implemented in vivid, such as the data structures we use and data formatting, function design, filtering techniques, and more. We illustrate each function by way of a practical example. Our package is available on the Comprehensive R Archive Network at https://cran.r-project.org/web/packages/vivid or on GitHub at https://github.com/AlanInglis/vivid.

In recent years machine learning (ML) algorithms have emerged as a valuable tool for both industry and science. However, due to the black-box nature of many of these algorithms it can be challenging to communicate the reasoning behind the algorithm's decision-making processes. With the need for transparency in ML growing it is important to gain understanding and clarity about how these algorithms are making predictions (Antunes et al. 2018; Felzmann et al. 2019). Many R packages are now available that aid in creating interpretable machine learning (IML) models such as iml (Molnar, Bischl, and Casalicchio 2018), DALEX (Biecek 2018), and lime (Hvitfeldt, Pedersen, and Benesty 2022). For a comprehensive review of IML, see Molnar (2022) and Biecek and Burzykowski (2021).

How we choose to visualize aspects of the model output is of vital importance to how a researcher can interpret and communicate their findings. Consequently, model summaries such as variable importance and variable interactions (VImp and VInt; together we term these VIVI) are frequently used in various fields to comprehend and explain the hidden structure in an ML fit. In ecology they are employed to determine the causes of ecological phenomena (e.g. Murray and Conner 2009); in meteorology VImp measures and partial dependence plots are used to examine air quality (e.g. Grange et al. 2018); in bioinformatics, understanding gene-environment interactions have made these measures an important tool for genomic analysis (e.g. Chen and Ishwaran 2012).

In Table 1 we summarize VIVI measures and visualizations provided by a selection of R packages. VIVI measures are categorized as global or local, depending on whether they refer to the entire predictor space or a specific sub-region. For instance, Individual Conditional Expectation (ICE) curves (Goldstein et al. 2015) and Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro, Singh, and Guestrin 2016) are considered local methods (implementations of which can be found in the ICEbox package (Goldstein et al. 2015) and the lime package (Hvitfeldt, Pedersen, and Benesty 2022)), whereas partial dependence and permutation importance represent global methods. Packages that incorporate local or global methods can be found under the column 'Method' in Table 1. Additionally, VIVI measures can be broken down into model specific (embedded) methods or model agnostic methods. The packages listed in Table 1 are grouped by whether they can compute model specific or agnostic measures and can be found under the column 'Measure'. In embedded methods the variable importance is incorporated into the ML algorithm. For example, random forests (RF) (Breiman

Table 1: Summary of a selection of R packages that can be used to assess the variable importance, variable interactions, or partial dependence and if these metrics are global or local and model-specific or model-agnostic. A brief description of available visualizations for evaluating model behavior is also provided.

Package	Description	VIVI	Measure	Method
vip	A general framework for analyzing the behavior of ML models. Also provides PDP based importance and ability to plot Shapley values. Built with ggplot2.	Both	Agnostic	Global
iml	A general framework for analyzing the behavior of ML models. Ability to create lollipop, dot, and barplots. Also includes univariate and bivariate PDPs, ICE curves, LIME, and Shapley visualizations. Built with ggplot2.	Both	Agnostic	Both
flashlight	A general framework for analyzing the behavior of ML models. Ability to plot VIVI measures using barplots. Includes univariate and bivariate PDPs, ICE curves, Global surrogate, and SHAP visualizations. Built with ggplot2.	Both	Agnostic	Global
DALEX	A general framework for analyzing the behavior of ML models. Contains a suite of visualizations including Ceteris Paribus, Shapley, PDPs, model performance, and diagnostic plots. Built with ggplot2.	Both	Agnostic	Both
lime	A general framework for fitting a local interpretable model. Ability to create VImp and model visualizations using barplots and heatmaps. Can also create interactive plots. Built with ggplot2.	VImp	Agnostic	Local
pdp	A general framework for constructing PDPs from various types machine learning models, bivariate, and trivariate PDPs and ICE curves. Built with ggplot2.	VInt	Agnostic	Global
ICEbox	Used to create Individual Conditional Expectation (ICE) plots. Provides univariate and bivariate PDPs and ICE curves. Built with ggplot2.	VInt	Agnostic	Local
randomForestExplainer	Contains a set of model-specific tools to determine which random forests variables are most important. Ability to create VIVI plots displaying the mean minimal depth distribution and conditional minimal depth. Can also display multi-way importance, pairs plots containing different metrics, and Bivariate PDP. Built with ggplot2.	Both	Specific	Global
randomForest	Used to build random forest models. Offers VImp, error rate, and univariate PDPs. Built using base R.	VImp	Specific	Global
EIX	Contains a set of model-specific tools to determine which GBM variables are most important. Ability to create VIVI plots using lollipops, barplots, and heatmaps. Can also display dot and radar plots. Built with ggplot2.	Both	Specific	Global
varImp	Computes random forest VImps for the conditional inference random forest of the party package.	Vimp	Specific	Global
bartMachine	Used to build Bayesian additive regression tree models. Ability to plot VIVI measures with uncertainty included using barplots. Also includes a suite of model diagnostic plots and univariate PDP. Built using base R.	Both	Specific	Global

2001) and gradient boosting machines (GBM) (Friedman 2000) use the tree structure to evaluate the performance of the model. Bayesian additive regression tree models (BART) (Chipman, George, and McCulloch 2010) also use an embedded method to obtain VIVI measures by looking at the proportion of splitting rules for variables or variable pairs used in the trees. The package **randomForestExplainer** (Paluszynska, Biecek, and Jiang 2020) provides a set of tools to understand what is happening inside a random forest and uses the concept of minimal depth (Ishwaran et al. 2010) to assess both importance and interaction strength by examining the position of a variable within the trees. Additionally the **varImp** (Probst 2020) package can be used to compute importance scores for the conditional inference random forest of the **party** package (Strobl et al. 2008). For gradient boosted machines, the **EIX** (Maksymiuk, Karbowiak, and Biecek 2021) package can be used to measure and identify VIVI and visualize the results.

Model-agnostic methods are techniques that can, in principle, be applied to any ML algorithm. Agnostic methods not only provide flexibility in relation to model selection but are also useful for comparing different fitted ML models. An example of a model agnostic approach for evaluating VImps is permutation importance (Breiman 2001). This method calculates the difference in a model's predictive performance following a variable's permutation; implementations are available in the packages **iml**, **flashlight** (Mayer 2023), **DALEX** and **vip** (Greenwell and Boehmke 2020). Each package provides options for specifying the performance metric to use in computing the model performance as well as providing options to select the number of permutations, with **flashlight** and **DALEX** packages additionally providing an option to select the number of observations that should be sampled for the calculation of variable importance. The **vip** package also provides a PDP/ICE-based variable importance method, which quantifies the variability in PDP and ICE plots. For VInts, Friedman's H-statistic (Friedman and Popescu 2008) is an agnostic interaction measure derived from the partial dependence by comparing a pair of variables' joint effects with the sum of their marginal effects. Packages **iml** and **flashlight** provide implementations.

Partial dependence plots (PDPs) were first introduced by Friedman (2000) as a model agnostic way to visualize the relationship between a specified predictor variable and the fit, averaging over other predictors' effects. Similar to PDPs, individual conditional expectation curves (ICE) (Goldstein et al. 2015) show the relationship between a specified predictor and the fit, fixing the levels of other predictors at those of a particular observation. PDP curves are then the average of the ICE curves over all observations in the dataset. R packages offering PDPs include **iml**, **DALEX** (which calls the **ingredients** package (Biecek and Baniecki 2023)), **flashlight**, and **pdp** (Greenwell 2017); the package **ICEbox** (Goldstein et al. 2015) provides ICE curves and variations. Each package provides options for selecting the size of the grid for evaluating the predictions, with **flashlight** and **DALEX** providing options to select the number of observations to consider in calculating the partial dependence.

In **vivid** we provide a suite of functions (see Table 2) for calculating and visualizing variable importance, interactions and the partial dependence. Our displays conveniently show (either model specific and agnostic) VImp and VInt jointly using heatmaps and network graphs. Through the use of seriation techniques, we group together variables with the greatest impact on the response. For our network plot, additional filtering to remove less influential variables and clustering options to group highly interacting variables are provided, thus providing a more informative picture identifying relevant features. Additionally, our implementation makes it possible to apply our methods to subsets of data, which leads to locally-based importance measures. Our generalized PDP (GPDP) displays partial dependence plots in a matrix layout combining univariate and bivariate partial dependence plots with variable scatterplots. This layout, coupled with seriation, allows for quick assessment of of how pairs of variables have an impact on the model fit. We furthermore provide a more compact version of the GDPD, the so-called zen-partial dependence plot (ZPDP) consisting only of those bivariate partial dependence plots with high VInt. All of our displays are designed to quickly identify how variables, both singly and jointly, affect the fitted response and can be used for regression or classification fits. As the output of our displays are **ggplot2** objects (Wickham 2016), they are easily customizable and provide the flexibility to create custom VIVI visualizations.

This paper is structured as follows. First we introduce a dataset and fit models that will be used as examples throughout this paper. Following this, we describe **vivid** functionality for calculating VIVI. We then move on to visualizations and focus on the functionality provided by the two functions `viviHeatmap` and `viviNetwork` for displaying VIVI, and two functions for displaying PDPs namely, `pdpPairs` and `pdpZen`. Finally we provide some concluding discussion.

2 Example: Data and Models

The well-known Boston housing data (Harrison Jr and Rubinfeld 1978) from the R package **MASS** (Venables and Ripley 2002) concerns prices of 506 houses and 14 predictor variables including property attributes such as number of rooms and social attributes including crime rate and pollution levels. The

Table 2: Summary of functions available in the vivid package. The main construction function is vivi which is used to calculate the VIVI values for subsequent use in the visualizations.

Function	Description	Type
vivi	Create a VIVI matrix of class vivid	VIVI construction
viviReorder	Reorders a square matrix so high VIVI values are pushed to the top left of the matrix	VIVI construction
viviHeatmap	Heatmap plot of VIVI values	Visualization
viviNetwork	Network plot of VIVI values	Visualization
pdpVars	Univariate partial dependence plot with ICE curves displayed as a grid	Visualization
pdpPairs	Pairs plot showing bivariate PDP, ICE/univariate PDP, and data	Visualization
pdpZen	A zigzag expanded navigation plot (zenplot) displaying partial dependence values	Visualization
CVpredictfun	Predict function	Utility
zPath	Constructs a zenpath for connecting and displaying pairs to be used with pdpZen	Utility
as.data.frame.vivid	Takes a matrix of class vivid and turns it into a data frame	Utility
vip2vivid	Takes measured importance and interactions from the vip package and turns them into vivid matrix which can be used for plotting	Utility

response is the median value of owner-occupied homes in $1000s (medv).

We first fit a random forest (using the **randomForest** package). In order to access all the available embedded variable importance scores, the importance argument must be TRUE when calling the randomForest function. This allows any of the provided importance metrics to be used in **vivid**. However, in our following examples, we use an agnostic VImp measure supplied by the **vivid** package, which allows us to directly compare VImp values across different fits.

```
library("randomForest")
library("MASS")
set.seed(1701)
data("Boston")

rf <- randomForest(medv ~., data = Boston)
```

Next we fit a gradient boosted machine (using the **xgboost** package). For the GBM we set the maximum number of boosting iterations, nrounds, to 100 as no default is provided in **xgboost**.

```
library("xgboost")
gbst <- xgboost(data = as.matrix(Boston[,1:13]),
                label =  as.matrix(Boston[,14]),
                nrounds = 100,
                verbose = 0)
```

In the following sections we will explain how aspects of the two fits can be compared with **vivid** software. We will also explain aspects of our software design with reference to these fits.

3   Calculating VIVI

The first step in using **vivid** is to calculate variable importance and interactions for a model fit. The vivi function calculates both of these, creating a square, symmetric matrix containing variable importance on the diagonal and variable interactions on the off-diagonal. Required inputs are a fitted ML model, a data frame on which the model was trained, and the name of the response variable for the fit. The returned matrix has importance and interaction values for all variables in the supplied data frame, excluding the response. Our visualizations functions viviHeatmap and viviNetwork are designed to show the results of a vivi calculation, but will work equally well for any square matrix with identical row and column names. (Note, the symmetry assumption is not required for viviHeatmap and viviNetwork uses interaction values from the lower-triangular part of the matrix only.)

The code snippet below shows the creation of a vivid matrix for the random forest fit. For clarity, we include all of the vivi function arguments for the random forest fit, though only the first three are required. Other inputs will be described in the section vivi function additional arguments.

```
library("vivid")

set.seed(1701)
viviRf <- vivi(fit = rf,
               data = Boston,
               response = "medv",
               reorder = FALSE,
               normalized = FALSE,
               importanceType = "agnostic",
               gridSize = 50,
               nmax = 500,
               class = 1,
               predictFun = NULL,
               numPerm = 4)
```

In the absence of any model-specific importance measure we use an agnostic permutation method described by Fisher, Rudin, and Dominici (2019) to obtain the variable importance scores. In this method a model error score (root mean square error) is calculated, then each feature is randomly permuted and the model error is re-calculated. The difference in performance is considered to be the variable importance score for that feature. By default the permutation is set to be replicated four times to account for variability. However, we provide an option to select a desired number of permutations (via the numPerm argument)).

The vivi function calculates both the importance and interactions using S3 methods. By default, the agnostic importance and interaction scores in **vivid** are computed using the generalized predict function from the **condvis2** package (C. Hurley, OConnell, and Domijan 2022). Consequently, **vivid** can be used out-of-the-box with any model type that works with **condvis2** predict (see CVpredict from **condvis2** for a full list of compatible model types). To allow **vivid** to run with other model fits, a custom predict function must be passed to the predictFun argument (as discussed below).

The S3 method used to obtain the importance is called vividImportance. vivid relies on the **flashlight** package to calculate agnostic importance via flashlight::light_importance which currently works for numeric and numeric binary responses only. For model-specific variable importance, we provide individual methods to access importance scores for some of the most popular model fitting R packages, namely; **ranger** (Wright and Ziegler 2017), **randomForest**, **mlr** (Bischl et al. 2016), **mlr3** (Lang et al. 2019), and **parsnip** (Kuhn and Vaughan 2022) (however, more could be added via additional methods). To access any available model-specific variable importance from the aforementioned packages, the importanceType argument must be set to equal the selected importance metric. For example, to select the percent increase in mean square error importance score from the **randomForest** package, the importanceType argument must be set to equal this measure as it is called within the **randomForest** package, that is; importanceType = "%IncMSE". If the importanceType is not set or is set to equal agnostic, then the agnostic importance is calculated. It should be noted that when comparing different model fits, using model-specific variable importance will result in importance measures that are not directly comparable (however, comparing model specific scores could be useful when comparing the same fitting procedure evaluated using different parameters). In the case that a practitioner may wish to use our visualizations to compare different model fits, we recommend using the agnostic permutation approach supplied by **vivid** to make a direct comparison of the importance measures.

For variable interactions, we use the model-agnostic Friedman's H-statistic to identify any pairwise interactions. As discussed in Inglis, Parnell, and Hurley (2022a), we recommend the unnormalized version of the H-statistic which prevents detection of spurious interactions which can occur when the bivariate partial dependence function (used in the construction of the H-statistic) is flat. In the case of a binary response classification model, we follow Hastie, Tibshirani, and Friedman (2009) and compute the H-statistic and partial dependence on the logit scale.

The vivi function calculates interactions using an S3 method called vividInteraction, which again relies on the **flashlight** package to calculate Friedman's H-statistic via flashlight::light_interaction. Friedman's H-statistic is the only interaction measure currently available in **vivid**, though the method of Greenwell, Boehmke, and McCarthy (2018) could also be used for this purpose. Embedded interaction measures could easily be incorporated via S3 methods in future.

flashlight simplifies the calculation of VIVI values as it allows a custom predict function to be supplied for the calculation of agnostic importance and the H-statistic; this flexibility means

importance and the *H*-statistic can be calculated for any ML model. Note that as **flashlight** importance and interaction functions act in a model agnostic way, they will give a VIVI of zero for variables in the dataset (except the response) that are not used by the supplied ML fit. We supply an internal custom predict function called `CVpredictfun` to both `flashlight::light_importance` and `flashlight::light_interaction`. `CVpredictfun` is a wrapper around `CVpredict` from the **condvis2** package, which adds an option for the classification to select (via the `class` argument to `vivi`) the class to be used for prediction and calculates predictions on the logit scale by default. `CVpredict` accepts a broad range of fit classes thus streamlining the process of calculating VIVI.

In situations where the fit class is not handled by `CVpredict` (as is the case for the GBM model created from **xgboost**), supplying a custom predict function to the `vivi` function by way of the `predictFun` argument allows the agnostic VIVI values to be calculated. In the code snippet below, we build the `vivid` matrix for the GBM fit using a custom predict function, which must be of the form given in the code snippet. For brevity we omit some of the optional `vivi` function arguments. By default, the agnostic variable importance is used to allow for a direct comparison of the importance measures for both model fits.

```
# predict function for GBM
pFun <- function(fit, data, ...) predict(fit, as.matrix(data[,1:13]))

set.seed(1701)
viviGBst <- vivi(fit = gbst,
                 data = Boston,
                 response = "medv",
                 reorder = FALSE,
                 normalized = FALSE,
                 predictFun = pFun)
```

### 3.1 `vivi` function additional arguments

The `vivi` function has 11 arguments. Some of these have been discussed above, including `fit`, `data`, `response`, `importanceType`, and `predictFun`. Here we provide a summary of the remaining arguments. First, the `normalized` argument determines if Friedman's *H*-statistic should be normalized or not (see Inglis, Parnell, and Hurley (2022a), for the pros and cons of each version). The arguments `gridSize` and `nmax` are used to set the size of the grid for evaluating the predictions and maximum number of data rows to consider, respectively, in the calculation of the *H*-statistic. Lowering the grid size can provide a significant speed boost, though at the expense of predictive accuracy. Additionally, sampling the data via `nmax` offers a speed boost. The default values for `gridSize` and `nmax` are 50 and 500, respectively.

3.2 Speed tests

A drawback of using Friedman's *H*-statistic as a measure of interaction is that it is a computationally expensive calculation, and may be especially time-consuming for models where prediction is slow. Figure 1 shows the build time (rounded to the nearest second) averaged over five runs for the creation of a `vivid` matrix with default parameters for different ML algorithms using the Boston Housing data. As the Boston housing data has 13 predictor variables, Friedman's *H*-statistic is computed for 91 predictor pairs. The ML algorithms are: GBM, random forest, support vector machine (SVM), neural network (NN), and k-nearest neighbors (KNN). The SVM, NN, and KNN were built using the **e1071** (Meyer et al. 2021), **nnet** (Venables and Ripley 2002), and **kknn** (Schliep and Hechenbichler 2016) packages, with the KNN being built through the **mlr3** (Lang et al. 2019) framework. Each of the models were built using their default settings and, for each model fit, the agnostic VIVI was measured. The speed tests were performed on both a 2017 Mac-book Pro 2.3 GHz Dual-Core Intel Core i5 with 8GB of RAM and a 2021 32GB Mac-book M1 Pro. Here we are essentially comparing predict times for the various fits. The NN fit created using the **nnet** package was the fastest, followed by GBM. Both random forests are the slowest, and surprisingly, the older Mac beats its higher spec cousin for the **randomForest** fit.

3.3 Alternative construction of a `vivid` matrix

A `vivid` matrix may also be obtained from variable importance and interaction values calculated elsewhere. The package **vip** offers these, and evaluates interactions using a method called the *feature importance ranking measure* (FIRM, see Greenwell, Boehmke, and McCarthy (2018), for more details).
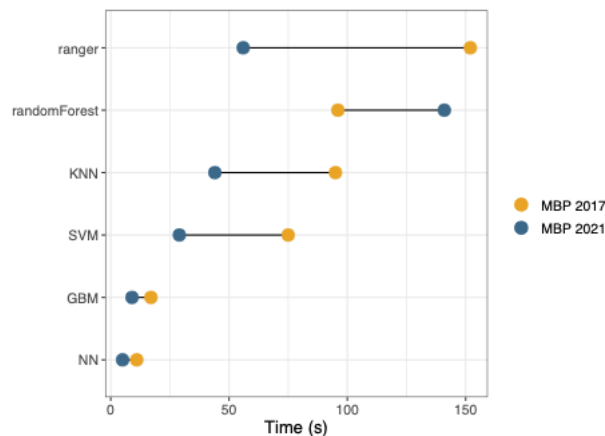
Figure 1: Mean time over five runs, on two MacBooks, for the creation of a vivid matrix for different models. Times are highly dependent on the model fit, with NN the fastest and random forests the slowest.

The `vip2vivid` function we provide in **vivid** takes VIVI values created in **vip** and turns them into a vivid matrix, that can be subsequently used with our plotting tools. For example, in the code below, model-specific VImp and FIRM VInt scores are calculated for the random forest fit, and subsequently arranged into a vivid matrix with the VImps on the diagonal and VInts on the off-diagonal.

```
library("vip")
# get model specific VImps using vip package
vipVImp <- vi(rf, method = 'model')
# get VInts using vip package
vipVInt <- vint(rf, feature_names = names(Boston[-14]))

# turn into vivi-matrix
vipViviMat <- vip2vivid(importance = vipVImp, interaction = vipVInt)
```

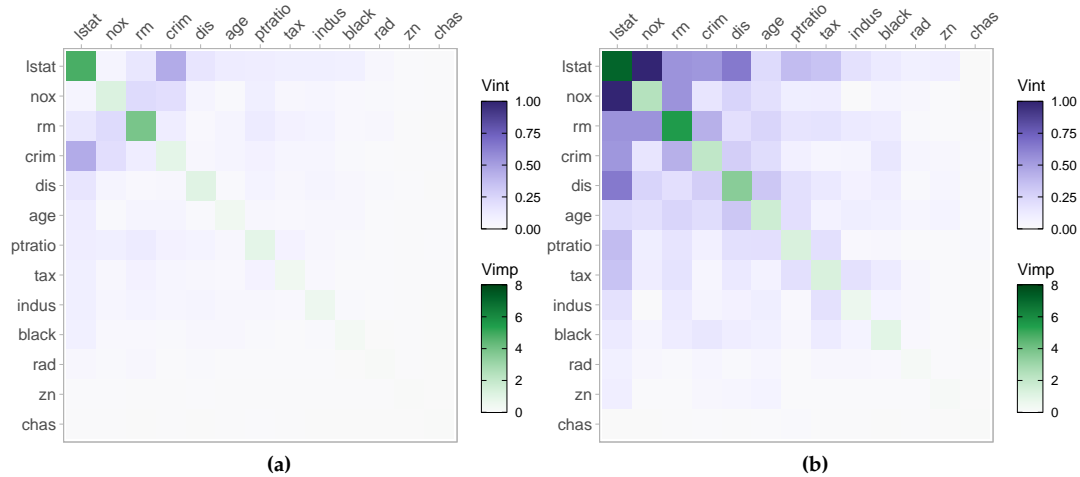## 4 Heatmap of Variable Importance and Variable Interactions

The `viviHeatmap` function constructs a heatmap displaying both importance and interactions, with importance on the diagonal and interactions on the off-diagonals. A vivid matrix is the only required input, which does not necessarily need to be symmetric (for example, the interaction measures from the **randomForestExplainer** package are asymmetric and could be visualized using our heatmap. Color palettes for the importance and interactions are optionally provided via `impPal` and `intPal` arguments. For the default color palette we choose single-hue, color-blind friendly sequential color palettes from Zeileis et al. (2020), where low and high VIVI values are represented by low and high luminance color values respectively, aiding in highlighting values of interest.

The ordering of the heatmap is taken from the ordering of the input matrix. As `reorder` was set to `FALSE` when building both the random forest and GBM fit vivid matrix, the ordering of the heatmaps matches the variable order in the dataset. This is useful for directly comparing multiple heatmaps, however it does not necessarily lend itself for easy identification of the largest VIVI values. If we were to seriate both vivi matrices separately, we would end up with different optimal orderings for each matrix. An alternative is to create a common ordering by averaging over the two vivid matrix objects and applying the `vividReorder` function to the result. (This function uses a seriation algorithm based on the techniques of Earle and Hurley (2015) designed to place high interaction variables adjacently and to pull high VIVI variables towards the top-left; see Inglis, Parnell, and Hurley (2022a) for details.) Both VIVI matrices are then re-ordered using the newly obtained variable order.

```
# average over matrices and seriate to get common ordering
viviAvg <- (viviRf + viviGBst) / 2
viviAvgReorder <- vividReorder(viviAvg)

# reorder vivi-matrices
ord <- colnames(viviAvgReorder)
viviRf <- viviRf[ord,ord]
```

Figure 2: Agnostic variable importance and variable interaction scores for a random forest fit in (a) and GBM fit in (b) on the Boston housing data displayed as a heatmap. The random forest fit has weaker interactions and lower importance scores than the GBM fit. Both fits identify *lstat* as the most important followed by *rm*. In both fits, *lstat* has numerous interactions with other variables, notably *crim* in the random forest fit and *nox* in the GBM fit.

```
viviGBst <- viviGBst[ord,ord]
```

Arguments `impLims` and `intLims` specify the range of importance and interaction values to be mapped to colors. Default values are calculated from the maximum and minimum VIVI values in the `vivid` matrix. Importance and interaction values falling outside the supplied limits are squished to the closest limit. It can be useful to specify these limits in the situation where there is an extremely large VIVI value that dominates the display, or where we wish two or more plots to have the same limits for comparison purposes, as in the example below. The `angle` argument is used to rotate the x-axis labels.

Figure 2 shows our improved ordering so that variables with high VIVI values are pushed to the top left of the plots. Filtering can also be applied to the input matrix to display a subset of variables. When compared to the GBM fit in (b), the random forest fit in (a) has weaker interactions and lower importance scores. Both plots identify *lstat* as being the most important. Both fits also show that *lstat* (the percentage of lower status of the population) interacts with several other variables, though the interactions are much stronger for the GBM. Notably, the strongest interaction in both fits are different. These are *lstat* : *crim* (where *crim* is the per capita crime rate by town) for the random forest fit and *lstat* : *nox* (where *nox* is parts per 10 million nitrogen oxides concentration) for the GBM fit.

```
viviHeatmap(viviRf, angle = 45, intLims = c(0,1), impLims = c(0,8))
viviHeatmap(viviGBst, angle = 45, intLims = c(0,1), impLims = c(0,8))
```

5 Network of Variable Importance and Variable Interactions

The `viviNetwork` function constructs a network graph displaying both importance and interactions. Similarly to `viviHeatmap`, this function takes a `vivid` matrix as the only required input and provides a visual representation of the magnitude of the importance and interaction values through the size of the nodes and edges in the graph, in addition to color. In the plot each variable is represented as a node, with its importance being represented through size and color such that larger, darker nodes indicate a higher importance. Each pairwise interaction is represented by a connecting edge, where larger interaction values get thicker, darker edges; Figure 3 provides an example. This type of plot enables the user to quickly identify the magnitude of the importance and interactions of the variables that have the most impact on the response. The `viviNetwork` function optional arguments follows the same conventions as `viviHeatmap`: custom color palettes for importance and interactions are provided via the `impPal` and `intPal`, and the range of VIVI values to be mapped to the colors are specified via the `impLims` and `intLims`.

By default, we choose a circular layout to display the graphs, as when coupled with the seriated `vivid` matrix, variables with high VIVI are grouped in a clock-wise arrangement starting at the top. This arrangement allows for easy identification of variables with high VIVI. Custom layouts are

possible by providing a numeric matrix with two columns and one row per node to the layout argument. Additionally, any of the layouts available in the **igraph** package (Csardi and Nepusz 2006) can be specified. The subject of network graph layouts has been extensively studied (for examples see Purchase (1997), Herman, Melançon, and Marshall (2000), Freeman (2000)). It has been shown that certain layouts of network graphs can significantly aid in cognitive interpretation. For example, Purchase (1997) note that reducing the number of edge crossovers is by far the most important aesthetic (even for small amounts of data), while maximizing symmetry has a lesser effect. Several of the layouts provided by the **igraph** package can aid interpretation (such as the Sugiyama layout algorithm (Sugiyama, Tagawa, and Toda 1981), which tries to minimize edge crossover.)

We provide options to filter the graph via the `intThreshold` and `removeNode` arguments. This helps to highlight variables with high VIVI scores, which is useful in settings with many predictors. The `intThreshold` argument filters edges with weight (i.e., VInt value) below a specified value and `removeNode` removes nodes with no connecting edges after thresholding interaction values. We can optionally cluster similar variables together with respect to their VIVI scores via the `cluster` argument, thereby aiding in the process of highlighting variables of interest. The `cluster` argument can take either a vector of cluster memberships for nodes or an appropriate **igraph** clustering function.

We demonstrate network plots displaying VIVI values for the GBM fit. In Figure 3, we show both a default network plot including all variables in (a) and a filtered and clustered network plot in (b). For the filtered plot we select VIVI values in the top decile. This selection allows us to focus only on the variables with the most impact on the response. The variables that remain are *lstat*, *nox*, *rm*, *crim*, *dis* (weighted mean of distances to five Boston employment centers), *tax* (full-value property-tax rate per $10,000), and *ptratio* (pupil-teacher ratio by town). We then perform a hierarchical clustering treating variable interactions as similarities, with the goal of grouping together high-interaction variables. Here we manually select the number of groups we want to show via the `cutree` function (which cuts clustered data into a desired number of groups). Finally we rearrange the layout using **igraph**. Here, `igraph::layout_as_star` places the first variable (deemed most relevant using the VIVI seriation process above) at the center, which in Figure 3 (b) emphasizes its key role as the most important predictor which also has the strongest interactions.

```
# default network plot for GBM
viviNetwork(viviGBst)

# clustered and filtered network for GBM
intVals <- viviGBst
diag(intVals) <- NA

# select VIVI values in top 10%
impTresh <- quantile(diag(viviGBst),.9)
intThresh <- quantile(intVals,.9,na.rm=TRUE)
sv <- which(diag(viviGBst) > impTresh |
                apply(intVals, 1, max, na.rm=TRUE) > intThresh)

h <- hclust(-as.dist(viviGBst[sv,sv]), method = "single")

viviNetwork(viviGBst[sv,sv],
            intLims = c(0,1),
            impLims = c(0,8),
            cluster = cutree(h, k = 3), # specify number of groups
            layout = igraph::layout_as_star)
```

In Figure 3(a), when displaying all the variables, we can clearly identify which variables have the highest VIVI values. The large darker nodes of *lstat* and *rm* indicate their importance and the dark, thick connecting edge between *lstat* and *nox* tell us that these two variables strongly interact. In (b), after applying a hierarchical clustering, we can see the strongest mutual interactions have been grouped together for the GBM fit. Namely, *lstat*, *nox*, *crim*, *rm*, and *dis* are all grouped together. The remaining variables are individually clustered.

We provide a conversion of `vivid` matrix objects to a data frame via an `as.data.frame` method, as demonstrated below. This facilitates plotting with base R and **ggplot2**, for example a barplot of either VImp or VInt values. Note that while `vivi` returns a matrix of class `vivid`, the class attribute was dropped when the matrix was re-ordered.

```
class(viviRf)<- c("vivid", class(viviRf))
head(as.data.frame(viviRf), 4)
```
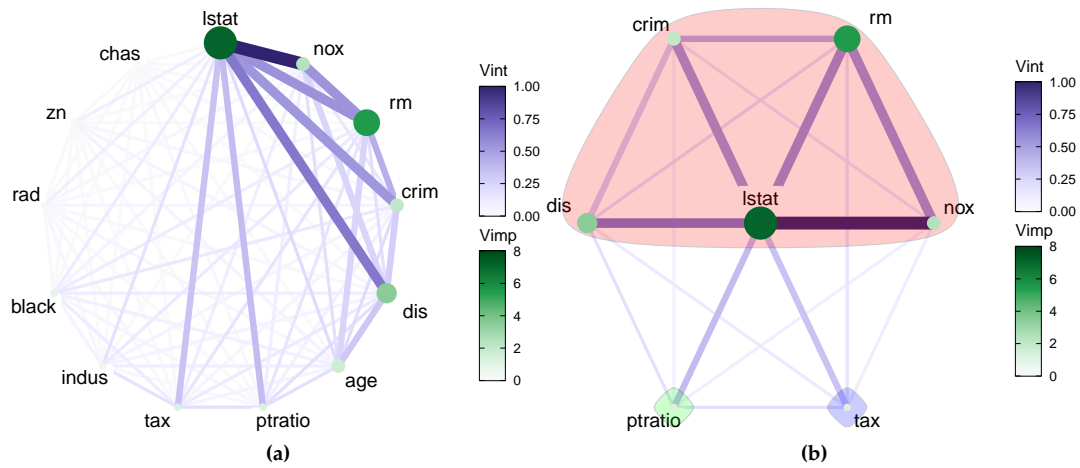
Figure 3: Network plots showing VIVI scores obtained from a GBM fit on the Boston housing data. In (a) we display the all values in a circle. In (b) we use a hierarchical clustering to group variable with high VIVI together and rearrange the layout using an igraph function.
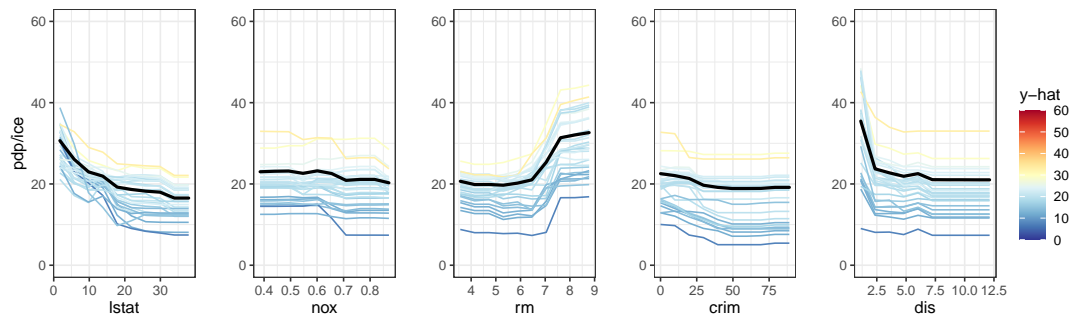
Figure 4: Partial dependence plots (black line) with individual conditional expectation curves (colored lines) of a GBM fit on the Boston housing data. The changing partial dependence and ICE curves of *lstat* and *rm* indicate that these variables have some impact on the response.

```
#>   Variable_1 Variable_2     Value Measure Row Col
#> 1      lstat      lstat 4.80970237    Vimp   1   1
#> 2        nox      lstat 0.06387693    Vint   2   1
#> 3         rm      lstat 0.15226649    Vint   3   1
#> 4       crim      lstat 0.44728494    Vint   4   1
```

6 Partial Dependence and Individual Conditional Expectation Curves

6.1 Univariate Partial Dependence Plot

The `pdpVars` function constructs a grid of univariate PDPs with ICE curves for selected variables. We use ICE curves to assist in the identification of linear or non-linear effects. The fit, data frame used to train the model, and the name of the response variable are required inputs. In the code below, we show an example of the partial dependence and ICE curves for the first five features from the GBM vivid matrix, with output shown in Figure 4. We use the custom GBM predict function given previously.

```
top5 <- colnames(viviGBst)[1:5]
pdpVars(data = Boston,
        fit = gbst,
        response = "medv",
        vars = top5,
        predictFun = pFun)
```

All of our PDP variants handle categorical responses and predictors. The color palette is customized via the pal argument. In all of our PDPs, this defaults to a diverging palette which accentuates

fitted values that differ from the average. Dark red and dark blue are used to indicate high and low values of $\hat{y}$ respectively. The middle values are displayed in yellow. The `nIce` argument specifies the number of ICE curves to be drawn. This is either a single number specifying the number of observations to be sampled for the ICE curves, or a vector of row indices, an option that is useful for example to display ICE curves from particular classes. The default value for `nIce` is 30, which allows individual curves to be seen.

The ordering of the PDPs is taken from the ordering of variables in the data set, or may be specified via the `vars` argument. In Figure 4, the ordering is taken directly from our seriated `vivid` matrix, thereby showing the top five most influential variables. As with the construction of the `vivid` matrix, the `gridSize` and `nmax` arguments determine the number of predictions.

In Figure 4 we can see from the changing PDP and ICE curves that *lstat* and *rm* have the clearest impact on the response, with the predicted median house price being higher for low values of *lstat* and high values of *rm*. Additionally, the predicted median house price appears to be higher for low values of *dis* before leveling off at around 2.5. The remaining variables have generally flat partial dependence and ICE curves.

6.2 Generalized Pairs Partial Dependence Plot

The `pdpPairs` function creates a generalized pairs partial dependence plot (GPDP). In our GPDP, we use a matrix layout and plot the univariate partial dependence (with ICE curves) on the diagonal, bivariate partial dependence on the upper diagonal and a scatterplot of raw variable values on the lower diagonal, where all colors are assigned to points and ICE curves by the predicted $\hat{y}$ value. In the case of categorical predictors, the partial dependence for each factor level is shown in the upper-diagonal (for an example of this, see Inglis, Parnell, and Hurley (2022a)). As with the univariate PDP, the fit, data frame used to train the model, and the name of the response variable are required inputs.

```
set.seed(1701)
pdpPairs(data = Boston,
         fit = gbst,
         response = "medv",
         gridSize = 20,
         nIce = 50,
         vars = top5,
         convexHull = TRUE,
         fitlims = "pdp",
         predictFun = pFun)
```

As with the univariate PDP, the ordering can be controlled via the `var` argument. By default, the ordering is taken from the order of the data. In the code above, we display only the interesting variables seen in previous plots by selecting the first five variables from our seriated `vivid` matrix. We also chose to display 50 ICE curves. As with `pdpVars`, additional arguments specify the color palette and number of ice curves, while `gridSize` and `nmax` determine the number of predictions.

For our GPDP, we follow the general design choices in **vivid** and specify the range of predicted values to be mapped to the colors via the `fitlims` argument. We set the default fit range for the color map for the GPDP to the range of the collection of PDP surfaces with `fitlims = "pdp"`. The setting of this argument at its default value allows for maximum resolution of the bivariate PDPs. Since predictions for specific observations and ICE curves would likely exceed these bounds, the closest value within the color map's bounds is used to allocate colors. Alternatively `fitlims = 'all"` specifies that limits are calculated as the full range of predictions shown.

In the upper diagonals we exclude extrapolated areas from the bivariate PDPs to prevent interpretation of the PDPs in areas where there is no data. The removal of extrapolated areas can be prevented by specifying `convexHull = FALSE`.

In Figure 5, in addition to the univariate PDPs, we capture the effects of the variables on the response via the bivariate PDP on the upper-diagonal and the distribution of the data in the lower-diagonal. The scatterplots are useful for determining if variables are highly correlated, as highly correlated variables may spuriously affect the partial dependence and give erroneous results (Apley and Zhu 2020). Of note are the variables *lstat* and *rm*. We can clearly see that when the number of rooms (*rm*) is high and the percentage of lower status of the population (*lstat*) is low, the predicted $\hat{y}$ median house price value is high. This is exemplified in the changing bivariate PDP.
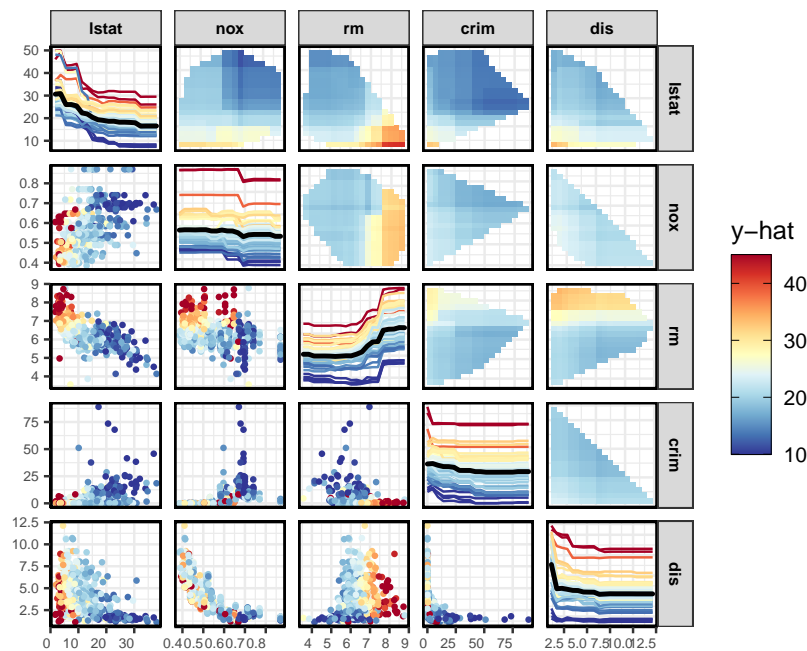
Figure 5: Filtered generalized pairs partial dependence plot for a GBM fit on the Boston housing data. From both the univariate and bivariate PDPs, we can see that *lstat* and *rm* have an impact on the response. As *lstat* decreases and *rm* increases, predicted median house price value goes up. The bivariate PDP of *lstat* : *nox* shows that as *nox* increases, the predicted value decreases.

7   Zen Partial Dependence Plots

The `pdpZen` function creates partial dependence plots utilizing a space-saving method based on graph Eulerians (Hierholzer and Wiener 1873). An Eulerian path, also known as an Eulerian trail, is a route that traverses each edge of a graph exactly once. When this path forms a closed loop, the traversal is referred to as an Eulerian tour. We call this display zen-partial dependence plots (ZPDP). The display is based on the zigzag expanded navigation plots, known as zenplots , available in the **zenplots** package (Hofert and Oldford 2020). Zenplots were created to display paired graphs of high-dimensional data focusing on the most important 2D displays. In our adaptation we show bivariate PDPs that focus on the variables with the largest interaction values in a compact zigzag layout, which is helpful when predictor space is high-dimensional.

The code below illustrates `pdpZen`, here displaying the first five variables from GBM's `vivid` matrix. Later we show an example focusing on high-interacting pairs of variables. We use the same convention as our previous PDPs with regard to color palette and limits, grid size, and the number of rows considered for evaluation. The ZPDP also has a variable rug plot on each axis to avoid interpretation problems that may occur in the presence of skewness.

```
pdpZen(data = Boston,
       fit = gbst,
       response = "medv",
       convexHull = TRUE,
       zpath = top5,
       predictFun = pFun)
```

The argument zpath specifies the variables to be plotted, defaulting to all dataset variables aside from the response. In the code above, zpath is the vector *lstat*, *nox*, *rm*, *crim* and *dis*. The resulting plot shown in Figure 6 presents the bivariate PDP for every consecutive pair of variables in a zigzag layout.

7.1 Zen-paths

ZPDP are most useful when the bivariate PDPs plotted are selected to be an interesting subset of all pairwise plots. To obtain this subset, we consider a network graph displaying VIVI values, such as that in Figure 3 (a). We then filter the edges below a selected interaction value, leaving only highly
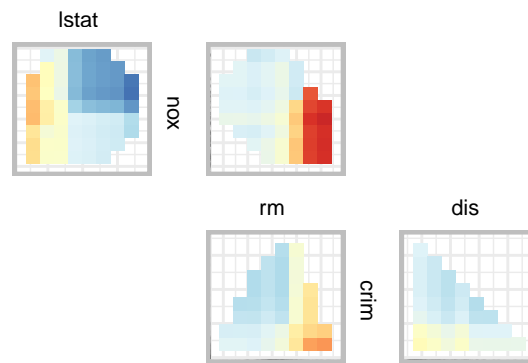
Figure 6: Zen partial dependence plot for the GBM fit on the Boston data. Here we display first five variables from the GBM's 'vivid' matrix. Only plots for consecutive variables are shown.

interacting variable pairs, as in Figure 3(b). Our goal is to then build a ZPDP consisting of the bivariate plots represented by each edge of the thresholded graph. The zPath function creates a sequence or sequences of variable paths for use in pdpZen.

The zPath function takes four arguments. These are: viv - a matrix of interaction values, cutoff - exclude interaction values below this threshold, method - a string indicating which method to use to create the path, and connect - a logical value indicating if separate Eulerians should be connected

Two methods are provided, either "greedy.weighted" or "strictly.weighted". The first option uses the greedy Eulerian path algorithm (C. B. Hurley and Oldford 2011, 2022) for connected graphs. This visits each edge at least once, beginning at the edge with the highest weight and traversing through the remaining edges, giving priority to the highest-weighted edge. Some edges may be visited more than once or additional edges may be visited if the number of nodes in the graph is not even. The second method "strictly.weighted" (provided by **zenplot**) visits edges strictly in decreasing order by weight (here the interaction values). If connect is TRUE the sequences obtained by the strictly weighted method are concatenated to form a single path.

In the code below, we provide two examples of creating zen-paths, from the top 10% of interaction scores in viviGBst.

```
intThresh <- quantile(intVals, .9, na.rm=TRUE)
# set zpaths with different parameters
zpGw  <- zPath(viv = viviGBst, cutoff = intThresh, method = "greedy.weighted")
zpGw

#> [1] "nox"     "lstat"   "dis"     "ptratio" "lstat"   "rm"      "crim"
#> [8] "lstat"   "tax"     "rm"      "nox"

zpSw <- zPath(viv = viviGBst, cutoff = intThresh, connect = FALSE, method = "strictly.weighted")
zpSw

#> [[1]]
#> [1] "nox"   "lstat" "dis"
#>
#> [[2]]
#> [1] "lstat" "rm"    "nox"
#>
#> [[3]]
#> [1] "lstat" "crim"  "rm"
#>
#> [[4]]
#> [1] "ptratio" "lstat"   "tax"
```
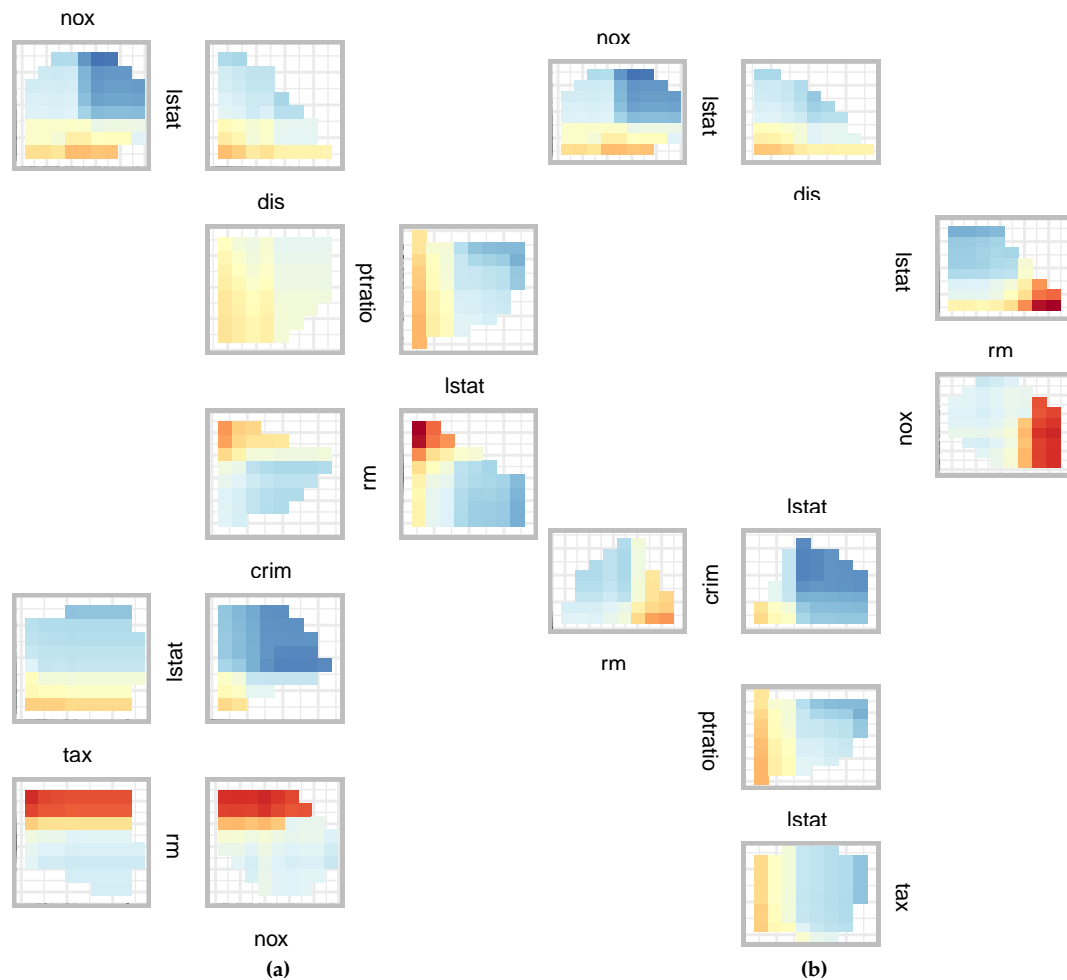
Our first created zen-path object, zpGw, uses the greedy.weighted method and visits each edge exactly once. The second zen-path, zpSw, uses the strictly.weighted method with connect = FALSE. zpSw consists of four unconnected paths. The zenplots for two of these paths are constructed below.

```
pdpZen(data = Boston,
       fit = gbst,
```

Figure 7: ZPDP for a GBM fit on the Boston data. In (a) the zpath is defined by the 'greedy.weighted' sorting method. In (b), the sorting method is defined by the 'strictly.weighted' method and is unconnected. For low values of *lstat* and and high values of *rm*, predicted median house price value increases.

```
        response = "medv",
        zpath = zpGw,
        convexHull = TRUE,
        predictFun = pFun)

pdpZen(data = Boston,
       fit = gbst,
       response = "medv",
       zpath = zpSw,
       convexHull = TRUE,
       predictFun = pFun)
```

Note that there are 7 different variables involved in high interactions, which could be displayed in a 7 × 7 GPDP, showing a total of 21 bivariate PDPs. But only 8 of these have VInt values above the 90% quantile, and Figure 7(b) using the `strictly.weighted` path shows just these bivariate PDPs' compact layout. Using the `greedy.weighted` sorting method in (a) produces a smaller, neater plot but at the expense of including some plots that are not particularly interesting (for example the pair *dis* : *ptratio*). However, it should be noted that any of the arguments from the `zenplot` function from the **zenplots** package can be used with the pdpZen function. These include multiple options for different or custom layouts.

8 Summary

We have presented a detailed exposition of our R package vivid which contains a suite of integrated functions implementing algorithms and novel visualizations for exploring variable importance and variable interactions in machine learning models. Our techniques are intuitive, adaptable, easy to customize and facilitate model comparison. When building the vivid matrix to use in our heatmap and network visualizations, VIVI metrics that are model specific or model-agnostic may be employed. For measuring interactions we currently only provide the option to use the agnostic Friedman's H-statistic. However, as outlined in the Calculating VIVI section, the inclusion of different VIVI measures is easily possible.

Our vivid package is a useful addition to the other packages in the area of model visualization, such as those discussed in the Introduction section. Our heatmap and network plots efficiently determine which variables have the greatest impact on the response. When coupled with the seriation, filtering, and clustering techniques, these visualizations enhance the interpretation of ML predictions. Our GPDP and ZPDP can be used to provide a thorough examination of the behavior of a fitted ML model by examining the individual variable effects and their pairwise interactions. These plots combine the bivariate PDP, ICE curves, and scatterplots of the raw variable values. They further allow focusing on subsets of variables with high VInt, and so allow us to efficiently explore a fitted ML model by focusing attention to only the most crucial aspects.

It has been noted that the presence of correlated variables can lead to biased VIVI measures. For example, Friedman and Popescu (2008) note that when subsets of variables are highly correlated, it becomes difficult to distinguish between low and higher order interactions among them in a straightforward manner when using the H-statistic. Similarly, Fisher, Rudin, and Dominici (2019) state that for permutation variable importance, the presence of highly correlated variables can affect the measured variable importance and propose using a method of conditional importance, such as that implemented for conditional random forests in the party package. Our GPDP can be used to identify any evident correlations by observing the plot of the data and the PDP with convex hull. However, careful consideration should be employed when interpreting the VIVI measures from vivid and we recommend evaluating any potential correlation between variables in conjunction with our proposed visualizations. Several R packages are available for assessing and visualising correlations, such as corrplot (Wei and Simko 2021) and corrgrapher (Morgen and Biecek 2020).

For future work, the inclusion of other model summaries could be incorporated into vivid, such as the interaction statistics described in Greenwell, Boehmke, and McCarthy (2018) or the use of accumulated local effects (ALE) of Apley and Zhu (2020). This latter method was created to address bias problems with partial dependency functions and could be used in place of the bivariate PDPs seen in both the GPDP and ZPDP. However the calculation of an agnostic, easily interpretable variable interaction measure that accounts for correlated variables remains an ongoing research goal. Allowing a user to view the variation across replicates when performing permutation importance could also be incorporated. This could be added into the heatmap graphic by way of a value suppressing uncertainty palette (Correll, Moritz, and Heer 2018), where the uncertainty in included in the visualization (for example, see Inglis, Parnell, and Hurley (2022b)). Additionally, providing alternative metrics such as AUC or accuracy for computation of permutation importance in classification, as well as computations and visualizations of higher order interactions, interactive graphics and incorporating multivariate response variables are interesting areas for future research.

References

Antunes, Nuno, Leandro Balby, Flavio Figueiredo, Nuno Lourenco, Wagner Meira, and Walter Santos. 2018. "Fairness and Transparency of Machine Learning for Trustworthy Cloud Services." In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-w)*, 188–93. IEEE.

Apley, Daniel W, and Jingyu Zhu. 2020. "Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 82 (4): 1059–86.

Biecek, Przemyslaw. 2018. "DALEX: Explainers for Complex Predictive Models in R." *Journal of Machine Learning Research* 19 (84): 1–5. https://jmlr.org/papers/v19/18-416.html.

Biecek, Przemyslaw, and Hubert Baniecki. 2023. *Ingredients: Effects and Importances of Model Ingredients*. https://CRAN.R-project.org/package=ingredients.

Biecek, Przemyslaw, and Tomasz Burzykowski. 2021. *Explanatory Model Analysis: Explore, Explain and Examine Predictive Models*. Chapman; Hall/CRC.

Bischl, Bernd, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, and Zachary M. Jones. 2016. "mlr: Machine Learning in R." *Journal of Machine Learning*

Research 17 (170): 1–5. https://jmlr.org/papers/v17/15-066.html.

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32. https://doi.org/10.1023/A:1010933404324.

Chen, Xi, and Hemant Ishwaran. 2012. "Random Forests for Genomic Data Analysis." *Genomics* 99 (6): 323–29.

Chipman, Hugh A, Edward I George, and Robert E McCulloch. 2010. "BART: Bayesian Additive Regression Trees." *The Annals of Applied Statistics* 4 (1): 266–98.

Correll, Michael, Dominik Moritz, and Jeffrey Heer. 2018. "Value-Suppressing Uncertainty Palettes." In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–11.

Csardi, Gabor, and Tamas Nepusz. 2006. "The Igraph Software Package for Complex Network Research." *InterJournal* Complex Systems: 1695. https://igraph.org.

Earle, Denise, and Catherine Hurley. 2015. "Advances in Dendrogram Seriation for Application to Visualization." *Journal of Computational and Graphical Statistics* 24 (March). https://doi.org/10.1080/10618600.2013.874295.

Felzmann, Heike, Eduard Fosch Villaronga, Christoph Lutz, and Aurelia Tamò-Larrieux. 2019. "Transparency You Can Trust: Transparency Requirements for Artificial Intelligence Between Legal Norms and Contextual Concerns." *Big Data & Society* 6 (1): 2053951719860542.

Fisher, Aaron, Cynthia Rudin, and Francesca Dominici. 2019. "All Models Are Wrong, but Many Are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously." *J. Mach. Learn. Res.* 20 (177): 1–81.

Freeman, Linton C. 2000. "Visualizing Social Networks." *Journal of Social Structure* 1 (1): 4.

Friedman, J. H. 2000. "Greedy Function Approximation: A Gradient Boosting Machine." *The Annals of Statistics* 29 (November). https://doi.org/10.1214/aos/1013203451.

Friedman, J. H., and B. E. Popescu. 2008. "Predictive Learning via Rule Ensembles." *The Annals of Applied Statistics.*, 916–54.

Goldstein, Alex, Adam Kapelner, Justin Bleich, and Emil Pitkin. 2015. "Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation." *Journal of Computational and Graphical Statistics* 24 (1): 44–65. https://doi.org/10.1080/10618600.2014.907095.

Grange, Stuart K, David C Carslaw, Alastair C Lewis, Eirini Boleti, and Christoph Hueglin. 2018. "Random Forest Meteorological Normalisation Models for Swiss PM 10 Trend Analysis." *Atmospheric Chemistry and Physics* 18 (9): 6223–39.

Greenwell, Brandon M. 2017. "pdp: An R Package for Constructing Partial Dependence Plots." *The R Journal* 9 (1): 421–36. https://journal.r-project.org/archive/2017/RJ-2017-016/index.html.

Greenwell, Brandon M., and Bradley C. Boehmke. 2020. "Variable Importance Plots—an Introduction to the vip Package." *The R Journal* 12 (1): 343–66. https://doi.org/10.32614/RJ-2020-013.

Greenwell, Brandon M., Bradley C. Boehmke, and Andrew J. McCarthy. 2018. "A Simple and Effective Model-Based Variable Importance Measure." *arXiv Preprint arXiv:1805.04755.*

Harrison Jr, David, and Daniel L Rubinfeld. 1978. "Hedonic Housing Prices and the Demand for Clean Air." *Journal of Environmental Economics and Management* 5 (1): 81–102.

Hastie, T., R. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd ed. Springer Series in Statistics. Springer-Verlag.

Herman, Ivan, Guy Melançon, and M Scott Marshall. 2000. "Graph Visualization and Navigation in Information Visualization: A Survey." *IEEE Transactions on Visualization and Computer Graphics* 6 (1): 24–43.

Hierholzer, Carl, and Chr Wiener. 1873. "Über Die möglichkeit, Einen Linienzug Ohne Wiederholung Und Ohne Unterbrechung Zu Umfahren." *Mathematische Annalen* 6 (1): 30–32.

Hofert, Marius, and Wayne Oldford. 2020. "Zigzag Expanded Navigation Plots in R: The R Package zenplots." *Journal of Statistical Software* 95 (4): 1–44. https://doi.org/10.18637/jss.v095.i04.

Hurley, Catherine B., and R. W. Oldford. 2011. "Eulerian Tour Algorithms for Data Visualization and the PairViz Package." *Computational Statistics* 26 (December): 613–33. https://doi.org/10.1007/s00180-011-0229-5.

———. 2022. *PairViz: Visualization Using Graph Traversal.* https://CRAN.R-project.org/package=PairViz.

Hurley, Catherine, Mark OConnell, and Katarina Domijan. 2022. *Condvis2: Interactive Conditional Visualization for Supervised and Unsupervised Models in Shiny.*

Hvitfeldt, Emil, Thomas Lin Pedersen, and Michaël Benesty. 2022. *Lime: Local Interpretable Model-Agnostic Explanations.* https://CRAN.R-project.org/package=lime.

Inglis, Alan, Andrew Parnell, and Catherine B Hurley. 2022a. "Visualizing Variable Importance and Variable Interaction Effects in Machine Learning Models." *Journal of Computational and Graphical Statistics* 31 (3): 766–78. https://doi.org/10.1080/10618600.2021.2007935.

Inglis, Alan, Andrew Parnell, and Cathrine Hurley. 2022b. "Visualizations for Bayesian Additive Regression Trees." *arXiv Preprint arXiv:2208.08966.*

Ishwaran, Hemant, Udaya B Kogalur, Eiran Z Gorodeski, Andy J Minn, and Michael S Lauer. 2010.

"High-Dimensional Variable Selection for Survival Data." *Journal of the American Statistical Association* 105 (489): 205–17.

Kuhn, Max, and Davis Vaughan. 2022. *Parsnip: A Common API to Modeling and Analysis Functions*. https://CRAN.R-project.org/package=parsnip.

Lang, Michel, Martin Binder, Jakob Richter, Patrick Schratz, Florian Pfisterer, Stefan Coors, Quay Au, Giuseppe Casalicchio, Lars Kotthoff, and Bernd Bischl. 2019. "mlr3: A Modern Object-Oriented Machine Learning Framework in R." *Journal of Open Source Software*, December. https://doi.org/10.21105/joss.01903.

Maksymiuk, Szymon, Ewelina Karbowiak, and Przemyslaw Biecek. 2021. *EIX: Explain Interactions in 'XGBoost'*. https://CRAN.R-project.org/package=EIX.

Mayer, Michael. 2023. *flashlight: Shed Light on Black Box Machine Learning Models*. https://CRAN.R-project.org/package=flashlight.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. 2021. *E1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. https://CRAN.R-project.org/package=e1071.

Molnar, Christoph. 2022. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. 2nd ed. https://christophm.github.io/interpretable-ml-book.

Molnar, Christoph, Bernd Bischl, and Giuseppe Casalicchio. 2018. "iml: An R Package for Interpretable Machine Learning." *JOSS* 3 (26): 786. https://doi.org/10.21105/joss.00786.

Morgen, Pawel, and Przemyslaw Biecek. 2020. *Corrgrapher: Explore Correlations Between Variables in a Machine Learning Model*. https://CRAN.R-project.org/package=corrgrapher.

Murray, Kim, and Mary M Conner. 2009. "Methods to Quantify Variable Importance: Implications for the Analysis of Noisy Ecological Data." *Ecology* 90 (2): 348–55.

Paluszynska, Aleksandra, Przemyslaw Biecek, and Yue Jiang. 2020. *randomForestExplainer: Explaining and Visualizing Random Forests in Terms of Variable Importance*. https://CRAN.R-project.org/package=randomForestExplainer.

Probst, Philipp. 2020. *varImp: RF Variable Importance for Arbitrary Measures*. https://CRAN.R-project.org/package=varImp.

Purchase, Helen. 1997. "Which Aesthetic Has the Greatest Effect on Human Understanding?" In *Graph Drawing*, 97:248–61.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. "" Why Should i Trust You?" Explaining the Predictions of Any Classifier." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–44.

Schliep, Klaus, and Klaus Hechenbichler. 2016. *kknn: Weighted k-Nearest Neighbors*. https://CRAN.R-project.org/package=kknn.

Strobl, Carolin, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. 2008. "Conditional Variable Importance for Random Forests." *BMC Bioinformatics* 9 (307). https://doi.org/10.1186/1471-2105-9-307.

Sugiyama, Kozo, Shojiro Tagawa, and Mitsuhiko Toda. 1981. "Methods for Visual Understanding of Hierarchical System Structures." *IEEE Transactions on Systems, Man, and Cybernetics* 11 (2): 109–25.

Venables, W. N., and B. D. Ripley. 2002. *Modern Applied Statistics with s*. Fourth. New York: Springer. https://www.stats.ox.ac.uk/pub/MASS4/.

Wei, Taiyun, and Viliam Simko. 2021. *R Package 'corrplot': Visualization of a Correlation Matrix*. https://github.com/taiyun/corrplot.

Wickham, Hadley. 2016. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wright, Marvin N., and Andreas Ziegler. 2017. "ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R." *Journal of Statistical Software* 77 (1): 1–17. https://doi.org/10.18637/jss.v077.i01.

Zeileis, Achim, Jason C. Fisher, Kurt Hornik, Ross Ihaka, Claire D. McWhite, Paul Murrell, Reto Stauffer, and Claus O. Wilke. 2020. "Colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software, Articles* 96 (1): 1–49. https://doi.org/10.18637/jss.v096.i01.

Alan Inglis
Maynooth University
Department of Mathematics and Statistics
Maynooth, Ireland
ORCiD: 0000-0002-1151-6657
alan.n.inglis@gmail.com

Andrew Parnell
Maynooth University

Hamilton Institute
Maynooth, Ireland
ORCiD: *0000-0001-7956-7939*
andrew.parnell@mu.ie

Catherine Hurley
Maynooth University
Department of Mathematics and Statistics
Maynooth, Ireland
ORCiD: *0000-0003-2758-5531*
catherine.hurley@mu.ie