The R Journal

Volume 15/1, March 2023

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

Contents

Editorial 4

Contributed Research Articles

A Hexagon Tile Map Algorithm for Displaying Spatial Data 6
A Clustering Algorithm to Organize Satellite Hotspot Data for the Purpose of Tracking Bushfires Remotely 17
asteRisk - Integration and Analysis of Satellite Positional Data in R 34
gplsim: An R Package for Generalized Partially Linear Single-index Models 55
Non-Parametric Analysis of Spatial and Spatio-Temporal Point Patterns 65
nlmeVPC: Visual Model Diagnosis for the Nonlinear Mixed Effect Model 83
Likelihood Ratio Test-Based Drug Safety Assessment using R Package **pvLRT** 101
GCPBayes: An R package for studying Cross-Phenotype Genetic Associations with Group-level Bayesian Meta-Analysis 122
rankFD: An R Software Package for Nonparametric Analysis of General Factorial Designs 142
The segmetric Package: Metrics for Assessing Segmentation Accuracy for Geospatial Data 159
Onlineforecast: An R Package for Adaptive and Recursive Forecasting 173
markovMSM: An R Package for Checking the Markov Condition in Multi-State Survival Data 195
Robust Functional Linear Regression Models 212
Fairness Audits and Debiasing Using **mlr3fairness** 234
ClusROC: An R Package for ROC Analysis in Three-Class Classification Problems for Clustered Data 254
Resampling Fuzzy Numbers with Statistical Applications: FuzzyResampling Package 271
combinIT: An R Package for Combining Interaction Tests for Unreplicated Two-Way Tables. 284
Estimating Causal Effects using Bayesian Methods with the R Package BayesCACE. . 297
A Framework for Producing Small Area Estimates Based on Area-Level Models in R . 316

News and Notes

Changes on CRAN 342
R Foundation News 344

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

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

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

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

Editorial

by Simon Urbanek

On behalf of the editorial board, I am pleased to present Volume 15 Issue 1 of the R Journal.

Catherine Hurley has stepped down as Editor-in-Chief, but continues to serve on the editorial board as an Executive Editor. Catherine has overseen the expansion of the journal as EIC to accommodate four issues a year as a response to the growing number of publications. Due to her relentless work she was able to maintain stability in challenging times when personal and external circumstances have affected availabilty of our editorial teams. We would like to welcome Vincent Arel-Bundock to our team of Associate Editors.

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

News from CRAN and the R Foundation and RForwards are included in this issue.

This issue features 19 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

Spatial analysis

- A Clustering Algorithm to Organize Satellite Hotspot Data for the Purpose of Tracking Bushfires Remotely
- asteRisk - Integration and Analysis of Satellite Positional Data in R
- Non-parametric analysis of spatial and spatio-temporal point patterns
- The segmetric Package: Metrics for Assessing Segmentation Accuracy for Geospatial Data

Graphics and visualisation

- A Hexagon A Hexagon Tile Map Algorithm for Displaying Spatial Data
- nlmeVPC: Visual Model Diagnosis For The Nonlinear Mixed Effect Model

Bayesian inference

- GCPBayes: An R package for studying Cross-Phenotype Genetic Associations with Group-level Bayesian Meta-Analysis
- Estimating Causal Effect using the Bayesian Method with the R Package BayesCACE

Robust statistical methods

- Resampling Fuzzy Numbers with Statistical Applications: FuzzyResampling Package
- onlineforecast: An R Package for Adaptive and Recursive Forecasting
- Robust functional linear regression models
- A Framework for Producing Small Area Estimates Based on Area-Level Models in R
- GPLSIM: An R Package for Penalized Spline Estimation for Generalized Partially Linear Single-index Models

Design and analysis of experiments

- rankFD: An R Software Package for Nonparametric Analysis of General Factorial Designs
- combinIT: An R Package for Combining Interaction Tests for Testing Interaction in Unreplicated Two-Way Tables

Model deployment

- Likelihood Ratio Test-Based Drug Safety Assessment Using R Package pvLRT
- markovMSM: An R package for checking the Markov Condition in Multi-state Survival Data
- Fairness Audits And Debiasing Using mlr3fairness
- ClusROC: An R package for ROC analysis in three-class classification problems for clustered data

Simon Urbanek
University of Auckland

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

A Hexagon Tile Map Algorithm for Displaying Spatial Data

by Stephanie Kobakian, Dianne Cook, and Earl Duncan

Abstract Spatial distributions have been presented on alternative representations of geography, such as cartograms, for many years. In modern times, interactivity and animation have allowed alternative displays to play a larger role. Alternative representations have been popularised by online news sites, and digital atlases with a focus on public consumption. Applications are increasingly widespread, especially in the areas of disease mapping, and election results. The algorithm presented here creates a display that uses tessellated hexagons to represent a set of spatial polygons, and is implemented in the R package called sugarbag. It allocates these hexagons in a manner that preserves the spatial relationship of the geographic units, in light of their positions to points of interest. The display showcases spatial distributions, by emphasising the small geographical regions that are often difficult to locate on geographic maps.

Introduction

Many cancer atlases present geospatial cancer data on a choropleth map display. The Australian Cancer Atlas (Cancer Council Queensland, Queensland University of Technology, and Cooperative Research Centre for Spatial Information 2018) is a recent addition to the many cancer atlas maps worldwide. The ground-breaking atlas for Australia presents a central map that shows the landmass overlaid with administrative boundaries. This choropleth display can highlight the geographic patterns in geospatially related cancer statistics (Moore and Carpenter 1999).

Over time, the population density in major cities has increased as residents have gathered to live near urban areas (Dorling 2011). Populations tend to be distributed unevenly across geographic regions. When comparing sets of administrative geographic units such as states or electorates, area size is seldom proportional to the population size. In a choropleth map display, the geographic units are coloured to represent the value of the statistic for each unit (Tufte 1990). This can cause some values to be emphasised over others, and allows choropleth map displays to misrepresent the spatial distributions of human related statistics due to area-size bias (Skowronnek 2016).

The Australian Cancer Atlas is an online, interactive display of Australian cancer statistics in each of the Statistical Areas at Level 2 (SA2s), used by the Australian Bureau of Statistics (2011). The dataset of estimated standardised incidence ratios (SIRs) of thyroid cancer for females was downloaded from the publicly accessible Australian Cancer Atlas website and presented in Figure 1, a choropleth map that uses colour to display the distribution. The Australian choropleth map display draws attention to the expanse of dark and light blue areas across the rural communities in all states. The SA2s on the east coast around Brisbane and in northern New South Wales stand out as more orange and red. However, this display neglects the vast amount of Australian residents living in the densely populated capital cities.

The solutions to this visualisation problem begin with the geography. Alternative maps can be created to shift the focus from land area and shape to the value of the statistics (Dougenik, Chrisman, and Niemeyer 1985) in a cartogram display. Cartogram methods apply different transformations to the geographic areas, to highlight the values of the statistic of interest. Alternative maps can result in a distortion of the map space to represent features of the distribution across the areas (Dougenik, Chrisman, and Niemeyer 1985) as the statistic of interest is used to determine the cartogram layout.

Alternative mapping methods, like cartograms implemented in **cartogram** (Jeworutzki 2020) for R (R Core Team 2012), promote better understanding of the spatial distribution of a variable across the population, by representing the population in each administrative area fairly (Levison and Haddon Jr 1965). This acknowledges that the number of residents can be different and also recognises that each area, or person within it is equally important.

This paper contains a discussion of existing mapping practices, followed by details of the algorithm. The implementation section explains the algorithm available in the **sugarbag** package. How to utilise animation with the hexagon map is described. The paper finishes with a summary and possible new directions for the algorithm.

Existing mapping practices

Typically, chloropleth maps are used to orient the users within the geographic context. However, the strong salience of the land mass can hide or downplay the features of the distribution in densely

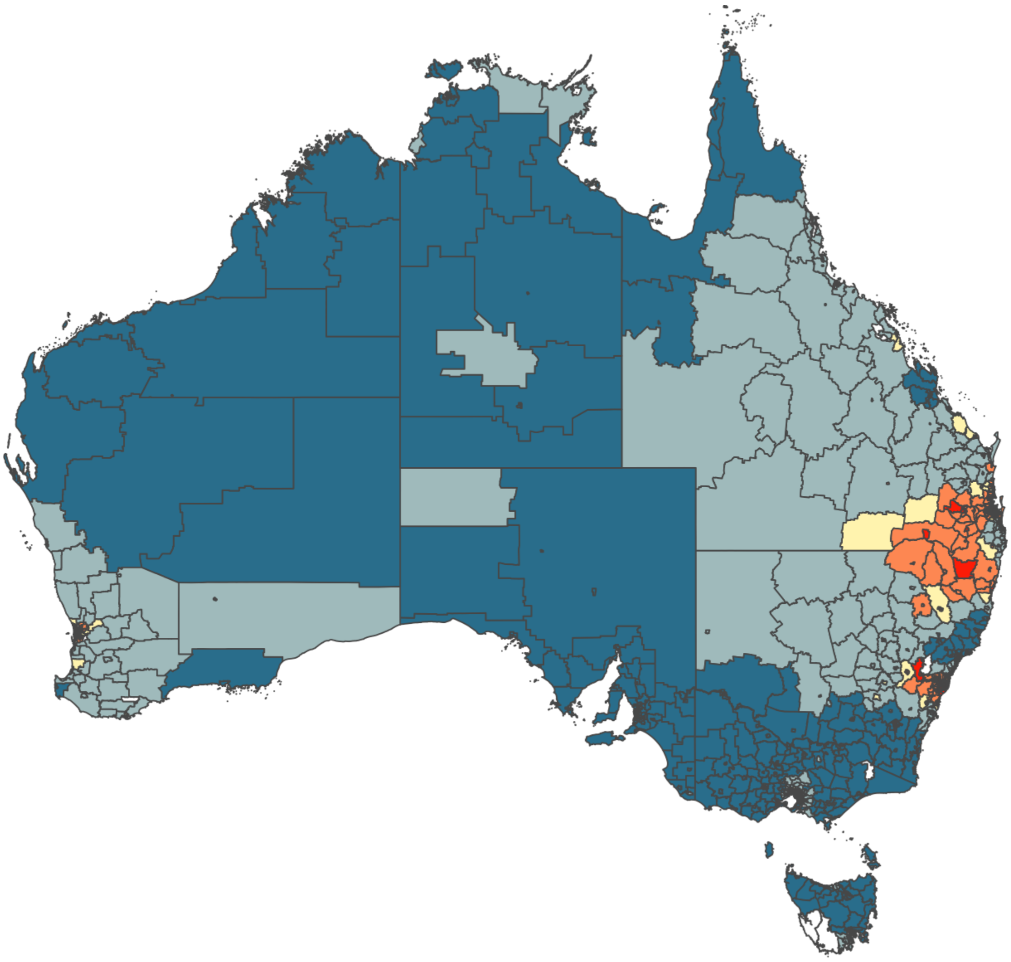


Figure 1: A choropleth map of thyroid incidence among females across the Statistical Areas of Australia at Level 2. Blue indicates lower than average and red indicates higher than average incidence. A cluster of high incidence is visible on the east coast.

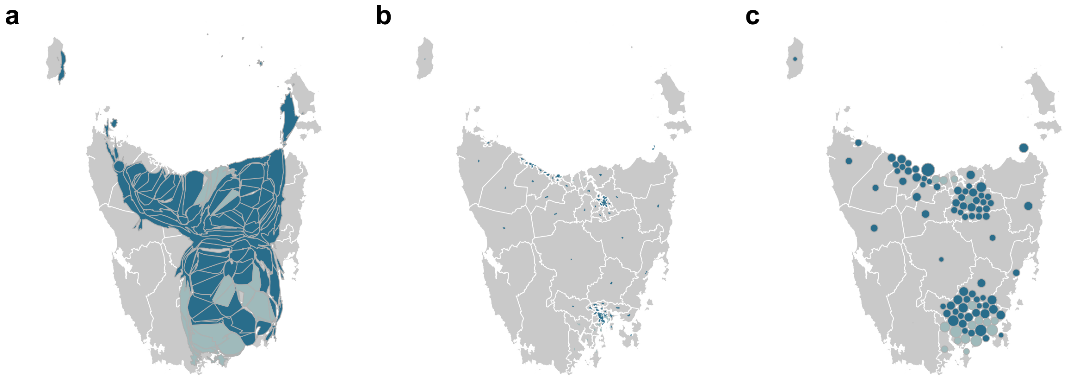


Figure 2: The three displays show alternative maps of the Australian state of Tasmania at SA2 level: (a) contiguous cartogram, (b) non-contiguous cartogram and (c) Dorling cartogram of Tasmania. The contiguous cartogram looks like the state has an hourglass figure, while the non-contiguous cartogram shrinks areas into invisibility. The Dorling expands the metropolitan regions.

populated communities due to the small size on the display (Dorling 2011). The unique shapes of boundaries can be helpful for orienting users but may not contribute to their understanding of the spatial disease distribution as many of the communities are not visible in a choropleth display (Dorling 2012).

Administrative areas are often used to aggregate census data and used to understand demographics within communities of the Australian population. Each SA2 (Australian Bureau of Statistics 2011) was designed to represent a community. This collection of communities presents an opportunity for explicit map transformations to improve communication of spatial distributions (Kocmoud and House 1998). In Figure 2 the choropleth map can be seen underneath each alternative map display to allow for comparisons to be made.

There are several established alternative visualisation methods. Rectangular cartograms (Kreveld and Speckmann 2007) and Dorling cartograms (Dorling 2011) implemented in **cartogram** and tile maps implemented in **tilemaps** (Rosenberg 2020), all use one simple shape to represent each geographic unit. They all minimise the salience of the size or shape of geographic areas. These alternative map displays highlight the relationship between neighbours, preserve connections, and discard the unique shapes of the administrative boundaries. Figure 2 shows a collection of alternative map displays, this includes a) a contiguous cartogram, b) a non-contiguous cartogram, and c) a Dorling cartogram.

When communicating information that is relevant to the population, each member of the population can be given equal representation by transforming the map (Dorling 2012). The connectedness of the units can be preserved by using transformations that maintain the connection between boundaries. The contiguous cartogram displayed in Figure 2a draws attention to smaller geographic units when they are rescaled according to the population (Walter 2001). These new shapes can now be coloured to represent a second variable. The algorithm uses the geographic shape of the areas and iterates toward sizing the areas to represent the population. This display can create twisted and unfamiliar shapes from the geographic units as the algorithms must satisfy the topology conditions, especially when there are communities located geographically far from their neighbours (Dorling 2012).

The non-contiguous cartogram in Figure 2b also uses the population to rescale the geographic units. Unlike the contiguous cartogram, the SA2 areas maintain their geographic shape, but they may not retain the connection with their neighbours. The population of the SA2 areas is used to scale the geographic units in the non-contiguous cartogram (Olson 1976). The amount of background space can be meaningful in non-contiguous cartograms (Keim et al. 2002). However, the relative size of units in comparison to the reference unit can lead to scaling issues. The size disparity between rural areas and urban areas result in reasonable display within the south eastern city of Hobart, but these units are not visible in the context of the Tasmania state map. Depending on the difference between the population and geographic land size, the amount of white space can also prevent meaningful understanding of the distribution (Dorling 2012).

The Dorling cartogram presents each geographic unit as a circle, the size of the circle is scaled according to the population value of each area (Dorling 2011). Figure 2c shows the Tasmania SA2 areas as an individual circle located as close as possible to the geographic centroid location. This map draws attention to the collection of coastal cities in Tasmania that were not apparent in Figure 2 a or b. This display also highlights the difference in the population of each unit, as there is some disparity in the sizes of the circles.

Another common practice is to use equal-sized polygons and treat all geographic units as equal.

The Dorling algorithm can be induced to do this, to make equallly sized circles. Cano et al. (2015) describes a group of alternative maps called "mosaic cartograms" for tile displays. Similarly, the geography of USA has been transformed into a square-binned tile map manually and made available in the R package **statebins** (Rudis 2020).The algorithm described below generates an automatic layout of hexagon tiles, where a contingity constraint is relaxed. This gives equal weight to each geographic area even if the population differs.

The nature of Australian population boundaries means that there are many connecting neighbours of population areas. This solution was designed to visualise Australian population analysis, inspired by tile maps three sided triangles and four sided squares were considered, but this would not have allowed sufficient neighbours to connect. Hexagon tile maps utilise the tessellating features of hexagons to maximise the data ink and allow connectivity between six potential neighbours. Hexagons allow the alignment of geographic borders, this aides in comparison of colour values.

Algorithm

The purpose of this algorithm is to create an alternative map display that highlights distributional features across wide ranges of geographical area size and population density. There has been an increasing need for displays that recognise the large number of people that live in dense urban environments. The algorithm intends to maintain the spatial relationships of a group of geographic units using the relationship between each unit and the closest focal point. The algorithm allocates geographic units to a representative hexagon, in order of their proximity to the closest focal point.

The algorithm is implemented in the **sugarbag** package for R, named after the common name for the *Trigona carbonaria* bee – a species which builds flat layers of hexagonal brood cells, spiralling out from a central point (Vit, Pedro, and Roubik 2013). This unusual pattern also provided the inspiration for the algorithm, where the maps are constructed by building out from multiple focal points on the geographic map base in a spiral fashion.

The result of the algorithm applied to the same data represented in Figure 1 shows rates of thyroid cancer for females in each of the SA2 areas of Australia as hexagons in Figure 3. The difference to the choropleth map in Figure 1 are clear. Now the higher thyroid cancer incidence in the densely populated areas in Sydney, Brisbane and Perth are visible. Interestingly, there is no clear divide between rural and urban SA2 areas, as many rural areas and the cities of Melbourne, Darwin, Adelaide and Hobart have low rates of thyroid incidence for females. The hexagon tile map display provides a more accurate representation of the spatial distribution of thyroid cancer incidence across Australia.

Figure 3 highlights the density of Australian capital cities, as it draws attention to the many communities in Sydney, Melbourne and Hobart. This display also highlights the disparity in the burden of thyroid cancer for females in the communities of these cities. There are several collections of red hexagons in Sydney that represent the communities with much higher rates of diagnosis than the Australian average. Brisbane also experiences higher than average rates of diagnosis, but has more orange than red. The females in the cities of Adelaide and Perth show much lower rates of diagnosis.

Compared to the choropleth map display in Figure 1, the low rates in the rural Australian communities no longer dominate the display. While the corresponding hexagons are still visible against the black background, the lower rates in rural Australia are less prominent.

There are several key steps in the creation of the hexagon tile map as described in the flow chart in Figure 4. First, derive the set of centroids from the polygons provided, then create the grid of hexagon locations. These two processes are described in the blue left column of the flow chart in figure 4. Each centroid can then be allocated to an available hexagon location. The steps for the allocation process are detailed in the right column of Figure 4. There are several filter steps to speed up the process of selecting an appropriate hexagon to represent each geographic unit. To make tessellated plots with the hexagon allocations, the point locations are converted into hexagon shapes.

Implementation

Hexagon tile maps can be useful for visualising distributions across a collection of geographic areas. However, these maps are not easy to create manually, especially as the number of areas increases. This algorithm was developed to automate the process, and reduce the workload involved in creating and implementing alternative displays. This allows map makers and data communicators to spend their time choosing the most effective display.

The **sugarbag** package contains a set of functions that help R users to create a hexagon tile map. The algorithm presented in the **sugarbag** package operates on a set of simple feature geometry objects , known as **sf** objects (Pebesma 2018). This package allows R users to create **sf** objects by importing polygons stored in various formats. Users should provide a set of polygons that define geographic

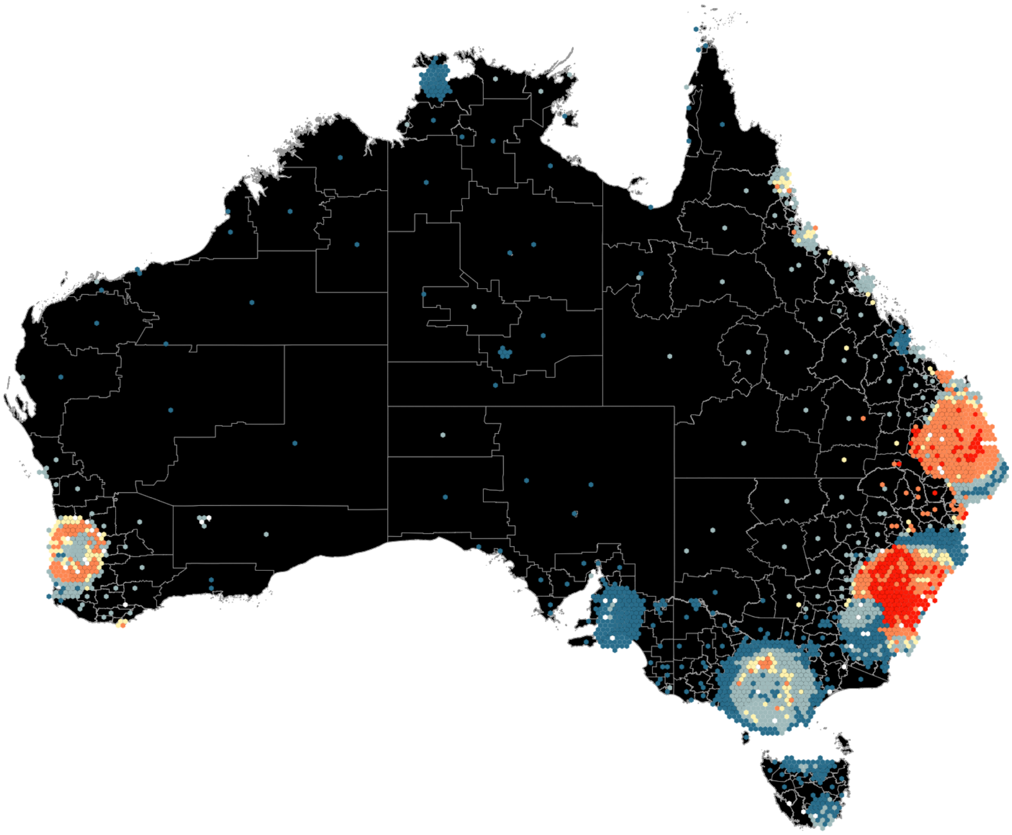


Figure 3: A hexagon tile map of female thyroid cancer incidence in Australia, the same data as shown in the choropleth map in Figure 1. The high incidence in several of the metropolitan regions (Brisbane, Sydney and Perth) can now be seen, along with numerous isolated spots.

units by their administrative boundaries. The functions arrange the geographic units in order of proximity to a set of locations provided, such as the centre of major cities. The centroid location of each geographic unit is used to measure the proximity. It emphasises the major cities as population hubs, rather than emphasizing the size of large, rural geographic units.

The user can tweak the parameters of the hexagon map using additional arguments to the `create_hexmap` function, but these options may affect the speed of the algorithm. The hexagon size may need adjustments depending on the density of the population; users can provide an appropriate hexagon size and re-run the algorithm. The buffer distance may need to be increased if the coastal cities need to extend beyond the geographic land mass.

Algorithm steps

The package can be installed from CRAN and the development version can be installed from the GitHub repository: https://github.com/srkobakian/sugarbag.

The following steps create the hexagon tile map for all the Statistical Areas at Level 2 in Tasmania. These steps can be executed by the main function, `create_hexmap`, or can be run separately for more flexibility.

If a user would like to perform steps of the algorithm themselves, additional user input will be needed for the functions that perform each step. For example, if the user wishes to use a set of centroids, rather than polygons, the `allocate` function can be used directly.

The set of SA2 polygons for Tasmania, using the 2011 definition of SA2s, was accessed from the **absmapsdata** package (Mackey 2022). A single column of the data set is used to identify the unique areas. In this case, the unique SA2 names for each SA2 have been used.

The Australian capital cities are used as focal points to allocate each geographic area around the closest capital city. Hobart will be the focal point for this example because only the state of Tasmania is being processed.

The buffer distance, hexagon size, hexagon amount to filter and width of angle are parameters that will be determined within `create_hexmap`, if they are not provided. They are created as they are

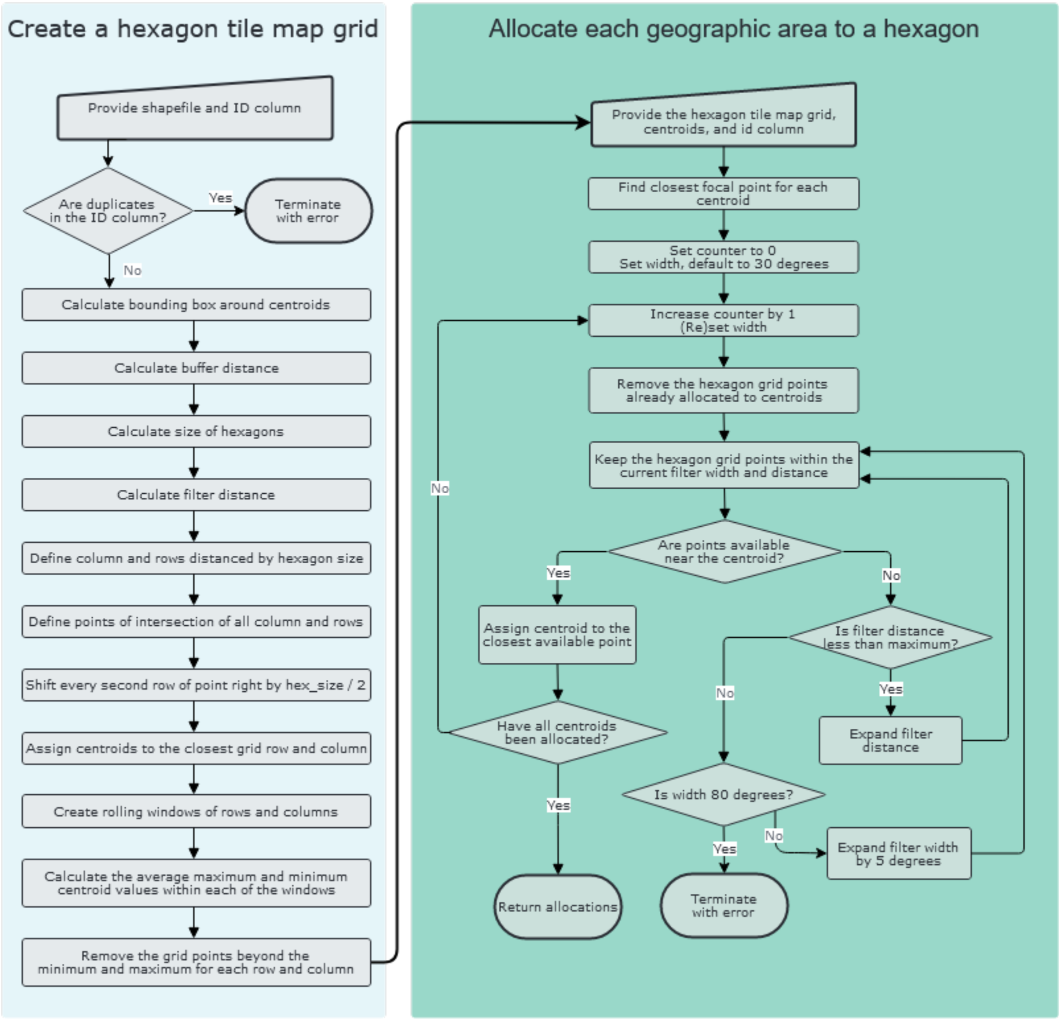

Create a hexagon tile map grid
Provide shapefile and ID column
Are duplicates in the ID column?
Yes
Terminate with error
No
Calculate bounding box around centroids
Calculate buffer distance
Calculate size of hexagons
Calculate filter distance
Define column and rows distanced by hexagon size
Define points of intersection of all column and rows
Shift every second row of point right by hex_size / 2
Assign centroids to the closest grid row and column
Create rolling windows of rows and columns
Calculate the average maximum and minimum centroid values within each of the windows
Remove the grid points beyond the minimum and maximum for each row and column
Allocate each geographic area to a hexagon
Provide the hexagon tile map grid, centroids, and id column
Find closest focal point for each centroid
Set counter to 0 Set width, default to 30 degrees
Increase counter by 1 (Re)set width
Remove the hexagon grid points already allocated to centroids
Keep the hexagon grid points within the current filter width and distance
Are points available near the centroid?
Yes
No
Assign centroid to the closest available point
Is filter distance less than maximum?
Yes
No
Expand filter distance
Have all centroids been allocated?
No
Yes
Is width 80 degrees?
Yes
No
Expand filter width by 5 degrees
Return allocations
Terminate with error

Figure 4: A flow diagram detailing the steps taken to create a hexagon tile map. There are two basic processes, one to make the grid, and the other to allocate centroids to grid points.

needed throughout the following example.

The results from various steps in the process are illustrated in Figure 5.

Derive the set of centroid points

The set of polygons should be provided as an **sf** object, this is a data frame containing a `geometry` column. The `read_shape` function can assist in creating this object for use in R.

The centroids can be derived from an **sf** object using the `create_centroids` function:

Step 1: Create the hexagon grid points

A grid is created to allow tessellation of the hexagons that represent the geographic units. For a hexagon tile map, the grid of possible hexagon locations is made using the `create_grid` function. It uses the centroids, the hexagon size and the buffer distance. Figure 5 1a shows the initial locations of grid points created for Tasmania.

```
centroids <- create_centroids(
  shp_sf = sa2 %>% filter(state_name_2011 == "Tasmania"),
  sf_id = "sa2_name_2011")
```

(a) Creating a tessellated grid A set of longitude columns, and latitude rows are created to define the locations of the hexagons. The distance between each row and column is the size specified by `hex_size`. Equally spaced columns are created from the minimum longitude minus the buffer distance, up to the maximum longitude plus the buffer distance. Similarly, the rows are created from the latitude values and the buffer distance. A unique hexagon location is created from all intersections of the longitude columns and latitude rows. Figure 5 1a shows the hexagon grid after every second latitude row on the grid is shifted right, by half of the hexagon size.

```
grid <- create_grid(centroids = centroids, hex_size = 0.2, buffer_dist = 2)
```

(b) Filtering the grid Not all of the grid points will be used, especially if islands result in a large grid space. To filter the grid for appropriate hexagon locations for allocation, the `create_grid` function will call the `create_buffer` function. It finds the grid points needed to best capture the set of centroids on a hexagon tile map. Reducing the buffer size will decrease the amount of time that the algorithm needs to run for, as there will be less points over the water to consider for each centroid allocation.

For each centroid location, the closest latitude row and longitude column are found. Then rows and columns of centroids are divided into 20 groups. The number of rows in each latitude group and the number of columns in each longitude group are used as the width of rolling windows. The first rolling window step finds the minimum and maximum centroid values within each of the sliding window groups of longitude columns, and the groups of latitude rows. The second rolling window step finds the average of the rolling minimum and maximum centroid values, for the longitude columns and latitude rows.

The hexagon grid points are kept only if they fall within the rolling average of the minimum and maximum centroid values after accounting for the buffer distance, for each row and column of the grid. Figure 5 1b displays remaining hexagon grid points after applying the buffer filter. The sparsely populated South-West region of National Park has fewer points available in the water compared to the South-East region near the city of Hobart.

Centroid to focal point distance The distance is calculated between each centroid in the set, and each of the focal points provided. The order for allocation is determined by the distance between the polygon centroid and it's closest focal point. For this example, this distance is only calculated for the capital city of Hobart, represented in Figure 5 2a and 2b as a white cross.

Step 2: Allocate each centroid to a hexagon grid point

To allocate all centroids the set of polygon centroids and the hexagon map grid are required. The polygon centroids are ordered from the centroid closest to the focal point(s), to the furthest. This preserves spatial relationships with the focal point, as the inner city areas are allocated first and placed closest to the capital, the areas that are further will then be accommodated. The following example considers the first of the Statistical Areas at Level 2. Within the algorithm, these steps are repeated for each polygon.

```
hex_allocated <- allocate(centroids = centroids,
                          sf_id = "SA2_NAME16",
                          hex_grid = grid,
                          hex_size = 0.2, # same size used in create_grid
                          hex_filter = 10,
                          use_neighbours = tas_sa2,
                          focal_points = capital_cities,
                          # same column used in create_centroids
                          width = 30, verbose = TRUE)
```

(a) Filter the grid for unassigned hexagon points After each centroid is located, it is removed from the set of grid points, and is no longer considered in the next step. Keeping only the available hexagon points prevents multiple geographic units from being allocated to the same hexagon. This is demonstrated in Figure 5 2a and 2b by the black hexagons that represent the seven closest polygons to the capital city of Hobart. As the allocation process begins for the eighth closest centroid there are seven unavailable hexagon locations.

(b) Filter the grid points for those closest to the centroid The algorithm creates a circle of points, by only keeping points within a certain radial distance around the original centroid location. Only the hexagons which are close to the centroid and have not been assigned are considered. The number of possible hexagon locations to consider for a centroid is determined by the hexagon filter. This is the maximum number of hexagons between the centroid and the furthest considered hexagon. It is used to subset possible grid points to only those surrounding the polygon centroid within an appropriate range. A smaller distance will increase speed, but can decrease accuracy when width of the angle increases.

The `width` parameter is used to take a slice of the remaining points. The slice centres on the angle from the focal point to centroid location. This uses the angle from the closest capital city, to the current centroid as seen in Figure 5 2a. This allows the spatial relationship to be preserved, even when it is allocated to a hexagon that is further from the focal point then the original centroid location.

If no available hexagon grid point is found within the original filter distance and angle, the distance is expanded and only when a maximum distance is reached will the angle expand to accommodate more possible grid points. By default the angle filter for the hexagon grid points create plus and minus 30 degree bounds of the angle from the focal point to the geographic centroid. This will increase if no points can be found within the `hex_filter` distance. The default angle of 30 was chosen to allow the algorithm to find hexagons that best maintained the spatial relationship between the focal point and geographic centroid.

User choices

Only two inputs are necessary to begin the algorithm; the shapefile, and the ID variable. The ID variable should uniquely identify each geographic unit in the shapefile.

The centroids are derived from the shapefile. The number of centroids within the geographic map is used to determine an appropriate hexagon size if one is not provided. The centroids are used to construct a grid. The grid initially covers the entire map, encompassing all the centroid locations and extending in all directions up to the buffer distance. This buffer distance can help to account for densely populated coastal areas, allowing the hexagon locations to spread beyond the coastline.

The centroid set and hexagon tile map grid are necessary for the allocation process. Additionally, a set of reference locations can be provided as focal points, typically compact and/or densely populated areas such as major cities. The algorithm will use the focal points to create the order of allocation, prioritising the closest centroid locations to the focal points. The user can also specify the variable that should be used to determine the order for the allocation.

When allocating representative hexagons, the width parameter can be used to determine the flexibility of positioning. Using the relationship with the nearest focal point, a larger width parameter will increase the amount of available hexagons nearer to the original centroid of the geographic unit. A smaller width will maintain the orientation from the focal point to the centroid when selecting the hexagon location. However, this could mean it is placed further away.

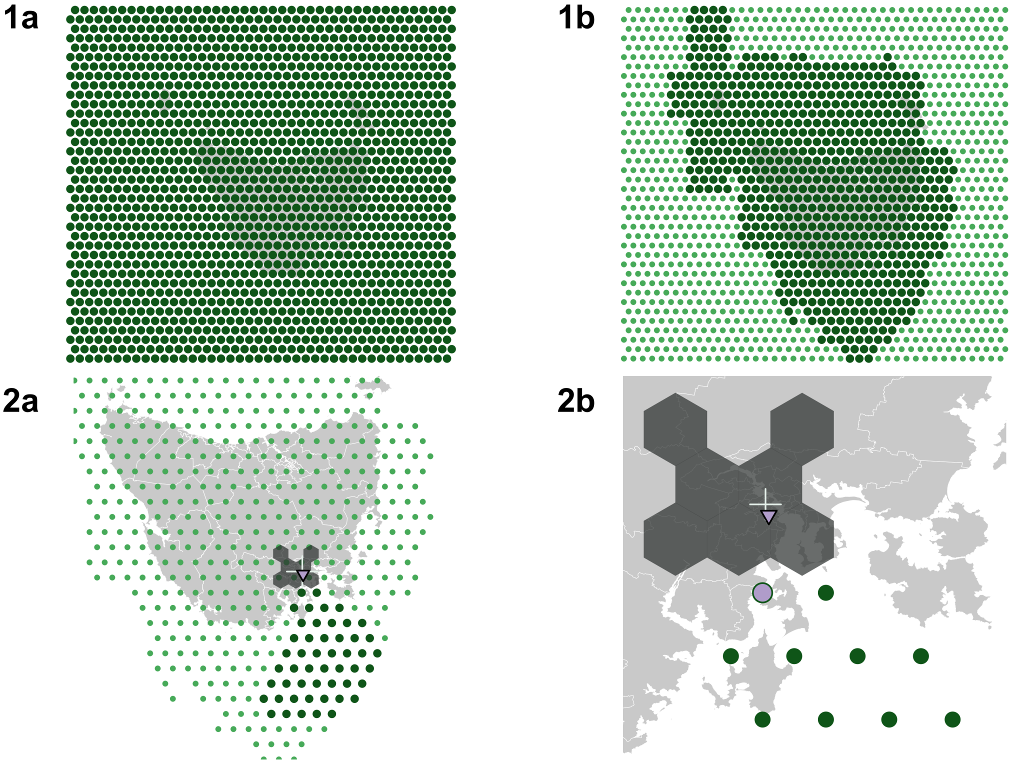


Figure 5: Illustration of key steps of the algorithm: (1a) full hexagon grid is created first; (1b) buffer is applied, shown as dark green circles, to accommodate irregularly shaped regions; (2a, 2b) allocation process, relative the center of Hobart, showing the 8th centroid to be allocated. The relationship between Hobart (the cross) and the centroid (the purple triangle) is used to filter the potential locations from the grid within a wedge. Hexagons already allocated are shown in black, and the purple circle indicates the hexagon to be assigned to the 8th centroid.

Animation

Creating an animation connecting these two displays can allow users to grasp the meaning of the connected hexagons. It will also highlight the density of the communities using the rapid expansion of the inner-city areas, these hexagons will move the furthest and will move rapidly in the animation from their geographic locations. The rapid decrease in size of the large rural areas can show the large size of their collective landmass. The **gganimate** (Pedersen and Robinson 2019) package can be used to make an animation. It connects the polygons for each area in the two displays using the `sf_id` variable, the names of the statistical areas, and creates intermediate displays as they geographic areas shrink or expand to the chosen hexagon size.

Discussion

The present work describes an algorithm called the hexagon tile map. It also shows how this algorithm addresses the potential issues found with contiguous, non-contiguous and Dorling cartograms when applied to the geography of Australia. It achieves a display that effectively presents the density of the population in small geographic areas and de-emphasises the large rural geographic space between the densely populated capital cities. The hexagon tile map display acknowledges that the amount of residents can differ but each administrative area is equally important. This solution uses equally sized areas, and maintain neighbourhood boundary connections.

The hexagon tile map display acknowledges that the amount of residents can differ but each administrative area is equally important. This solution uses equally sized areas, and maintain neighbourhood boundary connections. The **sugarbag** package for R automates the creation of tessellated hexagon tile maps by providing an algorithm to design these displays. The Australian application preserves spatial relationships, and emphasises capital cities. However, similar to the choropleth map display, the tessellation does not allow the size of the hexagons to represent another variable. The algorithm is heavily dependent on the location and number of focal points used, as this determines the order of allocation. With careful consideration of the choropleth map, the small geographic inner city areas may not have been noticed by viewers, but the hexagon tile map display emphasises them. The communities in northern Tasmania and the Northern territory do not draw attention because of their size as in the choropleth, but their colour is still noticeably below average when contrasted with the hexagons in New South Wales.

Australia serves as a good use case for the algorithm as it represents an extremely sparse geography and small dense population areas which cannot be satisfactorily represented by existing cartogram-like algorithms. However, the algorithm is sufficiently general to apply to other geographic regions, and indeed has been applied to make hexagon maps of the UK, USA and France.

There are many possible extensions to the algorithm that might prove valuable in the future. The buffer allows extension beyond the furthest centroids, but there is no mechanism to protect borders by forcing centroids to be located within the geographic borders of a set of regions (e.g. country or state). The algorithm is also currently limited to representing each geographic area with one hexagon only. In the filter step for final selection of a hexagon grid point for each centroid, a direct angle constraint is currently implemented, but it may narrowly miss potentially better hexagon allocation. Some other constraints, such as a logarithmic function may help to refine the final allocation to hexagons closer to the original centroid location. In a country like Australia, the vast empty interior produces a hexagon map with many small isolated hexagons, even when the city centres are hugely expanded. It could be useful to allow variable sized hexagons, primarily to increase the size of the isolated hexagons to improve their visibility.

This new algorithm provides an effective start point for automatically creating hexagon tile maps for any spatial polygon data, and contributes to an extensive body of work that encourages the use of alternative map displays for statistical displays of spatial data.

References

Australian Bureau of Statistics. 2011. “Statistical Area Level 2 (SA2) ASGS Ed 2011 Digital Boundaries in ESRI Shapefile Format.” data cube: ESRI Shapefile, cat. no. 1270.0.55.001. 2011.

Cancer Council Queensland, Queensland University of Technology, and Cooperative Research Centre for Spatial Information. 2018. “Australian Cancer Atlas.” https://atlas.cancer.org.au.

Cano, R. G., K. Buchin, T. Castermans, A. Pieterse, W. Sonke, and B. Speckmann. 2015. “Mosaic Drawings and Cartograms.” In *Computer Graphics Forum*, 34:361–70. 3. Wiley Online Library.

Dorling, Daniel. 2011. “Area Cartograms: Their Use and Creation.” In *Concepts and Techniques in Modern Geography (CATMOG)*, 59:252–60. https://doi.org/10.1002/9780470979587.ch33.

———. 2012. *The Visualisation of Spatial Social Structure*. John Wiley; Sons Ltd.

Dougenik, James A., Nicholas R. Chrisman, and Duane R. Niemeyer. 1985. "An Algorithm to Construct Continuous Area Cartograms." *The Professional Geographer* 37 (1): 75–81. https://doi.org/10.1111/j.0033-0124.1985.00075.x.

Jeworutzki, Sebastian. 2020. *Cartogram: Create Cartograms with r*. https://CRAN.R-project.org/package=cartogram.

Keim, D. A, S. C North, C Panse, and J Schneidewind. 2002. "Efficient Cartogram Generation: A Comparison." In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002*, 2002:33–36. IEEE.

Kocmoud, C, and D House. 1998. "A Constraint-based Approach to Constructing Continuous Cartograms." In *Proc. Symp. Spatial Data Handling*, 236–46.

Kreveld, Marc van, and Bettina Speckmann. 2007. "On Rectangular Cartograms." *Computational Geometry* 37 (3): 175–87. https://doi.org/10.1016/j.comgeo.2006.06.002.

Levison, M. E., and W. Haddon Jr. 1965. "The Area Adjusted Map. An Epidemiologic Device." Journal Article. *Public Health Reports* 80: 55–59.

Mackey, Will. 2022. *Absmapsdata: A Catalogue of Ready-to-Use ASGS (and Other) Sf Objects*.

Moore, Dale A., and Tim E. Carpenter. 1999. "Spatial Analytical Methods and Geographic Information Systems: Use in Health Research and Epidemiology." *Epidemiologic Reviews* 21 (2): 143–61. https://doi.org/10.1093/oxfordjournals.epirev.a017993.

Olson, Judy M. 1976. "Noncontiguous Area Cartograms." *The Professional Geographer* 28 (4): 371–80. https://doi.org/10.1111/j.0033-0124.1976.00371.x.

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.

Pedersen, Thomas Lin, and David Robinson. 2019. *gganimate: A Grammar of Animated Graphics*. https://CRAN.R-project.org/package=gganimate.

R Core Team. 2012. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. http://www.R-project.org/.

Rosenberg, Kaelyn. 2020. *Tilemaps: Generate Tile Maps*. https://CRAN.R-project.org/package=cartogram.

Rudis, Bob. 2020. *Statebins: Create United States Uniform Cartogram Heatmaps*. https://CRAN.R-project.org/package=statebins.

Skowronnek, Alsino. 2016. "Beyond Choropleth Maps – A Review of Techniques to Visualize Quantitative Areal Geodata." Infovis Reading Group WS 2015/16. 2016. https://alsino.io/static/papers/BeyondChoropleths_AlsinoSkowronnek.pdf.

Tufte, Edward R. 1990. *Envisioning Information*. Graphics Press.

Vit, Patricia, Silvia R. M. Pedro, and David Roubik. 2013. *Pot-Honey: A legacy of stingless bees*. Springer-Verlag New York. http://dx.doi.org/10.1007/978-1-4614-4960-7.

Walter, S. D. 2001. *Disease Mapping: A Historical Perspective*. Oxford University Press. https://doi.org/https://dx.doi.org/10.1093/acprof:oso/9780198515326.003.0012.

Stephanie Kobakian

Sydney, Australia
ORCiD: 0000-0002-4616-0048
stephanie.kobakian@gmail.com

Dianne Cook
Monash University
Department of Econometrics and Business Statistics
ORCiD: 0000-0002-3813-7155
dicook@monash.edu

Earl Duncan
Queensland University of Technology
School of Mathematical Sciences
ORCiD: 0000-0002-5146-7810
earl.duncan@qut.edu.au

A Clustering Algorithm to Organize Satellite Hotspot Data for the Purpose of Tracking Bushfires Remotely

by Weihao Li, Emily Dodwell, and Dianne Cook

Abstract This paper proposes a spatiotemporal clustering algorithm and its implementation in the R package spotoroo. This work is motivated by the catastrophic bushfires in Australia throughout the summer of 2019-2020 and made possible by the availability of satellite hotspot data. The algorithm is inspired by two existing spatiotemporal clustering algorithms but makes enhancements to cluster points spatially in conjunction with their movement across consecutive time periods. It also allows for the adjustment of key parameters, if required, for different locations and satellite data sources. Bushfire data from Victoria, Australia, is used to illustrate the algorithm and its use within the package.

Introduction

The 2019-2020 Australia bushfire season was catastrophic in the scale of damage caused to agricultural resources, property, infrastructure, and ecological systems. By the end of 2020, the devastation attributable to these Black Summer fires totalled 33 lives lost, almost 19 million hectares of land burned, over 3,000 homes destroyed and AUD $1.7 billion in insurance losses, as well as an estimated 1 billion animals killed, including half of Kangaroo Island's population of koalas (Filkov et al. 2020). According to the Australian Government Bureau of Meteorology (2021), 2019 was the warmest year on record in Australia, and the period from 2013-2020 represents eight of the ten warmest years in recorded history. There is concern and expectation that impacts of climate change – including more extreme temperatures, persistent drought, and changes in plant growth and landscape drying – will worsen conditions for extreme bushfires (CSIRO and Australian Government Bureau of Meteorology 2020; Deb et al. 2020). Contributing to the problem is that dry lightning represents the main source of natural ignition, and fires that start in remote areas deep in the temperate forests are difficult to access and monitor (Abram et al. 2021). Therefore, opportunities to detect fire ignitions, monitor bushfire spread, and understand movement patterns in remote areas are important for developing effective strategies to mitigate bushfire impact.

Remote satellite data provides a potential solution to the challenge of active fire detection and monitoring. Development of algorithms that process satellite imagery into hotspots – points that represent likely fires – is an active area of research (see for example Giglio, Schroeder, and Justice (2016), Xu and Zhong (2017), Wickramasinghe et al. (2016), Jang et al. (2019)). Throughout this paper and the associated package, we make use of the Japan Aerospace Exploration Agency (JAXA) Himawari-8 satellite wildfire product (P-Tree System 2020) that identifies the location and fire radiative power (FRP) of hotspots across East Asia and Australia according to an algorithm developed by Kurihara et al. (2020). It contains records of 1,989,572 hotspots from October 2019 to March 2020 with a 0.02^o (~2km) spatial resolution and 10-minute temporal resolution. Detection of bushfire ignition and movement requires the clustering of satellite hotspots into meaningful clusters, which may then be considered in their entirety or summarized by a trajectory.

In this paper, we present a spatiotemporal clustering algorithm to organize hotspots into groups in order to estimate bushfire ignition locations and track bushfire movements over space and time. Inspired by two existing clustering algorithms, namely Density Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al. 1996) and Fire Spread Reconstruction (FSR) (Loboda and Csiszar 2007), our algorithm adopts the notion of noise from DBSCAN, while drawing upon the fire movement dynamics presented in FSR. We generalize the latter's specification of spatiotemporal parameters, thereby providing an intuitive, straightforward, and extendable approach to the complex problem of bushfire identification and monitoring that may be applied to any satellite wildfire product. In clustering hotspots into bushfires of arbitrary shape and size, we aim to capture key fire behavior:

- fire evolution occurs only forwards in time,
- fires can smolder undetectably for some time and then flare up again,
- fires can merge with other bushfires, and
- solitary hotspots, which we classify as noise, may not represent true fires or are otherwise very brief ignitions that do not spread.

This algorithm is implemented in the R package spotoroo, available on CRAN. By enabling the user to cluster satellite hotspot data across space and time, this software provides the ability to relate findings to key factors in bushfire ignition (e.g. weather, proximity to roads and campsites, and fuel sources) and patterns in their spread.

The core functionality of this spatiotemporal clustering algorithm determines whether a hotspot represents a new ignition point or a continuation of an existing bushfire by comparing and combining cluster membership information via incremental updates from one time window to the next. Our algorithm first slices the hotspot data by its temporal dimension into fixed time intervals, according to a user-defined time step. Doing so divides the overall spatiotemporal clustering task into many smaller spatial clustering tasks that may be completed in parallel. Within each time window, which can be considered a static snapshot of hotspots observed across a user-specified number of time intervals, hotspots that fall within the threshold of a user-defined spatial metric of each other are joined into a cluster. Then, proceeding sequentially, we identify whether or not a hotspot was observed in the previous time window. If so, it retains its cluster membership from the previous time window; if not, the hotspot adopts the membership of the nearest hotspot with which it has been clustered. If no such neighbor exists, a hotspot represents the start of a new fire. It is important to note that each hotspot does not necessarily represent an individual fire, so those clusters that do not pass the threshold of a minimum number of hotspots or exist for a minimum amount of time are labelled noise.

As emphasized by Kisilevich et al. (2009), the selection of spatial resolution and time granularity – and relevance of domain knowledge in their choice – are imperative to the understanding and interpretation of resulting clusters. These choices can influence the shape and number of clusters discovered. In the case of satellite hotspot data, these parameters depend on the spatial resolution and temporal frequency at which images are captured. We suggest that parameter tuning can be assessed using a visual heuristic, which also enables selection of appropriate values in a unit-free manner, independent of a satellite's spatial and temporal resolutions.

This paper is organized as follows. The next section provides an introduction to the literature on spatiotemporal clustering and applications to bushfire modeling. Section Algorithm details the steps of the clustering algorithm, and Section Package introduces its implementation in **spotoroo** on CRAN, including demonstration of the package's key functions and visualization capabilities. We illustrate the clustering algorithm's functionality to study bushfire ignition and spread in Victoria, Australia throughout the 2020 bushfire season in Application, and describe a visual heuristic to inform parameter selection. Finally, we give a brief conclusion of the paper and discuss potential opportunities for use of the clustering algorithm.

Background

Han, Kamber, and Pei (2012) overviews clustering methods and groups algorithms into five types: partitioning, hierarchical, density-based, grid-based, and model-based methods. Clustering of hotspot data lends itself nicely to hierarchical and density-based methods because they allow for the identification of clusters of various shapes and sizes without requiring that the user pre-specify the number of clusters. Particularly, density-based methods have the notion of noise, which is convenient for eliminating non-fire events from the clustering result. We therefore focus on a review of density-based methods and refer the reader to Han, Kamber, and Pei (2012) for algorithms in other categories and Kisilevich et al. (2009) for appropriate extensions to spatiotemporal data.

Spatiotemporal clustering based on DBSCAN

Density-based methods separate regions constituting a high density of points from low-density regions by identifying pairwise distances between points, and then requiring that a threshold for their grouping be satisfied (Han, Kamber, and Pei 2012). Density Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al. 1996) is an implementation of this methodology designed to address three challenges of clustering algorithms: (1) requirements of domain knowledge to determine the hyperparameters, (2) arbitrary shape of clusters and (3) computational efficiency. DBSCAN defines a maximum radius to construct a neighborhood around each point. It distinguishes between a core point, for which the number of points that fall in its neighborhood meets a minimum threshold, and a boundary point, whose neighborhood does not meet this threshold, but can be reached via overlapping neighborhoods extending from the core point. Intersecting neighborhoods are defined to be a cluster, while points that cannot be assigned to a cluster are identified as noise. DBSCAN also provides a heuristic to inform selection of threshold and cluster size.

What is often identified as a limitation of DBSCAN – its inability to differentiate between clusters of different densities and those adjacent to each other (Birant and Kut 2007) – is of less concern for the application to satellite data, which by nature is a set of points corresponding to the equidistant center of pixels on grid of latitudes and longitudes. However, its application to spatiotemporal clustering problems, which contain at least three components – spatial location (e.g. latitude and longitude) and time – requires specification of temporal granularity and treatment of temporal similarity (Kisilevich et al. 2009). As such, several extensions to DBSCAN's spatial clustering functionality have been proposed for spatiotemporal clustering solutions.

ST-DBSCAN (Birant and Kut 2007) was developed as an extension of DBSCAN's functionality to cluster points according to their non-spatial, spatial, and temporal attributes, and it simultaneously addresses DBSCAN's limitations regarding identification of clusters of varying densities and differentiation of adjacent clusters. ST-DBSCAN also introduces a second metric that considers similarity of variables associated with temporal neighbors.

Another spatiotemporal extension of DBSCAN called MC1 was developed by Kalnis, Mamoulis, and Bakiras (2005). The authors slice the data into snapshots along the time dimension and define moving clusters using the percentage of common objects shared by clusters in two consecutive snapshots. The clustering result is obtained by linking the outputs of DBSCAN applied on each snapshot with the definition of moving clusters. This algorithm enables tracking of moving clusters such as a convoy of cars moving in a city efficiently. However, the definition of moving clusters assumes that data points are individual objects with unique IDs that can move throughout time, which does not translate to hotspots.

Spatiotemporal clustering with FSR

Satellite hotspot data has been effectively clustered and visualized using DBSCAN, as demonstrated in studies by Nisa, Andrianto, and Mardhiyyah (2014) and Hermawati and Sitanggang (2016). However, it should be noted that DBSCAN does not enable the tracking of fire ignition and movement over time.

FSR (Loboda and Csiszar 2007) addresses this limitation with a focus on fire dynamics, which aim to characterize the ignition location, spatial progression, and rate of movement of individual fire events. It introduces what is effectively a hierarchical-based clustering algorithm to identify fire spread in the Russian boreal forest based on active fire detection from MODIS (Moderate Resolution Imaging Spectroradiometer), which has a temporal frequency of six hours. The algorithm proposed by the authors constructs a tree based on three rules: (1) the earliest observed hotspot is the root of the tree, (2) any node is within a 2.5km radius from its parent and (3) any node is observed no later than four days from its parent. When the tree is closed and there are still unassigned hotspots, the algorithm continues at the earliest unassigned hotspot to construct a new tree. Finally, each tree is a cluster, and the earliest observed hotspot(s) is defined as the ignition point (i.e. a cluster may have multiple ignition points).

The selection of parameters for FSR is tailored to the specific region and data product being utilized. As a result, these parameter settings cannot be immediately generalized or applied to other sources of satellite hotspot data.

Limitations of existing methods for the purposes of bushfire monitoring

When considered in the context of clustering hotspot data, the existing methods discussed in the previous section are not ideal. Here we clarify these limitations that have led to different choices in our algorithm, particularly in combining the clustering results of two consecutive time periods.

ST-DBSCAN's consideration of the similarity of variables associated with consecutive time periods adds an unnecessary complexity for the spatiotemporal clustering of hotspots, for which we focus on three variables (latitude, longitude, time). It also does not guarantee that clusters are processed temporally, which is required to determine bushfire ignition sites. For this reason, we have developed a different handling of the temporal variable for our hotspot clustering.

MC1 applies DBSCAN to each snapshot of data, which has the potential to treat some fire ignitions as noise. It is possible for a fire ignition to be initially identified by only a few hotspots, which may not be enough to meet DBSCAN's density constraint. While we believe it is important to introduce the concept of noise to filter out non-fire events, their identification and removal should not occur at each temporal snapshot for this reason. We have therefore implemented the noise filter differently.

We also consider how best to handle the spatial convergence of clusters, that is, when two or more clusters from a previous time period merge together in the current time period. In such situations, both MC1 and FSR assign only one membership to the cluster of hotspots in the current time period. The consequence of such treatment is twofold: spatial coverage of one fire may increase dramatically in a short amount of time, which may not accurately reflect the natural speed of a bushfire's spread, convergence or shifts in directions. This is not necessarily appropriate in light of our goal of identifying bushfire ignition locations. For this reason, and in contrast to MC1 and FSR, our algorithm inherently enables tracking of individual fires through their potential merging by keeping separate cluster ids in the current time period. This is because cluster membership is informed by a hotspot's appearance in previous time periods. In the event that multiple hotspots are observed in a cluster's first appearance, we calculate their centroid and record it as the ignition location.

Algorithm

Our spatiotemporal clustering algorithm consists of four steps: (1) divide hotspots into equal time intervals, (2) cluster hotspots spatially across a series of time intervals, which we will refer to as a time window, (3) combine cluster information between consecutive time windows, and (4) identify hotspots that represent noise so they can be filtered out. These four steps are described in detail in this section.

1. Divide hotspots into intervals

Because fires progress through time and our aim is to determine the ignition point of any particular bushfire, time is a critical component for determining a fire's starting point and tracking its subsequent progression. To manage this analysis, it is convenient to partition time into intervals to support the spatial clustering of hotspots and eventual combination of results with those of future intervals.

Selection of an appropriate time interval for clustering hotspot data represents a balance between maintaining the resolution of the original satellite data (which may record observations every five minutes or every hour or every four hours, for example) and computational efficiency of the algorithm. For example, a time interval of half an hour produces twice the number of iterations of the spatial clustering algorithm than a time interval of an hour; computation time is approximately linear with number of iterations. Himawari-8 hotspot data is recorded every 10 minutes; using this original temporal granularity has the potential to introduce significant noise, and would represent many more iterations of the clustering algorithm. For mitigation of this noise, we may choose an interval of 60 minutes = 1 hour. We then maintain an overall index of each hour across our entire data set, denoted as $t = 1, ..., T$ where T is the total number of hours in the data set.

A hotspot is a likely indication of a burning fire, although it is possible that a hotspot may disappear between two consecutive intervals, perhaps due to cloud cover or moisture that dampens the fire such that it becomes temporarily undetectable. The algorithm must take these considerations into account when combining information across consecutive time intervals. We need a parameter to measure how long a fire may burn undetected even if we don't see a hotspot; that is, the amount of time a bushfire is considered to be active but unaccounted for in the hotspot data.

For this reason, the parameter *activeTime* is defined to reflect in intervals the maximum amount of time a fire may stay smoldering but undetectable by satellite before flaring up again. For example, if it is reasonable to assume that a bushfire not observed for 24 hours has burned out, the implication on *activeTime* is as follows: if time is divided into half hour intervals, the parameter *activeTime* will be 48, whereas with hour intervals, *activeTime* is 24. The *activeTime* parameter is a unitless integer.

Let $\boldsymbol{S}_t$ be a series of time intervals defined by

$$\boldsymbol{S}_t = [max(1, t - activeTime), t] \quad t = 1, 2, ..., T,$$

where $max(.)$ is the maximum function, t is an integer time index, and T is the integer length of the time window.

Suppose the data set contains T hours of hotspot data, our time interval is one hour, and $activeTime = 24$. This produces a sequence of time windows $\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_T$, where

$$\begin{aligned}
\boldsymbol{S}_1 &= [1, 1], \\
\boldsymbol{S}_2 &= [1, 2], \\
&... \\
\boldsymbol{S}_{25} &= [1, 25], \\
\boldsymbol{S}_{26} &= [2, 26], \\
&... \\
\boldsymbol{S}_{47} &= [23, 47], \\
\boldsymbol{S}_{48} &= [24, 48], \\
&... \\
\boldsymbol{S}_T &= [T - 24, T].
\end{aligned}$$

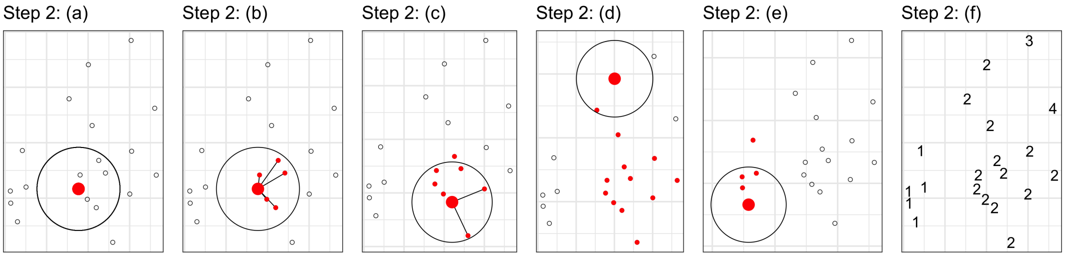


Figure 1: Illustration showing Step 2 of the clustering algorithm on a sample of 20 hotspots in one time window $\boldsymbol{S}_t$. Initially (a), a hotspot is selected randomly ($\boldsymbol{P}$) in order to seed a cluster. The circle indicates the maximum neighborhood distance ($adjDist$). Nearby hotspots as shown in red are clustered with $\boldsymbol{P}$ (b) to initialize list $\boldsymbol{L}$. The neighborhood is moved following every point in that collected list $\boldsymbol{L}$ and new observations are added (c), until there no more points that can be grouped (d). Then a new hotspot is selected external to the existing cluster, and the process is repeated (e). At the end, all the hotspots will be clustered (f).

2. Cluster hotspots spatially within each time window

We next cluster the hotspots within each time window. Each time window represents a static unified view of all hotspots observed across 25 consecutive time intervals (as of S_{25}). The clustering within a time window disregards the specific hourly time index associated with each observed hotspot to focus exclusively on the spatial relationship between them. A hierarchical-based method with a maximum distance stopping criterion $adjDist$ is applied.

The parameter $adjDist$ is used to represent the maximum distance between two adjacent hotspots, which are connected to form a graph. For example, the Himawari-8 hotspot data collection has 0.02^o resolution, which corresponds to approximately 2000m. Thus, $adjDist$ is set to 3000m to indicate diagonal distance between observations. Every connected component of the graph is then considered as an individual cluster.

Given $adjDist > 0\ m$ and a window $\boldsymbol{S}_t$, the algorithm performs the following substeps:

(a) Append a randomly selected hotspot h_i to a empty list $\boldsymbol{L}$, where h_i is the ith hotspot in the time window $\boldsymbol{S}_t$. Let point $\boldsymbol{P}$ be the first item of the list $\boldsymbol{L}$.

(b) For every $h_i \notin \boldsymbol{L}$, if $geodesic(h_i, \boldsymbol{P}) \leq adjDist$, append h_i to the list $\boldsymbol{L}$.

(c) Let point $\boldsymbol{P}$ be the next item of the list $\boldsymbol{L}$.

(d) Repeat (b) and (c) until the point $\boldsymbol{P}$ reaches the end of the list $\boldsymbol{L}$.

(e) For all hotspots $h_i \in \boldsymbol{L}$, assign a new membership to them. Remove these hotspots from the time window $\boldsymbol{S}_t$. Repeat (a) to (e) until time window $\boldsymbol{S}_t$ is empty.

(f) Recover the time window $\boldsymbol{S}_t$ and record the memberships.

Figure 1 provides an example of this step.

3. Update memberships for next time window

Step 3 assigns membership to hotspots in the next time window ($t = 2, ..., T$) based on memberships in the previous window, as illustrated by Figure 2. The algorithm performs the following substeps:

(a) Collect the hotspots, h_i, belonging to $\boldsymbol{S}_t$, where some may have also been present in $\boldsymbol{S}_{t-1}$, into the set $\boldsymbol{H}_t = \{h_1, h_2, ...\}$. Hotspots that were present in $\boldsymbol{S}_{t-1}$ are labelled, and new hotspots are unlabeled.

(b) Construct the connectivity graph, based on $adjDist$, measuring proximity between all hotspots in $\boldsymbol{H}_t$. There may be some hotspots from multiple $\boldsymbol{S}_{t-1}$ clusters connected together here, due to new hotspots appearing between previous hotspots. This is resolved at the next step.

(c) For all unlabeled hotspots, use the label of the closest labelled point, if it is connected in the graph. Any unlabeled hotspots not connected to labelled hotspots are assigned a new label (cluster 5 in Figure 2). Also, any connected graph where memberships from multiple $\boldsymbol{S}_{t-1}$ clusters occur are effectively split into the same multiple clusters (clusters 2 and 4 in Figure 2). This would correspond to two existing fires burning into each other, but they keep their original label. These fires might die at this time, or they might progress in different directions.

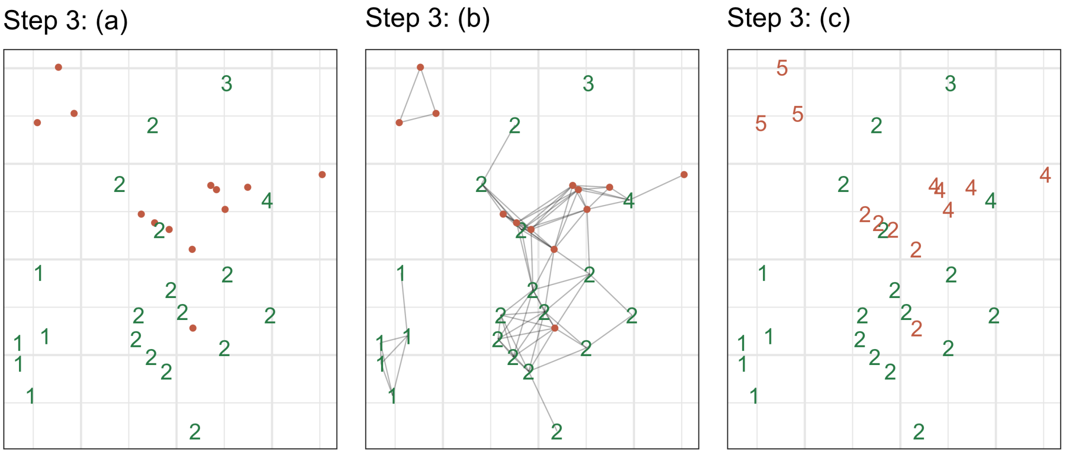


Figure 2: Illustration of clustering Step 3, which involves combining results from one time window to the next. There are 33 hotspots at S_t, where 20 (green) of them have been previously clustered at S_{t-1} (Figure 1 f) and 13 (orange) of them are new hotspots. The connected graph show the clustering in this time window. Hotspots previously clustered at S_{t-1} keep their cluster labels. The 13 new hotspots are assigned labels of the nearest hotspot's cluster label. This might mean that a big cluster S_t (indicated by the graph) would be split back into two, if it corresponded to two clusters at S_{t-1} (e.g. clusters 2, 4). New clusters of hotspots are assigned a new label (e.g. cluster 5).

This consideration of cluster memberships in consecutive time windows enables us to capture the long-term behavior of individual fires, namely whether they go undetected for 24 hours or more. Intuitively, a fire is undetected for 24 hours if a cluster is represented in S_{t-1} but not S_t; that is, all hotspots in the cluster as of time window S_{t-1} were from time index $t - 25$. By moving through the temporal sequence of time windows this way, we gradually identify the ignition time and location of new fires by considering hotspot activity in a region over the last 24 hours.

4. Handle noise in the clustering result

After performing step 3, all hotspots will have been assigned a membership label. However, there may exist small clusters, where small refers to number of associated hotspots and/or length of time for which a cluster is observed. These small clusters are less important for bushfire monitoring due to their limited spread, and we may be less certain that they represent true fires. To address this issue, we provide a noise filter as the last step.

Parameter $minPts$ specifies the minimum number of hotspots a cluster can contain and parameter $minTime$ specifies the minimum amount of time for which a cluster can exist and still be considered a bushfire. Any cluster that does not satisfy these two conditions will be reassigned membership label -1 to indicate they represent noise.

Result

The result of the spatiotemporal clustering algorithm applied to the hotspot data is a vector of memberships with length equal to the number of observations in the data.

Package

Our spatiotemporal clustering algorithm detailed in the previous section is implemented in the R package spotoroo, which is available for download on CRAN:

```
install.packages("spotoroo")
```

The development version can be installed from Github:

```
devtools::install_github("TengMCing/spotoroo")
```

The following demonstration of the package's functionality assumes spotoroo and dplyr have been loaded. We also want to load the built-in data set `hotspots`.

```
library(spotoroo)
library(dplyr)
data(hotspots)
```

The `hotspots` data is a subset of the original Himawari-8 wildfire product and contains 1070 observations observed in the state of Victoria in Australia from December 2019 through February 2020.

To create this illustrative data set, records from the original October 2019-March 2020 data set were filtered to include only those hotspots in the boundary of Victoria's borders with an irradiance over 100 watts per square meter. This threshold of the hotspot's intensity is proposed by landscape ecologist and spatial scientist Dr. Grant Williamson (2020) to reduce the likelihood of including hotspots that do not represent true fires in our analysis. These two processing steps capture 75,936 observations, for which we preserve fields longitude, latitude, and observed date and time; we sample approximately 1% of these records for the package's final `hotspots` data set.

Performing spatiotemporal cluster analysis

The main function of this package is `hotspot_cluster()`, which implements the spatiotemporal clustering algorithm on satellite hotspot data input by the user.

This function accepts a data set of hotspots (`hotspots`) which must contain fields corresponding to longitude (`lon`), latitude (`lat`), and observed time (`obsTime`). Arguments `activeTime`, `adjDist`, `minPts`, and `minTime` were previously defined in the Algorithm section, and represent parameters for the algorithm's functionality. Arguments `timeUnit` and `timestep` represent the conversion from a hotspot's observed time to its corresponding integer time index.

We illustrate the use of function `hotspot_cluster()` below. Our `hotspots` data set has columns with names that correspond to the first four arguments. Parameters `timeUnit` and `timeStep` define the difference between two adjacent time indices as 1 hour, and `activeTime` is 24 time indexes, `adjDist` is 3000 meters, `minPts` is 4 hotspots, and `minTime` is 3 time indices. That is, we consider any cluster that lasts longer than 3 hours and contains at least 4 hotspots to be a bushfire, although since the definition of a bushfire may vary according to geography, these parameters may be adjusted accordingly by the user.

```
result <- hotspot_cluster(hotspots = hotspots,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          timeUnit = "h",
                          timeStep = 1)
```

This function returns a `spotoroo` object, which is a three-object `list` containing two `data.frames` – `hotspots` and `ignition` – and a `list` called `setting`. Printing this output tells us that of the 1070 original hotspots, 10 were identified as noise points, and the remaining 1060 were clustered into 6 clusters.

```
result

#> i spotoroo object: 6 clusters | 1070 hot spots (including noise points)
```

Within `result`, the first item is a `hotspots` data frame. This contains the three fields from our original input data (namely, `lon`, `lat`, and `obsTime`), as well as each hotspot's assigned time index (`timeID`) and cluster `membership`. Each hotspot is also associated with the calculated distance from and time since its associated fire's ignition, `distToIgnition` and `timeFromIgnition`, respectively. Any hotspot with a membership of -1 is identified as noise (`noise == TRUE`).

Our original hotspot data was recorded at 10-minute intervals, and we defined each time interval to be one hour. A hotspot's `timeID` is assigned starting from the first observed timestamp in the data. For example, `2019-12-29 13:10:00` is the earliest `obsTime` in `hotspots` and begins the hour for which `timeID == 1`; any hotspot observed from `2019-12-29 14:10:00` to `2019-12-29 15:00` is assigned `timeID == 2`, and so on. This enumeration continues through the last hour for which we observe hotspots in the data.

```
result$hotspots %>% arrange(obsTime) %>% glimpse()
```

```
#> Rows: 1,070
#> Columns: 10
#> $ lon                   <dbl> 149.30, 149.30, 149.32, 149.30, 149.30, 149.32, 1~
#> $ lat                   <dbl> -37.75999, -37.78000, -37.78000, -37.75999, -37.7~
#> $ obsTime               <dttm> 2019-12-29 13:10:00, 2019-12-29 13:10:00, 2019-1~
#> $ timeID                <int> 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 4, 4, 5, 5, 7, 10, ~
#> $ membership            <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ noise                 <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, ~
#> $ distToIgnition        <dbl> 1111.885, 1111.885, 2080.914, 1111.885, 1111.885,~
#> $ distToIgnitionUnit    <chr> "m", "m", "m", "m", "m", "m", "m", "m", "m", "m",~
#> $ timeFromIgnition      <drtn> 0.0000000 hours, 0.0000000 hours, 0.3333333 hour~
#> $ timeFromIgnitionUnit <chr> "h", "h", "h", "h", "h", "h", "h", "h", "h", "h",~
```

The second item within `result` is the `ignition` data set, which contains one row for each cluster identified by the algorithm. Each cluster's row captures information regarding its ignition location, observed time of ignition, number of hotspots in the cluster, and for how long the cluster was observed. The coordinates of each ignition point represent the earliest observed hotspot associated with the cluster, or the calculated centroid if there are multiple hotspots observed in the first time index associated with the cluster.

```
glimpse(result$ignition)
```

```
#> Rows: 6
#> Columns: 8
#> $ membership         <int> 1, 2, 3, 4, 5, 6
#> $ lon                <dbl> 149.3000, 146.7200, 149.0200, 149.1600, 146.7067, 1~
#> $ lat                <dbl> -37.77000, -36.84000, -37.42000, -37.29000, -36.993~
#> $ obsTime            <dttm> 2019-12-29 13:10:00, 2020-01-08 01:40:00, 2020-01-0~
#> $ timeID             <int> 1, 229, 258, 280, 327, 859
#> $ obsInCluster       <dbl> 146, 165, 126, 256, 111, 256
#> $ clusterTimeLen     <drtn> 116.1667 hours, 148.3333 hours, 146.3333 hours, 12~
#> $ clusterTimeLenUnit <chr> "h", "h", "h", "h", "h", "h"
```

The final object contained in `result` is a list that records the user-input values for the algorithm's parameters.

```
result$setting
```

```
#> $activeTime
#> [1] 24
#>
#> $adjDist
#> [1] 3000
#>
#> $minPts
#> [1] 4
#>
#> $ignitionCenter
#> [1] "mean"
#>
#> $timeUnit
#> [1] "h"
#>
#> $timeStep
#> [1] 1
```

The user can run the function `summary()` to return key distributions that summarize the clustering results. For clusters, it captures the distribution of the number of hotspots in a cluster and the duration of the cluster (in hours). Similarly, for hotspots, it captures the distribution of their distance to ignition, both spatially (in meters) and temporally (in hours). It also notes the number of noise observations identified by the clustering algorithm.

Extracting a subset of clusters

Some users may prefer to engage with a data frame of the results, rather than parsing the spotoroo list object described above. For this purpose, the package provides a function extract_fire(), which converts a spotoroo object to a data.frame by collapsing the hotspots and ignition objects. A new field, type, records whether a row in this data frame represents a hotspot, ignition point, or noise point. Cluster information is available for each hotspot that was not identified to be noise. Users can include or disregard noise points by toggling the argument noise = TRUE.

```
all_fires <- extract_fire(result, noise = TRUE)
all_fires %>% arrange(obsTime) %>% glimpse()
```

```
#> Rows: 1,076
#> Columns: 14
#> $ lon                   <dbl> 149.30, 149.30, 149.30, 149.32, 149.30, 149.30, 1~
#> $ lat                   <dbl> -37.75999, -37.78000, -37.77000, -37.78000, -37.7~
#> $ obsTime               <dttm> 2019-12-29 13:10:00, 2019-12-29 13:10:00, 2019-1~
#> $ timeID                <int> 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 4, 4, 5, 5, 7, 1~
#> $ membership            <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
#> $ noise                 <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, ~
#> $ distToIgnition        <dbl> 1111.885, 1111.885, 0.000, 2080.914, 1111.885, 11~
#> $ distToIgnitionUnit    <chr> "m", "m", "m", "m", "m", "m", "m", "m", "m", "m",~
#> $ timeFromIgnition      <drtn> 0.0000000 hours, 0.0000000 hours, 0.0000000 hour~
#> $ timeFromIgnitionUnit <chr> "h", "h", "h", "h", "h", "h", "h", "h", "h", "h",~
#> $ type                  <chr> "hotspot", "hotspot", "ignition", "hotspot", "hot~
#> $ obsInCluster          <dbl> 146, 146, 146, 146, 146, 146, 146, 146, 146, 146,~
#> $ clusterTimeLen        <drtn> 116.1667 hours, 116.1667 hours, 116.1667 hours, ~
#> $ clusterTimeLenUnit    <chr> "h", "h", "h", "h", "h", "h", "h", "h", "h", "h",~
```

By providing a vector of indices to the argument cluster, the function will extract the corresponding clusters from the clustering result.

```
fire_1_and_2 <- extract_fire(result, cluster = c(1, 2), noise = FALSE)
```

Visualizing the clustering result

The package provides three basic methods to visualize the clustering results, all of which can be produced by the function plot(). Users can draw advanced graphics using the results provided by the algorithm.

Default plot for visualizing the spatial distribution of clusters The default plot allows for visualization of the spatial distribution of the fires. As shown in Figure 3, it presents the hotspots of each cluster identified by the algorithm with a single color. The black dot on top of each colored set of points is the ignition location of that fire. This graphical representation only provides a static view of the hotspot data. To get a dynamic view, users can draw subplots grouped by time.

```
plot(result, bg = plot_vic_map())
```

Fire movement plot for visualizing the fire dynamics Figure 4 shows the path of each fire's movement, which is produced by setting the plot argument type = 'mov'. The fire movement is computed by the get_fire_mov() function. Its argument step controls the number of time intervals combined for calculation of the hotspots' centroid for the purpose of this visualization. Using a small value of step will produce a complex path; for example, step = 1 means that the centroid of hotspots in each hour will be calculated to trace the fire's path, while step = 12 combines intervals over the course of half a day.

Note that centroids won't adequately summarize unusual cluster shapes, and it may be necessary to change the visualization to show a full shape with perhaps a convex hull. This would make for a complicated visualization of temporal movement, though. Also if a fire spreads simultaneously in different directions, the centroids would remain in approximately the same locations.

```
plot(result, type = "mov", step = 12)
```

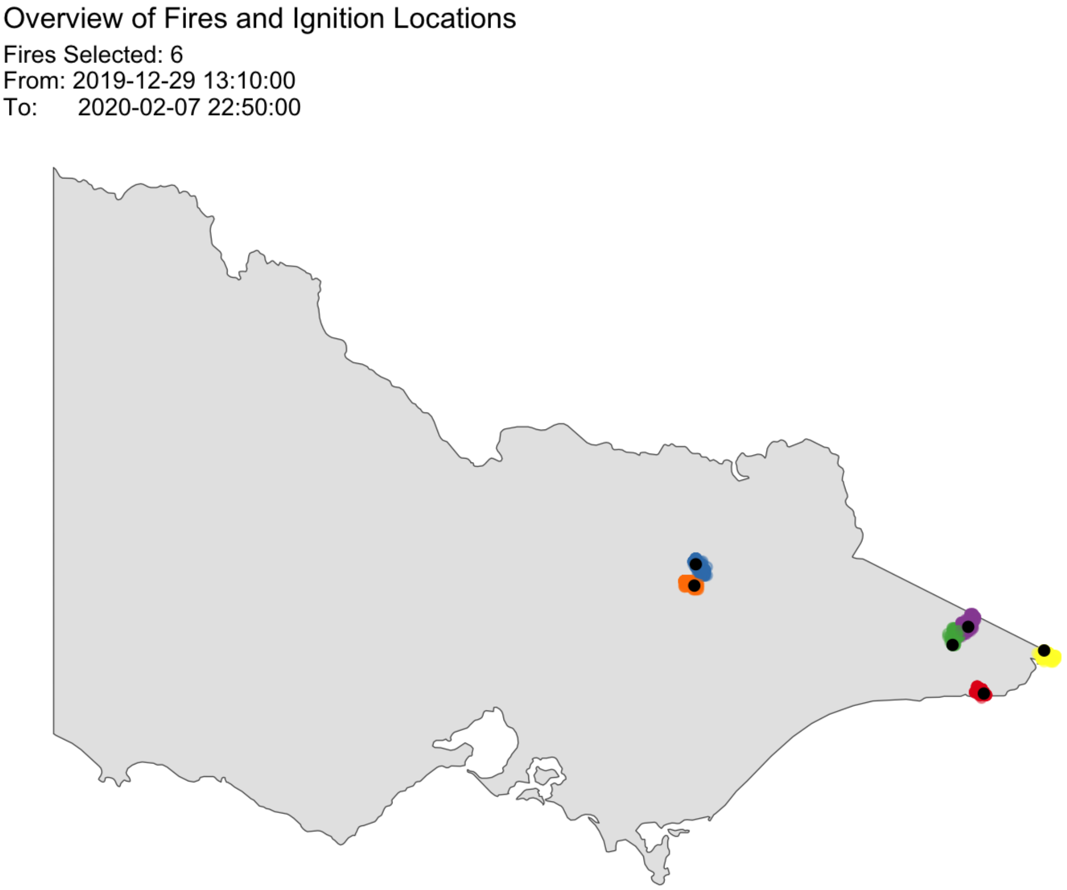

Overview of Fires and Ignition Locations
Fires Selected: 6
From: 2019-12-29 13:10:00
To: 2020-02-07 22:50:00

Figure 3: This is the default plot for visualizing the spatial distribution of clusters. In the results shown there are six clusters, which correspond to six fires, shown using different colors. The black dots indicate the ignition site for each fire.

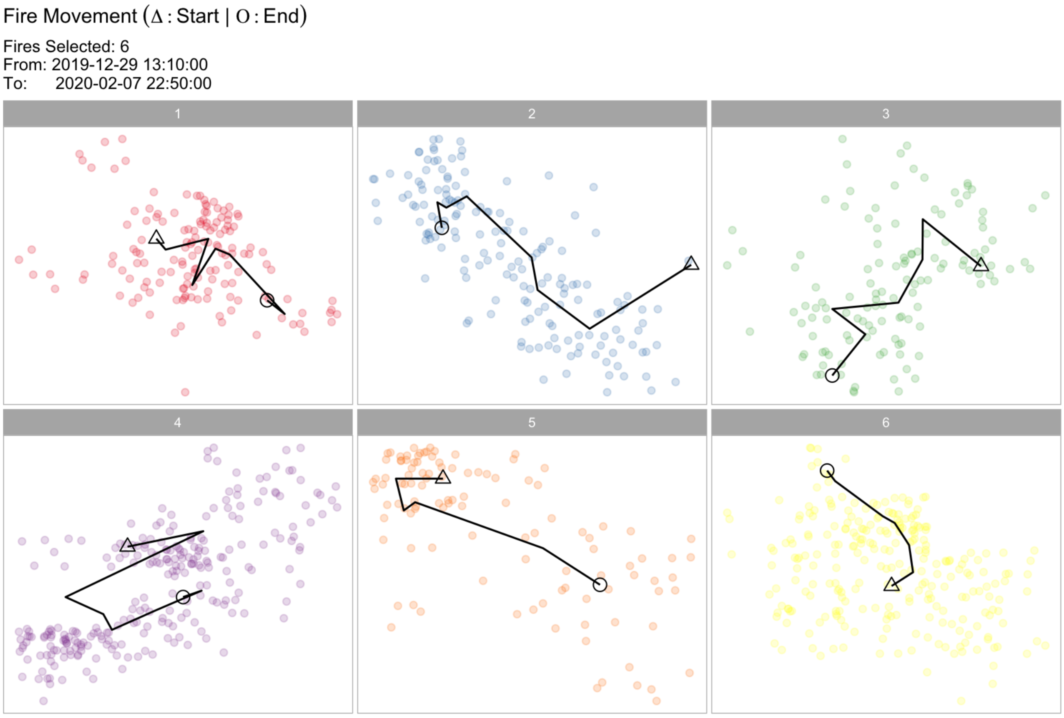


Figure 4: This is the fire movement plot for visualizing the fire dynamics. Here there are six clusters, corresponding to six different fires. The path between the ignition point and the end point is drawn with black line, where the triangle is the ignition point and the circle is the end point. (Note that the aspect ratio of the plot reflects the relative spatial ratio of latitude and longitude.)

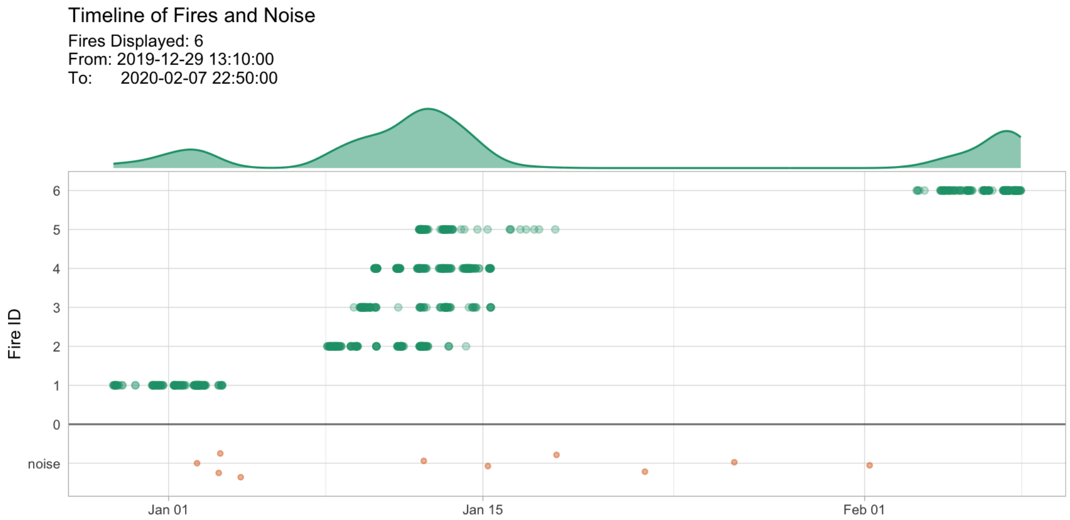


Figure 5: This is the timeline plot for providing an overview of the bushfire season. The x-axis is the date and the y-axis is the cluster membership. The observed time of hotspots are shown as dot plots (green). The density plot at the top display the temporal frequency of fire occurrence over the timeframe. The dot plot at the bottom (orange) shows the observed time of hotspots that are considered to be noise.

In this plot, hotspots are slightly jittered from their original recorded locations on the spatial grid so repeat observations at a specific latitude and longitude are visible. The triangle identifies the fire's ignition point and the circle is the fire's final location. This provides us with a view into the distribution of hotspots over the course of the fire, as well as the bushfire's overall movement. For example, fire 1 ended southeast of its original ignition location.

Timeline plot for providing an overview of the bushfire season Figure 5 shows a timeline of fires produced by setting plot argument `type = 'timeline'`, which enables study of the intensity of the bushfire season. Hotspots' observed times are plotted along the horizontal line corresponding to their assigned clusters. The green curve along the top captures the temporal density of all fires throughout the season. Noise points are represented with orange dots. The plots demonstrates that the majority of fires in our sample data set burned in mid-January.

```
plot(result, type = "timeline")
```

Application

We illustrate the use of the algorithm by applying it to the entirety of the 75,936 hotspots in Victoria. The spatial distribution of these hotspots for October 2019-March 2020 is shown in Figure 6. This example takes around 5 minutes to run on a 2020 Apple MacBook Pro.

Clustering the Victorian hotspot data

To perform the clustering algorithm on the Victorian hotspot data, we first convert the observed time of each hotspot to its corresponding time index by setting the time difference between two successive indices to be 1 hour via parameters `timeStep` and `timeUnit`. As initially discussed in Algorithm, because the original Himawari-8 data is recorded at 10-minute intervals, reassigning each hotspot to its corresponding hourly time index does not cause great loss of information, but serves to significantly shorten the computation time. We then let `activeTime` be 24 time indices and `adjDist` be 3000 meters; see Selecting parameter values for further discussion of and guidance regarding selection of these choices. Finally, `minPts` is 3 hotspots and `minTime` is 3 time indices.

```
result <- hotspot_cluster(hotspots = vic_hotspots,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          timeUnit = "h",
                          timeStep = 1)
```

The clustering result shows that 407 bushfires are identified among the 75,936 hotspots. This result is reasonable, as historically the number of bushfires reported in Victoria in a year ranges from about 200 to 700 (Department of Environment, Land, Water & Planning 2019).

```
result

#> i spotoroo object: 407 clusters | 75936 hot spots (including noise points)
```

Determining the ignition point and time for individual fires

The clustering result provides the ignition location of the bushfire represented by each cluster, and the `plot()` function now enables us to examine the spatial distribution of bushfire ignitions over the course of the entire summer. The result is given in Figure 6. From this plot, we observe that the large majority of fires burned in eastern Victoria, with a smattering in the south west. These areas are primarily forests and mountains. Very few fires started in or around city of Melbourne, which is located along the southern coast in the middle of the state.

Note that because there are now too many fires to color each of them uniquely (as done previously in Figure 3, the default `plot` function colors each cluster of fires black with their respective ignition points in red. This change in plot output occurs once the number of clusters is greater than nine.

```
plot(result, bg = plot_vic_map(), hotspot = TRUE)
```

The ignition time of individual fires and representation of how long they burned is produced with the following code, the output of which is Figure 7. This plot shows that the majority of fires were ignited from late December 2019 to early January 2020. We observe a significant number of hotspots identified to be noise points in mid-December, which may imply there are some undetected fire events that were short-lived.

```
plot(result, type = "timeline", mainBreak = "1 month", dateLabel = "%b %d, %y")`.
```

Tracking fire movement

We further study the movement of the four most intensive fires, as identified by the number of hotspots clustered together. These fires burned in the eastern part of Victoria, and the output of the following code is shown in Figure 8. These fires burned during the time period from December 18th through January 4th. According to the plot, 3 out of the 4 fires moved over the course of their burning, while the center of fire 163 does not appear to move much. This is because simultaneous spread in different directions kept the centroid of the cluster in a similar location over time.

```
plot(result,
     type = "mov",
     cluster = order(result$ignition$obsInCluster,
                     decreasing = TRUE)[1:4],
     step = 12,
     bg = plot_vic_map())
```

Selecting parameter values

We previously introduced two key parameters, *adjDist* and *activeTime*, in our explanation of the algorithm. While their optimal choice is unknown, a visualization tool enables their tuning such that the user can select reasonable values.

Consider the relationships between *adjDist*, *activeTime*, and the percentage of hotspots observed to be noise points. Increase of either *adjDist* or *activeTime* represents greater spatial and temporal

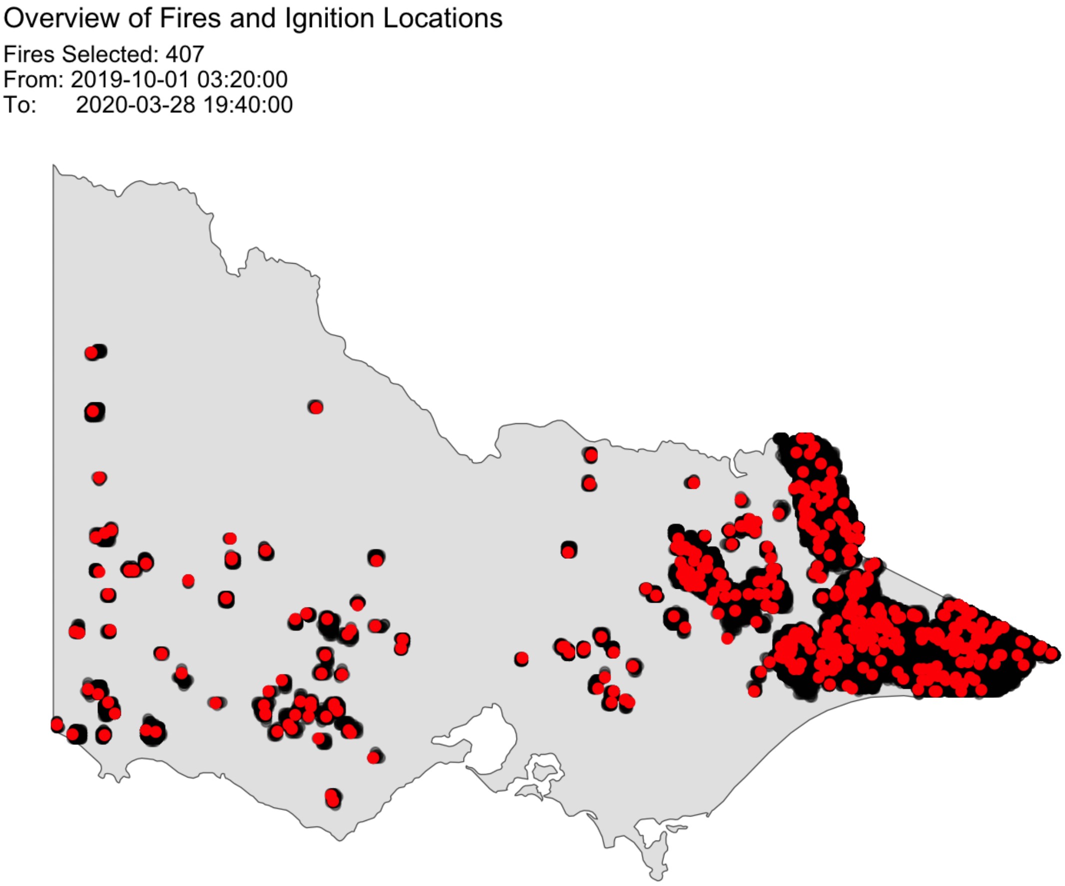

Overview of Fires and Ignition Locations
Fires Selected: 407
From: 2019-10-01 03:20:00
To: 2020-03-28 19:40:00

Figure 6: The distribution of hotspots (black) and bushfire ignitions (red) in Victoria during 2019-2020 Australian bushfire season. The spatial distribution of the ignition locations suggest that most of the fires were observed in the east of Victoria.

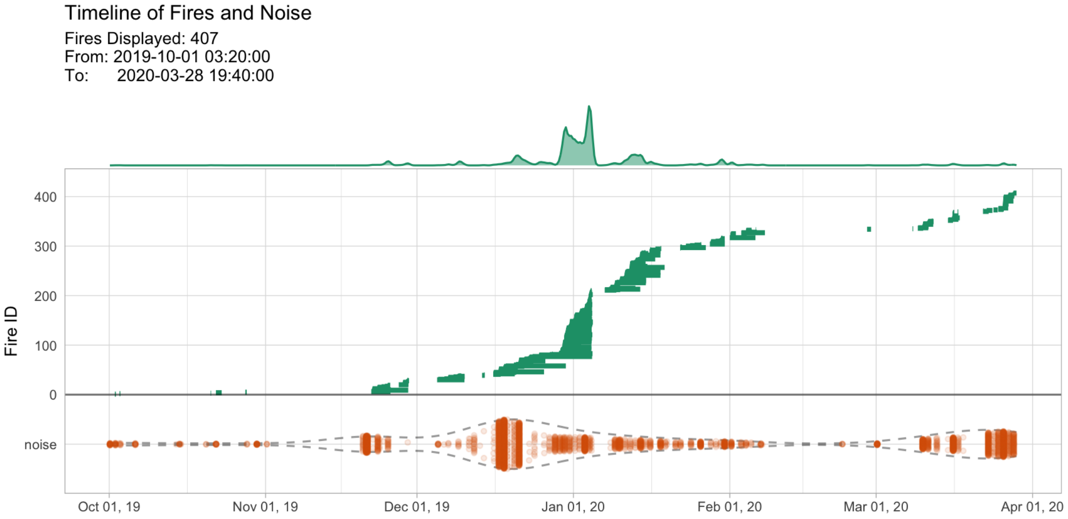


Figure 7: Timeline of fires observed in Victoria during the 2019-2020 Australian bushfire season. Clustered hotspots are shown as dotplots (green). The density display of the timeline shows that most fires started in late December and early January. Noise is shown at the bottom (orange), with the dashed lines indicating the density. This plot shows there is a significant number of hotspots that could be considered to be noise, especially in mid-December. It might also suggest that there are lots of short-lived and spatially constrained fires.

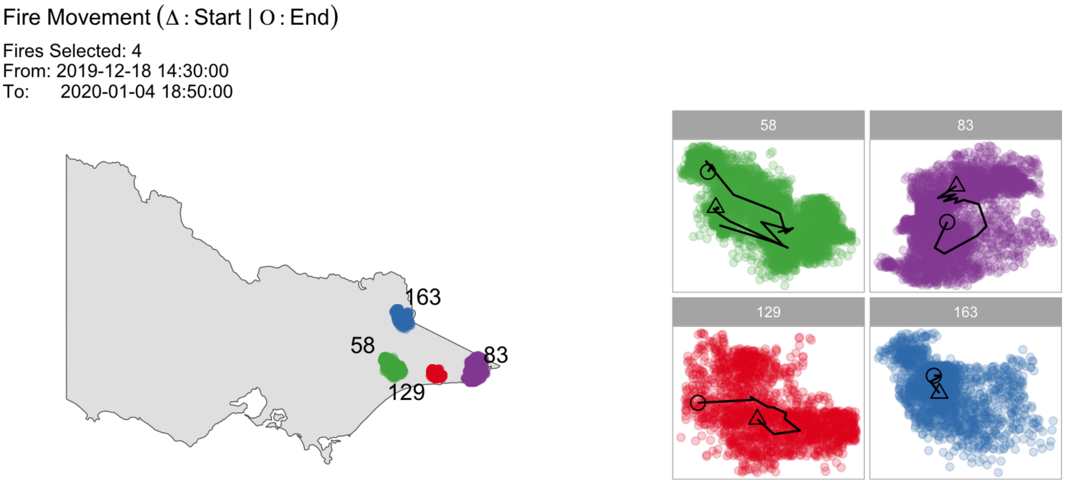

Fire Movement (Δ : Start | O : End)
Fires Selected: 4
From: 2019-12-18 14:30:00
To: 2020-01-04 18:50:00
163
58
83
129
58
83
129
163

Figure 8: Examining the dynamics of the four most intensive fires in Victoria during the 2019-2020 Australian bushfire season. All of the fires covered similar spatial areas over their lifetimes, but the trajectory was quite different. Fire 163 may have spread in many directions simultaneously over the time period, as indicated by the near constant location of the centroid.

tolerance, respectively, either of which usually reduces the percentage of noise points. However, if the noise points are spatially and temporally far from the meaningful clusters, increasing either of these two parameters may not significantly reduce the number of clusters. Therefore, to identify what are likely real noise points, we find the smallest values for each of *adjDist* and *activeTime* after which the percentage of noise points no longer decreases significantly. Based on this logic, we develop a visualization tool inspired by the scree plot (Cattell 1966) used in the principal component analysis and the sorted k-dist plot used in DBSCAN (Ester et al. 1996). Similar to these two plots, users need to determine the values of *activeTime* and *adjDist* for which the greatest decrease in the percentage of noise points is captured. Figure 9 shows the parameter tuning process using this visualization tool. To produce these plots, a grid of parameter values must be evaluated, which can be computationally intensive. For this reason, we do not include this tool in the **spotoroo**, but leave it for the user to build. The final choice of *activeTime* is 24 hours and *adjDist* is 3000 meters.

Conclusions

In this paper, we have proposed a spatiotemporal clustering algorithm to organize satellite hotspot data for the purpose of determining points of bushfire ignition and tracking their movement remotely. This algorithm can be used to cluster hotspots into bushfires of arbitrary shape and size, while following some basic characteristics of bushfire dynamics. It is necessary to tune the parameters of the algorithm. This can be done visually, as demonstrated in the previous section's example application, or determined by expert knowledge of fire dynamics for a geography of interest.

We have provided an implementation of the algorithm in the R package **spotoroo**, which supports three types of visualizations for presenting different aspects of the bushfire data: the spatial distribution of fires, the movement of individual bushfires, and the timeline of fire ignitions and length of burning. In addition, we have illustrated the use of this package in studying the 2019-2020 Australian bushfire season. The main benefit of this work is that information otherwise infeasible to collect about fires in remote areas can be extracted from satellite hotspot data and processed with the use of this package.

We have previously discussed the ways in which existing algorithms for spatial clustering that incorporate a temporal component are not suitable for our intended purpose. Furthermore, there are no R packages on CRAN that provide implementations of ST-DBSCAN, FSR or MC1 in order to run a direct comparison of computational efficiency or clustering performance. There is an implementation of DBSCAN in R package **dbscan** that runs extremely fast (< 10 seconds) on hotspot data from our example application with the proper choice of parameters. However, it clusters over 30% of the hotspots into one fire, which is not very helpful for tracking fire ignition and spread.

There are possible modifications that could be made to our algorithm for the purpose of improving quality of the clustering result or reduced computation time. We might consider alternative clustering techniques – for example, grid-based clustering – at each time window. It may also be possible to use an adaptable *adjDist* at different time windows. We leave these extensions for future research.

More complex analyses can be conducted when clustering results are merged with data such as

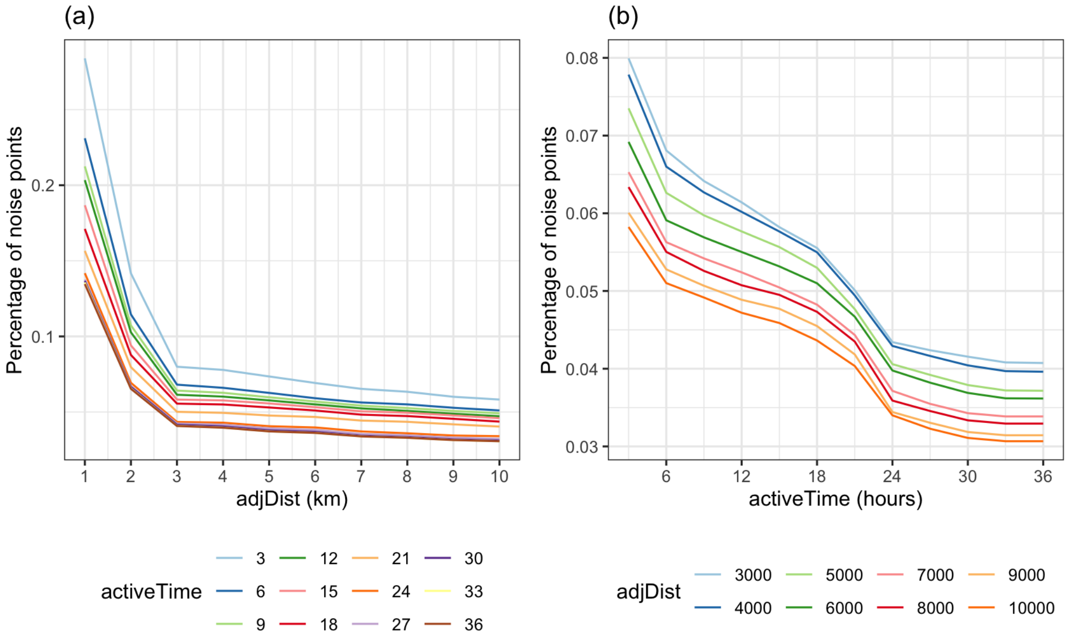


Figure 9: Parameter tuning plots, where the best choice of parameter is at a large drop in the percentage of noise points. Here, these are at $AdjDist = 3000$ and $activeTime = 24$.

recreation sites, fire stations, roads, weather and vegetation types. An analysis of this type to predict the causes of bushfires in Victoria during the 2019-2020 bushfire season is described in Cook (2020), which yields meaningful suggestions for future fire prevention. We hope others can build upon this work to make use of satellite hotspot data for bushfire research, especially in the context of future bushfire planning and prevention.

Acknowledgements

This work was made possible through funding by the Australian Centre of Excellence for Mathematical & Statistical Frontiers, which supported Emily's travel.

This paper was written in **rmarkdown** with **knitr** and **rjtools**. All analyses were performed using R version 4.2.2 (R Core Team 2022). Data were processed with **readr** and **dplyr**. Plots were produced with **ggplot2**, **ggforce** and **patchwork**. The code and data to reproduce this paper are available on GitHub.

References

Abram, Nerilie J., Benjamin J. Henley, Alex Sen Gupta, Tanya J. R. Lippmann, Hamish Clarke, Andrew J. Dowdy, Jason J. Sharples, et al. 2021. "Connections of Climate Change and Variability to Large and Extreme Forest Fires in Southeast Australia." *Communications Earth & Environment* 2 (1): 8. https://doi.org/10.1038/s43247-020-00065-8.

Australian Government Bureau of Meteorology. 2021. "Annual Climate Statement 2020." Australian Government Bureau of Meteorology. http://www.bom.gov.au/climate/current/annual/aus/.

Birant, Derya, and Alp Kut. 2007. "ST-DBSCAN: An Algorithm for Clustering Spatial-Temporal Data." *Data & Knowledge Engineering* 60 (1): 208–21. https://doi.org/10.1016/j.datak.2006.01.013.

Cattell, Raymond B. 1966. "The Scree Test for the Number of Factors." *Multivariate Behavioral Research* 1 (2): 245–76.

Cook, Dianne. 2020. "Open Data Shows Lightning, Not Arson, Was the Likely Cause of Most Victorian Bushfires Last Summer." https://theconversation.com/open-data-shows-lightning-not-arson-was-the-likely-cause-of-most-victorian-bushfires-last-summer-151912.

CSIRO and Australian Government Bureau of Meteorology. 2020. "State of the Climate 2020." CSIRO and Australian Government Bureau of Meteorology. http://www.bom.gov.au/state-of-the-climate/documents/State-of-the-Climate-2020.pdf.

Deb, Proloy, Hamid Moradkhani, Peyman Abbaszadeh, Anthony S. Kiem, Johanna Engstrom, David Keellings, and Ashish Sharma. 2020. "Causes of the Widespread 2019-2020 Australian Bushfire Season." *Earth's Future* 8 (11): e2020EF001671. https://doi.org/10.1029/2020EF001671.

Department of Environment, Land, Water & Planning. 2019. “Fire Origins - Current and Historical.” https://discover.data.vic.gov.au/dataset/fire-origins-current-and-historical.

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.” In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 226–31. KDD’96. Portland, Oregon: AAAI Press. https://dl.acm.org/doi/10.5555/3001460.3001507.

Filkov, Alexander I., Tuan Ngo, Stuart Matthews, Simeon Telfer, and Trent D. Penman. 2020. “Impact of Australia’s Catastrophic 2019/20 Bushfire Season on Communities and Environment. Retrospective Analysis and Current Trends.” *Journal of Safety Science and Resilience* 1 (1): 44–56. https://doi.org/10.1016/j.jnlssr.2020.06.009.

Giglio, Louis, Wilfrid Schroeder, and Christopher O. Justice. 2016. “The Collection 6 MODIS Active Fire Detection Algorithm and Fire Products.” *Remote Sensing of Environment* 178: 31–41. https://doi.org/10.1016/j.rse.2016.02.054.

Han, Jiawei, Micheline Kamber, and Jian Pei. 2012. *Data Mining: Concepts and Techniques, 3rd ed.* Morgan Kaufman. https://www.sciencedirect.com/book/9780123814791/data-mining-concepts-and-techniques.

Hermawati, Rachma, and Imas Sukaesih Sitanggang. 2016. “Web-Based Clustering Application Using Shiny Framework and DBSCAN Algorithm for Hotspots Data in Peatland in Sumatra.” *Procedia Environmental Sciences* 33: 317–23. https://doi.org/10.1016/j.proenv.2016.03.082.

Jang, Eunna, Yoojin Kang, Jungho Im, Dong-Won Lee, Jongmin Yoon, and Sang-Kyun Kim. 2019. “Detection and Monitoring of Forest Fires Using Himawari-8 Geostationary Satellite Data in South Korea.” *Remote Sensing* 11 (3). https://doi.org/10.3390/rs11030271.

Kalnis, Panos, Nikos Mamoulis, and Spiridon Bakiras. 2005. “On Discovering Moving Clusters in Spatio-Temporal Data.” In *Advances in Spatial and Temporal Databases: 9th International Symposium, SSTD 2005, Angra dos Reis, Brazil, August 22-24, 2005. Proceedings 9*, 364–81. Springer. https://link.springer.com/chapter/10.1007/11535331_21.

Kisilevich, Slava, Florian Mansmann, Mirco Nanni, and Salvatore Rinzivillo. 2009. “Spatio-Temporal Clustering.” In *Data Mining and Knowledge Discovery Handbook*, 855–74. Springer. https://link.springer.com/chapter/10.1007/978-0-387-09823-4_44.

Kurihara, Yukio, Kazuhisa Tanada, Hiroshi Murakami, and Misako Kachi. 2020. “Australian Bushfire Captured by AHI/Himawari-8 and SGLI/GCOM-C.” In *Proceedings of the JpGU-AGU Joint Meeting*. https://www.eorc.jaxa.jp/ptree/documents/Poster_H8Wfire_JpGU2020.pdf.

Loboda, T. V., and I. A. Csiszar. 2007. “Reconstruction of Fire Spread Within Wildland Fire Events in Northern Eurasia From the MODIS Active Fire Product.” *Global and Planetary Change* 56 (3): 258–73. https://doi.org/10.1016/j.gloplacha.2006.07.015.

Nisa, Karlina Khiyarin, Hari Agung Andrianto, and Rahmah Mardhiyyah. 2014. “Hotspot Clustering Using DBSCAN Algorithm and Shiny Web Framework.” In *2014 International Conference on Advanced Computer Science and Information System*, 129–32. https://doi.org/10.1109/ICACSIS.2014.7065840.

P-Tree System. 2020. “JAXA Himawari Monitor - User’s Guide.” https://www.eorc.jaxa.jp/ptree/userguide.html.

R Core Team. 2022. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Wickramasinghe, Chathura H., Simon Jones, Karin Reinke, and Luke Wallace. 2016. “Development of a Multi-Spatial Resolution Approach to the Surveillance of Active Fire Lines Using Himawari-8.” *Remote Sensing* 8 (11). https://doi.org/10.3390/rs8110932.

Williamson, Grant. 2020. “Example Code to Generate Animation Frames of Himawari-8 Hotspots.” https://gist.github.com/ozjimbob/80254988922140fec4c06e3a43d069a6.

Xu, Guang, and Xu Zhong. 2017. “Real-Time Wildfire Detection and Tracking in Australia Using Geostationary Satellite: Himawari-8.” *Remote Sensing Letters* 8 (11): 1052–61. https://doi.org/10.1080/2150704X.2017.1350303.

Weihao Li
Monash University
Econometrics and Business Statistics
ORCiD: 0000-0003-4959-106X
weihao.li@monash.edu

Emily Dodwell

emdodwell@gmail.com

Dianne Cook
Monash University
Econometrics and Business Statistics
`http://www.dicook.org`
ORCiD: 0000-0002-3813-7155
`dicook@monash.edu`

asteRisk - Integration and Analysis of Satellite Positional Data in R

by Rafael Ayala, Daniel Ayala, Lara Sellés Vidal, and David Ruiz

Abstract Over the past few years, the amount of artificial satellites orbiting Earth has grown fast, with close to a thousand new launches per year. Reliable calculation of the evolution of the satellites' position over time is required in order to efficiently plan the launch and operation of such satellites, as well as to avoid collisions that could lead to considerable losses and generation of harmful space debris. Here, we present **asteRisk**, the first R package for analysis of the trajectory of satellites. The package provides native implementations of different methods to calculate the orbit of satellites, as well as tools for importing standard file formats typically used to store satellite position data and to convert satellite coordinates between different frames of reference. Such functionalities provide the foundation for integrating orbital data and astrodynamics analysis in R.

1 Introduction

Since the launch of Sputnik 1 in 1957, the developments in space technology have enabled a fast growth of the amount of satellites placed in orbit around Earth. Such growth has become especially prominent in the last decade, as evidenced by the fact that over a third of the 11,697 satellite launches performed until September 2021 occurred over the last 5 years (United Nations). The number of organizations behind satellite launches has also considerably expanded, and currently includes not only government and military agencies, but also private entities. This is largely due to technological developments, such as miniaturized satellites, that have made the process of placing a satellite on orbit around Earth increasingly accessible (Toorian et al. 2008; Bouwmeester and Guo 2010; Kirillin et al. 2015).

Planning the launch, operation and orbital maneuvers of satellites requires predictions of the trajectory that a satellite will follow along time from a known state vector (a set of parameters that define the position, velocity and acceleration of the satellite at a given instant). Such methods are also useful for studying the simultaneous evolution of multiple satellites, with the aim to predict and avoid collisions. This is of special interest in the light of the increasing density of Earth-orbiting satellites, since uncontrolled collisions could lead to the onset of Kessler syndrome (Kessler and Cour-Palais 1978; Kessler et al. 2010), a positive-feedback scenario where the generated space debris impacts on other objects, producing further cascading collisions. This would eventually hinder space operations in some orbits for very long periods of time. For these reasons, multiple models have been developed to calculate the evolution (propagate) of the position of a satellite over time from a set of initial, known conditions.

We introduce **asteRisk**, an R package that aims to provide a suite for astrodynamics analysis in R. To that extent, implementations of some of the most frequently applied orbital propagators are provided, including the SGP4 and SDP4 models (Dong and Chang-yin 2010), as well as a high-precision numerical orbital propagator. Additionally, utilities for reading file formats commonly used to distribute satellite positions (ephemeris) are provided. Satellite ephemerides are relatively scarce compared to other types of positional data (such as that available for standard aviation) (Schäfer et al. 2014), due to the much higher technical complexity of the equipment required to obtain experimental observations that allow the calculation of the position of satellites, as well as to the sensitivity level of such information. As a consequence, only limited sources of data are available for most satellites, such as CelesTrak (Kelso) and Space-Track (SAIC), with the notable exception of Global Navigation Satellite Systems (GNSS) and Planet Labs nanosatellites (Foster et al. 2015). In spite of the scarcity of the data, the provided orbital propagators can extend its scope by calculating future and past ephemeris from a given known state vector. The package also provides tools for the calculation of orbital parameters from the coordinates and velocity of a satellite and vice versa, as well as for the conversion of coordinates between different frames of reference. Some of the functionalities provided with **asteRisk** require large data tables. These are provided in the accessory data package **asteRiskData**, which is distributed through a **drat** repository and can be installed by running `install.packages('asteRiskData',repos='https://rafael-ayala.github.io/drat/')`.

In the following sections, we describe the features of the package and provide examples of application to real data. Firstly, the supported file formats are described, together with examples of data sources, in Section File formats and data sources. Next, we introduce the different orbital propagators currently implemented in Section Orbital propagators. Finally, the different available frames of reference are presented, and conversion between them is demonstrated in Section Frames of reference.

2 File formats and data sources

As previously mentioned, positional information for satellites is relatively scarce compared to non-space aircraft. CelesTrak and Space-Track distribute data for nearly all well-known, non-classified, Earth-orbiting satellites in TLE format. Additionally, NASA provides information about GNSS satellites through the Crustal Dynamics Data Information System (CDDIS) (Noll 2010) in RINEX format. **asteRisk** provides functionalities for reading-in both types of files, which consist of plain text files structured according to the definitions provided by the organizations that implement each standard.

TLE files

The TLE (Two/Three Line Element) format was originally implemented by the North American Aerospace Deffense Command (NORAD), and has become the standard format for distributing positional information about Earth-orbiting satellites.

The format consists of an initial optional line with up to 24 characters (title line) with the name of the satellite corresponding to the TLE file, followed by two 69-character lines that include the information required to calculate orbital elements (a set of parameters that describe the orbit of an object at a given time point), as well as some additional metadata about the satellite (Kelso). It should be noted that a TLE file can contain multiple TLE data structures, which are concatenated in the same file with no additional separator.

TLE files can be read with the `readTLE` function, which receives the path to a file containing one or more TLEs. Both TLEs with and without the additional title line are supported, but all the TLEs in the file should be of the same type. Alternatively, the `parseTLElines` can be used to obtain the same information from a character vector where each element is a string representing a TLE (including the new-line characters).

It should be noted that TLE files were designed to be used in conjunction with the SGP4 or SDP4 orbital propagators (described in detail in Section The SGP4 model). This is because the values provided in TLEs are not osculating/Keplerian orbital elements of a satellite, but instead mean elements calculated to best fit multiple observations by the entity distributing the file. Therefore, as a general rule it is not recommended to use the information obtained from a TLE file as input to orbital propagators other than SGP4/SDP4. However, it is often the case that TLEs are the single available source of information for a given satellite. In such circumstances, if the application of other propagators is desired, it is advised to apply the SGP4/SDP4 model to propagate the orbit to the same instant corresponding to the values of orbital elements distributed in the TLE. This will generate position and velocity values in Cartesian coordinates that are better suited to be used as input for other algorithms, although unpredictable errors are still likely to be present (Janson et al. 2020).

In the following example, we read a file containing TLEs retrieved from CelesTrak for the International Space Station (ISS) and a Molniya satellite (a satellite model for military and communications purposes launched by the Soviet Union and later by the Russian Federation from 1965 to 2004):

```
## In this example, we read a test TLE file containing 2 TLEs, one for the
## Zarya module of the International Space Station and another one for a
## Molniya satellite

test_TLEs <- readTLE("./data/testTLEs.tle")

## We can now check the mean orbital elements contained in each TLE. For
## example, we can calculate the orbital period from the mean motion, provided
## in revolutions/day

meanMotion_ISS <- test_TLEs[[1]]$meanMotion
1/meanMotion_ISS * 24 * 60

## An orbital period of around 93 minutes is obtained for the ISS, in accordance
## with expectations

## Let us check some characteristic parameters of the Molniya satellite
```

```
meanMotion_Molniya <- test_TLEs[[2]]$meanMotion
1/meanMotion_Molniya * 24 * 60
test_TLEs[[2]]$eccentricity

## The Molniya satellite has a period of around 715 minutes and an eccentricity
## of 0.74, in accordance with the elliptical Molniya orbits in which such
## satellites were placed
```

RINEX navigation files

RINEX (Receiver INdependent EXchange) (Gurtner and Estey 2007) is a standard defining multiple formats to distribute satellite navigation systems data, including GNSS. The standard defines three types of files for navigation, observation and meteorological data. Among these, navigation data files provide positional information about satellites.

RINEX defines multiple navigation file formats for different constellations of satellites: GPS, GLONASS, Galileo, BeiDou, IRNSS/NavIC and other regional satellite-based augmentation systems (SBAS). The current version of **asteRisk** supports GPS and GLONASS navigation files, which can be read respectively with functions readGPSNavigationRINEX and readGLONASSNavigationRINEX. Both functions receive as an argument the path to a RINEX navigation file, and return a list with the values for the elements contained in the file. In the case of GLONASS navigation files, these include directly Cartesian coordinates values for the position, velocity and acceleration of the satellite in the ITRF system of coordinates. On the other hand, GPS navigation files provide values for the osculating orbital elements of the satellite in the GCRF system of coordinates, which can be converted to Cartesian coordinates. Functions readGPSNavigationRINEX and readGLONASSNavigationRINEX automatically perform corrections of satellite time to obtain the corresponding accurate UTC times following the procedures described in the specifications of both GNSS (Flores; Space Device Engineering). In the case of GPS RINEX navigation files, conversion of orbital elements to Cartesian coordinates in the ITRF frame is also performed. Additionally, due to the small scale of some of the involved clock corrections (below microseconds), the corrected ephemeris times in UTC are returned as objects of class "nanotime" from the **nanotime** package.

In the following example, a RINEX navigation file for a GPS satellite obtained from the CDDIS is read. A TLE file from CelesTrak for the same satellite at the same instant is also read, and the obtained orbital elements are compared:

```
## Let us read a GPS RINEX navigation file containing a single message

GPS_RINEX <- readGPSNavigationRINEX("./data/RINEX_GPS_1.rxn")

## The resulting list comprises 2 elements: "header" (which is common for all
## navigation messages present in the file) and "messages" (which is a list of
## lists, with one element per message in the top-level list and each of these
## containing elements for the different pieces of information provided in the
## navigation messages). Since there is only one message in the read file,
## "messages" is a list of length 1. We can retrieve orbital parameters from it.
## Note that the values for angle quantities are in radians, which we convert
## to degrees here. The mean motion is in radians per second, which we convert
## to revolutions per day

length(GPS_RINEX$messages) # 1
GPS_RINEX$messages[[1]]$correctedMeanMotion * 86400/(2*pi) # 2.005735
GPS_RINEX$messages[[1]]$eccentricity # 0.002080154
GPS_RINEX$messages[[1]]$inclination * 180/pi # 55.58636
GPS_RINEX$messages[[1]]$meanAnomaly * 180/pi # -37.86007
GPS_RINEX$messages[[1]]$perigeeArgument * 180/pi # 175.6259
GPS_RINEX$messages[[1]]$ascension * 180/pi # -22.95967

## Let us now read a TLE for the same satellite at approximately the same time

GPS_TLE <- readTLE("./data/TLE_GPS.tle")

## We can verify that both the TLE and the RINEX file correspond to the same
## satellite by comparing the PRN codes, which is in both cases 18. A PRN code
## is an identifier unique to each satellite of the GPS constellation.
```

```
GPS_RINEX$messages[[1]]$satellitePRNCode
GPS_TLE$objectName

## We can now check the mean orbital elements provided the TLE file

GPS_TLE$meanMotion # 2.005681
GPS_TLE$eccentricity # 0.0020743
GPS_TLE$inclination # 55.5757
GPS_TLE$meanAnomaly # -46.0714
GPS_TLE$perigeeArgument # 181.977
GPS_TLE$ascension # 37.1706

## As we can see, mean motion, eccentricity, inclination and argument of perigee
## are very similar between the two files. The value for mean anomaly, a
## measurement of where the satellite is along its orbital path, are also
## similar if we convert both of them to the [0, 2*pi) range. However, the
## values for the longitude of the ascending node differ significantly. This
## is due to the fact that the orbital elements provided in the TLE file are
## defined in the TEME frame of reference, while the values in the RINEX file
## are defined in the ITRF frame of reference.
```

3 Orbital propagators

An orbital propagator is a mathematical model for calculating the position of a satellite at future or past time points given a known state vector at a certain time. Multiple propagators have been developed, which differ in the underlying forces being considered and the extent to which assumptions are made to simplify the model. The different propagators therefore offer varying degrees of accuracy and computational costs, with the two often being inversely related.

Three main types of propagators exist: numerical, analytical and semi-analytical. Numerical propagators implement a formulation of a set of forces that act on the satellite, which leads to an expression for the acceleration of the satellite at a given instant (Flores et al. 2021). Numerical integrators are then used to solve the Ordinary Differential Equation (ODE) that defines acceleration as the second-order time derivative of position. While numerical propagators offer the highest accuracy provided that the relevant forces acting on the satellite are correctly modeled, they also have the highest computational costs (especially if implicit integration methods are applied).

Conversely, analytical propagators rely on a set of assumptions and simplifications to obtain closed-form solutions that can be directly evaluated at any given instant as a function of time, the initial known state vector and a set of model parameters. These offer the advantage of having much lower computational costs than numerical propagators, but at the expense of reduced accuracy, particularly for long-term predictions (Deprit 1981; Meeus 1991; Lara et al. 2014).

Semi-analytical methods are an intermediate approach, whereby short-period motions are modeled analytically, and long-term secular effects are solved with numerical integrators (Morand et al. 2013; Lara et al. 2016; Lévy et al. 2021).

In its current version (1.2.0), **asteRisk** provides implementation of two of the most widely applied analytical propagators, the SGP4 and SDP4 models, as well as a numerical high-precision orbital propagator (HPOP).

The SGP4 model

The SGP4 (Simplified General Perturbation 4) model was developed by Ken Cranford in 1970 (Lane and Hoots 1979), based on previous theoretical developments by Lane and Cranford (Lane 1965) whereby the Earth gravitational field is modeled with spherical harmonics up to degree 4 and atmospheric drag is modeled as being spherically symmetric, static, and with density following a power-law decay as altitude increases. The model is mainly designed to be applied to near-Earth satellites with an orbital period of 225 minutes or less, corresponding to an altitude of 5877.5 km if the orbit is assumed to be circular.

The native R implementation available in **asteRisk** is based on the C++ implementation by Vallado (Vallado), which includes minor corrections over the original FORTRAN implementation (Hoots et al. 1988). The validity of the implementation has been verified by comparing the results obtained with previous implementations for a set of test cases (Appendix: Test cases for SGP4/SDP4). The propagator can be applied through the sgp4 function. The function receives as input the following mean orbital elements of the satellite, typically obtained from TLE files: mean motion (in radians/min), mean eccentricity (dimensionless; ranging from 0, a perfectly circular orbit, to 1, a parabolic trajectory), mean orbital inclination (in radians), mean anomaly (in radians), mean argument of the perigee (in radians) and mean longitude of the ascending node (in radians).

Table 1: Description of the arguments taken as input by the SGP4 propagator.

Argument of sgp4 function	**Orbital parameter**	**Units**	**Description**
n0	Mean motion	*radians/min*	Angular speed of the satellite
e0	Eccentricity	Dimensionless	Value between 0 and 1 measuring how much the orbit deviates from a circular shape, with 0 indicating a perfectly circular orbit and 1 an extreme case of parabolic trajectory
i0	Inclination	*radians*	Angle between the orbital plane of the satellite and the equatorial plane
M0	Mean anomaly	*radians*	Angle between the direction of the perigee and the hypothetical point where the object would be if it was moving in a circular orbit with the same period as its true orbit after the same amount of time since it last crossed the perigee had ellapsed. Therefore, 0 denotes that the object is at the perigee
omega0	Argument of perigee	*radians*	Angle between the direction of the ascending node and the direction of the perigee
OMEGA0	Longitude of the ascending node	*radians*	Angle between the direction of the ascending node (the point where the satellite crosses the equatorial plane moving north) and the direction of the First Point of Aries (which indicates the location of the vernal equinox)
Bstar	–	$Earth\ radii^{-1}$	Drag coefficient of the satellite, which indicates how susceptible it is to atmospheric drag
initialDateTime	–	UTC date-time string	Time corresponding to the provided state vector of the satellite
targetTime	–	minutes or UTC date-time string	Time at which propagation should be performed, as a date-time string or in minutes from the initial time
keplerAccuracy	–	Dimensionless	Accuracy to consider Kepler's equation solved. If two consecutive solutions differ by a value lower than this accuracy, integration is considered to have converged
maxKeplerIterations	–	Dimensionless	Maximum number of iterations after which fixed-point integration of Kepler's equation will stop

The function also receives as an argument B^* in units of inverse Earth radii, a modified ballistic coefficient for the satellite which indicates how susceptible it is to atmospheric drag. The target times at which the position of the satellite should be calculated can be specified either as absolute date-time strings in UTC time, or as minutes from the time corresponding to the known state vector. The function outputs the position and velocity of the satellite at the target times in Cartesian coordinates in the TEME (True Equator, Mean Equinox) frame of reference (Section TEME).

In the following example, we demonstrate the application of the SGP4 model to propagate the orbit of the ISS from the previously read TLE:

```
## We can use the mean orbital elements of the TLE of the ISS to propagate its
## position. It should be kept in mind that the mean motion must be input in
## radians per minute, and the mean inclination, anomaly, argument of perigee
## and longitude of the ascending node must be provided in radians.
## Let us propagate the orbit of the ISS for 465 minutes, equivalent to 5
## orbital periods

ISS_TLE <- test_TLEs[[1]]

target_times_ISS <- seq(0, 465, by=5)

results_position_matrix_ISS <- matrix(nrow=length(target_times_ISS), ncol=3)
results_velocity_matrix_ISS <- matrix(nrow=length(target_times_ISS), ncol=3)

for(i in 1:length(target_times_ISS)) {
    new_result <- sgp4(n0=revDay2radMin(ISS_TLE$meanMotion),
                        e0=ISS_TLE$eccentricity,
                        i0=deg2rad(ISS_TLE$inclination),
                        M0=deg2rad(ISS_TLE$meanAnomaly),
                        omega0=deg2rad(ISS_TLE$perigeeArgument),
                        OMEGA0=deg2rad(ISS_TLE$ascension),
                        Bstar=ISS_TLE$Bstar,
                        initialDateTime=ISS_TLE$dateTime,
                        targetTime = target_times_ISS[i])
    results_position_matrix_ISS[i,] <- new_result[[1]]
    results_velocity_matrix_ISS[i,] <- new_result[[2]]
}

results_position_matrix_ISS = cbind(results_position_matrix_ISS, target_times_ISS)
colnames(results_position_matrix_ISS) <- c("x", "y", "z", "time")

## We can now visualize the resulting trajectory using a plotly animation
## In order to create the animation, we must first define a function to create
## the accumulated dataframe required for the animation, which indicates the
## trajectory up to each frame. Frames are defined by propagation time

accumulate_by <- function(dat, var) {
    var <- f_eval(var, dat)
    lvls <- plotly:::getLevels(var)
    dats <- lapply(seq_along(lvls), function(x) {
        cbind(dat[var %in% lvls[seq(1, x)], ], frame = lvls[[x]])
    })
    bind_rows(dats)
}

accumulated_df_ISS <- accumulate_by(as.data.frame(results_position_matrix_ISS), ~time)

## We can then create a plotly animation

orbit_animation_ISS <- plot_ly(accumulated_df_ISS, x = ~x, y=~y, z=~z, type = "scatter3d",
                           mode="lines+marker", opacity=0.8, line=list(width = 6,
                                                                  color = ~time,
                                                                  reverscale = FALSE),
                           frame= ~frame, showlegend=FALSE)

orbit_animation_ISS <- layout(orbit_animation_ISS, scene = list(
    xaxis=list(range=c(-7000, 7000)),
    yaxis=list(range=c(-7000, 7000)),
    zaxis=list(range=c(-7000, 7000))))

## We can also create an animation of a static spheric mesh to represent the
## Earth, and add it to the orbit animation
## First we generate Cartesian coordinates for a sphere of radius equal to
## the radius of Earth. Coordinates are generated along meridians and parallels
## that can be plotted as lines

sphere_theta <- seq(0, 2*pi, length.out=20)
sphere_phi <- seq(0, pi, length.out=20)
sphere_radius <- 6371
sphere_x <- sphere_y <- sphere_z <- numeric(0)

for(theta in sphere_theta) {
    for(phi in sphere_phi) {
        sphere_x <- c(sphere_x, sphere_radius * cos(theta) * sin(phi))
        sphere_y <- c(sphere_y, sphere_radius * sin(theta) * sin(phi))
        sphere_z <- c(sphere_z, sphere_radius * cos(phi))
    }
    sphere_x <- c(sphere_x, NULL)
    sphere_y <- c(sphere_y, NULL)
    sphere_z <- c(sphere_z, NULL)
```

```
}

for(phi in sphere_phi) {
    for(theta in sphere_theta) {
        sphere_x <- c(sphere_x, sphere_radius * cos(theta) * sin(phi))
        sphere_y <- c(sphere_y, sphere_radius * sin(theta) * sin(phi))
        sphere_z <- c(sphere_z, sphere_radius * cos(phi))
    }
    sphere_x <- c(sphere_x, NULL)
    sphere_y <- c(sphere_y, NULL)
    sphere_z <- c(sphere_z, NULL)
}

## Then, we generate an extended dataframe with repetitions of the coordinates
## for a number of times equal to the number of frames in the orbit animation.
## We include a frame column to specify the frame corresponding to each sphere,
## matching the frame numbers

sphere_df <- data.frame(x = sphere_x, y = sphere_y, z = sphere_z)
sphere_df_ext_ISS <- sphere_df[rep(seq_len(nrow(sphere_df)),
                                   length(target_times_ISS)), ]
sphere_df_ext_ISS <- cbind(sphere_df_ext_ISS,
                           rep(target_times_ISS, each=nrow(sphere_df)))
colnames(sphere_df_ext_ISS) <- c("x", "y", "z", "frame")

## We can then use the extended dataframe to create an animation of a static
## sphere

sphere_animated_ISS <- plot_ly(sphere_df_ext_ISS, x=~x, y=~y, z=~z, frame=~frame,
                           type="scatter3d", mode="lines",
                           line=list(color='rgb(0,0,255)'), hoverinfo="skip",
                           showlegend=FALSE)
sphere_animated_ISS <- layout(sphere_animated_ISS, scene = list(
    xaxis = list(showspikes=FALSE),
    yaxis = list(showspikes=FALSE),
    zaxis = list(showspikes=FALSE)))

## The two animations can then be combined and used to visualize the orbit,
## which as we can see is relatively close to the surface of Earth. This is in
## accordance with the ISS being on a LEO (Low Earth Orbit)

combined_animation_ISS <- suppressWarnings(subplot(orbit_animation_ISS,
                                                   sphere_animated_ISS))
combined_animation_ISS <- animation_opts(combined_animation_ISS, frame=50)
combined_animation_ISS <- layout(combined_animation_ISS, scene = list(
    aspectmode = "cube"))
combined_animation_ISS
```

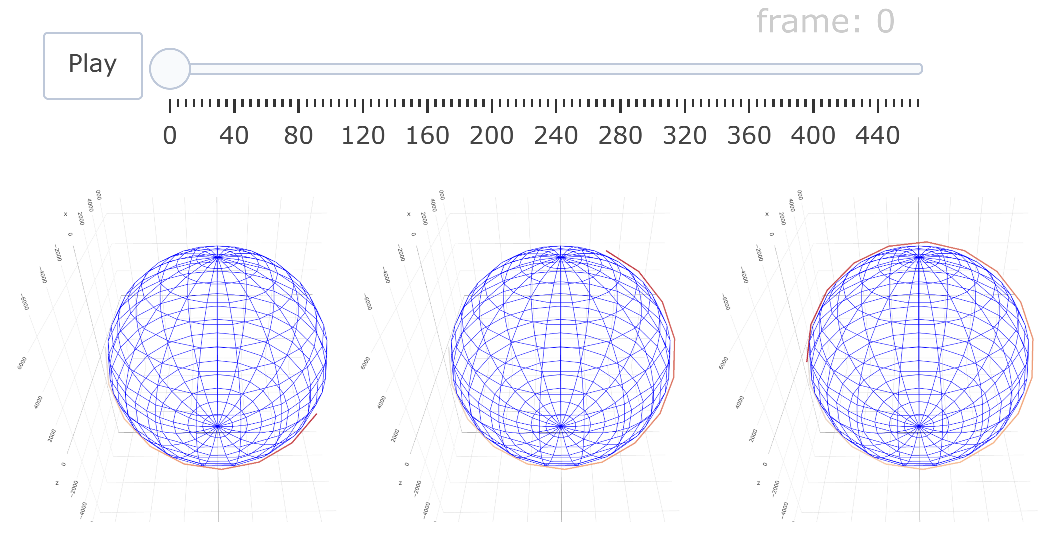


Figure 1: Trajectory of the International Space Station (ISS) calculated with the SGP4 propagator. The ISS is in a low Earth orbit, around 400 km above Earth's surface.

The SDP4 model

The SDP4 (Simplified Deep-space Perturbation 4) model (Hujsak 1979) is an extension of the SGP4 model designed to be applied to deep-space satellites, which are considered to be those with an orbital period over 225 minutes. The model introduces corrections to account for the gravitational effects exerted by the Sun and the Moon, as well as Earth gravitational resonance effects for orbits with periods of 12 h or 24 h. However, it employs a simplified atmospheric drag model, and therefore SGP4 should still be the preferred choice for near-Earth satellites.

The SDP4 implementation provided with **asteRisk** is also based on the revised C++ implementation by Vallado (Vallado). It can be accessed through the function sdp4, which takes the same arguments as input and returns an output in the same format as function sgp4. Alternatively, the function sgdp4 can be used to automatically determine and apply the most appropriate model based on the orbital period of the satellite (SGP4 for orbital periods below 225 minutes, and SDP4 for larger ones). The validity of our implementation has also been verified with previously published test cases (Appendix: Test cases for SGP4/SDP4).

In the following example, we demonstrate the application of the SDP4 model to propagate the orbit of the Molniya satellite, which follows a highly elliptical orbit with a period of approximately 12 h:

```
## Let us now propagate the position of a Molniya satellite, which should follow
## a highly elliptical orbit and go to distances much farther from Earth than
## the ISS. We will propagate the orbit for 715 minutes, one orbital period. In
## this case, we use the SDP4 propagator

molniya_TLE <- test_TLEs[[2]]
```

```
target_times_Molniya <- seq(0, 715, by=5)

results_position_matrix_Molniya <- matrix(nrow=length(target_times_Molniya), ncol=3)
results_velocity_matrix_Molniya <- matrix(nrow=length(target_times_Molniya), ncol=3)

for(i in 1:length(target_times_Molniya)) {
    new_result <- sdp4(n0=revDay2radMin(molniya_TLE$meanMotion),
                       e0=molniya_TLE$eccentricity,
                       i0=deg2rad(molniya_TLE$inclination),
                       M0=deg2rad(molniya_TLE$meanAnomaly),
                       omega0=deg2rad(molniya_TLE$perigeeArgument),
                       OMEGA0=deg2rad(molniya_TLE$ascension),
                       Bstar=molniya_TLE$Bstar,
                       initialDateTime=molniya_TLE$dateTime,
                       targetTime = target_times_Molniya[i])
    results_position_matrix_Molniya[i,] <- new_result[[1]]
    results_velocity_matrix_Molniya[i,] <- new_result[[2]]
}

results_position_matrix_Molniya = cbind(results_position_matrix_Molniya,
                                        target_times_Molniya)
colnames(results_position_matrix_Molniya) <- c("x", "y", "z", "time")

## We can follow a similar procedure as for the ISS to generate an animation
## of the trajectory of the Molniya satellite

accumulated_df_Molniya <- accumulate_by(
    as.data.frame(results_position_matrix_Molniya), ~time)

## We can then create a plotly animation

orbit_animation_Molniya <- plot_ly(accumulated_df_Molniya, x = ~x, y=~y, z=~z,
                                   type = "scatter3d", mode="lines+marker",
                                   opacity=0.8, line=list(width = 6,
                                                          color = ~time,
                                                          reverscale = FALSE),
                                   frame= ~frame, showlegend=FALSE)

sphere_df_ext_Molniya <- sphere_df[rep(seq_len(nrow(sphere_df)),
                                       length(target_times_Molniya)), ]
sphere_df_ext_Molniya <- cbind(sphere_df_ext_Molniya,
                               rep(target_times_Molniya, each=nrow(sphere_df)))
colnames(sphere_df_ext_Molniya) <- c("x", "y", "z", "frame")

sphere_animated_Molniya <- plot_ly(sphere_df_ext_Molniya, x=~x, y=~y, z=~z,
                                   frame=~frame, type="scatter3d", mode="lines",
                                   line=list(color='rgb(0,0,255)'),
                                   hoverinfo="skip", showlegend=FALSE)
sphere_animated_Molniya <- layout(sphere_animated_Molniya,
                                  scene = list(xaxis = list(showspikes=FALSE),
                                               yaxis = list(showspikes=FALSE),
                                               zaxis = list(showspikes=FALSE)))

combined_animation_Molniya <- suppressWarnings(subplot(sphere_animated_Molniya,
                                                       orbit_animation_Molniya))
combined_animation_Molniya <- animation_opts(combined_animation_Molniya, frame=15)
combined_animation_Molniya <- layout(combined_animation_Molniya, scene = list(
    aspectmode = "manual",
    aspectratio = list(x=1, y=(7000+24500)/(15000+21000),
                       z=(41000 + 7000)/(15000+21000)),
    xaxis=list(range=c(-21000, 15000)),
    yaxis=list(range=c(-24500, 7000)),
    zaxis=list(range=c(-7000, 41000))))
```

```
## We can now verify that the satellite follows a highly elliptical orbit. It can
## also be seen that, as expected, the satellite moves faster at the perigee
## (when it is closest to Earth) and slower at the apogee (when it is the
## farthest from Earth)

combined_animation_Molniya
```

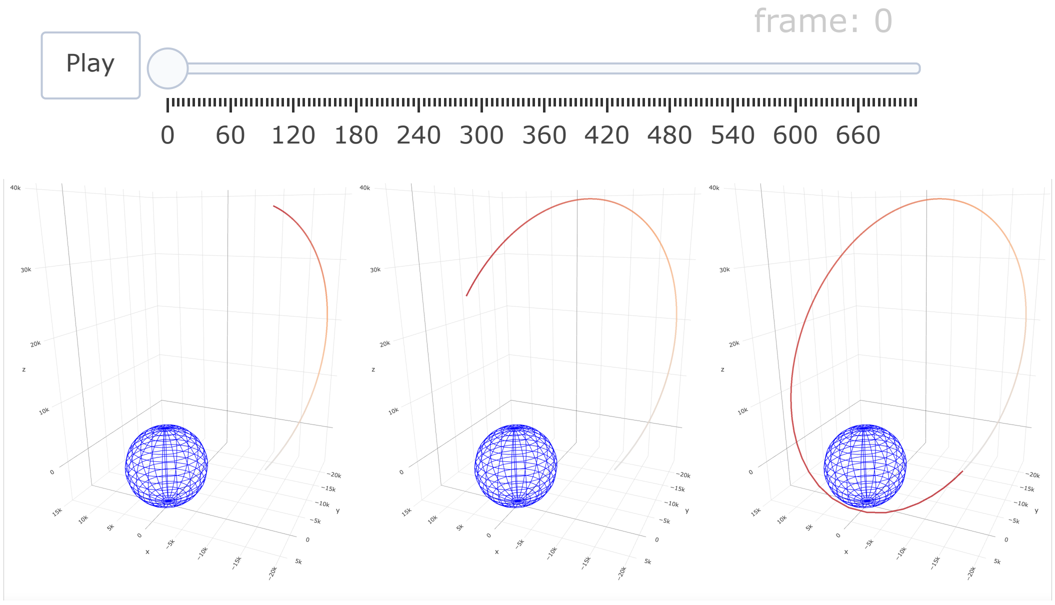


Figure 2: Trajectory of a Molniya satellite calculated with the SDP4 propagator. The satellite follows a highly elliptical orbit.

Numerical high-precision orbital propagator

asteRisk also provides a numerical orbital propagator. The implemented high-precision orbital propagator relies on the following fundamental motion equation:

$$\mathbf{a} = \frac{\partial^2 \mathbf{p}}{\partial t^2}$$

Where **a** denotes the acceleration, **p** the position, and t time. It can be simplified to a set of 6 first-order differential equations, where $p_x, p_y, p_z, v_x, v_y, v_z$ and a_x, a_y, a_z denote, respectively, orthogonal components of the position, velocity and acceleration:

$$v_x = \frac{\partial p_x}{\partial t}, v_y = \frac{\partial p_y}{\partial t}, v_z = \frac{\partial p_z}{\partial t}$$
$$a_x = \frac{\partial v_x}{\partial t}, a_y = \frac{\partial v_y}{\partial t}, a_z = \frac{\partial v_z}{\partial t}$$

In the current implementation, acceleration is calculated by considering the following forces known to play a significant role in orbital mechanics (Montenbruck and Gill 2012), comparable to the force field found in other HPOP implementations (Maisonobe et al. 2010; AGI; Mahooti):

- **Earth gravitational attraction.** The gravity field generated by Earth is calculated according to the GGM03S model (Tapley et al. 2007), using zonal, tesseral and sectoral spherical harmonics up to a degree and order of 180. The effects of ocean and solid Earth tides on the gravity field are also taken into account.
- **Third-body gravitational attractions.** The gravitational attractions exerted by the Sun, the Moon, the other 7 main planets of the Solar System (Mercury, Venus, Mars, Jupiter, Saturn, Uranus and Neptune) and Pluto are considered. All of them are modelled as point masses, and their positions are calculated using Jet Propulsion Laboratory Development Ephemeris 436 (Folkner et al. 2014).
- **Solar radiation pressure.** This is the force generated by the impact of photons from solar radiation on the satellite. A double conical shadow model to account for the reduction in the apparent solar disk seen by a satellite due to the Earth and the Moon (both of which are assumed to be spherical) eclipses is implemented (Li et al. 2019).
- **Atmospheric drag.** The drag effect caused by the atmosphere of Earth is calculated using the NRLMSISE-00 (Picone et al. 2002) atmospheric model.

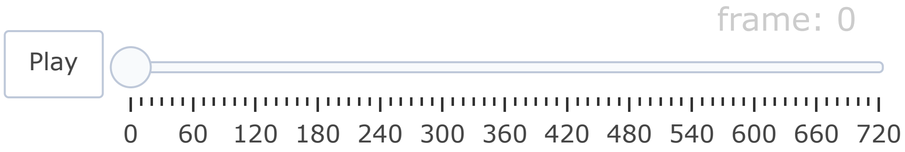
Additionally, a correction to the acceleration generated by the described forces to account for relativistic effects caused by the mass of the Earth is included (Montenbruck and Gill 2012). In the current implementation, only the Schwarzschild effect (Roh et al. 2016; Sośnica et al. 2021) is considered.

The HPOP can be accessed through the hpop function, which receives as inputs the initial position and velocity of the satellite, the time to which these correspond, a set of target times at which the position and velocity of the satellite should be calculated, the mass of the satellite, the effective area of the surface of the satellite subjected to drag and radiation pressure, and drag and reflectivity coefficients, that describe respectively how much the satellite is affected by drag and radiation pressure forces (Knocke et al. 1988).

The system of differential equations is solved with the deSolve package. By default, hpop uses the RADAU5 integrator, an implicit Runge-Kutta method of order 5 with adaptive step size (Wanner and Hairer 1996). However, additional arguments accepted by the ode function of the **deSolve** package can be directly provided to hpop to modify the values of the parameters for the integrator or to specify a different integrator that will be used to solve the motion equations.

In the following example, we apply the HPOP to the GPS satellite for which we previously read a RINEX navigation file, and compare the results with the real final position obtained by reading another RINEX file:

```
## We will use the ephemeris broadcasted in the previously read RINEX file
## for GPS satellite with PRN code 18 to obtain initial conditions for
## propagation. It should be noted that the HPOP requires Earth orientation
## parameters and other space data, which are provided through the companion
## asteRiskData package. Therefore, we must first install it if it is not
## already available:

if (!requireNamespace("asteRiskData", quietly = TRUE)) {
    install.packages('asteRiskData', repos='https://rafael-ayala.github.io/drat/')
}

## The initial position and velocity obtained from the RINEX file are in the
## ITRF frame, and must first be converted to the GCRF frame (more details
## about each frame are given in the following section). In order to perform
## such corrections, the latest Earth Orientation Parameters (which contain
## information about the precise location of the Earth rotation axis for every
## day) are required. These can be retrieved by running getLatestSpaceData(),
## after which they will be automatically used by all the functions that require
## them

GPS_RINEX_initialMessage <- GPS_RINEX$messages[[1]]

getLatestSpaceData()
initialEphemerisDateTime <- format(GPS_RINEX_initialMessage$ephemerisUTCTime,
                                   "%Y-%m-%d %H:%M:%EXS")
initialStateGCRF <- ITRFtoGCRF(GPS_RINEX_initialMessage$position_ITRF,
                              GPS_RINEX_initialMessage$velocity_ITRF,
                              initialEphemerisDateTime)

## Additionally, we also require some parameters describing the physical
## properties of the satellite. By November 2021, GPS satellite with PRN code
## 18 is USA-293, launched in August 2019 and belonging to GPS block III.
## Such satellites have an on-orbit mass of around 2600 kg. As an approximation,
## we can model GPS block III satellites as a central body with dimensions of
## 2.2 m x 1.8 m x 4.2 m with solar arrays of a total surface of 28.2 m2, as
## described by Steigenberger et al., 2020.  The central body therefore has
## faces with areas of 9.24 m2, 7.56 m2 and 3.96 m2. Without considering a more
## complex attitude model, the mean cross-sectional area of the satellite can
## be calculated as the sum of all these 4 areas divided by 2. We will use this
## value as the effective area for drag (which will anyway be negligible at the
## altitude of a GPS orbit). For the effective area for radiation pressure, we
## will use the mean cross-sectional area of the central body plus the full
## area of the solar arrays, since GPS block III satellites are equipped with
## systems to maintain them oriented towards the Sun. Finally, we will use the
## commonly employed values of 2.2 and 1.2 for drag and radiation coefficients.
## We will propagate the orbit for 12 hours at every minute.

propagationTimes <- seq(0, 43200, by=60)

## It should be noted that the following can take a few minutes to run

hpop_results_GPS <- hpop(position=initialStateGCRF$position,
                         velocity=initialStateGCRF$velocity,
                         dateTime=initialEphemerisDateTime,
                         times=propagationTimes,
                         satelliteMass=2600,
                         dragArea=(9.24 + 7.56 + 3.96 + 28.2)/2,
                         radiationArea=(9.24 + 7.56 + 3.96)/2 + 28.2,
                         dragCoefficient=2.2,
                         radiationCoefficient=1.2)

## We can now read another RINEX navigation file for the same satellite with
## the position 12 hours later, and compare it with the calculated one

GPS_RINEX_2 <- readGPSNavigationRINEX("./data/RINEX_GPS_2.rxn")

GPS_RINEX_endMessage <- GPS_RINEX_2$messages[[1]]
endEphemerisDateTime <- format(GPS_RINEX_endMessage$ephemerisUTCTime,
                              "%Y-%m-%d %H:%M:%EXS")
endStateGCRF <- ITRFtoGCRF(GPS_RINEX_endMessage$position_ITRF,
                           GPS_RINEX_endMessage$velocity_ITRF,
                           endEphemerisDateTime)

distance <- sqrt(
    sum(
        (endStateGCRF$position - hpop_results_GPS[nrow(hpop_results_GPS), 2:4])^2))
print(distance) # 19.7806

## The calculated position is less than 20 m away from the real position, thanks
## To the high precision of the propagator. We can also visualize the trajectory
## as previously with a plotly animation. To keep consistency, we first convert
## the distance units to km and propagation times to minutes

propagated_positions_GPS <- cbind(hpop_results_GPS[, 2:4]/1000,
                                  hpop_results_GPS[, 1]/60)
colnames(propagated_positions_GPS) <- c("x", "y", "z", "time")

## We will sample the trajectory once every 10 minutes

propagated_positions_GPS <- propagated_positions_GPS[
    seq(1, nrow(propagated_positions_GPS), by=10), ]
accumulated_df_GPS <- accumulate_by(as.data.frame(propagated_positions_GPS), ~time)

orbit_animation_GPS <- plot_ly(accumulated_df_GPS, x = ~x, y=~y, z=~z,
                               type = "scatter3d", mode="lines+marker",
                               opacity=0.8, line=list(width = 6,
                                                      color = ~time,
                                                      reverscale = FALSE),
                               frame= ~frame, showlegend=FALSE)

orbit_animation_GPS <- layout(orbit_animation_GPS, scene = list(
    xaxis=list(range=c(-27500, 27500)),
    yaxis=list(range=c(-27500, 27500)),
    zaxis=list(range=c(-27500, 27500))))

sphere_df_ext_GPS <- sphere_df[rep(seq_len(nrow(sphere_df)),
                                   length(propagated_positions_GPS[, "time"])), ]
sphere_df_ext_GPS <- cbind(sphere_df_ext_GPS,
                           rep(propagated_positions_GPS[, "time"],
                               each=nrow(sphere_df)))
colnames(sphere_df_ext_GPS) <- c("x", "y", "z", "frame")

sphere_animated_GPS <- plot_ly(sphere_df_ext_GPS, x=~x, y=~y, z=~z, frame=~frame,
                               type="scatter3d", mode="lines",
                               line=list(color='rgb(0,0,255)'), hoverinfo="skip",
                               showlegend=FALSE)
sphere_animated_GPS <- layout(sphere_animated_GPS,
                              scene = list(xaxis = list(showspikes=FALSE),
                                           yaxis = list(showspikes=FALSE),
                                           zaxis = list(showspikes=FALSE)))

combined_animation_GPS <- suppressWarnings(subplot(sphere_animated_GPS,
                                                   orbit_animation_GPS))
combined_animation_GPS <- animation_opts(combined_animation_GPS, frame=15)
combined_animation_GPS <- layout(combined_animation_GPS, scene = list(
    aspectmode = "cube"))
combined_animation_GPS
```



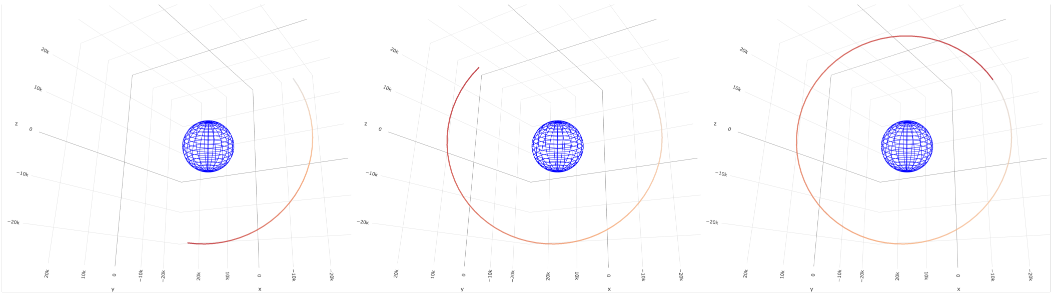


Figure 3: Trajectory of a GPS satellite calculated with the high-precision numerical propagator. It follows an almost circular orbit at an altitude of approximately 20200 km, which allows it to cover a larger portion of Earth at any given time.

4 Frames of reference

There are multiple frames of reference available to describe the position of objects in space. These differ in the location of the origin of coordinates, the presence or absence of acceleration for the reference frame itself and the orientation of the planes defining the main axes. Inertial frames are those that do not experience any acceleration, while non-inertial frames exhibit some sort of acceleration, such as rotation. The different available frames offer advantages and disadvantages for different applications. For example, a non-inertial frame that rotates with Earth is useful when analyzing the ground-track projected by a satellite onto the surface of Earth along its orbit, while inertial frames make the description of the forces acting on a satellite simpler.

asteRisk provides functions to convert between the frames of reference most relevant for the determination of Earth-orbiting satellites: ITRF (International Terrestrial Reference Frame), GCRF (Geocentric Celestial Reference Frame), TEME (True Equator Mean Equinox) and geodetic coordinates (latitude, longitude and altitude). Additionally, functions to obtain the orbital parameters of a satellite with known position and velocity in these frames are provided.

ITRF

The International Terrestrial Reference Frame (ITRF) (Cartography and Geodesy) is an Earth-Centered, Earth-Fixed (ECEF) frame of reference, i.e. a non-inertial Cartesian coordinate system that rotates with Earth. The origin of coordinates is placed at the center of mass of Earth, with the Z-axis extending along the true North as defined by the IERS reference pole (which differs slightly from the axis of rotation of Earth at a given instant due to its slight wobbling known as polar motion (Sandoval-Romero and Argueta-Diaz 2010)). The X-axis extends towards the intersection between the equator and the Greenwich meridian at any time (the point with 0º latitude and 0º longitude), and the Y-axis is set such that a right-handed orthogonal coordinate system is completed. As a consequence of these features, the position of an object fixed with respect to the surface of the Earth does not change with time.

GCRF

The Geocentric Celestial Reference Frame (GCRF) (Petit and Luzum 2010) is an inertial Cartesian coordinate system where the origin of coordinates is placed at the center of mass of the Earth and the axes are aligned with those of the International Celestial Reference System, which are defined based on the measurement of the position of extragalactic radio sources through very-long baseline interferometry. As a consequence, the axes are non-rotating, unlike in the ITRF frame. Even though GCRF, like any other body-centered frame of reference, is not a truly inertial frame of reference (given, for example, the fact that Earth itself is constantly orbiting around the Sun), it can be considered as such for Earth-orbiting satellites.

TEME

The True Equator Mean Equinox (TEME) frame is another Earth-centered inertial frame of reference used mainly by the SGP4/SDP4 orbital propagators to output coordinates. The origin of coordinates is also placed at the center of mass of the Earth, and the Z-axis extends towards the position of the Celestial Ephemeris Pole (CEP). The location of the CEP matches the mean instantaneous axis of rotation averaged over a period of 2 days (which makes the CEP ignore the diurnal and quasi-diurnal polar motions) (Capitaine et al. 1985). The X-axis extends towards the so-called uniform equinox, which is calculated by finding the intersection between the ecliptic (the plane defined by the orbit of the Earth around the Sun) and the mean equator (the plane containing Earth's equatorial line at a given time calculated by taking into consideration only precession, i.e., averaging over nutation effects), and then projecting this intersection (the mean equinox) over the true equator (the plane perpendicular to the previously defined Z-axis, which is therefore the equatorial plane at the same time calculated taking into consideration both precession and nutation) (Seago and Vallado 2000). Due to the complexity of its definition, the TEME frame is not currently widely used, except in the context of the results of propagating TLE elements with the SGP4/SDP4 models. It can also be inferred from its definition that the axes of the frame will slightly rotate over time as a consequence of the precession and nutation of Earth, and therefore it is important to know which time is taken as the reference when setting up the frame. While an explicit definition has not been given by NORAD, it is generally accepted that the reference elements (CEP, true equator and mean equinox) for the TEME frame are taken to be those at the time for which orbital propagation is performed (Seago and Vallado 2000; Vallado et al. 2006).

Orbital elements

Keplerian orbital elements are a set of six parameters used to describe the orbits of celestial objects, including satellites. While satellites do not follow a perfectly Keplerian orbit (i.e., a conic section that would be followed if only the point-like gravitational attraction between the satellite and the central body was considered), their state at any point can be defined by the orbital parameters that they would have if they were located at the same position with the same velocity following a perfectly Keplerian orbit (i.e., if perturbations caused by other forces were absent). These are called osculating orbital elements. Different parametrizations exist, with one of the most frequent sets of parameters being analogous to the mean orbital elements used by TLE previously described (it should be kept in mind that mean motion, semi major axis and orbital period provide the same information). Additional, alternative parameters exist for edge cases where some of the conventional ones are not well defined or meaningful. These are:

- **True anomaly**: unlike mean anomaly, true anomaly is the angle between the direction of the perigee and the actual position of the satellite.
- **Argument of latitude**: the angle between the equator and the position of the satellite. It is useful to define the position of satellites in circular orbits, where the argument of perigee and true anomaly are not well defined.
- **Longitude of perigee**: the angle between the vernal equinox and the perigee. It is useful for cases of orbits with 0 inclination, where the longitude of the ascending node and the argument of perigee are not well defined.
- **True longitude**: the angle between the vernal equinox and the position of the satellite. It is useful for cases of circular orbits with 0 inclination, where the longitude of the ascending node, the argument of perigee and true anomaly are not well defined.

It should also be noted that orbital elements are defined with respect to a set of elements of reference, such as the equatorial plane, which are usually chosen to define an Earth-centered inertial frame of reference. Although the specific choice of these must be taken into account when interpreting the meaning of a given set of orbital parameters, a complete set of six Keplerian orbital elements defines unequivocally the position and velocity of the satellite at a given time in a frame of reference with elements of reference equivalent to those used for the orbital elements. Therefore, osculating orbital elements can be used to calculate a corresponding set of Cartesian coordinates.

Conversion between different coordinate systems

The following table summarizes the functions available in **asteRisk** to convert coordinates between the different frames of reference previously described:

It should be noted that, while a velocity value is not strictly required to perform conversions between systems of Cartesian coordinates, it is required to convert to or from orbital elements.

In the following example, we apply multiple coordinate transformations to obtain a projection of the trajectory of the Molniya satellite previously analyzed over the surface of Earth:

```
## We previously calculated the trajectory of a Molniya satellite using the SDP4
## propagator. The output coordinates are given in the TEME frame. Let us first
## convert the final position to the standard GCRF frame of reference. Note that
## we need to add the propagation time to the original epoch specified in the
```

Table 2: Available functions for conversions between different systems of coordinates.

Origin system	Target system	Function
ITRF	GCRF	ITRFtoGCRF
ITRF	Geodetic	ITRFtoLATLON
GCRF	ITRF	GCRFtoITRF
GCRF	Geodetic	GCRFtoLATLON
GCRF	Orbital elements	ECItoKOE
TEME	GCRF	TEMEtoGCRF
TEME	ITRF	TEMEtoITRF
TEME	Geodetic	TEMEtoLATLON
TEME	Orbital elements	ECItoKOE
Geodetic	GCRF	LATLONtoGCRF
Geodetic	ITRF	LATLONtoITRF
Orbital elements	GCRF	KOEtoECI
Orbital elements	TEME	KOEtoECI

```
## TLE to obtain the correct date and time for conversion of reference frame

endState_Molniya_GCRF <- TEMEtoGCRF(results_position_matrix_Molniya[144, 1:3]*1000,
                                    results_velocity_matrix_Molniya[144, 1:3]*1000,
                                    as.character(as.POSIXct(
                                        molniya_TLE$dateTime, tz="UTC") +
                                            60*results_position_matrix_Molniya[144,4]))

print(endState_Molniya_GCRF$position)

#> [1] -14390694.30  -4572069.28    -95636.31

print(results_position_matrix_Molniya[144, 1:3]*1000)

#>           x           y           z
#> -14384056.0  -4592709.2   -104890.6

## The coordinates values are not wildly different, since differences are due to
## the precession and nutation of Earth's rotation axis.
## Let us know convert the coordinates to geodetic latitude and longitude to
## visualize a projection of the trajectory over the surface of Earth

geodetic_matrix_Molniya <- matrix(nrow=nrow(results_position_matrix_Molniya), ncol=4)

for(i in 1:nrow(geodetic_matrix_Molniya)) {
    new_dateTime <- as.character(as.POSIXct(molniya_TLE$dateTime, tz="UTC") +
                                     60*target_times_Molniya[i])
    new_geodetic <- TEMEtoLATLON(results_position_matrix_Molniya[i, 1:3]*1000,
                                 new_dateTime)
    geodetic_matrix_Molniya[i, 1:3] <- new_geodetic
    geodetic_matrix_Molniya[i, 4] <- target_times_Molniya[i]
}

colnames(geodetic_matrix_Molniya) <- c("latitude", "longitude", "altitude", "time")

## We can now visualize the ground track of the satellite with ggmap

groundTrack_Molniya <- ggmap(get_map(c(left=-180, right=180, bottom=-80, top=80),
                                     source = "stamen")) +
    geom_segment(data=as.data.frame(geodetic_matrix_Molniya),
                 aes(x=longitude, y=latitude,
                     xend=c(tail(longitude, n=-1), NA),
                     yend=c(tail(latitude, n=-1), NA)),
                 na.rm=TRUE) +
   geom_point(data=as.data.frame(geodetic_matrix_Molniya), aes(x=longitude, y=latitude),
              color="blue", size=0.3, alpha=0.8)

## As we can see, Molniya satellites spend most of the time over high latitudes,
## thanks to the fact that the apogee of their elliptical orbit is above such
## regions. This property made them useful for the Soviet Union.

groundTrack_Molniya
```

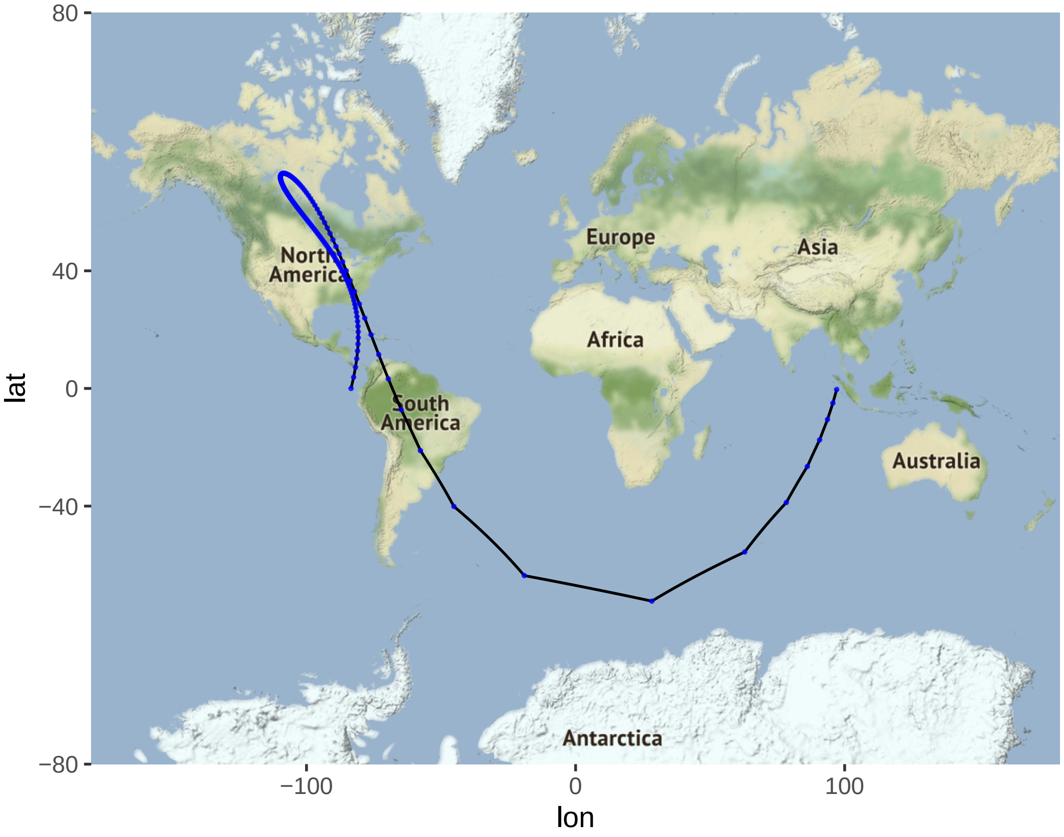


Figure 4: Ground track of a Molniya satellite. The satellite spent most of the time over regions of high latitude.

5 Future work

While the described functionalities provide a solid core for astrodynamics analysis in R, additional features are expected to be frequently added to **asteRisk**. Some of these will include:

- Expansion of the force model used by the HPOP to increase its accuracy. For example, by including albedo radiation pressure or considering the additional Lens-Thirring and De Sitter relativistic effects.
- Addition of functionalities to model and calculate orbital maneuvers.
- Support for more complex models for satellite orientation during propagation with the HPOP.
- Adding support for the Orbit Data Messages standard formats (Space Data Systems).
- Addition of more reference frames, including non-Earth centered frames.

6 Summary

Accurate calculation of the position of satellites at a given time is a key part of the design of successful space missions and continued operation and maintenance of satellites already in orbit. **asteRisk** provides tools to perform such astrodynamics analyses in R. These include well-documented orbital propagators and functions to parse the file formats most commonly used to distribute satellite positional data, as well as to convert coordinates between different coordinate frames. The provided toolbox should be specially useful in combination with the publicly available orbital data sources and the multiple statistical analysis and machine-learning functionalities already available in the R ecosystem, facilitating the development of novel orbital propagators with higher accuracy and lower computational costs.

7 Appendix: Test cases for SGP4/SDP4

The following code will run SGP4/SDP4 propagators on a series of standard test frequently used to test the validity of implementations of these models (Vallado et al. 2006). The results demonstrate that the implementation provided in **asteRisk** is in agreement up to at least tenths of millimeters with previous implementations.

```
## First, we create a list with the target propagation times for each test case,
## in the same order as the TLEs are given

target_times_verification <- list(
    "5"=seq(0, 4320, by=360),
    "4632"=c(0, -5184, -5064, -4944, -4896),
    "6251"=seq(0, 2880, by=120),
    "8195"=seq(0, 2880, by=120),
    "9880"=seq(0, 2880, by=120),
    "9998"=c(0, seq(-1440, -720, by=60)),
    "11801"=seq(0, 1440, by=720),
    "14128"=seq(0, 2880, by=120),
    "16925"=seq(0, 1440, by=120),
    "20413"=c(0, seq(1440, 4320, by=120)),
    "21897"=seq(0, 2880, by=120),
    "22312"=c(0, seq(54.2028672, 474.2028672, by=20)),
    "22674"=seq(0, 2880, by=120),
    "23177"=seq(0, 1440, by=120),
    "23333"=c(seq(0, 1560, by=120), 1600),
    "23599"=seq(0, 720, by=20),
    "24208"=seq(0, 1440, by=120),
    "25954"=c(0, seq(-1440, 1440, by=120)),
    "26900"=c(0, 9300, 9360, 9400),
    "26975"=seq(0, 2880, by=120),
    "28057"=seq(0, 2880, by=120),
    "28129"=seq(0, 1440, by=120),
    "28350"=seq(0, 1440, by=120),
    "28623"=seq(0, 1440, by=120),
    "28626"=seq(0, 1440, by=120),
    "28872"=seq(0, 50, by=5),
    "29141"=seq(0, 420, by=20),
    "29238"=seq(0, 1440, by=120),
    "88888"=seq(0, 1440, by=120),
    "33333"=seq(0, 20, by=5),
    "33334"=0,
    "33335"=seq(0, 1440, by=20),
    "20413"=c(0, seq(1844000, 1844340, by=5))
)

## We can now read the TLE file with the TLEs for all verification cases and
## propagate them at the target times. We can store the results for each case
## as a matrix of 7 columns where each row corresponds to a propagation time,
## column 1 stores propagation times, columns 2 to 4 the propagated positions
## and columns 5 to 7 the propagated velocities

verification_TLEs <- readTLE("data/verificationTLEs.tle")
verification_results <- vector(mode="list", length=length(verification_TLEs))
names(verification_results) <- names(target_times_verification)

for(i in 1:length(verification_TLEs)) {
    verification_results[[i]] <- matrix(nrow=length(target_times_verification[[i]]),
                                        ncol=7)
    for(j in 1:length(target_times_verification[[i]])) {
        propagation <- sgdp4(n0=revDay2radMin(verification_TLEs[[i]]$meanMotion),
                             e0=verification_TLEs[[i]]$eccentricity,
                             i0=deg2rad(verification_TLEs[[i]]$inclination),
                             M0=deg2rad(verification_TLEs[[i]]$meanAnomaly),
                                   omega0=deg2rad(verification_TLEs[[i]]$perigeeArgument),
                                   OMEGA0=deg2rad(verification_TLEs[[i]]$ascension),
                                   Bstar=verification_TLEs[[i]]$Bstar,
                                   initialDateTime=verification_TLEs[[i]]$dateTime,
                                   targetTime=target_times_verification[[i]][j])
            verification_results[[i]][j, 1] <- target_times_verification[[i]][j]
            verification_results[[i]][j, 2:4] <- propagation$position
            verification_results[[i]][j, 5:7] <- propagation$velocity
    }
}
```

8 Acknowledgements

The development of this work is supported by the following grants: a KAKENHI Grant-in-Aid for Research Activity Start-up Grant Number 21K20645 to Rafael Ayala, a JSPS Postdoctoral Fellowship for Research in Japan (Standard) Grant Number P20810 to Lara Sellés Vidal (Overseas researcher under Postdoctoral Fellowship of Japan Society for the Promotion of Science), and grants by the Spanish Ministry of Science and Innovation (grant code PID2019-105471RB-I00) and the Regional Government of Andalusia (grant code P18-RT-1060) to David Ruiz.

References

AGI. Systems tool kit (STK).,URL https://www.agi.com/products/stk. (accessed: 24.11.2021).

J. Bouwmeester and J. Guo. Survey of worldwide pico-and nanosatellite missions, distributions and subsystem technology. *Acta Astronautica*, 67(7-8): 854–862, 2010.

N. Capitaine, J. Williams and P. Seidelmann. Clarifications concerning the definition and determination of the celestial ephemeris pole. *Astronomy and Astrophysics*, 146: 381–383, 1985.

F. A. for Cartography and Geodesy. The international terrestrial reference frame (ITRF).,URL https://www.iers.org/IERS/EN/DataProducts/ITRF/itrf.html. (accessed: 26.11.2021).

A. Deprit. The elimination of the parallax in satellite theory. *Celestial Mechanics*, 24(2): 111–153, 1981.

W. Dong and Z. Chang-yin. An accuracy analysis of the SGP4/SDP4 model. *Chinese Astronomy and Astrophysics*, 34(1): 69–76, 2010.

A. Flores. GPS interface specification.,URL https://www.gps.gov/technical/icwg/IS-GPS-200M.pdf. (accessed: 26.11.2021).

R. Flores, B. M. Burhani and E. Fantino. A method for accurate and efficient propagation of satellite orbits: A case study for a molniya orbit. *Alexandria Engineering Journal*, 60(2): 2661–2676, 2021.

W. M. Folkner, J. G. Williams, D. H. Boggs, R. S. Park and P. Kuchynka. The planetary and lunar ephemerides DE430 and DE431. *Interplanetary Network Progress Report*, 196(1): 2014.

C. Foster, H. Hallam and J. Mason. Orbit determination and differential-drag control of planet labs CubeSat constellations. *arXiv preprint arXiv:1509.03270*, 2015.

W. Gurtner and L. Estey. RINEX - the receiver independent exchange format-version 3.00. *Astronomical Institute, University of Bern and UNAVCO, Bolulder, Colorado.*, 2007.

F. R. Hoots, R. L. Roehrich and T. S. Kelso. Spacetrack report no. 3., 1988. URL https://celestrak.com/NORAD/documentation/spacetrk.pdf. (accessed: 07.10.2021).

R. S. Hujsak. A restricted four body solution for resonating satellite with an oblate earth. In *American institute of aeronautics and astronautics conference*, 1979.

S. W. Janson, E. Jaime, M. Sergio, L. R. Hissa, R. A. de Carvalho, W. H. Steyn, V. J. Lappas, F. T. Nardini, C. Michele, R. Alexander, et al. Nanosatellites: Space and ground technologies, operations and economics. 2020.

T. S. Kelso. CelesTrack.,URL https://celestrak.com/. (accessed: 16.09.2021).

T. S. Kelso. Frequently asked questions: Two-line element set format.,URL http://www.celestrak.com/columns/v04n03/. (accessed: 07.10.2021).

T. S. Kelso. NORAD two-line element set format.,URL http://www.celestrak.com/NORAD/documentation/ADCOM/%20DO/%20Form/%2012.pdf. (accessed: 07.10.2021).

D. J. Kessler and B. G. Cour-Palais. Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research: Space Physics*, 83(A6): 2637–2646, 1978.

D. J. Kessler, N. L. Johnson, J. Liou and M. Matney. The kessler syndrome: Implications to future space operations. *Advances in the Astronautical Sciences*, 137(8): 2010, 2010.

A. Kirillin, I. Belokonov, I. Timbai, A. Kramlikh, M. Melnik, E. Ustiugov, A. Egorov and S. Shafran. SSAU nanosatellite project for the navigation and control technologies demonstration. *Procedia Engineering*, 104: 97–106, 2015.

P. Knocke, J. Ries and B. Tapley. Earth radiation pressure effects on satellites. In *Astrodynamics conference*, page. 4292 1988.

M. Lane. The development of an artificial satellite theory using a power-law atmospheric density representation. In *2nd aerospace sciences meeting*, page. 35 1965.

M. H. Lane and F. R. Hoots. General perturbations theories derived from the 1965 lane drag theory. AEROSPACE DEFENSE COMMAND PETERSON AFB CO OFFICE OF ASTRODYNAMICS. 1979.

M. Lara, J. F. San-Juan, D. Hautesserres and C. de Competence Technique. A semi-analytical orbit propagator program for highly elliptical orbits. *Order*, 10: 8, 2016.

M. Lara, J. F. San-Juan and L. M. López-Ochoa. Delaunay variables approach to the elimination of the perigee in artificial satellite theory. *Celestial Mechanics and Dynamical Astronomy*, 120(1): 39–56, 2014.

H. Lévy, É. Joffre, S. Lizy-Destrez and M. Zamaro. STORM: A semi-analytical orbit propagator for assessing the compliance with mars planetary protection requirements. *Acta Astronautica*, 2021.

Z. Li, M. Ziebart, S. Bhattarai and D. Harrison. A shadow function model based on perspective projection and atmospheric effect for satellites in eclipse. *Advances in Space Research*, 63(3): 1347–1359, 2019.

M. Mahooti. High precision orbit propagator matlab implementation.,URL https://jp.mathworks.com/matlabcentral/fileexchange/55167-high-precision-orbit-propagator. (accessed: 24.11.2021).

L. Maisonobe, V. Pommier and P. Parraud. Orekit: An open source library for operational flight dynamics applications. In *4th international conference on astrodynamics tools and techniques*, pages. 3–6 2010.

J. Meeus. Astronomical algorithms. *Richmond*, 1991.

O. Montenbruck and E. Gill. *Satellite orbits: Models, methods and applications.* Springer Science & Business Media, 2012.

V. Morand, J. C. Dolado-Perez, H. Fraysse, F. Delefie, J. Daquin and C. Dental. Semi analytical computation of partial derivatives and transition matrix using STELA software. In *6th ESA conference on space debris*, 2013.

C. E. Noll. The crustal dynamics data information system: A resource to support scientific analysis using space geodesy. *Advances in Space Research*, 45(12): 1421–1440, 2010.

G. Petit and B. Luzum. IERS conventions (2010). Bureau International des Poids et mesures sevres (france). 2010.

J. Picone, A. Hedin, D. P. Drob and A. Aikin. NRLMSISE-00 empirical model of the atmosphere: Statistical comparisons and scientific issues. *Journal of Geophysical Research: Space Physics*, 107(A12): SIA–15, 2002.

K.-M. Roh, S. M. Kopeikin and J.-H. Cho. Numerical simulation of the post-newtonian equations of motion for the near earth satellite with an application to the LARES satellite. *Advances in Space Research*, 58(11): 2255–2268, 2016.

SAIC. Space-Track.,URL https://www.space-track.org/. (accessed: 16.09.2021).

G. E. Sandoval-Romero and V. Argueta-Diaz. A simple theoretical comparison between two basic schemes in function of the earth's north pole detection: The static method. *Journal of Sensors*, 2010: 2010.

M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic and M. Wilhelm. Bringing up OpenSky: A large-scale ADS-B sensor network for research. In *IPSN-14 proceedings of the 13th international symposium on information processing in sensor networks*, pages. 83–94 2014. URL https://doi.org/10.1109/IPSN.2014.6846743.

J. Seago and D. Vallado. Coordinate frames of the US space object catalogs. In *Astrodynamics specialist conference*, page. 4025 2000.

K. Sośnica, G. Bury, R. Zajdel, K. Kazmierski, J. Ventura-Traveset, R. Prieto-Cerdeira and L. Mendes. General relativistic effects acting on the orbits of galileo satellites. *Celestial Mechanics and Dynamical Astronomy*, 133(4): 1–31, 2021.

C. C. for Space Data Systems. Orbit data messages - recommended standard.,URL https://public.ccsds.org/Pubs/502x0b2c1e2.pdf. (accessed: 26.11.2021).

R. I. of Space Device Engineering. Global navigation satellite system - interface control document.,URL https://russianspacesystems.ru/wp-content/uploads/2016/08/ICD_GLONASS_eng_v5.1.pdf. (accessed: 26.11.2021).

B. Tapley, J. Ries, S. Bettadpur, D. Chambers, M. Cheng, F. Condi and S. Poole. The GGM03 mean earth gravity model from GRACE. In *AGU fall meeting abstracts*, pages. G42A–03 2007.

A. Toorian, K. Diaz and S. Lee. The CubeSat approach to space access. In *2008 IEEE aerospace conference*, pages. 1–14 2008. IEEE.

United Nations. Register of Objects Launched into Outer Space.,URL https://www.unoosa.org/oosa/en/spaceobjectregister/index.html. (accessed: 15.09.2021).

D. Vallado. Astrodynamics software.,URL https://celestrak.com/software/vallado-sw.php. (accessed: 07.10.2021).

D. A. Vallado, P. Crawford, R. Hujsak and T. Kelso. Revisiting spacetrack report# 3: Rev 1. In

AIAA/AAS astrodynamics specialist conference and exhibit, 2006.

G. Wanner and E. Hairer. *Solving ordinary differential equations II*. Springer Berlin Heidelberg, 1996.

Rafael Ayala
Okinawa Institute of Science and Technology Graduate University
7542 Onna, Onna-son, Kunigami, Okinawa 904-0411
Japan
ORCiD: 0000-0002-9332-4623
rafael.ayala@oist.jp

Daniel Ayala
University of Seville
ETSI Informática, Avda. de la Reina Mercedes, s/n, Sevilla E-41012
Spain
ORCiD: 0000-0003-2095-1009
dayala1@us.es

Lara Sellés Vidal
Okinawa Institute of Science and Technology Graduate University
7542 Onna, Onna-son, Kunigami, Okinawa 904-0411
Japan
ORCiD: 0000-0003-2537-6824
lara.sellesvidal@oist.jp

David Ruiz
University of Seville
ETSI Informática, Avda. de la Reina Mercedes, s/n, Sevilla E-41012
Spain
ORCiD: 0000-0003-4460-5493
druiz@us.es

gplsim: An R Package for Generalized Partially Linear Single-index Models

by Tianhai Zu and Yan Yu

Abstract Generalized partially linear single-index models (GPLSIMs) are important tools in nonparametric regression. They extend popular generalized linear models to allow flexible nonlinear dependence on some predictors while overcoming the "curse of dimensionality." We develop an R package **gplsim** that implements efficient spline estimation of GPLSIMs, proposed by Yu and Ruppert (2002) and Yu et al. (2017), for a response variable from a general exponential family. The package builds upon the popular **mgcv** package for generalized additive models (GAMs) and provides functions that allow users to fit GPLSIMs with various link functions, select smoothing tuning parameter λ against generalized cross-validation or alternative choices, and visualize the estimated unknown univariate function of single-index term. In this paper, we discuss the implementation of **gplsim** in detail, and illustrate the use case through a sine-bump simulation study with various links and a real-data application to air pollution data.

1 Introduction

A popular approach to analyzing the relationship between a response variable and a set of predictors is generalized linear models or GLMs McCullagh and Nelder (1989), where the conditional mean of the response variable is linked to a linear combination of predictors via a link function. Although GLMs are simple and easy to interpret, in many complex real data applications the underlying assumption of linearity is often violated. Generalized partially linear single-index models (GPLSIMs) (e.g. Carroll et al., 1997; Yu and Ruppert, 2002; Yu et al., 2017) are flexible semiparametric models that allow for a non-linear relationship while retaining ease of interpretation. In particular, GPLSIMs include a partial linear component $\mathbf{z}\gamma$, and importantly a nonparametric single-index component, effectively reducing the dimensionality of p-dimensional predictors $\mathbf{x}$ to a univariate single index $\mathbf{x}^T\boldsymbol{\theta}$ with a flexible univariate function $\phi(\mathbf{x}^T\boldsymbol{\theta})$, avoiding the "curse of dimensionality" in multivariate nonparametric regression. GPLSIMs reduce to popular single-index models (Ichimura, 1993; Härdle et al., 1993; Xia and Härdle, 2006) when there are no partial linear terms. Another popular special case is partially linear models Härdle et al. (2012) when there is only one predictor in the nonparametric component.

GPLSIMs and the reduced models have been studied extensively in the literature. Applications lie in various fields, for example, discrete choice analysis, dose-response models, credit scoring, Framingham heart study (Yu et al., 2017, and references therein). Yu and Ruppert (2002), Xia and Härdle (2006), and Liang et al. (2010) studied partially linear single-index models for continuous responses. For responses from a general exponential family, Carroll et al. (1997) proposed local linear approach via quasi-likelihood for GPLSIMs estimation. However, as noted in Yu and Ruppert (2002), the algorithm using local linear methods in Carroll et al. (1997) may suffer from some computational issues and can become unstable. Yu and Ruppert (2002) proposed a stable and computationally expedient approach using penalized splines (P-splines) with non-linear least square minimization. Yu et al. (2017) further proposed an efficient profile likelihood algorithm for the P-splines approach to GPLSIMs.

We develop a package **gplsim**[1] in R (R Core Team, 2021) using splines for efficient estimation of the unknown univariate function in GPLSIMs following Yu and Ruppert (2002) and Yu et al. (2017). The **gplsim** R package mainly implements the profile likelihood estimation method described in Yu et al. (2017), utilizing the function `gam` (Wood, 2011) from the state-of-the-art R package **mgcv** (Wood, 2001). The model class `gplsim` extends on the `gam` for straightforward computation and implementation. A side benefit is that our **gplsim** package enjoys improvements and features as those made to the **mgcv** package. For example, **mgcv 1.5** added smoothness selection method "REML" and "ML" in addition to "GCV" to its core function "gam()", and **gplsim** can enjoy those new features naturally. Similarly, any spline basis adopted in **mgcv** package, such as the thin plate regression spline basis, is also available in our **gplsim** package. In addition, our **gplsim** package also implements the simultaneous non-linear least square minimization methods for continuous responses from Yu and Ruppert (2002) as an alternative option.

The **mgcv** package for generalized additive models (GAMs) is indeed a fundamental building block for our **gplsim** package. GAMs (Hastie and Tibshirani, 1986; Wood, 2011) are popular semiparametric models, which replaces the single-index components by summation of individual smooth functions.

[1] The package has been published at CRAN, and it is hosted at the package maintainer's public GitHub repository `github.com/zzz1990771/gplsim`.

As noted in Carroll et al. (1997) and Yu and Ruppert (2002), GPLSIMs are more parsimonious and can model some interactions. However, GPLSIMs are nonlinear and more difficult to estimate, especially given the widely available software for GAMs. One may view GAMs as a special case of GPLSIMs when the single-index coefficients are known. Alternatively, one may view GPLSIMs as special GAMs models with a nonlinear single-index effect. Single-index models can also be viewed as the base of more complex models, such as multi-index models (Xia, 2008), projection pursuit regression (Hall, 1989) and deep neural networks (Yang et al., 2017).

The rest of the paper is organized as follows. In the next section, we review the GPLSIMs and the penalized spline estimation for GPLSIMs. Next, we discuss the estimation algorithm implemented in this package. The following section describes the main features of the functions provided. The section "real data and simulation examples" illustrates the use of **gplsim** in R via an air pollution example and a sine-bump simulation study. The last section summarizes the method and presents directions for future research and application.

2 An overview of generalized partially linear single-index models

The GPLSIMs

For given predictor vectors of p-dimensional $X = \mathbf{x}$ and q-dimensional $Z = \mathbf{z}$, and under the assumption that the conditional density of the response variable Y arises from a general exponential family, the conditional mean $E(Y|\mathbf{x},\mathbf{z})$ can be modeled by

$$E(Y|\mathbf{x},\mathbf{z}) := \mu(\mathbf{x},\mathbf{z}) = g^{-1}\{\phi\left(\mathbf{x}^T\boldsymbol{\theta}\right) + \mathbf{z}\gamma\}, \tag{1}$$

where the single-index parameter $\boldsymbol{\theta}$ maps the p-dimensional predictors $\mathbf{x}$ to a univariate single index $\mathbf{x}^T\boldsymbol{\theta}$ by a linear projection, and $\phi(\cdot)$ is a univariate unknown function, while $g\{\cdot\}$ is a known link function. The parameter vector is constrained such that $\|\boldsymbol{\theta}\| = 1$ with first element θ_1 positive for identifiability (Yu and Ruppert, 2002).

One of the main challenges to estimate model (1) is that the p-dimensional single-index parameter $\boldsymbol{\theta}$ is nested within the unknown univariate function $\phi(\cdot)$, and hence a highly nonlinear problem.

Review of penalized spline estimation for GPLSIMs

When the single-index parameter $\boldsymbol{\theta}$ or the single-index $u = \mathbf{x}^T\boldsymbol{\theta}$ is given, we can estimate the unknown univariate function $\phi(\cdot)$ with penalized splines (Ruppert et al., 2003) such that $\phi(u) \approx \mathbf{H}(u)\boldsymbol{\beta}$. The systematic component of GPLSIMs can then be approximated by

$$g\{\mu(\mathbf{x},\mathbf{z})\} = \mathbf{H}(\mathbf{x}^T\boldsymbol{\theta})\boldsymbol{\beta} + \mathbf{z}\gamma, \tag{2}$$

where $\mathbf{H}(\cdot)$ is the spline basis, and $\boldsymbol{\beta}$ is the spline coefficient vector. We denote $\boldsymbol{\omega} = (\boldsymbol{\theta}, \boldsymbol{\beta}, \gamma)$ as the column parameter vector.

There are many choices of the spline basis functions $\mathbf{H}(\cdot)$, such as B-spline, truncated power basis, thin-plate spline, and their variations. For simplicity, we first illustrate using a truncated power basis of degree d:

$$\mathbf{H}(u)\boldsymbol{\beta} = \beta_0 + \beta_1 u + \cdots + \beta_d u^d + \sum_{k=1}^{K} \beta_{d+k}\left(u - v_k\right)_+^d,$$

where $\mathbf{H}(u) = \left\{1, u, \ldots, u^d, (u-v_1)_+^d, \ldots, (u-v_K)_+^d\right\}$ are spline bases with K interior knots placed at $(v_1, \ldots, v_K)$. Quadratic or cubic splines are commonly used. The interior knots are usually placed at equally-spaced quantiles within the domain.

Another popular choice of spline basis is the B-spline basis. Any B-spline basis functions $\mathbf{H}(\cdot)$ of degree higher than 0, can be defined by the following Cox-de Boor recursion formula (Boor, 2001):

$$\mathrm{H}_{k,\mathrm{d}}(u) = \frac{u - u_k}{u_{k+d-1} - u_k}\mathrm{H}_{k,d-1}(u) + \frac{u_{k+d} - u}{u_{k+d} - u_{k+1}}\mathrm{H}_{k+1,d-1}(u),$$

where

$$\mathrm{H}_{k,0}(u) = \begin{cases} 1, & \mathrm{u}_k \leq u \leq u_{k+1} \\ 0, & \text{otherwise.} \end{cases}$$

One of the appealing features of the B-spline is that, unlike truncated power basis, B-spline basis functions have local supports that can result in high numerical stability.

To avoid overfitting, a roughness penalty controlled by a smoothing parameter λ is applied to the log-likelihood. Specifically, we can obtain the penalized log-likelihood estimator of $\boldsymbol{\omega}$ by maximizing the following penalized log-likelihood function:

$$\begin{aligned} Q_{n,\lambda}(\boldsymbol{\omega}) =& \frac{1}{n}L_n(\boldsymbol{\omega}) - \frac{1}{2}\lambda\boldsymbol{\beta}^\top\mathbf{D}\boldsymbol{\beta} \\ =& \frac{1}{n}\sum_{i=1}^{n}\left[y_i\xi\left(\mathbf{x}_i,\mathbf{z}_i;\boldsymbol{\omega}\right) - b\left\{\xi\left(\mathbf{x}_i,\mathbf{z}_i;\boldsymbol{\omega}\right)\right\}\right] - \frac{1}{2}\lambda\boldsymbol{\beta}^\top\mathbf{D}\boldsymbol{\beta}, \end{aligned} \tag{3}$$

where ξ is the natural parameter in generalized linear models, $\mu(\mathbf{x}_i,\mathbf{z}_i) = b'\left\{\xi\left(\mathbf{x}_i,\mathbf{z}_i;\boldsymbol{\omega}\right)\right\}$, for observations $i = 1,\cdots,n$, and $\mathbf{D}$ is a positive semidefinite symmetric penalty matrix. Common penalty matrices include the usual quadratic integral penalty on second derivatives of $\phi(\cdot)$ or alternatively the diagonal penalty matrix with its last K diagonal elements constrained to equal to one and the rest equal to zero (see e.g. Ruppert and Carroll, 2000; Yu and Ruppert, 2002), which in effect penalizes the coefficients of the truncated power basis at the jump of d-th derivatives.

Maximizing the penalized log-likelihood function (3) can be achieved in several ways. We mainly focus on implementing an efficient profile log-likelihood method in Yu et al. (2017). We also present an option to implement a simultaneous nonlinear least square method in Yu and Ruppert (2002).

The selection of smoothing parameter λ is important as it controls the tradeoff between over-smoothing (possible underfitting) and under-smoothing (possible overfitting). We use an outer iteration to select λ against some selection criterion, as recommended by Wood (2011). For the default choice, we adopt generalized cross validation (GCV) to select the smoothing parameter λ. Alternatively, we can consider maximum likelihood (ML) (Anderssen and Bloomfield, 1974) or restricted maximum likelihood (REML) (Wahba, 1985) based approaches. A nice feature is that we can directly adopt criteria that have been provided by the "gam()" function arguments from R package **mgcv**, which is one of the main components in the implementation of our `gplsim` estimation algorithm.

Algorithm

We present the main algorithm for fitting the generalized partially linear single-index models (GPLSIMs) with penalized splines estimation with profile likelihood in detail as follows:

Input: Non-linear predictor vector of p-dimensional $X = \mathbf{x}$, partially linear predictor vector of q-dimensional $Z = \mathbf{z}$, and a response vector $Y = \mathbf{y}$ of family=family.

Output: The estimated single-index parameter $\widehat{\boldsymbol{\theta}}$, spline coefficient $\widehat{\boldsymbol{\beta}}$, partially linear coefficient $\widehat{\boldsymbol{\gamma}}$, and fitted response $\widehat{\mathbf{y}}$.

1 Obtain an initial estimate $\hat{\boldsymbol{\theta}}^{(0)}$ of the single-index parameter $\boldsymbol{\theta}$ from a generalized linear model (default), or a user-provided initial list.

2 With an estimate of $\boldsymbol{\theta}$ (equivalently, the single index $\left\{u_i = \mathbf{x}_i^T\boldsymbol{\theta} : i = 1,\ldots,n\right\}$), the spline coefficient $\boldsymbol{\beta}$ and partially linear coefficient $\boldsymbol{\gamma}$ can be written as implicit functions of $\boldsymbol{\theta}$ to maximize penalized log-likelihood:

$$\begin{aligned} &Q\left(\boldsymbol{\beta},\boldsymbol{\gamma},\lambda;u_1,\ldots,u_n\right) \\ &= \tfrac{1}{n}\textstyle\sum_{i=1}^{n}\left[y_i\xi\left(u_i,\mathbf{z}_i;\boldsymbol{\beta},\boldsymbol{\gamma}\right) - b\left\{\xi\left(u_i,\mathbf{z}_i;\boldsymbol{\beta},\boldsymbol{\gamma}\right)\right\}\right] \\ &-\tfrac{1}{2}\lambda\boldsymbol{\beta}^\top\mathbf{D}\boldsymbol{\beta}. \end{aligned}$$

The roughness penalty parameter λ is selected using generalized cross-validation score (default) or alternative options.

3 Given the spline coefficient vector $\widehat{\boldsymbol{\beta}}_\lambda(\boldsymbol{\theta})$ and partially linear coefficient vector $\widehat{\gamma}_\lambda(\boldsymbol{\theta})$ as implicit functions of $\boldsymbol{\theta}$, obtain the profile log-likelihood estimator of the single-index parameter $\boldsymbol{\theta}$ by maximizing:

$$\begin{aligned} Q(\boldsymbol{\theta}) =& \frac{1}{n}\sum_{i=1}^{n}\left[y_i\xi\left(\mathbf{x}_i^T\boldsymbol{\theta},\mathbf{z}_i;\widehat{\boldsymbol{\beta}}_\lambda(\boldsymbol{\theta}),\widehat{\gamma}_\lambda(\boldsymbol{\theta})\right)\right. \\ &\left. -b\left\{\xi\left(\mathbf{x}_i^T\boldsymbol{\theta},\mathbf{z}_i;\widehat{\boldsymbol{\beta}}_\lambda(\boldsymbol{\theta}),\widehat{\gamma}_\lambda(\boldsymbol{\theta})\right)\right\}\right]. \end{aligned}$$

4 With the estimated profile log-likelihood estimator $\widehat{\boldsymbol{\theta}}$ of the single-index parameter, obtain the final estimator $\widehat{\boldsymbol{\beta}}$ of spline coefficient and $\widehat{\gamma}$ of partially linear coefficient via step 2.

5 Obtain the final fitted response vector $\widehat{\mathbf{y}}$ from model (1).

Alternatively, for continuous responses under the default assumption of $family = gaussian$, maximizing the penalized log-likelihood estimator equation (3) is equivalent to minimizing the penalized sum of squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}\left\{y_i - \mathbf{H}(\mathbf{x}_i^T\boldsymbol{\theta})\boldsymbol{\beta} - \mathbf{z}_i\gamma\right\}^2 + \frac{1}{2}\lambda\boldsymbol{\beta}^\top\mathbf{D}\boldsymbol{\beta}.$$

For the simultaneous non-linear least square minimization methods in Yu and Ruppert (2002), we can directly apply a standard nonlinear least square (NLS) optimization algorithm on minimization of the above penalized sum of squared errors with respect to the full parameter $\boldsymbol{\omega} = (\boldsymbol{\theta}, \boldsymbol{\beta}, \gamma)$. This is useful to facilitate joint inferences as described in Yu and Ruppert (2002). This algorithm as presented in Yu and Ruppert (2002) is also implemented in our **gplsim** package.

Remark. In Step 2 of the profile-likelihood algorithm implementing Yu et al. (2017) as well as the simultaneous non-linear least square minimization methods implementing Yu and Ruppert (2002), the spline knots used for the basis functions depend on $\boldsymbol{\theta}$ because they are sample quantiles or equally-spaced placed on the single index $\{\boldsymbol{x}_i^T\boldsymbol{\theta}, i = 1, ..., n\}$. That is, the spline coefficient $\boldsymbol{\beta}$, along with the spline knots implicitly, depends on the single index coefficient $\boldsymbol{\theta}$. We refer to the original methodological papers Yu and Ruppert (2002), Yu et al. (2017), and references therein for more details.

3 The gplsim package

The R package **gplsim** consists of one core estimation function `gplsim` and some supporting functions such as visualization of the estimated curve for the unknown univariate function. The R package **gplsim** depends on the R package **mgcv** (Wood, 2001) and package **minpack.lm** (Elzhov et al., 2022). Unit tests using **testthat** are in place to ensure continued robustness of the package.

Main fitting function

The main estimation function `gplsim` implements the profile likelihood algorithm of Yu et al. (2017) as well as the non-linear least square method of Yu and Ruppert (2002) described in the previous section. The default method is the profile likelihood for responses from a general exponential family and non-linear least square method for others.

The usage and input arguments of the main fitting function `gplsim` are summarized as follows:

```
gplsim(Y, X, Z, family = gaussian, penalty = TRUE, profile = TRUE, user.init = NULL,
  bs= "ps", ...)
```

This function takes three required arguments: the response variable Y in vector format, the single-index nonlinear predictors X in the matrix or vector format, and the linear predictors Z in the matrix or vector format. Please note that all the input covariates are required to be numeric variables.

This function also takes several optional arguments for finer controls. The optional argument `family` is a family object for models from the built-in R package **stats**. This object is a list of functions and expressions for defining link and variance functions. Supported link functions include identity; logit, probit, cloglog; log; and inverse for the family distributions of Gaussian, Binomial, Poisson, and Gamma, respectively. Other families supported by `glm` and `mgcv::gam` are also supported. The optional argument `penalty` is a logical variable to specify whether to use penalized splines or un-penalized splines to fit the model. The default value is TRUE to implement penalized splines. The optional argument `profile` is a logical variable that indicates whether the algorithm with profile likelihood or the algorithm with NLS procedure is used. The default algorithm is set to the profile likelihood algorithm. The optional argument `user.init` is a numeric vector of the same length as the dimensionality of single-index predictors. The users can use this argument to pass in any appropriate user-defined initial single-index coefficients based on prior information or domain knowledge. The default value is NULL, which instructs the function to estimate initial single-index coefficients by a generalized linear model.

As we utilize `mgcv::gam` and `mgcv::s` as the underlying algorithms for the estimation of the unknown univariate function of the single index, there are several arguments that can be passed into `mgcv::gam` and `mgcv::s` for finer control. For example, the optional argument `bs` is a character variable that specifies the spline basis in the estimation of the single index function, and it will be passed into `mgcv::s`. The default has been set to "ps" (P-splines with B-spline basis) while other choices are "tr" (truncated power basis), "tp" (thin plate regression splines), and others (see the help page of `mgcv::smooth.terms`). Other `mgcv::gam` arguments can be passed to `mgcv::s` in `...` includes the optional numeric arguments `k`, which is the dimension of the basis of the smooth terms and the arguments `m`, which is the order of the penalty for the smooth terms. Additionally, users can also pass arguments `scale` into `gam` in `...`. It is a numeric indicator with a default value set to -1. Any negative value including -1 indicates that the scale of response distribution is unknown and thus needs to be estimated. Another option is 0, indicating a scale of 1 for Poisson and binomial distribution and unknown for others. Any positive value will be taken as the known scale parameter. The optional argument `smoothing_selection` is a character variable that specifies the criterion used in the selection of the smoothing parameter λ. This argument corresponds to the argument `method` in `mgcv::gam`, but it is renamed in this package to avoid confusion. The supported criteria include "GCV.Cp","GACV.Cp", "ML","P-ML", "P-REML" and "REML", while the default criterion is "GCV.Cp". For more details regarding arguments in `mgcv::gam` and `mgcv::s`, users may refer to the help page of `mgcv::gam` and `mgcv::s`.

The function `gplsim` returns an object class of `gplsim`, which extends the `gam` object and `glm` object.

Other functions

```
plot_si(gplsim.object,reference = NULL)
```

This function plots the estimated curve for the unknown univariate function ϕ from a gplsim-fitted model object. If the reference object is provided, this function will add a reference line accordingly.

```
summary.gplsim(gplsim.object)
print.summary.gplsim(gplsim.object)
```

The functions `summary.gplsim` and `print.summary.gplsim` provide detailed information related to the fitted model and summarize the results as illustrated in the next section. These two functions can be called directly by applying functions print and summary to gplsim.object.

```
simulation_data <- generate_data(n,true.beta=c(1, 1, 1)/sqrt(3),family="gaussian")
```

The function `generate_data` generates data from a sine-bump model with user-defined single index coefficients $\boldsymbol{\theta}$ via the argument `true.beta`. If single-index coefficients $\boldsymbol{\theta}$ are not provided, this function will generate data against the default coefficients $\boldsymbol{\theta} = (1,1,1)/\sqrt{3}$. The default response is Gaussian distributed, while Binomial, Poisson, and Gamma distributions are also supported.

4 Real data and simulation examples

In this section, we demonstrate the use of the R package **gplsim** via a real data analysis and a sine-bump simulation study.

Air Pollution Data

We consider an environmental study on how meteorological variables X affect the concentration of the air pollutant ozone y. Meteorological variables X contain wind speed, temperature, and radiation with $n = 111$ daily measurements. As the response variable y is a continuous variable, we adopt an identity link for Gaussian distribution. Note that we use the same sequence of predictor variables to keep the results directly comparable to Yu and Ruppert (2002).

```
library(gplsim)
data(air)
y=air$ozone                    # response
X=as.matrix(air[,c(3,4,2)])    # single-index term
colnames(X)=colnames(air[,c(3,4,2)])
Z=NULL
```

We allow all three predictor variables, temperature, wind speed, and radiation, to enter the single-index term to capture the non-linear dependency as in Yu and Ruppert (2002). This model collapses to the single-index model as there is no partially linear term in the model.

```
air.fit <- gplsim(y,X,Z=NULL,family = gaussian,bs="ps")
summary(air.fit)

#>
#> Family: gaussian
#> Link function: identity
#>
```

```
#> Formula:
#> y ~ s(a, bs = bs, fx = fx, m = 2, k = k)
#>
#> partial linear coefficients:
#>           Estimate Std. Error t value  Pr(>|t|)
#> Intercept 3.247784   0.043024  75.488 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>
#> single index coefficients:
#>             Estimate
#> temperature   0.5442
#> wind_speed   -0.8386
#> radiation     0.0223
#>
#> Approximate significance of smooth terms:
#>         edf Ref.df      F   p-value
#> s(a) 8.1431  9.173 34.867 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.741   Deviance explained =   76%
#> GCV = 0.22391  Scale est. = 0.20546   n = 111
```

The estimated normalized single-index coefficients with the profile likelihood algorithm are comparable to the results in Yu and Ruppert (2002). As shown in the figure below, the estimated unknown function is quite monotonic and exhibits clear curvature. The estimated coefficient is positive for temperature, negative for wind speed, and positive for radiation but in a smaller magnitude per the reported summary.

```
plot_si(air.fit,yscale=c(1,6),plot_data = TRUE)
```

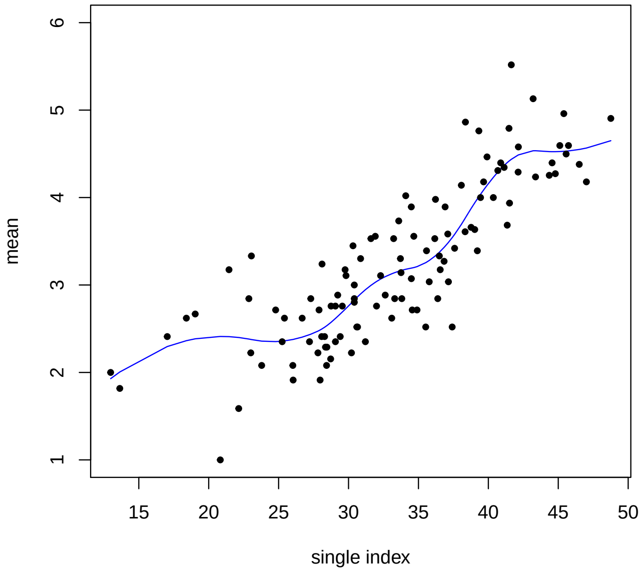


Figure 1: Single-index curve estimate of the air pollution data, an output of function `plot_si`. The data are represented by dots.

The above figure shows the single-index curve estimate of the air pollution data. The presence of curvature with multiple turning points is observed. This non-linear dependency is unlikely to be captured by a linear model. The single index that contains information from temperature, wind speed, and radiation contributes to the ozone concentration differently in different segments.

We also implemented the simultaneous non-linear least square minimization algorithm described in Yu and Ruppert (2002), where the original code was written in Matlab.

```
air.fit <- gplsim(y,X,Z=Z,family = gaussian,profile = FALSE,bs=”ps”)
summary(air.fit)
```

```
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> y ~ s(a, bs = bs, fx = !penalty, m = 2, k = 13)
#>
#> partial linear coefficients:
#>           Estimate Std. Error t value  Pr(>|t|)
#> Intercept 3.247784   0.043056  75.431 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>
#> single index coefficients:
#>             Estimate
#> temperature   0.5340
#> wind_speed   -0.8451
#> radiation     0.0235
#>
#> Approximate significance of smooth terms:
#>         edf Ref.df     F   p-value
#> s(a) 8.0012 9.0533 35.22 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =   0.74   Deviance explained = 75.9%
#> GCV = 0.22394  Scale est. = 0.20578   n = 111
```

Note that here outputs are directly from mgcv package for GAM model summary. The p-value and confidence intervals here do not take into account the uncertainty in the estimation of the single-index coefficients. If required, valid inference using bootstrap or asymptotic results from Yu and Ruppert (2002) and Yu et al. (2017) is available.

Simulations

We present a popular sine-bump simulation study that adopts the design as in (Carroll et al., 1997; Yu et al., 2017; Yu and Ruppert, 2002). The package can accommodate responses from a general exponential family, where the conditional mean is generated from the following model

$$g^{-1}\{\sin\left\{\pi\left(\mathbf{x}^T\boldsymbol{\theta}-c_1\right)/\left(c_2-c_1\right)\right\}+z\gamma\},$$

where $g\{\cdot\}$ is a link function; $\boldsymbol{\theta}=(1,1,1)/\sqrt{3}$ with each predictor x from independent uniform in $[0,1]$; $\gamma=0.3$; z is a binary predictor with 1 for even observations and 0 otherwise; $c_1=\sqrt{3}/2-1.645/\sqrt{12}$ and $c_2=\sqrt{3}/2+1.645/\sqrt{12}$ are two constants.

For demonstration, we first show simulation codes and outputs on one random replication. We use the supporting function generate_data to generate simulation data.

```
set.seed(2020)
# Gaussian family
# parameter settings
n=1000
M=200
true.theta = c(1, 1, 1)/sqrt(3)
# This function generates a sine-bump simulation data
data <- generate_data(n,true.theta=true.theta,family=”gaussian”,ncopy=M)
y=(data$Y)[[1]]       # Gaussian error with standard deviation 0.1
X=data$X       # single-index predictors
Z=data$Z       # partially linear predictors
```

We use default settings of the main estimation function `gplsim` on the simulated data, assuming no prior information. The codes and summary results are provided as follows.

```
result <- gplsim(y,X,Z,user.init=NULL,family = gaussian)
summary(result)

#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> y ~ s(a, bs = bs, fx = fx, m = 2, k = k) + z
#>
#> partial linear coefficients:
#>            Estimate Std. Error t value  Pr(>|t|)
#> Intercept 0.6439549  0.0046579 138.251 < 2.2e-16 ***
#> Z.1       0.3057685  0.0065963  46.354 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>
#> single index coefficients:
#>     Estimate
#> X.1   0.5786
#> X.2   0.5798
#> X.3   0.5736
#>
#> Approximate significance of smooth terms:
#>         edf Ref.df      F   p-value
#> s(a) 6.3561 7.4656 2129.9 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.947   Deviance explained = 94.7%
#> GCV = 0.010909  Scale est. = 0.010818  n = 1000
```

From the above summary results of the fitted model, we see that the estimated single-index coefficients $\widehat{\boldsymbol{\theta}}$ and partial-linear coefficients $\widehat{\gamma}$ are quite close to the true parameters.

We also plot the average estimated curve for the unknown univariate function over 200 replications. The dashed lines are the corresponding 2.5 and 97.5 quantiles bound. We observe that the average curve estimate virtually overlays the true curve.

```
#plot the estimated univariate function curve
plot_si(result,plot_data = FALSE)
par(new=T)
sort_index = order(X%*%true.theta)
lines((X%*%true.theta)[sort_index],data$single_index_values[sort_index],lty=1,
xaxt="n", yaxt="n",col="red")
legend("topright",legend=c("GPLSIM fit", "True"),lty=c(1,1),col = c("black","red"))
add_sim_bound(data)
```

Table 1 reports the mean, standard error (se) and bias for each parameter estimate with sample size $n = 1000$ over $M = 200$ replications. We use canonical link functions, that is, identity link for Gaussian family, logit link for Binomial family, and log link for Poisson family. One can see that the algorithm for our R package **gplsim** is effective in estimation of various GPLSIMs.

5 Summary

In this paper, we have presented an R package **gplsim** that implements generalized partial linear single-index models described in Yu et al. (2017) and Yu and Ruppert (2002). The functions and algorithms used in the package are able to accurately estimate the single-index coefficients, partial-linear coefficients, as well as the unknown univariate function with expedient computation. We believe this package will be useful to practitioners in diverse fields such as finance, econometrics, and medicine, where a flexible and interpretable models are desirable.

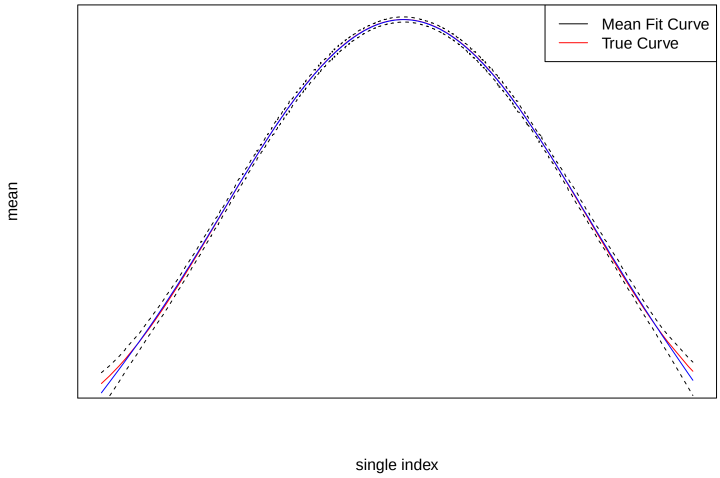


Figure 2: Curve estimates and confidence bands for the unknown univariate function. The red solid curve is the true curve. The blue solid curve is the average fitted curve over 200 replications. The dashed curves are the corresponding 2.5% and 97.5% quantiles.

	Gaussian		Binomial		Poisson	
	Mean(se)	Bias	Mean(se)	Bias	Mean(se)	Bias
$\hat{\theta}_1$	0.5771(0.0048)	-0.0002	0.5545(0.1040)	-0.0228	0.5808(0.0305)	0.0035
$\hat{\theta}_2$	0.5774(0.0048)	0.0005	0.5744(0.1111)	-0.0029	0.5738(0.0349)	-0.0035
$\hat{\theta}_3$	0.5765(0.0047)	-0.0004	0.5717(0.1126)	-0.0056	0.5745(0.0340)	-0.0028
$\hat{\gamma}$	0.2995(0.0065)	-0.0004	0.3094(0.1312)	0.0094	0.2978(0.0430)	-0.0022

Table 1: Summary of parameter estimates for various responses of sample size $n = 1000$. True $\boldsymbol{\theta} = (1,1,1)/\sqrt{3}, \gamma = 0.3$. The sample mean (mean), standard error (se, in parenthesis), and bias of the parameter estimates from generalized partially linear single-index models (GPLSIMs) by penalized splines from 200 replications.

Bibliography

R. S. Anderssen and P. Bloomfield. A Time Series Approach to Numerical Differentiation. *Technometrics*, 16(1):69–75, 1974. ISSN 0040-1706. doi: 10.2307/1267494. URL https://www.jstor.org/stable/1267494. [p57]

C. d. Boor. *A Practical Guide to Splines*. Springer, New York, Nov. 2001. ISBN 978-0-387-95366-3. [p56]

R. J. Carroll, J. Fan, I. Gijbels, and M. P. Wand. Generalized Partially Linear Single-Index Models. *Journal of the American Statistical Association*, 92(438):477–489, 1997. ISSN 0162-1459. doi: 10.2307/2965697. URL https://www.jstor.org/stable/2965697. [p55, 56, 61]

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*, 2022. URL https://CRAN.R-project.org/package=minpack.lm. R package version 1.2-2. [p58]

P. Hall. On Projection Pursuit Regression. *The Annals of Statistics*, 17(2):573–588, June 1989. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1176347126. [p56]

T. Hastie and R. Tibshirani. Generalized Additive Models. *Statistical Science*, 1(3):297 – 310, 1986. doi: 10.1214/ss/1177013604. URL https://doi.org/10.1214/ss/1177013604. [p55]

W. Härdle, P. Hall, and H. Ichimura. Optimal Smoothing in Single-Index Models. *The Annals of Statistics*, 21(1):157–178, 1993. ISSN 0090-5364. URL https://www.jstor.org/stable/3035585. [p55]

W. Härdle, H. Liang, and J. Gao. *Partially Linear Models*. Springer Science & Business Media, Dec. 2012. ISBN 978-3-642-57700-0. [p55]

H. Ichimura. Semiparametric least squares (SLS) and weighted SLS estimation of single-index models. *Journal of Econometrics*, 58(1):71–120, July 1993. ISSN 0304-4076. doi: 10.1016/0304-4076(93)90114-K. URL http://www.sciencedirect.com/science/article/pii/030440769390114K. [p55]

H. Liang, X. Liu, R. Li, and C.-L. Tsai. ESTIMATION AND TESTING FOR PARTIALLY LINEAR SINGLE-INDEX MODELS. *The Annals of Statistics*, 38(6):3811–3836, 2010. ISSN 0090-5364. URL https://www.jstor.org/stable/29765281. [p55]

P. McCullagh and J. A. Nelder. *Generalized Linear Models, Second Edition*. CRC Press, Aug. 1989. ISBN 978-0-412-31760-6. [p55]

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/. [p55]

D. Ruppert and R. J. Carroll. Theory & Methods: Spatially-adaptive Penalties for Spline Fitting. *Australian & New Zealand Journal of Statistics*, 42(2):205–223, 2000. ISSN 1467-842X. doi: 10.1111/1467-842X.00119. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-842X.00119. [p57]

D. Ruppert, M. P. Wand, and R. J. Carroll. *Semiparametric regression*. 12. Cambridge university press, 2003. [p56]

G. Wahba. A Comparison of GCV and GML for Choosing the Smoothing Parameter in the Generalized Spline Smoothing Problem. *Annals of Statistics*, 13(4):1378–1402, Dec. 1985. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1176349743. URL https://projecteuclid.org/euclid.aos/1176349743. [p57]

S. N. Wood. mgcv: GAMs and Generalized Ridge Regression for R. *R news*, Vol.1(2):20, June 2001. [p55, 58]

S. N. Wood. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(1):3–36, 2011. ISSN 1467-9868. doi: 10.1111/j.1467-9868.2010.00749.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2010.00749.x. [p55, 57]

Y. Xia. A Multiple-Index Model and Dimension Reduction. *Journal of the American Statistical Association*, 103(484):1631–1640, 2008. ISSN 0162-1459. doi: 10.1198/016214508000000805. URL http://www.jstor.org/stable/27640210. [p56]

Y. Xia and W. Härdle. Semi-parametric estimation of partially linear single-index models. *Journal of Multivariate Analysis*, 97(5):1162–1184, May 2006. ISSN 0047-259X. doi: 10.1016/j.jmva.2005.11.005. URL http://www.sciencedirect.com/science/article/pii/S0047259X05001995. [p55]

Z. Yang, K. Balasubramanian, and H. Liu. High-dimensional Non-Gaussian Single Index Models via Thresholded Score Function Estimation. In *ICML*, 2017. [p56]

Y. Yu and D. Ruppert. Penalized Spline Estimation for Partially Linear Single-Index Models. *Journal of the American Statistical Association*, 97(460):1042–1054, 2002. ISSN 0162-1459. URL https://www.jstor.org/stable/3085829. [p55, 56, 57, 58, 59, 60, 61, 62]

Y. Yu, C. Wu, and Y. Zhang. Penalised spline estimation for generalised partially linear single-index models. *Statistics and Computing*, 27(2):571–582, Mar. 2017. ISSN 0960-3174. doi: 10.1007/s11222-016-9639-0. URL https://doi.org/10.1007/s11222-016-9639-0. [p55, 57, 58, 61, 62]

Tianhai Zu
The University of Texas at San Antonio
One UTSA Circle
San Antonio, TX 78249
https://orcid.org/0000-0002-4634-7937
tianhai.zu@utsa.edu

Yan Yu
University of Cincinnati
2906 Woodside Drive
Cincinnati, OH 45221
https://orcid.org/0000-0002-2859-3093
Yan.YU@uc.edu

Non-Parametric Analysis of Spatial and Spatio-Temporal Point Patterns

by Jonatan A. González and Paula Moraga

Abstract The analysis of spatial and spatio-temporal point patterns is becoming increasingly necessary, given the rapid emergence of geographically and temporally indexed data in a wide range of fields. Non-parametric point pattern methods are a highly adaptable approach to answering questions about the real-world using complex data in the form of collections of points. Several methodological advances have been introduced in the last few years. This paper examines the current methodology, including the most recent developments in estimation and computation, and shows how various R packages can be combined to run a set of non-parametric point pattern analyses in a guided and intuitive way. An example of non-specific gastrointestinal disease reports in Hampshire, UK, from 2001 to 2003 is used to illustrate the methods, procedures and interpretations.

1 Introduction

A spatial point pattern $X = \{\mathbf{u}_i\}_{i=1}^n$ is a random collection of points observed within a bounded region of the plane, $W \subset \mathbb{R}^2$. Spatial point patterns arise in a broad range of applied disciplines such as climatology, ecology, epidemiology, and seismology. A spatial point pattern can represent the locations of forest fires (Turner 2009), the occurrence of species (Moraga 2021) or the residence locations of people with a certain disease (Moraga and Montes 2011). When the points evolve in space and time, we call them *spatio-temporal point patterns*. In this case, we treat the data as being generated as a snapshot in space-time, and they may be viewed as a spatial point process with a further (temporal) dimension (González et al. 2016). Similarly to the spatial case, we consider a spatio-temporal point pattern as a countable set of random points $\{(\mathbf{u}_i, v_i)\}_{i=1}^n$ (non-overlapping), where $\mathbf{u}_i$ represents the spatial location of the i^{th} event and all of them belong to a bounded planar region $W \subset \mathbb{R}^2$. Here, v_i is associated with time and it belongs to a compact positive interval $T \subset \mathbb{R}_+$. $W \times T$ is usually called the spatio-temporal observation window. Examples of spatio-temporal point patterns include the locations of individuals with an infectious disease in which the time of infection is known (P. J. Diggle 2013), the starting (or ending) locations of tornadoes with their registered times (González, Hahn, and Mateu 2019) or the location and time at which some crime was committed (Rodrigues and Diggle 2012). Point pattern analyses may also be performed in smaller dimensions, such as straight segments (Daley and Vere-Jones 2003). For example, we can consider a time segment as a straight segment and record the exact time the heartbeat of a patient with arrhythmia occurs during an electrocardiogram. In addition, we can also have *marked point patterns* in which each of the points has some additional characteristics. For example, a marked point pattern may represent a pattern of beta-type ganglion cells in a cat's retina that can be classified anatomically as the marks "on" or "off" (Wässle, Boycott, and Illing 1981).

When dealing with point patterns, it is desirable to learn as much as possible about the probability distribution that governs the occurrence of points in a given point pattern. This is a difficult task since, in general, there is only a single set of points or realisation of the point process. However, many methods have been developed that serve to estimate specific characteristics of the distribution, such as the intensity, which describes the expected number of points per unit area; or the clustering or aggregation behaviour, that is, whether the points have a particular attraction or repulsion mechanism. Statistical models for point processes can also be formulated, and these may consider previous knowledge to describe trends, variability, and different factors that may impact them. Point patterns are usually considered a random sample of a stochastic mechanism called *point process*. The most typical families of point processes are *Poisson processes*. There are two classes of Poisson processes; the *homogeneous*, whose realisations in any subset of the observation window are uniform random samples with a constant mean λ, and the *inhomogeneous*, obtained by replacing the intensity λ by a spatial, temporal or spatio-temporal varying one (Illian et al. 2008; P. J. Diggle 2013). Homogeneous Poisson processes are benchmark models for spatial and spatio-temporal point pattern data, often called *complete spatial random (CSR)* or *complete spatio-temporal random (CSTR)* point processes; i.e., point processes whereby point events appear in a completely random fashion.

A large body of literature has been developed in recent years to study point patterns and many statistical software packages have been developed for their estimation and analysis. For instance, packages such as **spatial** (Venables and Ripley 2002) or **splancs** (Rowlingson and Diggle 1993) initially developed for S-PLUS but available on CRAN are classics for displaying and analysing spatial point pattern data. Packages as **sp** (E. J. Pebesma and B. 2005; Bivand, Pebesma, and Gómez-Rubio 2013), or more recently, **sf** (E. Pebesma 2018), were created to provide a uniform interface for handling 2D and 3D data. **spatstat** (A. Baddeley and Turner 2005; Adrian Baddeley, Rubak, and Turner 2015) is one of the most popular and improved packages for analysing spatial point patterns It is divided into several subpackages: **spatstat.utils**, **spatstat.sparse**, **spatstat.data**, **spatstat.geom**, **spatstat.random**, **spatstat.explore**, **spatstat.model** and **spatstat.linnet**. Some extra specialised packages such as **smacpod** or **sparr** (T. M. Davies, Marshall, and Hazelton 2018) have methods for analysing case-control point patterns; or **spatgraphs** for graphs-related analyses of locations in 1D, 2D, 3D.

Even though authors have considered the combination of space and time in theory, and nowadays, spatial data often includes a time component, only a few examples show this in practice. This problem may be due to the complexity of the computations, added to the fact that the authors tend to delve into particular contexts. There are some tools available for dealing with spatio-temporal point pattern data. The **stpp** package (Gabriel, Rowlingson, and Diggle 2013) covers many models and functional descriptors related to point process methods to study spatio-temporal phenomena. The **PtProcess** package (Harte 2010) was designed to fit time-indexed point process models in the same simplified fashion as generalised linear models software. The **lgcp** (B. M. Taylor et al. 2013), as well as the **inla** (Rue, Martino, and Chopin 2009; Moraga 2019); `http://www.r-inla.org` and **inlabru** (Bachl et al. 2019) packages, are suitable options for statistical inference in space and time.

In this paper, we demonstrate how R (R Core Team 2023) can be used to study spatial and spatio-temporal point process data using a dataset of geographic locations and times of individuals with non-specific gastrointestinal infections in Hampshire, UK, from 2001 to 2003 (P. Diggle et al. 2003). We consider the data as realisations of spatial and spatio-temporal point processes that lack spatial and temporal homogeneity; i.e., the expected number of points in each area unit of the study region depends on their location and time. Consequently, the first problem we solve is the intensity estimation. We then investigate the nature of the stochastic interactions between the process points after adjusting for spatial and spatio-temporal inhomogeneity. We use second-order descriptors to describe these interactions.

Along with this paper, we employ several R packages to successfully analyse the complex point patterns seen in both the spatial and the spatio-temporal settings. For dealing with spatial point patterns, we mainly use the **spatstat** package. In addition, we use the **sparr** package for analysing the relative risk and the **GET** package (Myllymäki and Mrkvička 2020) for testing hypotheses with envelopes. For the spatio-temporal analysis, we use the **stpp** package. We also make use of other R packages such as **parallel**, **foreach** (Microsoft and Weston 2022b) and **doParallel** (Microsoft and Weston 2022a) to optimise the computing time by dividing the burden by several processors. An R script containing all the necessary code for reading the data and running the analyses shown in this paper is provided as supplementary material.

Spatio-temporal point pattern data

We use the geographic locations and time information of individuals who had non-specific gastrointestinal infections in Hampshire, UK, between 2001 and 2003 (P. Diggle et al. 2003). These data come from the *Ascertainment and Enhancement of Gastrointestinal Infection Surveillance and Statistics* (AEGISS) project (P. Diggle et al. 2003), which, using the Hampshire area as a test, seeks to identify irregularities, i.e., high or low intensity spots in the spatio-temporal distribution of non-specific gastrointestinal infections in the UK.

The data consist of a collection of points $\{(\mathbf{u}_i, v_i)\}_{i=1}^n$, where $\mathbf{u}_i$ is a spatial coordinate with two components, i.e., $\mathbf{u}_i = (a_i, b_i), a_i, b_i \in \mathbb{R}$, and $v_i \in \mathbb{R}_+$ is a temporal coordinate. These points represent the locations and times of daily healthcare providers reports of non-specific gastrointestinal disease in Hampshire from 2001 to 2003. After cleaning the data to remove non-Hampshire residents (out of the Hampshire region) and multiple (coincident) points, there were $n = 10443$ cases of non-specific gastrointestinal disease by a phone-in triage service operating within the UK's National Health Service. The command sequence

```
library(spatstat)
library(viridis)
load("AegissData.RData")
```

loads the data in the local environment of R. Each location corresponds to the centroid of the unit post-code of the residential address of the caller. The unit of distance is one kilometre. The unit of time is one day, with day 1 corresponding to 1 January 2001. We can inspect the first spatio-temporal coordinates of our dataset by writing

```
head(as.data.frame(Aegiss))
```

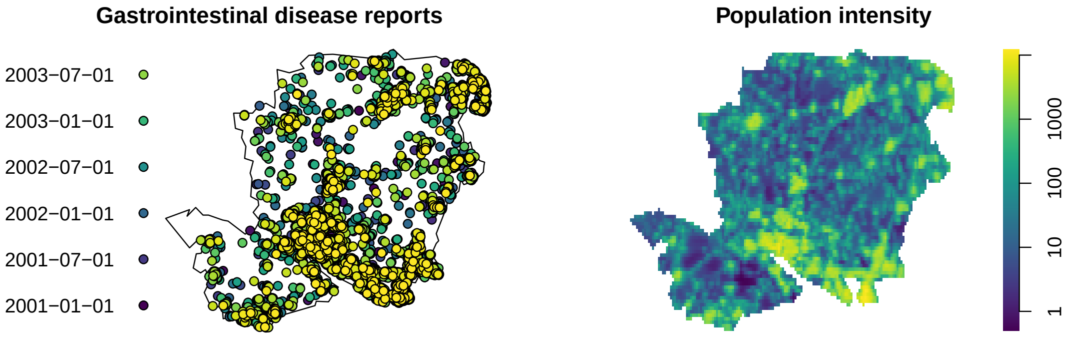


Figure 1: Left: Locations of 10443 reports of non-specific gastrointestinal disease in Hampshire from January 2001 to July 2003. The time is treated as a quantitative mark where darker dots correspond to the older events, and lighter dots match the most recent reports. Right: Hampshire population density averaged over the years 2001-2003. The colour map is evenly-spaced on a logarithmic scale.

```
#>        x      y marks
#> 1 478.55 134.85     1
#> 2 458.15 107.35     1
#> 3 449.05 128.95     1
#> 4 450.75 106.95     1
#> 5 461.05  99.55     1
#> 6 446.35 109.15     1
```

Notice that x and y are the coordinates. Times are not labelled as dates, but they are integers, and their column is named as 'marks'. Finally, the planar region that encloses the points corresponds to a 120-sided polygon representing the boundary of the study region of Hampshire. To visualise the spatio-temporal point pattern, we can depict a spatial map using the `symbolmap()` function of the **spatstat.geom** package setting the times as colours. The colours are selected by using the **viridis** (Garnier et al. 2023) (Figure 1).

```
Times <- Aegiss$marks
timelabels <- as.Date(Times - 1, origin = "2001-01-01")
colmap <- viridis(length(Times))
sy <- symbolmap(pch = 21, col = "black", bg = colmap, range = range(timelabels))
```

We use the locations and times of the point pattern and the population density of Hampshire to study the point process that generates these data. We obtain the mean population density of Hampshire from WorldPop (University of Southampton). The population information detailed for each squared kilometre is available in a yearly basis. Figure 1 shows the point pattern of the cases and the population density. First, we save only the spatial locations with no marks by writing `X <- unmark(Aegiss)`. Then, we attach the date labels to the points using the operator `%mark%`. Note that `symap = sy` sets the colour and point shapes previously defined and that the six dates in the legend are a sample to represent the colour variation. To summarise the population density, given in a set of three images, 2001, 2002 and 2003, we use the function `im.apply()` of the package **spatstat.geom** to average them.

```
par(mfrow = c(1, 2), mar = c(0,0,2,0.5))
X <- unmark(Aegiss)
PopDens <- im.apply(Population, mean)
plot(X %mark% timelabels, symap = sy,
     main = "Gastrointestinal disease reports")
plot(PopDens, col = viridis(prod(PopDens$dim)), log = T, box = F,
     main = "Population intensity")
```

2 Spatial analysis

Spatial intensity and relative risk

The *first-order intensity function* or the *intensity* of a point process can be defined as the expected number of events per unit space, or mathematically (P. J. Diggle 2013),

$$\lambda(\mathbf{u}) = \lim_{|d\mathbf{u}|\to 0} \frac{\mathbb{E}N(d\mathbf{u})}{|d\mathbf{u}|},$$

where $N()$ denotes the number of points of a region. In the inhomogeneous case, we can compute a kernel smoothed intensity function from a point pattern $\{\mathbf{u}_i\}_{i=1}^{n}$ without the temporal coordinates. The estimator is given by

$$\hat{\lambda}(\mathbf{u}) = \frac{1}{e^2(\mathbf{u})} \sum_{i=1}^{n} \kappa^2_\epsilon(\mathbf{u} - \mathbf{u}_i), \quad \text{and} \quad e^2(\mathbf{u}) = \int_W \kappa^2_\epsilon(\mathbf{s} - \mathbf{u}) d\mathbf{s}, \tag{1}$$

where $\mathbf{u} \in W$ and κ^2_ϵ is a two-dimensional Gaussian kernel. $e()$ is known as *uniform edge correction* and is intended to correct the bias of the estimation at the edges of the region (Adrian Baddeley, Rubak, and Turner 2015). One of the most challenging considerations is selecting the bandwidth ϵ; there is no single best procedure for doing that, but the estimation can be improved by using a spatially varying bandwidth despite its computational complexity (Tilman M. Davies and Baddeley 2018). For computing the adaptive estimate of the intensity function, we use the `adaptative.density()` function of the **spatstat.explore** package,

```
# This takes roughly 12 seconds to be executed in an
# iMAC 2019, 3GHz intel core i5 processor with 16GB of RAM (hereinafter)
SpatialDens <- adaptive.density(X, method = "kernel", edge = T)
```

The estimated intensity is shown in Figure 2 (left). We can see a high concentration of cases (more than two per square kilometre) in Southampton and its coastal environs. More to the north, there are also local foci in Winchester (the centre of the study region) and the primary urban centres located in the north and northeast of the study region.

The *kernel log relative risk* or *density ratio function* is a descriptor aimed to compare two estimated densities on the study region W. In epidemiology, it is handy to examine disease risk fluctuations based on case and control samples. In recent years, many improvements have been developed to the estimation methodology, including spatially adaptive smoothers and inference based on asymptotic theory (T. M. Davies, Marshall, and Hazelton 2018). The related techniques are implemented in the **sparr** package. Given two spatial point patterns, X, with intensity $\hat{\lambda}(\mathbf{u}|X)$ (that can be thought of as cases) and Y, with intensity $\hat{\lambda}(\mathbf{u}|Y)$ (that can be thought of as controls); the estimated log relative risk is defined as

$$\hat{r}(\mathbf{u}) = \log\left\{\frac{\hat{\lambda}(\mathbf{u}|X)}{\hat{\lambda}(\mathbf{u}|Y)} \cdot \frac{n_Y}{n_X}\right\}, \quad \mathbf{u} \in W, \tag{2}$$

where n_X and n_Y are the number of points of X and Y, respectively. When $\hat{r}(\mathbf{u}) \approx 0$, the densities are roughly the same; peaks greater than zero mean a higher local ratio of cases to controls, and $\hat{r}(\mathbf{u}) < 0$ indicates lack of concentration of cases than controls.

Here, we compute the relative risk by comparing the point pattern of gastrointestinal infections and a point pattern with controls derived from a random sample of the underlying population. We first generate a set of random controls by simulating $n = 1443$ locations with intensity proportional to the population density using the `rpoispp()` function of package **spatstat.random**. In order to do that, we calculate the population density as the population intensity normalised by its mass (its integral over the observation window) calculated by using the function `integral.im()` of the package **spatstat.geom** in each region of Hampshire. Then we can evaluate expressions involving one or more pixel images through the `eval.im()` function of package **spatstat.geom**,

```
N <- integral.im(PopDens)
n <- X$n
ControlDens <- eval.im((PopDens / N) * n)
Controls <- rpoispp(ControlDens)
```

Then we assign Hampshire's window to the new point pattern by using the `Window()` function of package **spatstat.geom**,

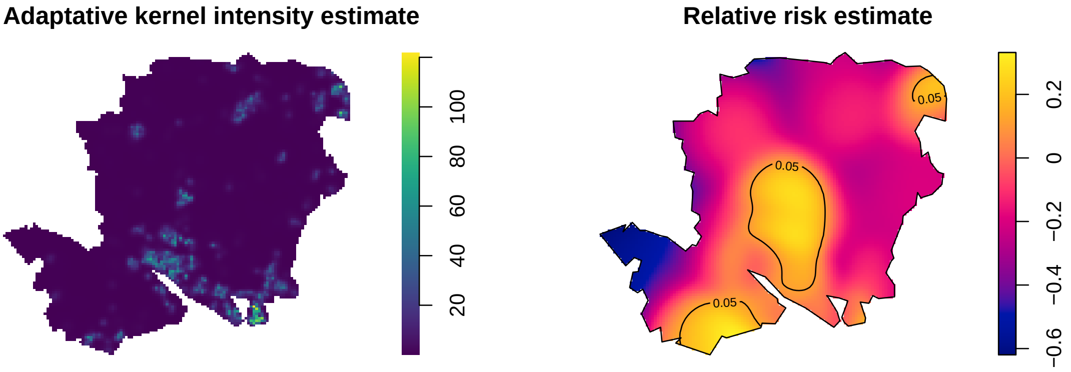


Figure 2: Left: Adaptive bandwidth kernel estimate of intensity for the gastrointestinal data. The intensity is expressed as number of reports per squared kilometre. Right: Adaptive spatial log-relative risk of non-specific gastrointestinal disease, with asymptotic tolerance contours for elevated risk.

```
Window(Controls) <- X$window
```

For estimating the relative risk, we use the **sparr** package to perform an adaptive smoothing adequately designed to deal with the reduced smoothing in densely populated areas. The code

```
# This takes roughly 22 seconds to be executed
library(sparr)
RelativeRisk <- risk(f = X, g = Controls, adapt = T,
                     pilot.symmetry = "pooled", tolerate = T)
```

takes both point patterns (cases f and controls g), computes an adaptive smoothing for estimating the risk, and computes asymptotic tolerance contours as per Hazelton and Davies (2009). Maps displayed in Figure 2 showing the intensity and relative risk estimates are produced by executing

```
par(mfrow = c(1, 2), mar = c(0,0,2,0.5))
plot(SpatialDens, col = viridis(1200), auto.axes = F,
     main = "Adaptative kernel intensity estimate", box = F)
plot(RelativeRisk, auto.axes = F, tol.type = "upper",
     main = "Relative risk estimate")
```

Figure 2 (right) provides pointwise areas of significant high relative risk. The null hypothesis is $H_0 : r(\mathbf{u}) = 0$, against an alternative of increased local risk $H_1 : r(\mathbf{u}) > 0$ for a point $\mathbf{u} \in W$. We superimpose the contours corresponding to significant values on the log relative risk surface. They are known as tolerance contours and allow us to identify areas of high risk (Kelsall and Diggle 1995). The relative abundance of gastrointestinal disease reports appears to be significantly higher in some areas where the intensity is not necessarily higher. Nor does the relative risk appear to increase in Southampton (located on the southern coast), the region's most urbanised area.

Second-order descriptors

Pair correlation and K-functions

The *second-order intensity function* $(\lambda^{(2)}(\mathbf{u}_1, \mathbf{u}_2), \mathbf{u}_1, \mathbf{u}_2 \in W)$ is, in point processes, analogous to the covariance function in classical statistics (Adrian Baddeley, Rubak, and Turner 2015). Also known as *product density function*, the second-order intensity function represents the infinitesimal two-dimensional joint distribution of the points. A more friendly summary statistic is the *pair correlation function*, which is the standardised probability density that an event occurs in each of two small spatial volumes $\mathrm{d}\mathbf{u}_1$ and $\mathrm{d}\mathbf{u}_2$. For CSR (Complete Spatial Random) point processes, the covariance is zero, and the pair correlation function takes the value of one (P. J. Diggle 2013). Different values than these benchmarks indicate how likely a pair of events will occur at the specified locations than in a CSR process with the same intensity; i.e., larger values indicate clustering, and smaller values indicate regularity. The pair correlation is defined as

$$g(\mathbf{u}_1, \mathbf{u}_2) = \frac{\lambda^{(2)}(\mathbf{u}_1, \mathbf{u}_2)}{\lambda(\mathbf{u}_1)\lambda(\mathbf{u}_2)}, \qquad \mathbf{u}_1, \mathbf{u}_2 \in W. \tag{3}$$

Whenever the intensity function of a point process is bounded away from zero and its pair correlation function depends only on the difference vector $r = ||\mathbf{u}_1 - \mathbf{u}_2||$, the point process is called *second-order intensity reweighted stationary* (SOIRS) and *isotropic*. For estimating the pair correlation function, consider a point pattern $X = \{\mathbf{u}_i\}_{i=1}^n$ with n points, then

$$\hat{g}(r) = \frac{1}{2\pi} \sum_{i=1}^{n} \sum_{j \neq i} \frac{\kappa_\epsilon(||\mathbf{u}_i - \mathbf{u}_j|| - r)}{\hat{\lambda}(\mathbf{u}_i)\hat{\lambda}(\mathbf{u}_j)||\mathbf{u}_i - \mathbf{u}_j||w_{ij}}. \tag{4}$$

Several parameters must be set before the estimation: a suitable interval for the spatial distances r, a bandwidth ϵ, a one-dimensional kernel $\kappa_\epsilon(\cdot)$ (Epanechnikov's kernel is usually recommended), a type of divisor (factor $||\mathbf{u}_i - \mathbf{u}_j||$ in the denominator of Eq. (4) can be replaced by r, this option brings a pole, i.e., an indetermination, in $r = 0$ (Wong and Stoyan 2021)); and an edge correction w_{ij} correcting for the loss of information of outsider points close to the edges. **spatstat** usually has smart defaults for these parameters based on the knowledge of the descriptors, which many authors have contributed. For this part, we use a fixed-bandwidth kernel estimate, with bandwidth set by Scott's rule of thumb (Scott 2015), through the function `bw.scott()` of the package **spatstat.explore**, as we need simulations where the intensity has to be estimated time after time. We employ the function `density.ppp()` of package **spatstat.explore** that provides a fixed bandwidth kernel smoothed intensity function from a point pattern according to Eq. (1).

```
sigmaD <- bw.scott(X)
MD <- density.ppp(X, diggle = T, positive = T, sigma = sigmaD)
MP <- density.ppp(X, diggle = T, sigma = sigmaD, at = "points")
```

In the above implementation, `diggle = T` sets the Jones-Diggle edge correction (P. J. Diggle 1985), `at = 'points'` means that the intensity is computed only at the points of X, and `sigma` is the bandwidth, which in this case is 3.5km for the horizontal axis and 4.19km for the vertical axis. In the case of the bandwidth for the pair correlation function, we employ a composite likelihood approach (Jalilian and Waagepetersen 2018) by using the function `bw.pcf()` of library **spatstat.explore** with the parameter `cv.method = 'compLik'`; the parameter `divisor = 'd'` refers to the term $||\mathbf{u}_i - \mathbf{u}_j||$ in Eq. (4), which can be replaced by r (Jalilian and Waagepetersen 2018), and `lambda` is the estimate of the intensity. We then write

```
# Time consuming: This takes roughly 24 minutes to be executed
bwG <- bw.pcf(X, cv.method = "compLik", divisor = "d", lambda = MP)
```

and obtain a bandwidth value of 2.7km. Note that this function can warn of undetermined contributions to the pair correlation. These caveats come from the divisor, which, when too small, can conflict with the numerical tolerance of R and eventually lead to indeterminacy. The **spatstat** package fixes this by simply removing the problematic values.

Before going further with the estimation of the pair correlation function, let us take a look into another classical second-order descriptor, the *inhomogeneous K-function*, which in the SOIRS case, can be defined as

$$K(r) = 2\pi \int_0^r s g(s) \mathrm{d}s.$$

For spatial Poisson point processes $K(r) = \pi r^2$. **spatstat** chooses a suitable range for r values based on some rules of thumb depending on geometric attributes of the process or the average intensity in the region. We decide to choose a proportion of 70% of the maximum value for r proposed by **spatstat.explore** through the `rmax.rule()` function, and set of 70 (71 including zero) different values for r,

```
r0 <- 0.7 * rmax.rule("K", X$window, intensity(X))
rr <- seq(0, r0, length.out = 71)
```

The classical estimator of the spatial K-function is given by

$$\hat{K}(r) = \sum_{i=1}^{n} \sum_{j \neq i} \frac{\mathbf{1}\left[||\mathbf{u}_i - \mathbf{u}_j|| \leq r\right]}{\hat{\lambda}(\mathbf{u}_i)\hat{\lambda}(\mathbf{u}_j)w_{ij}}.$$

Where $\mathbf{1}[\cdot]$ is the indicator function. An alternative descriptor defined as $L(r) - r = \sqrt{K(r)/\pi} - r$, and its straightforward estimator $\hat{L}(r) - r$ are intended for stabilising the variance (Besag 1977).

In addition to the estimation of the second-order descriptors, their standard deviation and confidence intervals are desirable for the analysis; this is not always simple, not even in the Poisson case, as there is dependence between the contributions of different points to the descriptor. However, we can use Monte Carlo simulation for computing envelopes through computational efforts. We use command envelope() of the **spatstat.explore** for getting the observed descriptor as well as its simulated counterparts.

We note that although Monte Carlo testing is generally used in point process contexts, it becomes invalid when the null hypothesis is composite, i.e., when parameters, such as the intensity function, must be estimated prior to the computing of the envelopes, which is our case for inference about the second-order descriptors. For dealing with such composite hypotheses, an extra set of simulations is needed to perform a balanced two-stage test (Adrian Baddeley et al. 2017). The **GET** package performs global envelope tests (where the significance level is controlled simultaneously for all the functions) for the second-order descriptors and provides a graphical interpretation (see, e.g., Myllymäki et al. 2017). We perform a global rank envelope test with the extreme rank length measure and retrieve the *p*-value. We warn the reader that this procedure of computing envelopes and two-stage Monte Carlo testing is computationally demanding. We calculate the first set of 39 envelopes for the centred *L*-function. This selection gives 39 (simulated) + 1 (observed) independent samples; it comes from classical settings of point process simulation (Ripley 1977; Adrian Baddeley, Rubak, and Turner 2015; Adrian Baddeley et al. 2017). We use the envelope.ppp() function of the **spatstat.explore** package; this function computes simulation envelopes of a summary function,

```
# Time consuming: This takes roughly 3 minutes to be executed
L1 <- envelope(X, nsim = 39, savefuns = TRUE, fun = "Linhom", diggle = T,
               transform = expression(.-r), sigma = sigmaD, r = rr,
               simulate = expression(rpoispp(lambda = MD)), verbose = F)
```

Then we generate another set of inhomogeneous Poisson point patterns with the estimated intensity,

```
Simpatterns <- rpoispp(lambda = MD, nsim = 39)
```

We design a function to calculate the intensity for each point pattern and to compute envelopes with point patterns simulated from the new intensity,

```
simL <- function(rep) {
  sim_fit <- density.ppp(Simpatterns[[rep]], diggle = T,
                         sigma = sigmaD, positive = T)
  envelope(Simpatterns[[rep]], nsim = 39, savefuns = T, fun = "Linhom",
           transform = expression(.-r), r = rr, diggle = T, sigma = sigmaD,
           simulate = expression(rpoispp(lambda = sim_fit)))
}
```

Now we apply the function for every point pattern of the list to get a list of envelope objects. This process is very slow as 39 *L*-functions have to be computed for each point pattern. We employ the package **foreach** to use multiple processors to accelerate this procedure in combination with the **parallel** and **doParallel** packages. The function detectCores() of the package **parallel** automatically detects the computer's CPU cores and the function registerDoParallel() of the **doParallel** package is used to register a parallel backend in the computer. The foreach() %dopar% function returns a list with the result of applying a function over every element of the main argument by using the registered cores of the computer processor,

```
# Time consuming: This takes roughly 21 minutes to be executed
library(foreach)
library(doParallel)
c0 <- parallel::makeCluster(detectCores() - 1)
doParallel::registerDoParallel(c0)
L.ls <- foreach(i = 1:39, .packages = c("spatstat")) %dopar% {simL(i)}
parallel::stopCluster(c0)
```

Finally, we perform the adjusted test using the function GET.composite() of the **GET** package and plot the envelopes. In this case, we are interested in establishing a possible positive departure of the observed centred *L*-function and its simulated counterparts. Thus, we set alternative = "greater", type = 'erl', which performs a rank envelope test based on extreme rank lengths (Myllymäki et al. 2017) based on several minimal pointwise ranks; and use the **ggplot2** (Wickham 2016) package for the plots,

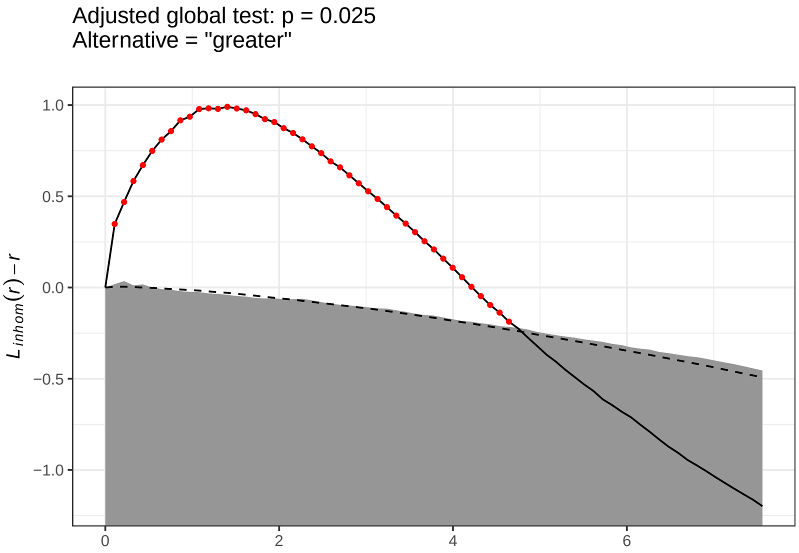


Figure 3: Graphical clustering tests of Hampshire data based on the L-functions. The gray bands represent the 95% global envelope. Red dots stand for the second-order descriptors outside the envelopes.

```
library(GET)
library(ggplot2)
resL <- GET.composite(X = L1, X.ls = L.ls, type = "erl",
                      alternative = "greater", savefuns = T)
```

Figure 3 shows the output of the rank envelope test. The rejection is explained by the attractive behaviour of gastrointestinal disease cases at short distances. The red-dot line indicates this clustering behaviour up to 5km, which shows a departure from the narrow grey band that represents the 95% adjusted rank envelope, i.e., the rank envelope that considers the composite hypothesis. Note that the more simulations, the better results for the test (Myllymäki et al. 2017). The dashed black line represents the central function (the mean) of only the first set of simulations; note that this function is not the mean value of the grey ribbon as it comes from envelopes under the composite null hypothesis. We find a significant positive departure from the envelopes up to approximately 5km; this means that observed data are more clustered in distances below 5km than might be expected in Poisson point patterns with the same intensity, and this clustering is statistically significant.

3 Spatio-temporal analysis

Spatio-temporal intensity and relative risk

First-order separability

Spatio-temporal point processes analyses can be carried out when every data point has associated geographic and time information. In these cases, estimating the first-order intensity function constitutes a primary interest. The intensity function provides information about the joint distribution of the spatial and temporal locations. Usually, this distribution is simplified by assuming, for example, separability. A point process is referred to as *separable* when its spatio-temporal intensity function can be factorised as

$$\lambda(\mathbf{u}, v) = \lambda_1(\mathbf{u})\lambda_2(v), \quad \mathbf{u} \in W, v \in T,$$

where $\lambda_1(\cdot)$ and $\lambda_2(\cdot)$ are non-negative functions of space and time, respectively. Authors often take separability for granted as it is a very pragmatic working assumption, but it should be tested before choosing an estimator. There are some specialised separability tests available in the literature. We use the package **kernstadapt** (González and Moraga 2022b) to test separability in our example by considering a division of the temporal interval $T = [1, 1095]$ into 16 disjoint equally-spaced subintervals. We split the spatial window W into 20 disjoint rectangular subsets with roughly the same area. As the separability of the intensity might be thought of as the independence of two random variables, it can be tested by a simple χ^2-test as proposed by Ghorbani et al. (2021), where the null hypothesis is separability. The test bases on the counts of points in disjoint spatial and temporal sub-regions; when there are expected counts of individual cells below 5, the authors recommend a Fisher's test. We employ Fisher's test calculating the p-value by Monte Carlo simulation (5000 replicates):

```
# This takes roughly 20 seconds to be executed
SepTest <- separability.test(Aegiss, nx = 5, ny = 4, nt = 16, nperm = 50000)

SepTest

#>
#>  Separability test based on Fisher's for counting data
#>
#> data:  Point pattern Aegiss
#> p-value = 0.00392
#> alternative hypothesis: The point pattern Aegiss is not spatio-temporal separable
```

As the p-value is significant, we opt for estimating the spatio-temporal intensity in a non-separable fashion. We estimate the intensity by employing the following estimator, which uses a Gaussian kernel for all coordinates (Choi and Hall 1999; González, Hahn, and Mateu 2019; Ghorbani et al. 2021)

$$\hat{\lambda}(\mathbf{u}, v) = \sum_{i=1}^{n} \frac{\kappa^2_\epsilon(\mathbf{u} - \mathbf{u}_i)\kappa^1_\delta(v - v_i)}{e^2(\mathbf{u}_i)e^1(v_i)}, \tag{5}$$

where

$$e^2(\mathbf{u}_i) = \int_W \kappa^2_\epsilon(\mathbf{u}_i - \mathbf{s})\mathrm{d}\mathbf{s}, \quad \text{and} \quad e^1(v_i) = \int_T \kappa^1_\delta(v_i - w)\mathrm{d}w;$$

and where κ^2_ϵ and κ^1_δ are the two- and one-dimensional Gaussian kernels with bandwidths ϵ and δ.

In this case, we use a fixed spatial bandwidth calculated by a rule of thumb that depends on the short size of the window and a temporal one calculated through the function `bw.nrd()`. Notice that the spatio-temporal estimator given in Eq.(5) resembles the spatial one given in Eq.(1), with an extra kernel weight for the temporal component. On the other hand, as the spatial edge correction, due to P. J. Diggle (1985), is already implemented in the function `density.ppp()` of **spatstat.explore**, we must calculate the temporal one. We employ the generic function `bw.nrd0()` of package **stats** for the temporal bandwidth, which uses Silverman's rule-of-thumb (Silverman 1986) for choosing the bandwidth of a Gaussian kernel density estimator. For computing the edge correction, we take advantage of the properties of the normal distribution noticing that $e^1(v_i) = \mathbb{P}(Y_i \in T)$ where $Y_i \sim N(v_i, \delta^2)$, indeed,

```
Times <- Aegiss$marks
bwt <- bw.nrd0(Times)
edgewt.t <- pnorm((max(Times) - Times) / bwt) - pnorm((min(Times)- Times) / bwt)
```

Then we proceed to calculate the non-separable estimator. To quicken the computation, for each fixed time t_0, we take temporal bins $t \in [t_0 - 3\delta, t_0 + 3\delta]$, as the normal distribution practically vanishes at larger or smaller values, then we compute the temporal kernel values and attach them to the spatial kernel as weights,

```
# This takes roughly 44 seconds to be executed
nonseparable <- function(time){
    contrib <- (Times >= time - 3 * bwt) & (Times <= time + 3 * bwt)
    Wh <- dnorm(x = Times[contrib], mean = time, sd = bwt) / edgewt.t[contrib]
    density.ppp(X[contrib], weights = Wh, diggle = TRUE)
}
nonsep <- lapply(unique(Times), nonseparable)
```

We have 1047 intensity estimates corresponding to each of the 1047 different times; note that 1047 is the size of the temporal coordinates set and not the maximum temporal value (1095). We decide to plot a subset of them, namely the estimates corresponding to times 01/01/2001, 01/01/2002, 01/01/2003 and 12/31/2003.

```
n.slices <- which(unique(timelabels) %in% as.Date(c("2001-01-01", "2002-01-01",
                                                    "2003-01-01", "2003-12-31")))
Snap <- list(nonsep[[n.slices[1]]], nonsep[[n.slices[2]]],
```

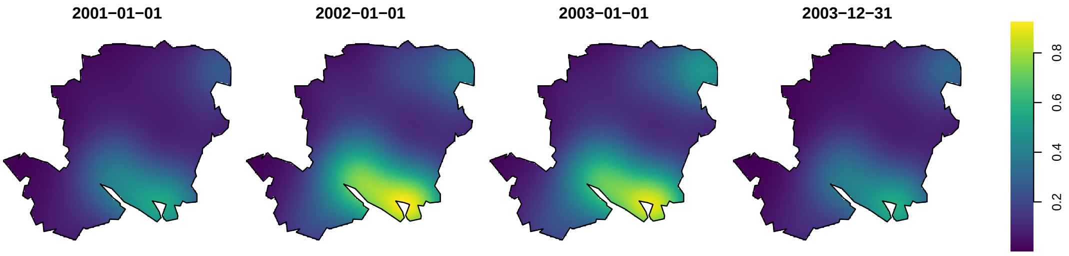


Figure 4: Temporal snapshots of the intensity $\hat{\lambda}(\mathbf{u}, v)$, of non-specific gastrointestinal reports in Hampshire, estimated in a non-separable way through Gaussian kernels. The units are rescaled by 100.

```
             nonsep[[n.slices[3]]], nonsep[[n.slices[4]]])
plot.imlist(Snap, equal.ribbon = T, ncols = 4, box = F, main = "", log = F,
            main.panel = unique(timelabels)[n.slices], col = viridis(1200),
            ribscale = 100,  mar.panel=c(0, 0, 1, 1), panel.end = X$window)
```

In the spatio-temporal case (Figure 4), we can see a behaviour in the intensity of cases much lower than in the spatial case (Figure 2 (left)) due to the temporal granularity. This scale difference is due to the number of events per unit of time, which is not considered in the only spatial estimation. When considering the temporal dimension, the total points must be divided by the number of bins in the temporal grid where the estimation is made. Therefore, there are fewer points per unit area at each time than only when spatial coordinates are considered.

The intensity maintains some urban centres as foci over time, especially those in the south, for example, in Portsmouth, where there is a high concentration of cases between 2002 and 2003.

In the next section, we show how to compute spatio-temporal descriptors. To do that, we need to know the intensity values at every point of the point pattern. We can extract the intensity values by using the `safelookup()` of **spatstat.geom**. Then, we attach those values, and the dates, as a mark to each point by using the operator `%mark%`,

```
lambda <- NULL
for (i in 1:length(nonsep)) {
    lambda <- c(lambda, safelookup(nonsep[[i]],
    Aegiss[marks(Aegiss) == unique(Times)[i]]))
}
PP <- X %mark% data.frame(time = Aegiss$marks, Lambda = lambda)
```

Spatio-temporal relative risk

The concept of relative risk straightforwardly extends to the spatio-temporal domain. In this regard, we must make additional considerations; for example, the spatial relative risk can be conditioned to a specific time (T. M. Davies, Marshall, and Hazelton 2018). Analogous to the classic probability, in the spatio-temporal context, the idea of conditioning on a specific time $v = t_0$ comes naturally. This means that for a fixed instant t_0, we consider the spatial density function given that time. Mathematically, it is calculated by normalising the spatio-temporal density by the density of the temporal coordinates evaluated at the fixed time t_0. On the other hand, if it is not conditioned to a particular time, the risk function is a function of three variables (two spatial coordinates and one temporal), which can be drawn through snapshots. Thus, we have two possible risk functions, $r(\mathbf{u}, v)$ (if plotted at a fixed time, it represents an unnormalised slice of the three-dimensional function) and $r(\mathbf{u}|v = t_0)$ (a spatial-only snapshot of relative risk).

Given that we have population information for 2001, 2002 and 2003, we generate controls from these three population densities considering uniform times within each year. Then, we superimpose the three control point patterns into one using the `superimpose()` function of the **spatstat.geom** package. We obtain a spatio-temporal point pattern of controls that will serve as input to calculate the denominator of the relative risk.

```
Ni <- lapply(Population, integral.im)
ni <- lapply(split(Aegiss, cut(Aegiss$marks, breaks = c(0, 365, 730, 1095))),
             npoints)
ControlDensi <- mapply(function(D, N, n) {eval.im((D / N) * n)},
                       D = Population, N = Ni, n = ni, SIMPLIFY = F)
Controlsi <- mapply(function(L, Breaks){
  PP <- rpoispp(lambda = L)
  marks(PP) <- sample(x = (Breaks[1] + 1):Breaks[2],
                      size = npoints(PP), replace = T)
  return(PP)},
  L = ControlDensi, Breaks = list(c(0, 365), c(365, 730), c(730, 1095)),
  SIMPLIFY = F)

Controlsi <- superimpose(as.solist(Controlsi))
Controlsi <- unmark(Controlsi) %mark% Controlsi$marks$origMarks
Window(Controlsi) <- Window(Aegiss)
```

For the spatio-temporal risk estimation exercise, we return to the **sparr** package. We follow the recommendations of (T. M. Davies, Marshall, and Hazelton 2018) and set *oversmoothing bandwidths* (a rule based on an optimal bandwidth upper bound for the asymptotic mean integrated squared error of the density estimate, this may be done using the command `OS.spattemp()`). The spatio-temporal densities can analysed through the command `spattemp.density()` of the **sparr** package.

```
# This takes roughly 4.87 minutes to be executed
Br <- OS.spattemp(Aegiss)
f.num <- spattemp.density(Aegiss, h = Br[1], lambda = Br[2])
d.den <- spattemp.density(Controlsi, h = Br[1], lambda = Br[2])
```

In this case, interest may rise about the anomalous regions of spatio-temporal risk considering joint and conditional risks (given a time t); i.e., we are interested in testing the null hypotheses $H_0 : r(\mathbf{u}, v) = 0$ or $H_0 : r(\mathbf{u}, |v = t) = 0$ against the alternatives of high risks. We employ the `spattemp.risk()` function of **sparr** to obtain a spatiotemporal relative risk surface based on the ratio of the two kernel estimates. In addition, since doing Monte Carlo simulations to obtain p-values is very complicated from the computational point of view, asymptotic approximations are implemented in **sparr** by setting `tolerate = T`,

```
# This takes roughly 5.47 minutes to be executed
st.risk <- spattemp.risk(f = f.num, g = d.den, tolerate = T)
```

We proceed to extract the unconditional and conditional risk slices,

```
risk.slices <- spattemp.slice(st.risk, tt = n.slices)
```

The variable `risk.slices` contains a list of lists of images, each of which corresponds to the requested times in n.slices and has the unconditional and conditional risk estimates and the p-values surfaces. We can plot the image estimates as follows:

```
plot.imlist(c(risk.slices$rr, risk.slices$rr.cond), equal.ribbon = T, ncols = 4,
            box = F, main = "", main.panel = c(timelabels[n.slices], rep("", 4)),
            mar.panel = c(0, 0, 1, 1), col = rocket(1200), ribmar = c(1, 3, 1, 2),
            ribwid = 0.4, panel.end = function(i,...){
              contour(c(risk.slices$P, risk.slices$P.cond)[[i]],
                      levels = 0.05, drawlabels = TRUE, ...)
              plot(X$window, add = T)})
```

In Figure 5, we can see how the high significant risk remains in the central Hampshire region. Although the two panels (top and bottom) look similar, there are differences. For example, in February 2001, we see how the conditional risk shows a larger statistically high area than its unconditional counterpart. We also see that the greatest risk is not present in the most densely populated areas (in this case, the south-central part of the region, where the capital is located).

Second-order descriptors

Pair correlation and K-functions

Let $\xi_1 = (\mathbf{u}_1, v_1), \xi_2 = (\mathbf{u}_2, v_2)$ be two points of $W \times T$. The second-order intensity function $(\lambda^{(2)}(\xi_1, \xi_2), \xi_1, \xi_2 \in W \times T)$ and the pair correlation function are defined in the spatio-temporal

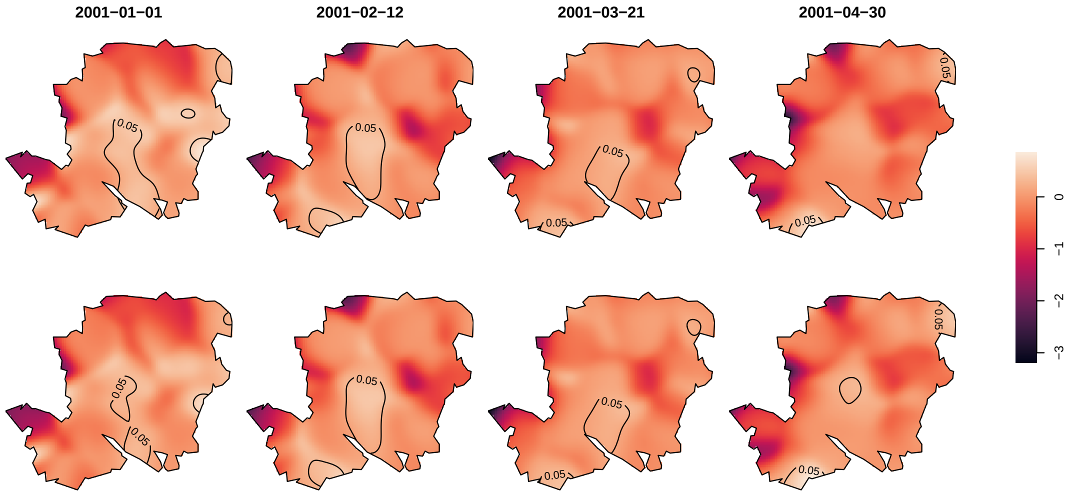


Figure 5: Temporal snapshots of the unconditional risk $\hat{r}(\mathbf{u}, v)$ (top) and conditional risk $\hat{r}(\mathbf{u}|v = t)$ (where t is 01/01/2001, 01/01/2002, 01/01/2003 and 12/31/2003) (bottom) of non-specific gastrointestinal disease in Hampshire. Solid lines delineate a statistical significant elevated risk at the 5% level.

context, in the very same way as the spatial ones. For CSTR point processes, the pair correlation function takes the value of one. Values other than these benchmarks indicate how likely a pair of "extra" points will appear in a CSTR process with the same intensity at the specified locations; i.e., larger values indicate clustering, and smaller values indicate regularity. The pair correlation (Gabriel and Diggle 2009; Møller 2012; González et al. 2016) is defined as

$$g(\xi_1, \xi_2) = \frac{\lambda^{(2)}(\xi_1, \xi_2)}{\lambda(\xi_1)\lambda(\xi_2)}, \qquad \xi_1, \xi_2 \in W \times T. \tag{6}$$

Second-order intensity reweighted stationarity (SOIRS) and isotropy are defined in the same vein as the spatial counterparts. For the estimation of the pair correlation function, consider a point pattern $X = \{(\mathbf{u}_i, v_i)\}_{i=1}^{n}$ with n points, the spatio-temporal pair correlation function (Eq. (6)) may be estimated by

$$\hat{g}(r,t) = \frac{1}{4\pi r} \sum_{i=1}^{n} \sum_{j \neq i} \frac{\kappa_\epsilon(\|\mathbf{u}_i - \mathbf{u}_j\| - r)\kappa'_\delta(|v_i - v_j| - t)}{\hat{\lambda}\left(\mathbf{u}_i, v_i\right) \hat{\lambda}\left(\mathbf{u}_j, v_j\right) w_{ij}}, \qquad r \in [\epsilon, r_0], t \in [\delta, t_0],$$

where κ_ϵ and κ'_δ are one-dimensional kernel functions with spatial and temporal bandwidths ϵ and δ, respectively; w_{ij} represent edge-correction weights (González et al. 2016). Selecting the bandwidths is often complicated because the estimation depends on those parameters, which means a wrong decision could be disastrous. As the second-order descriptors are distance-based, we can take advantage of some techniques for bandwidth selection. For example, for the temporal kernel, we use the `dpik()` function from the **KernSmooth** package (González and Moraga 2022b), which uses the method proposed by (Sheather and Jones 1991) to estimate a suitable bandwidth,

```
library(KernSmooth)
dt <- dist(unique(Times))
ht <- dpik(dt, kernel = "epanech", gridsize = 50, scalest = "iqr")
ht <- 2 * ht
```

Note that we set Epanechnikov's kernel as a default because the kernel selection is not as important as the bandwidth. We get a temporal bandwidth of 13.5 days (the output of the `dpik()` function). For the spatial bandwidth and upper bound r_0, we use the same as obtained for the spatial summary functions. For the temporal upper bound t_0, we set the 15% of the maximum temporal distances,

```
t0 <- 0.15 * max(dt)
```

We use the **stpp** package to estimate the descriptor. We first create data in spatio-temporal point format using the command `as.3dpoints()`; then, we create a two-column matrix to specify the polygonal region containing the points and set the spatial and temporal domains for the descriptor. Finally, we use the `PCFhat()` function to compute the estimate of the pair correlation function,

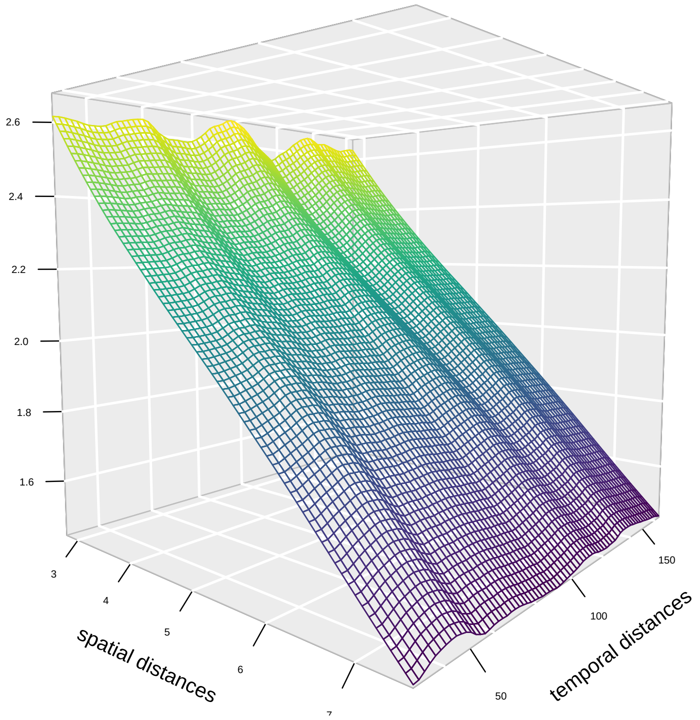


Figure 6: Spatio-temporal pair correlation function estimate of non-specific gastrointestinal disease data.

```
library(stpp)
FMD <- as.3dpoints(PP$x, PP$y, PP$marks$time)
s.region <- as.matrix(data.frame(x = PP$window$bdry[[1]]$x,
                                 y = PP$window$bdry[[1]]$y))
hs <- bwG
u1 <- seq(hs, r0, length.out = 71)[-1]
v1 <- seq(ht, t0, length.out = 71)[-1]

# Time consuming: This takes roughly 3.87 hours to be executed
g <- PCFhat(xyt = FMD, s.region = s.region, t.region = range(Times),
            lambda = lambda, dist = u1, times = v1, ks = "epanech",
            kt = "epanech", hs = hs, ht = ht)
```

To visualise the descriptor, we use the `persp3D()` function of the **plot3D** package,

```
library(plot3D)
par(mar = c(0,0,0,0))
persp3D(x = u1, y = v1, z = g$pcf, facets = NA, curtain = F, col =  viridis(200),
    colkey = F, bty = "g", pch = 20, cex = 1.5, theta = 40, phi = 5,
    border = NA, ticktype = "detailed", cex.axis = 0.5,  zlab = "",
    xlab = "spatial distances", ylab = "temporal distances")
```

The $\hat{g}(r,t)$ surface, shown in Figure 6, describes the spatio-temporal interaction structure of the disease; it shows clustering ($\hat{g}(r,t) \geq 1$) for all the distances at all times with the appearance of high clustering at small distances (up to 5km) that moves towards complete spatio-temporal randomness (CSTR) for more considerable distances. We then can suspect from this behaviour that the point pattern is only clustered in the spatial dimension, i.e., the interaction between gastrointestinal reports increases as long as they are close in space, but the clustering in time seems uniform along with the selected periods. We can also conclude that while the first-order intensity is not separable, perhaps the second-order one is. Note that the bandwidth selection explains the waves of the pair correlation function in the temporal dimension; narrow bandwidths produce spurious bumps in the estimate.

One of the most used second-order descriptor is the inhomogeneous spatio-temporal K-function Møller (2012). Under the SOIRS assumption, if $g(r,t)$ is the pair correlation function, then the K-function is defined as

$$K(r,t) = \int_{\mathbb{R}^2}\int_{\mathbb{R}} \mathbf{1}[||\mathbf{u}|| \leq r, |s| \leq t] g(||\mathbf{u}||, |s|) \mathrm{d}\mathbf{u}\mathrm{d}s, \quad \text{for } r > 0, t > 0.$$

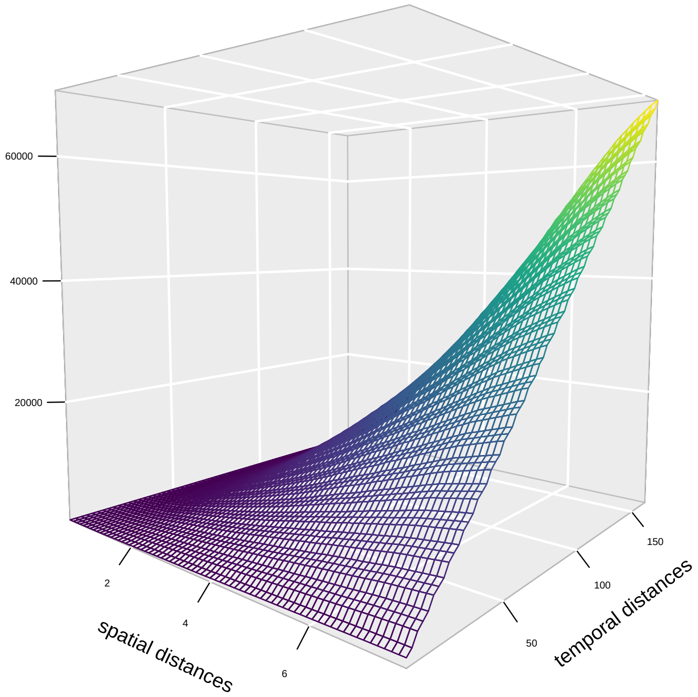


Figure 7: Spatio-temporal pair correlation function estimate of non-specific gastrointestinal disease data.

In the homogeneous case, $K(r,t)$ is simply the expected number of additional points within distance r and time lag t from the origin, given that the point pattern has a point at the origin. For a CSTR point processes $K(r,t) = \pi r^2 t$. Considering a spatio-temporal point pattern X again, an estimator of $K(r,t)$ is

$$\hat{K}(r,t) = \frac{1}{|W \times T|} \sum_{i=1}^{n} \sum_{j \neq i} \frac{\mathbf{1}\left[\left\|\mathbf{u}_i - \mathbf{u}_j\right\| \leq r\right] \mathbf{1}\left[\left|v_i - v_j\right| \leq t\right]}{\hat{\lambda}\left(\mathbf{u}_i, v_i\right) \hat{\lambda}\left(\mathbf{u}_j, v_j\right) w_{ij}}.$$

There is an alternative estimator $\hat{K}^*(r,t)$ matching the original definition of (Gabriel 2014) that into account the past of the process. To compute the K-function we employ the `STIKhat()` function from the **stpp** package,

```
u <- seq(0, r0, length.out = 71)
v <- seq(0, t0, length.out = 71)

# Time consuming: This takes roughly 55 minutes to be executed
stik <- STIKhat(xyt = FMD, s.region = s.region, t.region = range(Times),
                lambda = lambda, dist = u, times = v, infectious = F)
```

The parameter `infectious` controls the type of estimator, when it is set `FALSE`, past and future events are considered for the estimation. In this equation, the points $\xi_i = (\mathbf{u}_i, v_i)$ are ordered so that $v_i < v_{i+1}$ (ties are broken by random unrounding whenever necessary). To deal with temporal edge-effects, for each $t \in T = [T_0, T_1]$, n_t is the number of events for which $v_i \leq T_1 - t$. For spatial edge effects, the `STIKhat()` function employs Ripley's method (Gabriel and Diggle 2009). For plotting the spatio-temporal K-function we write,

```
KS <- stik$Khat - stik$Ktheo
par(mar = c(0,0,0,0))
persp3D(x = u[-1], y = v[-1], z = KS, facets = NA, zlab = "",
    curtain = F, col =  viridis(100), colkey = F, bty = "g", pch = 20,
    cex = 1.5, theta = 40, phi = 5, border = NA, ticktype = "detailed",
    cex.axis = 0.5, xlab = "spatial distances", ylab = "temporal distances")
```

Figure 7 shows the estimated K-function. Since the K-function is centred on its theoretical value $(2\pi r^2 t)$, we can easily interpret it; positive deviations suggest clustering, while negative deviations represent regularity or inhibition. In our case, we can see positive deviations that become larger as the spatial and temporal distances grow together. Therefore, as time passes and spatial distances become larger, the level of clustering also increases in our point pattern, at least in the range we have considered, that is, up to 7km in space and up to 5 months in time.

4 Discussion

Throughout this paper, we have shown how to analyse complex spatial and spatio-temporal point patterns through various techniques that aim to capture their first and second-order characteristics using several R packages. Specifically, we have toured very recent techniques such as variable bandwidths, relative risk inference, and composite hypotheses. The spatial methods presented are well-investigated and documented and can be easily applied using R. We have seen that some procedures, such as estimating *p*-values in composite hypotheses, require severe computational resources. The approximate computing time, including all the routines and set parameters suggested throughout the paper, is approximately six hours in a 3 GHz 6-Core Intel Core i5 iMac computer. This time depends on several factors, such as the number of simulations and the precision of desired outputs. It is mainly due to the number of points in the point pattern; the more data points, the more computational complexity, especially in the case of second-order descriptors, where pairwise calculations are required. Some R packages such as **foreach** can help accelerate the computations, even though these methods may quickly become unfeasible when the datasets have many points.

We have presented the most straightforward methods and the most widely known in the literature; we have done this, above all, because we intended to provide a solid and approachable learning resource. However, there are also alternative methods to deal with some of the descriptors we analyse in this document; for example, González and Moraga (2022a) presents a methodology to accurately and adaptively estimate the spatio-temporal intensity functions; this methodology can be applied using the **kernstadapt** package.

The analyses presented here can benefit future researchers who have initial questions about spatio-temporal point process data; these results can serve, for instance, as input for possible statistical models. We have not delved into statistical modelling and inference since many models exist in the spatio-temporal point processes literature. These statistical models can incorporate current knowledge of the phenomenon and many scientific considerations. Different models can be proposed and compared to decide which variables significantly influence the summary descriptors, making the results more informative (Adrian Baddeley, Rubak, and Turner 2015). For example, one of the most flexible families of models with the most methodological advantages is the family of log-Gaussian point processes. Log-Gaussian Cox processes are models for point patterns that intend to capture the environmental variability (Peter J. Diggle et al. 2013). They are spatio-temporal Poisson processes conditional on the realisation $\lambda(\mathbf{u}, v)$ of a stochastic intensity function $\Lambda(\mathbf{u}, v)$, $(\mathbf{u}, v) \in \mathbb{R}^2 \times \mathbb{R}$. In the SOIRS case, a convenient representation of the intensity is

$$\log\{\Lambda(\mathbf{u}, v)\} = \lambda(\mathbf{u}, v) S(\mathbf{u}, v),$$

where $S(\mathbf{u}, v)$ is a Gaussian stationary process with expectation 1 and covariance function $\gamma(r, t) = \sigma^2 s(r, t)$, where σ^2 is the variance of $S(\mathbf{u}, v)$ and $s(\cdot, \cdot)$ is a spatio-temporal correlation function, where r and t are spatial and temporal distances. Inference uses a discretised version of $S(\mathbf{u}, v)$, namely $\mathcal{S}$, defined on a regular grid, where observations X are the cell counts, and the intensity of the process is assumed constant within each cell (B. Taylor and Diggle 2014). $\mathcal{S}$ is a finite collection of Gaussian random variables, so its properties imply a multivariate Gaussian density. The covariance matrix elements are calculated by evaluating $\gamma(r, t)$ at the centroids of the cells. Predictive inference about $\mathcal{S}$ requires samples from the conditional distribution of the latent field given the observations, namely $[\mathcal{S}|X]$. For estimating the parameters of a space-time log-Gaussian Cox process, one can choose between computationally feasible methods, such as those based on moments or more demanding methods based on likelihood (Peter J. Diggle et al. 2013). Inference can be achieved for example, by using Markov chain Monte Carlo (MCMC) for which the **lgcp** package can be used, and by using integrated nested Laplace approximation (INLA) with the **inla** and **inlabru** packages.

To conclude, the methods presented here are primarily concerned with understanding the probability distribution that generate points observed in space or time. The study of such a distribution involves many routines, concepts and contexts that can help answer scientific questions and make decisions based on the data or inferences about them. The inclusion of technological advancements in R has led to improved efficiency and speed in calculations. Additionally, R offers a versatile and reliable environment for implementing statistical procedures and creating graphical visualisations. Also, modern developments in spatial statistics related to point patterns have been made possible thanks to the widespread availability of reproducible R scripts for analyses. In this paper, we have made extensive use of the ability to apply, extend and modify procedures in R to analyse spatial and spatio-temporal point patterns in the context of epidemiology. These approaches are also helpful for analysing point pattern data that arise in various fields such as climatology, ecology or seismology.

References

Bachl, Fabian E, Finn Lindgren, David L Borchers, and Janine B Illian. 2019. "inlabru: An R Package for Bayesian Spatial Modelling from Ecological Survey Data." *Methods in Ecology and Evolution* 10 (6): 760–66.

Baddeley, Adrian, Andrew Hardegen, Thomas Lawrence, Robin K. Milne, Gopalan Nair, and Suman Rakshit. 2017. "On Two-Stage Monte Carlo Tests of Composite Hypotheses." *Computational Statistics & Data Analysis* 114: 75–87.

Baddeley, Adrian, Ege Rubak, and Rolf Turner. 2015. *Spatial Point Patterns: Methodology and Applications with r*. Chapman & Hall Interdisciplinary Statistics Series. CRC Press, Boca Raton, Florida.

Baddeley, A, and R Turner. 2005. "spatstat: An R Package for Analyzing Spatial Point Patterns." *Journal of Statistical Software* 12(6): 1–42.

Besag, Julian. 1977. "Contribution to the Discussion of Dr Ripley's Paper." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 39: 193–95.

Bivand, Roger S., Edzer Pebesma, and Virgilio Gómez-Rubio. 2013. *Applied Spatial Data Analysis with R*. Second. Springer, New York.

Choi, Edwin, and Peter Hall. 1999. "Nonparametric Approach to Analysis of Space-Time Data on Earthquake Occurrences." *Journal of Computational and Graphical Statistics* 8: 733–48.

Daley, D., and D. Vere-Jones. 2003. *An Introduction to the Theory of Point Processes: Volume i: Elementary Theory and Methods*. Second. Springer-Verlag, New York.

Davies, T. M., J. C. Marshall, and M. L. Hazelton. 2018. "Tutorial on Kernel Estimation of Continuous Spatial and Spatiotemporal Relative Risk." *Statistics in Medicine* 37 (7): 1191–221.

Davies, Tilman M., and Adrian Baddeley. 2018. "Fast Computation of Spatially Adaptive Kernel Estimates." *Statistics and Computing* 28 (4): 937–56.

Diggle, P. J. 1985. "A Kernel Method for Smoothing Point Process Data." *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 34 (2): 138–47.

———. 2013. *Statistical Analysis of Spatial and Spatio-Temporal Point Patterns*. Third. Chapman & Hall Monographs on Statistics & Applied Probability. CRC Press, Boca Raton, Florida.

Diggle, Peter J., Paula Moraga, Barry Rowlingson, and Benjamin Taylor. 2013. "Spatial and Spatio-Temporal Log-Gaussian Cox Processes: Extending the Geostatistical Paradigm." *Statistical Science* 28: 542–63.

Diggle, P., L Knorr-Held, Rowlingson B, T Su, Pand Hawtin, and T Bryant. 2003. "On-Line Monitoring of Public Health Surveillance Data." In *Monitoring the Health of Populations: Statistical Principlesand Methods for Public Health Surveillance*, edited by Brookmeyer R. and Stroup D. F., 233–66. Oxford University Press.

Gabriel, Edith. 2014. "Estimating Second-Order Characteristics of Inhomogeneous Spatio-Temporal Point Processes." *Methodology and Computing in Applied Probability* 16: 411–31.

Gabriel, Edith, and Peter J. Diggle. 2009. "Second-Order Analysis of Inhomogeneous Spatio-Temporal Point Process Data." *Statistica Neerlandica* 63 (1): 43–51.

Gabriel, Edith, Barry Rowlingson, and Peter J. Diggle. 2013. "stpp: An R Package for Plotting, Simulating and Analyzing Spatio-Temporal Point Patterns." *Journal of Statistical Software* 53(2) (2): 1–29.

Garnier, Simon, Simon, Noam Ross, Bob Rudis, Marco Sciaini, Antonio Pedro, and Cedric Scherer. 2023. `viridis(Lite)` - *Colorblind-Friendly Color Maps for* `R`. https://doi.org/10.5281/zenodo.4679424.

Ghorbani, Mohammad, Nafiseh Vafaei, Jiří Dvořák, and Mari Myllymäki. 2021. "Testing the First-Order Separability Hypothesis for Spatio-Temporal Point Patterns." *Computational Statistics & Data Analysis* 161: 107245.

González, Jonatan A., Ute Hahn, and Jorge Mateu. 2019. "Analysis of Tornado Reports Through Replicated Spatiotemporal Point Patterns." *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 69 (1): 3–23.

González, Jonatan A., and Paula Moraga. 2022a. "An Adaptive Kernel Estimator for the Intensity Function of Spatio-Temporal Point Processes." arXiv. https://doi.org/10.48550/ARXIV.2208.12026.

———. 2022b. "`kernstadapt`: Spatio-Temporal Adaptive Kernel Estimators for Intensities." https://doi.org/https://CRAN.R-project.org/package=kernstadapt.

González, Jonatan A., Francisco J. Rodríguez-Cortés, Ottmar Cronie, and Jorge Mateu. 2016. "Spatio-Temporal Point Process Statistics: A Review." *Spatial Statistics* 18: 505–44.

Harte, David. 2010. "PtProcess: An R Package for Modelling Marked Point Processes Indexed by Time." *Journal of Statistical Software* 35 (8): 1–32.

Hazelton, Martin L., and Tilman M. Davies. 2009. "Inference Based on Kernel Estimates of the Relative

Risk Function in Geographical Epidemiology." *Biometrical Journal* 51 (1): 98–109.

Illian, J., P. A. Penttinen, H. Stoyan, and D. Stoyan. 2008. *Statistical Analysis and Modelling of Spatial Point Patterns*. Statistics in Practice. Wiley.

Jalilian, Abdollah, and Rasmus Waagepetersen. 2018. "Fast Bandwidth Selection for Estimation of the Pair Correlation Function." *Journal of Statistical Computation and Simulation* 88 (10): 2001–11.

Kelsall, Julia E., and Peter J. Diggle. 1995. "Kernel Estimation of Relative Risk." *Bernoulli* 1 (1/2): 3–16.

Microsoft, and Steve Weston. 2022a. `doParallel`*: Foreach Parallel Adaptor for the 'Parallel' Package*. https://CRAN.R-project.org/package=doParallel.

———. 2022b. `foreach`*: Provides Foreach Looping Construct*. https://CRAN.R-project.org/package=foreach.

Møller, M, J. And Ghorbani. 2012. "Aspects of Second-Order Analysis of Structured Inhomogeneous Spatio-Temporal Point Processes." *Statistica Neerlandica* 66: 472–91.

Moraga, Paula. 2019. *Geospatial Health Data: Modeling and Visualization with R-INLA and Shiny*. Biostatistics Series. Chapman & Hall/CRC, Boca Raton, Florida.

———. 2021. "Species Distribution Modeling Using Spatial Point Processes: A Case Study of Sloth Occurrence in Costa Rica." *The R Journal* 12 (2): 293–310.

Moraga, Paula, and Francisco Montes. 2011. "Detection of Spatial Disease Clusters with LISA Functions." *Statistics in Medicine* 30 (10): 1057–71.

Myllymäki, Mari, and Tomáš Mrkvička. 2020. "GET: Global Envelopes in R." *arXiv:1911.06583 [Stat.ME]*. https://arxiv.org/abs/1911.06583.

Myllymäki, Mari, Tomáš Mrkvička, Pavel Grabarnik, Henri Seijo, and Ute Hahn. 2017. "Global Envelope Tests for Spatial Processes." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 79 (2): 381–404.

Pebesma, E. J., and Roger S. B. 2005. "Classes and Methods for Spatial Data in R." *R News* 5 (2): 9–13.

Pebesma, Edzer. 2018. "Simple Features for R: Standardized Support for Spatial Vector Data." *The R Journal* 10 (1): 439–46.

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

Ripley, B D. 1977. "Modelling Spatial Patterns (with Discussion)." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 39 (2): 172–212.

Rodrigues, Alexandre, and Peter J. Diggle. 2012. "Bayesian Estimation and Prediction for Inhomogeneous Spatiotemporal Log-Gaussian Cox Processes Using Low-Rank Models, with Application to Criminal Surveillance." *Journal of the American Statistical Association* 107 (497): 93–101.

Rowlingson, B. S, and P. J Diggle. 1993. "splancs: Spatial Point Pattern Analysis Code in S-PLUS." *Computers & Geosciences* 19 (5): 627–55.

Rue, Håvard, Sara Martino, and Nicolas Chopin. 2009. "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): 319–92.

Scott, David W. 2015. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Second. Wiley Series in Probability and Statistics. John Wiley & Sons.

Sheather, S. J., and M. C. Jones. 1991. "A Reliable Data-Based Bandwidth Selection Method for Kernel Density Estimation." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 53 (3): 683–90.

Silverman, Bernard W. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall Monographs on Statistics & Applied Probability. CRC press, Boca Raton, Florida.

Taylor, Benjamin M, Tilman M Davies, Barry S Rowlingson, and Peter J Diggle. 2013. "Lgcp: An R Package for Inference with Spatial and Spatio-Temporal Log-Gaussian Cox Processes." *Journal of Statistical Software* 52: 1–40.

Taylor, Benjamin, and Peter Diggle. 2014. "INLA or MCMC? A Tutorial and Comparative Evaluation for Spatial Prediction in Log-Gaussian Cox Processes." *Journal of Statistical Computation and Simulation* 84 (10): 2266–84.

Turner, Rolf. 2009. "Point Patterns of Forest Fire Locations." *Environmental and Ecological Statistics* 16 (2): 197–223.

Venables, W. N., and B. D. Ripley. 2002. *Modern Applied Statistics with s*. Fourth. Statistics and Computing. Springer-Verlag, New York.

Wässle, H, Brian Blundell Boycott, and R-B Illing. 1981. "Morphology and Mosaic of on-and Off-Beta Cells in the Cat Retina and Some Functional Considerations." *Proceedings of the Royal Society of London. Series B. Biological Sciences* 212 (1187): 177–95.

Wickham, Hadley. 2016. `ggplot2`*: Elegant Graphics for Data Analysis*. Second. Springer-Verlag, New York.

Wong, Ka Yiu, and Dietrich Stoyan. 2021. "Poles of Pair Correlation Functions: When They Are Real?" *Annals of the Institute of Statistical Mathematics* 73: 425–40.

Jonatan A. González

King Abdullah University of Science and Technology (KAUST)
Computer, Electrical and Mathematical Sciences and Engineering Division
Thuwal, 23955-6900, Saudi Arabia
ORCiD: 0000-0002-2296-5271
`jonathan.gonzalez@kaust.edu.sa`

Paula Moraga
King Abdullah University of Science and Technology (KAUST)
Computer, Electrical and Mathematical Sciences and Engineering Division
Thuwal, 23955-6900, Saudi Arabia
`http://www.paulamoraga.com`
ORCiD: 0000-0001-5266-0201
`paula.moraga@kaust.edu.sa`

nlmeVPC: Visual Model Diagnosis for the Nonlinear Mixed Effect Model

by Eun-Hwa Kang, Myungji Ko, and Eun-Kyung Lee

Abstract A nonlinear mixed effects model is useful when the data are repeatedly measured within the same unit or correlated between units. Such models are widely used in medicine, disease mechanics, pharmacology, ecology, social science, psychology, etc. After fitting the nonlinear mixed effect model, model diagnostics are essential for verifying that the results are reliable. The visual predictive check (VPC) has recently been highlighted as a visual diagnostic tool for pharmacometric models. This method can also be applied to general nonlinear mixed effects models. However, functions for VPCs in existing R packages are specialized for pharmacometric model diagnosis, and are not suitable for general nonlinear mixed effect models. In this paper, we propose **nlmeVPC**, an R package for the visual diagnosis of various nonlinear mixed effect models. The **nlmeVPC** package allows for more diverse model diagnostics, including visual diagnostic tools that extend the concept of VPCs along with the capabilities of existing R packages.

1 Introduction

After fitting a model, diagnosing the fitted model is essential for verifying that the results are reliable (Nguyen et al. 2017). For linear models, the residuals are usually used to determine the goodness of fit of the fitted model. However, due to random effects and nonlinearity, residuals are less useful for diagnosing fit in nonlinear mixed effects models. Therefore, various diagnostic tools for this type of model have been developed. The nonlinear mixed effect model is useful when the data are repeatedly measured within the same unit, or the relationship between the dependent and independent variables is nonlinear. It is widely used in various fields, including medicine, disease mechanics, pharmacology, ecology, pharmacometrics, social science, and psychology (J. C. Pinheiro and Bates 2000; Davidian 2017). Recently, among the various diagnostic tools applicable to nonlinear mixed models, simulation-based diagnostic methods have been developed in the field of pharmacology (M. Karlsson and Savic 2007). The visual predictive check (VPC; M. O. Karlsson and Holford (2008)) is a critical diagnostic tool that visually tests for the adequacy of the fitted model. It allows for the diagnosis of fixed and random effects in mixed models (M. O. Karlsson and Holford 2008) in the original data space. Currently, it is a widely used method for diagnosing population pharmacometric models: Heus et al. (2022) used the VPC method to evaluate their population pharmacokinetic models of vancomycin, Mudde et al. (2022) checked their final population PK model for each regimen of antituberculosis drug using the VPC method, and Otto et al. (2021) compares the predictive performance of parent-metabolite models of (S)-ketamine with the VPC method. This VPC method can be used for general nonlinear mixed effect models, including hierarchical models.

The **psn** (Lindbom, Ribbing, and Jonsson 2004) and **xpose4** (R. J. Keizer, Karlsson, and Hooker 2013) packages provide various diagnostic methods for pharmacometric models in R (R Core Team 2023), including the VPC plot. These packages were developed only for the pharmacometricians, and the users needed to use the NONMEM software (Bauer 2011) for generating inputs of functions in these two packages. However, NONMEM is licensed software, and it is mainly designed towards the analysis of population pharmacometric models. Therefore, it is not easy for nonpharmacometrician to use NONMEM and to draw the VPC plot through **psn** and **xpose4**. Recently, the **vpc** (R. Keizer 2021) package and **nlmixr2** (Fidler et al. 2019) package have been developed to draw VPC plots in R without results from NONMEM. However, **vpc** package provides the function for drawing the original VPC plot only. **nlmixr2** was developed initially for fitting general dynamic models. To check the fitted model, **nlmixr2** provides a function to use graphical diagnostics in **xpose4** with the `nlmixr2` model object. Also, **nlmixr2** uses the VPC plot in the **vpc** package with the `nlmixr2` model object. Therefore, both packages only provide a function to draw the basic VPC plot, and the other newly developed simulation-based methods, including extensions of VPC, are not provided.

We have developed a new R package, **nlmeVPC**, to provide a suite for various visual checking methods for the nonlinear mixed models. This package includes various state-of-the-art model diagnostic methods based on the visual comparison between the original and simulated data. Most methods compare the statistics calculated from the observed data to the statistics from the simulated data. Percentiles, for example, the 10^{th}, 50^{th}, and 90^{th} percentiles, are widely used to summarize relevant statistics from the observed and simulated data. We compare the similarities between the statistics from the observed data and those from the simulated data in two different spaces: the data space and the model space (Hadley Wickham, Cook, and Hofmann 2015). The original data comprise the data space. Usually, the data space is represented by the independent and dependent variables, such as time and blood concentration, in the pharmacokinetic data. On the other hand, the model space is composed of quantities obtained from the fitted model, for example, residuals and summary values from the fitted model. From this viewpoint, we categorize the well-known visual diagnostic tools into two categories. One method compares the observed and simulated data in the original data space, and the other in the model space. In the method with the data space, we developed functions for the original VPC (`VPCgraph`), the additive quantile regression VPC (`aqrVPC`), and the bootstrap VPC (`bootVPC`). In addition, we proposed a new VPC method: the average shifted VPC (`asVPC`). In the method with the model space, the coverage plot (`coverageplot`) and the quantified VPC (`quantVPC`) are included. For a more detailed diagnosis, we developed a coverage detailed plot (`coverageDetailplot`). In **nlmeVPC**, the **ggplot2** package (H. Wickham 2016) is used to create all plots.

2 Nonlinear mixed effect model

The general nonlinear mixed effect model is defined as follow:

$$y_{ij} = f(x_{ij}, \theta, \eta_i, \epsilon_{ij})\eta_i \sim N(0, \Omega)\epsilon_{ij} \sim N(0, \Sigma)$$

where y_{ij} is the dependent variable of the j^{th} observation of the i^{th} individual, x_{ij} is the independent variable, f is the nonlinear function of x_{ij}, θ is the population parameter, η_i represents the variability of the individual i, and ϵ_{ij} represents the random error. From the data, θ , Ω, and Σ are estimated. For the simulated data, the fitted model $y_{ij} = f(x_{ij}, \hat{\theta}, \eta_i, \epsilon_{ij})$, $\eta_i \sim N(0, \hat{\Omega})$, $\epsilon_{ij} \sim N(0, \hat{\Sigma})$ are used.

3 Validation in the data space

The VPC is the most popular model validation method in the pharmacometrics area. It was developed to diagnose population pharmacokinetic/pharmacodynamic models visually. The main idea of the VPC is to compare the distribution of observations and the distribution of predicted values, where the distribution of predicted values is obtained from simulated data drawn from the fitted model. If the fitted model explains the observed data well, these two distributions should be similar. Both distributions can be represented in the original data space that consists X axis as the independent variable and Y axis as the dependent variable. It allows us to compare the observed data with the fitted model in the original data space. In **nlmeVPC**, we include an original VPC plot, an additive quantile regression VPC (Jamsen et al. 2018), and a bootstrap VPC (Post et al. 2008). We also proposed a new approach to draw the VPC: the average shifted VPC.

3.1 Visual Predictive Check

The visual predictive check (VPC; M. O. Karlsson and Holford (2008)) is based on the principle that if the fitted model adequately describes the observed data, the distribution of the simulated data from the fitted model should be similar to the distribution of the observed data. There are several ways to compare the similarities between the distributions. In the VPC approach, profiles of quantiles are used. Two profiles are mainly used to compare the distributions of observations and predictions. One profile is from the upper bound of the prediction intervals, and the other is from the lower bound. These prediction intervals are calculated from the simulated data. 90% prediction intervals are usually used. For small and sparse samples, 80% prediction interval is also used. The lower and upper bounds of 80% prediction interval are the 10^{th} and 90^{th} percentiles of the simulated data. Figure 1(A) shows the "scatter" type of the VPC plot. Dots indicate the observed data. Two dashed blue lines represent profiles of the 10^{th} and 90^{th} percentiles of the simulated data, and the solid blue line represents the 50^{th} percentile. If the fitted model represents the observed data well, most observed data should lie between profiles of 10^{th} and 90^{th} percentiles.

Figure 1(B) is the "percentile" type of the VPC plot. In this plot, profiles of percentiles from the observed data are compared to profiles of percentiles from the simulated data. Two dashed red lines represent profiles of the 10^{th} and 90^{th} percentiles of the observed data, and the solid red line represents profiles of the 50^{th} percentile of the observed data. If the fitted model represents the observed data well, two profiles in each percentile - one from the original data and the other from the simulated data - are similar.

Figure 1(C) is the "CI" type of the VPC plot. The solid red line represents the 50^{th} percentile of the observed data, and dashed red lines represent the 10^{th} and 90^{th} percentiles of the observed data.

Light blue areas represent the 95% confidence areas of the 10^{th} and 90^{th} percentiles, and pink areas represent the 95% confidence areas of the 50^{th} percentile. These confidence areas were calculated from the simulated data. After calculating percentiles in each simulated data, we find 95% confidence intervals for each percentile and use this to draw the areas. In this plot, it is necessary to verify that the profiles of the original data are in confidence areas of each profile from the simulated data in each percentile. If each percentile line of the observed data is in the corresponding confidence area, this can be evidence that the fitted model represents the observed data quite well. Otherwise, the fitted model needs to be improved. The "CI" type of the VPC plot is the most widely used type in pharmacometrics.

The percentiles of the dependent variable are calculated in each bin. To estimate the percentiles accurately, enough data points need to lie within each bin. No binning means that the number of bins is equal to the number of different independent variable values. In this case, the VPC plot shows all details of the relationship between the independent and dependent variables. However, the resulting lines become too irregular to show meaningful trends or patterns.

As the number of bins decreases, the lines become smoother and more regular, however this can come at the loss of information if too much smoothing is used. Therefore, the selection of the best number of bins is crucial. The way of determining cutoffs for bin also plays an important role. Lavielle and Bleakley (2011) proposed a procedure for finding the optimal bin configuration automatically, including the optimal number of bins and the optimal bin cutoffs. `VPCgraph` provides the automatic binning with `optK` and `makeCOVbin`; here, `optK` finds the optimal number of bins, and `makeCOVbin` finds the optimal cutoffs of bins using Lavielle and Bleakley's method.

3.2 Additive quantile regression VPC

To overcome the difficulties of making bins as well as determining the number of bins, Jamsen et al. (2018) used additive quantile regression to calculate the quantiles of the observed and simulated data. This regression method makes it possible to estimate quantiles without discrete binning, which is especially useful when the data are insufficient, irregular, or inappropriate to configure the bins. To fit the additive quantile regression, we used the `rqss` function in the **quantreg** (Koenker 2023) package and developed the `aqrVPC` function to draw the VPC type plot with additive quantile regression. Figure 2 shows the additive quantile regression VPC plot. The solid and dashed lines represent the 10^{th}, 50^{th}, and 90^{th} additive quantile regression lines of the observed data, and the pink and light blue areas represent the confidence areas of the additive quantile regression lines of the simulated data. Lines and areas in the additive quantile regression VPC plot are much smoother than those in the original VPC plot.

3.3 Bootstrap VPC

The bootstrap VPC (Post et al. 2008) compares the distribution of the simulated data to the distribution of the bootstrap samples drawn from the observed data. This plot reflects the uncertainty of the observed data and allows for more objective comparisons with the predicted median.

Figure 3 shows the bootstrap VPC plot using `bootVPC`. The solid and dashed blue lines represent the 10^{th}, 50^{th}, and 90^{th} percentiles of the simulated data. The solid red line represents the 50^{th} percentile line, and the pink areas represent the 95% confidence areas of the 50^{th} percentile line, calculated from the bootstrap samples of the observed data. If the solid blue line and the solid red line are similar, the solid blue line is in the pink area, and the pink area is located between two dashed blue lines, then this is evidence that the fitted model fit the observed data well.

3.4 Average shifted VPC

Even though binning mitigates the problem with highly irregular data, the VPC plot still has a precision problem with sparse data. In this paper, we propose a new approach to draw the adapted VPC plot from the average shifted histograms (Scott 1985). A histogram is a widely used method for displaying the density of a single continuous variable. However, histograms can look quite different based on different choice of bin width and anchor. This requires computing an optimal bin width. To overcome the problem with the choice of bin width and to obtain smoother estimates, Scott (1985) proposed the averaged shifted histogram (ASH). The idea behind ASH is to make K different histograms with different anchors and to combine them via a weighted average. For each histogram, the starting point is shifted by d/K, where d is the width of the bin. We extend this idea to the VPC graph and propose the average shifted visual predictive check (asVPC) plot.

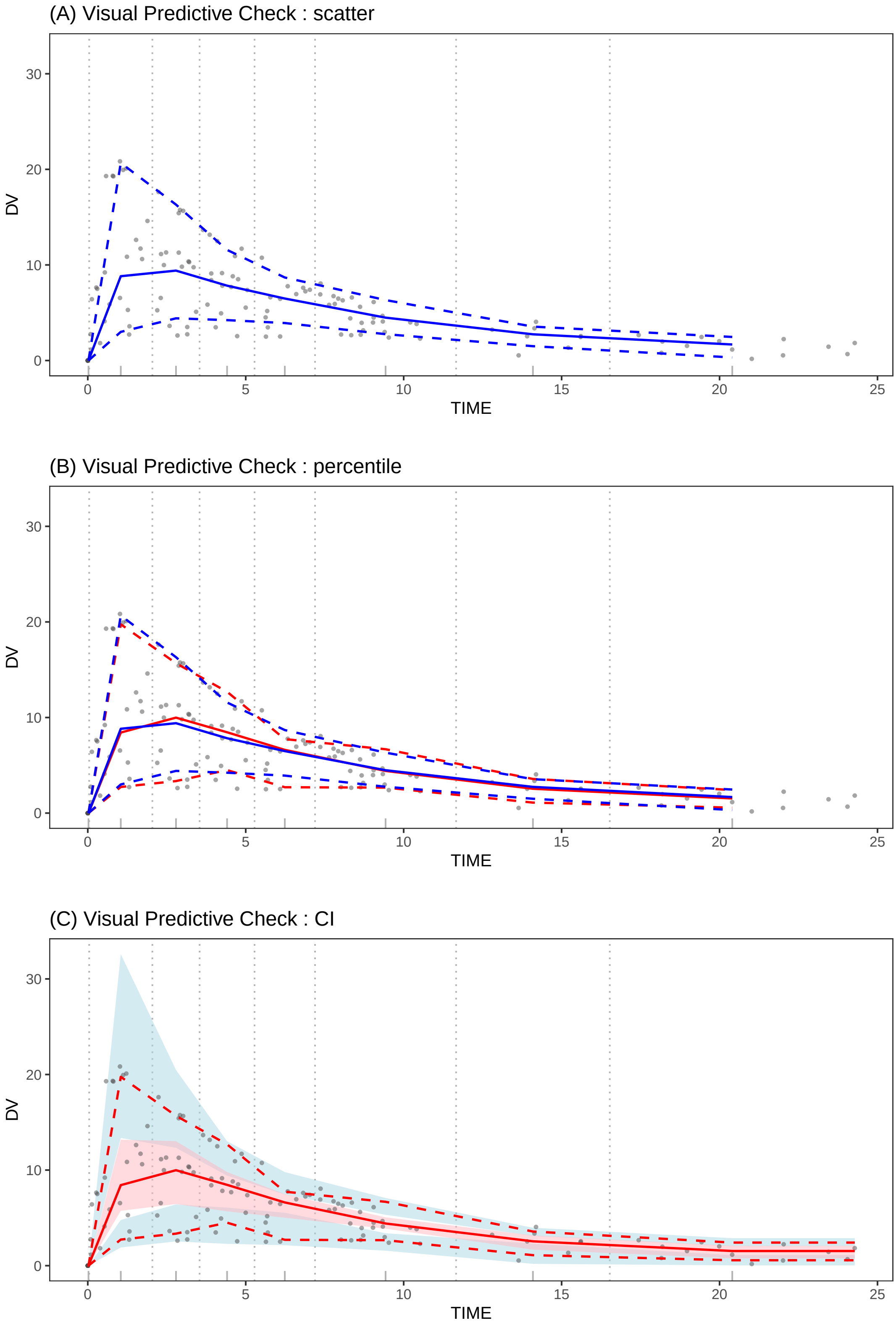


Figure 1: The visual predictive check plot. The solid red line represents the 50th percentile of the observed data, and dashed red lines represent the 10th and 90th percentiles of the observed data. The solid blue line represents the 50th percentile of the simularted data, and dashed blue lines represent the 10th and 90th percentiles of the simulated data. Light blue and pink areas represent the 95% confidence areas of the 10th, 50th and 90th percentile lines.

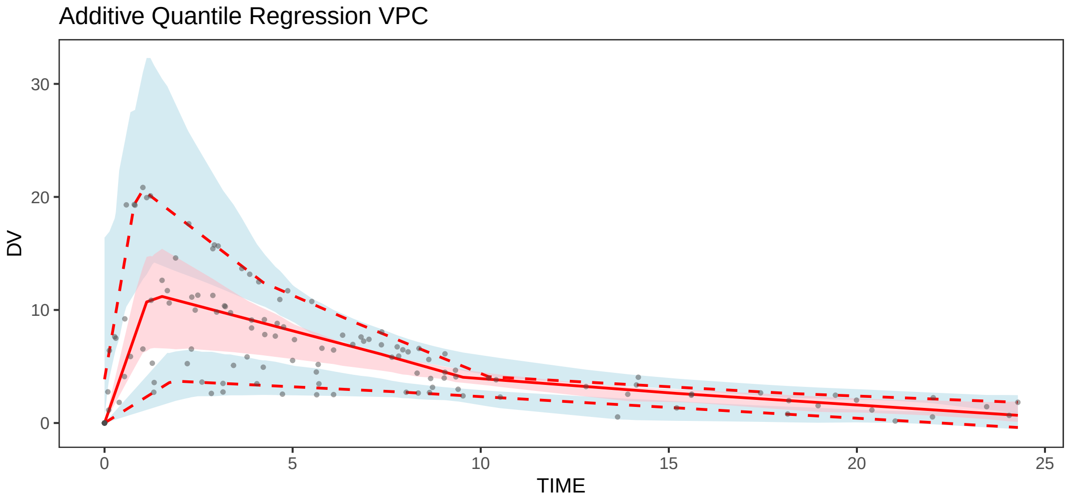


Figure 2: The additive equantile VPC plot. Dots indicate the observed data. The solid and dashed blue lines represent the 10^{th}, 50^{th}, and 90^{th} percentiles of the simulated data. The solid red line represents the 50^{th} percentile line. Light blue and pink areas represent the 95% confidence areas of the 10^{th}, 50^{th} and 90^{th} percentile lines.

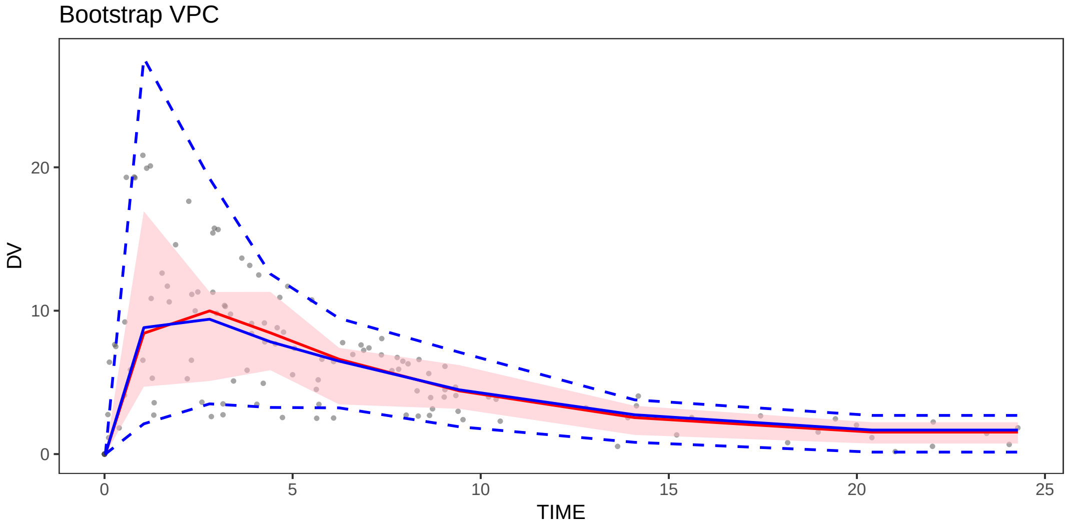


Figure 3: The bootstrap VPC plot. Dots indicate the observed data. The solid and dashed blue lines represent the 10^{th}, 50^{th}, and 90^{th} percentiles of the simulated data. The solid red line represents the 50^{th} percentile line, and the pink areas represent the 95% confidence areas of the 50^{th} percentile line, calculated from the bootstrap samples of the observed data.

The binning of the VPC varies from the traditional binning in a histogram. The width of the bins is determined such that each bins contains the same, fixed number of observations, and as such the width of each bin is different. To apply the ASH idea to the VPC, we divide our data into $K * B$ bins along with the independent variable, where each bin has a different width but the same number of observations. Here, B is the number of the original bins in the VPC plot, and K is the number of the histograms averaged in the asVPC plot.

To draw the VPC plots, the following information is needed from each bin:

(A) the median of the independent variable and percentiles of the dependent variable of the observed data.

(B) the median of the independent variable and percentiles of the dependent variable of the simulated data.

(C) 95% confidence interval of percentiles (usually 10^{th}, 50^{th}, and 90^{th} percentiles) of the dependent variable, calculated from the simulated data.

These values are also needed to produce confidence areas and percentile lines in the asVPC plot. Furthermore, additional steps are needed for asVPC. To find the median of the independent variable and percentiles of the dependent variable using the ASH algorithm, the following procedures are needed:

1. Divide the independent variable into $N = K * B$ bins.
2. For $i = 1, \cdots, N$,
   a) If $i < K$, combine $bin_1, \cdots, bin_{i+K-1}$\ else if $K \leq i \leq N - K$, combine $bin_{i-K+1}, \cdots, bin_{i+K-1}$\ else if $N - K < i$, combine $bin_{i-K+1}, \cdots, bin_N$
   b) Calculate the median of the independent variable and the weighted percentiles of the dependent variable in the combined bin
3. Collect the medians of the independent variable and the weighted percentiles of the dependent variable from 2, and connect them to the lines.

We can implement (A) and (B) by applying these procedures separately to the observed and simulated data. Additionally, (C) can be implemented using these procedures for each simulated dataset. First, we find the weighted percentiles, combine the results from each simulated dataset, and then calculate the 95% confidence intervals of each percentile. Using these three quantities (A), (B), and (C), we can draw the VPC type plot with the ASH approach, producing the asVPC plot.

Determining the weights

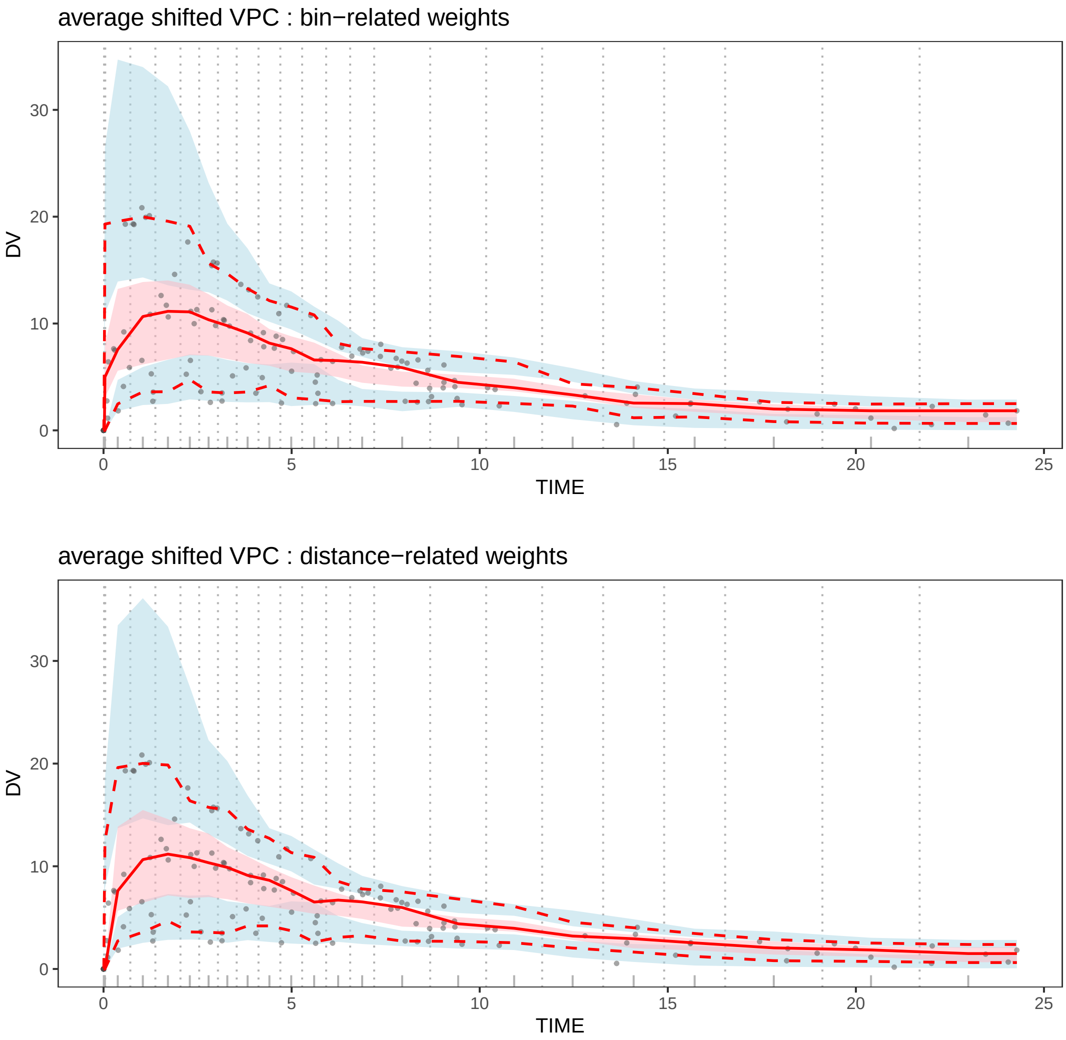
In the asVPC plot, the observations in each bin are combined using weights. Typically, the data near the center of the integrated bin have higher weights, and the data far from the center have smaller weights. This idea is used in the ASH algorithm as well as the density estimation literature. We suggest two different ways to apply weights for the asVPC calculation.

- Bin-related weight: For each bin in the combined bin, K-1 bins in both directions have weights and the other bins have no weight. Use $\frac{1}{K}, \cdots, \frac{i-1}{K}, \cdots, \frac{K-1}{K}, 1, \frac{K-1}{K}, \cdots, \frac{i-1}{K}, \cdots, \frac{1}{K}$ as weights for 2K-1 bins. This approach is the same as the original ASH approach.
- Distance-related weight: Use the reciprocal of the distance from the center of the independent variable in each combined bin so that the points near the center have higher weights and the points far from the center have lower weights.

Figure 4 shows the results from the `asVPC` function using bin-related weights and distance-related weights. The solid and dashed lines represent the average shifted quantile lines of the observed data, and the pink and light blue areas represent the confidence areas of the simulated data. The lines in the asVPC plot are smoother than those in the original VPC plot, and the confidence areas in the asVPC plot are thinner than those in the original VPC plot.

4 Validation in the model space

To validate the fitted model in the model space, we need to choose appropriate statistics to visualize and describe them in the model space. In **nlmeVPC**, we use the same statistics, that is quantiles, as the VPC plot in the data space, and compare them in two ways - numerically and visually. The numerical



average shifted VPC : bin−related weights

average shifted VPC : distance−related weights

Figure 4: The average shifted VPC plot. Dots indicate the observed data. The solid line represents the 50th quantiles of the observed data, and dashed lines represent the 10^{th} and 90^{th} percentiles of the observed data. Light blue and pink areas represent the 95% confidence areas of the 10^{th}, 50^{th}, and 90^{th} percentiles.

predictive check and the coverage plot are two commonly used methods in pharmacometrics. However, there is a limitation to detecting the illness of the fitted model. These methods combine the results of all ranges of the independent variable, and it is helpful to see the overall fitness. However, it is not easy to detect the detailed discrepancy between the fitted model and the observed data. To overcome this limitation, we developed a new plot, the coverage detailed plot, to compensate for the shortness of the coverage plot.

4.1 Numerical predictive check

The VPC plot visually compares the observed data to the simulated percentiles in the data space. On the other hand, the numerical predictive check (NPC; Wang and Zhang (2012); M. Karlsson and Savic (2007)) numerically compares the observed data to the simulated data. For a given level of prediction (for example, 90%), the predicted interval is calculated using the simulated data, and the number of the observed data points outside of the prediction interval is counted, both below the lower bound and above the upper bound. The expected number of points below the lower bound of the predicted interval (for example, 5% of observations) is also calculated and compared to the observed number. If these two numbers are similar, the suggested model is suitable (Maharaj et al. 2019).

`NumericalCheck` provides information summarizing the results of the NPC for various prediction levels. `PI` is the prediction level × 100 and `n` is the number of observations. `"Expected points below PI"` and `"Expected points above PI"` are respectively the expected numbers below and above the PI; `"points below PI"` and `"points above PI"` respectively represent the numbers of points below and above the PI; `"95%CIBelowFrom(%)"` and `"95%CIBelowTo(%)"` represent the 95% confidence interval of `"points below PI(%)"`; and `"95%CIAboveFrom(%)"` and `"95%CIAboveTo(%)"` represent the 95% confidence interval of `"points above PI(%)"`. If `"points below PI(%)"` is in the 95% confidence intervals of `"points below PI(%)"` and is similar to `"Expected points below PI"`, and if `"points above PI(%)"` is in the 95% confidence intervals of `"points above PI(%)"` and is similar to `"Expected points above PI"`, this is the evidence that the fitted model explains the data well.

```
NumericalCheck(origdata,simdata,pred.level=c(0,0.2,0.4,0.6,0.8,0.9),N_xbin=8)$NPC

#>   PI   n Expected points below PI points below PI points below PI(%)
#> 1  0 132                     66.0              57          43.18182
#> 2 20 132                     52.8              46          34.84848
#> 3 40 132                     39.6              37          28.03030
#> 4 60 132                     26.4              27          20.45455
#> 5 80 132                     13.2              17          12.87879
#> 6 90 132                      6.6               7           5.30303
#>   95%CIBelowFrom(%) 95%CIBelowTo(%) Expected points above PI points above PI
#> 1        35.6060606        56.81818                     66.0              63
#> 2        26.8750000        48.88258                     52.8              49
#> 3        20.4545455        36.76136                     39.6              35
#> 4        11.3636364        26.15530                     26.4              28
#> 5         3.7878788        16.66667                     13.2              12
#> 6         0.3598485        10.64394                      6.6               5
#>   points above PI(%) 95%CIAboveFrom(%) 95%CIAboveTo(%)
#> 1          47.727273         34.090909       55.303030
#> 2          37.121212         25.359848       46.609848
#> 3          26.515152         16.666667       36.003788
#> 4          21.212121         10.965909       25.000000
#> 5           9.090909          2.632576       15.549242
#> 6           3.787879          1.117424        8.333333
```

4.2 Coverage plot

The result of the NPC is a table with many values, which, while useful, can be difficult to parse visually. The coverage plot (M. O. Karlsson and Holford 2008) was developed to help visually check the fitted model with the NPC result. In each level of the predicted interval, the ratios between the expected number of data points (Exp) outside the prediction interval and the observed number of data points (Obs) outside the prediction interval are calculated. These ratios are calculated separately for the upper and lower bound of the prediction interval. For example, when the prediction level is 90, a 90% prediction interval is used, and 10% of the observations are expected to locate outside this prediction interval. To be more precise, 5% of observations are expected to be above the upper limit, and 5% of observations are expected to be below the lower limit.

The coverage plot with the NPC result can diagnose a model using multiple prediction intervals. The X-axis represents the prediction level, and the Y-axis represents Obs/Exp. The closer Obs/Exp is to 1, the more appropriate the model is. Furthermore, the confidence intervals of Obs/Exp values are obtained using simulated data and then expressed together in the plot for a more objective comparison. This plot can provide more information than the VPC plot, which interprets only a couple of quantiles - usually the 10%, 50%, and 90% percentiles.

The **xpose4** package (R. J. Keizer, Karlsson, and Hooker 2013) provides a coverage plot. However, to draw the coverage plot using **xpose4**, PsN (Lindbom, Ribbing, and Jonsson 2004) software is needed to calculate the NPC result. Therefore, we developed `coverageplot` to draw the coverage plot using the results from `NumericalCheck`.

Figure 5(A) shows the coverage plot. The X-axis shows the level of the prediction interval. The Y-axis show the ratio between the observed number of data and the expected number of data of the lower and upper parts in each level of the prediction interval. The white line is the reference line, and the gray area represents the confidence areas of the ratios in each prediction level. If the solid lines are near the white line or in the gray area, we can conclude that the suggested model is suitable.

4.3 Coverage detailed plot

Unlike the VPC plot, which represents the data space, the information in the observed data space does not come together in the coverage plot, which makes it difficult to determine whether the model is overestimated, underestimated, or adequate in the specific region of the data space. To overcome this limitation of the coverage plot, we propose a new method called the coverage detailed plot. The percentages of observations above the prediction interval are calculated in each bin of the independent variable. Additionally, the percentages of observations below the prediction interval are calculated. The white dots in the plot represent the expected percentages. If the percentages of upper and lower observations are near the white dots, we can conclude that the suggested model is suitable for the specific prediction interval.

Figure 5(B) is the result of `coverageDetailplot` when the prediction level is 80%. The white dots represent the expected percentages of the lower and upper the prediction intervals, 10%, and 90%, respectively. The upper and lower percentages of observation in each time bin are shown in darker gray. The left bin(before 0.045 hours) shows all light gray in the coverage detailed plot, and it is quite different patterns from the expected one. However, it is mainly due to the characteristics of this example data. All observations in this bin are 0. It makes the lower and upper bound of the prediction interval all 0, and the lower and upper percentages become 0.

4.4 Quantified VPC

Post et al. (2008) proposed the quantified VPC (QVPC). It expanded the existing VPC, including information about missing data. The QVPC plot visually represents actual and unavailable observations around the predicted medians through the form of percent, regardless of the observed data's density or shape. Here, "unavailable observations" refer to all kinds of missing values and unobserved data that occur for various reasons, including deviations and unexpected omissions.

If the model is appropriate, observations at each time bin are allocated randomly around the predicted median of the model. In the QVPC plot, white dots represent the model's predicted median in each bin. If the borderlines above and below are close to the white dots, we can conclude that the fitted model describes the observed data well.

Figure 6 shows the result of `quantVPC`. The darker gray areas represent the percentages below the median. The lighter gray areas represent the percentage above. The brightest gray areas represent the percent unavailable in each time bin. The white dots represent the ideal location where the above and the below percentages meet. In this example, there is no missing value.

5 The nlmeVPC package: structure and functionality

Table **??** shows the list of functions and Table **??** shows the list of arguments used in the functions of **nlmeVPC**. The following codes are for Figure 1 to Figure 6.

```
library(nlmeVPC)
data(origdata)
data(simdata)
```

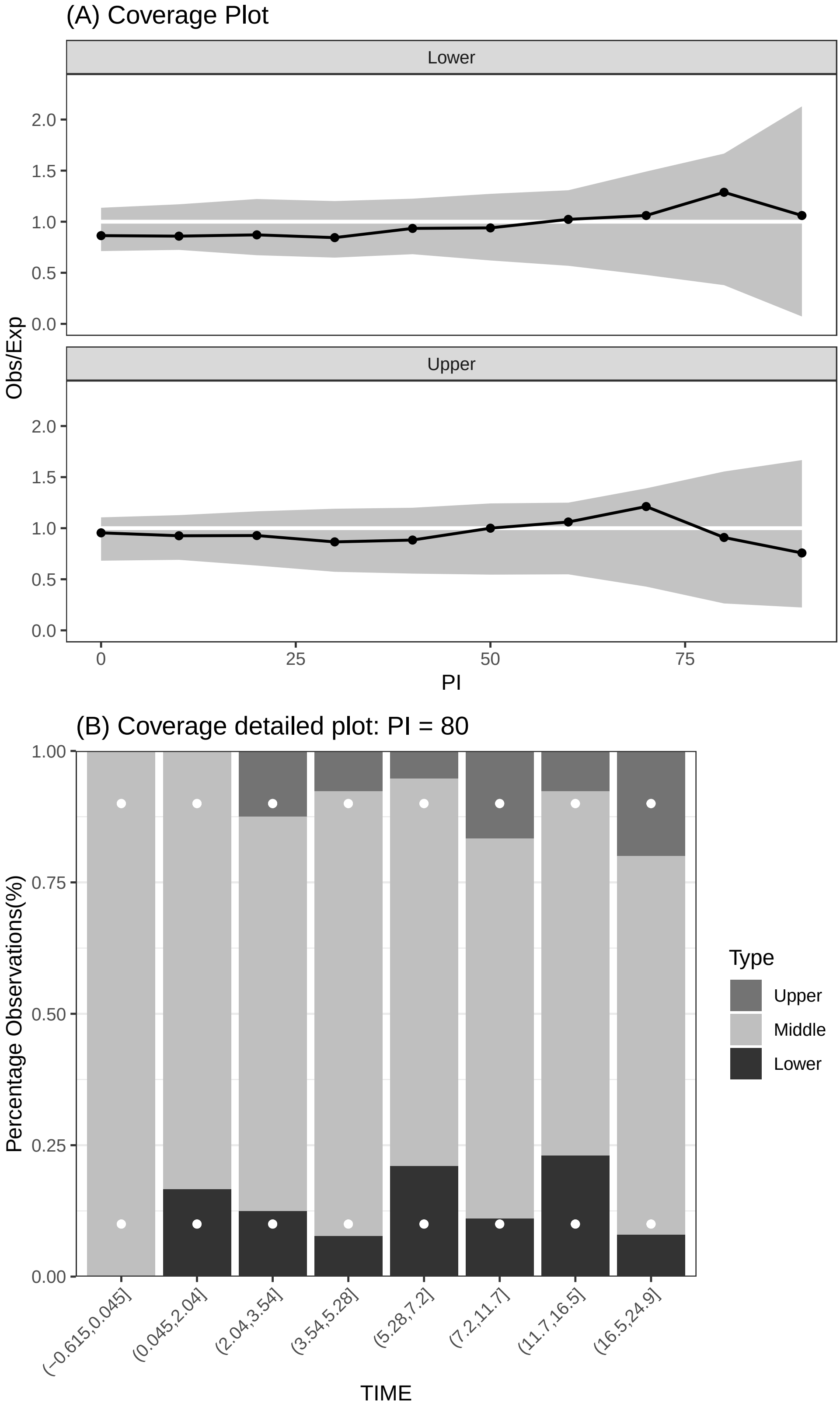


Figure 5: The coverage plot and the coverage detailed plot for the 80% prediction interval. In the coverage plot, the X-axis is the level of the prediction interval. The Y-axis is the ratio between the number of observed data and the number of expected data of the lower and upper parts in each level of the prediction interval. The white line is the reference line, and the gray area represents the confidence area of the ratios. If the solid lines are near the white line, we can conclude that the suggested model is suitable. In the coverage detailed plot, the white dots represent the expected percentages of lower and upper prediction intervals of, 10%, and 90%, respectively. The upper and lower percentages of observation in each time bin are darker gray.

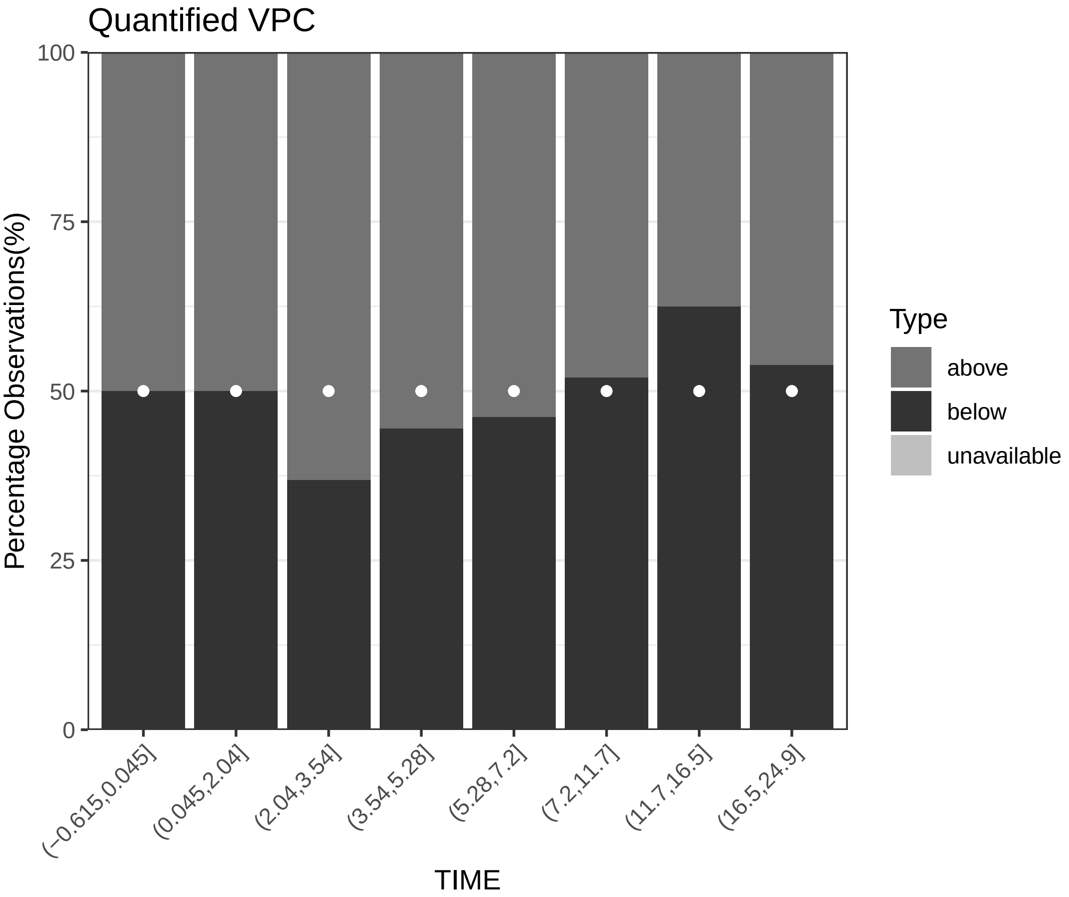


Figure 6: The quantified VPC plot. The darker gray areas represent the percentages below the median, the lighter gray areas represent the percentage above, and the brightest gray areas represent the percent unavailable in each time bin. The white dots represent the ideal location where the above and the below percentages meet.

```
optK(origdata$TIME)$K

#> [1] 8

# Figure 1

VPCgraph(origdata,simdata,N_xbin=8,type="scatter")+
  labs(title="(A) Visual Predictive Check : scatter",caption="")
VPCgraph(origdata,simdata,N_xbin=8,type="percentile")+
  labs(title="(B) Visual Predictive Check : percentile",caption="")
VPCgraph(origdata,simdata,N_xbin=8,type="CI")+
  labs(title="(C) Visual Predictive Check : CI",caption="")

# Figure 2

aqrVPC(origdata,simdata) +labs(caption="")

# Figure 3

bootVPC(origdata,simdata,N_xbin=8)+labs(caption="")

# Figure 4

asVPC(origdata,simdata,type="CI",N_xbin=8,N_hist=3,weight_method="bin")+labs(caption="")
asVPC(origdata,simdata,type="CI",N_xbin=8,N_hist=3,weight_method="distance")+labs(caption="")

# Numerical Predictive Check

NumericalCheck(origdata,simdata,N_xbin=8,pred.level=c(0,0.2,0.4,0.6,0.8,0.9))$NPC

# Figure 5


coverageplot(origdata,simdata,N_xbin=8) +ggtitle("(A) Coverage Plot")
coverageDetailplot(origdata,simdata,N_xbin=8,predL=0.8) +
ggtitle("(B) Coverage Detailed plot: PI = 80")

# Figure 6

quantVPC(origdata,simdata)
```

We use an example to show how to use functions in **nlmeVPC** and how they work.

5.1 Example

The `origdata` in **nlmeVPC** is from an experiment on the pharmacokinetics of theophylline. Twelve patients were given oral doses of theophylline, and blood concentrations were measured at 11 time points over the next 25 hours. Each patient had different time points. We consider the following first-order absorption one-compartment model:

$$y_{ij} = \frac{Amt_i * Ke_i * Ka_i}{Cl_i}\Big(\exp(-Ke_i * TIME_{ij}) - \exp(-Ka_i * TIME_{ij})\Big) + \varepsilon_{ij}.$$

In this model, y_{ij} is the theophylline concentration at $TIME_{ij}$ after an initial dose of Amt_i. The pharmacokinetic parameters are the absorption rate constant Ka, the elimination rate constant Ke, and the clearance Cl. In this example, two different models are fitted and diagnosed using functions in the **nlmeVPC** package. In Model 1, Ka and Cl are considered as random effects. In Model 2, Ke is considered as random effect, and Ka and Cl are considered only as a fixed effect. The **nlme** (J. Pinheiro et al. 2022) and **nlraa** (Miguez 2022) packages are used to fit the nonlinear mixed models and to generate the simulated data from the fitted model.

```
library(nlme)
library(nlraa)
library(nlmeVPC)
```

```
data(origdata)
origdataT <- groupedData(DV~TIME|ID,origdata)

# Model 1 (True)
T.nlme <- nlme(DV ~ exp(lKe+lKa-lCl)*AMT*
               (exp(-exp(lKe)*TIME) - exp(-exp(lKa)*TIME))/
               (exp(lKa)-exp(lKe)), data=origdataT,
             fixed=lKa+lKe+lCl~1,
             random=lKa+lCl~1,
             start=c(lKe=-2,lKa=1.5,lCl=-3))
set.seed(123456)
sim.T <- simulate_nlme(object=T.nlme,nsim=100,psim=3,level=1,value="data.frame",data = NULL)
simdata.T <- matrix(sim.T$sim.y,ncol=100)

# Model 2 (Wrong)
F.nlme <- nlme(DV ~ exp(lKe+lKa-lCl)*AMT*
                (exp(-exp(lKe)*TIME) - exp(-exp(lKa)*TIME))/
                (exp(lKa)-exp(lKe)), data=origdataT,
              fixed=lKa+lKe+lCl~1,
              random=lKe~1,
              start=c(lKe=-2,lKa=1.5,lCl=-3))

sim.F <- simulate_nlme(object=F.nlme,nsim=100,psim=3,level=1,value="data.frame",data = NULL)
simdata.F <- matrix(sim.F$sim.y,ncol=100)

# Figure 7

VPCgraph(origdata,simdata.T,type="CI",N_xbin=8)+labs(title="Model 1",caption="")
VPCgraph(origdata,simdata.F,type="CI",N_xbin=8)+labs(title="Model 2",caption="")

# Figure 8

aqrVPC(origdata,simdata.T)+labs(title="Model 1",caption="")
aqrVPC(origdata,simdata.F)+labs(title="Model 2",caption="")

# Figure 9

asVPC(origdata,simdata.T,type="CI",weight_method="distance",N_xbin=8)+labs(title="Model 1",caption="")
asVPC(origdata,simdata.F,type="CI",weight_method="distance",N_xbin=8)+labs(title="Model 2",caption="")

# Figure 10

bootVPC(origdata,simdata.T,N_xbin=8)+labs(title="Model 1",caption="")
bootVPC(origdata,simdata.F,N_xbin=8)+labs(title="Model 2",caption="")

# Figure 11

coverageplot(origdata,simdata.T,conf.level=0.9,N_xbin=8)+labs(title="Model 1")
coverageplot(origdata,simdata.F,conf.level=0.9,N_xbin=8)+labs(title="Model 2")

# Figure 12

coverageDetailplot(origdata,simdata.T,predL=0.5,N_xbin=8)+labs(title="Model 1")
coverageDetailplot(origdata,simdata.F,predL=0.5,N_xbin=8)+labs(title="Model 2")

# Figure 13

coverageDetailplot(origdata,simdata.T,predL=0.8,N_xbin=8)+labs(title="Model 1")
coverageDetailplot(origdata,simdata.F,predL=0.8,N_xbin=8)+labs(title="Model 2")

# Figure 14

quantVPC(origdata,simdata.T,N_xbin=8)+labs(title="Model 1")
quantVPC(origdata,simdata.F,N_xbin=8)+labs(title="Model 2")
```

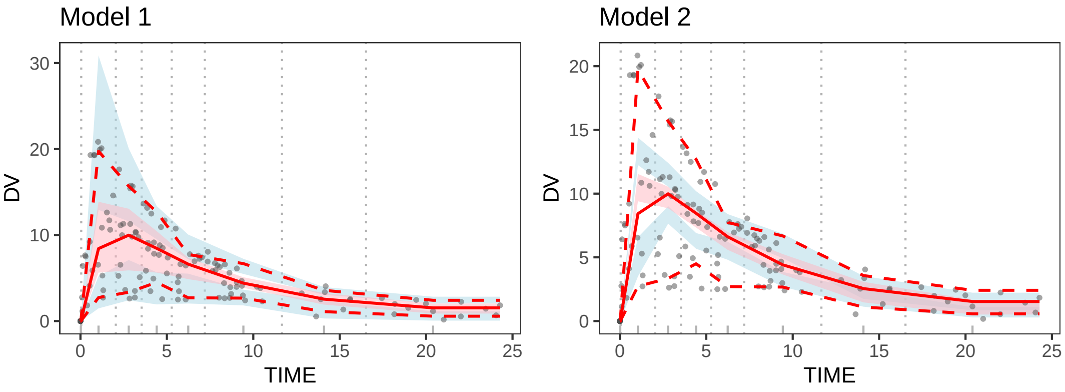


Figure 7: The VPC plots of Model 1 and Model 2.

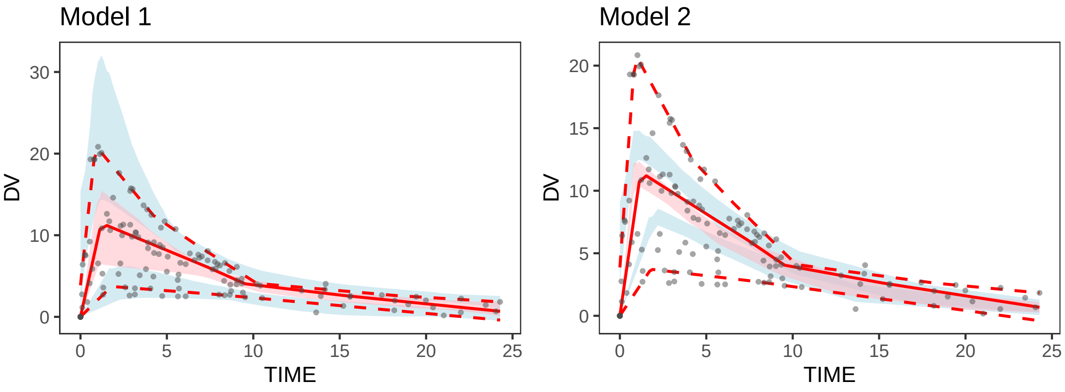


Figure 8: The additive quantile regression VPC plots for Model 1 and Model 2.

Figure 7 shows the VPCgraph for Model 1 and Model 2. The solid line represents the 50th percentile of the observed data, and the dashed lines represent the 10th and 90th percentiles of the observed data. In Model 1, all quantile lines (the solid and dashed lines) are in the confidence area. However, the 10th and 90th percentiles in Model 2 are mostly outside the confidence area, especially at 0 to 5 hours.

Figure 8 shows the results of aqrVPC, and Figure 9 shows the results of asVPC for Model 1 and Model 2. The additive quantile regression VPCs and the average shifted VPCs show similar patterns to the original VPCs. Model 1 shows all quantile lines in the confidence area, and the 10th and 90th percentiles in Model 2 are mostly outside the confidence area.

Figure 10 shows the results of bootVPC for Model 1 and Model 2. The solid and dashed blue lines show the 10th, 50th, and 90th percentiles of the simulated data. The solid red line represents the 50th percentile line of the observed data, and the pink areas represent the 95% confidence areas of the 50th percentile line, calculated from the bootstrap samples of the observed data. The solid blue line is in the pink area, and the two solid red and blue lines are almost identical in both models. The dashed blue lines in Model 1 cover most of the observed data. However, the dashed blue lines in Model 2 do not

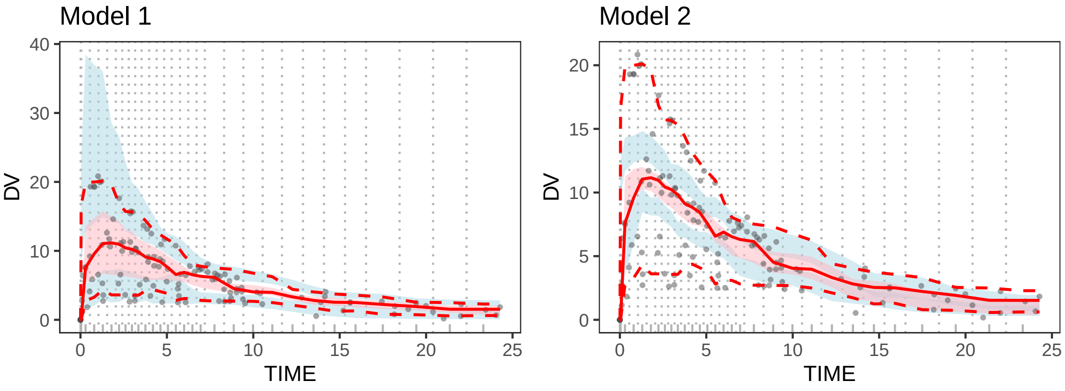


Figure 9: The average shifted VPC plots for Model 1 and Model 2.

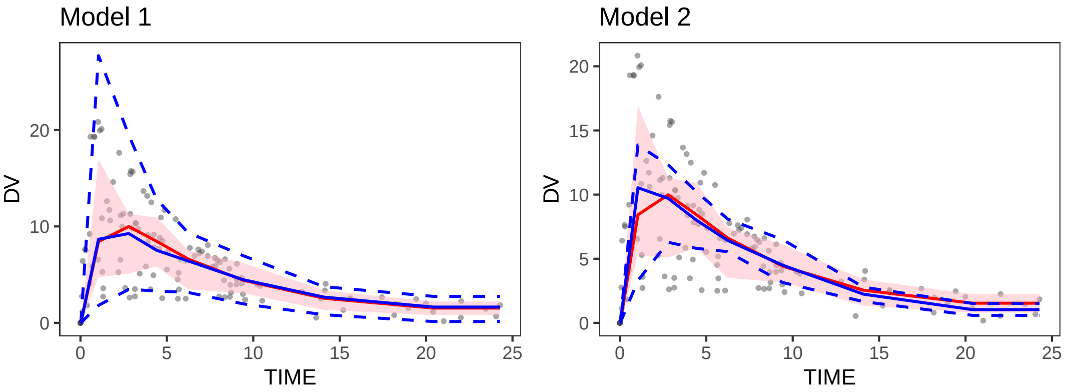


Figure 10: The bootstrap VPC plots for Model 1 and Model 2.

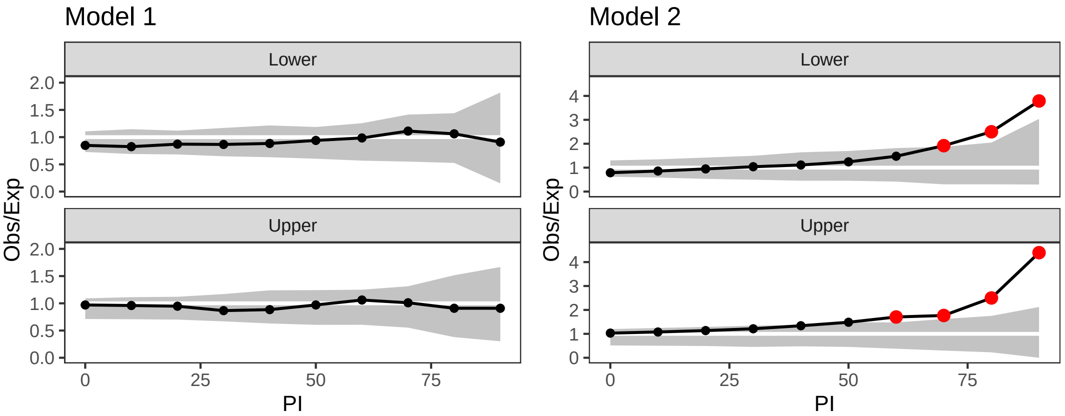


Figure 11: The coverage plots for Model 1 and Model 2.

cover the observed data. Many observed data points lie outside these dashed blue lines in Model 2, especially in 0 to 10 hours.

Figure 11 shows the coverageplot results for Model 1 and Model 2. The lines are in the gray area and close to the white line in Model 1. However, the lines in Model 2 are not in the gray area, especially when the PI value is large. Figures 12 and 13 show the results of coverageDetailplot for Model 1 and Model 2 when PIs are 50% and 80%. The upper and lower percentages in both figures are close to the white points in Model 1. On the other hand, the upper percentages of the most time bins are far from the white points in Model 2, especially the time bin (3.54,5.28] when PI = 50%. When PI = 80%, most upper and lower percentages are far from the white points.

Figure 14 shows the results of quantVPC for Model 1 and Model 2. In Model 2, the right tail area (after 11 hours) looks quite different from the expected pattern. The above percentages are much larger than the below percentages.

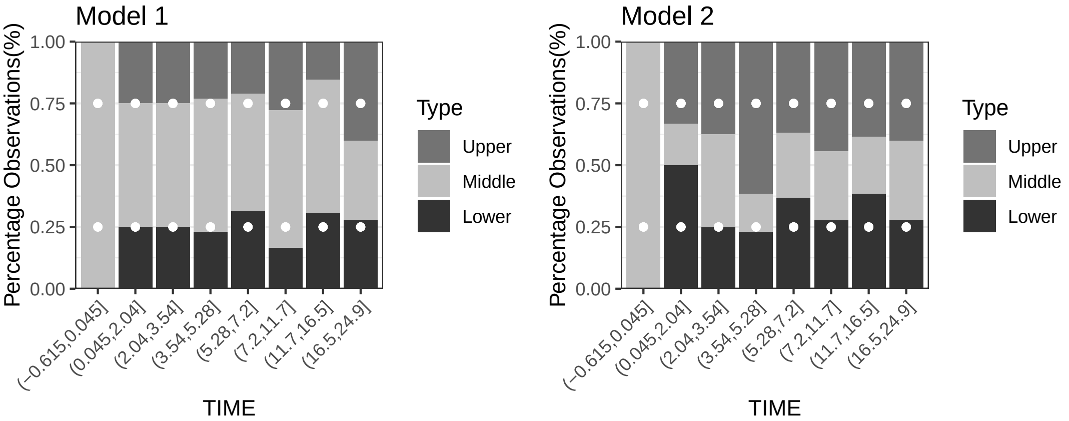


Figure 12: The coverage detailed plots for Model 1 and Model 2 when PI=50%.

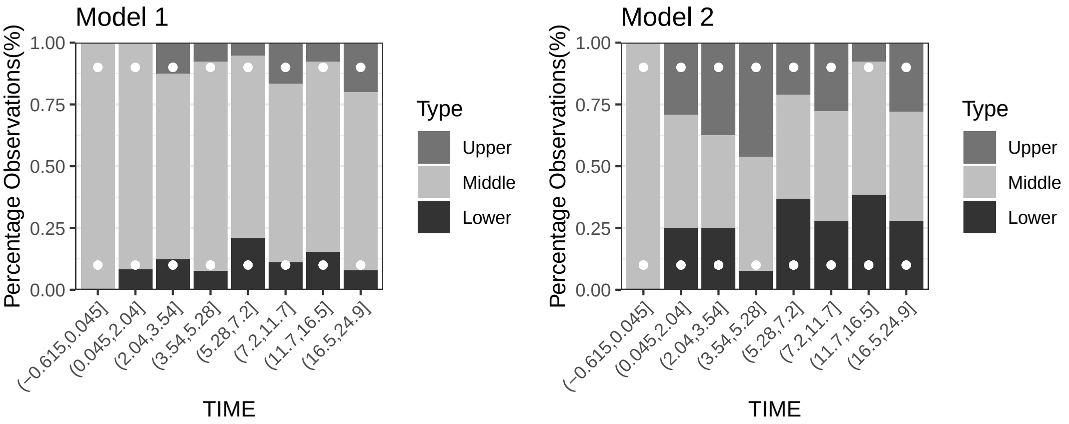


Figure 13: The coverage detailed plots for Model 1 and Model 2 when PI=80%.

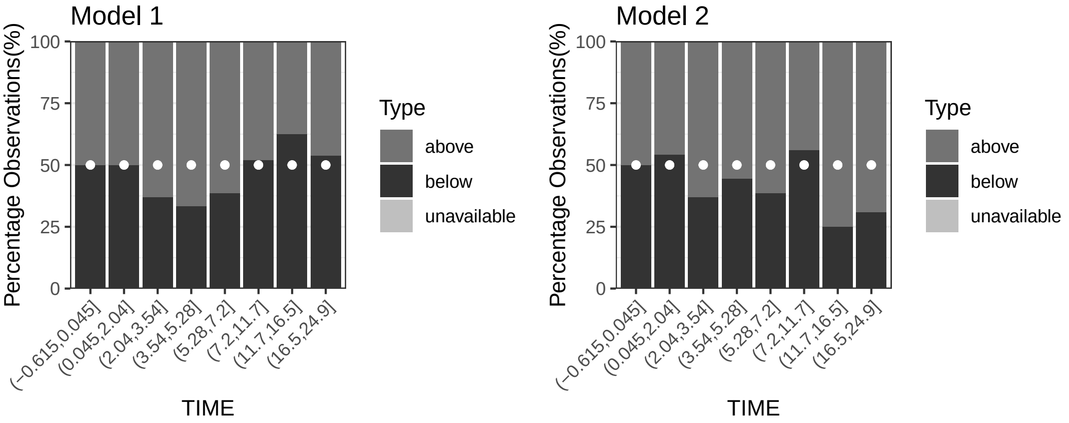


Figure 14: The quantified VPC plots for Model 1 and Model 2.

The results from Figure 7 through Figure 14 show that Model 1 explains the `origdata` quite well. However, Model 2 shows different patterns than Model 1 in most figures. We can conclude that Model 1 is better than Model 2, and treating *Ka* and *Cl* as random effects is better.

6 Summary

This paper introduces the **nlmeVPC** package. The VPC and its extensions are useful for validating the nonlinear mixed effect model. The **nlmeVPC** package provides various visual diagnostic tools for the nonlinear mixed effect model in two different approaches: validation in data space and model space. Both approaches are valuable. Validation in data space can compare the fitted model with the original data, and validation in model space provides detailed comparisons in various ways. In the **nlmeVPC** package, we also provide new approaches - `asVPC` and `coverageDetailplot`. Here, `asVPC` provides a more precise VPC plot, and `coverageDetailplot` provides the detailed feature of the fitted model that is not captured in the coverage plot. Even though the coverage plot does not show any problem with the fitted model, the coverage detailed plot can reveal the problem in a specific region (Figures 10, 11, and 12).

7 Acknowledgements

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (2018R1A2B6001251). This paper was prepared by extracting part of Eun-Hwa Kang and Myungji Ko's master thesis.

References

Bauer, Robert J. 2011. "NONMEM Users Guide Introduction to NONMEM 7.2.0." *ICON Development Solutions Ellicott City, MD*. https://www.iconplc.com/innovation/nonmem/.

Davidian, Marie. 2017. *Nonlinear Models for Repeated Measurement Data*. Routledge.

Fidler, Matthew, Justin J Wilkins, Richard Hooijmaijers, Teun M Post, Rik Schoemaker, Mirjam N Trame, Yuan Xiong, and Wenping Wang. 2019. "Nonlinear Mixed-Effects Model Development and Simulation Using Nlmixr and Related r Open-Source Packages." *CPT: Pharmacometrics & Systems Pharmacology* 8 (9): 621–33. https://doi.org/10.1002/psp4.12445.

Heus, Astrid, David W Uster, Veerle Grootaert, Nele Vermeulen, Annemie Somers, Diana Huis In't Veld, Sebastian G Wicha, and Pieter A De Cock. 2022. "Model-Informed Precision Dosing of Vancomycin via Continuous Infusion: A Clinical Fit-for-Purpose Evaluation of Published PK Models." *International Journal of Antimicrobial Agents* 59 (5): 106579. https://doi.org/10.1016/j.ijantimicag.2022.106579.

Jamsen, Kris M, Kashyap Patel, Keith Nieforth, and Carl MJ Kirkpatrick. 2018. "A Regression Approach to Visual Predictive Checks for Population Pharmacometric Models." *CPT: Pharmacometrics & Systems Pharmacology* 7 (10): 678–86. https://doi.org/10.1002/psp4.12319.

Karlsson, MATS O, and NICK Holford. 2008. "A Tutorial on Visual Predictive Checks." In *17th Meeting of the Population Approach Group in Europe, Marseille, France, Page Abstr*. Vol. 1434. https://www.page-meeting.org/?abstract=1434.

Karlsson, MO, and RM Savic. 2007. "Diagnosing Model Diagnostics." *Clinical Pharmacology & Therapeutics* 82 (1): 17–20. https://doi.org/10.1038/sj.clpt.6100241.

Keizer, R. 2021. "Vpc: Create Visual Predictive Checks. R Package Version 1.2.2." https://cran.r-project.org/web/packages/vpc/index.html.

Keizer, Ron J, MO Karlsson, and A Hooker. 2013. "Modeling and Simulation Workbench for NONMEM: Tutorial on Pirana, PsN, and Xpose." *CPT: Pharmacometrics & Systems Pharmacology* 2 (6): 1–9. https://doi.org/10.1038/psp.2013.24.

Koenker, Roger. 2023. "Quantreg: Quantile Regression. R Package Version 5.95." https://cran.r-project.org/web/packages/quantreg/index.html.

Lavielle, Marc, and Kevin Bleakley. 2011. "Automatic Data Binning for Improved Visual Diagnosis of Pharmacometric Models." *Journal of Pharmacokinetics and Pharmacodynamics* 38 (6): 861–71. https://doi.org/10.1007/s10928-011-9223-3.

Lindbom, Lars, Jakob Ribbing, and E Niclas Jonsson. 2004. "Perl-Speaks-NONMEM (PsN)—a Perl Module for NONMEM Related Programming." *Computer Methods and Programs in Biomedicine* 75 (2): 85–94. https://doi.org/10.1016/j.cmpb.2003.11.003.

Maharaj, Anil R, Huali Wu, Christoph P Hornik, and Michael Cohen-Wolkowiez. 2019. "Pitfalls of Using Numerical Predictive Checks for Population Physiologically-Based Pharmacokinetic Model Evaluation." *Journal of Pharmacokinetics and Pharmacodynamics* 46 (3): 263–72. https://doi.org/10.1007/s10928-019-09636-5.

Miguez, Fernando. 2022. "Nlraa: Nonlinear Regression for Agricultural Applications. R Package Version 1.5." https://cran.r-project.org/web/packages/nlraa/index.html.

Mudde, Saskia E, Rami Ayoun Alsoud, Aart van der Meijden, Anna M Upton, Manisha U Lotlikar, Ulrika SH Simonsson, Hannelore I Bax, and Jurriaan EM de Steenwinkel. 2022. "Predictive Modeling to Study the Treatment-Shortening Potential of Novel Tuberculosis Drug Regimens, Toward Bundling of Preclinical Data." *The Journal of Infectious Diseases* 225 (11): 1876–85. https://doi.org/10.1093/infdis/jiab101.

Nguyen, THT, M-S Mouksassi, Nicholas Holford, N Al-Huniti, I Freedman, Andrew C Hooker, J John, et al. 2017. "Model Evaluation of Continuous Data Pharmacometric Models: Metrics and Graphics." *CPT: Pharmacometrics & Systems Pharmacology* 6 (2): 87–109. https://doi.org/10.1002/psp4.12161.

Otto, ME, KR Bergmann, G Jacobs, and Michiel J van Esdonk. 2021. "Predictive Performance of Parent-Metabolite Population Pharmacokinetic Models of (s)-Ketamine in Healthy Volunteers." *European Journal of Clinical Pharmacology* 77 (8): 1181–92. https://doi.org/10.1007/s00228-021-03104-1.

Pinheiro, José C., and Douglas M. Bates. 2000. *Mixed-Effects Models in s and s-PLUS*. Springer New York. https://doi.org/10.1007/978-1-4419-0318-1.

Pinheiro, José, Douglas Bates, Saikat DebRoy, Deepayan Sarkar, Siem Heisterkamp, Bert Van Willigen, and R Maintainer. 2022. "Nlme: Linear and Nonlinear Mixed Effects Models. R Package Version 3.1-159." https://cran.r-project.org/web/packages/nlme/index.html.

Post, Teun M, Jan I Freijer, Bart A Ploeger, and Meindert Danhof. 2008. "Extensions to the Visual Predictive Check to Facilitate Model Performance Evaluation." *Journal of Pharmacokinetics and Pharmacodynamics* 35 (2): 185–202. https://doi.org/10.1007/s10928-007-9081-1.

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

Scott, David W. 1985. "Average Shifted Histograms: Effective Nonparametric Density Estimations in Several Dimensions." *Annals of Statistics* 13 (3): 1024–40. https://doi.org/10.1214/aos/1176349654.

Wang, Diane D, and Shuzhong Zhang. 2012. "Standardized Visual Predictive Check Versus Visual Predictive Check for Model Evaluation." *The Journal of Clinical Pharmacology* 52 (1): 39–54. https://doi.org/10.1177/0091270010390040.

Wickham, H. 2016. *Ggplot2 : Elegance Graphics for Data Analysis*. Springer.

Wickham, Hadley, Dianne Cook, and Heike Hofmann. 2015. "Visualizing Statistical Models: Removing the Blindfold." *Statistical Analysis and Data Mining: The ASA Data Science Journal* 8 (4): 203–25. https://doi.org/10.1002/sam.11271.

Eun-Hwa Kang
Ewha Womans University
Department of Statistics, Ewha Womans University, Seoul, Korea

Myungji Ko
Ewha Womans University
Department of Statistics, Ewha Womans University, Seoul, Korea

Eun-Kyung Lee
Ewha Womans University
Department of Statistics, Ewha Womans University, Seoul, Korea
ORCiD: 0000-0003-0817-5000
lee.eunk@ewha.ac.kr

Likelihood Ratio Test-Based Drug Safety Assessment using R Package pvLRT

by Saptarshi Chakraborty, Marianthi Markatou, and Robert Ball

Abstract Medical product safety continues to be a key concern of the twenty-first century. Several spontaneous adverse events reporting databases established across the world continuously collect and archive adverse events data on various medical products. Determining signals of disproportional reporting (SDR) of product/adverse event pairs from these large-scale databases require the use of principled statistical techniques. Likelihood ratio test (LRT)-based approaches are particularly noteworthy in this context as they permit objective SDR detection without requiring ad hoc thresholds. However, their implementation is non-trivial due to analytical complexities, which necessitate the use of computation-heavy methods. Here we introduce R package pvLRT which implements a suite of LRT approaches, along with various post-processing and graphical summary functions, to facilitate simplified use of the methodologies. Detailed examples are provided to illustrate the package through analyses of three real product safety datasets obtained from publicly available FDA FAERS and VAERS databases.

1 Introduction

Adverse events of medical products are a major concern worldwide, with serious global health implications particularly in the post-pandemic world. While clinical trials conducted pre-licensure remain the primary source of safety information about medical products, and other data sources such as claims and electronic health records are increasingly being used post-licensure, they exhibit several limitations. First, detecting extremely rare adverse events in clinical trials is difficult due to their strict inclusion/exclusion criteria and their relatively small counts compared to the population which will eventually receive the product. Second, randomized clinical trials for detecting multiple adverse events may be logistically and ethically infeasible. Despite having known issues such as self-selection, confounding, and missing data, etc., large-scale observational studies have been used in surveillance and post-licensure epidemiological studies to supplement traditional clinical trial-based approaches (Markatou and Ball 2014). Thanks to the proliferation of electronic health record systems worldwide, a vast trove of such data are now available and can collectively provide critical first alerts for emerging medical product safety concerns. Curated large-scale databases such as US Food and Drug Administration (FDA)'s Adverse Event Reporting System (FAERS) and Vaccine Adverse Event Reporting System (VAERS), the European Medicines Agency's Eudravigilance, and the World Health Organization's VigiBase provide collective lists of drugs/vaccines and adverse events from large number of reports, and thus are invaluable resources for product safety assessment.

Mining these databases to assess medical product safety is however challenging due to limitations of the data (e.g., biases in reporting, presence of duplicate reports, and missing/incomplete information) and the lack of a universally accepted statistical methodology. Several approaches to detecting signals of disproportionate reporting (SDR) as an indicator of a possible safety concern from adverse event data exist; these constitute a part of the pharmacovigilance process for assessing whether a medical product is the cause of an adverse event. Of the existing approaches, Proportional Relative Reporting Proportion (PRR), Reporting Odds Ratio (ROR), Multi-item Gamma Poisson Shrinker (MGPS), Bayesian Confidence Propagation Neural Network (BCPNN) and their false discovery rate (FDR)-adjusted variants are notable. Many of these approaches require one or both types of thresholding: (a) on the number of reports needed for the method to operate, and (b) on the underlying statistic to identify important/severe adverse events. However, determining appropriate thresholds remains a key challenge for these approaches, and in practice ad hoc thresholds are often used. This is problematic because both too low and too high thresholds lead to drastically different detection rates, and no separate measure is usually available to assess correctness of the detected signals. A principled approach permitting data-driven threshold determination with optimal guarantees on the detected signals is therefore essential to ensure statistical validity of results.

The LRT-based approaches to signal detection in the medical product safety datasets (Ding, Markatou, and Ball 2020; Huang, Zalkikar, and Tiwari 2011; Huang et al. 2017; Zhao, Yi, and Tiwari 2018; Chakraborty et al. 2022) provide a highly rigorous suite of statistical methods to address these problems. To provide a high-level summary, these methods assume a Poisson/zero-inflated Poisson model to parametrize associations between drugs and their adverse events. Then conditional on the total reported number of cases for each adverse event and each medical product, these methods quantify signal strength in the observed number of reports per adverse event/medical product pair

Table 1: Existing R packages on CRAN with functionalities for pharmacovigilance.

Package	Method	Notes
PhViD: Pharmacovigilance Signal Detection	PRR, ROR, BCPNN, GPS	Aimed towards drug safety.
openEBGM: EBGM Disproportionality Scores for Adverse Event Data Mining	MGPS	
sglr: An R package for power and boundary calculations in pre-licensure vaccine trials using a sequential generalized likelihood ratio test	Sequential Generalized Likelihood Ratio decision boundaries	Aimed towards sequential testing-based vaccine safety as used by the FDA
Sequential: Exact sequential analysis for Binomial and Poisson data	Max SPRT statistic	
AEenrich: Vaccine adverse event enrichment tests	a) Modified Fisher's exact test (for pre-selected significant AEs); b) Modified Kolmogorov Smirnov statistic	
mds: Medical Devices Surveillance	Data preprocessing	Provides functions for handling messy/unstructured medical devices data.
pvLRT: A suite of likelihood ratio test based methods to use in pharmacovigilance	(Pseudo) LRT approaches based on log-linear models	Our package.

through a (likelihood ratio) test statistic. Finally, significance of the pair is determined by comparing the observed test statistic to its null (independence) sampling distribution. There are several key advantages of these formal hypothesis test-based approaches which make them ideal for practical use. First, they aid coherent frequetist sampling distribution-based quantification of signal strengths, ensuring strong statistical and probabilistic rigor of the results. Second, they do not require the use of ad hoc thresholds for the observed report counts; instead, they quantify significance through probabilistic measures of uncertainty, e.g., p-values. This ensures that the identified signals are valid up to a pre-specified level of tolerance (usually 0.05). Third, because they are based on the highly formal likelihood ratio test theory for null hypothesis significance testing, they can rigorously control the type I error and the false discovery rate, while ensuring high power and sensitivity for signal detection. Moreover, by appropriately defining a maximum likelihood ratio test statistic, the method achieves automated FDR adjustment without requiring any separate p-value adjustment step.

There currently exist a few R packages in CRAN with functionalities relevant for medical product safety. This includes the packages **PhViD** (Ahmed and Poncet 2016), **openEBGM** (Canida and Ihrie 2017), **AEenrich** (Li et al. 2021), **Sequential** (Silva and Kulldorff 2021), **sglr** (Narasimhan and Shih 2012), and **mds** (Chung 2020). Among these **PhViD** and **openEBGM** provide functionalities for spontaneous adverse event data-driven pharmacovigilance: **PhViD** implements methods such as PRR, ROR, and BCPNN, and **openEBGM** implements the method of MGPS. By contrast, the other packages provide sequential testing-based approaches to vaccine safety. Table 1 lists these packages and their functionalities together with some high-level notes on their targeted uses. We note however that none of these packages implement LRT-based approaches to drug safety based on spontaneous adverse event data as considered in **pvLRT**. Indeed the development of **pvLRT** was motivated by the lack of easily accessible and comprehensive open source computational solutions to the LRT-based pharmacovigilance approaches. An important common ingredient of these LRT-based methods is a computation-intensive Monte Carlo simulation step required to facilitate (*exact/non-asymptotic*) inference from the analytically intractable *null* sampling distributions of the relevant test statistics. However, this requirement precludes immediate applications of the methods by pharmacovigilance practitioners in the absence of easily accessible software implementations. This is particularly relevant for zero-inflation based models where, in addition to the computation-heavy Monte Carlo step additional numerical optimization steps are necessary for the estimation of the zero-inflation parameter. We have developed **pvLRT** (Chakraborty and Markatou 2022) to cater to these needs while ensuring proper statistical rigor in the implementations.

The following are the major contributions of the **pvLRT** package. First, the package serves as a comprehensive software implementation of several LRT-based methods proposed over the past decade, including some recently developed methods. Both Poisson and Zero-inflated Poisson (ZIP) based models are implemented, and tests of both single and multiple simultaneous drugs (or medical products) are provided. Formal model comparison methods, such as Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC), can be used on the model fits to aid data-driven selection of Poisson vs. ZIP models on individual datasets. Second, three visual summary plots for the test results , namely bubble plot, barplot, and heatmap, are provided for immediate visualization. These plots provide information on sample size and test statistic and p-values for all AE/drug pairs tested, and hence permit quick exploration of signals in a (possibly large) dataset. These visualizations use **ggplot2** (Wickham 2016) as the plotting engine, and hence its attributes (color, text size, etc.) can be easily modified post hoc by changing some of the **ggplot2** graphical sttributes. In addition, leveraging the modern R ecosystem, the resultant plots can be easily made interactive via additional functions from other packages, such as `ggplotly()` from the package **plotly** (Sievert 2020). Third, aside from implementations of likelihood ratio tests, simple functions for random contingency table data generation from the Poisson and zero-inflated Poisson models are provided. This is particularly helpful in simulation experiments where different statistical methods for pharmacovigilance are compared. Finally, we have included processed contingency tables of AE/drug reports for two specific drug groups, viz. statins and Gadolinium-based Contrast Agents (GBCA; Zhao, Yi, and Tiwari (2018)), raw AE/drug incidence data, and a vaccine/AE dataset on rotavirus vaccines as R datasets. The drug data were obtained from the publicly available FDA FAERS database for the quarters 2014 Q1 – 2020 Q3 (for the processed contingency tables) and 2022 Q3 (raw incidence data), and the vaccine data were obtained from the FDA VAERS database for the year 1999; the package provides a convenient approach to accessing these processed data, particularly beneficial in methodological studies.

We note here that **pvLRT** is primarily an analysis package providing functions implementing statistical methodologies and subsequent post-processing and visualizations of results. The input data for the main analysis functions in **pvLRT** are always assumed to be pre-processed contingency tables (matrix-like objects) enumerating adverse event report counts with AEs along the rows and Drugs (or other medical products) along the columns. The package provides a convenience function to convert raw AE/drug incidence data (e.g., those downloaded from the FDA FAERS database) into analysis-ready processed contingency tables; however, no other functions for input raw data exploration and pre-processing, e.g. for filtering specific AEs or Drugs of interest or grouping specific AEs or drugs together are provided. We have deliberately made this design choice to (a) make the scope of the package well-defined, and (b) encourage the user to explore the raw adverse event data well before analyzing them. The modern R ecosystem contains several excellent general-purpose packages/package-collections, including **data.table** (Dowle and Srinivasan 2021) and **tidyverse** (Wickham et al. 2019) that provide a suite of principled and easy to use data pre-processing, munging, and visualization functions. We highly recommend leveraging these packages/functions to understand and preprocess the data well, before likelihood ratio tests are performed using **pvLRT**.

The purpose of the current article is to provide a high-level overview of the **pvLRT** package with detailed notes on its use, and guided examples with real world adverse event data exemplifying the use of LRT-based methodologies for pharmacovigilance. We note however that this article is not meant to serve as a comprehensive manual for **pvLRT**; all functions/objects provided in **pvLRT** come with detailed documentation, and this documentation serves as the definitive resource for **pvLRT**. It is also important to note that the LRT-based disproportionality analysis exemplified herein is only one part of the collective medical product/adverse event relationship assessment process. Current pharmacovigilance practice for medical product safety assessment includes review of summary statistics, disproportionality scores, cases summaries, and other sources of information about the medical products and the adverse events. The remainder of the article is organized as follows. We begin with a brief review of the LRT-based approaches to pharmacovigilance implemented in **pvLRT**. We then exemplify **pvLRT** by analyzing three sets of real pharmacovigilance datasets, two concerning drug safety and one concerning vaccine safety. We conclude the article with a brief discussion and some potential future directions.

2 A brief review of the Likelihood Ratio test-based approaches to pharmacovigilance

This section reviews the underlying theories behind the likelihood ratio test (LRT)-based methods for drug safety assessment. We begin by fixing our notation. Consider a drug safety database cataloging I AEs detected among J medical products. Hereinafter we describe the methodologies in terms of 'drugs' as the medical products of concern; however, the methodologies trivially extend to other medical products including vaccines and medical devices. Let n_{ij} denote the number of reported cases for the i-th AE in the j-th drug. For the $I \times J$ contingency table (n_{ij}), let $n_{i\bullet} = \sum_{j=1}^{J} n_{ij}$ and $n_{\bullet j} = \sum_{i=1}^{I} n_{ij}$ denote the i-th row total and j-th column total respectively, $i = 1, \ldots, I$; $j = 1, \ldots, J$; and let $n_{\bullet\bullet}$ denote the grand total. Interest lies in determining whether the observed report count n_{ij} for (i, j)-th AE/drug pair is substantially larger than what is expected had there been no association between the i-th AE and the j-th drug. Within the null hypothesis significance testing framework this is achieved by testing a hypothesis of absence of a "signal" against that of its presence at the corresponding AE/drug pair. We formally define these hypotheses and the underlying parametric models below.

Model and Parametrization for LRT-based pharmacovigilance. A Poisson log-linear model for LRT-based SDR detection has been proposed in the literature to parametrize AE/drug associations based on the observed report counts $\{n_{ij}\}$. At the outset we note that these AE/drug associations can also be studied through a series of logistic regression models each with binary occurrences of each individual AE as the response and the presence/absence of various drugs (or other medical products) as predictors. When a Poisson model is assumed for the report counts $\{n_{ij}\}$, the resulting log-linear model and the collective logistic regression models lead to identical maximum likelihood-based inferences (Agresti 2013). However, the log linear model is more flexible than the logistic models in that, by relaxing the underlying Poisson assumption for the report counts, it can handle a richer set of data. Two different parametrizations, namely the reporting proportion (Huang, Zalkikar, and Tiwari 2011) and the relative reporting proportion (Dumouchel 1999), have been considered within the log-linear modeling framework for pharmacovigilance. These parametrizations differ by how they define and handle "signals". In the reporting proportion parametrization a signal is determined at a specific AE for a given drug if its reporting proportion is substantially larger than the overall reporting proportions of all other AEs combined. In contrast, in the relative reporting proportion parametrization one focuses on the relative reporting proportion of a specific AE for a given drug and tests whether it is substantially larger than 1. The latter parametrization does not require consideration of the category of "all other AEs combined" explicitly in each comparison, and hence has a simpler interpretation. In applications the two parametrizations incur similar computational expenses, and produce virtually identical results when the true data generating process is Poisson. However, the results may vary substantially when there is a model misspecification, including the presence of zero-inflation, i.e., excess zero-report counts that are inherently different from the *sampling* zero counts governed by the Poisson law and are produced by an exogenous process. A note on different types of zero-inflation possibly occurring in adverse event datasets is provided in the next paragraph. Our recent methodological study (Chakraborty et al. 2022) has shown that the relative reporting proportion-based Poisson LRT is more robust against zero-inflated data than its reporting proportion based counterpart. Extensions of the Poisson LRTs have been proposed to explicitly handle excess zeros under a zero-inflated Poisson (ZIP) model. The relative reporting proportion parametrization, within a pseudo likelihood ratio testing framework, aids a straightforward extension to the ZIP model that adds only a small overhead (one single additional optimization of a smooth univariate function) to the overall computational burden, and produces a test statistic whose functional form is identical to the ordinary (non-zero inflated) Poisson LRT. This computational simplicity makes the approach highly scalable for large data sizes, similar to the Poisson LRT. Collectively, these simplifications aid the construction of a unified LRT framework based on the relative reporting proportion-based parametrization that can simultaneously handle a Poisson model and a ZIP model, with near identical computational burden. By contrast, the reporting proportion-based LRT under a ZIP model suffers from a substantially increased computation burden (Huang et al. 2017) which hinders its scalability. For these reasons, both in this article and in the **pvLRT** package we primarily focus on the relative reporting proportion parametrization; we do note however that the **pvLRT** package does provide an implementation of the reporting proportion parametrization under an ordinary Poisson model setup.

A brief note on excess zeros or *zero-inflation* in medical product safety data. We now briefly note how zero-inflation occurs in pharmacovigilance. In adverse event reporting datasets, an important source of excess zero report counts is *structural zeros* associated with AE/drug pairs that are physically impossible to occur. If structural zeros are present, then the corresponding cells in a contingency table enumerating the number of reported cases of various AE/drug pairs will always be zero, regardless of the total number of reported counts. Moreover, inference on these structural zero positions can be made from the observed data: intuitively, a cell with a positive report count in any observed table cannot be a structural zero, and only an observed zero cell can be a structural zero. This is in contrast with another source of zero-inflation, namely data corruption, that may also occur in adverse events data. Here, due to noise in data recording process some *real* report counts in the AE/drug contingency tables are randomly recorded as zero. Under this setting, there is no population level true zero positions, and the zero inflated positions vary from one sample to another. In this article we primarily focus on the structural zero-type zero-inflation, as it is the primary type of zero-inflation that the FAERS datasets stored in pvLRT feature. Indeed, the FAERS data undergo several rounds of rigorous reviews and checks to safeguard against data corruption. However, we note that pvLRT does provide functions for handling corruption-type zero-inflation; interested readers are referred to the the documentation of the `pvlrt()` function.

Parametric hypothesis testing for signals in pharmacovigilance based on a Zero-inflated Poisson (ZIP) model. To facilitate a unified treatment we consider a zero-inflated Poisson model to describe the test procedure. A discrete random variable X is said to follow a ZIP(θ, ω) distribution where $\omega \in [0, 1)$ and $\theta > 0$ if the probability mass function of X is given by

$$\mathrm{P}(X = x) = \begin{cases} \omega + (1-\omega)\exp(-\theta), & x = 0 \\ (1-\omega)\exp(-\theta)\frac{\theta^x}{x!}, & x = 1, 2, 3, \ldots \end{cases},$$

i.e., if the distribution of X is a $(1-\omega)$ and ω mixture of a Poisson(θ) distribution and a point mass at zero. The model reduces to an ordinary Poisson model when the zero-inflation probability $\omega = 0$. For adverse event data report counts we consider the following zero-inflated Poisson model for the observed count n_{ij} for the i-th AE of a specific drug j: $n_{ij} \sim \mathrm{ZIP}(\lambda_{ij} \times E_{ij}, \omega_j)$, where $E_{ij} = n_{i\bullet} n_{\bullet j} / n_{\bullet\bullet}$ is the expected number of reports for the (i, j)-th pair when there is no association between the pair, and λ_{ij} is the relative reporting proportion. Under this set up, null hypothesis significance tests are performed to determine signals at specific pairs. Suppose we are interested in determining significant adverse events among K out of the J drugs, labeled $1, \ldots, K$. The global null hypothesis is $H_0^{1:K} : \{\lambda_{ij} = 1 \text{ for all } i = 1, \ldots, I \text{ and } j = 1, \ldots, J\}$ and is tested against the *global alternative* $H_a^{1:K} = \{\lambda_{ij} > 1 \text{ for at least one } (i, j) : i = 1, \ldots, I; j = 1, \ldots, K\}$ which represents the hypothesis of "at least one signal' '. If the global null hypothesis is rejected in favor of the global alternative, our focus then shifts on identifying the individual AE/drug pairs with strong associations. This is achieved by post hoc test of the global null hypothesis against the individual alternative hypotheses $H_{a,ij} : \lambda_{ij} > 1$.

Pseudo Likelihood Ratio Test. A pseudo likelihood ratio test of the above hypotheses involves computation of the individual pseudo likelihood ratios

$$\mathrm{LR}_{ij} = \begin{cases} 1 & n_{ij} = 0 \\ \exp(-(\hat{\lambda}_{ij} - 1)E_{ij})\hat{\lambda}_{ij}^{n_{ij}} & n_{ij} > 0 \end{cases},$$

where $\hat{\lambda}_{ij} = \max\{n_{ij}/E_{ij}, 1\}$ is the maximum likelihood estimator of λ_{ij} under $H_0^{1:K} \cup H_a^{1:K}$, and subsequently, the maximum likelihood ratio statistic $\mathrm{MLR}^{1:K} = \max_{i=1,\ldots,I; j=1,\ldots,K} \mathrm{LR}_{ij}$. The global null is rejected in favor of the global alternative hypothesis if $\mathrm{MLR}^{1:K}$ is large, in which case subsequent post hoc tests of individual alternatives $H_{a,ij}$ can be performed based on the observed values of LR_{ij}. Computation of the global and post hoc p-values are described in the following paragraph. Note that the computed or "observed" values of LR_{ij} aid rigorous statistical quantification of the signal strength in a pair (i, j). Thus, we can use these quantities to rank the individual AE/drug pairs in terms of their signal strengths detected in the dataset under consideration. This is particularly useful for determining, the "most prominent" adverse events for a specific drug or a groups of drugs as detected in a dataset. Note also that the functional form of LR_{ij} and hence that of $\mathrm{MLR}^{1:K}$ does not contain the zero-inflation parameters ω_j. However, the null sampling distributions necessary for performing the hypothesis tests do involve them. Computational approximations of these null sampling distributions are described next.

Bootstrapped null sampling distribution of the pseudo likelihood ratio test statistic, and computing p-values. The null sampling distributions of the individual likelihood ratio statistics $\{\mathrm{LR}_{ij}\}$ and the maximum likelihood ratio statistic $\{\mathrm{MLR}^{1:K}\}$ are analytically intractable, and they must be approximated using computational techniques to facilitate inference. In pvLRT we use a parametric bootstrap resampling scheme for this purpose (Davison and Hinkley 1997). To this end, zero-inflated Poisson models are fitted to the observed data under the global null hypothesis $H_{0,ij}^{1:K}$, and null resampled counts $\{\tilde{n}_{ij}\}$ are subsequently generated from the fitted model. Fitting zero-inflated Poisson models requires estimation of the zero-inflation parameters $\{\omega_j\}$ from the observed data. In **pvLRT** the maximum likelihood estimates of $\{\omega_j\}$ are used for this purpose, which are obtained by maximizing the profile likelihood for ω_j:

$$l_j(\omega) = (I - I_{0j}) \log(1-\omega) + \sum_{\{i: n_{ij}=0\}} \log\left(\omega + (1-\omega)\exp(-E_{ij})\right),$$

for $j = 1, \ldots, J$. Here I_{0j} denotes the number of zero n_{ij} values for each drug j. The conditional (posterior) probability of encountering a zero-inflation at the (i, j) pair, given the corresponding observed count n_{ij} is subsequently estimated and computed as:

$$\hat{\eta}_{ij}^{\text{obs}} = \begin{cases} 0, & n_{ij} > 0 \\ \frac{\hat{\omega}_j^{\text{obs}}}{\hat{\omega}_j^{\text{obs}} + (1-\hat{\omega}_j^{\text{obs}})\exp(-E_{ij})} & n_{ij} = 0 \end{cases},$$

for all $i = 1, \ldots, I$ and $j = 1, \ldots, J$. Then, using these estimated conditional zero-inflation probabilities, null parametric bootstrap resamples are generated from $\tilde{n}_{ij} \sim \text{ZIP}(E_{ij}, \hat{\eta}_{ij})$. Under an ordinary Poisson model, the parameters ω_j and η_{ij} do not arise (they are equal to zero from a ZIP model perspective), and the null bootstrap resamples are simply generated from $\tilde{n}_{ij} \sim \text{Poisson}(E_{ij})$; hence, the *bootstrap* null resamples become *exact* null resamples. In either case, these null resamples $\tilde{n}_{ij}$ produce null resampled values of the test statistic $\text{MLR}^{1:K}$, and they collectively approximate the corresponding sampling distributions. The p-value for the global null against the global alternative hypothesis is $P^{1:K} = \text{P}(\text{MLR}^{1:K} \geq \text{MLR}^{1:K,\text{obs}} \mid H_0^{1:K})$, and can be approximated by the proportion of the null resampled $\text{MLR}^{1:K}$ values that exceed the observed value of $\text{MLR}^{1:K,\text{obs}}$:

$$P^{1:K,\text{computed}} = \frac{1 + \sum_{h=1}^{M} \mathbb{1}\left(\widetilde{\text{MLR}}^{1:K,(h)} \geq \text{MLR}^{1:K,\text{obs}}\right)}{M+1},$$

where $\{\widetilde{\text{MLR}}^{1:K,(h)} : h = 1, \ldots, M\}$ are bootstrapped resampled null values of $\text{MLR}^{1:K}$. The global null can be rejected at level α in favor of the global alternative hypothesis if $P^{1:K,\text{computed}} < \alpha$. The post hoc p-values $\{P_{ij} = \text{P}(\text{MLR}^{1:K} \geq \text{LR}_{ij}^{\text{obs}} \mid H_0^{1:K})\}$ for the individual alternative hypotheses $\{H_{a,ij}\}$ are also approximated using the same null resampled values of $\text{MLR}^{1:K}$:

$$P_{ij}^{\text{computed}} = \frac{1 + \sum_{h=1}^{M} \mathbb{1}\left(\widetilde{\text{MLR}}^{1:K,(h)} \geq \text{LR}_{ij}^{\text{obs}}\right)}{M+1}.$$

Note that these post hoc p-values are based on the null sampling distribution of the *same* random test statistic $\text{MLR}^{1:K}$; only the thresholds where the tail probabilities are evaluated are different (the individual observed $\text{LR}_{ij}^{\text{obs}}$ values). Consequently, no inflation in the overall type I error occurs due to multiplicity of hypothesis tests with independent test statistics. This also ensures that the false discovery rate of the entire procedure is controlled at the same level as the overall type I error.

Model selection: determining whether to use a Poisson model or a ZIP model on a given dataset. As noted before, the ZIP model above assumes that the report counts for a specific drug j are zero-inflated with probability ω_j. If ω_j is small, then the ZIP model effectively reduces to an ordinary Poisson model; however in general, the ZIP model produces probabilities that are different from the Poisson probabilities. A natural question of particular practical importance is therefore which of these two models should be used on a given dataset. Clearly, if there is knowledge about the existence (or non-existence) of structural zero pairs in a given dataset (the exact positions of the structural zeros may possibly be unknown), then one should use a ZIP (ordinary Poisson) model. However, in practice no such knowledge may be available, and the optimal model must be determined entirely on the basis of the observed data. In other words, one must address what is known as a model selection problem. Because we are in a parametric likelihood framework, a convenient and canonical approach would involve the use of some penalized likelihood model selection criterion, such as the AIC or the BIC (Sakamoto, Ishiguro, and Kitagawa 1986) defined as:

$$\text{AIC}(k) = -2\log(\hat{L}) + kp,$$

where $\hat{L}$ denotes the maximized value of the likelihood function for a given model, k is a penalty parameter, and p is the total number of parameters in the model. The choice $k = 2$ produces the classical AIC, and $k = \log N$ where N denotes the total number of data points, produces the BIC. The model with the minimum AIC (or BIC) is regarded as optimal. Note that in the current context for both the Poisson and the ZIP models, $\log(\hat{L})$ is obtained from the unrestricted model with each $\{\lambda_{ij}\}$ assuming values in $[1, \infty)$; the total number of parameters are $p = I \times J$ in the Poisson model and $(I+1)J$ in the ZIP model, and the total sample size N is the grand total $n_{\bullet\bullet}$ of the contingency table.

Testing for zero-inflation among individual drugs. The above model selection procedure addresses the question of whether to use the ZIP model or an ordinary Poisson model for a given dataset. When a ZIP model is determined to be optimal, interest may then lie in determining the statistical strength/significance of individual zero-inflation parameters $\{\omega_j\}$ specific to the individual drugs. In a null hypothesis significance testing framework, this can be translated as a test of $H^{\omega}_{0,j^*} : \omega_{j^*} = 0$ against $H^{\omega}_{a,j^*} : \omega_{j^*} > 0$, for each drug j^*. The likelihood ratio test statistic for this problem is $\mathrm{LR}^{\omega}_{j^*} = \exp\left[l_j(\hat{\omega}_{j^*}) - l_j(0)\right]$, where $l_{j^*}(\omega)$ is the profile log likelihood function for ω_{j^*} as described before, and $\hat{\omega}_{j^*}$ is the corresponding maximum likelihood estimator obtained by maximizing $l_{j^*}(\omega)$. Similar to the LRT for λ_{ij} described before, the null sampling distribution of $\mathrm{LR}^{\omega}_{j^*}$ is analytically intractable, but can be approximated using a parametric bootstrap. To this end, null resamples $\{\tilde{n}_{ij^*}\}$ are generated from the $\mathrm{ZIP}(\hat{\lambda}_{ij^*} E_{ij^*}, \omega_{j^*} = 0) \equiv \mathrm{Poisson}(\hat{\lambda}_{ij^*} E_{ij^*})$ distribution, where $\hat{\lambda} ij^* = \max\{n_{ij^*}/E_{ij^*}, 1\}$ is the maximum likelihood estimator of λ_{ij^*}. From these null resamples $\{\tilde{n}_{ij}\}$, null values of the likelihood ratio test statistics $\{\widetilde{\mathrm{LR}}^{\omega}_{j^*}\}$ are computed. The p-value of the test of H^{ω}_{0,j^*} against H^{ω}_{a,j^*} is subsequently computed using Monte Carlo estimated probability (proportion) of *null* $\{\widetilde{\mathrm{LR}}^{\omega}_{j^*}\}$ values that are bigger than or equal to the "observed" value $\mathrm{LR}^{\omega,\text{ obs}}_{j^*}$. If zero-inflation in several drugs j^* are tested simultaneously, then the overall p-value needs to be adjusted for multiplicity; in **pvLRT** the Benjamini-Hochberg adjusted p-values (q-values; Benjamini and Hochberg (1995)) are used by default for this purpose.

3 Adverse Event Data Analysis with pvLRT

In this section we provide data analysis examples based on four real datasets contained in the **pvLRT** package. These datasets are stored as pre-processed contingency tables (in matrix-like forms) cataloging the aggregated report count for each AE/drug pair. To produce such contingency tables from raw adverse event data the **pvLRT** function `convert_raw_to_contin_table()` may be used. We note, however, that care must be taken before summarizing raw data into such contingency tables. First, a decision needs to be made regarding which reports to include in the analysis, e.g., only consider primary and/or secondary suspect drugs or also include concomitant and interacting drugs as cataloged in the FAERS database. The processed data stored in **pvLRT** only considers reports for the primary and secondary suspect drugs. Second, one needs to identify which drugs to include in the "Other Drug" category which plays the key role of "baseline" drugs. Appropriate specification of this baseline category is required because of its contribution to the total report counts for each AE, even though the AE/baseline drug category associations are not of primary importance. Typically, one identifies a handful of drugs of particular interest (e.g., the statins as discussed in the examples below) and combines all remaining drugs as baseline (Ding, Markatou, and Ball 2020). Finally, one needs to determine which AEs to include in the testing. The proposed LRT-based methods permit strong control over the global type I error and false discovery rates while ensuring high power (Chakraborty et al. 2022), regardless of the total number of AEs. However, computation of the bootstrap p-values become increasingly more expensive with a growing number of AEs, and thus in practice one may encounter certain upper bounds to the total number of AEs that can be tested at a given time. We refer to Ding, Markatou, and Ball (2020) for a detailed discussion on these considerations. The analyses provided below (divided into subsections) showcase examples with both small ($I = 46$) and large ($I \approx 6000$) number of AEs. We start by loading the **pvLRT** package into in `R` using `library()`.

```
library(pvLRT)
```

Analysis of the statin dataset with 47 adverse events

We begin with an analysis of the statin dataset labeled `statin46`, previously identified as being associated with the statin drugs. There are 47 rows in the data, corresponding to these 46 adverse events, and a collapsed category labeled "other" that consists of all other adverse events tabulated in the FAERS database during this time span. An older version of the data with the same 46 adverse events have been previously considered in (Ding, Markatou, and Ball 2020); our analysis that follows corroborates findings from that study. To begin the analysis, we first load the data into an R session and view the first few rows and columns with the following codes:

```
data("statin46")
head(statin46)[, 1:3]
```

```
#>                                           Atorvastatin Fluvastatin Lovastatin
#> Acute Kidney Injury                               1353          42          7
#> Anuria                                              71           0          0
#> Blood Calcium Decreased                             14           2          0
#> Blood Creatine Phosphokinase Abnormal               34           0          0
#> Blood Creatine Phosphokinase Increased            1175         125         32
#> Blood Creatine Phosphokinase Mm Increased            2           0          0
```

Our interest lies in finding the most important adverse events of these 6 statin drugs. We note at the outset that the adverse events potentially caused by these drugs cannot be considered in isolation due to similarity among functions of statins, and thus we must consider all 6 drugs simultaneously while determining their important adverse events. This is achieved by performing a simultaneous test for all drugs of interest collectively within an *extended LRT* framework, which ensures that the overall type-I error of the process is preserved and the false discovery rate is controlled. See the previous section for a note on the statistical properties of the *extended LRT* test.

Analysis based on an ordinary Poisson model. We first perform an LRT analysis based on the ordinary Poisson model on this dataset. The R codes for performing this task are as follows.

```
set.seed(100)
test_statin46_poisson <- pvlrt(
  statin46,
  zi_prob = 0,
  test_drug_idx = 1:6
)
```

The first argument specifies the contingency table on which the LRT is to be performed, which is `statin46` in the current analysis. The argument `zi_prob = 0` pre-specifies the zero-inflation probability in the model for each drug to be 0, thus reducing the model to an ordinary Poisson model. Note that the ordinary Poisson model-based test can also be invoked through the wrapper function `lrt_poisson()`. Under the hood, `lrt_poisson()` sets `zi_prob = 0` and then calls `pvlrt()` itself. Next, the argument `test_drug_idx = 1:6` indicates that hypothesis tests are to be performed only on the first 6 columns, i.e., excluding the last column for "other" drugs. By default, all drugs supplied in `test_drug_idx` are tested simultaneously as a single group/class in an extended LRT framework; if other class/group structures exist among the drugs or if the drugs are to be tested separately (i.e., each tested drug forms its own class) those can be passed through `drug_class_idx`.

Running the code above creates a `pvlrt` S3 object. A `pvlrt` object is simply a reclassified `matrix` of computed (log) LRTs; it has the same dimensions as the input `contin_table`, with each cell containing the computed test statistic value for the corresponding AE/Drug pair. However, in addition to the usual `dimnames` attributes, a `pvlrt` object has several other attributes including the p-values associated with these log LRTs and estimates for zero-inflation parameters, and some meta-data including the type of test performed, the input data, etc. The printing method for `pvlrt` objects provides a high-level textual summary of the test. To glance at the overall results from the above LR tests, we simply print `test_statin46_poisson` on the R console:

```
test_statin46_poisson
```

```
#> Relative reporting rate (lambda)-based ordinary-LRT on 47 AE & 7 drugs.
#> Hypothesis tests performed on 6 drugs using parametric bootstrap.
#>
#> Running time of the original pvlrt call: 1.59371 secs
#>
#> No zi considered in the model.
#> Total 110 AE-drug pairs are significant at level = 0.05.
#>
#> Extract all LR statistics and p-values using `summary()`.
```

The above output shows that there are 110 significant AE/drug pairs at level $\alpha = 0.05$. The likelihood ratio test statistics and p-values for all AE/drug pairs can be extracted using summary(). This will produce a data table, with each row providing the sample size, likelihood ratio, and p-value for each AE/drug pair present in the data[1]:

```
# summary(test_statin46_poisson)
```

The output is omitted for brevity. Note that for pairs that are not tested (here, all entries in the 7-th column of the statin46 table) the test results will be missing (denoted by NA in R). **pvLRT** currently has implementations for three visual summary methods for the test results, namely bubble-plot, barplot, and heatmap. These can either be accessed explicitly through the functions bubbleplot_pvrlt, barplot (which is an S3 method for pvlrt objects), and heatmap_pvlrt, or directly through the plot method for pvlrt objects with the specifications of type = "bubbleplot" or type = "barplot", or type = "barplot" respectively. As noted before in the Introduction, these plots are constructed using the **ggplot2** package (Wickham 2016). As a result, several aspects of the produced plots can be updated post hoc, as facilitated by the principles of *grammar of graphics* that **ggplot2** implements. Examples of resizing axes label text sizes for pvlrt plots are provided below.

In the following we first create a bubble-plot top 15 AE/drug pairs across all 6 statin drugs, with pairs being ranked by the magnitudes of their likelihood ratio test statistic values. The figure is stored in pl_bubble_statin46_poisson, which is visualized upon printing.

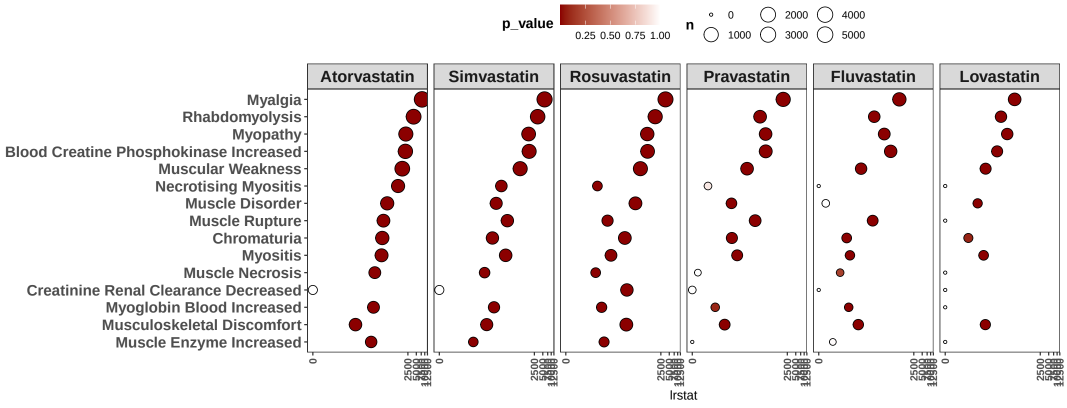
```
pl_bubble_statin46_poisson <- plot(
  test_statin46_poisson,
  type = "bubbleplot",
  x_axis_measure = "lrstat",
  size_measure = "n",
  fill_measure = "p_value",
  AE = 15
)
```

The arguments used in plot() above are described as follows. The first two arguments specify the pvlrt object under consideration and the type of plot to be made. All three visualizations implemented (bubbleplot, barplot and heatmap) display the test results by plotting AEs across the rows and Drugs across the columns, in their respective plots. The arguments x_axis_measure = "lrstat", size_measure = "n", and fill_measure = "p_value" indicate that the log LRT values will be displayed along the x-axis, the circle sizes in the bubbleplot will represent the corresponding sample size of an AE/drug pair, and that the circles will be color coded according to the p-values of the corresponding test statistics, respectively. Note that the three dimensions (lrstat, n and p_value) displayed through these three arguments can be interchanged. The argument AE = 15 indicates that the top 15 adverse events with the largest log likelihood ratio test statistic values are to be plotted. Here these “largest” values are obtained by comparing across all drugs tested during the original pvlrt run. Instead of looking at “all” drugs while determining the top AEs, one can also focus on a specific subset of drugs by supplying the appropriate subset through the argument Drug. Moreover, instead of looking at the top adverse events, one can also explicitly specify which adverse events to display by directly passing their names through the same AE argument. When passing names through AE and Drug, partial string patterns can be supplied and then matched against their full names via regular expressions; this is achieved by setting grep = TRUE. For example, specifying AE = "muscle" and setting grep = TRUE will show all adverse events that have the character "muscle" in its name. We refer to the package documentation for further details on these two arguments. To visualize the plot produced, we can simply print pl_bubble_statin46_poisson on the R console or save it to a graphic device via ggsave. As noted before, post hoc editing of various graphical parameters in pl_heat_statin46_poisson can be made using **ggplot2** functions. Note that to use ggplot2 functions, we need to load the ggplot2 package. Below we show how the axes label sizes and legend text sizes can be modified. (Changing the default labels etc. may cause **ggplot2** to throw some warnings). We refer to the **ggplot2** package documentations and vignettes for further details on modifications of **ggplot2** objects and saving **ggplot2** objects to graphic devices.

```
library(ggplot2)
```

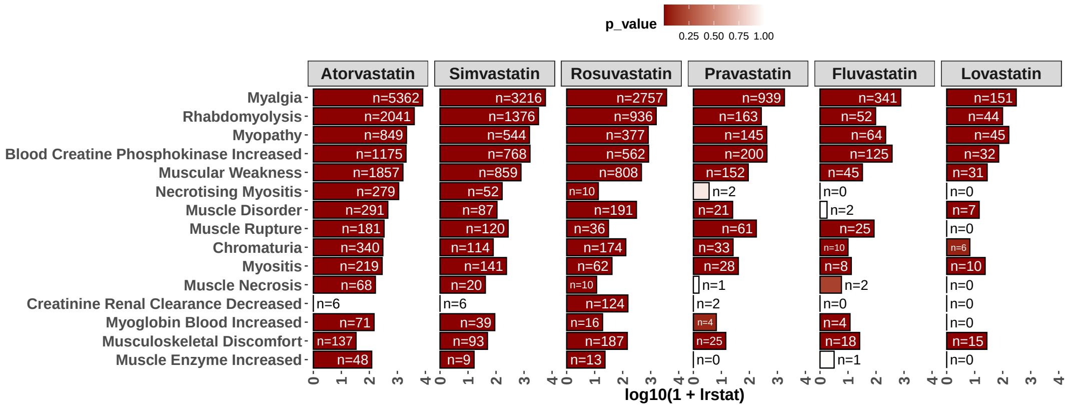
[1] The default printing specifications for a data.table object applies to the resulting summary, which abbreviates the output to show results for the top 5 and bottom 5 AE/drug pairs (arranged with respect to the computed LRT values) on the console for large summary tables. To print more pairs, use e.g., summary(test_statin46_poisson) %>% print(n) with a larger n. Storing the summary output to an object, e.g., summary_test <- summary(test_statin46_poisson) allows access to results from all pairs.

```
pl_bubble_statin46_poisson +
  theme(
    axis.text.y = element_text(size = 13, face = "bold"),
    axis.text.x = element_text(size = 9, face = "bold"),
    legend.title = element_text(size = 12, face = "bold"),
    strip.text = element_text(size = 14, face = "bold"),
    legend.position = "top"
  )
```



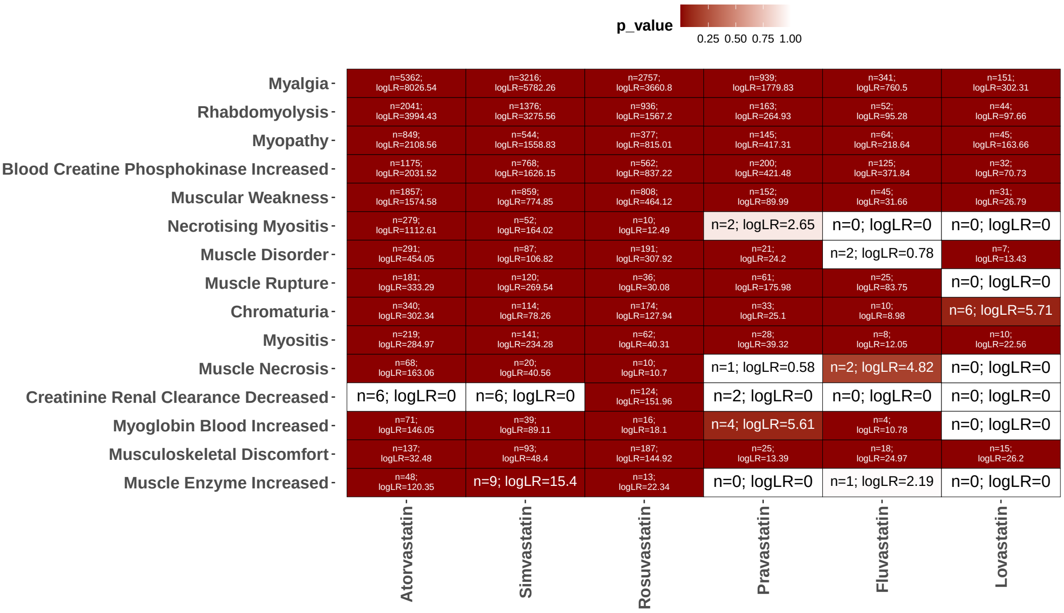
The above bubbleplot visualizes three simultaneous dimensions, viz., `lrstat`, `n` and `p_value` along the x-axis, circle size, and circle color respectively. An alternative mode of visualization is made by paneled barplots which can visualize two simultaneous dimensions, e.g., the x-axis showing the log likelihood ratio statistic, and the colors representing p-values as in the bubbleplot. The sample sizes can be displayed on the bar as text, by specifying `show_n = TRUE`. The following R codes create the barplot.

```
pl_bar_statin46_poisson <- plot(
  test_statin46_poisson,
  type = "barplot",
  fill_measure = "p_value",
  x_axis_measure = "lrstat",
  AE = 15,
  show_n = TRUE,
  border_color = "black"
)
pl_bar_statin46_poisson +
  theme(
    axis.text.y = element_text(size = 13, face = "bold"),
    axis.text.x = element_text(size = 13, face = "bold"),
    axis.title.x = element_text(size = 14, face = "bold"),
    legend.title = element_text(size = 12, face = "bold"),
    strip.text = element_text(size = 14, face = "bold"),
    legend.position = "top"
  )
```



Finally, we can visualize only one dimension, say `p_value` as cell-colors in a heatmap, and inscribe as texts the other two dimensions, i.e., `lrstat` and `n` (by specifying `show_n = TRUE` and `show_lrstat = TRUE`). The following R codes perform this task.

```
pl_heat_statin46_poisson <- plot(
  test_statin46_poisson,
  type = "heatmap",
  fill_measure = "p_value",
  AE = 15,
  show_n = TRUE,
  show_lrstat = TRUE,
  border_color = "black"
)
pl_heat_statin46_poisson +
  theme(
    axis.text = element_text(size = 13, face = "bold"),
    legend.title = element_text(size = 12, face = "bold"),
    strip.text = element_text(size = 14, face = "bold"),
    legend.position = "top"
  )
```



Note that all three plots above can be made interactive using external R packages; consider, e.g., `plotly::ggplotly(pl_bubble_statin46_poisson)`. Such interactive plots can be particularly useful in exploratory analyses performed on large adverse event datasets. We refer to the **plotly** (Sievert 2020) package documentation for further details on `plotly::ggplotly` and other interactive plotting functions. The html version of this article displays an interactive variant of the above figure.

The following observations are made from the above plots. First, Myalgia appears to be the AE with the largest computed log likelihood ratio (LR) statistic across all 6 statin drugs combined. It has the largest LR in Atorvastatin, followed by Simvastatin, Rosuvastatin, Pravastatin, Fluvastatin, and Lovastatin. Next is Rhabdomyolysis, which displays a near similar pattern as Myalgia in terms of computed signal/association strength among the six drugs, except here Lovastatin has a slightly larger computed LR value than Fluvastatin. Second, not all of the 15 adverse events are statistically significant across all six statin drugs. For example, Necrotising Myositis is a statistically significant adverse event of Atorvastatin, Simvastatin, and Rosuvastatin, but not for Pravastatin, Fluvastatin, and Lovastatin. Third, the sample size n_{ij} of a specific AE/drug pair (i, j) alone does not provide much information about its strength of association; for example, the pairs (Muscle Necrosis, Rosuvastatin), (Myositis, Lovastatin), (Chromaturia, Fluvastatin), and (Necrotising Myositis, Rosuvastatin) all have sample sizes $n_{ij} = 10$, but their computed likelihood ratio statistics, and hence the strengths of their associations vary. Moreover, a pair with a smaller sample size may harbor a stronger signal than one with a larger sample size; for example for Fluvastatin, Chromaturia has a sample size of $n_{ij} = 10$ and Myoglobin Blood Increased has a sample size of $n_{ij} = 4$; yet the former AE has a smaller LRT statistic value than the latter.

Analysis based on a zero-inflated Poisson model. We now consider zero-inflated Poisson model-based LRTs to identify signals in the `statin46` data. To this end, the same `pvlrt` function can be used, with two changes in the parameters: (a) the argument `zi_prob` is now set to `NULL`, which ensures that the zero-inflation parameters are estimated from the data, and (b) the logical argument `test_zi` is now set to `TRUE`, which specifies that a separate (pseudo) likelihood ratio test for the significance of each drug-specific zero inflation parameter will be performed. The following are the R codes used:

```
set.seed(100)
test_statin46_zip <- pvlrt(
  statin46,
  test_drug_idx = 1:6,
  zi_prob = NULL,
  test_zi = TRUE
)
```

A textual summary of the analysis can be obtained by printing the output of `test_statin46_zip` on the R console:

```
test_statin46_zip
```

```
#> Relative reporting rate (lambda)-based pseudo-LRT on 47 AE & 7 drugs.
#> Hypothesis tests performed on 6 drugs using parametric bootstrap.
#>
#> Running time of the original pvlrt call: 6.675102 secs
#>
#> Drug-specific estimated zi probabilities: 0 (Atorvastatin), 0.07
#> (Fluvastatin), 0.16 (Lovastatin), 0.06 (Pravastatin), 0 (Rosuvastatin),
#> 0 (Simvastatin), 0 (Other)
#>
#> LR test of significance for zi probabilities: LRstat= 0.00, q>0.90
#> (Atorvastatin), LRstat= 5.28, q<0.001 (Fluvastatin), LRstat= 3.81,
#> q<0.001 (Lovastatin), LRstat=32.49, q<0.001 (Pravastatin), LRstat=
#> 0.00, q>0.90 (Rosuvastatin), LRstat= 0.00, q>0.90 (Simvastatin),
#> LRstat= 0.00, q=<NA> (Other)
#> Total 110 AE-drug pairs are significant at level = 0.05.
#>
#> Extract all LR statistics and p-values using `summary()`.
```

It follows from the output that the estimated drug-specific zero-inflation probabilities are significantly different from zero (q-value or FDR adjusted p-value < 0.05) only for Fluvastatin and Lovastatin, where the point estimates of the zero-inflation parameters are $\omega = 0.07$ and $\omega = 0.16$ respectively. The total number of significant AE/drug is 110, which is the same as the number of significant pairs detected from the Poisson model. To see which 110 pairs detected to be significant in the two models, we can use the function `extract_significant_pairs` to extract a `data.table` containing the test results of significant AE/drug pairs and at a given level $\alpha = 0.05$ from the two models, and subsequently compare these results.

```
# extract_significant_pairs(test_statin46_poisson)
# extract_significant_pairs(test_statin46_zip)
all.equal(
  extract_significant_pairs(test_statin46_poisson),
  extract_significant_pairs(test_statin46_zip)
)
```

```
#> [1] "Column 'p_value': Mean relative difference: 0.06278027"
```

As it appears the above two `pvlrt` objects differ only in the computed p-values, but not in any other fields/statistics. That the computed LR statistics are identical is not surprising, since the functional form of the test statistic is exactly the same for the two models. The p-values, on the other hand, depend on the null sampling distributions of LR statistics, which are different for the two models. Moreover, here the p-values are numerically computed using Monte Carlo methods based on bootstrap resampling, which means that even for the same null sampling distributions, the computed p-values may show random variations between two instances of the computation. However, that there is no difference between the identified significant (computed p-value < 0.05) AE/drug pairs for the two models essentially indicates concordance of the signals identified.

While the two models identify the same AE/drug pairs to be significant at the 0.05 level, they may differ in terms of predictive accuracy/goodness of fit, which can be compared through model selection criteria such as BIC. The logLik methods for pvlrt objects implemented in the package (see the documentation for logLik.pvlrt) allow automated computations of the BIC and AIC of the fitted models, using the stats S3 functions BIC and AIC. Below we use these two functions to compare the predictive accuracies of the fitted Poisson and ZIP models.

```
cbind(BIC(test_statin46_poisson, test_statin46_zip),
      AIC(test_statin46_poisson, test_statin46_zip))[, -3] # remove duplicated df col

#>                        df      BIC      AIC
#> test_statin46_poisson 329 15962.47 10707.02
#> test_statin46_zip     336 16005.13 10637.85
```

It follows that the Poisson model has smaller BIC but larger AIC than the ZIP model for the current dataset. This essentially implies that the Poisson model and the ZIP model both perform equally well in modeling the statin46 dataset.

Analysis of the full statin data with 6039 adverse events

We next analyze the full statin dataset with all 6039 adverse events. This dataset differs from the smaller statin46 dataset in that it explicitly catalogs report-counts of many adverse events that are grouped together as "Other AE" in statin46. The first few rows of the bigger statin dataset can be displayed using head(statin); we omit the output for brevity.

```
data("statin")
# head(statin)
```

In the following we first run a Poisson LRT, followed by a ZIP LRT, and then compare the two models. The model that fits the data better (assessed through BIC/AIC) will subsequently be used for inference. The following commands run a Poisson LRT on the statin dataset, with the default 10,000 resamples for computation of p-values. The remaining arguments are analogous to the case of analysis of statin46. Once created, the pvlrt objects will output brief textual summary of the results; the outputs are omitted for brevity.

```
set.seed(100)
test_statin_poisson <- pvlrt(
  statin,
  zi_prob = 0,
  test_drug_idx = 1:6
)
# test_statin_poisson # print the results
```

Next we run the zero-inflated Poisson model. Codes are as follows.

```
set.seed(100)
test_statin_zip <- pvlrt(
  statin,
  test_drug_idx = 1:6,
  zi_prob = NULL,
  test_zi = TRUE
)
# test_statin_zip # print the results
```

As it appears, in this bigger dataset all drugs (except the "other" group) have > 10% zero inflation probabilities with statistically significant FDR adjusted p-values (q-values). This indicates that the zero-inflated Poisson model based test would likely be more appropriate here. We formally check this via model comparison using AIC and BIC as follows:

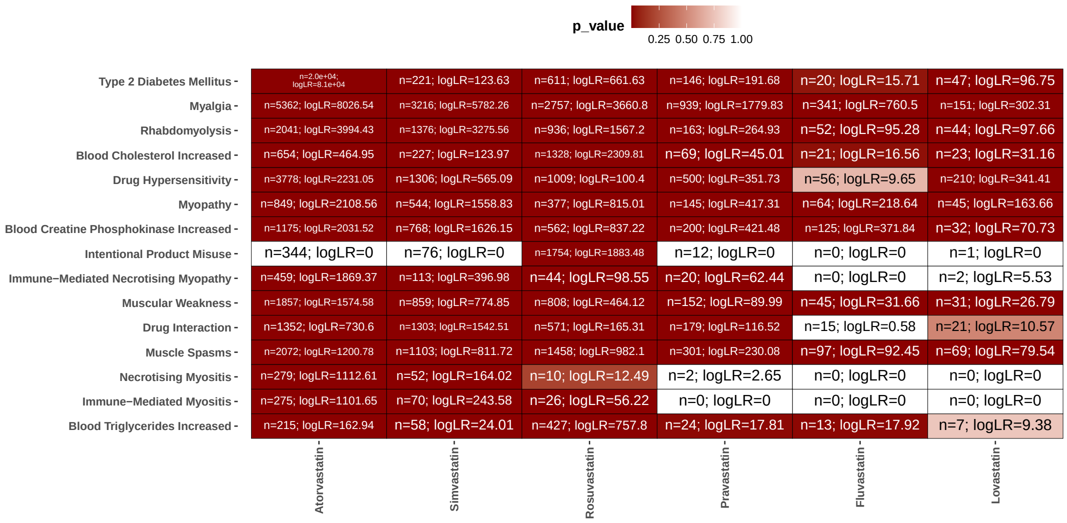
```
cbind(BIC(test_statin_poisson, test_statin_zip),
      AIC(test_statin_poisson, test_statin_zip))[, -3] #remove duplicated df col
```

```
#>                         df     BIC      AIC
#> test_statin_poisson 42273 1063654 388384.2
#> test_statin_zip     42280 1043314 367931.7
```

It follows that here the zero-inflated Poisson model has both smaller AIC and BIC than the Poisson model, thus confirming our suspicion that the former model is a better fit for the data. We shall therefore conduct the subsequent analysis based on the zero-inflated Poisson model-based tests.

To get a visual understanding of the results we plot a heatmap of the p-values with LR test statistics and samples sizes inscribed as texts:

```
pl_heat_statin_zip <- heatmap_pvlrt(
  test_statin_zip,
  AE = 15,
  show_n = TRUE,
  show_lrstat = TRUE
)
pl_heat_statin_zip +
  theme(
    axis.text = element_text(face = "bold"),
    legend.title = element_text(face = "bold"),
    strip.text = element_text(face = "bold"),
    legend.position = "top"
  )
```



The following observations are made. First, (Type 2 Diabetes Mellitus, Atorvastatin) appears to be the *most prominent* association in terms of LR statistic (log LR = 8.0632317×10^4) across all AE/drug pairs. The next four prominent signals are observed in (Myalgia, Atorvastatin), (Myalgia, Simvastatin), (Rhabdomyolysis, Atorvastatin), and (Myalgia, Rosuvastatin) respectively.

Analysis of the full gbca data with 1707 adverse events

We next analyze the full gbca dataset with all 1707 adverse events and 9 drugs. This example arises in the context of an FDA evaluation of Gadolinium-based contrast agents (GBCA) from which FDA concluded there were no adverse events from GBCA retention (https://www.fda.gov/drugs/postmarket-drug-safety-information-patients-and-providers/information-gadolinium-based-contrast-agents). Several disproportionality methods were compared in the FAERS database (Zhao, Yi, and Tiwari 2018) using the GBCAs as an example. We use this example to further illustrate the LRT methods, and assess their computational efficiency in datasets with many zero-report counts. The observed proportions of zero counts are noticeably large in most drugs:

```
data("gbca")
# head(gbca) # show the first few rows

obs_prop_0 <- apply(gbca, 2, function(x) mean(x == 0))
round(obs_prop_0, 2)
```

```
#>      Gadobenate      Gadobutrol     Gadodiamide    Gadofosveset   Gadopentetate
#>            0.60            0.42            0.71            0.99            0.61
#>      Gadoterate     Gadoteridol Gadoversetamide      Gadoxetate           Other
#>            0.51            0.70            0.87            0.82            0.00
```

These large proportions of zero counts provide a heuristic justification for using a ZIP-model based LRT in this example: because the total zero counts comprise both *true/structural* zero counts (zero-inflation) and *sampling* zero counts, some substantial parts of these observed zeros may be attributable to zero inflation. To formalize this intuition, in the following we first run a Poisson LRT, followed by a ZIP LRT, and then compare goodness of fits for the two models in the dataset. The model that fits the data better (assessed through BIC/AIC) will be subsequently used for inference. We begin with the Poisson LRT, with the default 10,000 resamples for computation of p-values. The codes are as follows. The resulting output, when printed on console will display textual summary of test results which we omit for brevity.

```
set.seed(100)
test_gbca_poisson <- pvlrt(
  gbca,
  zi_prob = 0,
  test_drug_idx = 1:9
)
# test_gbca_poisson ## print results
```

Next we run the zero-inflated Poisson model. Codes are as follows.

```
set.seed(100)
test_gbca_zip <- pvlrt(
  gbca,
  test_drug_idx = 1:9,
  zi_prob = NULL,
  test_zi = TRUE
)
```

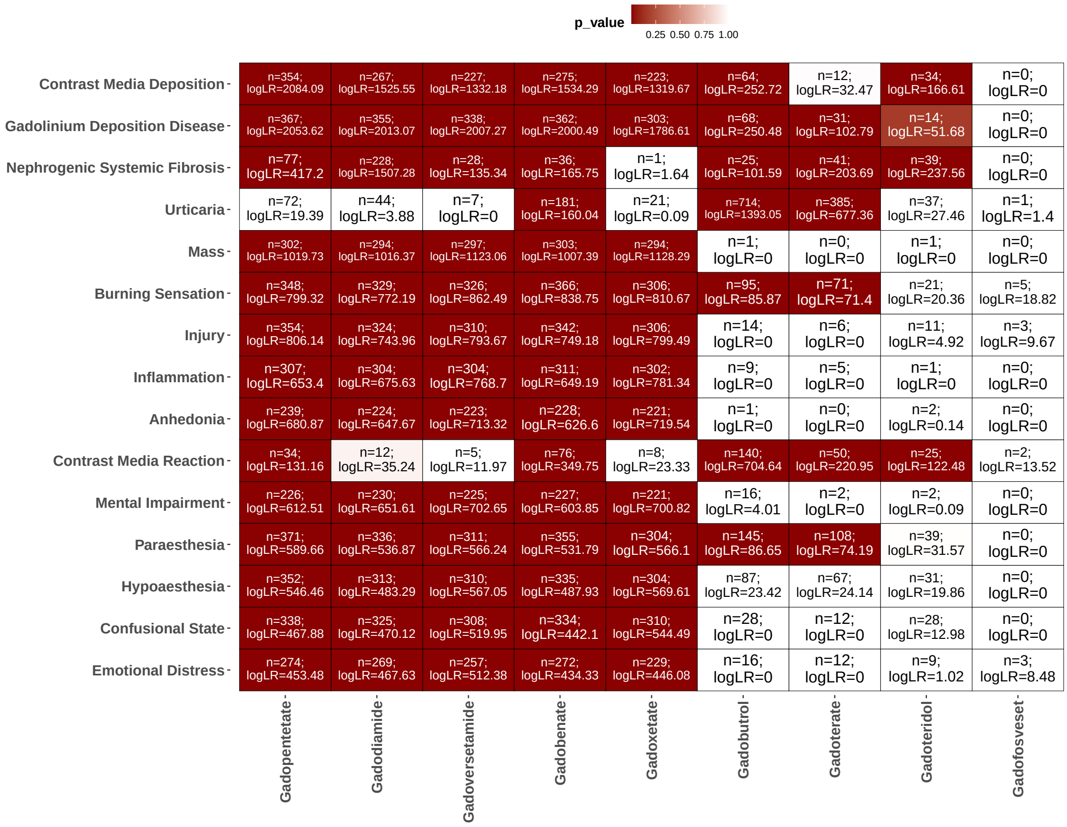
It follows that several drug groups have substantially large (> 40%) zero inflation probabilities with statistically significant FDR adjusted p-values (q-values). This agrees with the earlier exploratory observation of large proportions of observed zeros. Similar to the full `statin` data analysis, it therefore seems that the zero-inflated Poisson model would be more appropriate here. We use AIC and BIC to formally check this:

```
cbind(BIC(test_gbca_poisson, test_gbca_zip),
      AIC(test_gbca_poisson, test_gbca_zip))[, -3] # remove

#>                      df      BIC      AIC
#> test_gbca_poisson 17070 416208.7 147441.5
#> test_gbca_zip     17080 384316.9 115392.2
```

Indeed, both AIC and BIC are smaller for the zero-inflated Poisson model than the Poisson model. We therefore use the former model for subsequent analysis.

To get a visual understanding of the results we plot a heatmap of the p-values with LR test statistics and samples sizes inscribed as texts:

```
pl_heat_gbca_zip <- heatmap_pvlrt(
  test_gbca_zip,
  AE = 15,
  show_n = TRUE,
  show_lrstat = TRUE
)
pl_heat_gbca_zip +
  theme(
    axis.text = element_text(size = 13, face = "bold"),
    legend.title = element_text(size = 12, face = "bold"),
    strip.text = element_text(size = 14, face = "bold"),
    legend.position = "top"
  )
```



The following observations are made. First, (Contrast Media Deposition, Gadopentetate) appears to be the most prominent AE/drug pair in terms of LR statistic (log LR = 2084.09) across all AE/drug pairs. The next four prominent signals are observed in (Gadolinium Deposition Disease, Gadopentetate), (Gadolinium Deposition Disease, Gadodiamide), (Gadolinium Deposition Disease, Gadoversetamide), and (Gadolinium Deposition Disease, Gadobenate) respectively.

Analysis of the rotavirus vaccine datasets `rv`, `rvold`, and `rvyoung`

Finally, to exemplify the use of pvLRT in analyzing vaccine safety we consider rotavirus vaccine datasets. These vaccines are an important example for vaccine safety assessment. Intussusception, a telescoping of the intestines that can cause serious illness, had led to the withdrawal of the first rotavirus vaccines from the market. Several AEs have since been identified in analyses of the FDA/CDC VAERS data (Niu, Erwin, and Braun 2001; Haber et al. 2004) as being associated with intusscusception. Below in our analysis we consider several of these vaccine/AE combinations; these combinations are well-known and well-studied and hence they aid informative assessment of the performance of the disproportionality methods. In pvLRT, the rotavirus vaccine datasets are obtained from the FDA VAERS database for the year of 1999, and they summarize adverse event report counts for two vaccine groups, namely the rotavirus vaccine ("Rv") and "Other" vaccines (37 different vaccines combined). pvLRT contains three such datasets namely `rvold`, `rvyoung`, and `rv` tabulating the number of occurrences of 727, 346, and 794 AEs among individuals of "old" (age $\geq$ 1 year), "young" (age $<$ 1 year), and "combined" (collective old and young) age groups respectively, with most counts in the combined dataset arising from the young age group. Our interest here lies in identifying the most prominent AEs of the rotavirus vaccine and then comparing them across the three datasets/age groups. The rotavirus data has been previously analyzed in the literature using other approaches such as the EBGM (Niu, Erwin, and Braun 2001) and proportional morbidity analysis for vaccine safety (Haber et al. 2004). Here we will utilize the LRT method as implemented in pvLRT: first, in each dataset we will perform an LRT with an *optimally selected* model (Poisson or ZIP); subsequently, based on the LRT results we will determine a few prominent AEs with the largest LR statistics in each dataset and compare these AEs across datasets.

We first fit Poisson and ZIP models to the three datasets:

```
set.seed(100)
test_rv_poisson_list <- lapply(
  list(old = rvold, young = rvyoung, combined = rv),
  function(this_data) {
    pvlrt(
      this_data,
      zi_prob = 0,
      test_drug_idx = 1
    )
  }
)

set.seed(100)
test_rv_zip_list <- lapply(
  list(old = rvold, young = rvyoung, combined = rv),
  function(this_data) {
    pvlrt(
      this_data,
      test_drug_idx = 1,
      zi_prob = NULL
    )
  }
)
```

The objects test_rv_poisson_list and test_rv_zip_list list the Poisson and ZIP LRT results for the three datasets respectively. The estimated zero-inflation probabilities for the "Rv" vaccine can be extracted from the latter list as follows. In the code chunks below we use the **magrittr** (Bache and Wickham 2020) forward pipe %>% infix operator for better readability of chained computations. The operator is automatically loaded with **pvLRT**; see the **magrittr** package manual for further details on the pipe operator.

```
rv_zi_prob <- test_rv_zip_list %>%
  sapply(function(test) extract_zi_probability(test)["Rv"]) %>%
  round(3)
rv_zi_prob

#>      old.Rv    young.Rv combined.Rv
#>       0.248       0.000       0.215
```

It follows from the above estimated zero inflation probabilities that a ZIP model could be appropriate for the combined and the old age groups whereas an ordinary Poisson model is likely adequate for the young age groups. We formalize the model selection via AIC and BIC, and find the optimal models for the three datasets. Codes are as follows.

```
compare_models_rv <- mapply(
  function(test_poisson, test_zip, data_name) {
    AIC_both <- AIC(test_poisson, test_zip)$AIC
    BIC_both <- BIC(test_poisson, test_zip)$BIC
    data.frame(
      name = data_name,
      AIC_poisson = AIC_both[1],
      BIC_poisson = BIC_both[1],
      AIC_zip = AIC_both[2],
      BIC_zip = BIC_both[2],
      optimal_AIC = c("poisson", "zip")[which.min(AIC_both)],
      optimal_BIC = c("poisson", "zip")[which.min(AIC_both)]
    )
  },
  test_rv_poisson_list,
  test_rv_zip_list,
  names(test_rv_poisson_list),
  SIMPLIFY = FALSE
) %>%
  unname() %>%
  do.call(rbind, .)

compare_models_rv
```

```
#>       name AIC_poisson BIC_poisson  AIC_zip  BIC_zip optimal_AIC optimal_BIC
#> 1      old    5749.153   18631.535 5751.783 18651.88     poisson     poisson
#> 2    young    3829.600    9000.445 3833.600  9019.39     poisson     poisson
#> 3 combined    8308.330   22800.414 8051.049 22561.38         zip         zip
```

It follows from the above AIC and BIC values that while the ZIP model is a clear winner for the combined age group dataset, the Poisson model displays slightly smaller AIC and BIC values in the other two datasets. We therefore first create a list of *optimal* tests from these two models fitted to the three datasets, and use these optimal models for subsequent determination of prominent AEs as follows:

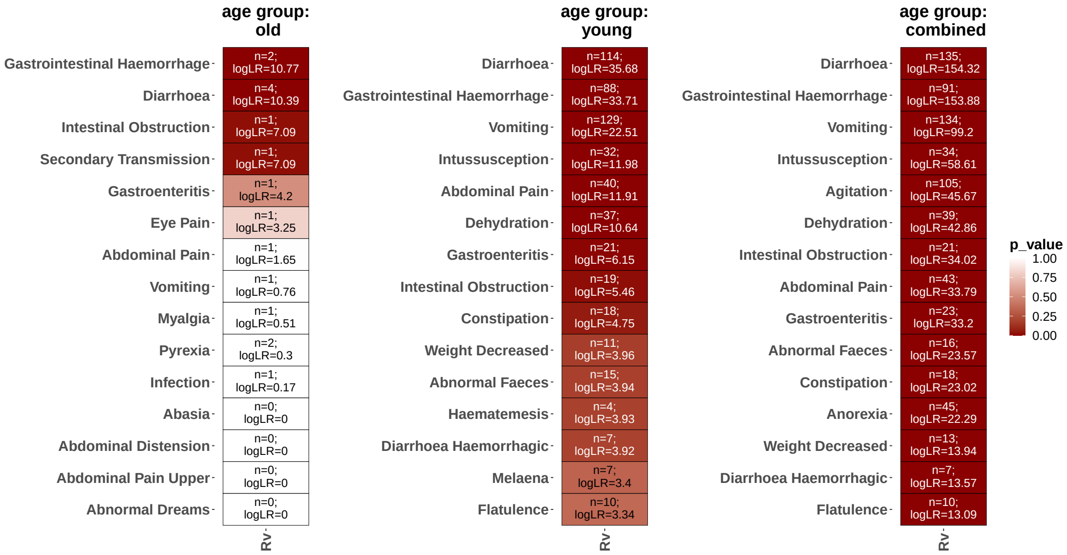
```
test_rv_optimal_list <- list(
  old = test_rv_poisson_list$old,
  young = test_rv_poisson_list$young,
  combined = test_rv_zip_list$combined
)
```

We may extract the statistically significant AE/vaccine pairs (p-value < 0.05) from these tests as follows.

```
signif_pairs_rv <- lapply(test_rv_zip_list, extract_significant_pairs)
```

For visualization, we consider heatmaps for top 15 AEs (in terms of LRT statistics) in the three datasets, and place them side by side. Because **pvLRT** uses **ggplot2** as its plotting engine, these individual heatmaps can be easily combined via external **ggplot2** post-processing packages such as **patchwork** (Pedersen 2020). Codes are as follows:

```
pl_heatmap_rv_list <- mapply(
  function(this_test, this_name) {
    heatmap_pvlrt(
      this_test,
      AE = 15,
      show_n = TRUE,
      show_lrstat = TRUE
    ) +
      ggtitle(paste("age group:\n", this_name)) +
      theme(plot.title = element_text(hjust = 0.5))
  },
  test_rv_optimal_list,
  names(test_rv_optimal_list),
  SIMPLIFY = FALSE
)
## install patchwork if not installed
# if (!requireNamespace("patchwork")) install.packages("pacthwork")
library(patchwork)
pl_heatmap_rv_comb <- pl_heatmap_rv_list$old +
  pl_heatmap_rv_list$young +
  pl_heatmap_rv_list$combined +
  # combine the legends
  plot_layout(guides = "collect") &
  # unified coloring scheme,
  # see the 'fill_gradient_range' argument for heatmap_pvlrt
  scale_fill_gradientn(
    colors = c("darkred", "white"),
    limits = c(0, 1)
  ) &
  theme(
    axis.text = element_text(size = 16, face = "bold"),
    legend.title = element_text(size = 16, face = "bold"),
    legend.text = element_text(size = 14),
    title = element_text(size = 16, face = "bold")
  )

pl_heatmap_rv_comb
```



The following observations are made from the above heatmaps and associated test results (stored in `signif_pairs_rv`). First, in terms of LR statistics and p-values, the "old" age group has only 4 statistically significant (p-value < 0.05) AEs, viz., Gastrointestinal Haemorrhage, Diarrhoea, Intestinal Obstruction, and Secondary Transmission. Note, however, the extremely small sample sizes for these AEs within the dataset; care must therefore be taken while interpreting these results. By contrast, the young and the combined age group data sets contain sufficient sample sizes. There are 8 and 21 significant AEs in the young and the combined age group datasets respectively. These prominent AEs for the combined and the young age groups agree, which is expected as the most reported cases in the combined dataset correspond to the young age group. Several of these prominent AEs detected in our analysis associate with Intussusception, which has been determined to be of particular concern in the literature (Niu, Erwin, and Braun 2001; Haber et al. 2004). However, unlike the approaches used in those studies, our LRT-based approach for signal detection does not require ad hoc thresholding, and provides rigorous statistical guarantees on the results obtained.

4 Discussion and Future Directions

Likelihood ratio test (LRT)-based approaches to pharmacovigilance constitute a class of rigorous statistical methods that permit objective signal determination without the need for specifying ad hoc thresholds. The lack of comprehensive software implementing the LRT methods (in R or otherwise) however has hindered their applicability in practice – especially in cases where the adverse event count data are zero-inflated.

The package pvLRT is our contribution to this area, providing a platform that implements a suite of formal statistical tools and post-processing functions for analyzing medical product safety datasets, and is readily accessible to practitioners. There are various directions in which the software can be extended in future releases. Some possible features/updates under consideration include the following.

1. Parallelize bootstrap resampling. This may permit substantial computational gains in sizable datasets. Care must however be taken while generating random draws in the parallel workers to ensure reproducibility and validity (see the discussion of page 5, Section 6 of the R package parallel (R Core Team 2022)).
2. Include additional processed datasets, and expand the current documentations with examples on these additional datasets.
3. Create visualizations for the estimated zero inflation probabilities along with the LRT results.

5 Acknowledgment

Dr. Markatou acknowledges grant support from KALEIDA Health Foundation (Troup Fund) and BAA HDDISWP#97, FDA. These resources supported the work of the University at Buffalo affiliated authors.

References

Agresti, Alan. 2013. *Categorical Data Analysis*. John Wiley & Sons.

Ahmed, I., and A. Poncet. 2016. *PhViD: an R package for PharmacoVigilance signal Detection*.

Bache, Stefan Milton, and Hadley Wickham. 2020. *Magrittr: A Forward-Pipe Operator for r*. https://CRAN.R-project.org/package=magrittr.

Benjamini, Yoav, and Yosef Hochberg. 1995. "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing." *Journal of the Royal Statistical Society: Series B (Methodological)*. https://doi.org/10.1111/j.2517-6161.1995.tb02031.x.

Canida, Travis, and John Ihrie. 2017. "openEBGM: An R Implementation of the Gamma-Poisson Shrinker Data Mining Model." *The R Journal* 9 (2): 499–519. https://journal.r-project.org/archive/2017/RJ-2017-063/index.html.

Chakraborty, Saptarshi, Anran Liu, Robert Ball, and Marianthi Markatou. 2022. "On the Use of the Likelihood Ratio Test Methodology in Pharmacovigilance." *Statistics in Medicine* 41 (27): 5395–5420.

Chakraborty, Saptarshi, and Marianthi Markatou. 2022. *pvLRT: Likelihood Ratio Test-Based Approaches to Pharmacovigilance*.

Chung, Gary. 2020. *mds: Medical Devices Surveillance*. https://CRAN.R-project.org/package=mds.

Davison, A. C., and D. V. Hinkley. 1997. *Bootstrap Methods and Their Application*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511802843.

Ding, Yuxin, Marianthi Markatou, and Robert Ball. 2020. "An Evaluation of Statistical Approaches to Postmarketing Surveillance." *Statistics in Medicine* 39 (7): 845–74. https://doi.org/10.1002/sim.8447.

Dowle, Matt, and Arun Srinivasan. 2021. *Data.table: Extension of 'Data.frame'*. https://CRAN.R-project.org/package=data.table.

Dumouchel, William. 1999. "Bayesian Data Mining in Large Frequency Tables, with an Application to the FDA Spontaneous Reporting System." *The American Statistician* 53 (3): 177–90. https://doi.org/10.1080/00031305.1999.10474456.

Haber, Penina, Robert T. Chen, Lynn R. Zanardi, Gina T. Mootrey, Roseanne English, Miles M. Braun, and the VAERS Working Group. 2004. "An Analysis of Rotavirus Vaccine Reports to the Vaccine Adverse Event Reporting System: More Than Intussusception Alone?" *Pediatrics* 113 (4): e353–59. https://doi.org/10.1542/peds.113.4.e353.

Huang, Lan, Jyoti Zalkikar, and Ram C. Tiwari. 2011. "A Likelihood Ratio Test Based Method for Signal Detection with Application to FDA's Drug Safety Data." *Journal of the American Statistical Association* 106 (496): 1230–41. https://doi.org/10.1198/jasa.2011.ap10243.

Huang, Lan, Dan Zheng, Jyoti Zalkikar, and Ram Tiwari. 2017. "Zero-Inflated Poisson Model Based Likelihood Ratio Test for Drug Safety Signal Detection." *Statistical Methods in Medical Research* 26 (1): 471–88. https://doi.org/10.1177/0962280214549590.

Li, Shuoran, Hongfan Chen, Lili Zhao, and Michael Kleinsasser. 2021. *AEenrich: Adverse Event Enrichment Tests*. https://CRAN.R-project.org/package=AEenrich.

Markatou, M., and R. Ball. 2014. "A Pattern Discovery Framework for Adverse Event Evaluation and Inference in Spontaneous Reporting Systems." *Stat. Anal. Data Min.* https://doi.org/10.1002/sam.11233.

Narasimhan, Balasubramanian, and Mei-Chiung Shih. 2012. *sglr: An r Package for Computing the Boundaries for Sequential Generalized Likelihood Ratio Test for Pre-Licensure Vaccine Studies*. http://CRAN.R-project.org/package=sglr.

Niu, Manette T., Diane E. Erwin, and M. Miles Braun. 2001. "Data Mining in the US Vaccine Adverse Event Reporting System (VAERS): Early Detection of Intussusception and Other Events After Rotavirus Vaccination." *Vaccine* 19 (32): 4627–34. https://doi.org/10.1016/S0264-410X(01)00237-7.

Pedersen, Thomas Lin. 2020. *Patchwork: The Composer of Plots*. https://CRAN.R-project.org/package=patchwork.

R Core Team. 2022. "Package 'Parallel'." https://stat.ethz.ch/R-manual/R-devel/library/parallel/doc/parallel.pdf.

Sakamoto, Yosiyuki, Makio Ishiguro, and Genshiro Kitagawa. 1986. "Akaike Information Criterion Statistics." *Dordrecht, The Netherlands: D. Reidel* 81 (10.5555): 26853.

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

Silva, Ivair Ramos, and Martin Kulldorff. 2021. *Sequential: Exact Sequential Analysis for Poisson and Binomial Data*. https://CRAN.R-project.org/package=Sequential.

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

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source*

Software 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Zhao, Yueqin, Min Yi, and Ram C Tiwari. 2018. "Extended Likelihood Ratio Test-Based Methods for Signal Detection in a Drug Class with Application to FDA's Adverse Event Reporting System Database." *Statistical Methods in Medical Research* 27 (3): 876–90. https://doi.org/10.1177/0962280216646678.

Saptarshi Chakraborty
Department of Biostatistics, SUNY University at Buffalo
School of Public Health and Health Professions
704 Kimball Tower, Buffalo, NY 14214, USA
ORCiD: 0000-0002-3121-9174
chakrab2@buffalo.edu

Marianthi Markatou
Department of Biostatistics, SUNY University at Buffalo
School of Public Health and Health Professions
726 Kimball Tower, Buffalo, NY 14214, USA
ORCiD: 0000-0002-1453-8229
markatou@buffalo.edu

Robert Ball
Center for Drug Evaluation and Research, U.S. Food and Drug Administration
Office of Surveillance and Epidemiology
Silver Spring, Maryland, USA
ORCiD: 0000-0002-1609-7420
Robert.Ball@fda.hhs.gov

GCPBayes: An R package for studying Cross-Phenotype Genetic Associations with Group-level Bayesian Meta-Analysis

by Taban Baghfalaki, Pierre-Emmanuel Sugier, Yazdan Asgari, Thérèse Truong, and Benoit Liquet

Abstract Several R packages have been developed to study cross-phenotypes associations (or pleiotropy) at the SNP-level, based on summary statistics data from genome-wide association studies (GWAS). However, none of them allow for consideration of the underlying group structure of the data. We developed an R package, entitled GCPBayes (Group level Bayesian Meta-Analysis for Studying Cross-Phenotype Genetic Associations), introduced by Baghfalaki et al. (2021), that implements continuous and Dirac spike priors for group selection, and also a Bayesian sparse group selection approach with hierarchical spike and slab priors, to select important variables at the group level and within the groups. The methods use summary statistics data from association studies or individual level data as inputs, and perform Bayesian meta-analysis approaches across multiple phenotypes to detect pleiotropy at both group-level (e.g., at the gene or pathway level) and within group (e.g., at the SNP level).

1 Introduction

Many cross-phenotype associations have been identified in genome-wide association studies (GWAS), providing evidence that pleiotropy (the idea that a single genetic variant, usually a Single-Nucleotide Polymorphism (SNP), is affecting multiple traits) is a common phenomenon in human complex traits (Watanabe et al. 2019). As a result, pleiotropy has attracted a great deal of attention among genetic epidemiologists (see, Solovieff et al. 2013; Verma et al. 2016; Hackinger and Zeggini 2017; Liu and Lin 2018; Majumdar et al. 2018; Krapohl et al. 2018; Lu et al. 2017; Trochet et al. 2019; Ray and Chatterjee 2020; Baghfalaki et al. 2021; Broc, Truong, and Liquet 2021), and several R packages have been developed to perform meta-analysis methods adapted for cross-phenotype association detection based on summary statistics data from GWAS as input (i.e., estimated effect size, standard error, and p-value of the association between each SNP and a trait, Ray and Chatterjee 2020). It should be noted that pleiotropy is only one possible explanation of cross-phenotype association, but for simplicity, we will use the term pleiotropy for cross-phenotype association to describe any cross-phenotype association in this paper. Below is a list of the most common methods for detecting pleiotropy between multiple traits.

ASSET (Bhattacharjee et al. 2012) is a frequentist method, which is an extension of standard fixed effects meta-analysis that considers the effects of a variable (such as a SNP) in each study (representing a phenotype) to be in either the same direction or opposite directions, allowing for detection of antagonistic pleiotropic effects. This approach uses summary statistics and does not take into account group structure. In the case of GWAS, the outputs of this approach are a p-value of the global association of each SNP with all studies (or phenotypes), and an optimal subset of non-null studies or phenotypes that are associated with each SNP.

GPA (Chung et al. 2014) implements a flexible statistical framework for the joint analysis of multiple GWAS and its integration with various genetic and genomic data. It implements a flexible parametric mixture modeling approach for such integrative analysis and also provides hypothesis testing procedures for pleiotropy and annotation enrichment. **PLACO** is a recently created R package using GWAS summary statistics data to identify pleiotropic signals between two traits (Ray and Chatterjee 2020). The computed method derives a composite null hypothesis that 'at most one trait is associated with the genetic variant' vs. the alternative that 'both traits are associated', based on the product of the Z-statistics of the genetic variants across two studies. CPBayes (Majumdar et al. 2018) is a Bayesian meta-analysis approach that uses univariate spike and slab prior and performs the MCMC technique via a Gibbs sampler. CPBayes uses the summary statistics data for a SNP across multiple traits. Two different measures are estimated for evaluating the overall pleiotropic association: the local false discovery rate and the Bayes factor. The optimal subset of associated traits underlying a pleiotropic signal is defined as the maximum a posteriori (MAP) estimate. The marginal trait-specific posterior probability of association (PPA), the direction of associations, and the credible interval of true genetic effects, are some examples of additional insights into pleiotropic signals that are provided by CPBayes.

All packages developed for these summary statistics-based methods are only designed to test pleiotropy at the variable-level (SNP-level). Though the structure of the common mechanisms shared by multiple phenotypes can be more complex, for example, different variants in the same locus can be associated with multiple traits and affect the same gene, and therefore can have an impact on the same trait. Thus, extending CPBayes to the gene or pathway level, which takes into account the group structure of the data, could provide additional power to detect pleiotropic signals between multiple diseases. By incorporating prior biological information in GWAS, such as group structure information (gene or pathway), our approaches could uncover new pleiotropic signals (Li et al. 2020; Baghfalaki et al. 2021).

In this paper, we present the GCPBayes R package (Bayesian meta-analysis of pleiotropic effects using group structure) for studying pleiotropy between multiple phenotypes using GWAS data by considering group structure information from prior biological knowledge. This package is able to consider and detect pleiotropy at both the variable-level and group-level. The inputs to the developed functions are SNP-level summary statistics data derived from GWAS by considering all the estimated regression coefficients of the variables (SNPs) in a group (a gene or a pathway) and its covariance matrix.

The methods that can be performed by the package offer a Bayesian paradigm using multivariate spike and slab priors for group-level pleiotropy using either continuous spike (CS, George and McCulloch 1993) or Dirac spike (DS, Mitchell and Beauchamp 1988) formulations. Also, a hierarchical sparsity prior (HS, Xu, Ghosh, et al. 2015) using two levels of Dirac spike and slab components to achieve group and within-group selection can be applied. Both tests for the global null hypothesis of no association and for detecting pleiotropy are included in the package. group-level and variable-level are considered. The free RStudio interface is advised to be used with GCPBayes. The GCPBayes package can be installed by typing:

```
install.packages("GCPBayes")
```

To be used, the GCPBayes package has to be loaded each time R or RStudio are opened, via the following code:

```
library(GCPBayes)
```

The paper is organized as follows: The next section ("Data structure") describes how a data should look in order to be used by functions of the GCPBayes package. The section "Usage" presents the different functions of GCPBayes. Some details for using different case studies and a practical guideline for the detection of pleiotropic effects are provided. The "Guidelines" section includes a statistical inference pipeline and recommendations for carrying out the various stages of a pleiotropy analysis with CRANpkg and Bayes. The computational time of the functions of the GCPBayes package is explored using real data in the "Computational time for GCPBayes" section. Finally, in the "Concluding remarks" section, we discuss some remarks about and limitations of GCPBayes. Besides, future works for a newer version of the package are discussed at the end of the section. In addition, the paper includes two appendices. The first is "Material and method", where notations and models are described briefly. The second one describes the design of the simulated data and their corresponding R codes, which are also embedded in the package.

2 Data structure

GCPBayes is intended to work with summary statistics data. Though summary statistics can be derived from individual-level data when available, this allows for taking into account correlations between variables to improve the accuracy of the method.

It should be noted that GCPBayes is not restricted to a specific type of outcome. Summary statistics data can be estimated through various kinds of models (linear, logistic, Cox, etc.) and based on different types of phenotypes (categorical or continuous traits). Therefore, it is possible to use the GCPBayes package for genetic studies, gene expression analyses, and other omics of studies.

In this paper, we will illustrate the use of our method in genetic studies of breast and thyroid cancers. We will explore the cross-phenotype association between these two traits and SNPs located in one gene in particular, *PARP2*. So, this section includes two different scenarios. The first part first part deals with summary statistics on the association between breast cancer risk or thyroid cancer risk and SNPs located in the *PARP2* gene (Example 1). The second part considers the individual level data from which the summary statistics of Example 1 were derived as input and then explores multicollinearity through the data (Example 2), and computes summary statistics and the covariance matrix from the individual-level data for *PARP2* gene (Example 3).

We will consider the estimated regression coefficients and their covariance matrices from all studies:

$$\hat{\beta}_k,\ \hat{\Sigma}_k = \mathrm{cov}(\hat{\beta}_k),\ \ k = 1, \cdots, K.$$

where $\hat{\beta}_k$ is an $m-$dimensional vector containing the regression coefficients from the k^{th} study for a given gene (or pathway) and $\hat{\Sigma}_k$ is the corresponding covariance matrix. If only summary statistics are available, the covariance matrix is replaced by a diagonal matrix with an estimated variance of the regression coefficients.

Summary statistics level data

GWAS summary statistics data are available through various public databases such as GWAS Catalog. In most cases, summary statistics include effect size estimate (beta), standard error, and p-value for each SNP. The **GCPBayes** package accepts inputs related to betas and standard errors as lists: specifically, a list of regression coefficients (betas) and a list of the matrices of the computed estimates of the variance of the regression coefficients.

Example 1: Inputs of GCPBayes using summary statistics from two case-control studies (on breast and thyroid cancers) for *PARP2* gene GWAS summary statistics are available through various public databases, such as the GWAS Catalog. In most cases, a summary of the statistical data includes the effect size estimate (beta), standard error, and p-value for each SNP:

```
library(GCPBayes)
data(PARP2_summary)
Breast <- PARP2_summary$Breast
Thyroid <- PARP2_summary$Thyroid
genename <- "PARP2"
snpnames <- rownames(Breast)
Betah <- list(Breast$beta,Thyroid$beta)
Sigmah <- list(diag(Breast$se^2),diag(Thyroid$se^2))
print(Betah,digits=2)

#> [[1]]
#> [1] -0.081  0.073 -0.346 -0.222  0.095  0.166
#>
#> [[2]]
#> [1] -0.033 -0.470  0.398  0.276  0.160  0.040

print(Sigmah,digits=2)

#> [[1]]
#>       [,1]  [,2]  [,3]  [,4]   [,5]  [,6]
#> [1,] 0.029 0.000 0.000 0.000 0.0000 0.000
#> [2,] 0.000 0.056 0.000 0.000 0.0000 0.000
#> [3,] 0.000 0.000 0.061 0.000 0.0000 0.000
#> [4,] 0.000 0.000 0.000 0.031 0.0000 0.000
#> [5,] 0.000 0.000 0.000 0.000 0.0064 0.000
#> [6,] 0.000 0.000 0.000 0.000 0.0000 0.051
#>
#> [[2]]
#>       [,1] [,2]  [,3] [,4]   [,5]  [,6]
#> [1,] 0.031 0.00 0.000 0.00 0.0000 0.000
#> [2,] 0.000 0.16 0.000 0.00 0.0000 0.000
#> [3,] 0.000 0.00 0.068 0.00 0.0000 0.000
#> [4,] 0.000 0.00 0.000 0.04 0.0000 0.000
#> [5,] 0.000 0.00 0.000 0.00 0.0063 0.000
#> [6,] 0.000 0.00 0.000 0.00 0.0000 0.043
```

This gene included six SNPs (rs3093872, rs3093921, rs1713411, rs3093926, rs3093930, and rs878156). So, the Betah list must include summary statistics for six elements for every study, and Sigmah list must contain a diagonal matrix of [6 × 6] for each study. A user can check the data using the "print" method.

Individual level data

When individual level data are available, summary statistics should be computed using external packages. Here, we explore the available approaches and R packages for computing the summary statistics. Let $Y_1, \cdots, Y_K$ be the response variables of K studies, and $Y_k = (Y_{k1}, \cdots, Y_{kn_k})'$ represent the n_k phenotype observations, $k = 1, cdots, Y_K$. For study k, let the genetic information be organized into G groups for research purposes. For example, this can be a set of SNPs belonging to a gene or a set of SNPs belonging to a set of genes acting together in the same biological pathway. We denote Z_{kg} as a $n_k \times m_g-$ matrix of the m_g covariates for group g, $g = 1, 2, \cdots, G$. As all inferences are performed gene by gene, we remove the index g and consider m as the number of variables in each specific group for simplicity. Considering one of the groups, we assume that a generalized linear model (GLM, McCullagh 2019) is fitted separately for each study as follows:

$$\eta\left(\mathrm{E}\left(Y_k\right)\right) = \alpha_k 1_{n_k} + Z_k \beta_k, \;\; k = 1, \cdots, K, \tag{1}$$

where $\eta(\cdot)$ is a link function, α_k is the intercept of the model and 1_{n_k} is a vector of n_k ones. Here $\beta_k = (\beta_{k1}, \cdots, \beta_{km})'$ denotes the m-dimensional regression coefficients for the group and the k^{th} study.

The "glm" function from R can be applied to fit generalized linear models and so get the summary statistics or parameter estimation of model 1 (Dunn and Smyth 2018). If the phenotype is binary, then 1 is the usual logistic or probit model (Agresti 2018). If the phenotype is continuous, model 1 is reduced to multiple linear regression and can also be fitted by the "lm" function from R which will provide the same results as the "glm" function. The "vcov" function can be applied to get the estimated covariance matrix of the regression coefficients, whereas the "glm" and "lm" functions calculate standard errors, which yield only the diagonal elements.

Another strategy should be considered in the case of multicollinearity. Multicollinearity is the presence of near-linear relationships among variables (Malo, Libiger, and Schork 2008). This phenomenon is widespread in genetic data, where non-random association of alleles at different loci in a given population is frequent, introducing large structures of correlation between SNPs. It is known as linkage disequilibrium (LD). LD can lead to inaccurate estimates of the regression coefficients, and also inflate the standard errors of the regression coefficients (Saleh, Arashi, and Kibria 2019). Multicollinearity can be visualized by drawing pairwise scatter plots of variables or by inspection of the correlation matrix. The variance inflation factor (VIF) can be used to assess the presence of multicollinearity (Fox and Weisberg 2018). It can be computed by the "vif" function from the **car** package (Fox et al. 2012). Most research papers consider a VIF > 10 as an indicator of multicollinearity (Menard 2002; Johnston, Jones, and Manley 2018; Gareth et al. 2013; Vittinghoff et al. 2011), but some choose a more conservative threshold of 5 (Gareth et al. 2013) or even 2.5 (Johnston, Jones, and Manley 2018).

Example 2: Exploring multicollinearity/linkage disequilibrium in *PARP2* gene using individual level data In this example, we show the existence of multicollinearity in the data. Here, we consider the individual-level data from which the summary statistics of Example 1 were derived. The summary statistics and the corresponding VIFs for both studies could be obtained using the usual "glm" function as follows:

```
library(GCPBayes)
library(car)

#> Loading required package: carData

data(PARP2)
Breast <- PARP2$Breast
Thyroid <- PARP2$Thyroid
Fit1 <- glm(y1~ ., family=binomial(link="logit"),data=Breast)
print(vif(Fit1))

#> rs3093872 rs3093921 rs1713411 rs3093926 rs3093930  rs878156
#>  1.696966  1.584775  3.095049  2.700594  1.347861  4.438011

Fit2 <- glm(y2~ ., family=binomial(link="logit"),data=Thyroid)
print(vif(Fit2))

#> rs3093872 rs3093921 rs1713411 rs3093926 rs3093930  rs878156
#>  1.565507  1.478632  3.552804  3.465883  1.397995  5.490866
```

A high VIF value (>5) for one SNP indicates multicollinearity in the data. Hence, the user should not use the “glm” function to obtain correct estimates of regression coefficients and their standard errors. Instead, the user could perform Ridge regression (Hilt and Seegrist 1977). Although many R packages are available to perform Ridge regression, only **lrmest**, **ridge** and **lmridge** can estimate the standard errors of the regression coefficients. Only **lmridge**, and only for the linear model, can compute their covariance matrix using the “vcov” or “vcov.lmridge” functions.

Since a usual approach for obtaining the covariance matrix is to apply a very time-consuming Bootstrap method (Efron 1992), we recommend performing a Bayesian hierarchical GLM by using Gaussian priors to get summary statistics of each group (gene/pathway). Thus, a user could apply the **bayesreg** package (Makalic and Schmidt 2016) or the **BhGLM** package (Yi et al. 2019) that provides fast and stable algorithms to estimate parameters for high-dimensional clinical and genomic data as well as highly correlated variables.

Example 3: Inputs of GCPBayes using individual level data from two case-control studies (on breast and thyroid cancers) for *PARP2* gene We propose to estimate the regression coefficients and their covariance matrices using a Bayesian hierarchical logistic model, which would lead to more powerful inputs for **GCPBayes** (as we mentioned earlier) than the inputs used in Example 1:

```
library(GCPBayes)
data(PARP2)
Breast <- PARP2$Breast
Thyroid <- PARP2$Thyroid
Fit1 <- bayesreg::bayesreg(as.factor(y1)~ ., model = "logistic", prior = "ridge",data=Breast)
Betah1 <-  Fit1$mu.beta
Sigmah1 <- cov(t(Fit1$beta))
Fit2 <- bayesreg::bayesreg(as.factor(y2)~ ., model = "logistic", prior = "ridge",data=Thyroid)
Betah2 <-  Fit2$mu.beta
Sigmah2 <- cov(t(Fit2$beta))
Betah <- list(Betah1,Betah2)
Sigmah <- list(Sigmah1,Sigmah2)
#print(Betah,digits=1)
#print(Sigmah,digits=1)
```

By running the code above, we can estimate the regression coefficients and their covariance matrices, using the individual-level data from the two studies. The results can be checked using the last two “print” commands.

As a final note in this section, we demonstrated in a previous study (Baghfalaki et al. 2021) that running GCPBayes on summary statistics data could result in a loss of power because it requires consideration of a diagonal covariance matrix of the effects (without information on off-diagonal components). So, statistically speaking, when individual-level data are available, we recommend computing the covariance matrix of the effects and using it as inputs to GCPBayes in order to increase the power of the method.

3 Usage

In this section, we describe how to use the **GCPBayes** package that includes four functions: DS, CS, HS, and MCMCplot. First, Examples 4-7 show the outputs of each of the four functions while applying GCPBayes to the summary statistics described in Example 1 to test whether the gene *PARP2* is associated with both thyroid cancer and breast cancer risk. Then, in Example 8, we compared results obtained with the DS function applied to summary statistics of *PARP2* gene with those obtained with individual-level data (as described in Examples 2-3) as input. Finally, in Example 9, we show an example of output for the gene *DNAJC1* which was significantly associated with breast and thyroid cancers using the DS function. The inputs are summary statistics from two larger studies than those used in Example 1 and will be detailed below. In addition to the real datasets described in Examples 1-3 and Example 9, three more simulated data are provided in Appendix B and are embedded in the **GCPBayes** package, including summary statistics level data for $K = 5$ studies with binary outcome, individual level data for $K = 3$ studies with continuous outcome and individual level data for $K = 2$ studies for survival outcomes and gene expression data. All commands used for the examples considered in this section are available through the R documentation of the **GCPBayes** package.

DS function

DS runs a Gibbs sampler for a multivariate Bayesian sparse group selection model with a Dirac spike prior for the detection of pleiotropic effects on the traits. As we mentioned in the previous section, the DS function is designed to use estimated regression coefficients and their estimated covariance matrices from K studies. The statistical details of the DS approach are given in Appendix A. Besides, more details can be found at (Baghfalaki et al. 2021). The DS function and its parameters have the following format:

```
DS(Betah, Sigmah, kappa0, sigma20, m, K, niter = 2000, burnin = 1000, nthin = 2,
    nchains = 2, a1 = 0.1, a2 = 0.1, d1 = 0.1, d2 = 0.1, snpnames, genename)
```

where

- Betah: A list containing m-dimensional vectors of the regression coefficients for K studies.
- Sigmah: A list containing the $m \times m$-dimensional positive definite covariance matrices which are the estimated covariance matrices of K studies. If individual-level data are not available, it corresponds to the diagonal matrices with the estimated variance of the regression coefficients for the K studies.
- kappa0: Initial value for kappa such that its dimension is equal to the number of Markov chains (nchains). Although, the best strategy for choosing the initial values are previous studies, however, the domain for choosing any value is equal to the domain of the priors which is given in the hierarchical setup A.2 in Appendix A. Kappa0 can be in range from 0 to 1.
- sigma20: Initial value for sigma2 such that its dimension is equal to nchains. The domain for initial values of sigma2 is $\Re^+$.
- m: Number of variables in the group.
- K: Number of traits.
- niter: Number of iterations for the Gibbs sampler.
- burnin: Number of burn-in iterations. The burnin=1000 value is the default.
- nthin: The lag of the iterations used for the posterior analysis (or thinning rate). nthin=2 is the default value.
- nchains: Number of Markov chains. When nchains > 1, the function calculates the Gelman-Rubin convergence statistic (Brooks and Gelman 1998; Gelman, Rubin, et al. 1992). Default value is nchains=2.
- a1, a2: Hyperparameters of kappa. Default values are a1=0.1 and a2=0.1.
- d1, d2: Hyperparameters of sigma2. d1 = 0.1 and d2 = 0.1 are the default values.
- snpnames: Variable names for the group.
- genename: The group's name.

Example 4: Usage of the DS function to analyse pleiotropy between breast and thyroid cancers at *PARP2* gene Here we consider the individual data for the *PARP2* gene from the two case-control studies on breast and thyroid cancer (Examples 2-3) as input. As we explained earlier, first we estimated the effect size of the six SNPs in *PARP2* and the corresponding covariance matrix for both studies. Then, we applied the DS function to the results.

```
library(GCPBayes)
data(PARP2)
Breast <- PARP2$Breast
Thyroid <- PARP2$Thyroid
genename <- "PARP2"
snpnames <- names(PARP2$Breast)[-1]
Fit1 <- bayesreg::bayesreg(as.factor(y1)~ ., model = "logistic", prior = "ridge",data=Breast)
Betah1 <-  Fit1$mu.beta
Sigmah1 <- cov(t(Fit1$beta))
Fit2 <- bayesreg::bayesreg(as.factor(y2)~ ., model = "logistic", prior = "ridge",data=Thyroid)
Betah2 <-  Fit2$mu.beta
Sigmah2 <- cov(t(Fit2$beta))
Betah <- list(Betah1,Betah2)
Sigmah <- list(Sigmah1,Sigmah2)
```

To detect pleiotropic effects for this gene, two chains with 2,000 iterations and a burn-in period of 1,000 iterations were used.

```
set.seed(123)
RES1 <- DS(Betah, Sigmah, kappa0=c(0.2,0.5), sigma20=c(1,2),
           m=6, K=2, niter=2000, burnin=1000, nthin=2, nchains=2,
           a1=0.1, a2=0.1, d1=0.1, d2=0.1, snpnames, genename)
```

The result of the DS function was saved in the “RES1” list, which includes four lists (MCMCChain, Summary, Criteria, and Indicator). The “MCMCChain” list contains posterior samples for the unknown parameters, while the “Summary” list includes summary statistics of the posterior samples for the unknown parameters including the name of the SNP, the posterior mean, the posterior standard deviation, the quantile 2.5%, median, the quantile 97.5% and the Gelman-Rubin convergence statistic (BGR, Brooks and Gelman 1998; Gelman, Rubin, et al. 1992).

Using a “print” command for the “Summary” list would result in two lists showing Bayesian estimation for all analyzed SNPs in both studies.

```
print(RES1$Summary,digits=2)
```

In order to check if the chains converged to a stationary distribution or not, a user could look at the BGR column. In our example, as the values of BGR are not close to one for some of the variables, the chains do not converge to a stationary distribution. To check this criteria, we also repeat the DS function with $niter = 20000$ and $burnin = 10000$.

```
set.seed(123)
RES1 <- DS(Betah, Sigmah, kappa0=c(0.2,0.5), sigma20=c(1,1.5),
           m=6, K=2, niter=20000, burnin=10000, nthin=2, nchains=2,
           a1=0.1, a2=0.1, d1=0.1, d2=0.1, snpnames, genename)
print(RES1$Summary,digits=2)
```

As it is clear, after running the DS function with new parameters, the values of the BGR column now confirm convergence to the stationary distribution. Note that in the case of $nchain = 1$, the BGR column can never be computed.

The two other output lists of the “RES1” (“Criteria” and “Indicator”) contain information for checking group and variable pleiotropy, respectively. For each SNP, the ‘Criteria’ list contains the name of the gene, name of the SNP, the posterior probability of association (PPA), the logarithm (to base 10) of the Bayes factor (BF), the local false discovery rate (locFDR) and θ of equation (4) of the Guidelines section. A user could check it with the following command:

```
print(RES1$Criteria,digits=2)

#> $`Name of Gene`
#> [1] "PARP2"
#>
#> $`Name of SNPs`
#> [1] "rs3093872" "rs3093921" "rs1713411" "rs3093926" "rs3093930" "rs878156"
#>
#> $PPA
#> $PPA[[1]]
#> [1] 0
#>
#> $PPA[[2]]
#> [1] 0.022
#>
#>
#> $log10BF
#> [1] -2.1
#>
#> $lBFDR
#> [1] 0.98
#>
#> $theta
#> [1] 1.8e-05
```

Based on the PPA1 and PPA2 values, the gene *PARP2* isnot significantly associated with the traits in the study. This is also confirmed by the values of lBFDR and BF. One can conclude that the global null hypothesis of no association at the group level cannot be rejected. Now, we can check the results of the pleiotropic effect at the group level by considering the value of theta. By considering the threshold of 0.5, it shows that there is no group-level pleiotropic effect for textit{PARP2} gene.

Also, an "Indicator" list is given for checking variable pleiotropy using two different statistics: (95% credible interval and the median threshold. Each row of the lists includes the name of the SNP, a binary indicator for the significant regression coefficient of each study (0=no,1=yes) and the total number of significant regression coefficients in the studies. Finally, the last column indicates whether the test for pleiotropy at the variable level is significant or not.

```
print(RES1$Indicator,digits=2)
```

In addition, *summaryDS* is a generic function that generates DS function result summaries:

```
summaryDS(RES1)

#> $`Name of Gene`
#> [1] "PARP2"
#>
#> $`Number of SNPs`
#> [1] 6
#>
#> $`Name of SNPs`
#> [1] "rs3093872" "rs3093921" "rs1713411" "rs3093926" "rs3093930" "rs878156"
#>
#> $log10BF
#> [1] -2.124921
#>
#> $lBFDR
#> [1] 0.9779942
#>
#> $theta
#> [1] 1.830575e-05
#>
#> $`Significance based on CI`
#>           Study 1 Study 2 Total Pleiotropic effect
#> rs3093872       0       0     0                 No
#> rs3093921       0       0     0                 No
#> rs1713411       0       0     0                 No
#> rs3093926       0       0     0                 No
#> rs3093930       0       0     0                 No
#> rs878156        0       0     0                 No
#>
#> $`Significance based on median thresholding`
#>           Study 1 Study 2 Total Pleiotropic effect
#> rs3093872       0       0     0                 No
#> rs3093921       0       0     0                 No
#> rs1713411       0       0     0                 No
#> rs3093926       0       0     0                 No
#> rs3093930       0       0     0                 No
#> rs878156        0       0     0                 No
```

As can be seen, there is no variable-level pleiotropic effect for this gene, which means that none of its SNPs were significant in both studies.

CS function

The CS runs a Gibbs sampler for a multivariate Bayesian sparse group selection model with a continuous spike prior for the detection of pleiotropic effects on K traits. CS uses the same inputs as the DS function. The details of the approach are given in the material and methods section (Appendix A). More information can be found at (Baghfalaki et al. 2021). The CS function has the following format in general:

```
CS(Betah, Sigmah, kappa0, tau20, zeta0, m, K, niter = 1000, burnin = 500,
nthin = 2, nchains = 2, a1 = a1, a2 = a2, c1 = c1, c2 = c2, sigma2 = 10^-3,
snpnames = snpnames, genename = genename)
```

where Betah, Sigmah, kappa0, m, K, niter, burnin, nthin, nchains, a1, a2, snpnames and genename are the same as those introduced for the DS function. For the other arguments we have

- zeta0: Initial value for zeta. For the elicitation of initial values of zeta, first the regression coefficients and their standard errors for each study are considered. After that, by adjusting their p-values, we can find the initial values of zeta, $k = 1, cdots, andK$. As a result, if one group contains at least one non-null signal, the group is a non-null and zeta0$[k] = 1$, otherwise zeta0$[k] = 0$.
- c1, c2: Hyperparameters of tau2. Default values are c1=0.1 and c2=0.1.
- sigma2: The variance of the spike (multivariate normal distribution with a diagonal covariance matrix with small variance) represents the null effect distribution. The default setting is $10 - 3$.

Example 5: Using the CS method to analyze pleiotropy between breast and thyroid cancers at *PARP2* gene For this example, we consider the same data used for Example 4 as input. So, first, we estimated the regression coefficients for each SNP and corresponding covariance matrix for both studies (the same estimated regression coefficients and covariance matrices were computed in Example 4), and then, we performed the CS function on the results. To get the values for $xi(0)_k$, a user must first calculate the p-values and initial values of xi_k for each regression coefficient using the code below (for more information, see (Baghfalaki et al. 2021):

```
K <- 2
m <- 6
pvalue <- matrix(0,K,m)
for(k in 1:K){
 pvalue[k,] <- 2*pnorm(-abs(Betah[[k]]/sqrt(diag(Sigmah[[k]]))))
}
zinit <- rep(0,K)
for(j in 1:K){
 index <- 1:m
 PVALUE <- p.adjust(pvalue[j,])
 SIGNALS <- index[PVALUE<0.05]
 modelf1 <- rep(0,m)
 modelf1[SIGNALS] <- 1
 if(max(modelf1)==1)(zinit[j] <- 1)
}
```

The initial value is a two-dimensional vector (as K=2) of 0 or 1:

```
print(zinit)
```

```
#> [1] 0 0
```

Now, we could use the CS function to look for a pleiotropic signal:

```
set.seed(123)
RES1 <- CS(Betah, Sigmah, kappa0=c(0.2,0.5), tau20=c(1,2), zeta0=zinit,
      m=m, K=K, niter=2000, burnin=1000, nthin=2, nchains=2,
      a1=0.1, a2=0.1, c1=0.1, c2=0.1, sigma2=10^-3, snpnames, genename)
```

The output of CS (here RES1) is the same as DS and has similar interpretations, *summaryCS* is a generic function used to produce result summaries of the CS function.

HS function

The HS function runs a Gibbs sampler for a multivariate Bayesian sparse group selection model with hierarchical spike prior for the detection of pleiotropic effects associated with the traits at group-level and variable-level. As in the case of CS and DS, this function is designed to use summary statistics as inputs, containing estimated regression coefficients and their estimated covariance matrices. The following is a general usage example of the HS function:

```
HS(Betah, Sigmah, kappa0 = kappa0, kappastar0 = kappastar0, sigma20 = sigma20, s20 = s20,
    m, K, niter = 1000, burnin = 500, nthin = 2, nchains = 2, a1 = 0.1, a2 = 0.1,
    d1 = 0.1, d2 = 0.1, c1 = 1, c2 = 1, e2 = 1, snpnames, genename)
```

where Betah, Sigmah, kappa0, sigma20, m, K, niter, burnin, nthin, nchains, a1, a2, d1, d2, snpnames, and genename have similar definitions as the DS function. For the other arguments we have

- kappastar0: Initial value for kappastar such that its dimension is equal to nchains. The domain for initial values of kappastar0 is $(0, 1)$.
- s20: Initial value for s2 such that its dimension is equal to nchains such that the domain for initial values of s2 is $\Re^+$.
- c1, c2: Hyperparameters of the kappastar. Defaults are c1=0.1 and c2=0.1.
- e2: Initial value for doing the Monte Carlo EM algorithm to estimate the hyperparameter of s2.

Example 6: Use of the HS method to analyze pleiotropy between breast and thyroid cancers at *PARP2* gene The same data as in Example 4 is used as input. We used the same estimated regression coefficients and their covariance matrices.

```
library(GCPBayes)
data(PARP2)
Breast <- PARP2$Breast
Thyroid <- PARP2$Thyroid
genename <- "PARP2"
snpnames <- names(PARP2$Breast)[-1]
Fit1 <- bayesreg::bayesreg(as.factor(y1)~ ., model = "logistic", prior = "ridge",data=Breast)
Betah1 <-  Fit1$mu.beta
Sigmah1 <- cov(t(Fit1$beta))
Fit2 <- bayesreg::bayesreg(as.factor(y2)~ ., model = "logistic", prior = "ridge",data=Thyroid)
Betah2 <-  Fit2$mu.beta
Sigmah2 <- cov(t(Fit2$beta))
Betah <- list(Betah1,Betah2)
Sigmah <- list(Sigmah1,Sigmah2)

set.seed(123)
RES <- HS(Betah, Sigmah, kappa0=c(0.5,0.3), kappastar0=c(0.5,0.3), sigma20=c(2,1), s20=c(1,2),
     m=6, K=2, niter=2000, burnin=1000, nthin=2, nchains=2,
     a1=0.1, a2=0.1, d1=0.1, d2=0.1, c1=1, c2=1, e2=1, snpnames, genename)
```

In this example, the HS function is applied to detect group and variable pleiotropic signals. The outputs of the HS function based on median threshold are as follows:

```
 summaryHS(RES1)
```

The overall output of HS (RES1) is the same as that of DS and has similar interpretations.

MCMCplot function

Monitoring the convergence of the MCMC statistic is important and the Gelman-Rubin statistic is not the only way to assess convergence. MCMCplot presents some visual plots, such as trace, auto-correlation function (ACF) and posterior density plots to check the convergence of the MCMC chains. The trace plot is the plot of the generated values over the course of the iterations. If all values fall within a uniform bandwidth around the posterior mean and do not display any periodicity or non-stationary trends, we can assume convergence. The density plot represents the signal's estimated posterior distribution. Also, monitoring ACF plots is very useful since low or high values indicate fast or slow convergence, respectively. In fact, we need to monitor the ACF of the MCMC generated sample and select a sampling lag larger than 1 [=L]. Then, we can produce an independent sample by keeping the first generated values in every batch of L iterations, which is called the thinning rate. In practice, the ACF should be close to zero for a sufficiently large number of lags.

The inputs of the MCMCplot function are the generated results by DS, CS, or HS function, the number of studies for drawing plots, the number of MCMC chains in the result, and the names of favorable SNPs for drawing the plots. Up to 10 plots can be plotted simultaneously. In general, a usage for the MCMCplot function is as follows:

MCMCplot(Result, k, nchains, whichsnps, betatype, acftype, dencol, denlty, denbg)

where

- Result: All of the results produced by the DS, CS, and HS functions.
- k: The number of plotting studies, where k = 1, 2,. . . , K.
- nchains: The number of Markov chains that were run to produce Result.
- which SNPs: The SNPs' names
- betatype: The type of plot desired. The following values are possible: "p" for points, "l" for lines, "b" for both points and lines, "c" for empty points joined by lines, "o" for overplotted points and lines, "s" and "S" for stair steps, and "h" for histogram-like vertical lines. Finally, "n" does not produce any points or lines.
- acftype: String giving the type of ACF to be computed. Allowed values are "correlation" (the default), "covariance" or "partial" will be partially matched.
- dencol: The color used to fill out the density plot.
- denlty: The line type to be used in the density plot.
- denbg The color to be used for the background of the density plot.

Example 7: Using the MCMCplot function to check convergence of the DS method For this example, the output of the DS function for *PARP2* gene is used when drawing the MCMCplot. We select this gene because it is a special case because it required more MCMC iterations to converge. To run this example, a user needs to run the first fourteen lines of code in Example 4 (briefly, these commands compute regression coefficients and covariance matrices based on the individual-level data). Then the DS function needs to be run as follows:

```
library(GCPBayes)
data(PARP2)
Breast <- PARP2$Breast
Thyroid <- PARP2$Thyroid
genename <- "PARP2"
snpnames <- names(PARP2$Breast)[-1]
Fit1 <- bayesreg::bayesreg(as.factor(y1)~ ., model = "logistic", prior = "ridge",data=Breast)
Betah1 <-  Fit1$mu.beta
Sigmah1 <- cov(t(Fit1$beta))
Fit2 <- bayesreg::bayesreg(as.factor(y2)~ ., model = "logistic", prior = "ridge",data=Thyroid)
Betah2 <-  Fit2$mu.beta
Sigmah2 <- cov(t(Fit2$beta))
Betah <- list(Betah1,Betah2)
Sigmah <- list(Sigmah1,Sigmah2)

set.seed(123)
RES1 <- DS(Betah, Sigmah, kappa0=c(0.2,0.5), sigma20=c(1,2),
           m=6, K=2, niter=200, burnin=100, nthin=1, nchains=2,
           a1=0.1, a2=0.1, d1=0.1, d2=0.1, snpnames, genename)
MCMCplot(Result = RES1, k = 2, nchains = 2, whichsnps = snpnames,
                     betatype = "l",
                     acftype = "correlation",
                     dencol = "white", denlty = 1, denbg = "white")

set.seed(123)
RES1 <- DS(Betah, Sigmah, kappa0=c(0.2,0.5), sigma20=c(1,2),
           m=6, K=2, niter=2000, burnin=1000, nthin=2, nchains=2,
           a1=0.1, a2=0.1, d1=0.1, d2=0.1, snpnames, genename)
MCMCplot(Result = RES1, k = 2, nchains = 2, whichsnps = snpnames,
                     betatype = "l",
                     acftype = "correlation",
                     dencol = "white", denlty = 1, denbg = "white")
```

The plots for the thyroid cancer study with 200 MCMC iterations and 100 burn-in are given in Figure 2. The top row of the figure shows strong periodicities in the posterior samples for some

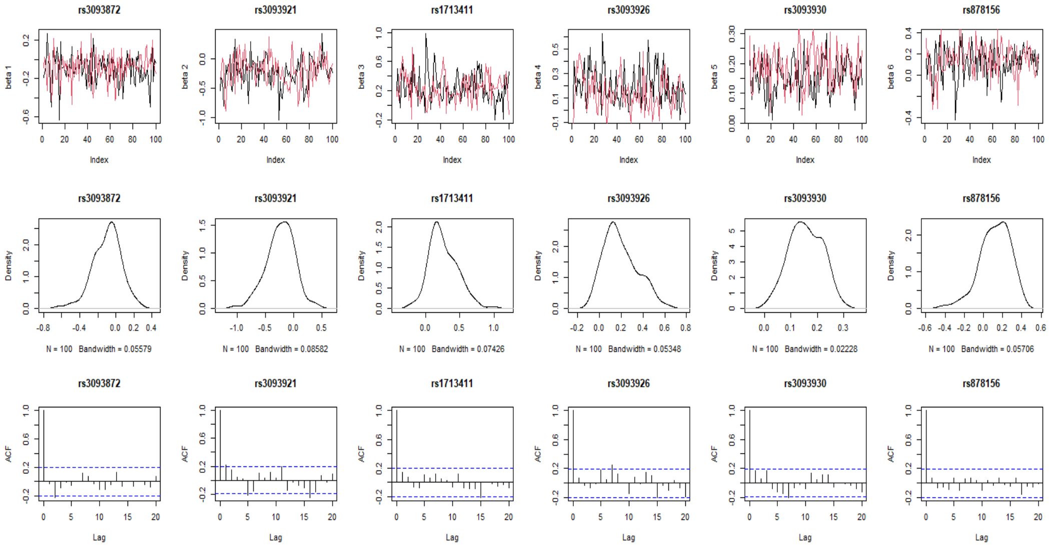


Figure 1: Trace, density and ACF plot for the posterior samples of the regression coefficients (niter=200, burnin=100, nthin=2) for thyroid study. Based on the top and last rows, the results do not show a convergence.

SNPs, which are confirmed by their corresponding ACF plots (last row). As a result, more samples are needed to achieve convergence for the *PARP2* gene. In the second DS command, the number of iterations was increased to 2000; half of this number was considered burn-in, and the thinning rate was set to 2. The results are shown in Figure 3 and the resulting regularities in the plot implies that increasing the number of iterations has made the model converge successfully.

Comparing GCPBayes results using summary statistics level versus individual level data as input

As we mentioned earlier, we showed that the performance of the GCPBayes package is better when using non-diagonal covariance matrices (individual-level data) compared to diagonal covariance matrices (summary level data, Baghfalaki et al. (2021)). In this section, we show two examples to deal with such differences.

Example 8: Comparing DS function outputs using summary statistics level versus individual-level data of *PARP2* gene as inputs First, we consider summary statistics level data for *PARP2* gene as an input of the GCPBayes package and run the DS function for this data as follows:

```
library(GCPBayes)
data(PARP2_summary)
Breast <- PARP2_summary$Breast
Thyroid <- PARP2_summary$Thyroid
genename <- "PARP2"
snpnames <- rownames(Breast)
Betah <- list(Breast$beta,Thyroid$beta)
Sigmah <- list(diag(Breast$se^2),diag(Thyroid$se^2))
print(Betah,digits=2)
print(Sigmah,digits=2)
set.seed(123)
RES_sum <- DS(Betah, Sigmah, kappa0=c(0.2,0.5), sigma20=c(1,2),
                            m=6, K=2, niter=2000, burnin=1000, nthin=2, nchains=2,
                            a1=0.1, a2=0.1, d1=0.1, d2=0.1, snpnames, genename)
print(RES_sum$Criteria,digits=2)
```

Now, we use individual data for the *PARP2* gene, then compute regression coefficients and covariance matrices based on the individual data and used it as the input to the GCPBayes package. Finally, we run the DS function for this data as follows:

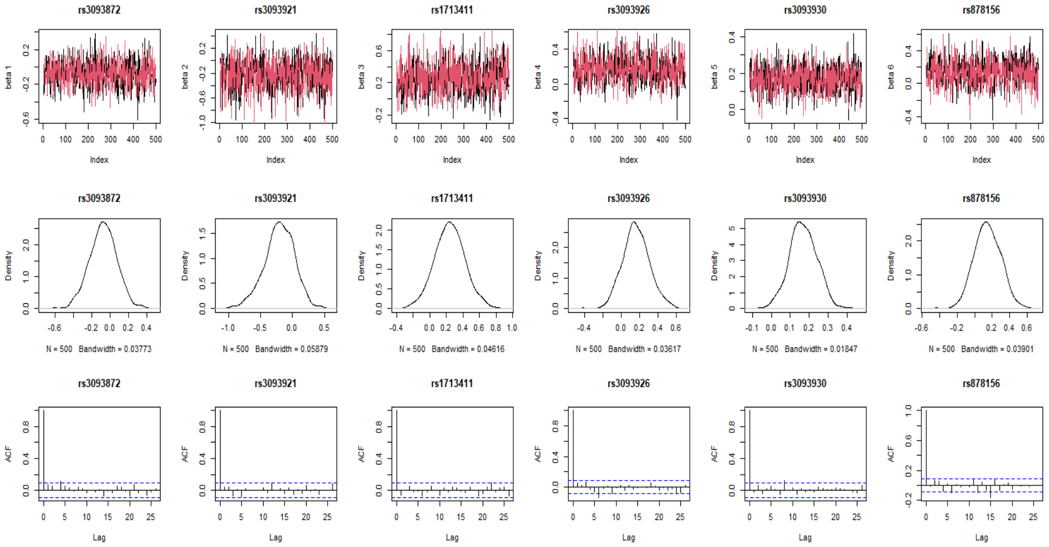


Figure 2: Trace, density and ACF plot for the posterior samples of the regression coefficients (niter=2000, burnin=1000, nthin=5) for thyroid study. Based on the top and last rows, the results imply a convergence.

```
library(GCPBayes)
data(PARP2)
Breast <- PARP2$Breast
Thyroid <- PARP2$Thyroid
genename <- "PARP2"
snpnames <- names(PARP2$Breast)[-1]
Fit1 <- bayesreg::bayesreg(as.factor(y1)~ ., model = "logistic", prior = "ridge",data=Breast)
Betah1 <-  Fit1$mu.beta
Sigmah1 <- cov(t(Fit1$beta))
Fit2 <- bayesreg::bayesreg(as.factor(y2)~ ., model = "logistic", prior = "ridge",data=Thyroid)
Betah2 <-  Fit2$mu.beta
Sigmah2 <- cov(t(Fit2$beta))
Betah <- list(Betah1,Betah2)
Sigmah <- list(Sigmah1,Sigmah2)
set.seed(123)
RES_ind <- DS(Betah, Sigmah, kappa0=c(0.2,0.5), sigma20=c(1,2),
                            m=6, K=2, niter=2000, burnin=1000, nthin=2, nchains=2,
                            a1=0.1, a2=0.1, d1=0.1, d2=0.1, snpnames, genename)
#print(RES_ind$Criteria,digits=2)
```

By comparing the results for these two scenarios, it is clear that using individual-level data, which includes more information about the SNP correlations, would lead to less biased results and hence better signals at the end.

Example 9: Investigation of pleiotropic effect between breast cancer and thyroid cancer at *DNAJC1* gene in large datasets For the final example, we consider the summary statistics for the *DNAJC1* protein-coding gene, specifically betas and standard errors, for fourteen SNPs from two studies (on breast and thyroid cancers), and try to investigate any potential pleiotropic signal at group and individual levels. The summary statistics of the breast and thyroid cancer studies were extracted from the Breast Cancer Association Consortium (BCAC) (Zhang et al. 2020) and the EPITHYR consortium (Truong et al. 2021), respectively. The data is embedded in the GCPBayes package, and a user can run the DS function as follows:

```
library(GCPBayes)
data(DNAJC1)
Breast <- DNAJC1$Breast
Thyroid <- DNAJC1$Thyroid
genename <- "DNAJC1"
```

```
snpnames <- Breast$snp
Betah <- list(Breast$beta,Thyroid$beta)
Sigmah <- list(diag(Breast$se^2),diag(Thyroid$se^2))
K <- 2
m <- 14
set.seed(123)
RES1 <- DS(Betah, Sigmah, kappa0=c(0.2,0.5), sigma20=c(1,2),
            m=m, K=K, niter=2000, burnin=1000, nthin=2, nchains=2,
            a1=0.1, a2=0.1, d1=0.1, d2=0.1, snpnames, genename)
```

Now, we run the following command for testing the global null hypothesis:

```
print(RES1$Criteria)

#> $`Name of Gene`
#> [1] "DNAJC1"
#>
#> $`Name of SNPs`
#>  [1] "rs10740997" "rs10764330" "rs10828266" "rs12570400" "rs1970467"
#>  [6] "rs2666762"  "rs2807967"  "rs35759613" "rs3951780"  "rs4747438"
#> [11] "rs60773921" "rs6650129"  "rs7075508"  "rs77353976"
#>
#> $PPA
#> $PPA[[1]]
#> [1] 1
#>
#> $PPA[[2]]
#> [1] 1
#>
#>
#> $log10BF
#> [1] 47.29012
#>
#> $lBFDR
#> [1] 1.709075e-48
#>
#> $theta
#> [1] 0.9998224
```

Based on the values of log10BF and lBFDR, the null hypothesis is rejected. Thus, we check the value of theta which is equal to 0.999, and conclude that the gene has a group pleiotropic effect.

A user should run the following command to test for variable pleiotropic effects:

```
print(RES1$Indicator$`Significant studies and Pleiotropic effect based on CI`)

#>            Study 1 Study 2 Total Pleiotropic effect
#> rs10740997       1       0     1                 No
#> rs10764330       1       0     1                 No
#> rs10828266       1       0     1                 No
#> rs12570400       1       0     1                 No
#> rs1970467        1       0     1                 No
#> rs2666762        1       1     2                Yes
#> rs2807967        1       0     1                 No
#> rs35759613       1       0     1                 No
#> rs3951780        1       0     1                 No
#> rs4747438        1       1     2                Yes
#> rs60773921       1       0     1                 No
#> rs6650129        1       0     1                 No
#> rs7075508        1       0     1                 No
#> rs77353976       0       0     0                 No
```

Based on the results, two SNPs are significantly associated with both traits (rs4747438, and rs2666762).

4 Guidelines

In this section, we provide some guidelines for working with the **GCPBayes** package and for extracting potential pleiotropic effects at group and variable-levels. We first give some practical advice on which methods to consider and in what order, and then we provide a decision pipeline based on statistical inferences, firstly for a primary analysis by using DS (or CS), and secondly for a precision analysis by using HS.

Practical hints

We do not recommend that a user do initial exploration by using the HS function because it may take a long time to compute when working with large datasets.

Besides, the HS function does not show better power in for detecting group pleiotropy compared to the other methods (DS and CS). In addition, DS and CS methods have similar strategies, but we showed that DS performed better than CS during the simulations for all tested scenarios (Baghfalaki et al. 2021). That is why we suggest the user first performs the the pleiotropic analysis on all groups in the data by using the DS function (Figure 3). Thus, in order to select groups with probable pleiotropic effects, we recommend performing a first batch of analyses by using the DS method, with a reasonable number of iterations such that most of the groups converge, using 2 chains to compute the BGR and test the convergence of the method. We suggest performing the method with 2,000 iterations and 1,000 burn-in. Then, if the BGR values for one or more SNPs in the group are significantly different from 1 (values between 0.9 and 1.1 are typically used as threshold for reasonable convergence), the method should be run again for the group using higher number for iterations and burn-in. We recommend to repeatedly double these values until the convergence threshold is reached.

Statistical inference pipeline for primary analysis

We provide a diagram in Figure 3 that helps the user detect group and variable pleiotropic effects using the **GCPBayes** package. The first step for detection of a pleiotropic signal is to test the global null hypothesis of no association against the global alternative hypothesis of association with at least one trait by considering DS (or CS).

Considering the CS method of the package (please refer to the corresponding equations in Appendix A), we define the following global null hypothesis of no association as follows:

$$\mathrm{H}_0 : \xi_1 = \xi_2 = \cdots = \xi_K = 0 \quad \text{versus } \mathrm{H}_1 : \text{at least one } \xi_k \neq 0, k = 1, \cdots, K. \tag{2}$$

To perform this test, the package computes two quantities. Let et $P(\mathrm{H}_0|\mathcal{D})$ be the posterior probability of the null hypothesis, i.e. the probability of making a false discovery for a non-null effect, which (Efron 2012) refers to as the $\mathcal{D}$ local Bayesian false discovery rate (denoted by $lBFDR$). The phrase "local" comes from a single point, $\{0\}$, as the domain of the null hypothesis tested (Efron 2012). Note that small values of $lBFDR$ show strong evidence for the existence of a substantive effect. We recommend using the usual value of 0.05 as a threshold to decide whether to reject the null hypothesis or not. The Bayes factor (BF), which describes the evidence of H_1 versus H_0 could also be considered. In contrast to the lBFDR, a high Bayes factor (BF) value indicates strong evidence in favor of H_1. We recommend considering a value of 1 as a threshold (or log10BF > 0).

The marginal study-specific posterior probability of association (PPA) proposed by Majumdar et al. (Majumdar et al. 2018) is also computed to quantify the relative contribution of a study underlying the signal. This is defined for each study k by $PPA_k = \frac{1}{M} \sum_{r=1}^{M} \xi_k^{(r)}$. This corresponds to the probability that the study k was drawn from the slab distribution, i.e. the probability that the group has a non-zero effect.

For checking any association by using DS, the global test is defined by:

$$\mathrm{H}_0 : \beta_1 = \cdots = \beta_K = 0, \text{versus } \ \mathrm{H}_1 : \text{at least one } \mathrm{fi}_\mathrm{k} \neq 0, \ \mathrm{k} = 1, \cdots, \mathrm{K}. \tag{3}$$

The existence of an association when using DS can be defined in the same way as for CS.

If there is no sufficient evidence for H_1 (i.e., a non-null effect in at least one study), then there is no need to further consider group pleiotropy or variable pleiotropy. If there is evidence for H_1, the user can calculate the value of the DS function's *theta*. The strategy to detect group pleiotropic effects based on CS is to compute θ as follows:

$$\theta = P\left(\sum_{k=1}^{K} \xi_k \geq 2|\hat{\beta}_k, \hat{\Sigma}_k,\ k = 1, \cdots, K\right). \tag{4}$$

Note that for computing θ for DS, we can follow the same strategy as for CS. Hence, θ represents the probability of having a non-zero effect in at least two studies. Thus, it is reasonable to consider a threshold t of 0.5 to statistically determine whether there is pleiotropy or not. However, a user wishing to generate many tentative hypotheses could choose a less stringent threshold (i.e., 0.1). Conversely, a user could choose a very strict threshold (i.e., 0.9) to select only the most certain groups. If $\theta \leq t$, no pleiotropic effect is detected at the group level, and thus no pleiotropy signal is present at the variable level. However, if $\theta > t$, we can consider the group to have a pleiotropic effect. It should be mentioned that when a pleiotropic effect is detected for a group, we can investigate the pleiotropy for each variable in the group. In this case, as CS and DS are defined group by group, the use of Bayesian point estimations (the means of the posterior distributions) and 95% credible intervals from the DS function can be used as criteria for variable selection. The j^{th} variable has a pleiotropic effect if at least two β_{kj}s are chosen. For example, assume that a pleiotropic effect is detected in an analysis of two studies. The posterior estimates $\hat{\beta}_{1j}$ and $\hat{\beta}_{2j}$ can then be used as a pleiotropy decision tool for the j^{th} variable $j = 1, 2, \cdots, m$.Hence, if β_{1j} and β_{2j} are two non-zero signals, Then the j^{th} variable is declared to have a pleiotropic effect; otherwise, the j^{th} variable has no pleiotropic effect.

Statistical inferences for analysis by using HS

Because HS has shown a higher power to detect pleiotropy at the variable-level in simulations, we recommend using the HS approach for the groups initially detected as pleiotropic via the DS function. A user can move down the θ threshold in order to explore potential pleiotropic effects that could have been detected with a larger sample size (i.e. to consider $\theta \geq 0.1$ from DS). Then, the user can compare the results at the variable-level for DS and HS (Figure 3). Using this overall strategy, a user would have to run the HS method on a smaller number of groups and thus avoids dealing with long computational time of the HS function for all groups at once.

To detect a pleiotropic signal for the j^{th} variable, the posterior estimates $\hat{\beta}_{kj}$, $j = 1, 2, \cdots, m$, $k = 1, 2, \cdots, K$, and $k = 1, 2, \cdots, K$ can be used. We consider the variable j to have pleiotropic effect if and only if there are at least two non-zero signals among the K studies. It should be noted that, in addition to a 95% CI, the sparseness of the posterior median of the regression coefficient β_{kj}, $k = 1, \cdots, K$, $j = 1, 2, \cdots, m_g$ can be used for this purpose.

Pleiotropy should be considered at the variable level based on the posterior median: for example, the j^{th} variable can be considered pleiotropic if the posterior median of a β_{kj} is different from 0 for at least two studies k. Finally, pleiotropy is considered at the group level if at least one pleiotropic variable is found within the group. Moreover, HS allows for the detection of pleiotropy at the group-level in other situations. Note that the pleiotropy effect can take on different shapes. As presented in (Solovieff et al. 2013), biological pleiotropy at the gene level can occur when two different signals within the same gene are associated with two different phenotypes. This can affect both phenotypes in a similar way. Therefore, we also consider pleiotropy at the group-level when an association is detected (not necessarily detected as pleiotropic) for different variables within the group in at least two different studies.

5 Computational time for GCPBayes

In this section, the computational time of **GCPBayes** package is explored by re-analysing of the experimental data for nine groups (genes) described in (Baghfalaki et al. 2021). The analysis has been conducted using a PC with a Core(TM) i5-4690K CPU @ 3.50 GHz processor and 8 GB of memory. For this purpose, we consider nine genes with different numbers of variables (SNPs). To fit the models, we used 2,000 MCMC iterations and discarded half as burn-in. Table 1 represents the results of computational times using the **GCPBayes** package for analyzing the data. As expected, the computational time increases when the number of variables within a group increases. The computational time for the DS method is the shortest, and the more complex HS model HS has the longest computational time, as expected. Another exploration of computational time for the **GCPBayes** package, based on simulation studies, see (Baghfalaki et al. 2021).

We have also applied GCPBayes on whole genome data of breast and ovarian cancers (summary statistics data). Running GCPBayes over all coding genes (number of genes: 18,244 and number of SNPs: 1 to 9,595) took about 17 days (on a PC with Intel Core(TM) i7-1165G7 @ 2.80 GHz, 32 GB RAM).

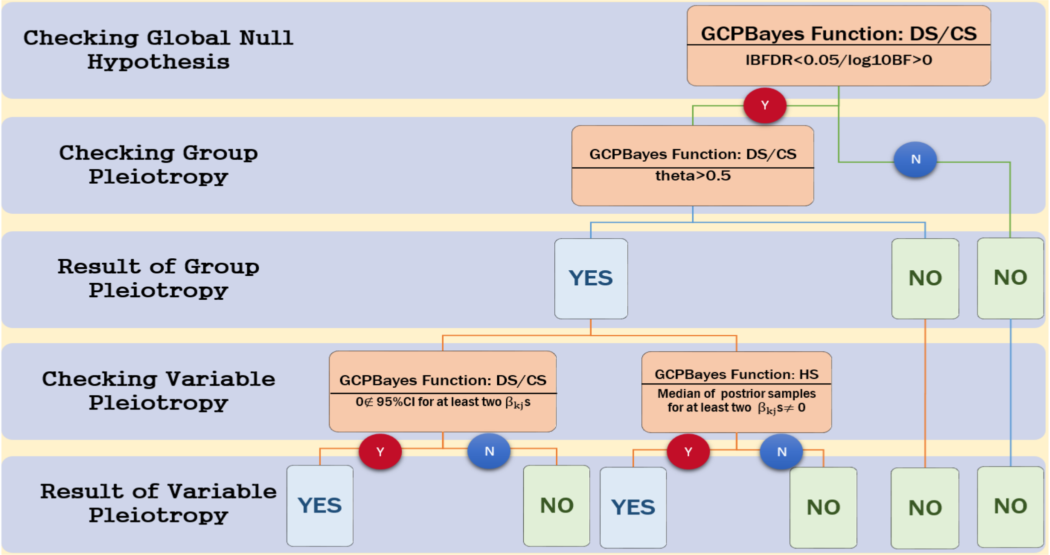

Checking Global Null Hypothesis
GCPBayes Function: DS/CS
lBFDR<0.05/log10BF>0
Y
N
Checking Group Pleiotropy
GCPBayes Function: DS/CS
theta>0.5
Result of Group Pleiotropy
YES
NO
NO
Checking Variable Pleiotropy
GCPBayes Function: DS/CS
0∉ 95%CI for at least two β_kj s
GCPBayes Function: HS
Median of postrior samples for at least two β_kj s≠ 0
Y
N
Y
N
Result of Variable Pleiotropy
YES
NO
YES
NO
NO
NO

Figure 3: A diagram for detecting group and variable pleiotropic effects using GCPBayes package. For more information see the text.

However, the user takes advantage of parallelization (using "foreach" command), the computational time can be reduced. For instance, the computational time was about 40 hours using the same PC but running six genes in each loop (i.e., using six CPUs instead of one).

Table 1: Computational time (in mins) of GCPBayes.

Name of gene	Number of SNPs	CS	DS	HS
ADH1A	2	0.070	0.030	0.139
ACE	5	0.069	0.036	0.172
ADH1B	10	0.084	0.049	0.243
RFC3	14	0.105	0.066	0.347
AOX1	24	0.158	0.127	0.629
ABCC8	30	0.220	0.190	0.966
BCAT1	50	0.555	0.539	3.123
EBF2	60	0.779	0.775	5.156
EGFR	103	2.697	2.882	30.794

6 Concluding remarks

We have developed the R package GCPBayes that implements several Bayesian meta-analysis methods for studying cross-phenotype genetic associations (pleiotropy), taking into account group structure from prior biological knowledge. GCPBayes offers multivariate spike and slab priors for group-level pleiotropy, using either DS or CS formulations, and for pleiotropy at both group and variable-level using the hierarchical spike prior (HS) formulation. These approaches take into account the heterogeneity in size and direction of the genetic effects across traits. Although DS and CS are not designed for variable selection, these methods do allow us to investigate the pleiotropy for each variable in the group by using 95% credible intervals and the median threshold as criteria for variable selection in groups for which a pleiotropic effect has been detected. On the other hand, the posterior estimates can be used to detect a pleiotropic signal for each variable using HS. Thus, a variable is considered to have a pleiotropic effect if and only if there are at least two non-zero signals among the studies. For this purpose, in addition to the 95% credible intervals, the sparseness of the posterior median of the regression coefficients is reported. Also, it should be mentioned that GCPBayes package provides marginal trait-specific posterior probability of association of each study (PPA), direction of associations for each variable, Bayes factor and local Bayesian false discovery rate of the global hypothesis, a statistical criterion for detection of group pleiotropic signal (θ), credible interval and median of the variables, summary information of the variables, and the generated MCMC samples. Besides, as we mentioned, HS allows us to consider the pleiotropy at the group-level if at least one pleiotropic variable is detected within the group. It also allows detection of pleiotropy at group-level without pleiotropy at variable-level if two or more different variables are significant in different studies

(Baghfalaki et al. 2021). This might happen if, for example, the same gene (group) may have a similar effect on several phenotypes through different genetic markers (Solovieff et al. 2013).

We provided nine examples to demonstrate the different ways of working with the package using both simulated and real data. In addition, three more examples are provided in Appendix B to show the ability of the **GCPBayes** package to deal with various types of input data. As we mentioned earlier, while the input of the package is summary statistics data, it is possible to use the package for investigating pleiotropy in various kinds of phenotypes. So, a user look for pleiotropic signals in different types of phenotypes, including categorical, continuous, mix categorical and continuous, survival data, without any limitation.

However, there are still some limitations that a user should consider while working with the package. For example, for more accurate detection of pleiotropic signals, a large enough sample size for the input data should be provided. This means having a larger sample size in the individual-level data (from which the summary statistics are calculated from) would lead to better accuracy of the method for detecting pleiotropic signals. Besides, the **GCPBayes** package is designed for uncorrelated studies (no overlapping samples between studies). So, an improvement of the package for correlated studies could be considered for the future. Also, GCPBayes can deal with correlated samples in uncorrelated studies as it uses summary statistics. Though the user should exercise caution, the summary statistics has been calculated using methods which take into account such correlations between samples. More user-friendly functions for preparing the data, for pre-selecting, and assigning key variables into groups before applying the **GCPBayes** methods, are currently in development. Furthermore, since the main focus of the package is on the detection of pleiotropic effects using GWAS data, we are currently developing guidelines to optimize the package for use in real-world large-scale data by taking into account factors such as optimal iterations of the MCMC step for convergence and the development of a general strategy for dealing with genes with a large number of SNPs or large sizes.

Acknowledgements

The authors acknowledge the calculus center MCIA (Mésocentre de Calcul Intensif Aquitain) for providing its facilities. The 'Ligue contre le Cancer' is acknowledged as well for its support for the "Cross Cancer Genomic Investigation of Pleiotropy project". The INSERM and Aviesan ITMO cancer are acknowledged as well for their support for "Advanced Machine Learning Algorithms for leveraging Pleiotropy effect project". This project is also part of the INSERM cross-cutting project GOLD.

References

Agresti, Alan. 2018. *An Introduction to Categorical Data Analysis*. John Wiley & Sons.

Baghfalaki, Taban, Pierre-Emmanuel Sugier, Therese Truong, Anthony N Pettitt, Kerrie Mengersen, and Benoit Liquet. 2021. "Bayesian Meta-Analysis Models for Cross Cancer Genomic Investigation of Pleiotropic Effects Using Group Structure." *Statistics in Medicine* 40 (6): 1498–518.

Bhattacharjee, Samsiddhi, Preetha Rajaraman, Kevin B Jacobs, William A Wheeler, Beatrice S Melin, Patricia Hartge, Meredith Yeager, Charles C Chung, Stephen J Chanock, and Nilanjan Chatterjee. 2012. "A Subset-Based Approach Improves Power and Interpretation for the Combined Analysis of Genetic Association Studies of Heterogeneous Traits." *The American Journal of Human Genetics* 90 (5): 821–35.

Broc, Camilo, Therese Truong, and Benoit Liquet. 2021. "Penalized Partial Least Squares for Pleiotropy." *BMC Bioinformatics* 22 (1): 1–31.

Brooks, Stephen P, and Andrew Gelman. 1998. "General Methods for Monitoring Convergence of Iterative Simulations." *Journal of Computational and Graphical Statistics* 7 (4): 434–55.

Chung, Dongjun, Can Yang, Cong Li, Joel Gelernter, and Hongyu Zhao. 2014. "GPA: A Statistical Approach to Prioritizing GWAS Results by Integrating Pleiotropy and Annotation." *PLoS Genet* 10 (11): e1004787.

Dunn, Peter K, and Gordon K Smyth. 2018. *Generalized Linear Models with Examples in r*. Springer.

Efron, Bradley. 1992. "Bootstrap Methods: Another Look at the Jackknife." In *Breakthroughs in Statistics*, 569–93. Springer.

———. 2012. *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction*. Vol. 1. Cambridge University Press.

Fox, John, and Sanford Weisberg. 2018. *An r Companion to Applied Regression*. Sage publications.

Fox, John, Sanford Weisberg, Daniel Adler, Douglas Bates, Gabriel Baud-Bovy, Steve Ellison, David Firth, et al. 2012. "Package 'Car'." *Vienna: R Foundation for Statistical Computing* 16.

Gareth, James, Witten Daniela, Hastie Trevor, and Tibshirani Robert. 2013. *An Introduction to Statistical Learning: With Applications in r*. Spinger.

Gelman, Andrew, Donald B Rubin, et al. 1992. "Inference from Iterative Simulation Using Multiple Sequences." *Statistical Science* 7 (4): 457–72.

George, Edward I, and Robert E McCulloch. 1993. "Variable Selection via Gibbs Sampling." *Journal of the American Statistical Association* 88 (423): 881–89.

Hackinger, Sophie, and Eleftheria Zeggini. 2017. "Statistical Methods to Detect Pleiotropy in Human Complex Traits." *Open Biology* 7 (11): 170125.

Hilt, Donald E, and Donald W Seegrist. 1977. *Ridge, a Computer Program for Calculating Ridge Regression Estimates*. Vol. 236. Department of Agriculture, Forest Service, Northeastern Forest Experiment ....

Johnston, Ron, Kelvyn Jones, and David Manley. 2018. "Confounding and Collinearity in Regression Analysis: A Cautionary Tale and an Alternative Procedure, Illustrated by Studies of British Voting Behaviour." *Quality & Quantity* 52 (4): 1957–76.

Krapohl, Eva, Hamel Patel, Stephen Newhouse, Charles J Curtis, Sophie von Stumm, Philip S Dale, Delilah Zabaneh, Gerome Breen, Paul F OReilly, and Robert Plomin. 2018. "Multi-Polygenic Score Approach to Trait Prediction." *Molecular Psychiatry* 23 (5): 1368–74.

Li, Xihao, Zilin Li, Hufeng Zhou, Sheila M Gaynor, Yaowu Liu, Han Chen, Ryan Sun, et al. 2020. "Dynamic Incorporation of Multiple in Silico Functional Annotations Empowers Rare Variant Association Analysis of Large Whole-Genome Sequencing Studies at Scale." *Nature Genetics* 52 (9): 969–83.

Liu, Zhonghua, and Xihong Lin. 2018. "Multiple Phenotype Association Tests Using Summary Statistics in Genome-Wide Association Studies." *Biometrics* 74 (1): 165–75.

Lu, Qiongshi, Boyang Li, Derek Ou, Margret Erlendsdottir, Ryan L Powles, Tony Jiang, Yiming Hu, et al. 2017. "A Powerful Approach to Estimating Annotation-Stratified Genetic Covariance via GWAS Summary Statistics." *The American Journal of Human Genetics* 101 (6): 939–64.

Majumdar, Arunabha, Tanushree Haldar, Sourabh Bhattacharya, and John S Witte. 2018. "An Efficient Bayesian Meta-Analysis Approach for Studying Cross-Phenotype Genetic Associations." *PLoS Genetics* 14 (2): e1007139.

Makalic, Enes, and Daniel F Schmidt. 2016. "High-Dimensional Bayesian Regularised Regression with the BayesReg Package." *arXiv Preprint arXiv:1611.06649*.

Malo, Nathalie, Ondrej Libiger, and Nicholas J Schork. 2008. "Accommodating Linkage Disequilibrium in Genetic-Association Analyses via Ridge Regression." *The American Journal of Human Genetics* 82 (2): 375–85.

McCullagh, Peter. 2019. *Generalized Linear Models*. Routledge.

Menard, Scott. 2002. *Applied Logistic Regression Analysis*. Vol. 106. Sage.

Mitchell, Toby J, and John J Beauchamp. 1988. "Bayesian Variable Selection in Linear Regression." *Journal of the American Statistical Association* 83 (404): 1023–32.

Ray, Debashree, and Nilanjan Chatterjee. 2020. "A Powerful Method for Pleiotropic Analysis Under Composite Null Hypothesis Identifies Novel Shared Loci Between Type 2 Diabetes and Prostate Cancer." *PLoS Genetics* 16 (12): e1009218.

Saleh, AK Md Ehsanes, Mohammad Arashi, and BM Golam Kibria. 2019. *Theory of Ridge Regression Estimation with Applications*. Vol. 285. John Wiley & Sons.

Solovieff, Nadia, Chris Cotsapas, Phil H Lee, Shaun M Purcell, and Jordan W Smoller. 2013. "Pleiotropy in Complex Traits: Challenges and Strategies." *Nature Reviews Genetics* 14 (7): 483–95.

Trochet, Holly, Matti Pirinen, Gavin Band, Luke Jostins, Gilean McVean, and Chris CA Spencer. 2019. "Bayesian Meta-Analysis Across Genome-Wide Association Studies of Diverse Phenotypes." *Genetic Epidemiology* 43 (5): 532–47.

Truong, Therese, Fabienne Lesueur, Pierre-Emmanuel Sugier, Julie Guibon, Constance Xhaard, Mojgan Karimi, Om Kulkarni, et al. 2021. "Multiethnic Genome-Wide Association Study of Differentiated Thyroid Cancer in the EPITHYR Consortium." *International Journal of Cancer* 148 (12): 2935–46.

Verma, Anurag, Shefali S Verma, Sarah A Pendergrass, Dana C Crawford, David R Crosslin, Helena Kuivaniemi, William S Bush, et al. 2016. "eMERGE Phenome-Wide Association Study (PheWAS) Identifies Clinical Associations and Pleiotropy for Stop-Gain Variants." *BMC Medical Genomics* 9 (1): 19–25.

Vittinghoff, Eric, David V Glidden, Stephen C Shiboski, and Charles E McCulloch. 2011. *Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models*. Springer Science & Business Media.

Watanabe, Kyoko, Sven Stringer, Oleksandr Frei, et al. 2019. "A Global Overview of Pleiotropy and Genetic Architecture in Complex Traits." *Nature Genetics* 51 (9): 1339–48.

Xu, Xiaofan, Malay Ghosh, et al. 2015. "Bayesian Variable Selection and Estimation for Group Lasso." *Bayesian Analysis* 10 (4): 909–36.

Yi, Nengjun, Zaixiang Tang, Xinyan Zhang, and Boyi Guo. 2019. "BhGLM: Bayesian Hierarchical

GLMs and Survival Models, with Applications to Genomics and Epidemiology." *Bioinformatics* 35 (8): 1419–21.

Zhang, Haoyu, Thomas U Ahearn, Julie Lecarpentier, Daniel Barnes, Jonathan Beesley, Guanghao Qi, Xia Jiang, et al. 2020. "Genome-Wide Association Study Identifies 32 Novel Breast Cancer Susceptibility Loci from Overall and Subtype-Specific Analyses." *Nature Genetics* 52 (6): 572–81.

Taban Baghfalaki
Tarbiat Modares University
Faculty of Mathematical scinces
Tehran, Iran.
Université Paris-Saclay; UVSQ; INSERM, U1018; Gustave Roussy; CESP, Team Exposome and Heredity
Villejuif, France
ORCiD: 0000-0002-2100-4532
t.baghfalaki@modares.ac.ir

Pierre-Emmanuel Sugier
Laboratoire de Mathématiques et de leurs Applications de Pau
Université de Pau et des Pays de l'Adour,
UMR CNRS 5142, E2S-UPPA, France.
Université Paris-Saclay; UVSQ; INSERM, U1018; Gustave Roussy; CESP, Team Exposome and Heredity
Villejuif, France
ORCiD: 0000-0002-5846-1104
pe.sugier@univ-pau.fr

Yazdan Asgari
Université Paris-Saclay; UVSQ; INSERM, U1018; Gustave Roussy; CESP, Team Exposome and Heredity
Villejuif, France
ORCiD: 0000-0001-6993-6956
yazdan.asgari@inserm.fr

Thérèse Truong
Université Paris-Saclay; UVSQ; INSERM, U1018; Gustave Roussy; CESP, Team Exposome and Heredity
Villejuif, France
ORCiD: 0000-0002-2943-6786
therese.truong@inserm.fr

Benoit Liquet
Macquarie University
School of Mathematics and Physical Sciences
NSW, Australia
Universite de Pau et des Pays de l'Adour
Laboratoire de Mathématiques et de leurs Applications de Pau
UMR CNRS 5142, E2S-UPPA, France
ORCiD: 0000-0002-8136-2294
benoit.liquet-weiland@mq.edu.au, benoit.liquet@univ-pau.fr

rankFD: An R Software Package for Nonparametric Analysis of General Factorial Designs

by Frank Konietschke, Markus Pauly, Arne C. Bathke, Sarah Friedrich and Edgar Brunner

Abstract Many experiments can be modeled by a factorial design which allows statistical analysis of main factors and their interactions. A plethora of parametric inference procedures have been developed, for instance based on normality and additivity of the effects. However, often, it is not reasonable to assume a parametric model, or even normality, and effects may not be expressed well in terms of location shifts. In these situations, the use of a fully nonparametric model may be advisable. Nevertheless, until very recently, the straightforward application of nonparametric methods in complex designs has been hampered by the lack of a comprehensive R package. This gap has now been closed by the novel R-package **rankFD** that implements current state of the art nonparametric ranking methods for the analysis of factorial designs. In this paper, we describe its use, along with detailed interpretations of the results.

1 Introduction

Nonparametric methods and in particular rank-based methods are commonly used for the analysis of experiments when it cannot be assumed that the observations derive from a normal population distribution. In online discussion fora regarding the application of statistical methods one can often find questions such as: "Does anybody know whether there is a nonparametric analog of ANOVA?". The common response is: "You may use rank methods" which usually prompts the next question: "Does anybody know a software package performing the computations for a nonparametric ANOVA / rank ANOVA?". The answers to this question vary: some list more or less popular statistical software packages, others give the heuristic advice of simply replacing the observations by their ranks and then performing regular ANOVA on the ranks. This suggests that there is a lack of clear advice on not just how to implement rank-based methods, but also how to interpret and understand the theoretical background. As such, the goal of the present article is to both explain when and how to use the procedures implemented in **rankFD**, and also provide the reader with enough of the theoretical background so that they can interpret the results correctly.

In order to provide a more precise answer regarding the nonparametric analog of ANOVA, one has to discuss the quantities by which a potential effect in a trial can be intuitively described. Such effects may be the differences or ratios of the means of the observations or of some other parameter or estimand defined in a semi-parametric model. To compare the differences of means in semi-parametric models where the normal distribution cannot be assumed, the so-called studentized permutation procedures (Janssen, 1997; Pauly et al., 2015; Smaga, 2015) are appropriate. These procedures provide quite accurate results even in case of small to moderate sample sizes, depending on the type of the data and the underlying population distribution. However, there are several situations where differences or linear combinations of means may not be appropriate to describe intuitive treatment effects – for example if the data have floor and ceiling effects or if the distributions have completely different shapes. In case of ordinal data, means are not even defined, and using a numerical encoding of the ordered categories as seemingly metric data may lead to incorrect conclusions (Kahler et al., 2008). In such cases, treatment effects can reasonably be described by the so-called *relative effect* which was introduced by Mann and Whitney (1947) and Putter (1955). For independent observations $X \sim F_1$ and $Y \sim F_2$, the relative effect is defined as $\theta = P(X < Y) + \frac{1}{2}P(X = Y)$, which can be equivalently written as $\theta = \int F_1 dF_2$. It may be noted that this effect has been known under many different names in the literature, for example Wilcoxon functional (Janssen, 1999a), Mann-Whitney type effect (Dobler et al., 2019), stochastic superiority (D'Agostino et al., 2006), or probabilistic index (Acion et al., 2006; Thas et al., 2012). We prefer the expression "relative effect" or "nonparametric relative treatment effect" with reference to Birnbaum and Klose (1957).

The relative effect θ can be estimated by replacing the distribution functions F_1 and F_2 with their empirical counterparts, $\widehat{F}_1$ and $\widehat{F}_2$, the so-called empirical distribution functions. This leads to the estimator $\widehat{\theta} = \int \widehat{F}_1 d\widehat{F}_2 = \frac{1}{n_1}\left(\overline{R}_{2\cdot} - \frac{n_2+1}{2}\right)$, where $\overline{R}_{2\cdot} = \frac{1}{n_2}\sum_{k=1}^{n_2} R_{2k}$ denotes the mean of the overall ranks R_{2k} of the observations $X_{21}, \ldots, X_{2n_2}$ among all $N = n_1 + n_2$ observations in the experiment. It is well-known that $\widehat{\theta}$ is an unbiased and L_2-consistent estimator of the relative effect θ and thus, the mean of the ranks provides the basis for estimating θ and for statistical inference regarding θ.

For two random variables X and Y, a relative effect $\theta > 1/2$ indicates a tendency that X takes smaller values than Y, while $\theta < 1/2$ means that X tends to have larger values than Y. No tendency in either direction corresponds to a relative effect of $\theta = \frac{1}{2}$. Crucially, the presence of a relative effect does not translate to a difference in means, and likewise, the absence of a relative effect does not suggest that the means are the same. In other words, if X has a mean μ_x and Y has a mean μ_y, then we may have $\theta \neq 1/2$ when $\mu_x = \mu_y$, or $\theta = \frac{1}{2}$ when $\mu_x \neq \mu_y$. Analogously, for the medians $\widetilde{\mu}_x$ and $\widetilde{\mu}_y$, it is possible that $\theta \neq \frac{1}{2}$ and $\widetilde{\mu}_x = \widetilde{\mu}_y$, or that $\theta = \frac{1}{2}$ and $\widetilde{\mu}_x \neq \widetilde{\mu}_y$. Thus, from a significant result of a rank test it cannot be concluded that $\mu_x \neq \mu_y$ or $\widetilde{\mu}_x \neq \widetilde{\mu}_y$. In this sense, rank tests based on $\widehat{\theta}$ (e.g., the Wilcoxon-Mann-Whitney test, the Fligner-Policello test, or the Brunner-Munzel test) are not tests of the equality of means or medians, and therefore not simply nonparametric analogs of the t-test since the hypotheses and consistency regions of these tests are not identical. Note that the consistency region contains all distribution functions for which the power of the test tends to 1 as sample sizes tend to ∞. In most parametric models, the set of distribution functions contained in the hypothesis and in the consistency region are complementary. In some nonparametric models, however this is in general not the case which may lead to difficulties interpreting "significant" results obtained by rank-based tests (Brunner et al., 2020). Some details will be explained in Section 2.2. Similar remarks apply to rank tests for multiple samples or even in factorial designs. This is ultimately the reason why the heuristic approach of replacing the observations by their ranks may lead to non-valid procedures in general (Conover and Iman, 1981). Especially in factorial designs, linear combinations of means may have different meanings than linear combinations of relative effects. With this in mind, users of the R-package for rank tests described in this paper should know that they might get different results than obtained by using a common ANOVA package.

The second question often read in discussion fora —'what software package should I use' —can be answered more easily. Most statistical software packages provide options for the classical nonparametric rank-based methods, however, these can still be quite limited and more contemporary and/or appropriate methods may not be available. For example, most statistical software packages offer the Wilcoxon-Mann-Whitney and the Kruskal-Wallis test for independent observations, as well as some particular procedures from the literature. However, more modern nonparametric rank-based methods developed during the last decades (Ruymgaart, 1980; Akritas and Arnold, 1994; Akritas et al., 1997; Brunner and Puri, 1996; Konietschke et al., 2012; Brunner et al., 2017, 2019) are not mplemented in most packages. Moreover, in software tools following a more classical paradigm, ties (i.e., two or more different observations with exactly the same value, as frequently is the case in ordinal or count data) are often considered in form of "corrections" that are added to the case of no ties, instead of considering the situation of no ties as a special case of a general model allowing for arbitrary ties (only the trivial case of one-point distributions should generally be excluded). Also, quick algorithms (Streitberg and Röhmel, 1986; Mehta et al., 1988) for the computation of exact p values for permutation-based procedures are rarely used, and general methods for purely nonparametric effects in factorial designs are not provided in standard implementations. However, exactly such procedures are often needed in applications. Researchers are then tempted to use heuristic procedures as described above, although the conclusions drawn from them might be misleading.

Finally, confidence intervals for purely nonparametric effects, such as the relative effect θ, are not provided in standard software, in spite of the fact that appropriate confidence intervals for the effect measures being used in the analysis have been required by the pertinent guidelines for decades. Instead, some software packages offer confidence intervals for location shift effects which in general may be neither compatible to the decisions of the rank tests nor justified regarding the types of alternatives or the scales of the measurements in the experiment. Recall that the relative effect is not a measure of mean or median differences, and therefore confidence intervals for mean or median shifts are not congruent with hypothesis tests based on the relative treatment effect, such as the Wilcoxon-Mann-Whitney and the Kruskal-Wallis tests, among others.

The R package **rankFD** intends to close these gaps. It includes the classical rank tests for continuous observations as special cases, allows for situations with arbitrary ties, and extends these procedures to factorial designs. The hypotheses tested in factorial designs are expressed as linear hypotheses in terms of the distribution functions as introduced in Akritas et al. (1997) or as linear combinations of the relative effects as discussed in Brunner et al. (2017). Ranking procedures for testing equalities of distribution functions in factorial longitudinal data (repeated measures) and multivariate data are implemented in R packages **nparLD** (Noguchi et al., 2012), **npmv** and **nparMD** (Burchett et al., 2017; Kiefel et al., 2022), respectively. Semiparametric methods for testing null hypotheses in general factorial designs in means are implemented in the R packages **GFD** (Friedrich et al., 2017) and **MANOVA.RM** (Friedrich et al., 2019b).

In any case, it must be clearly noted that rank methods, especially in factorial designs, answer different questions than those considered by the ANOVA in common factorial designs. The relations between linear combinations of the expectations of the observations and their respective counterparts expressed in terms of rank or pseudo-rank means depend on the underlying distribution functions. Questions investigated by parametric factorial designs are related to the expected values of the observations, while questions investigated by using rank- and pseudo-rank-based methods are related to relative effects. The latter compare the distributions in the different treatment groups to an average distribution. Thus, it should not be a surprise to obtain different answers if different questions are posed. This must be kept in mind when responding to the seemingly simple question: “Does anybody know whether there is a nonparametric analog of ANOVA?”.

The paper is organized as follows. Section 2.2 discusses the statistical models and explains the concepts and methodology underlying the inferential procedures provided by the package **rankFD** while the corresponding test statistics are described in Section 2.3. Section 2.4 lists and explains the different functions used in this package, as well as examples demonstrating the usage of these functions on real-life data. The paper closes with a discussion of the meaning and interpretation of these methods and their relations to some procedures implemented in other R packages

2 Statistical models, effects, and hypotheses

First we consider the simple experimental design involving only one factor A with a levels involving n_i independent observations in each level i. These are modeled as

$$X_{ik} \sim F_i,\ i = 1, \dots, a;\ k = 1, \dots, n_i. \tag{1}$$

Throughout, we assume that the observations X_{ik} are measured at least on an ordinal scale, whereas F_i denotes an arbitrary distribution (or its cdf), with the exception of one-point distributions. In total, there are $N = \sum_{i=1}^{a} n_i$ observations in the trial. This statistical model does not involve any explicit parameters or parametrization that could be used to describe appropriate treatment effects. To describe effects in such a general model, we therefore define weighted and unweighted *relative effects*

$$\theta_i = \int H_N dF_i = P(Y < X_{ik}) + \frac{1}{2}P(Y = X_{ik}), \quad i = 1, \dots, a, \tag{2}$$

$$\psi_i = \int G dF_i = P(Z < X_{ik}) + \frac{1}{2}P(Z = X_{ik}), \quad i = 1, \dots, a. \tag{3}$$

In this general definition of a relative treatment effect, each distribution function F_i is compared either to a weighted average $H_N = \frac{1}{N}\sum_{i=1}^{a} n_i F_i$ or an unweighted average $G = \frac{1}{a}\sum_{i=1}^{a} F_i$ of the distribution functions. This can be regarded as comparing each observation $X_{ik} \sim F_i$ with either an artificial independent observation $Y \sim H_N$ of the weighted mean distribution or $Z \sim G$ of the unweighted mean distribution. The former leads to the weighted relative effect θ_i, while the latter leads to the unweighted relative effect ψ_i. In case of equal sample sizes, both effects coincide.

The unweighted relative effects ψ_i can be interpreted as follows: If $\psi_i < \frac{1}{2}$, then the observations in group i tend to be smaller than those coming from the average distribution G. If $\psi_i = \psi_j$, then in relation to the average distribution G, the observations coming from distributions F_i and F_j have the exact same tendency towards smaller or larger observations. Thus, it is reasonable to consider the case of $\psi_i = \psi_j$ as *no (relative) treatment effect* between levels i and j. The relations and interpretations for the weighted effects θ_i and θ_j follow analogously. In the following, we collect all distribution functions and relative effects in the vectors $\mathbf{F} = (F_1, \dots, F_a)^\top$ and $\boldsymbol{\psi} = (\psi_1, \dots, \psi_a)^\top$ or $\boldsymbol{\theta} = (\theta_1, \dots, \theta_a)^\top$, respectively.

Estimators of the weighted relative effects θ_i defined in (2) can be obtained using the ranks R_{ik} of the observations X_{ik}. In fact, $\widehat{\theta}_i = \frac{1}{N}\left(\overline{R}_{i\cdot} - \frac{1}{2}\right)$ is an unbiased and consistent estimator of θ_i, where $\overline{R}_{i\cdot} = \frac{1}{n_i}\sum_{k=1}^{n_i} R_{ik}$, and R_{ik} denotes the rank of X_{ik} among all $N = \sum_{i=1}^{d} n_i$ observations. In case of ties, mid-ranks must be used. Formally, the mid-rank R_{ik} is obtained from the empirical weighted average distribution function $\widehat{H}_N(x) = \frac{1}{N}\sum_{i=1}^{d} n_i \widehat{F}_i(x)$ by $R_{ik} = \frac{1}{2} + N\widehat{H}(X_{ik})$.

In the same way, the unweighted relative effects ψ_i defined in (3) are estimated using the so-called pseudo-ranks $R_{ik}^{\psi} = \frac{1}{2} + N\widehat{G}(X_{ik})$, where $\widehat{G}(x)$ denotes the empirical unweighted average distribution function. An unbiased and consistent estimator $\widehat{\psi}_i$ of ψ_i is given by

$$\widehat{\psi}_i \quad = \quad \frac{1}{N}\left(\overline{R}_{i\cdot}^{\psi} - \frac{1}{2}\right), \tag{4}$$

where $\overline{R}_{i\cdot}^{\psi} = \frac{1}{n_i}\sum_{k=1}^{n_i} R_{ik}^{\psi}$. For details we refer to Brunner et al. (2019), Section 2.3.2 or to Happ et al. (2020), Section 2. Basically, the estimators $\widehat{\boldsymbol{\theta}} = (\widehat{\theta}_1, \dots, \widehat{\theta}_a)^\top$ and $\widehat{\boldsymbol{\psi}} = (\widehat{\psi}_1, \dots, \widehat{\psi}_a)^\top$ are vectors whose components are linear functions of the rank means $\overline{R}_{i\cdot}$ or the pseudo-rank means $\overline{R}_{i\cdot}^{\psi}$, respectively.

Thus, rank tests are related to the weighted relative effects θ_i in (2), while pseudo-rank tests are related to the unweighted relative effects ψ_i in (3).

Hypotheses formulated in terms of distribution functions

Classical rank-based methods for a one-way layout, (e.g., Kruskal-Wallis test, Kruskal (1952); Kruskal and Wallis (1952); or Hettmansperger-Norton test, Hettmansperger and Norton (1987)) can be used to test null hypotheses formulated in terms of the distribution functions, such as

$$H_0^F : \qquad F_1 = \ldots = F_a, \tag{5}$$

where obviously, equal distribution functions imply equal variances if H_0^F in (5) is true (if second moments exist).

Two- and higher way layouts are covered within model (1) by sub-indexing the index i, similar to the theory of linear models. For instance, a two-way design involving a factor A with a levels and a factor B with b levels, respectively, can be written as

$$X_{ijk} \quad \sim \quad F_{ij}, i = 1, \ldots, a; j = 1, \ldots, b; k = 1, \ldots, n_{ij}, \tag{6}$$

and the distribution functions and relative effects are then collected in the structured vectors $\mathbf{F} = (F_{11}, \ldots, F_{ab})^\top$ and $\boldsymbol{\psi} = (\psi_{11}, \ldots, \psi_{ab})^\top$ or $\boldsymbol{\theta} = (\theta_{11}, \ldots, \theta_{ab})^\top$, respectively.

Consequently, Akritas and Arnold (1994), Brunner and Puri (1996), and Akritas et al. (1997) suggested to formulate null hypotheses in two- and higher-way layouts in a similar way as in linear models, with the expected values being replaced by the corresponding distribution functions. In a two-way layout, for example, hypotheses of no (distribution-)main effects A or B and no (distribution-)interaction (AB) are written as

$$\begin{array}{llll}
H_0^F(A): & \overline{F}_{1\cdot} = \cdots = \overline{F}_{a\cdot}, & \overline{F}_{i\cdot} = \frac{1}{b}\sum_{j=1}^{b} F_{ij}, & i = 1, \ldots, a, \\
H_0^F(B): & \overline{F}_{\cdot 1} = \cdots = \overline{F}_{\cdot b}, & \overline{F}_{\cdot j} = \frac{1}{a}\sum_{i=1}^{a} F_{ij}, & j = 1, \ldots, b, \\
H_0^F(AB): & F_{ij} = \overline{F}_{i\cdot} + \overline{F}_{\cdot j} - \overline{F}_{\cdot\cdot}, & \overline{F}_{\cdot\cdot} = \frac{1}{ab}\sum_{r=1}^{a}\sum_{s=1}^{b} F_{rs}, & i = 1, \ldots, a; j = 1, \ldots, b\,.
\end{array} \tag{7}$$

In order to extend the hypotheses in (5) or (7) to higher-way layouts, general hypotheses are written using matrix notation as

$$H_0^F(\mathbf{C}) : \mathbf{CF} \quad = \quad \mathbf{0}, \tag{8}$$

where $\mathbf{C}$ denotes an appropriate hypothesis matrix, in the same way as in linear models, only replacing means with the respective distribution functions. Note that $\mathbf{0}$ is here understood to be a vector of functions which are identically 0. Testing these hypotheses H_0^F of no distribution effects can be performed using the argument `hypothesis="H0F"` in the `rankFD` function. More details are provided in Section 2.4.

Hypotheses formulated in terms of relative effects

In general, researchers may not be interested in detecting the somewhat abstract alternative $H_1^F : \mathbf{CF} \neq \mathbf{0}$ that H_0^F in (8) is not true, but instead they want to detect whether a tendency to smaller or larger values exists between treatment levels. In a one-way layout, for example, the latter corresponds to the testing problem

$$H_0^P : \psi_1 = \ldots = \psi_a, \tag{9}$$

formulated in terms of the relative effects ψ_i. Here, the symbol H_0^P refers to the probabilities ψ_i in (3).

Remark: Of course, one can also state the hypothesis

$$H_0^P : \theta_1 = \ldots = \theta_a, \tag{10}$$

but it must be kept in mind that the hypothesis (10) depends on the relative sample sizes n_i/N in groups $i = 1, \ldots, a$. Thus, the rejection region of such a test is not invariant, but it changes with the ratios n_i/N of the sample sizes. In extreme cases, this might lead to surprising results when compared to the results obtained in designs with equal sample sizes. For details we refer to Brunner et al. (2020) and Brunner et al. (2019). The unweighted mean distribution is, however, one reference distribution of choice that helps in reducing the issues obtained with the weighted version. Whether the unweighted version is the "best" one, can not be answered and guaranteed, in general (Zimmermann et al., 2022).

In a two-way layout, for example, the hypotheses of no main effects or no interactions in terms of the relative effects $\psi_{ij} = \int G dF_{ij}$ are written as

$$\begin{aligned} H_0^P(A): &\quad \overline{\psi}_{1\cdot} = \cdots = \overline{\psi}_{a\cdot}, &\quad i = 1,\ldots,a, \\ H_0^P(B): &\quad \overline{\psi}_{\cdot 1} = \cdots = \overline{\psi}_{\cdot b}, &\quad j = 1,\ldots,b, \\ H_0^P(AB): &\quad \psi_{ij} = \overline{\psi}_{i\cdot} + \overline{\psi}_{\cdot j} - \overline{\psi}_{\cdot\cdot}, &\quad i = 1,\ldots,a;\ j = 1,\ldots,b, \end{aligned} \tag{11}$$

where $\overline{\psi}_{i\cdot} = \frac{1}{b}\sum_{j=1}^{b}\psi_{ij}$, $\overline{\psi}_{\cdot j} = \frac{1}{a}\sum_{i=1}^{a}\psi_{ij}$, and $\overline{\psi}_{\cdot\cdot} = \frac{1}{2}$. The matrix notation of these hypotheses is, analogously to (7) and (8),

$$H_0^P(\mathbf{C}): \mathbf{C}\boldsymbol{\psi} \quad = \quad \mathbf{0}, \tag{12}$$

where $\boldsymbol{\psi}$ denotes the vector of unweighted relative effects. For a detailed explanation of using matrix notation in factorial designs we refer to, e.g., Brunner et al. (2017) or Brunner et al. (2019), Sect. 5.2 and Sect. 8.7.1.

In a similar way as in the one-way layout, the hypotheses involving the weighted relative effects θ_{ij} in the two-way layout can be stated by replacing ψ_{ij}, $\overline{\psi}_{i\cdot}$, and $\overline{\psi}_{\cdot j}$ in (11) with θ_{ij}, $\overline{\theta}_{i\cdot}$, and $\overline{\theta}_{\cdot j}$, respectively. It may be noted, however, that – unlike in the one-way layout – in two- or higher-way layouts surprising results may already be obtained in case of moderate unequal samples sizes in simple shift-effect models. These basic models cannot be considered "extreme cases". This means that unequal sample sizes in two- or higher-way layouts constitute a serious challenge for rank tests while this is not the case for pseudo-rank tests. For more details we refer to Brunner et al. (2019), Chapter 5 and Brunner et al. (2020), Section 4.

Note that H_0^P in (12) neither implies variance homogeneity nor equal shapes of the distributions. In the case of two samples, this situation is also known as the *nonparametric Behrens-Fisher* problem (Fligner and Policello, 1981; Brunner and Munzel, 2000; Konietschke et al., 2012). In general, it is easier to estimate the covariance matrix of the empirical relative effects under the stronger null hypothesis H_0^F than under H_0^P. Therefore, statements about the sampling distribution of test statistics based on ranks have traditionally been formulated under H_0^F, even though it is well-known that those test statistics can only detect alternatives of the form $H_1^P : \mathbf{C}\boldsymbol{\theta} \neq \mathbf{0}$ or $H_1^P : \mathbf{C}\boldsymbol{\psi} \neq \mathbf{0}$.

Remark: The **rankFD** package implements the current state-of-the-art methods for testing H_0^P (using ranks as well as pseudo-ranks) in general factorial designs (Konietschke et al., 2012; Brunner et al., 2017), and it allows for the computation of a wide range of nonparametric test statistics. It explicitly also includes the classical tests based on weighted relative effects θ_i (using ranks) and on unweighted relative effects ψ_i (using pseudo-ranks). Both types of ranking procedures are included in **rankFD**. A reason for including the former tests is that it allows users to reproduce findings that have been obtained by other researchers using rank tests. Also, it offers the possibility to directly compare procedures which may facilitate a transparent discussion in that regard.

Multiple comparisons

So far, both null hypotheses H_0^F and H_0^P have been written as global null hypotheses. If they get rejected, one may only conclude that *some* factor level differs from the others (at corresponding significance level α). However, it still remains unknown specifically *which one* differs. Therefore, testing global null hypotheses often does not answer the particular research question of interest to scientists applying statistical methods, namely the specific localization of those treatment groups that are "driving" the significant results. In order to accomplish this goal, testing linear contrasts using a $q \times a$ contrast matrix

$$\mathbf{C} = \begin{pmatrix} \mathbf{c}_1^\top \\ \vdots \\ \mathbf{c}_q^\top \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1a} \\ c_{21} & c_{22} & \cdots & c_{2a} \\ \vdots & \vdots & \vdots & \vdots \\ c_{q1} & c_{q2} & \cdots & c_{qa} \end{pmatrix};\ \sum_{i=1}^{a} c_{\ell i} = 0,\ \ell = 1,\ldots,q,$$

in terms of multiple null hypotheses $H_0^{(\ell)} : \mathbf{c}_\ell^\top \boldsymbol{\psi} = 0$ (or $H_0^{(\ell)} : \mathbf{c}_\ell^\top \boldsymbol{F} = 0$) is the key. Here, each row vector $\mathbf{c}_\ell^\top$ describes one of q different contrasts reflecting the researcher's particular question. For instance, in a one-way layout with $a = 4$ levels, many-to-one (Dunnett-type) (Dunnett, 1955) or all pairwise (Tukey-type) comparisons are performed with the contrast matrices

$$\mathbf{C} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix} \qquad \text{or} \qquad \mathbf{C} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

Note: left shows many-to-one (Dunnett-type); right shows all-pairwise (Tukey-type) contrast matrix

respectively. Which contrast to use depends on the respective research question of interest. Bretz et al. (2001) provide a broad overview of different contrast matrices, which are numerically available within the `contrMat` function of the **multcomp** package in R (Hothorn et al., 2008). In general factorial designs involving more than one factor, multiple comparisons in terms of means of the levels of the main effects are a meaningful and valuable asset of a fundamental data analysis. For instance, in a 2 × 4 two-way design, many-to-one comparisons to the control group ($j = 1$ of factor B) are expressed as

$$\begin{aligned} H_0^{P(1)} : \overline{\psi}_{\cdot 1} &= \overline{\psi}_{\cdot 2} \\ H_0^{P(2)} : \overline{\psi}_{\cdot 1} &= \overline{\psi}_{\cdot 3} \\ H_0^{P(3)} : \overline{\psi}_{\cdot 1} &= \overline{\psi}_{\cdot 4} \end{aligned} \qquad \mathbf{C} = \left(\begin{array}{cccccccc} \psi_{11} & \psi_{12} & \psi_{13} & \psi_{14} & \psi_{21} & \psi_{22} & \psi_{23} & \psi_{24} \\ 1/2 & -1/2 & 0 & 0 & 1/2 & -1/2 & 0 & 0 \\ 1/2 & 0 & -1/2 & 0 & 1/2 & 0 & -1/2 & 0 \\ 1/2 & 0 & 0 & -1/2 & 1/2 & 0 & 0 & -1/2 \end{array}\right).$$

The `rankFD` function implements a broad list of pre-defined contrasts as well as flexible options allowing for user-defined contrast matrices for making multiple comparisons of the levels of the main or interaction effects. We provide computational details in Section 2.4.

Confidence intervals

To comply with the basic principle "*no test without a confidence interval*", the **rankFD** package also provides confidence intervals for the nonparametric quantities upon which the test is based. Two-sided $(1-\alpha)$-confidence intervals for ψ_i and $\theta = \psi_2 - \psi_1$ are obtained from the asymptotic distribution of the estimators $\widehat{\psi}_i$ in (4) by

$$CI = \left[\widehat{\psi}_i \mp z_{1-\alpha/2} \frac{\widehat{s}_i}{\sqrt{N}}\right], \tag{13}$$

where $z_{1-\alpha/2}$ denotes the $(1-\alpha/2)$ quantile of the standard normal distribution. Here, the variance estimator $\widehat{s}_i^2$ is a quite involved linear combination of different quadratic forms obtained from different rankings of the observations X_{ik}. For details we refer to Brunner et al. (2019), Sect. 4.6.1.

The confidence intervals in (13) may suffer from poor coverage probability if ψ_i is close to the limits 0 or 1 and, moreover, the limits of the confidence interval may exceed the boundaries 0 or 1. In this case, so-called *range preserving* intervals can be obtained by using the *logit*-transformation. The limits thus obtained are then "back-transformed" using the *expit*-transformation. For details we refer to Brunner et al. (2019), Sect. 4.6.2.

In **rankFD**, these confidence intervals are computed by the function `rankFD()` using the options CI.method = "normal" for the limits in (13) or CI.method = "logit" for the range preserving confidence intervals obtained by the *logit*-transformation. By default, `rankFD()` provides confidence intervals for both, ψ_i and θ_i. Regarding the confidence intervals for θ_i the same remarks as in Sect. 2.2.2 apply. Furthermore, since the Wilcoxon-Mann-Whitney test (and relative methods) use variance estimators that are only consistent under the respective null hypothesis H_0^F formulated in terms of the distribution functions, the tests cannot be inverted into confidence intervals for ψ_i.

3 Test statistics

The **rankFD** package implements a broad class of different test statistics for testing the general null hypotheses $H_0^F : \mathbf{CF} = \mathbf{0}$, $H_0^P : \mathbf{C}\boldsymbol{\psi} = \mathbf{0}$, and $H_0^P : \mathbf{C}\boldsymbol{\theta} = \mathbf{0}$, respectively. They include global test procedures (quadratic forms) and multiple contrast tests (linear statistics) for the analysis of data from general factorial designs, as well as methods specifically designed for the evaluation of two independent samples including the classical rank tests.

In the following, we will briefly explain these procedures. They are all based on the (asymptotic) distribution of standardized vectors of point estimators $\widehat{\boldsymbol{\theta}} = (\widehat{\theta}_1, \ldots, \widehat{\theta}_d)^\top$ or $\widehat{\boldsymbol{\psi}} = (\widehat{\psi}_1, \ldots, \widehat{\psi}_d)^\top$ of the *weighted* or *unweighted* relative effects as defined in (2) and (3), respectively. Since both of them denote the probabilities (appropriately weighted) of data being smaller in group i than in the joint sample, estimators can be constructed using the (usual) ranks R_{ik} or the so-called *pseudo-ranks* R_{ik}^{ψ} (Happ et al., 2020). In **rankFD** these point estimators are obtained by

`effect=weighted`	(scaled) mean of ranks R_{ik}
`effect=unweighted`	(scaled) mean of pseudo-ranks R_{ik}^{ψ}

For more details, we refer to (Brunner et al., 2019, Section 2.3.2). Besides the vectors of point estimators $\widehat{\boldsymbol{\theta}}$ or $\widehat{\boldsymbol{\psi}}$, their (estimated) covariance matrices are needed for the computation of test statistics. In the general nonparametric setup considered here, we can take advantage of the type of hypothesis we aim to test. Assuming H_0^F to hold, then the covariance matrices of $\sqrt{N}(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta})$ and of $\sqrt{N}(\widehat{\boldsymbol{\psi}} - \boldsymbol{\psi})$ have (much) simpler structures than under H_0^P (Konietschke et al., 2012). This property carries over to its estimation and therefore the estimators used in the statistics for testing H_0^F or H_0^P are different. However, for the ease of notation, we denote with $\widehat{\mathbf{V}}_N$ their estimators in a general way having both versions in mind. In the following, we therefore provide the statistics using $\widehat{\boldsymbol{\psi}}$ (and in turn the pseudo-ranks) for the ease of convenience only. For more details we refer to Brunner et al. (2020).

Global test procedures

In order to test the null hypothesis H_0^F as given in (8), the **rankFD** package implements the *Wald-type* statistic

$$W_N(\mathbf{C}) \quad = \quad N\widehat{\boldsymbol{\psi}}^\top \mathbf{C}^\top \left[\mathbf{C}\widehat{\mathbf{V}}_N\mathbf{C}^\top\right]^+ \mathbf{C}\widehat{\boldsymbol{\psi}}. \tag{14}$$

Here, the matrix $[\mathbf{A}]^+$ denotes the Moore-Penrose inverse of the matrix $\mathbf{A}$. Under the hypothesis H_0^F, the statistic $W_N(\mathbf{C})$ follows, for large sample sizes, a χ^2_w-distribution with $w = \text{rank}(\mathbf{C}\widehat{\mathbf{V}}_N\mathbf{C}^\top)$ degrees of freedom. Since the statistic involves the estimators and the known contrast matrix only, its numerical computation is feasible. However, very large sample sizes ($n_i \geq 50$; depending on the actual design) are necessary for an accurate type-1 error rate control. Therefore, Akritas et al. (1997) and Brunner et al. (2017) propose the so-called *ANOVA-type* statistic

$$A_N(\mathbf{C}) = N \cdot \frac{\widehat{\boldsymbol{\psi}}^\top \mathbf{A} \widehat{\boldsymbol{\psi}}}{\text{trace}(\mathbf{A}\widehat{\mathbf{V}}_N)}, \; \mathbf{A} = \mathbf{C}^\top \left[\mathbf{C}\mathbf{C}^\top\right]^+ \mathbf{C}, \tag{15}$$

and approximate its distribution by an F-distribution with $\widehat{f}_1$ and $\widehat{f}_2$ degrees of freedom (obtained via Box-type approximation as derived by Brunner et al. (1997)). In comparison with the *Wald-type* statistic $W_N(\mathbf{C})$ in (14), the *ANOVA-type* statistic $A_N(\mathbf{C})$ controls the type-I error much better in small sample sizes; $n_i \geq 15$ depending on the design and hypothesis of interest.

Moreover, the approximation of the distribution of $A_N(\mathbf{C})$ is also valid under the more general hypothesis H_0^P. We note that, basically, both statistics can also be computed using the ranks R_{ik} instead of the pseudo-ranks R_{ik}^{ψ}. But the general remarks in Sections 2.2.2 regarding the usual ranks R_{ik} must be carefully considered. We also note that the asymptotic distribution of the *Wald-type* statistic $W_N(\mathbf{C})$ under the more general hypothesis H_0^P is not the χ^2_w-distribution with $w = \text{rank}(\mathbf{C}\widehat{\mathbf{V}}_N\mathbf{C}^\top)$ in general. This would require an additional assumption on the sequence of the empirical covariance matrices $\widehat{\mathbf{V}}_N$ which cannot be verified in practice.

The preceeding comments and discussion might appear somewhat difficult to understand but they are necessary to explain the different options in the printout of **rankFD**. At this point, it becomes evident that the question "Does anybody know whether there is a nonparametric analog of ANOVA?" cannot be answered by some simple statements and that the heuristic technique replacing observations by their ranks and then performing an 'ANOVA on the ranks' may lead to non valid procedures and incorrect conclusions in general.

Multiple contrast test procedures

Both the Wald-type and ANOVA-type statistics are *global* tests, i.e. if the respective hypothesis H_0^F or H_0^P is rejected, the only available information is that *any* of the factor levels (or their combinations) differ at pre-assigned significance level α. The identification of the factor levels which are responsible for the difference is, however, often of major interest and a key research question. Local test decisions in terms of adjusted p-values and simultaneous confidence intervals are of primary importance and key elements of a complete data evaluation. These can be exposed using *Multiple Contrast Test Procedures* (MCTP) (Bretz et al., 2001; Hothorn et al., 2008; Konietschke et al., 2012), which are also known as *max-t-test* type procedures in parametric models (Konietschke et al., 2021). In order to test the local null hypothesis $H_0^{(\ell)} : \mathbf{c}_\ell^\top \boldsymbol{\psi} = 0$, we use the test statistic

$$T_\ell = \sqrt{N}\frac{\mathbf{c}_\ell^\top \widehat{\boldsymbol{\psi}}}{\mathbf{c}_\ell^\top \widehat{\mathbf{V}}_N \mathbf{c}_\ell}\ , \tag{16}$$

where the contrast vector $\mathbf{c}_\ell$ reflects the researcher's particular question. Typical contrast vectors are discussed by Bretz et al. (2001).

Since the statistics T_ℓ and $T_{\ell'}$ are not necessarily independent when $\ell \neq \ell'$, we collect them in the vector $\mathbf{T} = (T_1, \ldots, T_q)^\top$, which follows, asymptotically, as $N \to \infty$, a multivariate normal distribution with expectation $\mathbf{0}$ and correlation matrix $\mathbf{R}$. Since $\mathbf{R}$ is unknown, we replace it with the estimator $\widehat{\mathbf{R}}$ obtained from standardizing $\mathbf{C}^\top \widehat{\mathbf{V}}_N \mathbf{C}$, see Konietschke et al. (2012). For large sample sizes, we reject the individual null hypothesis $H_0^{(\ell)}$ at significance level α, if $|T_\ell| \geq z_{1-\alpha}(\widehat{\mathbf{R}})$, where $z_{1-\alpha}(\widehat{\mathbf{R}})$ denotes the two-sided $(1-\alpha)$-equicoordinate quantile from the $N(\mathbf{0}, \widehat{\mathbf{R}})$ distribution. For details we refer to Konietschke et al. (2012); Umlauft et al. (2019). Compatible $(1-\alpha) \times 100\%$ simultaneous confidence intervals are obtained by $CI_\ell = \left[\mathbf{c}_\ell^\top \widehat{\boldsymbol{\psi}} \mp \frac{z_{1-\alpha}(\widehat{\mathbf{R}})}{\sqrt{N}}\sqrt{\mathbf{c}_\ell^\top \widehat{\mathbf{V}}_N \mathbf{c}_\ell}\right]$. Finally, the global null hypothesis H_0^P (or H_0^F) is rejected, if

$$T_0 = \max\{|T_1|, \ldots, |T_q|\} \geq z_{1-\alpha}(\widehat{\mathbf{R}}). \tag{17}$$

For small sample sizes, Konietschke et al. (2012) suggest to use t quantiles rather than normal and the Fisher-transformation for the computation of range-preserving confidence intervals. The `rankFD` function implements all of the different procedures.

4 Software and examples

In the following, we will analyze different data sets to illustrate the application of the implemented functions in **rankFD**. They differ in their complexity and cover two- and several samples as well as a factorial design, respectively. We note that the wrapper function `rankFD()` realizes the actual statistical design from the given `formula` argument. However, few of the statistical methods are available for two independent samples only and we therefore implemented the function `rank.two.samples` for their exclusive analysis. First, we will explain the syntax of the two functions and then illustrate their application using real data sets.

Syntax

Two samples: The `rank.two.samples()` function implements current state of the art methods for testing the null hypothesis $H_0 : \theta = \frac{1}{2}$ versus $H_1 : \theta \neq \frac{1}{2}$ along with the computation of $(1-\alpha) \times 100\%$ confidence intervals for θ. Its most important arguments are

```
rank.two.samples(formula, data, method = c("t.app", "logit", "probit","normal"),
permu = TRUE, alternative = c("two.sided", "less", "greater"),
wilcoxon = c("asymptotic","exact"),shift.int = TRUE,
nperm = 10000,conf.level = 0.95, info = TRUE,rounds = 4)
```

- `formula` plus `data`
  is the standard way of specifying regression relationships in R/S introduced in Chambers and Hastie (1992).
- `method`
  specifies the approximate method, where `t.app` computes the Brunner-Munzel test (Brunner

and Munzel, 2000) with t-approximation, `normal` uses the standard normal quantiles and range-preserving confidence intervals are obtained by `logit` or `probit` tranformation functions (Pauly et al., 2016).

- `permu`
  indicates whether additional studentized permutation tests shall be computed (Janssen, 1999b; Neubert and Brunner, 2007; Pauly et al., 2016)
- `alternative`
  Two-sided and one-sided tests and confidence intervals are available using the argument `alternative`.
- `wilcoxon`
  gives the option to compute additional Wilcoxon-Mann-Whitney tests for testing the equality of the two distributions $H_0^F : F_1 = F_2$ of the two samples. We use the **coin** package for these computations (Zeileis et al., 2008). Both the asymptotic as well as exact distribution of the test is available.
- `shift.int`
  can be used for the computation of a confidence interval for the shift-effect (Hodges-Lehmann).
- `nperm`, `conf.level`, `info` and `rounds`
  list optional arguments specifying the numbers of permutation, coverage probability, output explanation and decimals.

The use of the `plot()` function to a `rank.two.samples` object displays a plot of the confidence interval for θ.

Several samples and factorial designs: In addition, `rankFD()` implements statistical methods for the analysis of general nonparametric factorial designs. Its most important arguments are:

```
rankFD(formula, data, CI.method = c("logit", "normal"),
        effect = c("unweighted", "weighted"), hypothesis = c("H0F", "H0P"),
        contrast = NULL, sci.method = c("fisher", "multi.t"),
        info = TRUE, rounds=4)
```

- `formula` plus `data`
  is the standard way of specifying regression relationships in R/S introduced in Chambers and Hastie (1992).
- `CI.method`
  specifies the computational method of the confidence intervals, either using the normal approximation or the logit transformation function.
- `effect`
  defines the effect to be estimated, in particular,
  `effect = "weighted"` or `effect = "unweighted"`
  estimate the weighted or unweighted relative effect, respectively. As explained above, this choice either leads to using traditional ranks (weighted) or pseudo-ranks (unweighted).
- `hypothesis`
  defines the null hypothesis of interest (either H_0^F or H_0^P formulated in terms of distribution functions or relative effects, respectively).
- `contrast`
  is specified to perform multiple contrast tests. The argument must be given as a `list()` specifying the factor level and the kind of contrast (optional). The user can chose from a pre-implemented list of possible contrasts or commit a user-specific contrast matrix.
- `sci.method`
  defines the computational method of the simultaneous confidence intervals.
- `Factor.Information`
  is a logical argument whether descriptive information (effect estimators, standard error and confidence intervals) for each factor and interaction effect is of interest and shall be displayed.
- `info` and `rounds`
  list optional arguments specifying the numbers of output explanation and decimals.

Plot options: In order to visualize the results of the analysis, the confidence intervals can be plotted by using the generic `plot()` function (being applied to a rankFD object). In two- and higher way layouts, the user is asked to type the name of the main or interaction effect the confidence intervals of which should be drawn. All standard font, width and color arguments apply (lwd, pch, cex, etc.). Furthermore, the argument `cex.ci` sets the "cex" (number indicating the amount by which plotting text and symbols should be scaled relative to the default) of the confidence interval limits.

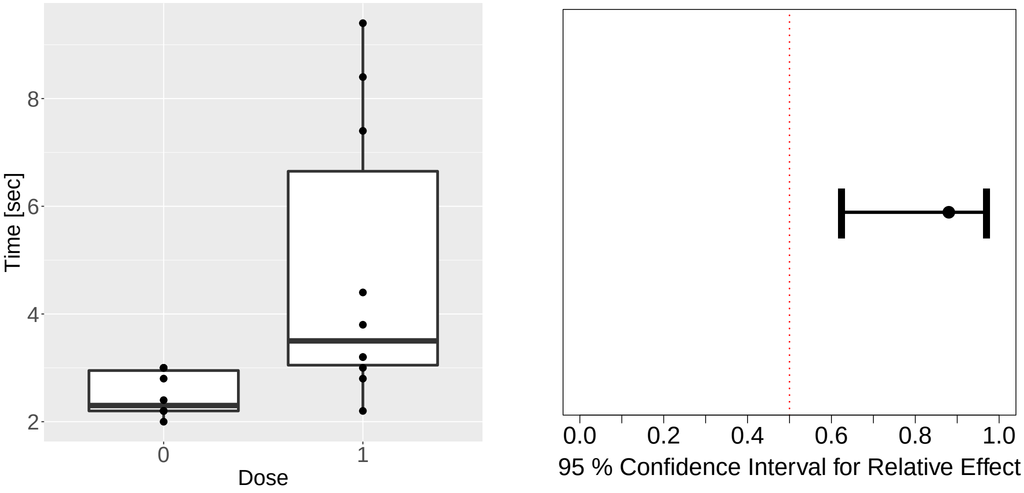


Figure 1: Boxplots (left) and 95%-confidence interval (right) for the relative effect of the reaction time data.

Two independent samples

As an illustrating example, we use a part of the reaction time data provided by Shirley (1977). In this animal experiment, $N = 40$ mice were randomized to $a = 4$ dose groups ($n = 10$ animals per group). The observations are the reaction times [in seconds] of mice to stimuli applied to their tails. Here, we only use the data from dose group 0 (negative control) and dose group 1 and thus reduce the data set to two independent samples. The boxplots of the reaction times as displayed in Figure 1 confirm our initial conjecture of quite skewed distributions. In this case, the *Wilcoxon-Mann-Whitney* effect:

$$\theta = P(X_{01} < X_{11}) + \tfrac{1}{2}P(X_{01} = X_{11}),$$

may have a better interpretation for the researcher than the difference of the two means. Recall that for $\theta < \frac{1}{2}$ the observations coming from the control group tend to be larger than those from group 1. If $\theta = \frac{1}{2}$, then none of the observations tend to be smaller or larger. *No treatment effect* is therefore indicated by $\theta = \frac{1}{2}$. The reaction time data set is analyzed with the `rank.two.samples()` function. As approximate method, we use the `logit` approach, compute the `exact` Wilcoxon-Mann-Whitney test but omit estimation of shift effects:

```
library("rankFD")
data("reaction")

A <- rank.two.samples(Time ~ Group, data = reaction, method = "logit",
+                         wilcoxon = "exact", shift.int = FALSE)


               Nonparametric Methods for 2 Independent Samples

 #Alternative: Relative Effect is unequal to 1/2
 #Method: Logit Transformation
 #Interpretation: If p(0,1) >1/2, then data in group 1 tend to be
                  larger than those in group 0
 #Confidence Level: 95 %
 #Number of permutations: 10000
 #Wilcoxon-Mann-Whitney Test: exact
 #Shift-Effect: NA
---------------------------------------------------------------------------
Call:
Time ~ Group

Descriptive:
  Sample Size
0      0   10
```

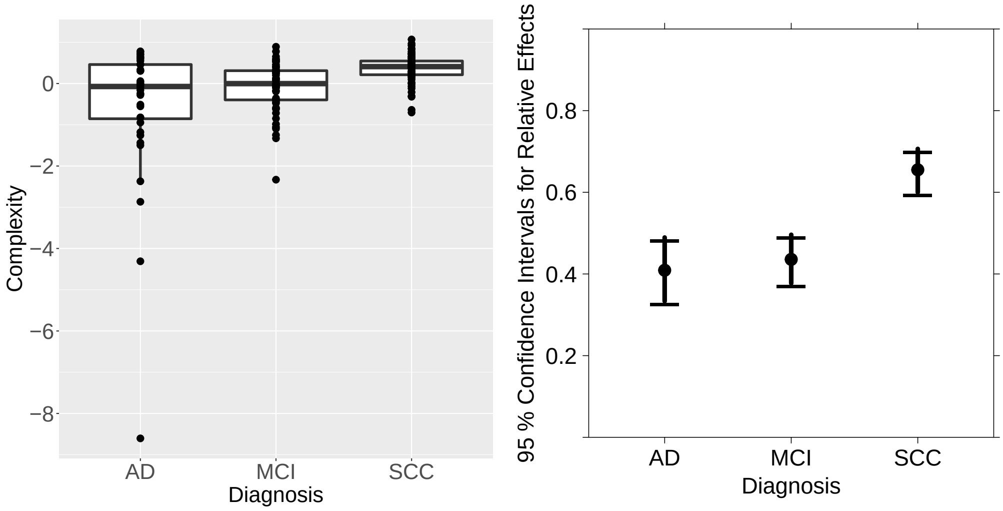


Figure 2: Boxplots (left) and (local) 95%-confidence intervals for the relative effects of the EEG values (right) .

```
1      1   10
----------------------Analysis of Relative Effects-------------------------
Test Results:
 Effect Estimator Std.Error      T  Lower  Upper p.Value
 p(0,1)      0.88    0.0801 2.6277 0.6239 0.9701  0.0086

Studentized Permutation Test:
 Effect Estimator Std.Error      T  Lower  Upper p.Value
 p(0,1)      0.88    0.0801 2.6277 0.6551 0.9673  0.0016

-------------------Analysis of Distribution Functions-----------------------

Wilcoxon-Mann-Whitney Test:
 Effect Estimator Statistic p.Value
 p(0,1)      0.88       143  0.0029

plot(A)
```

The estimated relative effect $\widehat{\theta} = 0.88$ and thus, the estimated probability that untreated mice react faster than treated ones is 88%. Furthermore, the data provide the evidence to reject $H_0^\theta : \theta = \frac{1}{2}$ at 5% level of significance ($p < 5\%$) which is also evident in the compatible confidence interval (not containing $1/2$).

A one-way factorial design

As an example of a one-way factorial design we use the data set EEG that is included in the package **MANOVA.RM** (Friedrich et al., 2019a, 2021). The data set contains EEG measurements of 160 patients who were diagnosed with either Alzheimer's Disease (AD), mild cognitive impairments (MCI), or subjective cognitive complaints without clinically significant deficits (SCC), based on neuropsychological diagnostics (Bathke et al., 2018). For demonstration purposes, we restrict our analysis to the measurement of Hjorth complexity (represents change in frequency) obtained at central electrode positions. The question of interest is whether this EEG value tends to be larger or smaller than the mean Mann-Whitney effect across the different diseases and therefore, the relative effects defined in (3) are used for the analysis.

The EEG data is analyzed using the function rankFD(). Here, we calculate confidence intervals with the logit approach and estimate the unweighted relative treatment effects to test the null hypothesis H_0^p. Moreover, we specify a multiple contrast test based on Tukey-type contrasts for the pairwise comparisons of the three diagnosis groups.

```
library("MANOVA.RM")
data("EEGwide")
B <- rankFD(complexity_central ~ diagnosis, data = EEGwide,
+        CI.method = "logit", effect = "unweighted", hypothesis = "H0p",
+        contrast = list("diagnosis", "Tukey"))


Nonparametric Methods for General Factorial Designs

---------------------------------------------------------------------------
#Hypotheses: Tested in Relative Effects
#Ranking Method: Pseudo-Ranks
#Confidence Intervals: 95 % with Logit-Transformation

#MCTP: Fisher Transformation and multivariate T-Approximation
---------------------------------------------------------------------------

Call:
complexity_central ~ diagnosis

Descriptive:
   diagnosis Size Rel.Effect Std.Error  Lower  Upper
1        AD   36     0.4091    0.0400 0.3335 0.4893
2       MCI   57     0.4357    0.0304 0.3773 0.4960
3       SCC   67     0.6551    0.0272 0.6002 0.7063

Wald.Type.Statistic:
          Statistic df p-Value
diagnosis   36.2624  2       0

ANOVA.Type.Statistic:
          Statistic    df1     df2 p-Value
diagnosis   11.1605 1.6222 80.9562   2e-04

MCTP:
$Contrast.Matrix
1  2 3
C1 -1  1 0
C2 -1  0 1
C3  0 -1 1

$Local.Results
  Effect Std.Error      T   Lower  Upper p.value
C1 0.0266    0.0657 0.4044 -0.1308 0.1827  0.9119
C2 0.2460    0.0613 3.8510  0.0941 0.3868  0.0010
C3 0.2194    0.0415 5.1125  0.1176 0.3167  0.0000

$Global.Result
   T0     p.value
1 5.1125       0

$DF
[1] 46

$Quantile
[1] 2.4042

plot(B)
```

The output consists of several parts: First, a brief description of the methods is given. `B$Descriptive` returns the sample sizes, the estimated relative effects as well as their standard errors and confidence intervals for the factor levels. `B$ Wald.Type.Statistic` and `B$ANOVA.Type.Statistic` return the results of the Wald-type and ANOVA-type test as described in Section 2.3, respectively. Since we specified our null hypothesis in terms of H_0^p, Kruskal-Wallis test is not performed. The part `B$MCTP` finally contains the results of the multiple contrast test: the contrast matrix (Tukey-type), the local test results T_ℓ as well as the global results T_0 along with the t-quantile and the corresponding degrees of freedom (Konietschke et al., 2012) are reported, see Section 2.3.2 for details. The significant difference between the diagnosis groups and the results of the post-hoc tests reveal that SCC patients differ

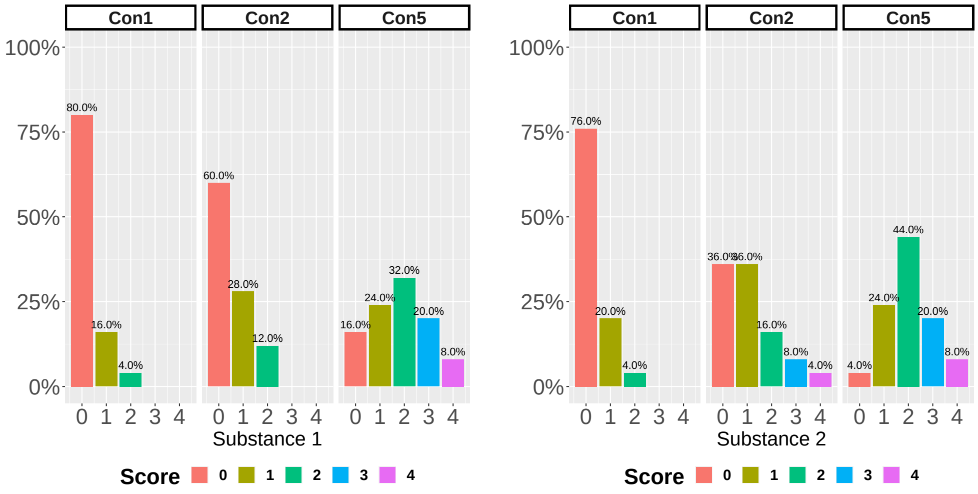


Figure 3: Barplots (percent) of the nasal mucosa scores.

significantly from the other two groups, see also Figure 2.

A two-way factorial design

As an illustrative example of a two-way factorial design, we chose the *Irritation of the Nasal Mucosa* trial provided by Brunner et al. (2019, Chapter B.3.2) and included in the package. In this trial, the researchers investigated the damage of two gaseous substances (factor A) on the nasal mucous membrane of mice. Hereby, both substances were given in three different concentrations (1[ppm], 2[ppm] and 5[ppm]) (factor B) to 25 mice each. The degree of irritation and damage was histopathologically assessed using an ordinal score ranging from 0 to 4 with 0 = "no irritation", 1 = "mild irritation", 2 = "strong irritation", 3 = "severe irritation" and 4 = "irreversible damage", respectively. The outcome is displayed in Figure 3. The code to analyze this data is similar to that provided above, but we additionally include an interaction term in the formula. In this example, we formulate the null hypothesis in terms of the distribution functions to show the R-code for testing this hypothesis. Note that due to the balanced design, both weighted and unweighted estimators give the same results.

```
data(nms)
rankFD(score ~ conc * subst, data = nms,
+          hypothesis = "H0F")


Nonparametric Methods for General Factorial Designs

---------------------------------------------------------------------------
#Hypotheses: Tested in Distribution Functions
#Ranking Method: Pseudo-Ranks
#Confidence Intervals: 95 % with Logit-Transformation

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


Call:
score ~ conc * subst

Descriptive:
  conc subst Size Rel.Effect Std.Error  Lower  Upper
1    1     1   25     0.3053    0.0310 0.2481 0.3693
2    1     2   25     0.3193    0.0320 0.2601 0.3851
3    2     1   25     0.3927    0.0386 0.3200 0.4704
4    2     2   25     0.5296    0.0459 0.4397 0.6176
5    5     1   25     0.6925    0.0429 0.6027 0.7698
6    5     2   25     0.7605    0.0310 0.6947 0.8159

Wald.Type.Statistic:
```

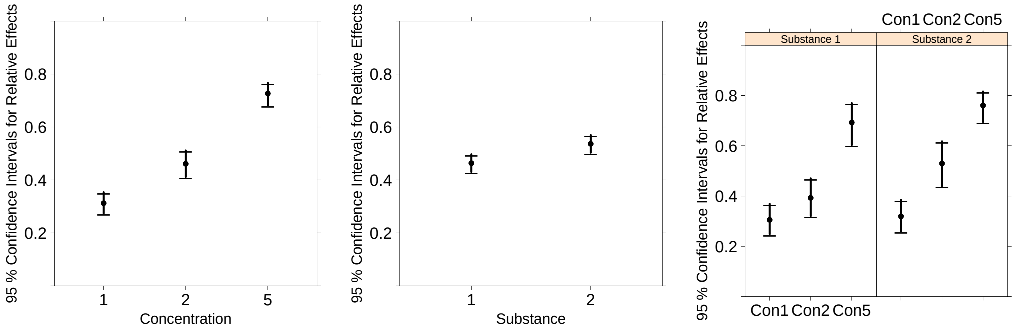


Figure 4: Local 95%-confidence interval for the (relative) main and interaction effects of the reaction time data.

```
            Statistic df p-Value
conc        114.9046  2  0.0000
subst         4.5200  1  0.0335
conc:subst    2.2174  2  0.3300

ANOVA.Type.Statistic:
            Statistic    df1     df2   p-Value
conc         49.8167 1.9289 127.0195  0.0000
subst         4.5200 1.0000 127.0195  0.0354
conc:subst    1.0741 1.9289 127.0195  0.3428
```

The right plot in Figure 4 shows that the relative effects increase at a similar rate in both levels of the main effect suggesting no qualitative interaction between the factor substance and the concentration.

5 Summary

The **rankFD**-package implements current state of the art rank methods for nonparametric inference in general factorial designs with independent observations. It comprises of functions for computing various test statistics for testing null hypotheses formulated either in distribution functions or in relative effects using ranks or pseudo-ranks, respectively. Up until now, no other software package for testing null hypotheses in relative effects in general factorial designs have existed. Besides global procedures (Wald-type and ANOVA-type statistics) using quadratic forms, **rankFD** implements multiple contrast tests and simultaneous confidence intervals for relative effects. The possibility of testing contrasts between the main and interaction effects makes **rankFD** a powerful tool for the application of nonparametric methods in data analysis and a useful addition to **nparcomp** (Konietschke et al., 2015). Besides the inference methods discussed above, **rankFD** furthermore implements formulas for computing sample sizes using the functions `WMWSSP()` and `noether()` (Happ et al., 2019). Since these methods apply for two independent samples only, we did not discuss them in the present manuscript.

We designed the package and its functions to be similar to the well known *R*-functions `lm()`, `aov()` for the analysis of linear models and the `glht()` function of the **multcomp** package for the computation of multiple contrast tests in means. Both **rankFD** and **multcomp** use the **mvtnorm** package (Genz et al., 2021) for the computation of critical values. However, as explained in detail in the Introduction, the effect measures used in multcomp and mvtnorm are different from those used in **rankFD**. In general parlance, this means that the parametric and nonparametric methods are not comparable at hand.

We plan to update **rankFD** frequently with novel procedures. For instance, various international research groups are currently investigating rank-based methods for the analysis of clustered data, see also the package **clusrank** Jiang et al. (2020) for the analysis of two samples, sample size planning, as well as analysis of covariance methods. We plan to add these methods in the future. The package **rankFD** is online available on *CRAN*.

Acknowledgement: The authors are grateful to the Editor and two anonymous referees for helpful comments which considerably improved the paper. This work was supported by the German Research Foundation project DFG KO 4680/4-1.

Bibliography

L. Acion, J. J. Peterson, S. Temple, and S. Arndt. Probabilistic index: an intuitive non-parametric approach to measuring the size of treatment effects. *Statistics in medicine*, 25(4):591–602, 2006. [p142]

M. G. Akritas and S. F. Arnold. Fully nonparametric hypotheses for factorial designs i: Multivariate repeated measures designs. *Journal of the American Statistical Association*, 89(425):336–343, 1994. [p143, 145]

M. G. Akritas, S. F. Arnold, and E. Brunner. Nonparametric hypotheses and rank statistics for unbalanced factorial designs. *Journal of the American Statistical Association*, 92(437):258–265, 1997. [p143, 145, 148]

A. C. Bathke, S. Friedrich, M. Pauly, F. Konietschke, W. Staffen, N. Strobl, and Y. Höller. Testing mean differences among groups: Multivariate and repeated measures analysis with minimal assumptions. *Multivariate Behavioral Research*, 53(3):348–359, 2018. [p152]

Z. W. Birnbaum and O. M. Klose. Bounds for the variance of the mann-whitney statistic. *The Annals of Mathematical Statistics*, 28(4):933–945, 1957. [p142]

F. Bretz, A. Genz, and L. Hothorn. On the numerical availability of multiple comparison procedures. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, 43(5):645–656, 2001. [p147, 149]

E. Brunner and U. Munzel. The nonparametric behrens-fisher problem: asymptotic theory and a small-sample approximation. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, 42(1): 17–25, 2000. [p146, 149]

E. Brunner and M. L. Puri. Nonparametric methods in design and analysis of experiments. *Handbook of Statistics*, 13:631–703, 1996. [p143, 145]

E. Brunner, H. Dette, and A. Munk. Box-type approximations in nonparametric factorial designs. *Journal of the American Statistical Association*, 92(440):1494–1502, 1997. [p148]

E. Brunner, F. Konietschke, M. Pauly, and M. L. Puri. Rank-based procedures in factorial designs: hypotheses about non-parametric treatment effects. *Journal of the Royal Statistical Society-Series B*, 79 (5):1463–1485, 2017. [p143, 146, 148]

E. Brunner, A. C. Bathke, and F. Konietschke. *Rank and Pseudo-Rank Procedures for Independent Observations in Factorial Designs*. Springer, 2019. [p143, 144, 146, 147, 148, 154]

E. Brunner, F. Konietschke, A. C. Bathke, and M. Pauly. Ranks and Pseudo-ranks – Surprising Results of Certain Rank Tests in Unbalanced Designs. *International Statistical Review*, 2020. [p143, 146, 148]

W. W. Burchett, A. R. Ellis, S. W. Harrar, and A. C. Bathke. Nonparametric inference for multivariate data: the r package npmv. *Journal of Statistical Software*, 76:1–18, 2017. [p143]

J. M. Chambers and T. J. Hastie, editors. *Statistical Models in S*. Chapman & Hall, London, 1992. [p149, 150]

W. J. Conover and R. L. Iman. Rank transformations as a bridge between parametric and nonparametric statistics. *The American Statistician*, 35(3):124–129, 1981. [p143]

R. B. D'Agostino, M. Campbell, and J. Greenhouse. The mann–whitney statistic: continuous use and discovery: Special papers for the 25th anniversary of statistics in medicine, 2006. [p142]

D. Dobler, S. Friedrich, and M. Pauly. Nonparametric manova in meaningful effects. *Annals of the Institute of Statistical Mathematics*, pages 1–26, 2019. [p142]

C. W. Dunnett. A multiple comparison procedure for comparing several treatments with a control. *Journal of the American Statistical Association*, 50(272):1096–1121, 1955. [p147]

M. A. Fligner and G. E. Policello. Robust rank procedures for the behrens-fisher problem. *Journal of the American Statistical Association*, 76(373):162–168, 1981. [p146]

S. Friedrich, F. Konietschke, and M. Pauly. Gfd: an r package for the analysis of general factorial designs. *Journal of Statistical Software*, 79:1–18, 2017. [p143]

S. Friedrich, F. Konietschke, and M. Pauly. Resampling-Based Analysis of Multivariate Data and Repeated Measures Designs with the R Package MANOVA.RM. *The R Journal*, 2(11):380–400, 2019a. doi: 10.32614/RJ-2019-051. [p152]

S. Friedrich, F. Konietschke, and M. Pauly. Resampling-based analysis of multivariate data and repeated measures designs with the r package manova. rm. *R Journal*, 11(2):380, 2019b. [p143]

S. Friedrich, F. Konietschke, and M. Pauly. **MANOVA.RM**: *Resampling-Based Analysis of Multivariate Data and Repeated Measures Designs*, 2021. R package version 0.4.3. [p152]

A. Genz, F. Bretz, T. Miwa, X. Mi, F. Leisch, F. Scheipl, B. Bornkamp, M. Maechler, and T. Hothorn. Package 'mvtnorm'. *Journal of Computational and Graphical Statistics*, 11:950–971, 2021. [p155]

M. Happ, A. C. Bathke, and E. Brunner. Optimal sample size planning for the wilcoxon-mann-whitney test. *Statistics in medicine*, 38(3):363–375, 2019. [p155]

M. Happ, G. Zimmermann, E. Brunner, and A. C. Bathke. Pseudo-ranks: How to calculate them efficiently in R. *Journal of Statistical Software, Code Snippets*, 95(1):1–22, 2020. [p144, 148]

T. P. Hettmansperger and R. M. Norton. Tests for patterned alternatives in k-sample problems. *Journal of the American Statistical Association*, 82(397):292–299, 1987. [p145]

T. Hothorn, F. Bretz, and P. Westfall. Simultaneous inference in general parametric models. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, 50(3):346–363, 2008. [p147, 149]

A. Janssen. Studentized permutation tests for non-iid hypotheses and the generalized behrens-fisher problem. *Statistics & Probability Letters*, 36(1):9–21, 1997. [p142]

A. Janssen. Nonparametric symmetry tests for statistical functionals. *Mathematical Methods of Statistics*, 8(3):320–343, 1999a. [p142]

A. Janssen. Testing nonparametric statistical functionals with applications to rank tests. *Journal of Statistical Planning and Inference*, 81(1):71–93, 1999b. [p150]

Y. Jiang, X. He, M.-L. T. Lee, B. Rosner, and J. Yan. Wilcoxon rank-based tests for clustered data with r package clusrank. *Journal of Statistical Software*, 96:1 – 26, 2020. [p155]

E. Kahler, A. Rogausch, E. Brunner, and W. Himmel. A parametric analysis of ordinal quality-of-life data can lead to erroneous results. *Journal of Clinical Epidemiology*, 61(5):475–480, 2008. [p142]

M. Kiefel, A. C. Bathke, and M. M. Kiefel. Package 'nparmd'. 2022. [p143]

F. Konietschke, L. A. Hothorn, and E. Brunner. Rank-based multiple test procedures and simultaneous confidence intervals. *Electronic Journal of Statistics*, 6:738–759, 2012. [p143, 146, 148, 149, 153]

F. Konietschke, M. Placzek, F. Schaarschmidt, and L. A. Hothorn. nparcomp: an r software package for nonparametric multiple comparisons and simultaneous confidence intervals. *Journal of statistical software*, 64(1):1–17, 2015. [p155]

F. Konietschke, K. Schwab, and M. Pauly. Small sample sizes: A big data problem in high-dimensional data analysis. *Statistical Methods in Medical Research*, 30(3):687–701, 2021. [p149]

W. H. Kruskal. A nonparametric test for the several sample problem. *The Annals of Mathematical Statistics*, pages 525–540, 1952. [p145]

W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952. [p145]

H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947. [p142]

C. R. Mehta, N. R. Patel, and P. Senchaudhuri. Importance sampling for estimating exact probabilities in permutational inference. *Journal of the American Statistical Association*, 83(404):999–1005, 1988. [p143]

K. Neubert and E. Brunner. A studentized permutation test for the non-parametric Behrens–Fisher problem. *Computational Statistics & Data Analysis*, 51(10):5192–5204, 2007. [p150]

K. Noguchi, Y. R. Gel, E. Brunner, and F. Konietschke. nparld: an r software package for the nonparametric analysis of longitudinal data in factorial experiments. *Journal of Statistical software*, 50:1–23, 2012. [p143]

M. Pauly, E. Brunner, and F. Konietschke. Asymptotic permutation tests in general factorial designs. *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, 77(2):461–473, 2015. [p142]

M. Pauly, T. Asendorf, and F. Konietschke. Permutation-based inference for the auc: a unified approach for continuous and discontinuous data. *Biometrical Journal*, 58(6):1319–1337, 2016. [p150]

J. Putter. The treatment of ties in some nonparametric tests. *The Annals of Mathematical Statistics*, 26(3): 368–386, 1955. [p142]

F. H. Ruymgaart. A unified approach to the asymptotic distribution theory of certain midrank statistics. In *Statistique non Parametrique Asymptotique*, pages 1–18. Springer, 1980. [p143]

E. Shirley. A non-parametric equivalent of Williams' test for contrasting increasing dose levels of a treatment. *Biometrics*, 33(2):386–389, 1977. [p151]

Ł. Smaga. Wald-type statistics using $\{2\}$-inverses for hypothesis testing in general factorial designs. *Statistics & Probability Letters*, 107:215–220, 2015. [p142]

B. Streitberg and J. Röhmel. Exact distributions for permutation and rank tests: an introduction to some recently published algorithms. *Statistical Software Newsletter*, 12(1):10–17, 1986. [p143]

O. Thas, J. D. Neve, L. Clement, and J.-P. Ottoy. Probabilistic index models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(4):623–671, 2012. [p142]

M. Umlauft, M. Placzek, F. Konietschke, and M. Pauly. Wild bootstrapping rank-based procedures: Multiple testing in nonparametric factorial repeated measures designs. *Journal of Multivariate Analysis*, 171:176–192, 2019. [p149]

A. Zeileis, M. A. Wiel, K. Hornik, and T. Hothorn. Implementing a class of permutation tests: the **coin** package. *Journal of Statistical Software*, 28(8):1–23, 2008. [p150]

G. Zimmermann, E. Brunner, W. Brannath, M. Happ, and A. C. Bathke. Pseudo-ranks: The better way of ranking? *The American Statistician*, 76(2):124–130, 2022. [p146]

Frank Konietschke
Charité Universitätsmedizin Berlin
Institute of Biometry and Clinical Epidemiology
Reinhardstr. 58
10117 Berlin, Germany
Frank.Konietschke@charite.de

Edgar Brunner
University Medical School Göttingen
Institut für Medizinische Statistik
Humboldtallee 32
37073 Göttingen, Germany
ebrunne1@gwdg.de

The segmetric Package: Metrics for Assessing Segmentation Accuracy for Geospatial Data

Rolf Simoes, Alber Sanchez, Michelle C. A. Picoli and Patrick Meyfroidt

Abstract Segmentation methods are a valuable tool for exploring spatial data by identifying objects based on images' features. However, proper segmentation assessment is critical for obtaining high-quality results and running well-tuned segmentation algorithms Usually, various metrics are used to inform different types of errors that dominate the results. We describe a new R package, **segmetric**, for assessing and analyzing the geospatial segmentation of satellite images. This package unifies code and knowledge spread across different software implementations and research papers to provide a variety of supervised segmentation metrics available in the literature. It also allows users to create their own metrics to evaluate the accuracy of segmented objects based on reference polygons. We hope this package helps to fulfill some of the needs of the R community that works with Earth Observation data.

1 Introduction

Earth Observation aims to collect data regarding the Earth's systems at several spatio-temporal resolutions. These data allow scientists to understand Earth's processes, such as greenhouse gas emissions and land cover change. Segmentation is among the most used unsupervised processing methods for extracting information from satellite imagery (Hossain and Chen, 2019). Segmentation is the process by which objects are extracted using image features. This process consists of delineating groups of adjacent pixels with similar characteristics such as intensity, color, and texture. Numerous segmentation algorithms are available in the Remote Sensing scientific literature (see e.g. Kotaridis and Lazaridou, 2021).

Assessing segmentation results is difficult due to under and over-segmetation errors. Undersegmentation occurs when the segmentation algorithm fails to separate a contiguous pixel group while oversegmentation is the opposite, that is, the segmentation algorithm unnecessarily splits a pixel group (Costa et al., 2018). Both under and over-segmentation come with assessment metrics that target segments' characteristics such as area, shape, and position. One way to assess these errors is to use supervised quality metrics (Costa et al., 2018). Supervised metrics compare segments to reference data, measuring their similarity or discrepancy in terms of under and over-segmentation (Clinton et al., 2010).

One significant challenge in the domain of Earth Observation data is the scarcity of software tools specifically designed for segmentation assessment. While several packages such as **imageseg** (Niedballa et al., 2022), **ExpImage** (Azevedo, 2022), **SuperpixelImageSegmentation** (Mouselimis, 2022a), **OpenImageR** (Mouselimis, 2022b), and **image.Otsu** (Wijffels, 2020) enable users to segment images but, when provided, they offer a limited set of facilities to assess the accuracy of the segmentation and some of them are not tailored to the needs of Earth Observation data or applications. This often requires users to adapt the code from these packages for their own purposes, which can be time-consuming and unrelated to their primary research goals.

In this paper, we introduce the **segmetric** package, which addresses the lack of R tools for assessing segmentation of Earth Observation data and provides a coherent set of metrics that can be used to compare and contrast different assessment methods for evaluating segmentation. Additionally, **segmetric** provides innovative visualization tools to assist qualitative spatial analysis as well as metrics that can be used to tune and assess segmentation algorithms.

2 Supervised segmentation metrics

Supervised metrics use reference data to assess segmentation accuracy. These metrics are grouped into two categories: geometric, which use the geometry of the objects (i.e. polygons) to determine the similarity between the segments and the reference data; and thematic, which use instead objects' attributes such as the land cover label associated with each object (Costa et al., 2018). The **segmetric** package focuses on geometric methods that require two sets of polygons as inputs, one for the segments and other for the reference data.

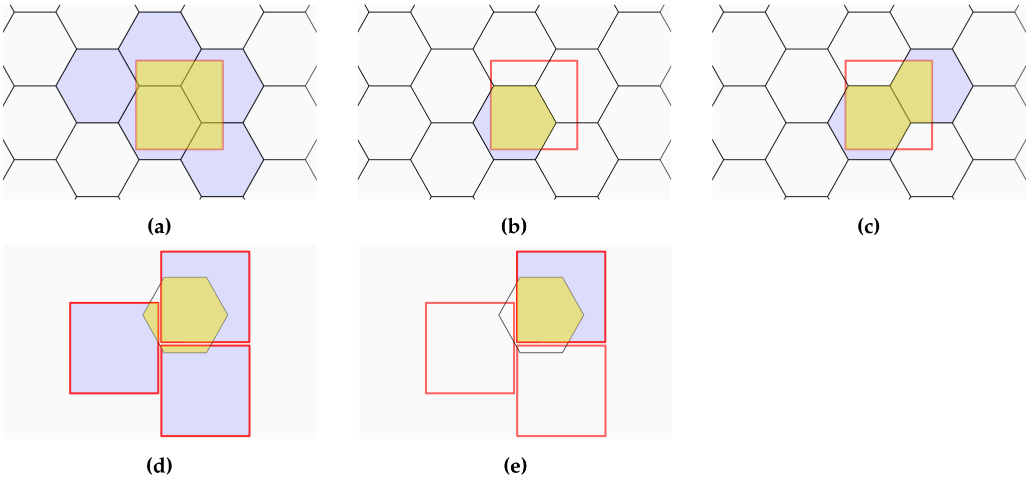
The segments' polygons, denoted by $Y = \{y_j : j = 1, ..., m\}$, are obtained from a segmentation method and the reference polygons, denoted by $X = \{x_i : i = 1, ..., n\}$, are typically collected *in-situ* by specialists. The quality metrics are defined considering different subsets of X and Y. The subsets of Y used to compute metrics for each reference i are defined as follows [1]:

- $\tilde{Y}_i \subset Y$, where $\tilde{Y}_i = \{y_j : \mathrm{area}(\mathrm{x_i} \cap \mathrm{y_j}) \neq 0\}$
- $Y_i' \subset Y$, where $Y_i' = \{y_j : max(\mathrm{area}(\mathrm{x_i} \cap \mathrm{y_j}))\}$
- $Ya_i \subset \tilde{Y}_i$, where $Ya_i = \{y_j : \mathrm{centroid}(\mathrm{x_i})\ \mathrm{in}\ \mathrm{y_j}\}$
- $Yb_i \subset \tilde{Y}_i$, where $Yb_i = \{y_j : \mathrm{centroid}(\mathrm{y_j})\ \mathrm{in}\ \mathrm{x_i}\}$
- $Yc_i \subset \tilde{Y}_i$, where $Yc_i = \{y_j : \mathrm{area}(\mathrm{x_i} \cap \mathrm{y_j})/\mathrm{area}(\mathrm{y_j}) > 0.5\}$
- $Yd_i \subset \tilde{Y}_i$, where $Yd_i = \{y_j : \mathrm{area}(\mathrm{x_i} \cap \mathrm{y_j})/\mathrm{area}(\mathrm{x_i}) > 0.5\}$
- Y_i^*, where $Y_i^* = Ya_i \cup Yb_i \cup Yc_i \cup Yc_i$
- Ycd_i, where $Ycd_i = Yc_i \cup Yd_i$
- $Ye_i \subset \tilde{Y}_i$, where $Ye_i = \{y_j : \mathrm{area}(\mathrm{x_i} \cap \mathrm{y_j})/\mathrm{area}(\mathrm{y_j}) = 1\}$
- $Yf_i \subset \tilde{Y}_i$, where $Yf_i = \{y_j : \mathrm{area}(\mathrm{x_i} \cap \mathrm{y_j})/\mathrm{area}(\mathrm{y_j}) > 0.55\}$
- $Yg_i \subset \tilde{Y}_i$, where $Yg_i = \{y_j : \mathrm{area}(\mathrm{x_i} \cap \mathrm{y_j})/\mathrm{area}(\mathrm{y_j}) > 0.75\}$

Likewise, the subsets of X used to compute metrics for each segment j are defined as:

- $\tilde{X}_j \subset X$, where $\tilde{X}_j = \{x_i : \mathrm{area}(\mathrm{y_j} \cap \mathrm{x_i}) \neq 0\}$
- $X_j' \subset X$, where $X_j' = \{x_i : max(\mathrm{area}(\mathrm{y_j} \cap \mathrm{x_i}))\}$

To illustrate these subsets definition, we depict some of them in Figure 1. Subsets contains all elements for which a metric value has to be computed. To obtain a single metric value, a summary function can be applied on all values, typically a mean or a weighted mean. The range of possible values can differ from metric to metric. Also, the optimal value varies for each metric. Table 1 lists all implemented metrics in **segmetric**, their ranges and optimal values. The corresponding subsets used to compute values are shown in their formulas definition.



(a) **(b)** **(c)**

(d) **(e)**

Figure 1: Subsets are used to compute segmentation metrics. The red squares and black hexagons represent reference and segmented polygons, respectively. (a) $\tilde{Y}$ is made of segmentation polygons overlapping any reference polygon. (b) Y' is made of the segmentation polygon overlapping the most of a reference polygon. (c) Y^* is made of polygons in the segmentation that either their centroids fall inside the reference, or they cover or overlap more than half of the reference polygon, or the reference polygon centroid falls inside them. (d) $\tilde{X}$ is made of the reference polygons overlapping any segmentation polygon. (e) X' is made of the reference polygons, which overlap the most with the segmentation polygons. Yellow represents the intersection between the references and segments included in a subset. The magenta areas are excluded from the reference-segmentation intersection but included in the subset. Adapted from Jozdani and Chen (2020).

[1] We are following the notation used by Clinton et al. (2010) and Costa et al. (2018)

Table 1: Metrics implemented

Metric	Range	Opt.	References
$\text{OS1}_{ij} = 1 - \frac{\text{area}(x_i \cap y_j)}{\text{area}(x_i)},\ y_j \in Y^*{}_i$	$[0,1]$	0	Clinton et al. (2010)
$\text{OS2}_{ij} = 1 - \frac{\text{area}(x_i \cap y_j)}{\text{area}(x_i)},\ y_j \in Y'{}_i$	$[0,1]$	0	Persello and Bruzzone (2010)
$\text{OS3}_{ij} = 1 - \frac{\text{area}(x_i \cap y_j)}{\text{area}(x_i)},\ y_j \in \text{Ycd}_i$	$[0,1]$	0	Yang et al. (2014)
$\text{US1}_{ij} = 1 - \frac{\text{area}(x_i \cap y_j)}{\text{area}(y_i)},\ y_j \in Y^*{}_i$	$[0,1]$	0	Clinton et al. (2010)
$\text{US2}_{ij} = 1 - \frac{\text{area}(x_i \cap y_j)}{\text{area}(y_i)},\ y_j \in Y'{}_i$	$[0,1]$	0	Persello and Bruzzone (2010)
$\text{US3}_{ij} = 1 - \frac{\text{area}(x_i \cap y_j)}{\text{area}(y_i)},\ y_j \in \text{Ycd}_i$	$[0,1]$	0	Yang et al. (2014)
$\text{AFI}_{ij} = \frac{\text{area}(x_i) - \text{area}(y_j)}{\text{area}(x_i)},\ y_j \in Y'{}_i$	$(-\infty,1]$	0	Lucieer and Stein (2002); Clinton et al. (2010)
$\text{QR}_{ij} = 1 - \frac{\text{area}(x_i \cap y_j)}{\text{area}(x_i \cup y_j)},\ y_j \in Y^*{}_i$	$[0,1]$	0	Weidner (2008); Clinton et al. (2010)
$\text{D}_{ij} = \sqrt{\frac{\text{OS}_{ij}^2 + \text{US}_{ij}^2}{2}}$	$[0,1]$	0	Levine and Nazif (1982); Clinton et al. (2010)
$\text{precision}_{ij} = \frac{\text{area}(x_i \cap y_j)}{\text{area}(y_i)},\ y_j \in Y'{}_i$	$[0,1]$	1	van Rijsbergen (1979); Zhang et al. (2015)
$\text{recall}_{ij} = 1 - \frac{\text{area}(x_i \cap y_j)}{\text{area}(x_i)},\ y_j \in Y'{}_i$	$[0,1]$	1	van Rijsbergen (1979); Zhang et al. (2015)
$\text{UMerging}_{ij} = \frac{\text{area}(x_i) - \text{area}(x_i \cap y_j)}{\text{area}(x_i)},\ y_j \in Y^*{}_i$	$[0,1]$	0	Levine and Nazif (1982); Clinton et al. (2010)
$\text{OMerging}_{ij} = \frac{\text{area}(y_j) - \text{area}(x_i \cap y_j)}{\text{area}(x_i)},\ y_j \in Y^*{}_i$	$[0,\infty)$	0	Levine and Nazif (1982); Clinton et al. (2010)
$\text{M}_{ij} = \sqrt{\frac{\text{area}(x_i \cap y_j)^2}{\text{area}(x_i)\text{area}(y_j)}},\ y_j \in Y'_i$	$[0,1]$	1	Janssen and Molenaar (1995); Feitosa et al. (2010)
$\text{E}_{ij} = \frac{\text{area}(y_j) - \text{area}(x_i \cap y_j)}{\text{area}(y_i)} \times 100,\ x_i \in X'{}_j$	$[0,100]$	0	Carleer et al. (2005)
$\text{RAsub}_{ij} = \frac{\text{area}(x_i \cap y_j)}{\text{area}(x_i)},\ y_j \in \tilde{Y}_i$	$[0,1]$	1	Moller et al. (2007); Clinton et al. (2010)
$\text{RAsuper}_{ij} = \frac{\text{area}(x_i \cap y_j)}{\text{area}(y_i)},\ y_j \in \tilde{Y}_i$	$[0,1]$	1	Moller et al. (2007); Clinton et al. (2010)
$\text{PI}_i = \sum_{j=1}^{m} \frac{\text{area}(x_i \cap y_j)^2}{\text{area}(x_i)\text{area}(y_i)},\ y_j \in \tilde{Y}_i$	$[0,1]$	1	Van Coillie et al. (2008)
$\text{Fitness}_{ij} = \frac{\text{area}(x_i) + \text{area}(y_i) - 2\,\text{area}(x_i \cap y_j)}{\text{area}(y_i)},\ x_i \in X'_i$	$[0,\infty)$	0	Costa et al. (2008)
$\text{ED3}_{ij} = \sqrt{\frac{\text{OS3}_{ij}^2 + \text{US3}_{ij}^2)}{2}}$	$[0,1]$	0	Yang et al. (2014)
$\text{F-measure}_{ij}{}^* = \frac{1}{\frac{\alpha}{\text{precision}_{ij}} + \frac{(1-\alpha)}{\text{recall}_{ij}}}$	$[0,1]$	1	van Rijsbergen (1979); Zhang et al. (2015)
$\text{IoU}_{ij} = \frac{\text{area}(x_i \cap y_j)}{\text{area}(x_i \cup y_j)},\ y_j \in Y'{}_i$	$[0,1]$	1	Jaccard (1912); Rezatofighi et al. (2019)
$\text{SimSize}_{ij} = \frac{\min(\text{area}(x_i),\text{area}(y_j))}{\max(\text{area}(x_i),\text{area}(y_j))},\ y_j \in Y^*{}_i$	$[0,1]$	1	Zhan et al. (2005)
$\text{qLoc}_{ij} = \text{dist}(\text{centroid}(x_i), \text{centroid}(y_j)),\ y_j \in Y^*{}_i$	$[0,\infty)$	0	Zhan et al. (2005)
$\text{RPsub}_{ij} = \text{dist}(\text{centroid}(x_i), \text{centroid}(y_j)),\ y_j \in \tilde{Y}_i$	$[0,\infty)$	0	Moller et al. (2007); Clinton et al. (2010)
$\text{RPsuper}_{ij} = \frac{\text{dist}(\text{centroid}(x_i),\text{centroid}(y_j))}{\max_j(\text{dist}(\text{centroid}(x_i),\text{centroid}(y_j)))},\ y_j \in Y^*{}_i$	$[0,1]$	0	Moller et al. (2007); Clinton et al. (2010)
$\text{OI2}_i = \max_j \left(\frac{\text{area}(x_i \cap y_j)}{\text{area}(x_i)} \frac{\text{area}(x_i \cap y_j)}{\text{area}(y_j)} \right),\ y_j \in \tilde{Y}_i$	$[0,1]$	1	Yang et al. (2017)
$\text{Dice}_{ij} = \frac{2\,\text{area}(x_i \cap y_j)}{\text{area}(x_i) + \text{area}(y_j)},\ y_j \in Y'{}_i$	$[0,1]$	1	Dice (1945)

* It takes the optional weight argument $\alpha \in [0,1]$ (the default is 0.5).

Metrics can be computed either from scratch using subsets or by combining other metrics. Examples of metrics using subsets include: Oversegmentation (OS), Undersegmentation (US), Area Fit Index (AFI), Quality Rate (QR), Precision, Recall, Undermerging (UMerging), Overmerging (OMerging), Match (M), Evaluation measure (E), Relative area (RAsub and RAsuper), Purity Index (PI), and Fitness Function (Fitness). The metrics computed by combining other metrics include: Index D (D), Euclidean Distance (ED3), and F-measure (F_measure). Some of these metrics are not intended to be summarized such as Relative position (RPsub and RPsuper).

3 The segmetric package

Installation

The stable release of **segmetric** package can be installed from CRAN, using:

```
install.packages("segmetric")
```

Computing metrics

segmetric depends on the **sf** package (Pebesma, 2018) to open and manipulate geographic vector data sets. **sf** is an implementation of a standard issued by the Open Geospatial Consortium (OGC, 2011), which was further formalized in ISO 19125-1 (2004). This standard defines a common way to store and access spatial data in the context of geographic information systems.

To start with **segmetric**, users should create a `segmetric` object using `sm_read(ref_sf,seg_sf)` passing to it a reference spatial data set and a segmentation spatial data set. The parameters `ref_sf` and `seg_sf` should be either `sf` objects or paths to a supported file vector format (e.g., '`shapefile`').

```
library(segmetric)

# load example data sets
data("sample_ref_sf", package = "segmetric")
data("sample_seg_sf", package = "segmetric")

# create a segmetric object
m <- sm_read(ref_sf = sample_ref_sf, seg_sf = sample_seg_sf)
```

To compute a metric, users should run the function `sm_compute(m,metric_id,...)`, where `m` is a `segmetric` object and `metric_id` is the identification of a metric in **segmetric**. Any extra parameter necessary to compute metrics can be informed using the `ellipsis` parameter. The list of available metrics can be obtained using `sm_list_metrics()` which returns a `character` vector listing all registered metrics.

The `sm_compute()` function can compute a set of metrics by passing a vector of values to the `metric_id` parameter or making a sequence of function calls using a `pipe` operator. The two examples below produce equivalent results:

```
# compute three metrics
sm_compute(m, c("AFI", "OS1", "US1"))

# compute the same three metrics as above
sm_compute(m, "AFI") %>%
  sm_compute("OS1") %>%
  sm_compute("US1")
```

Most metrics are computed by feature (i.e., by reference or segment). To summarize the values of a set of metrics, users can run the function `summary(object,...)`, which computes aggregated values for the metrics returned by `sm_compute()`.

```
# compute three metrics
sm_compute(m, c("AFI", "OS1", "US1")) %>%
  summary()
```

Once created, a `segmetric` object stores in the cache every computed subset. Further subset requests are retrieved from the cache, speeding up the computation.

How to extend segmetric

The **segmetric** package is extensible by providing functions to implement new metrics. To implement a new metric, users can use `sm_new_metric()` to create a new metric object and register it using `sm_reg_metric()` function. Users can type `?sm_reg_metric()` to find more details on how new metrics can be implemented. The following example implements the Jaccard index (Jaccard, 1912), also known as Intersection over Union (*IoU*) (Rezatofighi et al., 2019), which is defined between 0 and 1 (optimal):

```
# register 'IoU' metric
sm_reg_metric(
    metric_id = "IoU",
    entry = sm_new_metric(
        fn = function(m, s, ...) {
            # m is the metric object, s is the subset
            #  for IoU, s is equivalent to sm_yprime(m)
            sm_area(s) / sm_area(sm_subset_union(s))
        },
        fn_subset = sm_yprime,
        name = "Intersection over Union",
        optimal = 1,
        description = "Values from 0 to 1 (optimal)",
        reference = "Jaccard (1912); Rezatofighi et al. (2019)"
    )
)

# describes the 'IoU' metric
sm_desc_metric("IoU")
#> * IoU (Intersection over Union)
#>   Values from 0 to 1 (optimal)
#>   reference: Jaccard (1912); Rezatofighi et al. (2019)
```

Contributions to the package are welcome at GitHub [2] and more details on how to contribute can be found in segmetric home-page at https://michellepicoli.github.io/segmetric.

4 Package segmetric in action

The specific steps involved in a segmentation workflow can vary depending on researcher goals, characteristics of the input data, and the task requirements. In general, a segmentation workflow typically includes the steps in Figure 2. First, researchers working with segmentation data need to obtain satellite images and preprocess them via methods such as radiometric and geometric corrections, image mosaicking, cloud masking, indices computation, and texture extraction. Second, a segmentation method is used to obtain the segments. Typically, researchers can use supervised and unsupervised machine learning methods such as convolutional neural network (Fukushima, 1980), U-Net (Ronneberger et al., 2015), multi-resolution segmentation (Baatz and Schape, 2000), and watershed segmentation (Beucher, 1992). In this step, the segments can be stored in a vector format. Finally, the accuracy of the segmentation can be assessed by supervised quality metrics. Using reference data, researchers compute metrics to evaluate the segmentation. The last two steps may be repeatedly iterated until the desired level of accuracy is reached.

In the following section, we demonstrate an application of the **segmetric** package to assess several segmentation parameters and guide users to select the most accurate one.

Data

In agriculture studies, mapping characteristics such as the size and number of fields can provide information about productivity and other important variables such as food security, socioeconomic status, and environmental status. To demonstrate **segmetric**, we used data on the Luís Eduardo Magalhães (LEM) municipality, west of Bahia state, Brazil. This municipality belongs to the Brazilian agricultural frontier known as MATOPIBA, which includes the states of Maranhão (MA), Tocantins (TO), Piauí (PI), and Bahia (BA) (Figure 3).

We used three PlanetScope images acquired on Feb 18, 2020, with a 3.7-meter resolution and four spectral bands (blue, green, red, and near-infrared). Radiometric and geometric corrections were

[2] https://github.com/michellepicoli/segmetric

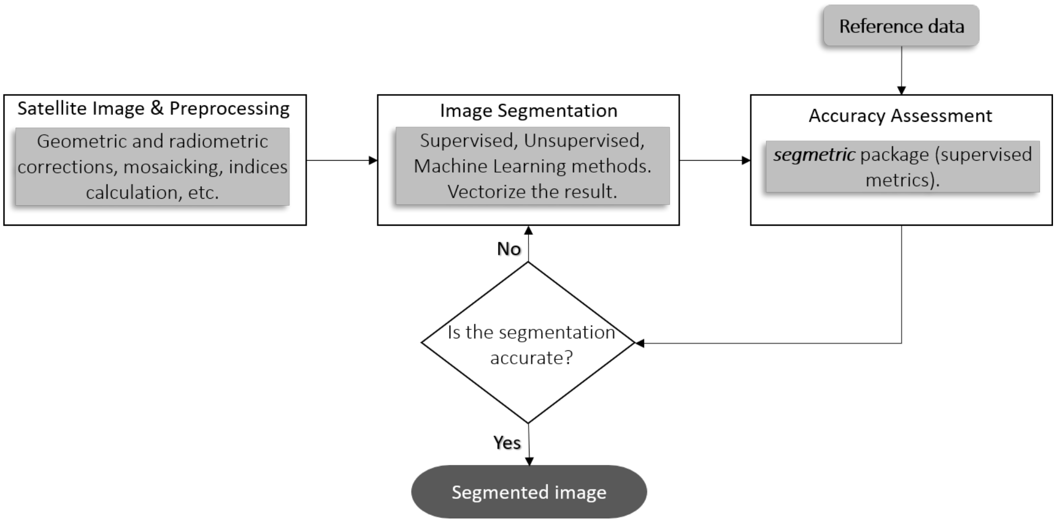

Reference data
Satellite Image & Preprocessing
Geometric and radiometric corrections, mosaicking, indices calculation, etc.
Image Segmentation
Supervised, Unsupervised, Machine Learning methods. Vectorize the result.
Accuracy Assessment
segmetric package (supervised metrics).
No
Is the segmentation accurate?
Yes
Segmented image

Figure 2: General steps of segmentation workflow.

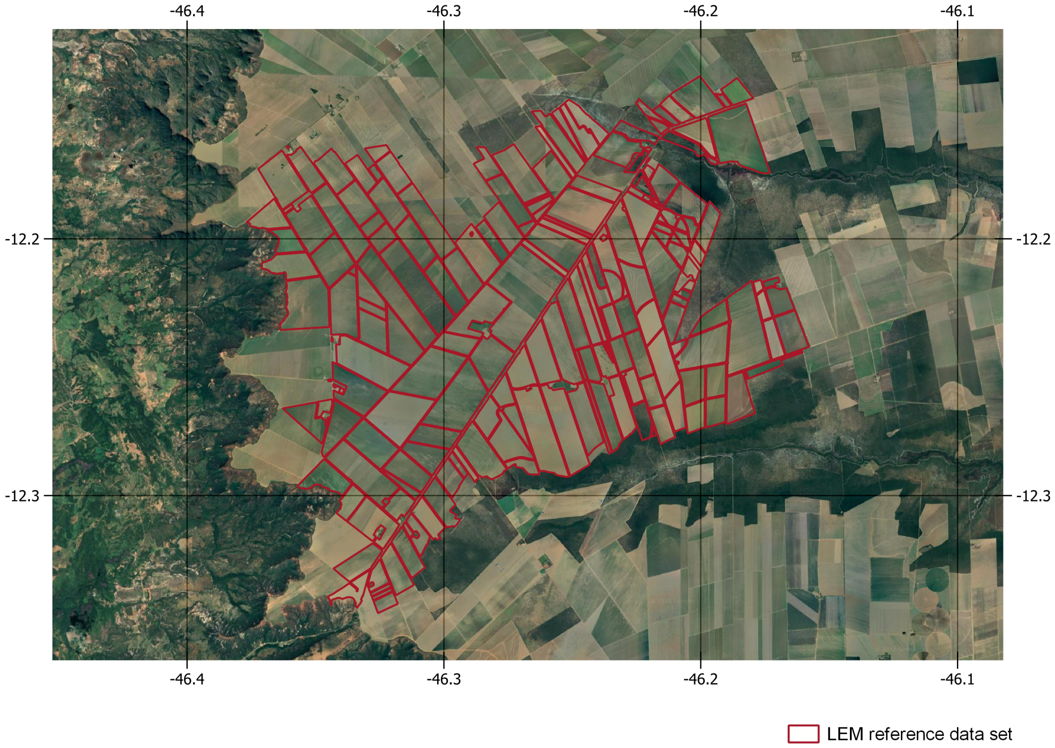


Figure 3: Study area in Luís Eduardo Magalhães municipality, west of Bahia state, Brazil (Google Earth imagery). Reference data (in red) was provided by Oldoni et al. (2020).

applied to the image (level 3B) (Planet Team, 2017). The images were in the same projection (UTM zone 23S) and we mosaicked them.

We segmented the image applying a multi-resolution segmentation approach (Baatz and Schape, 2000). We tested four scale parameters (SP) to segment the image: 200, 500, 800, and 1000; shape parameter: 0.9; and compactness: 0.1. The resulting polygons were simplified using the Douglas-Peucker algorithm (Douglas and Peucker, 1973) (distance parameter: 10 meters) in QGIS software (version 3.22.2). The Self-intersections were removed using SAGA's Polygon Self-Intersection tool (version 7.8.2). The final segmentation set is composed of polygons intersecting the reference data with an area-perimeter ratio above 25. The segmentation results are provided as part of the **segmetric** package.

The reference data set (ref_sf), provided by Oldoni et al. (2020), was collected in two fieldwork campaigns in March and August 2020. Oldoni et al. (2020) draw the field boundaries in-situ on top of images Sentinel-2, with a spatial resolution of 10 meters. **segmetric** includes only a portion of this data set. The spatial data sets can be loaded into R using **sf** objects. To create a **segmetric** object, use function `sm_read()`:

```
library(segmetric)

# load data sets
data("ref_sf", package = "segmetric")
data("seg200_sf", package = "segmetric")
data("seg500_sf", package = "segmetric")
data("seg800_sf", package = "segmetric")
data("seg1000_sf", package = "segmetric")

# create a segmetric object
m200 <- sm_read(ref_sf = ref_sf, seg_sf = seg200_sf)
m500 <- sm_read(ref_sf = ref_sf, seg_sf = seg500_sf)
m800 <- sm_read(ref_sf = ref_sf, seg_sf = seg800_sf)
m1000 <- sm_read(ref_sf = ref_sf, seg_sf = seg1000_sf)
```

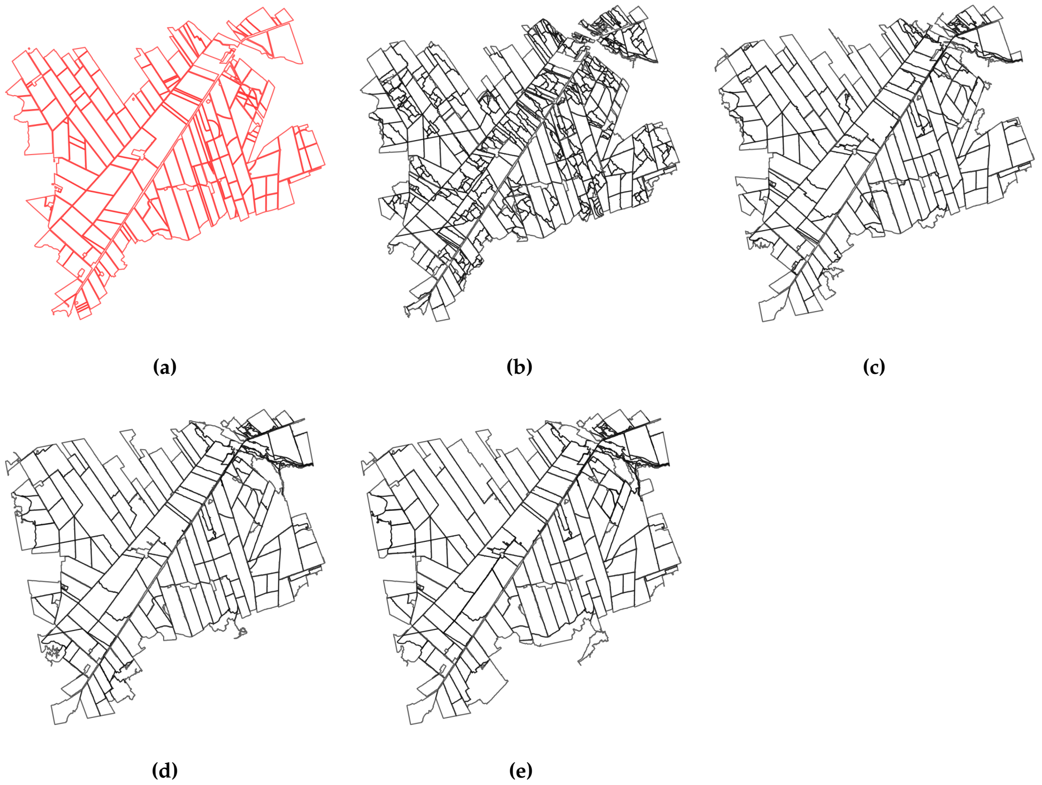
Analysis

This analysis assesses four different segmentations with different Scale Parameters (SP) to verify which one fits better with the reference polygons. First, we visualize the reference polygons and the four segmentations individually using the plot() function (Figure 4).

```
# plot layers
plot(m200, layers = "ref_sf", plot_centroids = FALSE)
plot(m200, layers = "seg_sf", plot_centroids = FALSE)
plot(m500, layers = "seg_sf", plot_centroids = FALSE)
plot(m800, layers = "seg_sf", plot_centroids = FALSE)
plot(m1000, layers = "seg_sf", plot_centroids = FALSE)
```

The metrics available in the package can be consulted using the function `sm_list_metrics()`. In this example, the metrics chosen to evaluate the accuracy of the segmentations and verify the best value of the scale parameter were: Area Fit Index (Carleer et al., 2005), F-measure (van Rijsbergen, 1979) (Zhang et al., 2015), Quality Rate (Weidner, 2008) (Clinton et al., 2010), Oversegmentation (Clinton et al., 2010), and Undersegmentation (Clinton et al., 2010).

```
# compute all metrics
metrics <- c("QR", "F_measure", "IoU", "M", "OS2", "US2")
m200 <- sm_compute(m200, metrics)
m500 <- sm_compute(m500, metrics)
m800 <- sm_compute(m800, metrics)
m1000 <- sm_compute(m1000, metrics)

# results
summary(m200)
#>        QR F_measure       IoU         M       OS2       US2
#> 0.7394817 0.6988555 0.4988198 0.6569973 0.2948025 0.2585708

summary(m500)
#>         QR   F_measure        IoU          M        OS2        US2
```



(a) (b) (c) (d) (e)

Figure 4: (a) reference polygons; (b) segmentation using SP = 200; (c) segmentation using SP = 500; (d) segmentation using SP = 800; (e) segmentation using SP = 1000.

```
#> 0.50380348 0.80671198 0.56837519 0.70140431 0.07982693 0.37207120

summary(m800)
#>         QR  F_measure        IoU          M        OS2        US2
#> 0.47487615 0.78764418 0.54923433 0.68297970 0.04300207 0.43014287


summary(m1000)
#>         QR  F_measure        IoU          M        OS2        US2
#> 0.50311268 0.75742922 0.51745883 0.65548524 0.03679037 0.46524463
```

The computed metrics are presented in Table 2; the optimal value of QR, OS2, and US2 is 0, and 1 for F-measure, M, and IoU. These results indicate that segmentation using SP equal to 200 had the highest oversegmentation while using SP equal to 1000 had the highest undersegmentation. Observing the metrics F-measure, IoU, and M, we conclude that the best SP is 500.

Users must pay attention to which metric better fits their goals of accuracy assessment. For more information, we suggest the user consult comparative studies dedicated to geometric metrics such as Clinton et al. (2010), Räsänen et al. (2013), Yang et al. (2015), Costa et al. (2018), and Jozdani and Chen (2020).

Table 2: Accuracy metrics of Quality Rate (QR), F-measure, Intersection over Union (IoU), Match (M), Oversegmentation (OS2), and Undersegmentation (US2) for four segmentations with different Scale Parameters (SP).

	QR	F_measure	IoU	M	OS2	US2
seg 200	0.739	0.699	0.499	0.657	0.295	0.259
seg 500	0.504	0.807	0.568	0.701	0.080	0.372
seg 800	0.475	0.788	0.549	0.683	0.043	0.430
seg 1000	0.503	0.757	0.517	0.655	0.037	0.465

The **segmetric** package allows users to visualize subsets used to compute metrics. The example in Figure 5 shows the results of the function to plot the subset Y_tilde over the reference and the segmentation polygons (SP = 500). This allows analyzing the overlap between the reference and segmentation polygons visually.

```
plot(
    x = m500,
    type = "subset",
    subset_id = "Y_tilde",
    plot_centroids = FALSE,
    plot_legend = TRUE,
    extent = sm_seg(m500)
)
```

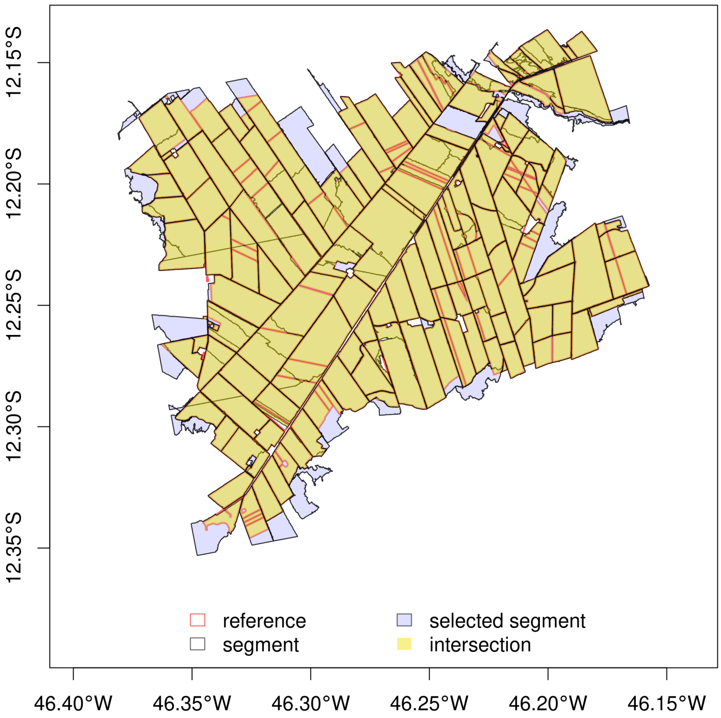


Figure 5: Overlapping between reference polygons and segmentation objects (SP = 500).

It is also possible to visualize the metrics for each segment in choropleth maps using the function:

```
plot(
    x = m500,
    type = "choropleth",
    metric_id = c("QR", "IoU", "M", "OS2", "US2"),
    break_style = "jenks",
    choropleth_palette = "RdYlBu",
    plot_centroids = FALSE
)
```

Legend bar of choropleth maps are generated automatically, and users can further customize it with options such as the number of breaks and the palette. The legend consistently uses the same color for the optimal metric value (for example, in Figure 6, blue is better while red is worse), except for those metrics in which the optimal value is in the middle of the color scale (e.g., AFI). The size and number of intervals in each color scale change accordingly to the metric values present in the data set. Users can choose the method to compute the intervals. To check available options use ?plot.segmetric and see break_style parameter.

Figure 6 presents the spatialized results of the calculated metrics. The F-measure metric was not plotted because it is a global metric with a single value for all objects. Figures 6a, d, and e show the similarity between the QR, OS, and US metrics results, for which the ideal value is zero. In the three

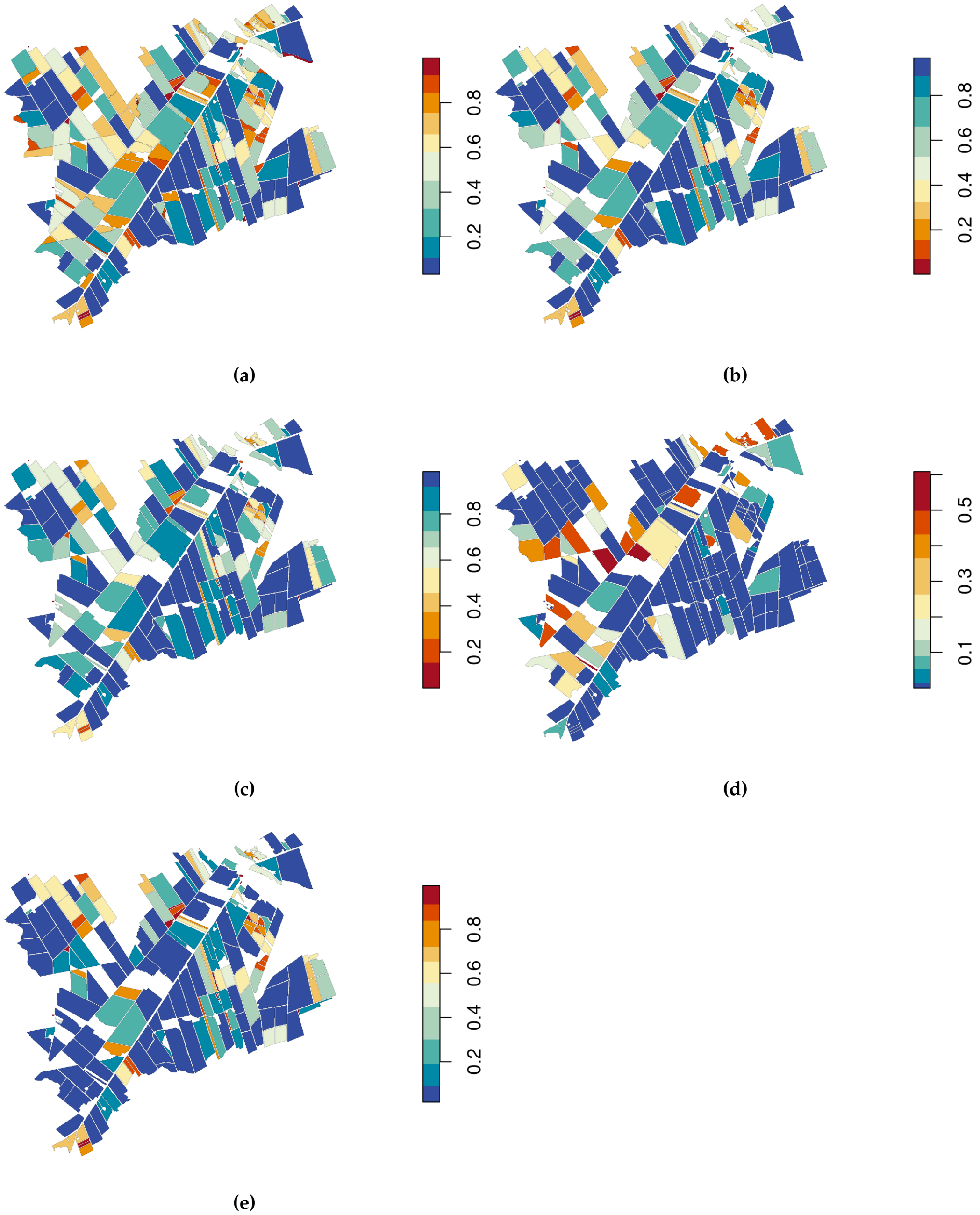


Figure 6: Spatial distribution of the metrics: (a) Quality Rate, (b) Intersection over Union, (c) Match, (d) Oversegmentation, and (e) Undersegmentation.

plots of these metrics, it is noted that the objects with the best results (close to zero) are located in the southeast part of the study area. The IoU and M metric maps (Figure 6b and c), for which the ideal value is 1, are also similar. We also observed that for these two metrics, the objects located in the southwest part of the study area have values close to 1. The figure shows differences in the number of objects plotted in each of the metrics, as the subsets used to calculate each of the metrics are different.

5 Summary

The segmetric package provides 28 metrics that can be used to evaluate and compare the results of segmentation methods. The package also offers innovative visualization options to assist qualitative spatial assessment, allowing diagnostics of the quality, issues, and potential biases of the segmentation. Plotting the segmented objects along their reference polygons and spatially visualizing the metrics may help users to evaluate and improve segmentation procedures, select segmentation parameters, and decide on adequate validation metrics.

To the extent of our knowledge, **segmetric** is the first available package in R that provides several supervised metrics based on reference polygons. **segmetric** also enables users to implement new metrics. In the future, we plan to add more supervised metrics and other ways to visualize metrics, and to use parallel processing to speed up computations.

Acknowledgments

This research was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement No 677140 MIDLAND); the Amazon Fund through the financial collaboration of the Brazilian Development Bank (BNDES) and the Foundation for Science, Technology and Space Applications (FUNCATE) (Process 17.2.0536.1); and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) (Process 350820/2022-8).

Bibliography

A. M. Azevedo. *ExpImage: Tool For Analysis of Images in Experiments*, 2022. URL https://cran.r-project.org/web/packages/ExpImage. R package version 0.6.0. [p159]

M. Baatz and A. Schape. Multiresolution Segmentation: An Optimization Approach for High Quality Multi-Scale Image Segmentation. In J. Strobl, T. Blaschke, and G. Griesbner, editors, *Proceedings of Angewandte Geographische Informationsverarbeitung, XII*, pages 12–23, Salzburg, 2000. Herbert Wichmann Verlag. [p163, 165]

S. Beucher. The watershed transformation applied to image segmentation. *Scanning microscopy*, 1992 (6):28, 1992. [p163]

A. Carleer, O. Debeir, and E. Wolff. Assessment of Very High Spatial Resolution Satellite Image Segmentations. *Photogrammetric Engineering & Remote Sensing*, 71(11):1285–1294, 2005. ISSN 0099-1112. URL https://doi.org/10.14358/PERS.71.11.1285. [p161, 165]

N. Clinton, A. Holt, J. Scarborough, L. Yan, and P. Gong. Accuracy Assessment Measures for Object-based Image Segmentation Goodness. *Photogrammetric Engineering & Remote Sensing*, 76(3):289–299, 2010. ISSN 0099-1112. URL https://doi.org/10.14358/PERS.76.3.289. [p159, 160, 161, 165, 166]

G. Costa, R. Feitosa, T. Cazes, and B. Feijó. *Genetic adaptation of segmentation parameters*, pages 679–695. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-77058-9. URL https://doi.org/10.1007/978-3-540-77058-9_37. [p161]

H. Costa, G. M. Foody, and D. S. Boyd. Supervised methods of image segmentation accuracy assessment in land cover mapping. *Remote Sensing of Environment*, 205:338–351, 2018. ISSN 0034-4257. URL https://doi.org/10.1016/j.rse.2017.11.024. [p159, 160, 166]

L. R. Dice. Measures of the Amount of Ecologic Association Between Species. *Ecology*, 26(3):297–302, 1945. URL https://doi.org/10.2307/1932409. [p161]

D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, dec 1973. ISSN 0317-7173. URL https://doi.org/10.3138/FM57-6770-U75U-7727. [p165]

R. Feitosa, R. Ferreira, C. Almeida, F. Camargo, and G. Costa. Similarity metrics for genetic adaptation of segmentation parameters. In *3rd International Conference on Geographic Object-Based Image Analysis (GEOBIA 2010)*, volume 29, Ghent, 2010. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. [p161]

K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980. [p163]

M. D. Hossain and D. Chen. Segmentation for object-based image analysis (obia): A review of algorithms and challenges from remote sensing perspective. *ISPRS Journal of Photogrammetry and Remote Sensing*, 150:115–134, 2019. ISSN 0924-2716. URL https://doi.org/https://doi.org/10.1016/j.isprsjprs.2019.02.009. [p159]

ISO 19125-1. Geographic information - Simple feature access - Part 1: Common architecture. Technical report, International Standard Organization, 2004. URL https://www.iso.org/standard/40114.html. [p162]

P. Jaccard. The Distribution of the Flora in the Alpine Zone. *New Phytologist*, 11(2):37–50, feb 1912. ISSN 0028-646X. URL https://doi.org/10.1111/j.1469-8137.1912.tb05611.x. [p161, 163]

L. Janssen and M. Molenaar. Terrain objects, their dynamics and their monitoring by the integration of GIS and remote sensing. *IEEE Transactions on Geoscience and Remote Sensing*, 33(3):749–758, 1995. URL https://doi.org/10.1109/36.387590. [p161]

S. Jozdani and D. Chen. On the versatility of popular and recently proposed supervised evaluation metrics for segmentation quality of remotely sensed images: An experimental case study of building extraction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 160(November 2019):275–290, feb 2020. ISSN 09242716. URL https://doi.org/10.1016/j.isprsjprs.2020.01.002. [p160, 166]

I. Kotaridis and M. Lazaridou. Remote sensing image segmentation advances: A meta-analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, 173:309–322, mar 2021. ISSN 09242716. URL https://doi.org/10.1016/j.isprsjprs.2021.01.020. [p159]

M. D. Levine and A. M. Nazif. *An experimental rule based for testing low level segmentation strategies*. Academic Press, New York, 1982. [p161]

A. Lucieer and A. Stein. Existential uncertainty of spatial objects segmented from satellite sensor imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 40(11):2518–2521, 2002. URL https://doi.org/10.1109/TGRS.2002.805072. [p161]

M. Moller, L. Lymburner, and M. Volk. The comparison index: A tool for assessing the accuracy of image segmentation. *International Journal of Applied Earth Observation and Geoinformation*, 9 (3):311–321, 2007. ISSN 0303-2434. URL https://doi.org/doi.org/10.1016/j.jag.2006.10.002. [p161]

L. Mouselimis. *SuperpixelImageSegmentation: Image Segmentation using Superpixels, Affinity Propagation and Kmeans Clustering*, 2022a. URL https://CRAN.R-project.org/package=SuperpixelImageSegmentation. R package version 1.0.5. [p159]

L. Mouselimis. *OpenImageR: An Image Processing Toolkit*, 2022b. URL https://CRAN.R-project.org/package=OpenImageR. R package version 1.2.7. [p159]

J. Niedballa, J. Axtner, T. F. Döbert, A. Tilker, A. Nguyen, S. T. Wong, C. Fiderer, M. Heurich, and A. Wilting. imageseg: An r package for deep learning-based image segmentation. *Methods in Ecology and Evolution*, 13(11):2363–2371, 2022. URL https://doi.org/10.1111/2041-210X.13984. [p159]

OGC. Simple Feature Access-Part 1: Common Architecture. Technical report, Open Geospatial Consortium, 2011. URL http://www.opengeospatial.org/standards/sfa. [p162]

L. V. Oldoni, I. D. Sanches, M. C. A. Picoli, R. M. Covre, and J. G. Fronza. LEM+ dataset: For agricultural remote sensing applications. *Data in Brief*, 33:106553, 2020. ISSN 2352-3409. URL https://doi.org/10.1016/j.dib.2020.106553. [p164, 165]

E. Pebesma. Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal*, 10(1): 439–446, 2018. URL https://doi.org/10.32614/RJ-2018-009. [p162]

C. Persello and L. Bruzzone. A novel protocol for accuracy assessment in classification of very high resolution images. *IEEE Transactions on Geoscience and Remote Sensing*, 48(3):1232–1244, 2010. URL https://doi.org/10.1109/TGRS.2009.2029570. [p161]

Planet Team. Planet Application Program Interface: In Space for Life on Earth, 2017. URL https://api.planet.com. [p165]

A. Räsänen, A. Rusanen, M. Kuitunen, and A. Lensu. What makes segmentation good? A case study in boreal forest habitat mapping. *International Journal of Remote Sensing*, 34(23):8603–8627, 2013. URL https://doi.org/10.1080/01431161.2013.845318. [p166]

H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019. URL https://doi.org/10.1109/CVPR.2019.00075. [p161, 163]

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer International Publishing, 2015. [p163]

F. Van Coillie, L. Verbeke, and R. De Wulf. *Semi-automated forest stand delineation using wavelet based segmentation of very high resolution optical imagery*, pages 237–256. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-77058-9. URL https://doi.org/10.1007/978-3-540-77058-9_13. [p161]

C. van Rijsbergen. *Information Retrieval*. Butterworths, 2nd edition, 1979. [p161, 165]

U. Weidner. Contribution to the assessment of segmentation quality for remote sensing applications. In *XXI International Society for Photogrammetry and Remote Sensing Congress (XXI ISPRS 2008)*, pages 479–484, Beijing, 2008. International Society for Photogrammetry and Remote Sensing. [p161, 165]

J. Wijffels. *image.Otsu: Otsu's Image Segmentation Method*, 2020. URL https://CRAN.R-project.org/package=image.Otsu. R package version 0.1. [p159]

J. Yang, P. Li, and Y. He. A multi-band approach to unsupervised scale parameter selection for multi-scale image segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 94:13–24, 2014. ISSN 0924-2716. URL https://doi.org/10.1016/j.isprsjprs.2014.04.008. [p161]

J. Yang, Y. He, J. Caspersen, and T. Jones. A discrepancy measure for segmentation evaluation from the perspective of object recognition. *ISPRS Journal of Photogrammetry and Remote Sensing*, 101:186–192, 2015. ISSN 0924-2716. URL https://doi.org/10.1016/j.isprsjprs.2014.12.015. [p166]

J. Yang, Y. He, J. P. Caspersen, and T. A. Jones. Delineating Individual Tree Crowns in an Uneven-Aged, Mixed Broadleaf Forest Using Multispectral Watershed Segmentation and Multiscale Fitting. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(4):1390–1401, 2017. URL https://doi.org/10.1109/JSTARS.2016.2638822. [p161]

Q. Zhan, M. Molenaar, K. Tempfli, and W. Shi. Quality assessment for geo-spatial objects derived from remotely sensed data. *International Journal of Remote Sensing*, 26(14):2953–2974, 2005. URL https://doi.org/10.1080/01431160500057764. [p161]

X. Zhang, X. Feng, P. Xiao, G. He, and L. Zhu. Segmentation quality evaluation using region-based precision and recall measures for remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 102:73–84, 2015. ISSN 0924-2716. URL https://doi.org/10.1016/j.isprsjprs.2015.01.009. [p161, 165]

Rolf Simoes
National Institute for Space Research (INPE)
Avenida dos Astronautas, 1758, 12227-010, Sao Jose dos Campos
Brazil
ORCiD: https://orcid.org/0000-0003-0953-4132
rolf.simoes@inpe.br

Alber Sanchez
National Institute for Space Research (INPE)
Avenida dos Astronautas, 1758, 12227-010, Sao Jose dos Campos
Brazil
ORCiD: https://orcid.org/0000-0001-7966-2880
alber.ipia@inpe.br

Michelle C. A. Picoli
Earth and Life Institute, UCLouvain
Place Louis Pasteur 3, 1348, Louvain-la-Neuve
Belgium
ORCiD: https://orcid.org/0000-0001-9855-2046
mipicoli@gmail.com

Patrick Meyfroidt
Earth and Life Institute, UCLouvain
Place Louis Pasteur 3, 1348, Louvain-la-Neuve

Belgium
ORCiD: https://orcid.org/0000-0002-1047-9794
patrick.meyfroidt@uclouvain.be

Onlineforecast: An R Package for Adaptive and Recursive Forecasting

by Peder Bacher, Hjörleifur G. Bergsteinsson, Linde Frölke, Mikkel L. Sørensen, Julian Lemos-Vinasco, Jon Liisberg, Jan Kloppenborg Møller, Henrik Aalborg Nielsen and Henrik Madsen

Abstract Systems that rely on forecasts to make decisions, e.g. control or energy trading systems, require frequent updates of the forecasts. Usually, the forecasts are updated whenever new observations become available, hence in an online setting. We present the R package onlineforecast that provides a generalized setup of data and models for online forecasting. It has functionality for time-adaptive fitting of dynamical and non-linear models. The setup is tailored to enable the effective use of forecasts as model inputs, e.g. numerical weather forecast. Users can create new models for their particular applications and run models in an operational setting. The package also allows users to easily replace parts of the setup, e.g. using new methods for estimation. The package comes with comprehensive vignettes and examples of online forecasting applications in energy systems, but can easily be applied for online forecasting in all fields.

1 Introduction

Time series analysis and forecasting are indispensable to numerous applied fields such as business, finance, science and engineering (Cryer and Chan, 2008). Time series analysis is the process of statistical modelling of time series, i.e. data which is sampled at different points in time over a period – often with a constant increment between the time-points, i.e. equidistant. Classical time series models for a single equidistant time series use past values of the response variable (model output) as the predictors (inputs). In this way, appropriate models describing the inherent auto-correlation structure of the time series can be realized. Examples of these models include exponential smoothing (e.g. Holt-Winters), AutoRegressive (AR), Moving Average (MA), and the combination of the latter two known as ARMA models. When multiple correlated time series are available, they can be used as simultaneous model inputs to improve the forecast. They are then called exogenous variables and the classical model becomes an ARMAX – hence the X indicates that exogenous input variables are included. ARMAX models are optimal for forecasting the output of linear time invariant (LTI) systems, however for most forecasting applications models that can handle non-linear systems are needed. A wide range of techniques for modelling non-linear systems exists, either based on input transformations or local fitting methods. The onlineforecast package implements an advanced model setup for modelling and forecasting the output of non-linear time varying systems. The setup was developed for applications such as forecasting wind power (Nielsen et al., 2002) and thermal loads in district heating (Nielsen and Madsen, 2006). The significance of the package is in the “online” term, indicating that at each sampling point the model parameter estimates are updated in an effective way for generating multi-step forecasts.

The use of ARMAX models and their variations for forecasting is still widespread (De Gooijer and Hyndman, 2006), especially for energy systems due to the high dependency between variables such as weather, load, renewable generation, and periodic phenomena. Load forecasting is an obvious example. A nice overview of electric load forecasting is given by Alfares and Nazeeruddin (2002) and Hong and Fan (2016), and for heat load by Dotzauer (2002) who demonstrates the dependency between the response variable, heat load, and the predictor – ambient temperature – using a piecewise linear function. It is also proposed to model the daily and weekly diurnal using hours of the week as inputs.

Bacher et al. (2009) demonstrated that solar power forecasting (Kleissl, 2013) can be improved by moving from a standard AR model to an AR model with an exogenous input (ARX), specifically by using numerical weather predictions (NWPs) as the exogenous inputs. The ARX model uses past observations and NWPs of global irradiance to forecast the power production from PV systems and the ARX model obtains higher accuracy than the AR model. Bacher et al. (2013) identified exogenous variables that are suitable for forecasting the heat load of a building, via similar models.

Energy systems are time-varying systems as they usually change over time due to wear and contamination, like dirt on solar panels or changes in usage. For example, with new tenants in a house, the dependency between heat load and other variables, such as calendar time and temperature, changes. Therefore, a forecast model needs to adapt: the model coefficients are not optimal if they are constant, they need to be updated and allowed to change over time. The Recursive Least Square (RLS) method provides a recursive estimation scheme for the coefficients in regression models, where they are updated at each step, when new data becomes available. A forgetting factor can be introduced to RLS to allow for the control of how fast the coefficients can change over time – this is referred to as adaptive recursive estimation, with exponential forgetting, in linear regression and autoregressive models. The method is described by Ljung and Söderström (1983), for advances that has been made since then see e.g. (Engel et al., 2004).

The objective of the onlineforecast package is to make it easy to set up and optimize non-linear models for generating online multi-step forecasts. The package contains functionalities not directly available elsewhere, such as:

- Use of forecasts, e.g. NWPs, as input to multi-step forecast models.
- Optimal tuning of models for multi-step horizons.
- Recursive estimation for tracking time-varying systems.

The package provides a framework for handling data and setting up models, which makes it easy to apply it in a wide range of forecasting applications.

Time series modelling and forecasting in R

A wide range of existing software for time series forecasting is currently available (Chatfield and Xing, 2019; Siebert et al., 2021). Below, an overview of the currently most relevant R packages for forecasting is given – generally, the same functionalities are available in Python packages.

Classical ARMAX models can be fitted with the `arima()` function from the stats package and the `Arima()` function from the forecast package (Hyndman and Khandakar, 2008) provides automatic model selection with `arima()`. R Packages like marima (Spliid, 1983), KFAS, sysid and dlm (Petris, 2010) can also be used for fitting ARMAX models. Spliid (1983) proposed a very fast and simple method for parameter estimation in large multivariate ARMAX models with a pseudo-regression method that repeats the regression estimation until it converges. The other packages represent time series and regression models as state-space models and use a Kalman or Bayesian filter to include exogenous variables in the model, and optimally reconstruct and predict the states. Compared to these classical ARMAX models, onlineforecast models offers several advantages, first and foremost the recursive fitting scheme which allows for much faster and adaptive fitting. Furthermore, model coefficients are tuned as a function of the forecast horizon. This optimize the use of multi-step forecasts as models inputs, such functionality is not available for ARMAX models.

State-space modelling is frequently used to describe time series data from a dynamical system, e.g. a falling body, see (Madsen, 2007). The dynamical system can in such cases be written as a system of differential equations or difference equations. State-space models use filter techniques to optimally reconstruct and predict the states, with examples including the Kalman filter, the extended Kalman filter, and other Bayesian filters. This gives the possibility of tracking the coefficients over time, i.e. time-varying parameter estimation. The KFAS package (Helske, 2017) provides state-space modelling, where the observations come from the exponential family, e.g. Gaussian or Poisson. The ctsm-r package provides a framework for identifying and estimating partially observed continuous-discrete time state space models, referred to as grey-box models. This modelling approach bridges the gap between physical and statistical modelling using Stochastic Differential Equations (SDEs) to model the system equations in continuous time and the measurement equations in discrete time. Packages for discrete time state-space modelling are: dlm for Bayesian analysis of dynamic linear models, MARSS and SSsimple for fitting multivariate state-space models. The onlineforecast models are basically fitted using a Kalman filter, as explained in Section Regression, thus existing packages could be applied. However, the use of forecasts as model inputs would be very cumbersome and is made very easy with the onlineforecast setup.

For non-parametric time series models, the number of available packages is growing rapidly. NTS provides simulation, estimation, prediction and identification for non-linear time series data. It also includes threshold autoregressive models (e.g. self-exciting threshold autoregressive models) and neural network estimation. tsDyn provides methods for estimating non-parametric time series models, including neural network estimation. Neural network, deep learning and machine learning methods are available in R. Recurrent neural networks are available in the rnn, the keras and tensorflow packages. Additive time series models, where non-linear trends are fitted with seasonality patterns, are available in prophet. Time adaptive neural networks, i.e. with recursive updating, can be implemented in various ways (Yang et al., 2019), however currently no effective implementation is available.

Some packages can be useful for forecast evaluation, e.g. ForecastTB presented in (Bokde et al., 2020). Packages like forecastML and modeltime (Alexandrov et al., 2020) provide functionality that simplifies the process of multi-step-ahead forecasting with machine learning algorithms. The handling of multi-step-ahead forecasts is also a key feature of the onlineforecast package. The classical time series models, such as ARMAX and Exponential Smoothing models, are mostly optimal for modelling Linear Time Invariant (LTI) systems, however, most systems are not LTI. Furthermore, since a model is always a simplification of reality, optimal multi-step forecasting is often not possible with the classical models, especially when using exogenous inputs. For optimal multi-step ahead forecasting the models must be tuned for each horizon – which is exactly what the onlineforecast package does.

Functionality of onlineforecast

A model is an approximation to the real world, thus it will always be a simplification and can never predict real-world data. One of the main challenges of identifying a good forecast model is to find the most informative input variables and the best structure of the model. The onlineforecast package provides functionality for defining, validating and selecting models in a systematic way.

To introduce the onlineforecast models consider the simplest model with one input. It is the linear model for the k'th horizon

$$Y_{t+k|t} = \beta_{0,k} + \beta_{1,k} u_{t+k|t} + \varepsilon_{t+k|t} \tag{1}$$

where $Y_{t+k|t}$ is the response variable and $u_{t+k|t}$ is the input variable. The coefficients are $\beta_{0,k}$ and $\beta_{1,k}$, note that they are subscripted with k to indicate that they are estimated for individually for every horizon. The error $\varepsilon_{t+k|t}$ represents the difference between the model prediction and the observed value for the k-step horizon. The interpretation of the subscript notation $t+k|t$ on a variable is, that it is the k-step prediction calculated using only available information at time t, usually referred to either "conditional on time t" or "given time t".

The options for estimating the coefficients in the package are either the Least Squares (LS) or Recursive Least Squares (RLS) method. In the LS method, the coefficients are constant, while the in RLS method the coefficients can change over time

$$Y_{t+k|t} = \beta_{0,k,t} + \beta_{1,k,t} u_{t+k|t} + \varepsilon_{t+k|t} \tag{2}$$

as indicated by the subscript t on the coefficients. This allows for tracking changes occurring over time.

The package allows for easy definition of transformations and thus the possibility to fit non-linear models e.g.

$$Y_{t+k|t} = \beta_{0,k,t} + \beta_{1,k,t} f(u_{t+k|t}; \alpha) + \varepsilon_{t+k|t} \tag{3}$$

where the function $f(u_{t+k|t}; \alpha)$ is some non-linear function of the input $u_{t+k|t}$ with parameter α, e.g. a low pass filter on the outdoor temperature to model building heat dynamics. The package sets up tuning of the non-linear function parameters, e.g. if the parameter α determines the degree of low-pass filtering it can be tuned with an optimizer to match the dynamics of the system at hand.

An example of generated forecasts can be appreciated in Figure 1. Hourly forecasts up to 36 steps ahead of heat load in a single building are shown for three consecutive steps. This is the typical structure of forecasts generated with the package. It can be seen how the forecasts change slightly as they are updated in each step, e.g. around 12:00 the second day, hence horizon $k = 23$ in the upper plot, which corresponds to $k = 21$ in the lower plot.

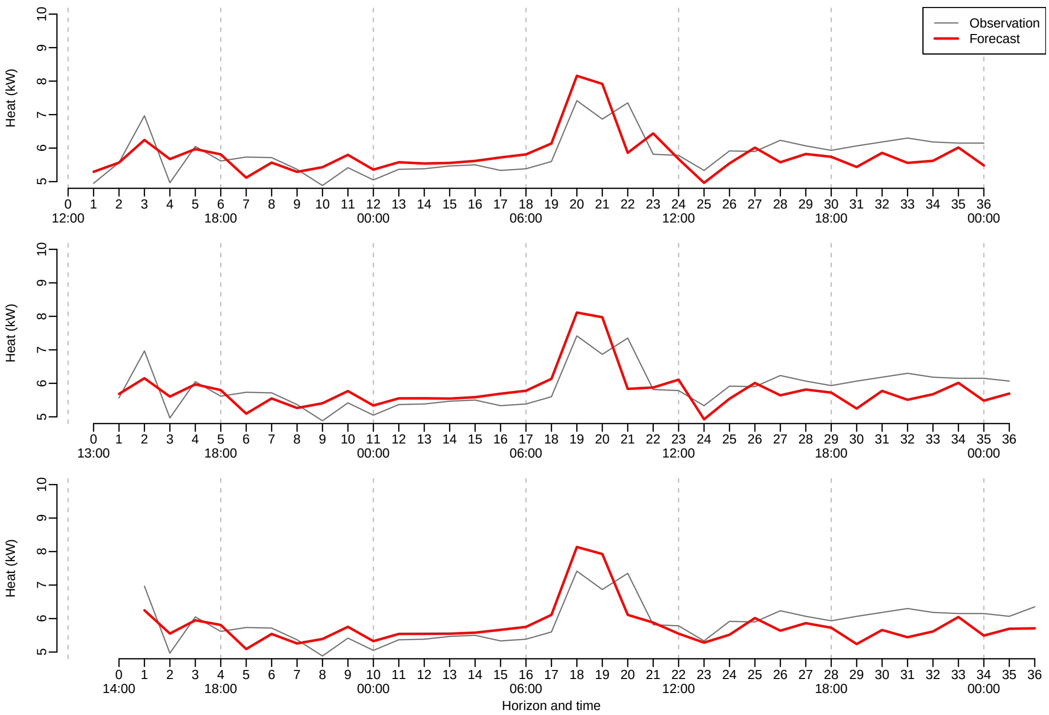


Figure 1: Example of hourly load forecasts at three consecutive time steps. The upper is calculated at 12:00, the middle is calculated at 13:00 and the lower at 14:00. It can be seen how the forecasts change slightly as they are updated in each step, most clearly seen around 12:00 on the second day.

Vignettes

A great way to get hands-on experience with the package is through vignettes. They are available when installing the package and on the website onlineforecasting.org, where also examples of different forecast applications can be found. The package vignettes are:

- setup-data covers how data must be set up. The vignette goes into detail on how observations and model inputs (forecasts) are set up. The vignette also focuses on the importance of aligning forecasts correctly in time.
- setup-and-use-model focus on how to set up a model and use it to generate forecasts.
- model-selection demonstrates how model selection can be carried out.
- forecast-evaluation covers the evaluation of forecasts, and how to use this information to improve a model.
- online-updating demonstrates how to update an operational model when new observations become available. This functionality is not covered in the R examples in the present paper.

Furthermore, one vignette is available only on the website:

- nice-tricks provides some useful tips on how to make the workflow easier with the package.

Paper structure

The paper is structured as follows: In Section Notation and forecast matrices the notation used in the paper and how to set up data is introduced. The core methodology is presented in Section Two-stage modelling procedure and important aspects of forecast modelling are outlined in Section Model selection and validation. In Section Example with R code examples with R code are presented to provide a short hands-on tutorial. The paper ends with a summary and conclusions in Section Discussion and conclusion.

In addition, three appendices are included with the paper. In Appendix Forecast model notation some guidelines on the mathematical notation of forecast models are provided. In Appendix Regression the regression schemes are covered in full detail.

2 Notation and forecast matrices

The notation in this article follows Madsen (2007) as close as possible. All time series considered are equidistantly sampled and the sampling period is normalized to 1. Hence, the time t is simply an integer indexing the value of a variable at time t. The same goes for k which indexes the forecast horizon k steps ahead. In the onlineforecast setup, forecasts are calculated at time t for each horizon up to n_k steps ahead. To achieve the desired notation that can deal with overlapping time series, a two dimensional index is required. The notation used is

$$u_{t+k|t} \tag{4}$$

which translates to: the value of variable u at time $t+k$ *conditional* on the information available at time t. The conditional term is indicated by the bar $|$. Thus, for $k > 0$ this is a forecast available at t and k is the horizon. When writing a forecast model the following convention is used

$$Y_{t+k|t} = \beta_{0,k} + \beta_{1,k} u_{t+k|t} + \varepsilon_{t+k|t} \tag{5}$$

where $Y_{t+k|t}$ is the model output, $\beta_{0,k}$ and $\beta_{1,k}$ are the coefficients and $\varepsilon_{t+k|t}$ with $\mathrm{Var}(\varepsilon_{t+k|t}) = \sigma_k^2$ is the error. The error process and variance σ_k^2 is thus separate for each horizon. Note, that the model is fitted separately for each horizon, so the coefficients take different values for each horizon, and the predictions and errors are separated for each horizon. This was a simplified example, see Appendix Forecast model notation on how to write the full forecast models.

Forecast matrix

A forecast matrix is the format of forecast data in the onlineforecast setup. See examples in the setup-data vignette. Data must have this format in order to be used as model input, and the forecasts are generated in this format. The forecast matrix holds for any time past *the latest available forecast along the row* for the corresponding time

$$\boldsymbol{u}_n = \begin{array}{cccccc} \textbf{k0} & \textbf{k1} & \textbf{k2} & \ldots & \textbf{k}n_k & \rightarrow \textbf{horizon/time} \downarrow \\ u_{1|1} & u_{2|1} & u_{3|1} & \cdots & u_{1+n_k|1} & 1 \\ u_{2|2} & u_{3|2} & u_{4|2} & \cdots & u_{2+n_k|2} & 2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ u_{t-1|t-1} & u_{t|t-1} & u_{t+1|t-1} & \cdots & u_{t-1+n_k|t-1} & t-1 \\ u_{t|t} & u_{t+1|t} & u_{t+2|t} & \cdots & u_{t+n_k|t} & t \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ u_{n|n} & u_{n+1|n} & u_{n+2|n} & \cdots & u_{n+n_k|n} & n \end{array} \tag{6}$$

where

- t is the counter of time for equidistant time points with sampling period of 1 (note that t is not included in the matrix, it is simply the row number).
- n is the number of time points in the matrix. Hence, the data is available and can be used as model input at time $t = n$.
- n_k is the longest forecasting horizon.
- The column names (in R) are indicated above the matrix, they are simply a '**k**' concatenated with the value of k, e.g. n_k in the last column.

Note, that the **k0** column holds values with forecast horizon $k = 0$, which could be real time observations. Usually, only the horizons to be forecasted should be included, hence often **k0** is not needed.

For example with a prediction horizon $n_k = 24$ at $t = 100$, we will have the forecast matrix

$$\begin{array}{c} \\ \boldsymbol{u}_{100} = \end{array} \begin{array}{ccccc} \textbf{k0} & \textbf{k1} & \textbf{k2} & \dots & \textbf{k24} \\ \end{array} \rightarrow \textbf{horizon/time} \downarrow$$

$$\boldsymbol{u}_{100} = \begin{pmatrix} u_{1|1} & u_{2|1} & u_{3|1} & \cdots & u_{25|1} \\ u_{2|2} & u_{3|2} & u_{4|2} & \cdots & u_{26|2} \\ \vdots & \vdots & \vdots & & \vdots \\ u_{99|99} & u_{100|99} & u_{101|99} & \cdots & u_{123|99} \\ u_{100|100} & u_{101|100} & u_{102|100} & \cdots & u_{124|100} \end{pmatrix} \begin{array}{c} 1 \\ 2 \\ \vdots \\ 99 \\ 100 \end{array} \tag{7}$$

In Section Setup of data examples of how data and forecast matrices are set up in R are given.

3 Two-stage modelling procedure

Two-stage modelling procedure is a widespread approach to modelling non-linear functional relations between inputs and outputs (see e.g. Breiman and Friedman (1985) and Weisberg (2005) for direct transformation of predictor variables, and Hastie et al. (2009) for non-parametric transformation techniques). Using transformations allows for fitting complex models with robust and fast estimation techniques. In the first stage, the *transformation stage*, the inputs are mapped by some function – potentially into a higher dimensional space. In the second stage, the *regression stage*, a linear regression model[1] is applied between the transformed inputs and the output.

As an example, a model with two inputs is presented. In this model the transformation stage consists of generating an intercept and mapping the two inputs (they are set up as forecast matrices $\boldsymbol{u}_{1,t}$ and $\boldsymbol{u}_{2,t}$)

$$\text{Intercept:} \quad x_{0,t+k|t} = 1 \tag{8}$$

$$\text{Input 1:} \quad x_{1,t+k|t} = f_1(u_{1,t+k|t}, \boldsymbol{\alpha}_1) \tag{9}$$

$$\text{Input 2:} \quad \boldsymbol{x}_{2,t+k|t} = f_2(u_{2,t+k|t}, \boldsymbol{\alpha}_2) \tag{10}$$

where the f's are transformation functions that map the inputs to regressors. Note, that the intercept is simply a constant passed on to the regression. The transformations result in multiple inputs for the regression – the latter actually as multiple variables indicated by the bold font notation. In the regression stage the linear model

$$Y_{t+k|t} = \beta_{0,k} x_{0,t+k|t} + \beta_{1,k} x_{1,t+k|t} + \boldsymbol{\beta}_{2,k}^T \boldsymbol{x}_{2,t+k|t} + \varepsilon_{t+k|t} \tag{11}$$

is fitted. The regression is carried out separately for each horizon k. Thus, the combined model has:

- An intercept
- Two inputs: $u_{1,t+k|t}$ and $u_{2,t+k|t}$
- Output: $Y_{t+k|t}$
- Transformation functions: f_1 and f_2
- Transformation parameters: $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$
- Regression coefficients: $\beta_{0,k}$, $\beta_{1,k}$ and $\boldsymbol{\beta}_{2,k}$

Some transformation parameters should be optimized for the data at hand, e.g. a low-pass filter coefficient depends on the system dynamics. The same goes for some parameters related to the regression scheme, e.g. the forgetting factor (introduced below). We will refer to them together as ***offline parameters***. The onlineforecast package provides a setup, where the offline parameters can be optimized using a heuristic optimization (e.g. a BFGS quasi-Newton method). The default score, which is minimized, is the Root Mean Square Error (RMSE) of the predictions – hence offline parameters in

[1] In the remaining of the text, when the term "regression" is used it is implicit that it is "linear regression".

the model above, given data from the period, $t = 1, 2, \ldots, n$, are optimized by solving

$$\min_{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2} \quad \frac{1}{n-k} \sum_{t=1}^{n-k} (y_{t+k} - \hat{y}_{t+k|t}(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2))^2 \tag{12}$$

Naturally, other scores can be minimized (e.g. MAE or the Huber psi-function, however the regression schemes should be modified accordingly, which is not trivial).

The regression coefficients are calculated with a closed-form scheme: either with the Least-Squares (LS) or the Recursive Least-Squares (RLS) scheme – in the latter the coefficients are allowed to vary over time. In both schemes the coefficients are gathered in the vector $\boldsymbol{\beta}_k$ and calculated separately for each horizon k. In Appendix Regression both schemes are presented in full detail.

In the LS scheme the coefficients are constant during the entire period. The output vector is $\boldsymbol{y}_{k,n}$ and for a given value of the transformation parameters (i.e. here $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$) the transformed data is calculated and set up in the design matrix $\boldsymbol{X}_{k,n}$. The LS coefficients are then calculated by

$$\hat{\boldsymbol{\beta}}_k = (\boldsymbol{X}_{k,n} \boldsymbol{X}_{k,n})^{-1} \boldsymbol{X}_{k,n} \boldsymbol{y}_{k,n} \tag{13}$$

and the predictions calculated by

$$\hat{\boldsymbol{y}}_{k,n} = \boldsymbol{X}_{k,n} \hat{\boldsymbol{\beta}}_k \tag{14}$$

where $\hat{\boldsymbol{y}}_{k,n} = \begin{bmatrix} \hat{y}_{1+k|1} & \hat{y}_{2+k|2} & \cdots & \hat{y}_{n|n-k} \end{bmatrix}^T$ are the predictions. Note, that for the LS scheme the predictions are "in-sample", since data from the entire period is used for the coefficient calculation.

In the RLS scheme the coefficients are calculated recursively, meaning that they are updated in every time step – the RLS is actually a Kalman filter with the model coefficients in the state vector. At each time t the coefficients are updated by

$$\boldsymbol{R}_{k,t} = \lambda \boldsymbol{R}_{k,t-1} + \boldsymbol{x}_{k,t} \boldsymbol{x}_{k,t}^T \tag{15}$$

$$\hat{\boldsymbol{\beta}}_{k,t} = \hat{\boldsymbol{\beta}}_{k,t-1} + \boldsymbol{R}_{k,t}^{-1} \boldsymbol{x}_{k,t} (y_t - \boldsymbol{x}_{k,t}^T \hat{\boldsymbol{\beta}}_{k,t-1}) \tag{16}$$

and the predictions by

$$\hat{y}_{t+k|t} = \boldsymbol{x}_{t+k|t} \hat{\boldsymbol{\beta}}_{k,t} \tag{17}$$

where $\boldsymbol{x}_{k,t}$ is the data available for horizon k at time t (a row in the design matrix $\boldsymbol{X}_{k,n}$), and $\boldsymbol{x}_{t+k|t}$ is the k'th horizon transformed input forecast available at time t, see the appendix for all details. The coefficients adapts to data over time and the level of adaptivity is controlled by the forgetting factor λ (with value between 0 and 1). When $\lambda = 1$, all past data at t is equally weighted. When $\lambda < 1$, higher weight is put on recent data – the smaller the value, the faster the model adapts to recent data. By optimizing the forgetting factor as an offline parameter, the model adaptivity can be tuned.

An important point to notice is that the offline parameters are always constant for the given period, hence all predictions are essentially "in-sample". However, depending on the regression scheme there is a difference: with the LS scheme the regression coefficients are calculated once using all data, thus the predictions are (fully) "in-sample", where as with the RLS scheme they adapt through the period and the predictions are "out-of-sample" (except for the offline parameters). This makes a difference, since model overfitting is less of a problem when using the RLS scheme.

The typical onlineforecast setup is to optimize the (usually few) offline parameters in an "offline" setting, but calculate the regression coefficients adaptively with the RLS. This has the advantage that the model adapts and tracks the systematic changes in input-output relations, while keeping the setup computationally very effective – updating the coefficients and calculating a forecast at each time t takes few operations. The optimization of offline parameters can be carried out when computational resources are available (e.g. every week for hourly forecasts).

Transformations

In the transformation stage the inputs are mapped using some function as demonstrated above, for more examples, see the setup-and-use-model vignette. The onlineforecast package has functions available for most common use, however it is easy to write and use new functions as they are simply R functions. The main functionality they have to fulfil is to return a forecast matrix (or a list of them), which is also the reason why some of the regular R functions and operators has been extended for the multi-horizon setup, e.g. for splines as explained below. For R examples, see Section Defining a model and the vignettes. The currently available transformation functions are:

- Low-pass filtering, `lp()`: A low-pass filtering for modelling linear dynamics as a simple RC-model. See e.g. Nielsen and Madsen (2006) for further information.
- Basis splines, `bspline()`: Use the `bs` function for calculating regression splines basis functions.
- Periodic basis splines, `pbspline()`: Use the `pbs` function for calculating periodic regression splines basis functions.
- Fourier series, `fs()`: Fourier series as periodic regression basis functions.
- Auto-regressive, `AR()`: For including Auto-Regressive (AR) terms.
- Intercept, `one()`: Generates a forecast matrix of ones, i.e. intercept.

In the following section, the low-pass filtering is shortly described below. For more examples of transformations, see the package vignettes.

The implementation in onlineforecast allows all parameters which are used in some way (except the regression coefficients) to be included in an optimization, using any available optimizer i R. This includes e.g. the RLS forgetting factor, knot points or order of splines, i.e. both continuous and integer variables. This functionality is achieved using a simple syntax as explained in Section Example with R code.

Low-pass filtering

When modelling time series from linear dynamical systems, the classical ARMAX model is often the optimal choice (Madsen, 2007). However, for multi-step forecasting, this is often not the case, especially for longer horizons. In the onlineforecast setup, where the regression model is fitted for each horizon, a "trick" can be used for modelling linear dynamics: simply apply a filter on the input and then use the filtered input in the regression stage. For example, dynamics between ambient air temperature and heat demand are slow due to the thermal mass of the building. Thus they can be modelled using a low-pass filter, see Nielsen and Madsen (2006) for modelling heat load in district heating and Bacher et al. (2013) for forecasting single buildings heat load.

In the package the simple low-pass filter

$$x_{t+k|t} = \frac{(1-a)u_{t+k|t}}{1 - a x_{t-1+k|t-1}} \tag{18}$$

is implemented. The filter coefficient a must take a value between 0 and 1 and should be tuned to match the time constant optimal for the particular data. When the current implemented low-pass filter is applied in the transformation stage on some forecast matrix $\boldsymbol{u}_{t+k|t}$, the filter is applied on each column, i.e. independently for each horizon k. More advanced filters can also be implemented.

4 Model selection and validation

Model selection

In statistics, different model selection procedures are used (Madsen and Thyregod, 2010). Essentially, a backward or a forward selection procedure can be applied, or some combined approach. In the onlineforecast package both procedures are implemented, as well as a combined approach (see the model-selection vignette for examples).

In each step of the selection process two properties of the model can be modified:

- Model inputs: In each step, inputs can either be removed or added.
- Integer offline parameters: In each step integer parameters, such as the number of knot points in a basis spline or the number of harmonics in a Fourier series can be incremented or decremented.

In each step of the process, the offline parameters are first optimized to minimize the score for each modified model (in most cases the appropriate score is the RMSE in Equation (12) summed for selected horizons). Then the scores of the modified models are compared with the score of the currently selected model and the model with the lowest score is selected for the next step. This continues until no further improvement of the score is achieved and the model with the lowest score is selected. It is important to note, that the implemented procedure should only be used with the RLS scheme, with the LS scheme the score is calculated fully in-sample leading to overfitting.

Model validation

The most important aspects of validation of forecast models are discussed in this section (see the forecast-evaluation vignette for examples).

Training and test set

One fundamental problem in data-driven modelling is overfitting. This can easily happen when the model is fitted (trained) and evaluated on the same data. There are essentially two ways of dealing with this: penalize increased model complexity (regularization) or divide the data into a training set and test set (cross-validation) (Tashman, 2000). In most forecasting applications the easiest and most transparent approach is cross-validation – many methods for dividing into sets are possible. In the onlineforecast setup, when a model is fitted recursively using the RLS, only past data is used when calculating the regression coefficients, so there is little need for dividing into a training and a test set.

The offline parameters (like the forgetting factor and low-pass filter coefficients) are optimized on a particular period, hence overfitting is possible, however, typically, only very few parameters compared with the number of observations are estimated – so it is very unlikely that a recursively-fitted model will overfit.

Scoring

The scoring of forecasts can be done in many ways, however in the onlineforecast package, where the conditional mean is estimated and when using the RLS scheme, we recommend choosing the Root Mean Square Error (RMSE) in Equation (12) as the best score to use. When using the LS scheme it can be favourable to include regularization penalty to avoid overfitting, hence AIC or BIC is preferable. One important point when comparing forecasts is to only include the complete cases, i.e. forecasts at time points with no missing values across all horizons and across all evaluated models. A function for easy selection of only complete cases given multiple forecasts is implemented, see the examples in Section Evaluation.

Residual analysis

Analysing the residuals is an important way of validating that a model cannot be further improved or learning how it can improved. The main difference from classical time series model validation, where only the one-step ahead error is examined, is that multiple horizons should be included in the analysis. The two most important ways of analyzing the residuals are to:

- Plot residual time series to find where large forecast errors occur.
- Plot scatter plots of the residuals vs. other variables to see if there are any apparent dependencies not captured by the model.

In order to dig a bit more into the result of a recursive estimation, the regression coefficients can be plotted over time. In this way, it is possible to learn how the relations between the variables in the model evolve over time. If large changes are found in some periods it might be worthwhile to zoom into those periods to learn what causes these changes and how to potentially improve the model. In case auto-correlation is left in the residuals, an error model can be used to improve the forecasts by applying an auto-regressive model on the residuals. This is somewhat equivalent to include an MA part in the original model.

As summarizing measures for validation of how well dynamics are modelled:

- Plot the auto-correlation function (ACF) of the one-step residuals.

- Plot cross-correlation functions from one-step residuals to other variables, see (Bacher et al., 2013).

Systematic patterns found in these functions lead to direct knowledge on how to improve the model, see for example the table on page 155 in Madsen (2007).

5 Example with R code

A short introduction to the functionalities and steps in setting up a model is given in the following – for more details, see the vignettes listed in Section Vignettes and the website onlineforecasting.org.

First, a few remarks on the implementation. onlineforecast models are set up using an object-oriented R6 class. The main reason for this is that R6 objects use reference pointers, which allows to make minimum changes in computer memory when updating a model fit with new data – this would not be possible with the regular S3 class objects, as they are always copied in memory when updated by a function.

Furthermore, it is noted, that model inputs and transformations simply are defined using R code. The regular formula class is not used, since it cannot operate as needed on the multi-horizon forecast matrices. The provided code for the inputs defines the transformations etc. and is executed for each input to generate the data used for regression.

Setup of data

Data must be set as variables in a list, here we have loaded D with the data for the examples:

```
class(D)


## [1] "data.list" "list"
```

As seen its class is data.list, which is inherited from the list class. Hence, it is simply a list extended with some modified and new functions (can be listed with methods(class="data.list")).

All inputs to be used must be formatted as forecast matrices and set in the list as data.frames. For example the ambient temperature forecasts:

```
class(D$Ta)


## [1] "data.frame"


head(D$Ta[ ,1:8], 4)


##          k1       k2      k3      k4       k5       k6       k7       k8
## 1 -2.82340 -3.20275 -3.1185 -3.0896 -3.13200 -3.16130 -3.16645 -3.08885
## 2 -2.90405 -3.11850 -3.0896 -3.1320 -3.16130 -3.16645 -3.08885 -2.77165
## 3 -2.93590 -3.08960 -3.1320 -3.1613 -3.16645 -3.08885 -2.77165 -2.32185
## 4 -2.89315 -3.11285 -3.0484 -3.1090 -3.11600 -2.80990 -2.36895 -2.00945
```

The time must be in a POSIXct vector named t:

```
D$t[1:4]


## [1] "2010-12-15 01:00:00 UTC" "2010-12-15 02:00:00 UTC"
## [3] "2010-12-15 03:00:00 UTC" "2010-12-15 04:00:00 UTC"
```

Observations must be in a numeric vector:

```
D$heatload[1:4]


## [1] 5.916667 5.850000 5.850000 5.883333
```

For more details on the data.list class, see the setup-data vignette – which demonstrates useful functions for manipulating, validating and exploring forecast data.

Defining a model

Models are set up using the R6 class forecastmodel. An object of the class is instantiated by:

```
model <- forecastmodel$new()
```

It holds variables and functions for representing and manipulating a model.

If we want to forecast the observed heatload variable in the data list D, we set that as the model output by:

```
model$output <- "heatload"
```

The model inputs must then be defined. We can add an input as a linear function by:

```
model$add_inputs(Ta = "Ta")
```

The code given as text simply evaluates into the Ta forecast matrix, which will lead to the k-step forecast of ambient temperature (i.e. a column in Ta) will be set directly into the design matrix for the k horizon regression (more explanation of this is given in the end of the current section).

Adding an intercept to a model can be done by:

```
model$add_inputs(mu = "one()")
```

where the function one() evaluates into a forecast matrix of 1's, which will be inserted in the design matrix, see details in Appendix Regression.

Functions for a range of useful transformations were already listed in Section Transformations. Dynamics can be modelled using filters, for example low-pass filtering of a variable with:

```
model$add_inputs(Ta = "lp(Ta, a1=0.9)")
```

will apply a low-pass filter along each column of Ta and return a forecast matrix with the modified data. The filter coefficient is set to $a = 0.9$. To illustrate the effect of this, see the vignette setup-and-use-model.

Non-linear effects can be modelled using basis functions. For mapping an input to basis splines the function bspline() is provided. It is a wrapper of the bs() function from the splines package and has the same arguments. To e.g. include a non-linear function of the ambient temperature:

```
model$add_inputs(Ta = "bspline(Ta, df=5)")
```

where df is the degrees of freedom of the spline function.

Functions can be nested, e.g. first a low-pass filter before mapping to basis splines:

```
model$add_inputs(Ta = "bspline(lp(Ta, a1=0.9), df=5)")
```

Varying-coefficient models can be realized with multiplication of inputs (Hastie and Tibshirani, 1993). For more details, see the solar-power-forecasting example on the website. For more examples of input transformations, e.g. fourier-series and auto-regressive inputs, see the setup-and-use-model vignette.

Execution of the input transformations

As seen above, R code is given for each input. The code is given as text and will simply be executed to calculate the data for regression. This is carried out with the function model$transform_data(data), inside which:

```
eval(parse(text=frml), data)
```

is executed for each input. The frml is the R code for the input (as a character e.g. as "Ta" or "bspline(Ta,df=5)" in the examples) and data is the list containing the variables used in the frml code (as D in the examples).

This way of defining the input formulas simply as code is very flexible. It also allows for easy debugging, for example a function used in the code can be set for debug and it is possible to step through its execution during the transformation – or even by setting "browser();" directly into in the input code to stop and step through the execution.

The only constraint to an inputs code is that it must just return a forecast matrix as a data.frame (or list of them). The regression can then be carried out *separately for every horizon* by, for the k horizon taking the k column from each of the returned matrices and bind these together into a design matrix, on which LS or RLS can be applied to calculate the k horizon forecasts.

Model fitting and offline parameter tuning

After setting up a model it can be fitted to data by carrying out the transformation and regression steps. To simply the code, functions of fitting the model are provided. Different functions implement different regression schemes. The two currently available fitting functions are lm_fit() and rls_fit() – they take offline parameters as a vector, fit a model and return the RMSE score (summed across all fitted horizons).

To demonstrate the model-fitting process, we first replace the inputs on the model defined above with two inputs by:

```
model$inputs <- NULL
model$add_inputs(mu = "one()",
                 Ta = "lp(Ta, a1=0.9)")
```

We also have to set the “score period”, which is simply a logical vector that specifies the observations to be included in the score calculation. It is useful for defining a burn-in period and dividing the data into test and training sets. For the current linear regression we simply include all points by:

```
D$scoreperiod <- rep(TRUE, length(D$t))
```

Now the summed RMSE for the horizons 1 to 6 steps ahead can be obtained by:

```
model$kseq <- 1:6
lm_fit(c(Ta__a1=0.8), model, D, scorefun=rmse, returnanalysis=FALSE)


## [1] 5.039611
```

The function can be passed to an optimizer, which can then find the parameter value(s) which minimizes the score.

Any optimizer function in R can be used, but, again to simplify the code, wrappers for optim() are included – similar wrappers can easily be made for other optimizers. The parameter(s) to optimize within the wrapper function is defined by:

```
model$add_prmbounds(Ta__a1 = c(min=0.8, init=0.95, max=0.9999))
```

Note, the double underscore syntax. The double underscore separates the input name and the name of the parameter. So in the case above the value of a1 in the R code for input Ta will be optimized, starting at an initial value of 0.95 and staying within the specified bounds.

We can then run the optimization calculating scores (to save computational time run on only a horizons 3 and 18 steps ahead):

```
lm_optim(model, D, kseq=c(3,18))
```

The `lm_optim()` function is a wrapper for the R optimizer function `optim()`. It returns the result from `optim()` and sets optimized parameters in:

```
model$prm


##    Ta__a1
## 0.8926346
```

i.e. a lower low-pass coefficient than the initial value of 0.95.

For more details, see the setup-and-use-model vignette.

Calculating forecasts

While developing models it is most convenient to use the fit functions for calculating predictions, e.g.:

```
model$kseq <- 1:24
fit <- lm_fit(model$prm, model, D)
```

will return a list holding the forecasts (in the forecast matrix `fit$Yhat`) and other useful information. Forecasts can also be calculated directly with the predict function:

```
lm_predict(model, model$transform_data(D))
```

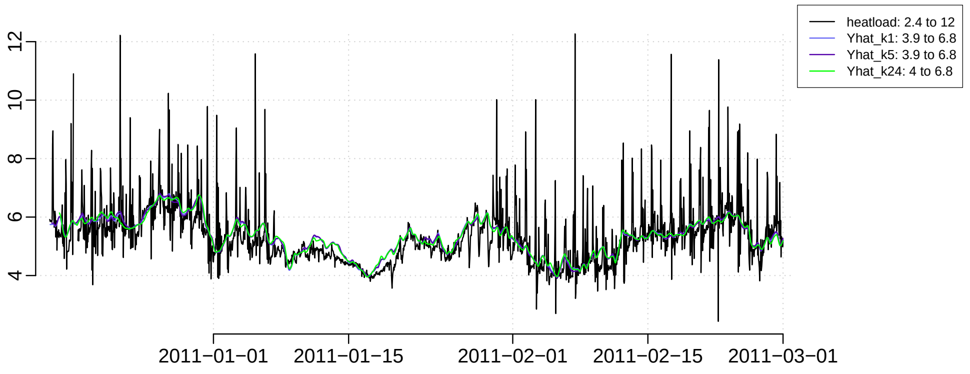
will return a forecast matrix using the input data in `D`.

Evaluation

Finally, it can be worthwhile to evaluate the forecasts and inspect the model for potential improvements. For a more comprehensive introduction, see the forecast-evaluation vignette.

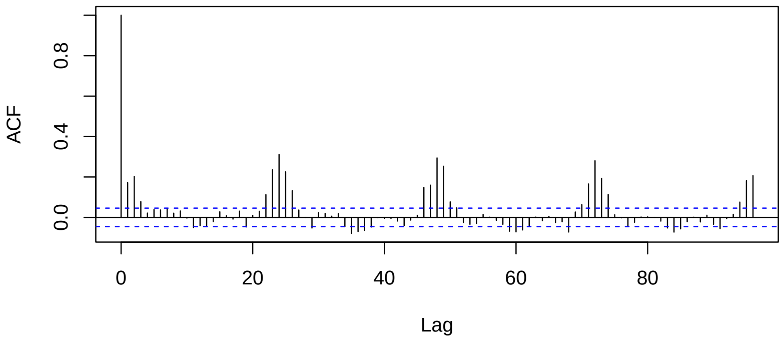
First, it is always a good idea to plot the model’s forecasts over time and see how well it predicts:

```
D$Yhat <- fit$Yhat
plot_ts(subset(D,D$scoreperiod), "heatload$|Yhat”, kseq=c(1,5,24), p=p)
```



We can plot the ACF of the one-step residuals by:

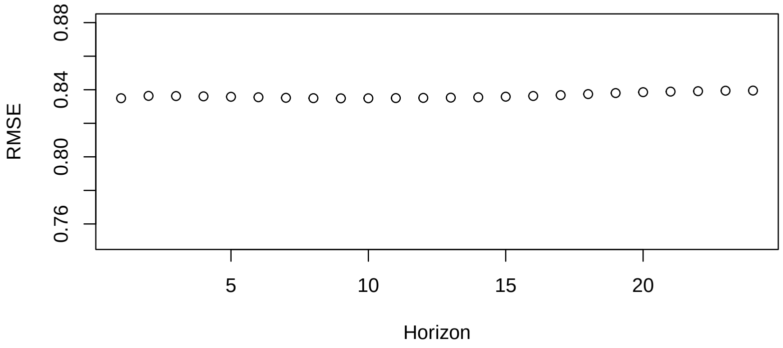
```
acf(residuals(fit)$h1, na.action=na.pass, lag.max=96, main="")
```



The ACF plot suggests that there remains a diurnal pattern to be modelled. It can be achieved by adding a diurnal curve to the model, e.g. with Fourier series basis functions. This is demonstrated in the vignette setup-and-use-model.

We may also want to calculate the score as a function of the horizon:

```
inscore <- D$scoreperiod & complete_cases(fit$Yhat)
RMSE <- score(residuals(fit), scoreperiod = inscore)
plot(RMSE, ylim=c(0.75,0.88), xlab="Horizon")
```



The trend is relatively constant, which makes sense since the model is very simple. The offline parameters were optimized for $k = 3$ and $k = 18$, which can explain why it is not monotonic increasing with the horizon.

6 Discussion and conclusion

Extending functionality

The current package is designed to make it easy to implement new transformation functions and regression schemes, as well as using other optimizers for tuning parameters.

Implementing a new transformation function is straightforward. The function must receive either a forecast matrix or a list of forecast matrices and return either after processing. Furthermore, when used in an online operational setup, where the transformation is executed whenever new data arrives, it is possible to save state information inside a transformation function, such that next time the function is called, the state can be read and used. See the `lp()` function for inspiration when writing a new transformation function.

A new regression scheme, e.g. a kernel or quantile regression, can also be implemented. A fitting function should be implemented in similar way as `lm_fit()` and `rls_fit()`, such that the first argument is the parameter vector and it returns a score value, which can be passed to an optimizer.

It is very easy to use other optimizers. The current fitting functions can simply be passed to any optimizer in R, which follows the `optim()` way of receiving a function for optimization, see the code in `lm_optim()`.

In future versions of the package, new regression techniques, e.g. kernel regression (local fitting) and quantile regression, might be added. The latter opens up the possibilities to calculate probabilistic forecasts, see (Nielsen et al., 2006) and (Bjerregaard et al., 2021), as well as carry out normalization and Copula transformations, which can be very useful for spatio-temporal forecast models, see (Tastu et al., 2011) or (Lemos-Vinasco et al., 2021).

Summary and conclusion

This paper provides an entry point and reference for working with the onlineforecast package. The paper covers version 1.0 of the package, which has been available on CRAN for almost one year at the time of writing.

The main contribution of the package is to make it easy to generate online multi-step forecasts in a flexible way. The package contains functionalities not directly available elsewhere, such as:

- Enabling the use of input variables given as forecasts, e.g. NWPs, in an easy and flexible way.
- Optimal tuning of non-linear models for multi-step horizons.
- Recursive estimation for tracking time-varying systems computationally efficient for multiple horizons.

The onlineforecast package has a significant value for anyone who needs to carry out operational online forecasting, for example, in energy scheduling, where recursive updated forecasts are needed as input to optimal decision making and real-time control of systems. It can also be very useful for companies that need online forecasts for other monitoring and real-time applications – specifically, the functionality for model updating with very little computational costs when new data becomes available, is a unique feature of the package.

Computational details

We have tried to make the onlineforecast package depend on as few other packages as possible. Only a few additional packages are used in the core functionalities: **R6** for the "usual" OOP functionalities and **Rcpp** (Eddelbuettel and Balamuta, 2018) along with **RcppArmadillo** (Eddelbuettel and Sanderson, 2014) for easy integration of fast compiled code. For extending the modelling possibilities the **splines** and **pbs** packages are essential, and for nice caching the **digest** package was used. We acknowledge the **devtools** and **knitr** (Xie, 2015), **rmarkdown** (Xie et al., 2018), **R.rsp**, **testthat** (Wickham, 2011) packages, which are indispensable for developing a package. We acknowledge the R community and the amazing work behind R done by many people over the years!

The results in this paper were obtained using R 4.3.1. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/`.

Acknowledgments

The software has been developed with funding from multiple projects: PTXHeatUtilisation (Energy Cluster Denmark), Flexible Energy Denmark, Heat 4.0 and Decision support tools for smart home energy management systems (Innovation Fund Denmark, No. 9045-00017B, 8090-00046B and 8053-00156B), TOP-UP (Innovation Fund Denmark and ERA-NET, No. 9045-00017B), Digital-twin, IEA Annex 71 and 83 Danish participation (EUDP, No. 64019-0570, 64017-05139 and 64020-1007), and finally SCA+ (EU Interreg, No. 20293290).

Bibliography

A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. S. Rangapuram, D. Salinas, J. Schulz, et al. Gluonts: Probabilistic and neural time series modeling in python. *J. Mach. Learn. Res.*, 21(116):1–6, 2020. [p175]

H. K. Alfares and M. Nazeeruddin. Electric load forecasting: Literature survey and classification of methods. *International Journal of Systems Science*, 33(1):23–34, 2002. doi: 10.1080/00207720110067421. URL https://doi.org/10.1080/00207720110067421. [p173]

P. Bacher, H. Madsen, and H. A. Nielsen. Online short-term solar power forecasting. *Solar Energy*, 83 (10):1772 – 1783, 2009. ISSN 0038-092X. doi: https://doi.org/10.1016/j.solener.2009.05.016. [p173]

P. Bacher, H. Madsen, H. A. Nielsen, and B. Perers. Short-term heat load forecasting for single family houses. *Energy and buildings*, 65:101–112, 2013. [p173, 180, 182]

M. B. Bjerregaard, J. K. Møller, and H. Madsen. An introduction to multivariate probabilistic forecast evaluation. *Energy and AI*, page 100058, 2021. [p187]

N. D. Bokde, Z. M. Yaseen, and G. B. Andersen. Forecasttb—an r package as a test-bench for time series forecasting—application of wind speed and solar radiation modeling. *Energies*, 13(10):2578, 2020. [p175]

L. Breiman and J. H. Friedman. Estimating optimal transformations for multiple regression and correlation. *Journal of the American statistical Association*, 80(391):580–598, 1985. [p178]

C. Chatfield and H. Xing. *The analysis of time series: an introduction with R*. Chapman and hall/CRC, 2019. [p174]

J. D. Cryer and K.-S. Chan. *Time series analysis: with applications in R*, volume 2. Springer, 2008. [p173]

J. G. De Gooijer and R. J. Hyndman. 25 years of time series forecasting. *International journal of forecasting*, 22(3):443–473, 2006. [p173]

E. Dotzauer. Simple model for prediction of loads in district - heating systems. *Applied Energy*, 73(3-4): 277–284, 2002. doi: 10.1016/S0306-2619(02)00078-8. [p173]

D. Eddelbuettel and J. J. Balamuta. Extending extitR with extitC++: A Brief Introduction to extitRcpp. *The American Statistician*, 72(1):28–36, 2018. doi: 10.1080/00031305.2017.1375990. [p187]

D. Eddelbuettel and C. Sanderson. Rcpparmadillo: Accelerating r with high-performance c++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063, March 2014. URL http://dx.doi.org/10.1016/j.csda.2013.02.005. [p187]

Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *IEEE Transactions on signal processing*, 52(8):2275–2285, 2004. [p174]

T. Hastie and R. Tibshirani. Varying-coefficient models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 55(4):757–779, 1993. [p183]

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009. [p178]

J. Helske. KFAS: Exponential family state space models in R. *Journal of Statistical Software*, 78(10):1–39, 2017. doi: 10.18637/jss.v078.i10. [p174]

T. Hong and S. Fan. Probabilistic electric load forecasting: A tutorial review. *International Journal of Forecasting*, 32(3):914–938, 2016. [p173]

R. J. Hyndman and Y. Khandakar. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, 26(3):1–22, 2008. URL https://www.jstatsoft.org/article/view/v027i03. [p174]

J. Kleissl. *Solar energy forecasting and resource assessment*. Academic Press, 2013. [p173]

J. Lemos-Vinasco, P. Bacher, and J. K. Møller. Probabilistic load forecasting considering temporal correlation: Online models for the prediction of households' electrical load. *Applied Energy*, 303: 117594, 2021. [p187]

L. Ljung and T. Söderström. *Theory and practice of recursive identification*. MIT press, 1983. [p174]

H. Madsen. *Time series analysis*. CRC Press, 2007. [p174, 177, 180, 182]

H. Madsen and P. Thyregod. *Introduction to general and generalized linear models*. CRC Press, 2010. [p180]

H. A. Nielsen and H. Madsen. Modelling the heat consumption in district heating systems using a grey-box approach. *Energy and buildings*, 38(1):63–71, 2006. [p173, 180]

H. A. Nielsen, H. Madsen, and T. S. Nielsen. Using quantile regression to extend an existing wind power forecasting system with probabilistic forecasts. *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology*, 9(1-2):95–108, 2006. [p187]

T. S. Nielsen, H. A. Nielsen, and H. Madsen. Prediction of wind power using time-varying coefficient functions. In *Proceedings of the XV IFAC World Congress*, 2002. [p173]

G. Petris. An r package for dynamic linear models. *Journal of Statistical Software*, 36(1):1–16, 2010. [p174]

A. H. Sayed and T. Kailath. A state-space approach to adaptive rls filtering. *IEEE signal processing magazine*, 11(3):18–60, 1994. [p193]

S. L. Shah and W. R. Cluett. Recursive least squares based estimation schemes for self-tuning control. *The Canadian Journal of Chemical Engineering*, 69(1):89–96, 1991. [p193]

J. Siebert, J. Groß, and C. Schroth. A systematic review of python packages for time series analysis. *arXiv preprint arXiv:2104.07406*, 2021. [p174]

H. Spliid. A fast estimation method for the vector autoregressive moving average model with exogenous variables. *Journal of the American Statistical Association*, 78(384):843–849, 1983. [p174]

L. J. Tashman. Out-of-sample tests of forecasting accuracy: an analysis and review. *International journal of forecasting*, 16(4):437–450, 2000. [p181]

J. Tastu, P. Pinson, E. Kotwa, H. Madsen, and H. A. Nielsen. Spatio-temporal analysis and modelling of short-term wind power forecast errors. *Wind Energy*, 14(1):43–60, 2011. [p187]

S. Weisberg. *Applied linear regression*, volume 528. John Wiley & Sons, 2005. [p178]

H. Wickham. testthat: Get started with testing. *The R Journal*, 3:5–10, 2011. URL https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf. [p187]

Y. Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition, 2015. URL https://yihui.org/knitr/. ISBN 978-1498716963. [p187]

Y. Xie, J. Allaire, and G. Grolemund. *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida, 2018. URL https://bookdown.org/yihui/rmarkdown. ISBN 9781138359338. [p187]

C. Yang, J. Qiao, Z. Ahmad, K. Nie, and L. Wang. Online sequential echo state network with sparse rls algorithm for time series prediction. *Neural Networks*, 118:32–42, 2019. [p174]

1 Forecast model notation

In this section it is shown how to write onlineforecast models in mathematical notation. Both in a full description and how to write a shorter summarized description. Note, that when variables are noted in bold font it indicates that they are multi-variate.

A model can be described in full detail as presented in the following.

The transformation stage

$$\begin{aligned}
\text{Intercept:} \quad & \mu_{t+k|t} &&= 1 && (19)\\
\text{Periodic:} \quad & \boldsymbol{x}_{\text{per},t+k|t} &&= f_{\text{fs}}(t; n_{\text{har}}) && (20)\\
\text{Part 1:} \quad & x_{1,t+k|t} &&= H(B; a)u_{1,t+k|t} && (21)\\
\text{Part 2:} \quad & \boldsymbol{x}_{23,t+k|t} &&= f_{\text{bs}}(u_{2,t+k|t}; n_{\text{deg}})u_{3,t+k|t} && (22)\\
\text{Part 3:} \quad & x_{4,t+k|t} &&= u_{4,t} && (23)
\end{aligned}$$

and the regression stage

$$Y_{t+k|t} = \beta_{0,k}\mu_{t+k|t} + \boldsymbol{\beta}_{1,k}^T\boldsymbol{x}_{\text{per},t+k|t} + \beta_{2,k}x_{1,t+k|t} + \boldsymbol{\beta}_{3,k}^T\boldsymbol{x}_{23,t+k|t} + \beta_{4,k}x_{4,t+k|t} + \varepsilon_{t+k|t} \tag{24}$$

Thus the model inputs are:

- t is simply the time value.
- $u_{1,t+k|t}$ some forecast input (e.g. NWP variable).
- $u_{2,t+k|t}$ some forecast input (e.g. could be a deterministic value, e.g. time of day which is always know (the $|t$ could be omitted)).
- $u_{3,t+k|t}$ some forecast input (e.g. NWP variable).
- $u_{4,t}$ some value at time t (e.g. an observation variable).

The functions which maps the inputs (u's) to the regression inputs (x's) are:

- $f_{\text{fs}}(t; n_{\text{har}})$ is a function generating Fourier series of some implicit period length.
- $H(B; a)$ is a low-pass filter.
- $f_{\text{bs}}(u_{2,t+k|t}; n_{\text{deg}})$ is a function generating basis splines.

Their parameters are the transformation parameters:

- n_{har} is the number of harmonics.
- a is the low-pass filter coefficient.
- n_{deg} is the degrees of freedom of the spline function.

which must be set or optimized.

The regression coefficients are

$$\begin{aligned}
\boldsymbol{\beta}_k &= \begin{bmatrix} \beta_{0,k} & \boldsymbol{\beta}_{1,k}^T & \beta_{2,k} & \boldsymbol{\beta}_{3,k}^T & \beta_{4,k} \end{bmatrix}^T && (25)\\
&= \begin{bmatrix} \beta_{0,k} & \beta_{1,1,k} & \beta_{1,2,k} & \cdots & \beta_{1,2n_{\text{har}},k} & \beta_{2,k} & \beta_{3,1,k} & \beta_{3,2,k} & \cdots & \beta_{3,n_{\text{deg}},k} & \beta_{4,k} \end{bmatrix}^T && (26)
\end{aligned}$$

If the model is fitted with a recursive scheme, thus the coefficients change over time, it should be indicated by adding a t to the subscript, e.g. $\beta_{0,k,t}$. Furthermore, other parameters can exist, which can enter an optimization at the transformation stage, e.g. the RLS forgetting factor λ. The parameters which are optimized in the transformation stage should be presented together.

Specifying the model in all details can be cumbersome to include in some texts, so it makes sense to simplify the notation. When using a simpler notation, as suggested below, it should be stated, what is implicit (e.g. the regression stage). Referencing the present text should be sufficient when using a simpler notation. Naturally, all inputs, functions, etc., should be described in some way.

A model can be specified in a simpler way, e.g. the model above in one equation

$$\begin{aligned} Y_{t+k|t} = \beta_{0,k} + \boldsymbol{\beta}_{1,k}^{T} f_{\mathrm{fs},k}(t; n_{\mathrm{har}}) + \beta_{2,k} H_k(B; a) u_{1,t+k|t} + \boldsymbol{\beta}_{3,k}^{T} f_{\mathrm{bs},k}(u_{2,t+k}; n_{\mathrm{deg}}) u_{3,t+k|t} \\ + \beta_{4,k} u_{4,t} + \varepsilon_{t+k|t} \end{aligned} \tag{27}$$

or writing the regression stage implicitly by removing the regression coefficients where it is meaningful

$$Y_{t+k|t} = \mu_k + f_{\mathrm{fs},k}(t; n_{\mathrm{har}}) + H_k(B; a) u_{1,t+k|t} + f_{\mathrm{bs},k}(u_{2,t+k|t}; n_{\mathrm{deg}}) u_{3,t+k|t} + \beta_k u_{4,t} + \varepsilon_{t+k|t} \tag{28}$$

It is then implicit that the functions are different from the previous stated functions, since they include the regression coefficients. Again, if fitted with a recursive scheme, then it can be indicated by adding a t subscript, e.g. $f_{\mathrm{fs},k,t}(t; n_{\mathrm{har}})$.

To simplify further the k on the functions can be implicit

$$Y_{t+k|t} = \mu + f_{\mathrm{fs}}(t; n_{\mathrm{har}}) + H(B; a) u_{1,t+k|t} + f_{\mathrm{bs}}(u_{2,t+k|t}; n_{\mathrm{deg}}) u_{3,t+k|t} + \beta u_{4,t} + \varepsilon_{t+k|t} \tag{29}$$

and similarly the transformation parameters can be implicit

$$Y_{t+k|t} = \mu + f_{\mathrm{fs}}(t) + H(B) u_{1,t+k|t} + f_{\mathrm{bs}}(u_{2,t+k|t}) u_{3,t+k|t} + \beta u_{4,t} + \varepsilon_{t+k|t} \tag{30}$$

Then the functions and their parameters, and the fitting scheme (i.e. with either LS or RLS for each horizon) should be described in some other way.

Finally, the most simplified notation would be to even remove the time indexing

$$Y = \mu + f_{\mathrm{fs}}(t) + H(B) u_1 + f_{\mathrm{bs}}(u_2) u_3 + \beta u_4 + \varepsilon \tag{31}$$

after making clear how all the variables are defined.

2 Regression

In this section the two regression schemes implemented in onlineforecast are described. When fitting a model, thus estimating the regression coefficients, data from a period $t \in (1, 2, \ldots, n)$ is used and passed on to either: the `lm_fit()` function which implements the Least Squares (LS) scheme, or the `rls_fit()` function, which implements the Recursive Least Squares (RLS) scheme.

One important difference between the two implementations is that in the LS the coefficients are estimated using data from the entire period, thus they are constant during the period and the calculated predictions are "in-sample". This is opposed to the RLS, where the coefficients are updated through the period using only past data at each time t. In that case the coefficients vary over time and the calculated predictions are "out-of-sample".

This difference is explained in the following and indicated by subscripting the coefficient vector with t only for the RLS.

Least squares

The regression coefficients for the k'th horizon is set in the vector

$$\boldsymbol{\beta}_k = \begin{bmatrix} \beta_{0,k} & \beta_{1,k} & \cdots & \beta_{p,k} \end{bmatrix}^T \tag{32}$$

Note, that t is not included in the subscript.

The input data for the k horizon is the design matrix

$$\boldsymbol{X}_{k,t} = \begin{bmatrix} x_{0,1+k|1} & x_{1,1+k|1} & \cdots & x_{p,1+k|1} \\ x_{0,2+k|2} & x_{1,2+k|2} & \cdots & x_{p,2+k|2} \\ \vdots & \vdots & & \vdots \\ x_{0,n-1|n-1-k} & x_{1,n-1|n-1-k} & \cdots & x_{p,n-1|n-1-k} \\ x_{0,n|n-k} & x_{1,n|n-k} & \cdots & x_{p,n|n-k} \end{bmatrix} \tag{33}$$

The output observations are in the vector

$$\boldsymbol{y}_{k,n} = [y_{1+k} \;\; y_{2+k} \;\; \cdots \;\; y_{n-1} \;\; y_n]^T \tag{34}$$

The LS estimates of the coefficients are

$$\hat{\boldsymbol{\beta}}_k = (\boldsymbol{X}_{k,n}\boldsymbol{X}_{k,n})^{-1}\boldsymbol{X}_{k,n}\boldsymbol{y}_{k,n} \tag{35}$$

The predictions are "in-sample" and calculated by

$$\hat{\boldsymbol{y}}_{k,n} = \boldsymbol{X}_{k,n}\hat{\boldsymbol{\beta}}_k \tag{36}$$

and returned when fitting a model with `lm_fit()`.

The estimated coefficients may now be used for "out-of-sample" prediction (for $t_{\text{new}} \geq n$), with the input

$$\boldsymbol{x}_{t_{\text{new}}+k|t_{\text{new}}} = \left[x_{0,t_{\text{new}}+k|t_{\text{new}}} \;\; x_{1,t_{\text{new}}+k|t_{\text{new}}} \;\; \cdots \;\; x_{p,t_{\text{new}}+k|t_{\text{new}}}\right]^T \tag{37}$$

by

$$\hat{y}_{t_{\text{new}}+k|t_{\text{new}}} = \boldsymbol{x}_{t_{\text{new}}+k|t_{\text{new}}}\hat{\boldsymbol{\beta}}_k \tag{38}$$

This can be done by providing new data to the `lm_predict()` function.

Recursive least squares

In the RLS scheme the coefficients are recursively updated through the period. Time t steps from 1 to n and in each step the "newly" obtained data at t is used for calculating updated coefficients. The coefficient vector has the same structure as for LS

$$\boldsymbol{\beta}_{k,t} = \left[\beta_{0,k,t} \;\; \beta_{1,k,t} \;\; \cdots \;\; \beta_{p,k,t}\right]^T \tag{39}$$

The only difference is that we now subscript with t because it varies over time.

Only the most recent input data at t (the row at t from the LS design matrix in Equation (33)) is used in each update

$$\boldsymbol{x}_{k,t} = \left[x_{0,t|t-k} \;\; x_{1,t|t-k} \;\; \cdots \;\; x_{p,t|t-k}\right]^T \tag{40}$$

and similarly the most recent output observation y_t. At each time t the coefficients are updated by

$$\boldsymbol{R}_{k,t} = \lambda \boldsymbol{R}_{k,t-1} + \boldsymbol{x}_{k,t}\boldsymbol{x}_{k,t}^T \tag{41}$$

$$\hat{\boldsymbol{\beta}}_{k,t} = \hat{\boldsymbol{\beta}}_{k,t-1} + \boldsymbol{R}_{k,t}^{-1}\boldsymbol{x}_{k,t}(y_t - \boldsymbol{x}_{k,t}^T\hat{\boldsymbol{\beta}}_{k,t-1}) \tag{42}$$

Hence, when applying RLS for data from the period $t \in (1, 2, \ldots, n)$ the RLS provides a new value of the coefficients for each time t (opposed to LS).

The predictions are calculated recursively as well by using the updated coefficients at each time t.

Given the inputs

$$\boldsymbol{x}_{t+k|t} = \begin{bmatrix} x_{0,t+k|t} & x_{1,t+k|t} & \cdots & x_{p,t+k|t} \end{bmatrix}^T \tag{43}$$

the prediction is

$$\hat{y}_{t+k|t} = \boldsymbol{x}_{t+k|t}\hat{\boldsymbol{\beta}}_{k,t} \tag{44}$$

Only past data has been used when calculating the predictions through the period, hence they are "out-of-sample" predictions (these predictions are returned by `rls_fit()`).

The initial value of R is set simply set to a zero matrix with diagonal $1/10000$ and β set to a zero vector.

An alternative updating scheme, which is actually the implemented scheme (gives the same results as the scheme above), is the Kalman gain scheme (Sayed and Kailath, 1994), where matrix inversion is avoided

$$\boldsymbol{K}_{k,t} = \frac{\boldsymbol{P}_{k,t-1}\boldsymbol{x}_{k,t}}{\lambda + \boldsymbol{x}_{k,t}^T\boldsymbol{P}_{k,t-1}\boldsymbol{x}_{k,t}} \tag{45}$$

$$\hat{\boldsymbol{\beta}}_{k,t} = \hat{\boldsymbol{\beta}}_{k,t-1} + \boldsymbol{K}_{k,t}(y_t - \boldsymbol{x}_{k,t}^T\hat{\boldsymbol{\beta}}_{k,t-1}) \tag{46}$$

$$\boldsymbol{P}_{k,t} = \frac{1}{\lambda}\left(\boldsymbol{P}_{k,t-1} - \boldsymbol{K}_{k,t}\boldsymbol{x}_{k,t}^T\boldsymbol{P}_{k,t-1}\right) \tag{47}$$

This actually opens up the possibilities for self-tuned variable forgetting (Shah and Cluett, 1991).

Peder Bacher
Dynamical Systems, Department of Applied Mathematics and Computer Science, Technical University of Denmark
Asmussens Allé, Building 303B
2800 Kgs. Lyngby, Denmark
E-mail: pbac@dtu.dk
URL: https://www.compute.dtu.dk/english/research/research-sections/dynsys/

Hjörleifur G. Bergsteinsson
Dynamical Systems, Department of Applied Mathematics and Computer Science, Technical University of Denmark
Asmussens Allé, Building 303B
2800 Kgs. Lyngby, Denmark
E-mail: hgbe@dtu.dk
URL: https://www.compute.dtu.dk/english/research/research-sections/dynsys/

Linde Frölke
Dynamical Systems, Department of Applied Mathematics and Computer Science, Technical University of Denmark
Asmussens Allé, Building 303B
2800 Kgs. Lyngby, Denmark
E-mail: hgbe@dtu.dk
URL: https://www.compute.dtu.dk/english/research/research-sections/dynsys/

Mikkel L. Sørensen
Dynamical Systems, Department of Applied Mathematics and Computer Science, Technical University of Denmark
Asmussens Allé, Building 303B
2800 Kgs. Lyngby, Denmark
E-mail: mliso@dtu.dk
URL: https://www.compute.dtu.dk/english/research/research-sections/dynsys/

Julian Lemos-Vinasco
Dynamical Systems, Department of Applied Mathematics and Computer Science, Technical University of Denmark
Asmussens Allé, Building 303B
2800 Kgs. Lyngby, Denmark

E-mail: jlvi@dtu.dk
URL: https://www.compute.dtu.dk/english/research/research-sections/dynsys/

Jon Liisberg
Dynamical Systems, Department of Applied Mathematics and Computer Science, Technical University of Denmark
Asmussens Allé, Building 303B
2800 Kgs. Lyngby, Denmark
E-mail: jlvi@dtu.dk
URL: https://www.compute.dtu.dk/english/research/research-sections/dynsys/

Jan Kloppenborg Møller
Dynamical Systems, Department of Applied Mathematics and Computer Science, Technical University of Denmark
Asmussens Allé, Building 303B
2800 Kgs. Lyngby, Denmark
E-mail: jkmo@dtu.dk
URL: https://www.compute.dtu.dk/english/research/research-sections/dynsys/

Henrik Aalborg Nielsen
ENFOR A/S
Røjelskær 11, 3.
2840 Holte, Denmark
E-mail: han@enfor.dk
URL: https://www.enfor.dk

Henrik Madsen
Dynamical Systems, Department of Applied Mathematics and Computer Science, Technical University of Denmark
Asmussens Allé, Building 303B
2800 Kgs. Lyngby, Denmark
E-mail: hmad@dtu.dk
URL: https://www.compute.dtu.dk/english/research/research-sections/dynsys/

markovMSM: An R Package for Checking the Markov Condition in Multi-State Survival Data

by Gustavo Soutinho and Luís Meira-Machado

Abstract Multi-state models can be used to describe processes in which an individual moves through a finite number of states in continuous time. These models allow a detailed view of the evolution or recovery of the process and can be used to study the effect of a vector of explanatory variables on the transition intensities or to obtain prediction probabilities of future events after a given event history. In both cases, before using these models, we have to evaluate whether the Markov assumption is tenable. This paper introduces the **markovMSM** package, a software application for R, which considers tests of the Markov assumption that are applicable to general multi-state models. Three approaches using existing methodology are considered: a simple method based on including covariates depending on the history; methods based on measuring the discrepancy of the non-Markov estimators of the transition probabilities to the Markovian Aalen-Johansen estimators; and, finally, methods that were developed by considering summaries from families of log-rank statistics where individuals are grouped by the state occupied by the process at a particular time point. The main functionalities of the **markovMSM** package are illustrated using real data examples.

1 Introduction

Multi-state models are commonly used to analyze complex longitudinal survival data involving multiple events of interest (Andersen et al., 1993; Hougaard, 2000; Putter et al., 2007; Meira-Machado et al., 2009; Meira-Machado and Sestelo, 2019). These models can be considered as a generalization of the survival process where survival is the ultimate outcome of interest, but where information is available about intermediate events that individuals may experience during the study period. For instance, in some biomedical applications, besides the 'healthy' initial state and the absorbing 'dead' state, one may observe intermediate (transient) states based on health conditions (e.g., heart or lung disease), disease stages (e.g., stages of cancer, HIV infection or Alzheimer's), clinical symptoms (e.g., bleeding episodes), biological markers (e.g., CD4 T-lymphocyte cell counts; serum immunoglobulin levels) or they can represent a non-fatal complication in the course of the illness (e.g., cancer recurrence, transplantation, etc.). The number of states and transitions among them defines the complexity of the model, which is illustrated by a directed graph with rectangular boxes representing possible states and arrows representing the allowed transitions. Among the multi-state models, the simplest is the mortality model for survival data (with only two states). The competing risks model (Andersen and Keiding, 2002; Putter et al., 2007) can be seen as an extension of the simple mortality model for survival data in which each individual may 'die' due to any of several causes. The most common is the illness-death model, also known as the disability model, which comprises three states. In the irreversible version of this model (Figure 1), individuals start in the 'alive and disease-free' state and subsequently move either to the 'diseased' state or to the 'dead' state. Individuals in the 'diseased' state will eventually move to the 'dead' state, without any possibility of recovery. Data that arises from the model depicted in Figure 1 has been termed as semi-competing risk data and can be seen as an emerging challenge for time-to-event data where two events of interest are studied: non-terminal (e.g., disease diagnosis) and terminal (e.g., death) events. The non-terminal event is observed only if it precedes the terminal event, which may occur before or after the non-terminal event. In this context, the terminal event is a competing risk for the intermediate state but not the contrary (Xu et al., 2010; Li and Peng, 2017; Nevo and Gorfine, 2020).

Multi-state models can be used to assess the effects of covariates on the transition intensities, but they can also be used to obtain conditional prediction probabilities of future events. From the transition intensities, it is possible to evaluate the probability of one event ahead, but the transition probabilities are the relevant predictive quantities since they consider several transitions ahead. The occupation probabilities, often used in practice, are particular examples of transition probabilities in which the time origin is taken as the current time. When the underlying stochastic process is Markov, an elegant theory can be used to link the transition intensities to the transition probabilities, leading to the Aalen-Johansen estimator (Aalen and Johansen, 1978). This assumption claims that given the present state, the future evolution of the process is independent of the states previously visited and the transition times among them; in other words, the history of the process is summarized by the current state. The Aalen-Johansen estimator is adapted to censoring, and benefits from the assumption of

Markovianity to get consistent estimates. However, when the multi-state model is non-Markov, this is no longer the case. In fact, there are many applications for which the Markov assumption may not be met; for example, the arrival time at the current state of the process often influences the transition intensities, leading to non-Markov structures (Andersen et al., 2000, Andersen and Keiding, 2002). Although the Aalen–Johansen estimator may be used to consistently estimate occupation probabilities for non-Markov multi-state models (Datta and Satten, 2001), in general it provides biased estimators if the Markov assumption does not hold. To tackle this issue, Meira-Machado et al. (2006) introduced, for the first time, estimators that do not rely on the Markov assumption. These authors showed the practical superiority of their estimators relative to the Aalen-Johansen estimator in situations in which the Markov condition is strongly violated. However, this approach has the drawback of requiring that the support of the censoring distribution contains the support of the lifetime distribution, otherwise they only report valid estimators for truncated transition probabilities. Recently, alternative estimators that are consistent regardless of the Markov condition and the referred assumption on the censoring support were proposed by de Uña-Álvarez and Meira-Machado (2015). The idea behind the proposed estimators is to use a procedure based on differences between Kaplan-Meier estimators derived from a subset of the data, consisting of all subjects observed to be in a given state at a given time. The same idea of subsampling, combined with the Aalen-Johansen estimator, was later used by Putter and Spitoni (2018) to introduce new estimators which were termed landmark Aalen-Johansen.

In the multi-state setting, it is of practical interest to determine whether the Markov property holds within a particular data set. This information can be used to assess the validity of the transition probability estimates as well as the best option for performing inference on the transition intensities when one aims to relate the individual characteristics to the intensity rates through a multi-state regression model. This assumption is usually checked by including covariates depending on the history (Kay, 1986; Andersen et al., 2000; Andersen and Keiding, 2002). For the progressive illness-death model, for example, the Markov assumption is particularly relevant for modeling the death transition after disease and, consequently, assessing whether this transition rate is affected by the time in the previous state. Alternative methods, based on a local Kendall's tau measuring the future-past association along time, were proposed by Rodriguez-Girondo and de Uña-Álvarez (2012) and Rodriguez-Girondo and de Uña-Álvarez (2016). The method proposed by these authors can be used for three-state progressive and illness-death models, but the extension of this test to general multi-state models is not straightforward, and thus flexible methods that may be used in general models are required. A very recent work by Titman and Putter (2020) considers new approaches for checking this assumption. In one of these approaches, local and general tests have been developed by considering summaries from families of log-rank statistics where individuals are grouped by the state occupied at varying initial time s. Chiou et al. (2018) also considered an equivalent problem to testing Markovianity (in the progressive illness-model) but involving tests for dependent truncation. Making use of the landmark approach, Soutinho and Meira-Machado (2021) introduces new methods for checking the Markov assumption through a local and global test given by measuring the discrepancy between the Aalen-Johansen estimator and the landmark estimators (de Uña-Álvarez and Meira-Machado, 2015, Putter and Spitoni, 2018) that are free of the Markov condition.

There are some software packages available for multi-state survival analysis. A comprehensive list of the available packages in the Comprehensive R Archive Network (CRAN) can be seen in the CRAN task view 'Survival Analysis' (Allignol and Latouche, 2022). In particular, in the context of multi-state models, **mstate** package provides functions for estimating hazards and probabilities, possibly depending on covariates. This package also permits checking the Markov assumption from families of log-rank statistics where individuals are grouped by the state occupied at different times (Titman and Putter, 2020). The **msm** package can be used to fit a Markov model with any number of states and any pattern of transitions to panel data. It includes several extensions, such as hidden Markov models and models whose transition intensities vary with individual-specific or time-varying covariates (Jackson, 2011). **survidm** package provides estimates of predictive probabilities, such as the transition probabilities, occupation probabilities, cumulative incidence functions, and waiting (sojourn) time distributions. It also allows one to perform multi-state regression for transition intensities as well as to implement global tests for the Markov assumption in illness-death models (Soutinho et al., 2021). Other examples are the **SemiMarkov** package that can be used to fit semi-Markov multistate models to longitudinal data proposed by Król and Saint-Pierre (2015) and the **frailtyEM** which contains functions for fitting shared frailty models with a semi-parametric baseline hazard with the Expectation-Maximization algorithm (Balan and Putter, 2019). Multi-state regression can be performed using several of the aforementioned packages by decoupling the whole process into various survival models, by fitting separate intensities to all permitted transitions using, for example, semi-parametric Cox proportional hazard regression models, while making appropriate adjustments to the risk set. To this end, the **JM** and **joineR** packages, proposed by Rizopoulos (2010) and Philipson et al. (2018), can be useful to estimate the parameters for the joint modeling of longitudinal and time-to-event data. In this context, we can also refer to the **joineRML** which extends the inference of the joint model to the case

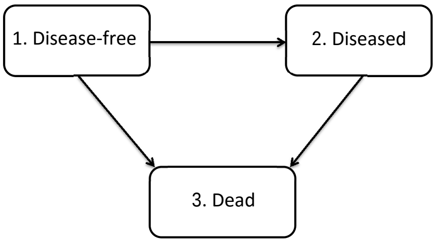


Figure 1: The progressive illness-death model with three states and three irreversible transitions. 'Disease-free' correspond to the initial state, 'Diseased' the intermediate state and the absorbing state 'dead'.

of multiple continuous longitudinal measures (Hickey et al., 2018).

The organization of this paper is as follows: In Section 2, we present a brief introduction of the notation and the mathematical background of existing methods for testing the Markov assumption. In Section 3, we introduce the **markovMSM** package, a software application for R that performs these local and global tests in the multi-state models described in Soutinho and Meira-Machado (2021) using three real data examples. Finally, the main conclusions are reported in Section 4.

2 Background and notation

This section contains a brief description of the mathematical background underlying the **markovMSM** package. Further details on the proposed methods can be found in Soutinho and Meira-Machado (2021).

Multi-state models and transition probabilities

A multi-state model $(X(t), t \in [0, \infty))$ is a model for a time continuous stochastic process. These models provide a relevant modeling framework for dealing with complex longitudinal survival data, since they allow the representation of the movement of the individuals among a set of finite states space $\mathcal{S} = 1, \ldots, K$. They can be fully characterized through the transition probabilities, for two states h, j and two time points $s < t$, that are given by the conditional probability $p_{hj}(s,t) = P(X(t) = j|X(s) = h, \mathcal{H}_{s-})$, where $\mathcal{H}_{s-}$ denotes the history of the process up to time s. This history can represent information about the process, such as the states previously visited, transition times, etc. Under the Markov assumption, $P(X(t) = j|X(s) = h, X(u) = y) = P(X(t) = j|X(s) = h)$, for any $0 \leq u < s$ and $y \in \mathcal{S}$, and thus, the future of the process after time s depends only on the state occupied at time s, not on the arrival time to that state or on the states previously visited.

Without loss of generality and for the purpose of simplicity, let us consider the progressive illness-death model, with state space $\{1, 2, 3\}$, given by the initial State (e.g., 'alive and disease-free'; State 1), the intermediate state (e.g., 'diseased'; State 2), and the absorbing state (e.g., 'dead'; State 3). For this type of model, there are five different transition probabilities to be considered that can be obtained as follows (Meira-Machado and Sestelo, 2019):

$$p_{11}(s,t) = \exp\left(-\int_s^t (\lambda_{12}(u) + \lambda_{13}(u))\, du\right)$$

$$p_{22}(s,t \mid t_{12}) = \exp\left(-\int_s^t \lambda_{23}(u, t_{12}) du\right)$$

$$p_{12}(s,t) = \int_s^t p_{11}(s, u-)\lambda_{12}(u) p_{22}(u, t \mid u) du.$$

Here, $p_{22}(s,t \mid t_{12})$ denotes the transition probability p_{22} conditional on a particular entry time t_{12}, and $\lambda_{hj}(.)$ correspond to the transition intensities that are the instantaneous hazards for movement from one state (h) to another (j). If the process is Markov, $h_{23}(t, t_{12}) = h_{23}(t)$ and $p_{22}(s,t \mid t_{12}) = p_{22}(s,t)$. The two other transition probabilities, $p_{13}(s,t)$ and $p_{23}(s,t)$, can be estimated from the two obvious relations that exist in the progressive illness-death model: $p_{11}(s,t) + p_{12}(s,t) + p_{13}(s,t) = 1$ and $p_{22}(s,t) + p_{23}(s,t) = 1$. Expressions for general models are not possible. Since the prognosis for an individual in the intermediate state may be influenced by the subject's specific arrival time, the illness-death model is not necessarily Markovian.

The estimation of the transition probabilities is a major goal in multi-state models since they allow for long-term predictions. A common nonparametric method to estimate these quantities is the Aalen-Johansen (AJ) estimator, (Aalen and Johansen, 1978) which can be obtained as the product–integral of the Nelson-Aalen estimators for the cumulative transition intensities (Andersen et al., 1993). Explicit formulae of the Aalen-Johansen estimator for the illness-death model can be found in Borgan (2005). When the multi-state process is Markovian, the AJ estimator provides consistent estimates of the transition probabilities, but this is no longer the case if the process is non-Markov. To avoid this problem, de Uña-Álvarez and Meira-Machado (2015) use the idea of subsampling, also referred to as landmarking (Van Houwelingen, 2007), which is based on (differences between) Kaplan-Meier estimators derived from a subset of the data consisting of all subjects observed to be in the given state at the given time. For the specific case of the illness-death model, given the time point s, to estimate $p_{1j}(s,t)$ for $j = 1,2,3$ the landmark analysis is restricted to the individuals observed in State 1 at time s; whereas, to estimate $p_{2j}(s,t)$, $j = 2,3$, the landmark analysis proceeds from the sample restricted to the individuals observed in State 2 at time s. Further details on the formulae and proofs can be seen in de Uña-Álvarez and Meira-Machado (2015). The subsampling approach was later used by Putter and Spitoni (2018) to derive a landmark Aalen-Johansen estimator (LMAJ) of the transition probabilities. The idea behind the proposed estimator is to use the Aalen-Johansen estimator of the state occupation probabilities derived from those subsets (consisting of subjects occupying a given state at a particular time) for which consistency has already been proved in multi-state models that are not necessarily Markov (Datta and Satten, 2001). It is worth mentioning that, in the illness-death model, the landmark Aalen-Johansen estimators (LMAJ) and the landmark estimators (LM) of the transition probabilities have shown similar performances. However, the former methods (LMAJ) can be used in general multi-state models, which can be considered an advantage.

From now on, we will use the abbreviation AJ for the Aalen-Johansen estimator, LM for the LandMark estimator proposed by de Uña-Álvarez and Meira-Machado (2015), and LMAJ for the LandMark Aalen-Johansen estimator (Putter and Spitoni, 2018).

Tests for the Markov assumption

Modeling particular transition intensities

Traditionally, the Markov condition is verified by modeling particular transition intensities on aspects of the history of the process using a proportional hazard model (Kay, 1986; Andersen et al., 2000; Andersen and Keiding, 2002). In the progressive illness-death model, the Markov condition is particularly relevant for the transition from the intermediate state 'disease' (State 2) to 'death' (State 3). Under this model, we can examine whether the time spent in the initial state (State 1) is important in the transition from the disease state to death or not. For doing that, let $\lambda_{23}(t)$ denote the hazard function of T for those individuals going from State 2 to State 3. Fitting a Cox model $\lambda_{23}(t \mid Z) = \lambda_{23,0}(t)\exp(\beta Z)$, where $\lambda_{23,0}$ is the baseline hazard, Z the 'time spent in the healthy (initial) state' and β a regression parameter, we now need to test the null hypothesis, if the transition rate from the disease state into death is unaffected by the time spent in the healthy state, $H_0 : \beta = 0$, against the general alternative, $H_1 : \beta \neq 0$. This 'global' test, based on the semiparametric Cox proportional hazards model, assumes proportional hazards and a linear effect on the hazard for the covariate. When either of these two assumptions is not fulfilled, the test may not be able to detect the lack of Markovianity.

Since the landmark methods (LM) for estimating the transition probabilities proposed by de Uña-Álvarez and Meira-Machado (2015), and (LMAJ) by Putter and Spitoni (2018) are free of the Markov assumption, they can be used as a basis for the construction of tests for Markovianity. These ideas were used by Soutinho and Meira-Machado (2021) to introduce local and global tests for Markovianity based on the discrepancy of the landmark estimators to Markovian Aalen-Johansen estimators (AJ). Subsampling, also known as landmarking, has been also used by Titman and Putter (2020) to introduce a general test based on summaries from families of log-rank statistics where individuals are grouped by the state they occupied at a given (landmark) time.

AUC Local Test

Soutinho and Meira-Machado (2021) introduce a local test based on the area under curve, AUC, given by the estimated transition probabilities, for a fixed time $s > 0$, that can be used for a general multi-state model for checking the Markovianity of the process. They propose the following test statistic that is based on the difference between the area under the estimated transition probability curve for the non-Markov LMAJ estimator and the AJ estimator, $U = \int_s^\tau \left(\widehat{p}_{hj}^{\,\texttt{LMAJ}}(s,u) - \widehat{p}_{hj}^{\,\texttt{AJ}}(s,u) \right) du$, where τ is the upper bound of the support of T. When the process is Markovian, it is expected that the test statistic will be close to zero. The Markov assumption becomes less likely as the test statistic gets further away from zero in either direction. Because of censoring, both estimators (LMAJ and AJ) may reveal high variability in the right tail, which may inflate the test statistic. In addition to this issue, since landmarking is based on reduced data, the maximum point for which the LMAJ transition probability estimate is strictly defined may be lower than the maximum point for AJ. To address these issues, we propose that when computing U, the minimum between the upper bound for which LMAJ is defined and the $90th$ percentile of the total time for the upper limit in the integral that defines the test statistic be used.

In the progressive illness-death model, besides the transition probability $\widehat{p}_{23}(s,t)$, $\widehat{p}_{12}(s,t)$ can also be used to test the Markov assumption. For general multi-state models, one can use transitions depending on history. In fact, if the goal is to decide which estimator is the most appropriate to use to estimate a specific transition probability $p_{hj}(s,t)$, then the test statistic should be the one based on that same transition probability.

To approximate the distributions of the test statistic, bootstrap methods with a large number of resamples, M, are used. We generate M bootstrap samples, and for each sample, we calculate the test statistic $U^{\star}$ which corresponds to the difference between the areas of the AJ and the LMAJ estimators. Finally, the variance $\sigma^{\star}_{(U^{\star})}$ is obtained from the M values of $U^{\star}$. Then, according to large sample asymptotic distribution theory, when M, the number of replicates, goes to infinity, we have the following statistic distributed approximately as a standard normal distribution with a mean of 0 and variance of 1: $V = (U-0)/\sigma^{\star}_{(U^{\star})} \sim N(0,1)$. The null hypothesis will be rejected if $V > v_{(1-\alpha/2)}$ or $V < v_{(\alpha/2)}$, where $v_{(\alpha/2)}$ and $v_{(1-\alpha/2)}$ denote the $\alpha/2$ and $1-\alpha/2$ percentiles, respectively, of a normal distribution with a mean of 0 and variance of 1.

AUC Global Test

Soutinho and Meira-Machado (2021) also introduce a global test, which can be achieved by combining the results obtained from local tests over different times. The testing procedure used here involves the following steps:

1. Using the original sample of the illness-death model, obtain the percentiles 5, 10, 20, 30, and 40 of the times spent by all individuals in State 1, the so-called waiting (soujorn) time. For general multi-state models, we recommend the use of the same percentiles of the subject's specific arrival time at the corresponding state.
2. For each of the values s obtained in Step 1, obtain the probability values for the local method as explained before.
3. Obtain the mean of the probability values for each closest pair; i.e., the mean of the probability values of the following pairs of percentiles: $(5,10)$, $(10,20)$, $(20,30)$ and $(30,40)$.
4. Finally, the p-value for the global test is equal to the minimum between the four probability values obtained in Step 3. In the case of this probability value being below a particular significance value, for example 0.05, we reject the null hypothesis of the process being Markovian.

Step 1 considers a global test based on local tests computed at low percentiles of subject-specific arrival times at the corresponding state. This is based on our experience that the failure of Markovianity often occurs at small transition times. Besides the hypothesis tests proposed above, we also suggest graphical local tests that can be used to check the Markov assumption in the illness-death model as well as for more complex multi-state models, possibly with reversible transitions between states. These graphical tests can be used to validate the default values proposed in Step 1 or to propose alternative values for which a discrepancy between the two methods (LMAJ and AJ) is more evident.

The procedure described in Step 3 can be used to ensure that there is a discrepancy between the two estimated curves over a large range of time values.

Log-rank test

The construction of the landmark Aalen-Johansen estimators (LMAJ) also motivated the development of new tests of the Markov assumption (Titman and Putter, 2020). Assume initially that interest lies in assessing the validity of transition probability estimates from a specific start point time s and a given state h, i.e. $p_{hj}(s,t)$. Under the landmark approach, the estimators of the transition probabilities should be derived from the subset of the data consisting of all subjects observed to be in the state h at time s, $S = \{i : X_i(s) = j, Y_i(s) = 1\}$ where $Y_i(s)$ is the at risk indicator of the process (de Uña-Álvarez and Meira-Machado, 2015; Putter and Spitoni, 2018). On the other hand, the Aalen-Johansen estimator would use also the set of subjects in $S^c = \{i : X_i(s) \neq j, Y_i(s) = 1\}$. Besides, under a Markov process, the transition intensities of the process for $t > s$ will be the same in the two groups. This motivated Titman and Putter (2020) to propose a local test for the Markov assumption that lies in

Function	Description
`PHM.test`	Performs a global test for the Markov assumption based on the Cox PH model.
`AUC.test`	Performs global and local tests for the Markov assumption using the Area Under Curves (AUC) method.
`LR.test`	Log-rank based test for the validity of the Markov assumption.
`plot.markovMSM`	Plot for an object of class `markovMSM`.
`eventsMSM`	Counts the number of observed transitions in the multi-state model.
`prepMSM`	Prepares the data set for multi-state modeling in a long format from a data set in wide format.
`transMatMSM`	Define transition matrix for a multi-state model.
`print.markovMSM`	Print for an object of class `markovMSM`.
`summary.markovMSM`	Summary for an object of class `markovMSM`.

Table 1: Summary of functions in the **markovMSM** package.

testing $H_0 : \lambda(t|X(s) = h) = \lambda(t|X(s) \neq h)$ for $t \geq s$ versus a general alternative. The proposed test is constructed using several log-rank statistics grouped by membership in S. Titman and Putter (2020) also propose a global test via a grid of times in which, for each s time, local tests are computed. In this paper, the grid of values is given by the percentile times.

The same authors pointed out that the use of the individual supremum or integrated statistics may be combined to provide an overall test of the dependence of the transition intensities on occupation in state j at previous times.

3 Application of the markovMSM

This section offers the guidelines for the use of the **markovMSM** package with the R statistical program (R Core Team, 2021). To this end, a description of the functions of the package is illustrated using three real data sets. The first one involves data from a clinical trial on colon cancer modeled using the progressive illness-death model (Moertel et al., 1995). Extensions to progressive processes beyond the three-state illness-death model are demonstrated using data from the European Group for Blood and Marrow Transplantation (EBMT) (Putter et al., 2007). Finally, we use data from a study with liver cirrhosis patients subjected to prednisone treatment (Andersen et al., 1993). The package comprises seven main functions, which are briefly summarized in Table 1.

Data manipulation

In this section, we reanalyze data from a large clinical trial on Duke's stage III patients affected by colon cancer who underwent curative surgery for colorectal cancer (Moertel et al., 1990). Of the 929 patients, 468 developed a recurrence, and among these, 414 died; 38 died without a recurrence. The remaining 423 patients were alive and disease-free up to the end of the follow-up. Besides the two event times (time to recurrence and time to death) and the corresponding indicator statuses, a vector of covariates including age, sex, number of lymph nodes, and extent of local spread is also available. Below is an excerpt of the data set with one row per individual. Individuals were chosen in order to represent all possible combinations of movements among the three states.

```
> library(markovMSM)
> data("colonMSM")
> db_wide <- colonMSM
> head(db_wide[c(1:2,16,21),1:11])
```

```
   time1 event1 Stime event      rx sex age obstruct perfor adhere nodes
1    968      1  1521     1 Lev+5FU   1  43        0      0      0     5
2   3087      0  3087     0 Lev+5FU   1  63        0      0      0     1
16  1323      1  3214     0     Obs   1  68        0      0      0     1
21  2789      0  2789     1     Obs   1  64        1      0      0     1
```

The four initial variables describe the movement of the patients among the three states of the illness-death model: `time1` denote the time measured in days from surgery to recurrence, whereas `Stime` is the total time or the time to death or censoring; `event1` and `event` denote the corresponding status/censoring indicator (1 for an event and 0 for censoring). Patient 1 had a recurrence after 968 days (i.e., observed a transition from the initial state to the intermediate state) and then died after 1521 days in the study. Patient 2 remained alive and without recurrence at the end of follow-up (`event1 = 0` and `event = 0`). The two event times are equal in these cases. The patient represented in the third line had a recurrence after 1323 days but remained alive at the end of the follow-up (i.e., in State 2). Finally, the patient represented in the last line died after 2789 days in the study without experiencing a recurrence.

As the original data set is in the wide format, the next step to implementing the proposed methods will be to convert the data into a long format, which is given by one line for each transition for which a subject is at risk. This can be done using functions `transMatMSM` and `prepMSM`. `transMatMSM` function defines the transition matrices, revealing which transitions are possible, whereas `prepMSM` provides a new dataset in a long format for which each row will correspond to a transition for which a patient is at risk. For the progressive illness-death model, these two functions are used as follows:

```
> positions <- list(c(2, 3), c(3), c())
> state.names <- c("Alive", "Rec",  "Death")
> tmat <- transMatMSM(positions, state.names)
> tmat
       to
from    Alive Rec Death
  Alive    NA   1     2
  Rec      NA  NA     3
  Death    NA  NA    NA

> timesNames <- c(NA, "time1", "Stime")
> status <- c(NA, "event1", "event")
> trans <- tmat
> db_long <- prepMSM(data = db_wide, trans, timesNames, status)
> db_long[1:10,]


   id from to trans Tstart Tstop time status
1   1    1  2     1      0   968  968      1
2   1    1  3     2      0   968  968      0
3   1    2  3     3    968  1521  553      1
4   2    1  2     1      0  3087 3087      0
5   2    1  3     2      0  3087 3087      0
6   3    1  2     1      0   542  542      1
7   3    1  3     2      0   542  542      0
8   3    2  3     3    542   963  421      1
9   4    1  2     1      0   245  245      1
10  4    1  3     2      0   245  245      0
```

Finally, in terms of manipulation of data, a useful function is `eventsMSM` since it allows one to summarise the number of transitions among states and their percentages:

```
> eventsMSM(db_long)

$Frequencies
       to
from    Alive Rec Death no event total entering
  Alive     0 468    38      423            929
  Rec       0   0   414       54            468
  Death     0   0     0      452            452

$Proportions
       to
from        Alive       Rec     Death  no event
  Alive 0.0000000 0.5037675 0.0409042 0.4553283
  Rec   0.0000000 0.0000000 0.8846154 0.1153846
  Death 0.0000000 0.0000000 0.0000000 1.0000000
```

Methods for testing the Markov condition in the illness-death model

The illness-death model is a special multi-state model with several applications in the biomedical literature. Many time-to-event data sets with multiple end points can be reduced to this generic structure. Unlike the survival or competing risk models, this model is not necessarily Markovian since the prognosis for an individual in the intermediate state may be influenced by the subject's specific arrival time. Traditionally, the Markov assumption is checked by including covariates depending on the history. In the particular case of the colon cancer data set, we are interested in assessing if the transition rate from the recurrence state into death is unaffected by the time spent in the previous state. This global test for the Markov assumption can be done using the function PHM.test. A brief description of the arguments of this function is shown in Table 2. Results for this global test of our data indicated that the effect of the time spent in State 1 is not significant (p-value of 0.154), revealing no evidence against the Markov model for the colon data. The corresponding input codes are the following:

```
> res <- PHM.test(data = db_long, from = 2, to = 3)
[1] 0.1543195
> res
$p.value
[1] 0.1543195
$from
[1] 2
$to
[1] 3
```

Argument	Description
data	A data frame in the long format containing the subject id; from corresponding to the starting state; the receiving state, to; the transition number, trans; the starting time of the transition given by Tstart; the stopping time of the transition, Tstop, and status (for the status variable, with 1 indicating an event (transition), 0 a censoring).
from	The starting state of the transition to check the Markov condition.
to	The last state of the considered transition to check the Markov condition.

Table 2: Summary of the arguments of the function PHM.test.

In the **markovMSM** package the local test proposed in Section 2.2.2 is performed using function AUC.test, through argument type='local'. A summary of the arguments of this function is presented in Table 3.

The input commands to perform the AUC local test, for a fixed time $s = 180$ and transitions 1→2 and 1→3 are the following:

```
> set.seed(1234)
> res2 <- AUC.test(data = db_long, times = 180, from = 1, to = 3, type = 'local',
                   replicas = 100, tmat = tmat)
> res2$localTest
    s    1->1        1->2        1->3
1 180 0.2902191 0.002982042 0.002992007
```

As a result, the function AUC.test returns the probability values for all attainable transitions from the initial state. To obtain the same local test for transition 2→3, we only need to put 2 in the parameter from as follows:

```
> set.seed(1234)
> res3 <- AUC.test(data = db_long, times = 180, from = 2, to = 3, type = 'local',
                   replicas = 100, tmat = tmat)
> res3$localTest
    s     2->2         2->3
1 180  0.02547708   0.04816232
```

Results reveal a possible failure of the Markov assumption with low probability values for transitions 1→2 and 2→3 for $s = 180$ (less than 5%). These findings agree with the results depicted in Figure 2, which reports the estimated transition probabilities. In fact, we can observe departures between the two Markov-free estimators (LM and LMAJ) and the Aalen-Johansen estimator (AJ) revealing a possible failure of the Markov assumption. The input commands to obtain the plots shown in Figure 2 are the following:

```
> plot(res2, to = 2, axis.scale = c(0,0.25), difP = FALSE) #Figure 2 (Left column)
> plot(res3, to = 3, axis.scale = c(0,1), difP = FALSE) # Figure 2 (Right column)
```

Argument	Description
`data`	A data frame in the long format containing the subject `id`; `from` corresponding to the starting state; the receiving state, `to`; the transition number, `trans`; the starting time of the transition given by `Tstart`; the stopping time of the transition, `Tstop`, and `status` for the status variable, with 1 indicating an event (transition), 0 a censoring.
`from`	The starting state of the transition probabilities.
`to`	The last receiving state considered for the estimation of the transition probabilities. All the probabilities among the first and the last states are also computed.
`type`	Type of test for checking the Markov condition: `local` or `global`. By default `type='global'`.
`times`	For the local test, times represents the starting times of the transition probabilities. In case of a global test, the argument is given by times between the minimum time and the third quartile times used in the formula of this test. Default to `NULL`.
`quantiles`	Quantiles used in the formula of the global test for the AUC methods.
`tmat`	The transition matrix for multi-state model.
`replicas`	Number of bootstrap samples to standardization of the T-statistic given by the difference of the areas of `AJ` and `LMAJ` transition probabilities estimates.
`limit`	Percentile of the event time used as the upper bound for the computation of the AUC-based test.
`positions`	List of possible transitions; x[[i]] consists of a vector of state numbers reachable from state i.
`namesStates`	A character vector containing the names of either the competing risks or the states in the multi-state model specified by the competing risks or illness-death model. names should have the same length as the list `x` (for `transMat`), or either K or $K+1$ (for `trans.comprisk`), or 3 (for `trans.illdeath`).
`timesNames`	Either 1) a matrix or data frame of dimension $n \times S$ (n being the number of individuals and S the number of states in the multi-state model), containing the times at which the states are visited or last follow-up time, or 2) a character vector of length S containing the column names indicating these times. In the latter cases, some elements of time may be `NA`
`status`	Either 1) a matrix or data frame of dimension $n \times S$, containing, for each of the states, event indicators taking the value 1 if the state is visited or 0 if it is not (censored), or 2) a character vector of length S containing the column names indicating these status variables. In the latter cases, some elements of status may be `NA`

Table 3: Summary of the arguments of function `AUC.test`.

Putting the parameter `difP=TRUE`, we can also obtain the discrepancy between the Aalen-Johansen estimator (Markovian) and the landmark non-Markovian estimator (`LMAJ`), for $p_{12}(s,t)$ and $p_{23}(s,t)$, for $s = 180$, measured through $D_{hj} = \widehat{p}_{hj}^{\mathtt{AJ}}(s,t) - \widehat{p}_{hj}^{\mathtt{LMAJ}}(s,t)$, $h = 1,2$, $j = h+1$. The 95% pointwise confidence limits were obtained using a simple bootstrap (Figure 3). The corresponding input commands in this case are the following:

```
> plot(res2, to = 2, axis.scale = c(-0.03,0.03), difP = TRUE) #Figure 3 (Left column)
> plot(res3, to = 3, axis.scale = c(-0.30,0.10), difP = TRUE) #Figure 3 (Right column)
```

The plots in Figure 3 can be thought of as graphical local tests for the Markov assumption. As expected, in both cases they reveal differences between the two methods for $s = 180$ since they show a clear deviation with respect to the horizontal line $y = 0$. From these, one gets some (graphical) evidence of the lack of Markovianity of the underlying process beyond half a year after surgery. Therefore, for this specific time, the application of the Aalen-Johansen method may not be recommended here, due to possible biases.

In Section 2, a global test was also introduced that combines the probability values of the local test over different times (given by the percentiles of the sojourn time in State 1). In the **markovMSM** package, this can be obtained using the function `AUC.test`. The arguments for this function are described in Table 3. Some examples of how to perform the proposed AUC global test are shown in the following input commands, in which we consider the default percentiles in the argument

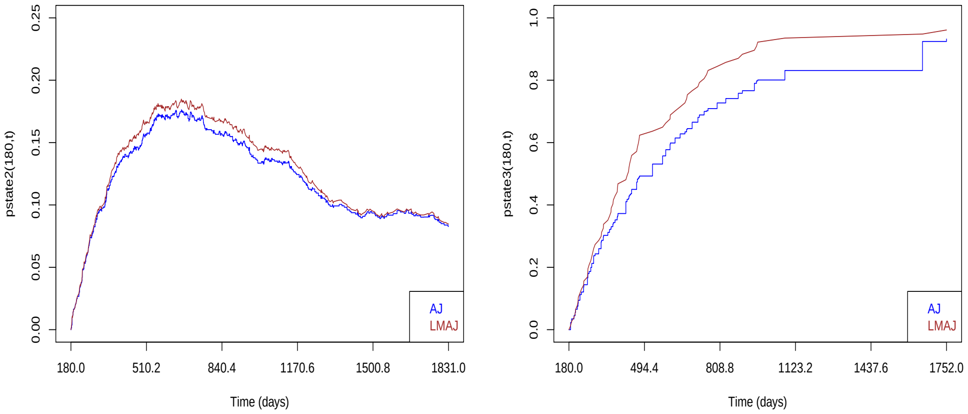


Figure 2: Estimated transition probabilities for the Aalen-Johansen (AJ) and Markov-free estimators (landmark and landmark Aalen-Johansen) using data from a colon cancer for $s = 180$. The displayed curves reveal a possible failure of the Markov assumption due to the departures between the LM and AJ estimates.

quantiles. Depending on the number of replicas, the implementation of the AUC.test function can be computationally time-consuming.

```
> set.seed(1234)
> res4 <- AUC.test(data = db_long, from = 1, to = 3, type = 'global', replicas = 100,
                   tmat = tmat)

> round(res4$globalTest,3)
   1->1  1->2  1->3
1 0.067 0.012 0.012

> set.seed(1234)
> res5 <- update(res4, from = 2)
> round(res5$globalTest,3)
    2->3
1  0.006
```

Results reported by the first command lines provide the probability values for the global test based on the AUC for the three transitions leaving State 1 (i.e., 1 → 1, 1 → 2 and 1 → 3). As expected, a higher probability value was obtained for transition 1→1 than the two remaining transitions, revealing evidence against the Markov condition. Results reported in the second set of input commands agree with previous findings, reporting a probability value of 0.006 for 2→3. Among the objects saved by this function, AUC.test displays the probability values for each percentile time (default to 5, 10, 20, 30, and 40) through the following codes:

```
> round(res4$localTest,3)
      s    1->1   1->2   1->3
1  102.4  0.978  0.081  0.081
2  173.0  0.118  0.025  0.025
3  290.6  0.015  0.000  0.000
4  469.2  0.679  0.056  0.056
5  726.8  0.635  0.176  0.176

> round(res5$localTest,3)
    s      2->2   2->3
1 102.4   0.015  0.015
2 173.0   0.011  0.011
3 290.6   0.000  0.000
4 469.2   0.050  0.050
5 726.8   0.156  0.156
```

These outputs show, for instance, that the probability value of the AUC local test reveal the fail of

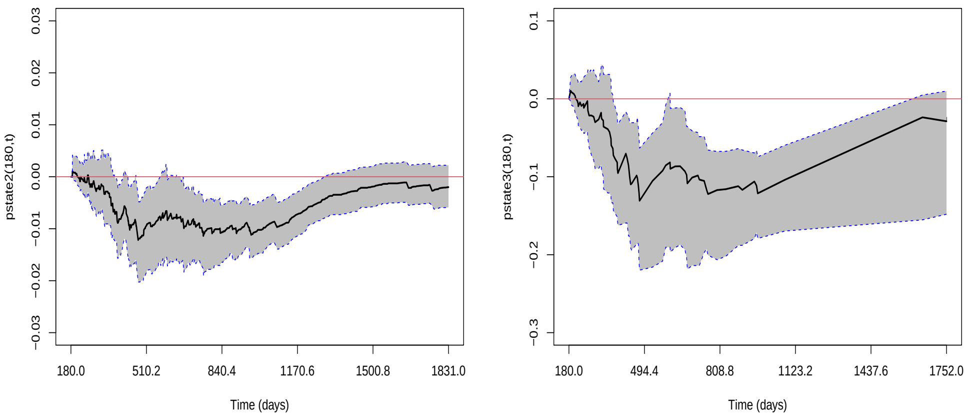


Figure 3: A local graphical test for the Markov condition for colon cancer data, for $s = 180$, based on the discrepancy between the Aalen-Johansen estimator (Markovian) and the Markov-free estimator (`LMAJ`) for each time from s. Bootstrap resamples were used to determine the 95% confidence interval.

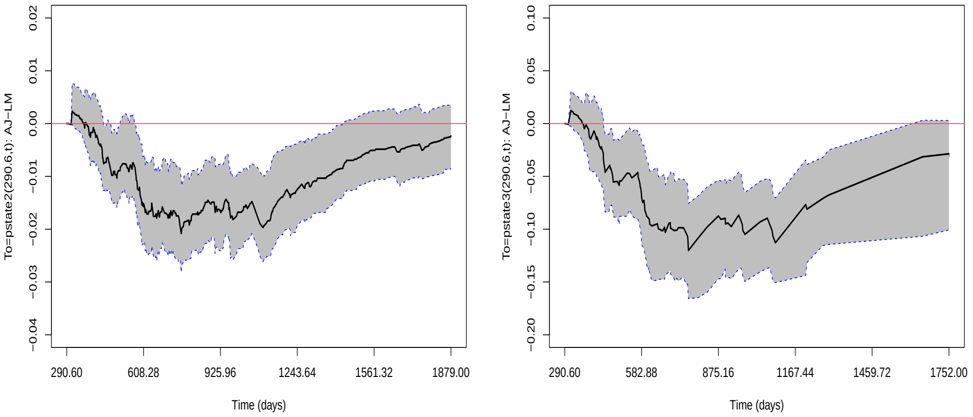


Figure 4: Local graphical test for the Markov condition for colon cancer data, for s equal to 290.6, the third percentile of the sojourn time in the initial state. Transition 1→2 is shown on the left hand side, and transition 2→3 is shown on the right hand side.

the Markovian assumption for the third percentile time (p-value < 0.001). Plots with the graphical local tests for the respective quantiles can be easily obtained using the following input commands:

```
> plot(res4, quantileOrder = 3, to = 2, axis.scale = c(-0.04, 0.02))
  #Figure 4 (Left column)

> plot(res5, quantileOrder = 3, to = 3, axis.scale = c(-0.20, 0.10))
  #Figure 4 (Right column)
```

The plots shown in Figure 4 display the differences between the `AJ` and `LMAJ` estimates for the third quantile (`quantileOrder`) of the sojourn time in the initial state being in accordance with the p-values obtained in the local tests.

Often, multi-state models include covariates, and it may be the omission of covariate effects that induces apparent non-Markovianity. The methods proposed in this paper can also deal with this problem since discrete covariates can be included in the estimation of the transition probabilities $p_{hj}(s,t)$ by splitting the sample for each level of the covariate and repeating the described procedures for each subsample. As an example, the following input codes show how to obtain the AUC global for individuals with the treatment 'Obs' of covariate 'rx'. To this end, we start to filter the wide data set

Argument	Description
data	Multi-state data in msdata format, which is created from the mstate package. It should also contain (dummy coding of) the relevant covariates; no factors allowed.
times	Grid of time points at which to compute the statistic.
from	The starting state of the transition to check the Markov condition.
to	The last state of the considered transition to check the Markov condition.
replicas	The number of wild bootstrap replications. This method, proposed by Wu (1986) is commonly applied to solve problems related to heterocedasticity. Beyersmann and Allignol (2012) found better small sample results using centered Poisson random variables rather than standard normal weights. For further information on the usage of the wild bootstrap, see Titman and Putter (2020).
formula	Right-hand side of the formula. If NULL will fit with no covariates (formula="1" will also work), offset terms can also be specified.
fn	A list of summary functions to be applied to the individual zbar traces (or a list of lists).
fn2	A list of summary functions to be applied to the overall chi-squared trace.
dist	Distribution of wild bootstrap random weights, either poisson for centred Poisson (default), or normal for standard normal.
min_time	The minimum time for calculating optimal weights.
other_weights	Other (than optimal) weights can be specified here.

Table 4: Summary of the arguments of the function LR.test.

colonMSM and run the prepMSM function to obtain the subgroups of the new data in the format long. At least, we just need to use the AUC.test function for the specific transitions as follows:

```
> db_wide_obs <- db_wide[db_wide$rx == 'Obs',]
> db_long_obs <- prepMSM(data = db_wide_obs, trans, timesNames, status)
> set.seed(12345)
> res4.obs <- AUC.test(data = db_long_obs, times = 365, from = 1, to = 3,
                       type='local', replicas = 100, tmat = tmat)
> res4.obs$localTest

    s      1->1         1->2         1->3
1 365   0.6295861   0.0003563306  0.0004307342

> set.seed(12345)
> res5.obs <- AUC.test(data = db_long_obs, times = 365, from = 2, to = 3,
                       type = 'local', replicas=100, tmat = tmat)
> res5.obs$localTest

    s         2->2         2->3
1  365    5.935795e-05  0.0001894722
```

The **markovMSM** package can also be used to compute the results of the global and local tests proposed by Titman and Putter (2020) which are based on log-rank statistics. A summary of the arguments of the LR.test function is shown in Table 4. The following input commands illustrate the usage of LR.test function to implement these tests:

```
> set.seed(1234)
> res6 <- LR.test(data = db_long, times = 180, from = 2, to = 3, replicas = 1000)
> res6$globalTestLR
  [1] 0.047
```

As we can see, the probability value of the global tests based on the log-rank statistics confirms the possible failure of Markovianity for lower values of s (P=0.047 < 0.05).

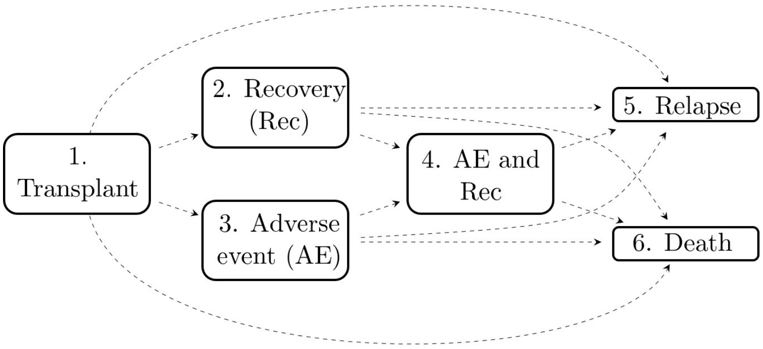


Figure 5: A six-state model given by the Blood and Marrow Transplantation (EBMT) data set for leukemia patients after bone marrow transplantation.

Tests for Markov assumption for more complex multi-state models

In this section, we use two data sets to illustrate the extension of the previous functions to more complex multi-state models. As a first example, we consider the data of 2279 patients transplanted by the European Society for Blood and Marrow Transplantation (EBMT) and, as a second example, data from liver cirrhosis patients subjected to prednisone treatment. Further details on the description of the data can be found in Putter et al. (2007) and Andersen et al. (1993), respectively.

The steps taken in the analysis are quite similar to those introduced for the illness-death model. To extend the proposed local and global tests to more complex models, we make use of the LMAJ estimator that produces consistent estimates of the transition probabilities in the case of non-Markovianity of the process.

We start by considering the data set comprising 2279 patients who suffered from blood cancers and who were treated at the EBMT between 1985 and 1998 after a transplant. The movement of the patients among the six states can be modeled through the multi-state model with the following six states: 'Alive and in remission, no recovery or adverse event' (state 1); 'Alive in remission, recovered from the treatment' (state 2); 'Alive in remission, occurrence of the adverse event' (state 3); 'Alive, both recovered and adverse event' (state 4); 'Alive, in relapse' (treatment failure) (state 5) and 'Dead (treatment failure)' (state 6). In total, there are 12 transitions: three intermediate events given by recovery (Rec), adverse event (AE) and a combination of the two (AE and Rec), and two absorbing states: relapse and death (Figure 5).

Since the original data ebmt4 is in the wide format, before implementing a global test we need to convert it into the long format using functions transMatMSM, prepMSM before using function AUC.test with the argument type='global':

```
> data("ebmt4")
> db_wide <- ebmt4
> positions <- list(c(2, 3, 5, 6), c(4, 5, 6), c(4, 5, 6),
                  c(5, 6), c(), c())
> state.names <-  c("Tx", "Rec", "AE", "Rec+AE", "Rel",  "Death")
> tmat <- transMatMSM(positions, state.names)
> timesNames <- c(NA, "rec", "ae","recae", "rel", "srv")
> status <- c(NA, "rec.s", "ae.s", "recae.s","rel.s", "srv.s")
> trans <- tmat
> db_long <- prepMSM(data = db_wide, trans, timesNames, status)
> db_long[1:10,]

Data:
   id from to trans Tstart Tstop time status
1   1    1  2     1      0    22   22      1
2   1    1  3     2      0    22   22      0
3   1    1  5     3      0    22   22      0
4   1    1  6     4      0    22   22      0
5   1    2  4     5     22   995  973      0
6   1    2  5     6     22   995  973      0
7   1    2  6     7     22   995  973      0
```

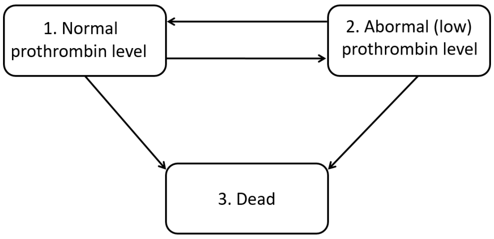

1. Normal prothrombin level
2. Aborml (low) prothrombin level
3. Dead

Figure 6: A reversible illness-death model from the data set of patients with liver cirrhosis subjected to prednisone treatment. Reversible transitions are possible in transient states defined by prothrombin levels.

```
8   2    1  2     1      0     12   12      0
9   2    1  3     2      0     12   12      1
10  2    1  5     3      0     12   12      0

> set.seed(1234)
> res7 <- AUC.test(data = db_long, from = 1, to = 5, type = 'global',
        quantiles = c(0.05, 0.10, 0.20, 0.30, 0.40),
        tmat = tmat, replicas = 100,
        positions = positions, namesStates = state.names,
        timesNames = timesNames, status = status)



> round(res7$globalTest, 4)
    1->1    1->2    1->3   1->4    1->5
1  0.1417 0.0094 0.0231 0.099 0.0068

> round(res7$localTests,4)
     s   1->1   1->2   1->3   1->4   1->5
1  9.9 0.2822 0.6030 0.0176 0.0848 0.0137
2 12.0 0.2833 0.5230 0.0285 0.4643 0.0000
3 15.0 0.0001 0.2106 0.0398 0.0191 0.1193
4 18.0 0.4584 0.0041 0.1763 0.1789 0.0240
5 21.0 0.0060 0.0147 0.4673 0.2363 0.4796
```

The interpretation of the output for the global test and for local tests of the five percentile times is similar to that shown for the illness-death model. Thus, object res7 is used to get a list with components giving the probability values for the Markov test for all transitions leaving State 1 (i.e., for 1→ j with $j \in \{1, 2, 3, 4, 5\}$). For instance, the probability value of 0.0094 corresponds to the result of the AUC global test for transition 1→2, while 0.0147 is the probability value for the local test, for $s = 21$ for the same transition.

The proposed methods can also be used in reversible multi-state models such as those applied to the data set of liver cirrhosis patients who were included in a randomized clinical trial at several hospitals in Copenhagen between 1962 and 1974. The study aimed to evaluate whether a treatment based on prednisone prolongs survival for patients with cirrhosis Andersen et al. (1993). State 1 corresponds to 'normal prothrombin level', State 2 to 'low (or abnormal) prothrombin level', and State 3 to 'dead'. The movement of the patients among these three states can be modeled using the reversible illness-death model shown in Figure 6. The input commands for the local and global tests of the AUC and Log-rank tests are presented as a vignette in the markovMSM package.

4 Conclusions

This paper discusses the implementation in R of new methods for checking the Markov assumption in multi-state models proposed by Soutinho and Meira-Machado (2021). Instead of testing, as usual, the Markovianity of the multi-state process by including covariates depending on the history, Soutinho and Meira-Machado (2021) propose global and local tests based on the comparison between estimated transition probabilities. This can be done by measuring the discrepancy between the Aalen-Johansen estimator, which gives consistent estimators in Markov processes, and recent approaches that do not rely on this assumption, given by the Landmark and the Landmark Aalen-Johansen estimators.

The use of local tests is recommended whenever the goal is to estimate the transition probabilities and, in particular, decide which estimator is the most appropriate to use: the Aalen-Johansen estimator or a robust estimator. A global test, such as the test proposed here, might be preferable for regression purposes. To this end, a common simplifying strategy is to decouple the whole process into various survival models. Then, for all permitted transitions, the transition intensities may be modeled using separate Cox models, assuming the process to be Markovian (also known as the clock forward modeling approach). When the test rejects the Markov assumption, one alternative approach is to use a semi-Markov Cox model in which the future of the process does not depend on the current time but rather on the duration of the current state.

A brief summary of the theory underlying these methods has been provided, along with an extensive review of the current literature. A detailed usage of the three main functions that compose the **markovMSM** package, `AUC.test`, `LR.test` and `PHM.test` are illustrated using three real data sets: `colonIDM`, `ebmt4` and `prothr`. Whereas the first data enables the application of the Markov tests to an illness-death model, the last two cases show how to extend the proposed methods to more complex multi-state models (with more than three states or with reversible transitions). The comparison of the proposed tests to the new methods given by the log-rank test described by Titman and Putter (2020) is also shown. We also present approaches to validate the suspicion of the failure of the Markovianity of the process using graphical tests. The **markovMSM** package is available at the first author's GitHub repository as well as the CRAN repository at https://cran.r-project.org/web/packages/markovMSM. Further details on the usage of its functions can be obtained from the corresponding help pages.

Acknowledgments

This research was financed by Portuguese Funds through FCT - "Fundação para a Ciência e a Tecnologia", within Projects projects UIDB/00013/2020, UIDP/00013/2020 and the research grant PD/BD/142887/2018.

Bibliography

O. Aalen and S. Johansen. An empirical transition matrix for non homogeneous markov and chains based on censored observations. *Scandinavian Journal of Statistics*, 5:141–150, 1978. [p195, 198]

A. Allignol and A. Latouche. Cran task view: Survival analysis. *Version 2022-03-07, URL http://CRAN.R-project.org/view=Survival*, 2022. [p196]

P. Andersen and N. Keiding. Multi-state models for event history analysis. *Statistical Methods in Medical Research*, 11(2):91–115, 2002. [p195, 196, 198]

P. Andersen, S. Esbjerg, and T. Sorensen. Multistate models for bleeding episodes and mortality in liver cirrhosis. *Statistics in Medicine*, 19(4):587–599, 2000. [p196, 198]

P. K. Andersen, Ø. Borgan, R. D. Gill, and N. Keiding. *Statistical Models Based on Counting Processes*. Springer-Verlag, New York, 1993. [p195, 198, 200, 207, 208]

T. A. Balan and H. Putter. frailtyEM: An R package for estimating semiparametric shared frailty models. *Journal of Statistical Software*, 90(7):1–29, 2019. doi: 10.18637/jss.v090.i07. [p196]

M. Beyersmann, J. Schumacher and A. Allignol. *Competing Risks and Multistate Models with R*. Springer, New York, 2012. [p206]

O. Borgan. *Encyclopedia of biostatistics: Aalen-Johansen estimator*. John Wiley & Sons, 2005. [p198]

S. Chiou, J. Qian, E. Mormino, and R. Betensky. Permutation tests for general dependent truncation. *Computational Statistics & Data Analysis*, 318:308–324, 2018. doi: 10.1016/j.csda.2018.07.012. [p196]

S. Datta and G. Satten. Validity of the aalen-johansen estimators of stage occupation probabilities and nelson aalen integrated transition hazards for non-markov models. *Statistics & Probability Letters*, 55: 403–411, 2001. [p196, 198]

J. de Uña-Álvarez and L. Meira-Machado. Nonparametric estimation of transition probabilities in the non-markov illness-death model: A comparative study. *Biometrics*, 71(2):364–375, 2015. ISSN 0006-341X. [p196, 198, 199]

G. L. Hickey, P. Philipson, A. Jorgensen, and R. Kolamunnage-Dona. *joineRML: a joint model and software package for time-to-event and multivariate longitudinal outcomes*, 2018. [p197]

P. Hougaard. *Analysis of Multivariate Survival Data*. Statistics for Biology and Health. Springer-Verlag, New York, 2000. [p195]

C. Jackson. Multi-state models for panel data: The msm package for r. *Journal of Statistical Software*, 38:8:1–28, 2011. [p196]

R. Kay. A markov model for analyzing cancer markers and disease states in survival studies. *Biometrics*, 42(4):457–481, 1986. [p196, 198]

A. Król and P. Saint-Pierre. SemiMarkov: An R package for parametric estimation in multi-state semi-markov models. *Journal of Statistical Software*, 66(6):1–16, 2015. URL `http://www.jstatsoft.org/v66/i06/`. [p196]

R. Li and L. Peng. *Handbook of Quantile Regression*. Chapman and Hall/CRC, London, 2017. [p195]

L. Meira-Machado and M. Sestelo. Estimation in the progressive illness-death model: A nonexhaustive review. *Biometrical Journal*, 61(2):245–263, 2019. doi: 10.1002/bimj.201500038. [p195, 197]

L. Meira-Machado, J. de Uña-Álvarez, and C. Cadarso-Suárez. Nonparametric estimation of transition probabilities in a non-markov illness-death model. *Lifetime Data Analysis*, 12:325–344, 2006. [p196]

L. Meira-Machado, J. de Uña-Álvarez, C. Cadarso-Suárez, and P. Andersen. Multi-state models for the analysis of time to event data. *Statistical Methods in Medical Research*, 18:195–222, 2009. [p195]

C. Moertel, T. Fleming, J. Macdonald, D. Haller, J. Laurie, C. Tangen, J. Ungerleider, W. Emerson, D. Tormey, J. Glick, M. Veeder, and J. Mailliard. Fluorouracil plus levamisole as effective adjuvant therapy after resection of stage iii. colon carcinoma: A final report. *The Annals of Internal Medicine*, 122(5):321–326, 1995. [p200]

C. G. Moertel, T. R. Fleming, J. S. Macdonald, D. G. Haller, J. A. Laurie, P. J. Goodman, J. S. Ungerleider, W. A. Emerson, D. C. Tormey, J. H. Glick, M. H. Veeder, and J. A. Mailliard. Levamisole and fluorouracil for adjuvant therapy of resected colon carcinoma. *New England Journal of Medicine*, 322 (6):352–358, 1990. [p200]

D. Nevo and M. Gorfine. Causal inference for semi-competing risks data. *arXiv*, 2020. doi: arXiv: 2010.04485. URL `https://doi.org/10.48550/arXiv.2010.04485`. [p195]

P. Philipson, I. Sousa, P. J. Diggle, P. Williamson, R. Kolamunnage-Dona, R. Henderson, and G. L. Hickey. *joineR: Joint Modelling of Repeated Measurements and Time-to-Event Data*, 2018. URL `https://github.com/graemeleehickey/joineR/`. R package version 1.2.6. [p196]

H. Putter and C. Spitoni. Non-parametric estimation of transition probabilities in non-markov multi-state models: The landmark aalen-johansen estimator. *Statistical Methods in Medical Research*, 27: 2081–2092, 2018. [p196, 198, 199]

H. Putter, M. Fiocco, and R. B. Geskus. Tutorial in biostatistics: Competing risks and multi-state models. *Statistics in Medicine*, 26(11):2389–2430, 2007. [p195, 200, 207]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL `http://www.R-project.org/`. [p200]

D. Rizopoulos. JM: An R package for the joint modelling of longitudinal and time-to-event data. *Journal of Statistical Software*, 35(9):1–33, 2010. URL `https://doi.org/10.18637/jss.v035.i09`. [p196]

M. Rodriguez-Girondo and J. de Uña-Álvarez. A nonparametric test for markovianity in the illness-death model. *Statistics in Medicine*, 31(30):4416–4427, 2012. doi: 10.1002/sim.5619. [p196]

M. Rodriguez-Girondo and J. de Uña-Álvarez. Methods for testing the markov condition in the illness-death model: a comparative study. *Statistics in Medicine*, 35(20):3549–3562, 2016. doi: 10.1002/sim.6940. [p196]

G. Soutinho and L. Meira-Machado. Methods for checking the markov condition in multi-state survival data. *Computational Statistics*, 2021. doi: 10.1007/s00180-021-01139-7. [p196, 197, 198, 199, 208]

G. Soutinho, M. Sestelo, and L. Meira-Machado. survidm: An R package for Inference and Prediction in an Illness-Death Model. *The R Journal*, 13(2):70–89, 2021. doi: 10.32614/RJ-2021-070. URL `https://doi.org/10.32614/RJ-2021-070`. [p196]

A. Titman and H. Putter. General tests of the markov property in multi-state models. *Bio-statistics*, 2020. doi: 10.1093/biostatistics/kxaa030. [p196, 198, 199, 200, 206, 209]

H. C. Van Houwelingen. Dynamic prediction by landmarking in event history analysis. *Scandinavian Journal of Statistics*, 34(1):70–85, 2007. [p198]

C. Wu. Jackknife, bootstrap, and other resampling methods in regression analysis. *Annals of Statistics*, 14(4):1261–1295, 1986. [p206]

J. Xu, J. Kalbfleisch, and B. Tai. Statistical analysis of illness-death processes and semicompeting risks data. *Biometrics*, 66:716–727, 2010. doi: 10.1111/j.1541-0420.2009.01340.x. URL https://doi.org/10.1111/j.1541-0420.2009.01340.x. [p195]

Gustavo Soutinho
EPIUnit, Institute of Public Health of the University of Porto (ISPUP)
Rua das Taipas 135, 4050-600 Porto
Portugal
ORCID: 0000-0002-0559-1327
gdsoutinho@gmail.com

Luís Meira-Machado
Department of Mathematics & Centre of Mathematics, University of Minho
Campus de Azurém - 4800-058 Guimarães
Portugal
ORCID: 0000-0002-8577-7665
lmachado@math.uminho.pt

Robust Functional Linear Regression Models

by Ufuk Beyaztas and Han Lin Shang

Abstract With advancements in technology and data storage, the availability of functional data whose sample observations are recorded over a continuum, such as time, wavelength, space grids, and depth, progressively increases in almost all scientific branches. The functional linear regression models, including scalar-on-function and function-on-function, have become popular tools for exploring the functional relationships between the scalar response-functional predictors and functional response-functional predictors, respectively. However, most existing estimation strategies are based on non-robust estimators that are seriously hindered by outlying observations, which are common in applied research. In the case of outliers, the non-robust methods lead to undesirable estimation and prediction results. Using a readily-available R package **robflreg**, this paper presents several robust methods build upon the functional principal component analysis for modeling and predicting scalar-on-function and function-on-function regression models in the presence of outliers. The methods are demonstrated via simulated and empirical datasets.

1 Introduction

The interest and need for developing statistical methods for analyzing functional data have increased in the last few decades. There have been many recent theoretical and applied developments in functional data analysis tools (see Ramsay and Dalzell, 1991; Ramsay and Silverman, 2002, 2006; Ferraty and Vieu, 2006; Horváth and Kokoszka, 2012; Cuevas, 2014; Hsing and Eubank, 2015; Marron et al., 2015; Srivastava and Klassen, 2016; Dryden and Mardia, 2016; Kokoszka and Reimherr, 2017). Among many others, the scalar-on-function linear regression model (SFLRM), where the response is scalar-valued and predictor(s) consist of random functions (see Cardot et al., 1991, 2003; James, 2002; Reiss and Ogden, 2007; Chen et al., 2011; Jiang and Wang, 2011; Goldsmith et al., 2011; Dou et al., 2012; Tucker et al., 2019; Ahn et al., 2020; Beyaztas and Shang, 2022), and the function-on-function linear regression model (FFLRM), where both the response and predictor(s) consist of random curves (see Yao et al., 2005; Harezlak et al., 2007; Şentürk and Müller, 2008; Matsui et al., 2009; Ivanescu et al., 2015; Scheipl et al., 2015; Chiou et al., 2016; Beyaztas and Shang, 2020), have received considerable attention among researchers as a tool for exploring the functional relationship between a scalar response-functional predictors and functional response-functional predictors, respectively. In addition, please see the available R packages **fda** (Ramsay et al., 2022) and **refund** (Goldsmith et al., 2022) for the implementation of many functional data analysis methods, including SFLRM and FFLRM.

Most of the existing methods developed to estimate the SFLRM and FFLRM are non-robust to outlying observations, i.e. observations which are generated by a stochastic process with a distribution different from that of the rest of the observations (see, e.g., Raña et al., 2015). In the case of outliers, the non-robust methods produce biased estimates; thus, predictions obtained from the fitted models become unreliable (see, e.g., Zhu et al., 2011; Maronna and Yohai, 2013; Shin and Lee, 2016; Kalogridis and Aelst, 2019; Boente et al., 2020; Hullait et al., 2021; Beyaztas and Shang, 2022). These methods may also lack robustness because they are coss-sectional and the fitted mean of the response variable may not be representative of the underlying data generating process. In this paper, we provide a hands-on tutorial for the implementation of functional principal component regression based on several robust approaches (available in the R package **robflreg**), for robustly modeling and predicting the SFLRM and FFLRM in the presence of outliers.

The methods available in the **robflreg** package are centered on the robust functional principal component analysis (RFPCA) approach of Bali et al. (2011). It uses the robust projection pursuit approach of Croux and Ruiz-Gazen (1996) combined with a robust scale estimator to produce functional principal components and the corresponding principal component scores. With this approach, the infinite-dimensional SFLRM and FFLRM are projected onto a finite-dimensional space of RFPCA bases. Then, for the SFLRM, the robust estimation methods, including the least trimmed squares (LTS) of Rousseeuw (1984), MM-type regression estimator (MM) of Yohai (1987) and Koller and Stahel (2011), S estimator, and the tau estimator of Salibian-Barrera et al. (2008), are used to estimate the parameter vector of regression model, where the scalar-valued response is predicted by the robust principal component scores of the functional predictors. For the FFLRM, on the other hand, the robust estimation methods, including the minimum covariance determinant estimator (MCD) of Rousseeuw et al. (2004), multivariate least trimmed squares estimator (MLTS) of Bali et al. (2008), MM estimator of Kudraszow and Moronna (2011), S estimator of Bilodeau and Duchesne (2000), and the tau estimator of Ben et al. (2006), are used to estimate the parameter matrix of the regression model between the robust principal component scores of the functional response and the functional predictor variables. Besides the robust procedures, the package **robflreg** allows for non-robust estimation of the functional principal component regression model using the classical functional principal component analysis (FPCA) of Ramsay and Silverman (2006) and the least-squares estimator.

The remainder of this paper is organized as follows. The SFLRM and FFLRM, as well as the techniques used for modeling and predicting these regression models, are reviewed and implemented using the **robflreg** package. Some ideas on how the available R package **robflreg** can be further improved are given at the end.

2 Functional linear regression models

We present the SFLRM and FFLRM, respectively.

The SFLRM

We consider a random sample $\{Y_i, \boldsymbol{\mathcal{X}}_i(s) : i = 1, \ldots, n\}$ from the pair $(Y, \boldsymbol{\mathcal{X}})$, where $Y \in \mathbb{R}$ is the scalar response and $\boldsymbol{\mathcal{X}} = [\mathcal{X}_1(s), \ldots, \mathcal{X}_P(s)]^\top$ with $\mathcal{X}_p(s) \in \mathcal{L}_2[0, \mathcal{I}]$, $\forall\ p = 1, \ldots, P$ is the vector of P set of functional predictors whose sample elements are denoted by curves belonging to $\mathcal{L}_2$ Hilbert space, denoted by $\mathcal{H}$, with bounded and closed interval $s \in \mathcal{I}$. We assume that the functional predictors $\mathcal{X}_p(s)$, for $p = 1, \ldots, P$, have finite second-order moments, i.e., $\text{E}[\|\mathcal{X}_p(s)\|] < \infty$. Without loss of generality, we also assume that Y and $\mathcal{X}_p(s)$, for $p = 1, \ldots, P$, are mean-zero processes, so that $\text{E}[Y] = \text{E}[\mathcal{X}_p(s)] = 0$ and $s \in [0, 1]$. Then, the SFRM is defined as follows:

$$Y_i = \int_0^1 \boldsymbol{\mathcal{X}}_i^\top(s) \boldsymbol{\beta}(s) ds + \epsilon_i, \tag{1}$$

where $\beta_p(s) \in \mathcal{L}_2[0, 1]$ is the regression coefficient function linking Y with $\mathcal{X}_p(s)$, and $\boldsymbol{\beta}(s) = [\beta_1(s), \ldots, \beta_P(s)]^\top \in \mathcal{L}_2^P[0, 1]$, and ϵ_i is the error term which is assumed to follow a Gaussian distribution with mean-zero and variance σ^2.

Simulation of a dataset for the SFLRM

The function `generate.sf.data()` in the package **robflreg** allows to simulate a dataset for the SFRM (1) as follows:

```
generate.sf.data(n, n.pred, n.gp, out.p = 0)
```

Here, the argument `n` denotes the number of observations for each variable to be generated, `n.pred` denotes the number of functional predictors to be generated, `n.gp` denotes the number of grid points (i.e. the number of breaks in a fine grid on the interval $[0, 1]$), and `out.p` is a number between 0 and 1, denoting percentage of outliers in the generated data. In the data generation process, first, `generate.sf.data()` simulates the functional predictors based on the following process:

$$\mathcal{X}(s) = \sum_{j=1}^{5} \kappa_j \nu_j(s),$$

where κ_j is a vector generated from a Normal distribution with mean one and variance $\sqrt{a} j^{-3/2}$, where a is is a uniformly generated random number between 1 and 4, and

$$\nu_j(s) = \sin(j\pi s) - \cos(j\pi s).$$

The regression coefficient functions are generated from a coefficient space that includes ten different functions, such as $b\sin(2\pi t)$ and $b\cos(2\pi t)$, where b is generated from a uniform distribution between 1 and 3. The error process is generated from the standard normal distribution. Finally, the scalar response is obtained using (1). Suppose outliers are allowed in the generated data, i.e., $out.p > 0$. Then, the randomly selected $n \times out.p$ cases are generated differently from the process mentioned above. In more detail, if $out.p > 0$, the regression coefficient functions (possibly different from the previously generated coefficient functions) generated from the coefficient space with b^* (instead of b), where b^* is generated from a uniform distribution between 3 and 5, are used to generate the outlying observations. Further, in this case, the following process is used to generate functional predictors:

$$\mathcal{X}^*(s) = \sum_{j=1}^{5} \kappa_j^* \nu_j^*(s),$$

where κ_j^* is a vector generated from a Normal distribution with mean one and variance $\sqrt{a}j^{-1/2}$ and

$$\nu_j^*(s) = 2\sin(j\pi s) - \cos(j\pi s).$$

Moreover, the error process is generated from a normal distribution with mean of zero and a variance of one. A graphical display of the generated dataset with five functional predictors and `n = 400` observations at 101 equally spaced points in the interval $[0, 1]$ obtained by `generate.sf.data()` is presented in Figure 1.

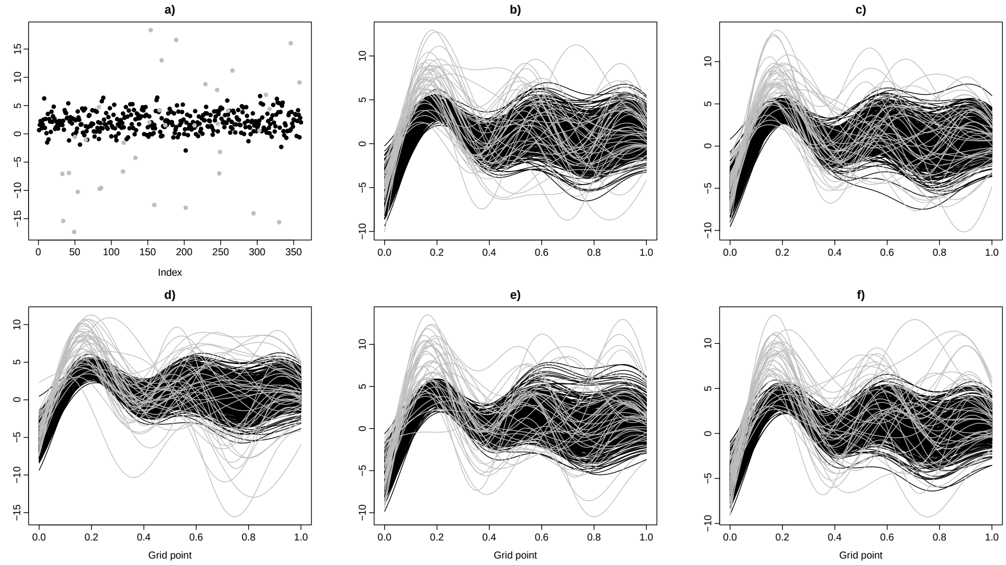


Figure 1: Plots of the simulated scalar response and the functional predictor variables. a) Observations generated under the main data generating process (black) and outlier points (grey). b-f) Curves of the functional predictors generated under the main data generating process (black) and outlier curves (grey).

Figure 1 can be produced by the following code (refer to the supplement code file for all the reproducible code):

```
library(robflreg)
library(fda.usc)
set.seed(202)

# Generate a dataset with five functional predictors and 400
# observations at 101 equally spaced point in the interval [0, 1]
# for each variable for the scalar-on-function regression model
sim.data <- generate.sf.data(n = 400, n.pred = 5, n.gp = 101, out.p = 0.1)

# Response variable
Y <- sim.data$Y
# Predictors
X <- sim.data$X
# Regression coefficient functions
coeffs <- sim.data$f.coef
# Plot the scalar response
out.indx <- sim.data$out.indx
plot(Y[-out.indx,], type = "p", pch = 16, xlab = "Index", ylab = "",
main = "Response", ylim = range(Y))
points(out.indx, Y[out.indx,], type = "p", pch = 16, col = "blue")
# Plot the first functional predictor
fX1 <- fdata(X[[1]], argvals = seq(0, 1, length.out = 101))
plot(fX1[-out.indx,], lty = 1, ylab = "", xlab = "", col = "black",
     main = expression(X[1](s)), mgp = c(2, 0.5, 0), ylim = range(fX1))
lines(fX1[out.indx,], lty = 1, col = "grey")
```

The FFLRM

Let us consider a random sample $\{\mathcal{Y}_i(t), \boldsymbol{\mathcal{X}}_i(s) \colon i = 1, 2, \ldots n\}$ from the pair $(\mathcal{Y}, \boldsymbol{\mathcal{X}})$, where $\mathcal{Y} \in \mathcal{L}_2[0,1]$ is the functional response and $\boldsymbol{\mathcal{X}} = [\mathcal{X}_1(s), \ldots, \mathcal{X}_P(s)]^\top$ with $\mathcal{X}_p(s) \in \mathcal{L}_2[0,1]$, $\forall\ p = 1, \ldots, P$ is the vector of P set of functional predictors. We assume that the functional response and functional predictors have finite second-order moments, i.e., $\mathrm{E}[\|\mathcal{Y}(t)\|] = \mathrm{E}[\|\mathcal{X}_p(s)\|] < \infty$, for $p = 1, \ldots, P$. Without loss of generality, we also assume that both $\mathcal{Y}(t)$ and $\mathcal{X}_p(s)$, for $p = 1, \ldots, P$, are mean-zero processes, so that $\mathrm{E}[Y(t)] = \mathrm{E}[\mathcal{X}_p(s)] = 0$. Then, the FFRM is defined as follows:

$$Y_i(t) = \int_0^1 \boldsymbol{\mathcal{X}}_i^\top(s) \boldsymbol{\beta}(s,t) ds dt + \epsilon_i(t), \tag{2}$$

where $\beta_p(s,t) \in \mathcal{L}_2[0,1]$ is the bivariate regression coefficient function linking $\mathcal{Y}(t)$ with $\mathcal{X}_p(s)$, and $\boldsymbol{\beta}(s,t) = [\beta_1(s,t), \ldots, \beta_P(s,t)]^\top \in \mathcal{L}_2^P[0,1]$, and $\epsilon_i(t) \in \mathcal{L}_2[0,1]$ is the error term which is assumed to be independent of $\mathcal{X}_p(s)$, for $p = 1, \ldots, P$ and $\mathrm{E}[\epsilon_i(t)] = 0$.

Simulation of a dataset for the FFLRM

The function `generate.ff.data()` allows for the simulation of a dataset for the FFLRM as follows:

```
generate.ff.data(n.pred, n.curve, n.gp, out.p = 0)
```

Similar to the `generate.sf.data()`, `n.pred` denotes the number of functional predictors to be generated, `n.curve` denotes the number of observations for each functional variable to be generated, `n.gp` denotes the number of grid points, and `out.p` is an integer between 0 and 1, denoting the outlier percentage in the generated data. When generating a dataset, first, the function `generate.ff.data()` first simulates the functional predictors via the following process:

$$\mathcal{X}(s) = \sum_{j=1}^{5} \kappa_j \nu_j(s),$$

where κ_j is a vector generated from a Normal distribution with mean one and variance $\sqrt{a} j^{-1/2}$, where a is is a uniformly generated random number between 1 and 4, and

$$\nu_j(s) = \sin(j\pi s) - \cos(j\pi s).$$

The bivariate regression coefficient functions are generated from a coefficient space that includes ten different functions such as $b \sin(2\pi s) \sin(\pi t)$ and $b e^{-3(s-0.5)^2} e^{-4(t-1)^2}$, where b is generated from a uniform distribution between 1 and 3. The error process $\epsilon(t)$, on the other hand, is generated from the Ornstein-Uhlenbeck process:

$$\epsilon(t) = l + [\epsilon_0(t) - l] e^{-\theta t} \sigma \int_0^t e^{-\theta(t-u)} dW_u,$$

where $l, \theta > 0, \sigma > 0$ are real constants, $\epsilon_0(t)$ is the initial value of $\epsilon(t)$ taken from W_u, and W_u is the Wiener process. If outliers are allowed in the generated data, i.e., $out.p > 0$, then, the randomly selected $n \times out.p$ cases are generated differently from the process mentioned above. In more detail, if $out.p > 0$, the regression coefficient functions (possibly different from the previously generated coefficient functions) generated from the coefficient space with b^* (instead of b), where b^* is generated from a uniform distribution between 1 and 2, are used to generate the outlying observations. In addition, in this case, the following process is used to generate functional predictors:

$$\mathcal{X}^*(s) = \sum_{j=1}^{5} \kappa_j^* \nu_j^*(s),$$

where κ_j^* is a vector generated from a Normal distribution with mean one and variance $\sqrt{a} j^{-3/2}$ and

$$\nu_j^*(s) = 2\sin(j\pi s) - \cos(j\pi s).$$

A graphical display of the generated dataset with five functional predictors and `n = 200` observations at 101 equally spaced points in the interval $[0,1]$ obtained by `generate.ff.data()` is presented in Figure 2.

Figure 2 can be produced by the following code (refer to the supplement code file for all the reproducible code):

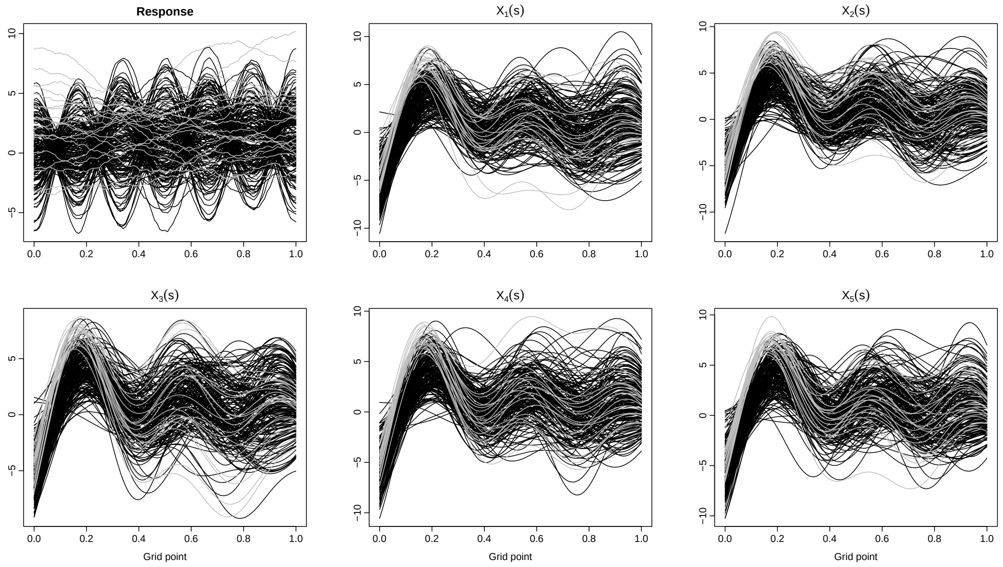


Figure 2: Plots of the simulated functional response and functional predictor variables: In the plots, the black curves denote the observations of the response and functional predictors generated under the smooth data generation process. The grey curves denote the outlying observations in the functional variables.

```
library(robflreg)
library(fda.usc)
set.seed(202)

# Generate a dataset with five functional predictors and 200
# observations at 101 equally spaced point in the interval [0, 1]
# for each variable for the function-on-function regression model
sim.data <- generate.ff.data(n.pred = 5, n.curve = 200, n.gp = 101, out.p = 0.1)

# Response variable
Y <- sim.data$Y
# Predictors
X <- sim.data$X
# Regression coefficient functions
coeffs <- sim.data$f.coef
# Plot the scalar response
out.indx <- sim.data$out.indx
fY <- fdata(Y, argvals = seq(0, 1, length.out = 101))
plot(fY[-out.indx,], lty = 1, ylab = "", xlab = "",
main = "Response", mgp = c(2, 0.5, 0), ylim = range(fY))
lines(fY[out.indx,], lty = 1, col = "grey")
# Plot the first functional predictor
fX1 <- fdata(X[[1]], argvals = seq(0, 1, length.out = 101))
plot(fX1[-out.indx,], lty = 1, ylab = "", xlab = "", col = "black",
main = expression(X[1](s)), mgp = c(2, 0.5, 0), ylim = range(fX1))
lines(fX1[out.indx,], lty = 1, col = "grey")
```

3 Estimation

We first review the classical and robust FPCA methods. Then, we will focus on the robust estimation of the SFLRM and FFLRM by the functional principal component regression.

Functional principal component analysis (FPCA)

For a functional random variable $\mathcal{X}(s)$, let us denote its covariance function by $\mathcal{C}(s_1, s_2) = \text{Cov}[\mathcal{X}(s_1), \mathcal{X}(s_2)]$ satisfying $\int_0^1 \int_0^1 \mathcal{C}(s_1, s_2) ds_1 ds_2 < \infty$. Then, by Mercer's Theorem, the following representation holds:

$$\mathcal{C} = \sum_{k=1}^{\infty} \kappa_k \psi_k(s_1) \psi_k(s_2), \quad \forall s_1, s_2 \in [0, 1],$$

where $\{\psi_k(s) : k = 1, 2, \ldots\}$ are orthonormal bases of eigenfunctions in $\mathcal{L}_2[0, 1]$ corresponding to the non-negative eigenvalues $\{\kappa_k : k = 1, 2, \ldots\}$ with $\kappa_k \geq \kappa_{k+1}$. In practice, most of the variability in functional variables can be captured via a finite number of the first K eigenfunctions. Thus, the covariance function of a functional variable is estimated using a pre-determined truncation constant K. In addition, the orthonormal bases of eigenfunctions are unknown in practice. Thus, they are approximated via a suitable basis expansion method such as B-spline, which is used in the **robflreg** package.

The RFPCA of Bali et al. (2011) follows a similar structure as the classical FPCA, but it uses a robust scale functional instead of variance. Now let $\|\alpha\|^2 = \langle \alpha, \alpha \rangle$ denote the norm generated by the inner product $\langle \cdot, \cdot \rangle$. Also, let $\mathcal{F}[\alpha]$ denote the distribution of $\langle \alpha, \mathcal{X} \rangle$ where $\mathcal{F}$ is the distribution of $\mathcal{X}$. Then, for a given M-scale functional $\sigma_M(\mathcal{F})$, the orthonormal bases of eigenfunctions defined by Bali et al. (2011) are as follows:

$$\begin{cases} \psi_k(\mathcal{F}) = \underset{\|\alpha\|^2=1}{\arg\max}\, \sigma_M(\mathcal{F}[\alpha]), & k = 1, \\ \psi_k(\mathcal{F}) = \underset{\|\alpha\|^2=1, \alpha \in \mathcal{B}_k}{\arg\max}\, \sigma_M(\mathcal{F}[\alpha]), & k \geq 2, \end{cases}$$

where $\mathcal{B}_k = \{\alpha \in \mathcal{L}_2[0, 1] : \langle \alpha, \psi_k(\mathcal{F}) \rangle = 0,\ 1 \leq k \leq\ K - 1\}$. The k-th largest eigenvalue is given by:

$$\kappa_k(\mathcal{F}) = \sigma_M^2(\mathcal{F}[\psi_k]) = \max_{\|\alpha\|^2=1, \alpha \in \mathcal{B}_k} \sigma_M^2(\mathcal{F}[\alpha]).$$

Denote by $\sigma_M(\mathcal{F}_n[\alpha])$ the functional for σ_M. Let $s_n^2 : \mathcal{L}_2[0, 1] \rightarrow \mathbb{R}$ denote the function of empirical M-scale functional such that $s^2(\alpha) = \sigma_M^2(\mathcal{F}[\alpha])$. Then, the RFPCA estimates of the orthonormal bases of eigenfunctions for $\mathcal{X}(s)$ are given by

$$\begin{cases} \widehat{\psi}_k(s) = \underset{\|\alpha\|^2=1}{\arg\max}\, s_n(\alpha), & k = 1, \\ \widehat{\psi}_k(s) = \underset{\alpha \in \widehat{\mathcal{B}}_k}{\arg\max}\, s_n(\alpha), & k \geq 2, \end{cases}$$

where $\widehat{\mathcal{B}}_k = \left\{\alpha \in \mathcal{L}_2[0, 1] : \|\alpha\| = 1, \langle \alpha, \widehat{\psi}_k \rangle = 0,\ \forall\, 1 \leq k \leq\ K - 1\right\}$. The corresponding eigenvalues, on the other hand, are given by

$$\widehat{\kappa}_k = s_n^2(\widehat{\psi}_k), \quad k \geq 1.$$

Main RFPCA function and its arguments

The main function to obtain the robust estimates of functional principal components and the corresponding principal component scores is called `getPCA()`:

```
getPCA(data, nbasis, ncomp, gp, emodel = c("classical", "robust"))
```

In the `getPCA()` function, the data set is provided in the `data` argument as a matrix. The grid points of the functional predictors are provided in the `gp` argument as a vector. `nbasis` denotes the number of B-spline basis expansion functions used to approximate the robust functional principal components. `ncomp` specifies the number of functional principal components to be computed. The argument `emodel` denotes the method used for functional principal component decomposition. If `emodel = "classical"`, then the classical functional principal component decomposition is performed. On the other hand, if `emodel = "robust"`, then the RFPCA method of Bali et al. (2011) is used to obtain the functional principal components and the corresponding principal component scores. Figure 3 presents the plot of five functional principal components computed from simulated functional data using RFPCA and `nbasis = 20` B-spline basis expansion functions.

Figure 3 can be produced by the following code:

```
library(robflreg)
# Generate a dataset with five functional predictors and 200
```

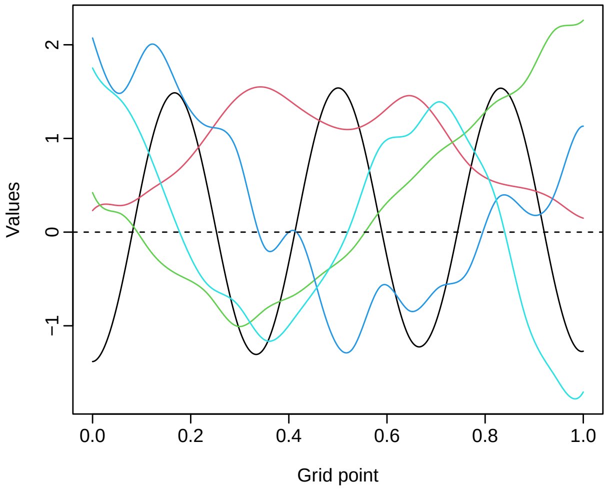


Figure 3: Plot of the first five functional principal components of the simulated functional data. The principal components were obtained using the RFPCA and different colors correspond to different principal components.

```
# observations at 101 equally spaced point in the interval [0, 1]
# for each variable for the function-on-function regression model
set.seed(202)
sim.data <- generate.ff.data(n.pred = 5, n.curve = 200, n.gp = 101)
# Response variable
Y <- sim.data$Y
gpY <- seq(0, 1, length.out = 101) # grid points

# Perform robust functional principal component analysis on the response variable Y
rob.fpca <- getPCA(data = Y, nbasis = 20, ncomp = 5, gp = gpY, emodel = "robust")

# Principal components
PCs <- rob.fpca$PCAcoef

plot(PCs, xlab = "Grid point", ylab = "Values", lty = 1)
```

Robust estimation of the SFLRM

In the robust estimation of the SFRM, we first consider the principal component decomposition of the functional predictors as follows:

$$\mathcal{X}_p(s) = \sum_{k=1}^{K_p} \xi_{pk} \psi_{pk}(s),$$

where K_p is the truncation constant for the p-th functional predictor $\mathcal{X}_p(s)$, $\psi_{pk}(s)$ is the k-th eigenfunction obtained by the RFPCA of Bali et al. (2011), and ξ_{pk} is the corresponding principal component score, given by:

$$\xi_{pk} = \int_0^1 \mathcal{X}_p(s) \psi_{pk}(s) ds.$$

In practice, the eigenfunctions are approximated via a basis expansion function such as B-spline. Let $\varphi_p(s)$ denote the B-spline basis expansion function, and $A_p = (a_{pl})$ be an $n \times L$-dimensional matrix of basis expansion coefficients for the p-th functional predictor variable. In addition, let $\boldsymbol{\varphi} = \int_0^1 \varphi_p(s) \varphi_p^\top(s) ds$ and $\boldsymbol{\varphi}^{1/2}$ denote the $L \times L$ dimensional matrix of inner products between the basis expansion functions and its square root, respectively. Then, the infinite-dimensional RFPCA of $\mathcal{X}_p(s)$ is equivalent to the multivariate principal component analysis of $A_p \boldsymbol{\varphi}^{1/2}$ and the k-th eigenfunction is given by $\psi_{pk}(s) = \boldsymbol{\varphi}^{-1/2} v_{pk}$, where v_{pk} denotes the p-th eigenvector of the sample covariance matrix of $A_p \boldsymbol{\varphi}^{1/2}$ (see, e.g., Ocana et al., 2007, for more information).

If we assume that the p-th regression coefficient function $\beta_p(s)$ admits the similar functional principal decomposition as the functional predictors as follows:

$$\beta_p(s) = \sum_{k=1}^{K_p} b_{pk}\psi_{pk}(s),$$

where $b_{pk} = \int_0^1 \beta_p(s)\psi_{pk}(s)ds$. Then, the infinite-dimensional SFRM in (1) is approximated by the finite-dimensional regression model of scalar response on all the functional principal component scores as follows:

$$Y = \sum_{p=1}^{P}\sum_{k=1}^{K_p} b_{pk}\xi_{pk}.$$

Main functions for the robust estimation of a SFRM and their arguments

The main function for robust estimation of SFRMs is called `rob.sf.reg()`:

```
rob.sf.reg(Y, X, X.scl = NULL, emodel = c("classical", "robust"),
fmodel = c("LTS", "MM", "S", "tau"), nbasis = NULL, gp = NULL, ncomp = NULL)
```

In the `rob.sf.reg()` function, the scalar response is provided in the `Y` argument as an $n \times 1$-dimensional column vector, where n is the sample size. The functional predictors, on the other hand, are provided as a list object in the `X` argument. Each element of `X` is an $n \times L_p$-dimensional matrix containing the observations of p-th functional predictor $\mathcal{X}_p(s)$, where L_p is the number of grid points for $\mathcal{X}_p(s)$. The `rob.sf.reg()` function also allows for scalar predictors, which can be provided in the `X.scl` as an $n \times R$-dimensional matrix, where R denotes the number of scalar predictors. In this case, the following SFRM is considered:

$$Y_i = \int_0^1 \boldsymbol{\mathcal{X}}_i^\top(s)\boldsymbol{\beta}(s)ds + \text{X.scl}_i\gamma + \epsilon_i,$$

where γ denotes the vector of coefficients for the scalar predictors' matrix. The functional principal component decomposition method is provided in the `emodel` argument. If `emodel = "classical"`, then the classical functional principal component decomposition is performed to obtain principal components and the corresponding principal component scores. The coefficient vector of the regression problem of scalar response on the principal component scores is estimated via the least-squares method. If `emodel = "robust"`, then the RFPCA of Bali et al. (2011) is performed to obtain the principal components and the corresponding principal component scores. In this case, the method used to estimate the coefficient vector of the regression problem constructed by the scalar response and principal component scores is provided in the `fmodel` argument. One of the methods among LTS, MM, S, and tau can be chosen to estimate the parameter vector (by specifying, for example, fmodel = tau"). The number of B-spline basis expansion functions used to approximate the functional principal components are provided in the `nbasis` argument as a vector with length p. Suppose `nbasis = NULL`, then $\min(20, L_p/4)$ number of B-spline basis expansion functions are used for each functional predictor. The grid points for the functional predictors are provided in the `gp` argument as a list object. The p-th element of `gp` is a vector containing the grid points of the p-th functional predictor $\mathcal{X}_p(s)$. If `gp = NULL`, then L_p equally spaced time points in the interval $[0, 1]$ are used for the p-th functional predictor. The number of principal components to be computed for the functional predictors are provided in the `ncomp` argument as a vector with length P. If `ncomp = NULL`, then, for each functional predictor, the number whose usage results in at least 95% explained variation is used as the number of principal components.

The function `get.sf.coeffs()` can be used to obtain the estimated regression coefficient functions from a fitted functional principal component regression:

```
get.sf.coeffs(object)
```

In this function, the argument `object` is the output object obtained using the function `rob.sf.reg()`. The function `get.sf.coeffs()` produces a list object whose p-th element is a vector with length L_p containing the p-th regression coefficient function $\beta_p(s)$.

The plots of the estimated regression coefficient functions can be obtained using the function `plot_sf_coeffs()`:

```
plot_sf_coeffs(object, b)
```

In Figure 4, the plots of the regression coefficient functions obtained from simulated data (outlier-contaminated) using `RFPCA` and MM estimator, as well as the classical functional principal component regression, are presented. In this function, the argument `object` is the output object obtained by the function `get.sf.coeffs()`. On the other hand, the argument `b` is an integer value indicating which regression parameter function to plot. It is clear from this figure that, compared with the classical functional principal component regression, the robust approach produces estimated parameter functions which are more similar to the true parameter functions.

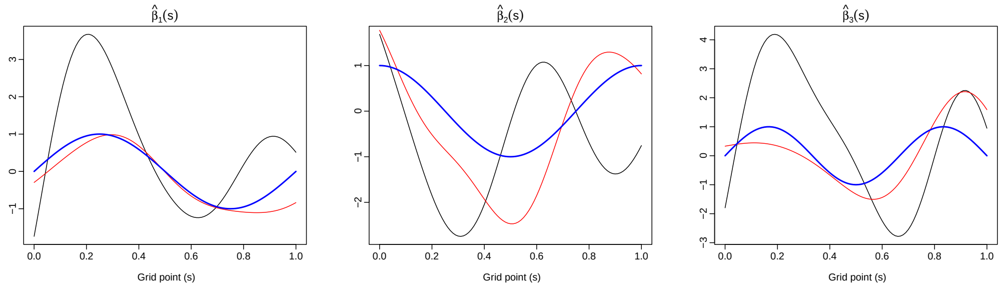


Figure 4: A plot of the estimated regression coefficient functions obtained from simulated data (outlier-contaminated) using RFPCA and MM estimator and the classical functional principal component regression. Blue lines denote the true parameter functions. On the other hand, black and red lines denote the estimated parameter functions by the classical functional principal component regression and robust MM-based functional principal component regression, respectively.

The following code can be used to produce this figure and the results (refer to the supplement code file for all of the reproducible code):

```
library(robflreg)
# Generate a dataset with three functional predictors and 400
# observations at 101 equally spaced point in the interval [0, 1]
# for each variable for the scalar-on-function regression model
set.seed(202)
sim.data <- generate.sf.data(n = 400, n.pred = 3, n.gp = 101, out.p = 0.1)

# True parameter functions
true.b1 <- sim.data\$f.coef[[1]]
true.b2 <- sim.data\$f.coef[[2]]
true.b3 <- sim.data\$f.coef[[3]]

# Response variable
Y <- sim.data\$Y
# Predictors
X <- sim.data\$X

gp <- rep(list(seq(0, 1, length.out = 101)), 3) # grid points of Xs

# Fit a functional principal component regression model for the generated data
# using the classical functional principal component analysis method:
classical.fit <- rob.sf.reg(Y, X, emodel = "classical", gp = gp)

# Fit a functional principal component regression model for the generated data
# using the robust functional principal component analysis method and tau estimator:
robust.fit <- rob.sf.reg(Y, X, emodel = "robust", fmodel = "MM", gp = gp)

# Estimated regression coefficient functions
classical.coefs <- get.sf.coeffs(classical.fit)
robust.coefs <- get.sf.coeffs(robust.fit)

# The first estimated regression coefficient function
plot_sf_coeffs(object = classical.coefs, b = 1)
lines(gp[[1]], robust.coefs\$coefficients[[1]], col = "red")
lines(gp[[1]], true.b1, col = "blue", lwd = 2)
```

Robust estimation of the FFLRM

Let us consider the functional principal decompositions of both the functional response and functional predictor variables as follows:

$$\mathcal{Y}(t) = \sum_{k=1}^{K} \zeta_k \phi_k(t), \quad \mathcal{X}_p(s) = \sum_{j=1}^{K_p} \xi_{pj} \psi_{pj}(s),$$

where $\phi_k(t)$ and $\psi_{pj}(s)$ respectively are the k-th and j-th eigenfunctions of $\mathcal{Y}(t)$ and $\mathcal{X}_p(s)$ obtained by the RFPCA and ζ_k and ξ_{pj} are the corresponding principal component scores given by

$$\zeta_k = \int_0^1 \mathcal{Y}(t)\phi_k(t)dt, \quad \xi_{pj} = \int_0^1 \mathcal{X}_p(s)\psi_{pj}(s)ds.$$

If we assume that the p-th bivariate regression coefficient function $\beta_p(s,t)$ admits the principal component decomposition with the eigenfunctions $\phi_k(t)$ and $\psi_{pj}(s)$ as follows:

$$\beta_p(s,t) = \sum_{k=1}^{K} \sum_{j=1}^{K_p} b_{pkj} \phi_k(t) \psi_{pj}(s),$$

where $b_{pkj} = \int_0^1 \int_0^1 \beta_p(s,t)\phi_k(t)\psi_{pj}(s)dtds$. Then, the infinite-dimensional FFRM in (2) is approximated by the finite-dimensional regression model of principal component scores of the functional response on all the functional principal component scores as follows:

$$\zeta_k = \sum_{p=1}^{P} \sum_{j=1}^{K_p} b_{pkj} \xi_{pj}.$$

Finally, the following regression model is obtained for the functional response

$$\mathcal{Y}(t) = \sum_{k=1}^{K} \left(\sum_{p=1}^{P} \sum_{j=1}^{K_p} b_{pkj} \xi_{pj} \right) \phi_k(t).$$

Main functions for the robust estimation of a FFRM and their arguments

The main function to estimate the FFRM robustly is called `rob.ff.reg()`:

```
rob.ff.reg(Y, X, model = c("full", "selected"), emodel = c("classical", "robust"),
fmodel = c("MCD", "MLTS", "MM", "S", "tau"), nbasisY = NULL, nbasisX = NULL,
gpY = NULL, gpX = NULL, ncompY = NULL, ncompX = NULL)
```

In the `rob.ff.reg()` function, the functional response is provided via the `Y` argument as a matrix. On the other hand, the functional predictors are provided in the argument `X` as a list object. Each element of `X` is a matrix containing the observations of p-th functional predictor. The model type to be fitted can be chosen with `model` argument. If `model = "full"`, then all of the functional predictors are used in the model. On the other hand, if `model = "selected"`, then only the significant functional predictor variables, determined by the forward variable selection procedure of Beyaztas and Shang (2021), are used in the model. The functional principal component decomposition method is provided via the `emodel` argument. If `emodel = "classical"`, then the classical functional principal component decomposition is performed to obtain principal components and the corresponding principal component scores and the coefficient vector of the regression problem of principal component scores of the functional response on the principal component scores are estimated via the least-squares method. If `emodel = "robust"`, then the RFPCA of Bali et al. (2011) is performed to obtain the principal components and the corresponding principal component scores. In this case, the method used to estimate the coefficient matrix of the regression problem constructed by the principal component scores is provided in the `fmodel` argument. Here, one of the methods, among MCD, MLTS, MM, S, and tau estimators, can be chosen to estimate the parameter matrix. The number of B-spline basis expansion functions used to approximate the functional principal components of response and predictor variables are provided in the `nbasisY` and `nbasisX` arguments, respectively. The argument `nbasisY` is a numeric value while the argument `nbasisX` is a vector with length P. Suppose `nbasisY = NULL` and `nbasisX = NULL`, then $\min(20, L_y)$ and $\min(20, L_p)$ B-spline basis expansion functions are used to approximate the functional principal components of functional response and p-th the functional predictor, where L_y and L_p respectively denote the number of grid points for $\mathcal{Y}(t)$ and $\mathcal{X}_p(s)$. The grid points for the functional response and functional predictors are provided in the gpY and gpX arguments, respectively. The argument gpY is a vector consisting of the grid points of the functional response $\mathcal{Y}(t)$. On the other hand, the argument gpX is a list object, and its p-th element is a vector containing the grid points of the p-th functional predictor $\mathcal{X}_p(s)$. If gpY = NULL and If gpX = NULL, then equally spaced time points in the interval $[0,1]$ are used for all the functional variables. The number of functional predictors to be computed for the functional response and functional predictors are provided in the arguments ncompY and ncompX, respectively. The argument ncompY is a numeric value while the argument ncompX is a vector with length P. If ncompY = NULL and ncompX = NULL, then the number whose usage results in at least 95% explained variation is used as the number of principal components for each functional variable.

The estimated bivariate regression coefficient functions from a fitted functional principal component regression model are obtained by the get.ff.coeffs() function:

```
get.ff.coeffs(object)
```

In this function, the argument object is the output object obtained using the function rob.ff.reg(). The function get.ff.coeffs() produces a list object whose p-th element is a matrix containing the p-th bivariate regression coefficient function $\beta_p(s,t)$.

The image plots of the estimated bivariate regression coefficient functions can be obtained using the function plot_ff_coeffs():

```
plot_ff_coeffs(object, b)
```

In Figure 5, the image plots of the regression coefficient functions obtained from simulated data using RFPCA and MM estimator are presented. In this function, the argument object is the output object obtained by the function get.ff.coeffs(). The argument b is an integer value indicating which regression parameter function is to be plotted.

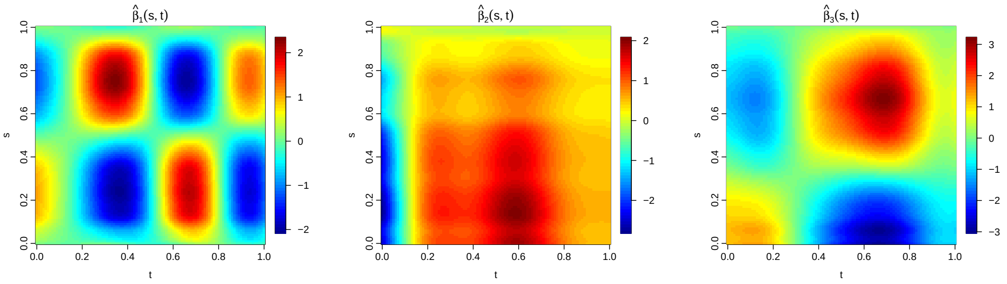


Figure 5: Image plots of the estimated bivariate regression coefficient functions obtained from simulated data using RFPCA and MM estimator.

This figure and the results can be produced by the following code (refer to the supplement code file for all the reproducible code):

```
library(robflreg)
# Generate a dataset with three functional predictors and 200
# observations at 101 equally spaced point in the interval [0, 1]
# for each variable for the function-on-function regression model
set.seed(202)
sim.data <- generate.ff.data(n.pred = 3, n.curve = 200, n.gp = 101)
# Response variable
Y <- sim.data$Y
# Predictors
X <- sim.data$X

gpY = seq(0, 1, length.out = 101) # grid points of Y
gpX <- rep(list(seq(0, 1, length.out = 101)), 3) # grid points of Xs

# Fit a functional principal component regression model for the generated data
# using the RFPCA and MM estimator:
model.fit <- rob.ff.reg(Y, X, model = "full", emodel = "robust",
fmodel = "MM", gpY = gpY, gpX = gpX)
```

```
# Estimated bivariate regression coefficient functions
coefs <- get.ff.coeffs(model.fit)
# Plot the first bivariate regression coefficient function
plot_ff_coeffs(object = coefs, b = 1)
```

Outlier detection in the functional response

Detection of outliers in functional data is an important problem (see, e.g., Sun and Genton, 2011; Arribas-Gil and Romo, 2014; Dai and Genton, 2018). From a fitted functional principal component regression for scalar response and scalar predictors, the **robflr** package with the function `rob.out.detect()` allows to detection of outliers in the functional response. This is achieved by applying the function depth-based outlier detection method of Febrero-Bande et al. (2008) together with the h-modal depth proposed by Cuaves et al. (2007) to the estimated residual functions obtained from `rob.ff.reg()` to determine the outliers in the response variable. In the outlier detection algorithm, the threshold value used to identify outliers is determined by the smoothed bootstrap procedure proposed by Febrero-Bande et al. (2008). The `rob.out.detect()` is as follows:

```
rob.out.detect(object, alpha = 0.01, B = 200, fplot = FALSE)
```

Herein, the argument `object` is an output object obtained from `rob.ff.reg()`. `alpha`, whose default value is 0.01, denotes the percentile of the distribution of the functional depth. `B` denotes the number of bootstrap samples (the default value is `B = 200`). `fplot` is a logical argument, if `fplot = TRUE`, then the outlying points flagged by the method are plotted along with the values of functional response $\mathcal{Y}(t)$.

To show how the function `rob.ff.reg()` works, we simulate an outlier-contaminated dataset for the FFRM. Then, we apply the outlier detection algorithm with the classical FPCA - least squares estimator and the RFPCA - MM estimator. The plots of the functional response and detected outlying observations are presented in Figure 6. The results show that the classical method fails to flag 13 outlying curves, while the robust procedure fails to flag only two outlying curves.

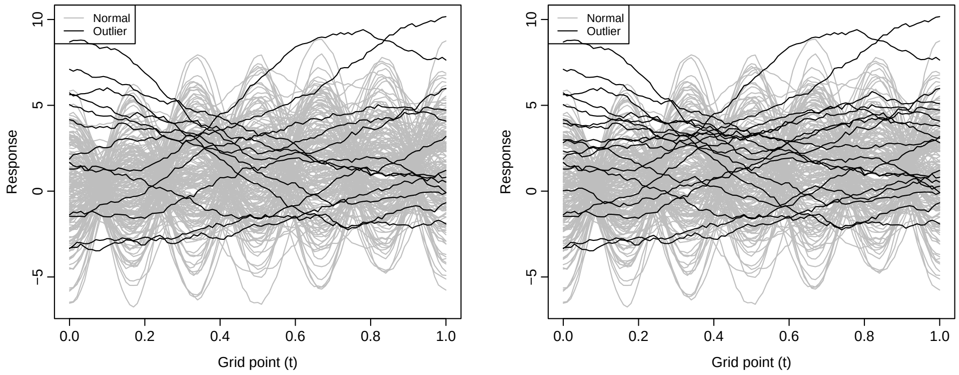


Figure 6: Plots of the functional response and detected outliers: Classical method (left panel) vs. Robust method (right panel). The detected outlying curves are denoted by black, while the non-outlying curves are denoted by grey.

The following code can produce the results and Figure 6.

```
library(robflreg)
# Generate a dataset with five functional predictors and 200
# observations at 101 equally spaced point in the interval [0, 1]
# for each variable for the function-on-function regression model
set.seed(202)
sim.data <- generate.ff.data(n.pred = 5, n.curve = 200, n.gp = 101, out.p = 0.1)
out.indx <- sim.data\$out.indx
# Response variable
Y <- sim.data\$Y
# Predictors
```

```
X <- sim.data\$X

gpY = seq(0, 1, length.out = 101) # grid points of Y
gpX <- rep(list(seq(0, 1, length.out = 101)), 5) # grid points of Xs

# Perform classical functional principal component regression using least-squares
model.classical <- rob.ff.reg(Y = Y, X = X, model = "full", emodel = "classical",
                              gpY = gpY, gpX = gpX)

# Perform robust functional principal component regression using MM-estimator
model.MM <- rob.ff.reg(Y = Y, X = X, model = "full", emodel = "robust", fmodel = "MM",
                       gpY = gpY, gpX = gpX)

# Detect outliers using rob.out.detect function
rob.out.detect(object = model.classical, fplot = TRUE)
# outlying functions are: 16 56 69 70 71 80 92 96 117 138 140 173 188
rob.out.detect(object = model.MM, fplot = TRUE)
# outlying functions are: 2 16 56 69 70 71 80 82 92 96 117 134 138 140
# 173 188 197 199

# Compare with the original outliers
sort(out.indx)
# [1] 2 16 47 56 69 70 71 80 82 92 96 117 134 138 140 162 173 188 197 199
```

4 Prediction

We review the prediction problem for a new set of functional predictors based on a fitted functional principal component regression model.

Prediction for the SFRM

When robustly predicting the unknown values of the scalar response variable for a given new set of functional predictors ($\mathcal{X}^*(s)$), the principal component scores of the new set of functional predictors (ξ^*) are obtained as follows:

$$\xi_{pk}^* = \int_0^1 \mathcal{X}_{pk}^*(s)\widehat{\psi}_{pk}(s)ds,$$

where $\widehat{\psi}_{pk}(s)$ is the k-th eigenfunction of the p-the functional predictor obtained by the RFPCA. Then, the predictions corresponding to the new set of functional predictors are obtained as follows:

$$\widehat{Y}^* = \sum_{p=1}^{P}\sum_{k=1}^{K_p} \widehat{b}_{pk}\xi_{pk}^*,$$

where $\widehat{b}_{pk}$ is the estimated parameter vector obtained from the fitted model `rob.sf.reg()`.

Main function for the robust prediction of a SFRM and its arguments

The main function for the robust prediction of a SFRM is called `predict_sf_regression()`:

```
predict_sf_regression(object, Xnew, Xnew.scl = NULL)
```

In the function `predict_sf_regression()`, the argument `object` is an output object obtained from `rob.sf.reg`. The new set of functional predictors is provided in the `Xnew` argument as a list object whose p-th element is a matrix denoting the new observations of $\mathcal{X}_p(s)$. `Xnew` must have the same length and the same structure as the input `X` of `rob.sf.reg`. If scalar predictors are used in the SFRM, then, in the prediction process, the new set of scalar predictors is provided as a matrix in the `Xnew.scl` argument. The argument `Xnew.scl` must have the same length and the same structure as the input `X.scl` of `rob.sf.reg`.

To evaluate the prediction performance of classical and robust methods, we simulate a dataset with size $n = 400$ for the SFRM. Then, the simulated data are divided into a training sample with a size of 280 and a test sample with a size of 120. Random outliers contaminate the training sample, and both the classical and robust methods with the tau estimator are applied to the training sample to predict the values of the response variable in the test sample. To compare both methods, we compute the mean squared prediction error (MSPE):

$$\text{MSPE} = \frac{1}{200}\sum_{i=1}^{200}(Y_i^* - \widehat{Y}_i^*)^2,$$

where Y_i^* and $\widehat{Y}_i^*$ denote the observed and predicted values of the scalar response in the test sample. Our results indicate that the robust method considerably outperforms the classical method. The MSPE computed from the classical method is 20.9388, while the MSPE obtained from the robust method is 1.868. The reproducible code to obtain those results is as follows:

```
library(robflreg)
# Generate a dataset with five functional predictors and 400
# observations at 101 equally spaced point in the interval [0, 1]
# for each variable for the scalar-on-function regression model
set.seed(202)
sim.data <- generate.sf.data(n = 400, n.pred = 5, n.gp = 101, out.p = 0.1)
out.indx <- sim.data\$out.indx
# Response variable
Y <- sim.data\$Y
# Predictors
X <- sim.data\$X

# Split the data into training and test samples.
indx.test <- sample(c(1:400)[-out.indx], 120)
indx.train <- c(1:400)[-indx.test]

Y.train <- Y[indx.train,]
Y.test <- Y[indx.test,]
X.train <- X.test <- list()
for(i in 1:5){
  X.train[[i]] <- X[[i]][indx.train,]
  X.test[[i]] <- X[[i]][indx.test,]
}

gp <- rep(list(seq(0, 1, length.out = 101)), 5) # grid points of Xs

# Perform classical functional principal component regression model using training samples
model.classical <- rob.sf.reg(Y.train, X.train, emodel = "classical", gp = gp)
# Perform robust functional principal component regression model
# using training samples and tau-estimator
model.tau <- rob.sf.reg(Y.train, X.train, emodel = "robust", fmodel = "tau", gp = gp)
# Predict the observations in Y.test using model.classical
pred.classical <- predict_sf_regression(object = model.classical, Xnew = X.test)
# Predict the observations in Y.test using model.tau
pred.tau <- predict_sf_regression(object = model.tau, Xnew = X.test)
# Compute mean squared errors for the test sample
round(mean((Y.test - pred.classical)^2), 4) # 2.49 (classical method)
round(mean((Y.test - pred.tau)^2), 4) # 1.1457 (tau method)
```

Prediction for the FFRM

In the robust prediction of the FFRM for a given new set of functional predictors, as in the scalar-on-function regression case, the principal component scores of the new set of functional predictors are first obtained:

$$\xi_{pk}^* = \int_0^1 \mathcal{X}_{pk}^*(s)\widehat{\psi}_{pk}(s)ds,$$

where $\widehat{\psi}_{pk}(s)$ is the k-th eigenfunction of the p-the functional predictor obtained by the RFPCA. Then, the predictions of functional response $(\widehat{\mathcal{Y}}(t))$ corresponding to the new set of functional predictors are obtained as follows:

$$\widehat{\mathcal{Y}}^*(t) = \sum_{k=1}^{K}\left(\sum_{p=1}^{P}\sum_{j=1}^{K_p}\widehat{b}_{pkj}\xi_{pj}^*\right)\widehat{\phi}_k(t),$$

where $\widehat{\phi}_k(t)$ is the k-th eigenfunction of the functional response obtained by RFPCA and $\widehat{b}_{pkj}$ is the estimated parameter matrix obtained from the fitted model `rob.ff.reg()`.

Main function for the robust prediction of a FFRM and its arguments

The main function for the robust prediction of a FFRM is called `predict_ff_regression()`:

```
predict_ff_regression(object, Xnew)
```

Here, the argument `object` is an output object obtained from `rob.ff.reg`. The new set of functional predictors is provided in the `Xnew` argument as a list object whose p-th element is a matrix denoting the new observations of $\mathcal{X}_p(s)$. `Xnew` must have the same length and the same structure as the input `X` of `rob.ff.reg`.

We simulate a dataset with size $n = 200$ for the FFRM to investigate and compare the prediction performance of the classical and robust methods. The simulated data are divided into a training sample with a size of 140 and a test sample with a size of 60. Random outliers contaminate the training sample, and both the classical and robust methods with MM estimator are applied to the training sample to predict the values of the response variable in the test sample. To compare both methods, we compute the following MSPE:

$$\text{MSPE} = \frac{1}{100} \sum_{i=1}^{200} \| \mathcal{Y}_i^*(t) - \widehat{\mathcal{Y}}_i^*(t)) \|_{\mathcal{L}_2}^2,$$

where $\mathcal{Y}_i^*(t)$ and $\widehat{\mathcal{Y}}_i^*(t))$ denote the observed and predicted values of the functional response in the test sample. Our results show that the robust method produces a significantly smaller MSPE value than the classical method. The MSPE computed from the classical method is 3.3213, while the MSPE obtained from the robust method is 0.5925. The reproducible code to obtain those results is as follows:

```
library(robflreg)
# Generate a dataset with five functional predictors and 200
# observations at 101 equally spaced point in the interval [0, 1]
# for each variable for the function-on-function regression model
set.seed(202)
sim.data <- generate.ff.data(n.pred = 5, n.curve = 200, n.gp = 101, out.p = 0.1)
out.indx <- sim.data\$out.indx
# Response variable
Y <- sim.data\$Y
# Predictor variables
X <- sim.data\$X
# Split the data into training and test samples.
indx.test <- sample(c(1:200)[-out.indx], 60)
indx.train <- c(1:200)[-indx.test]
Y.train <- Y[indx.train,]
Y.test <- Y[indx.test,]
X.train <- X.test <- list()
for(i in 1:5){
  X.train[[i]] <- X[[i]][indx.train,]
  X.test[[i]] <- X[[i]][indx.test,]
}

gpY = seq(0, 1, length.out = 101) # grid points of Y
gpX <- rep(list(seq(0, 1, length.out = 101)), 5) # grid points of Xs

# Perform classical functional principal component regression model using training samples
model.classical <- rob.ff.reg(Y = Y.train, X = X.train, model = "full",
                              emodel = "classical", gpY = gpY, gpX = gpX)
# Perform robust functional principal component regression
# using training samples and MM-estimator
model.MM <- rob.ff.reg(Y = Y.train, X = X.train, model = "full", emodel = "robust",
                       fmodel = "MM", gpY = gpY, gpX = gpX)
# Predict the functions in Y.test using model.classical
pred.classical <- predict_ff_regression(object = model.classical, Xnew = X.test)
# Predict the functions in Y.test using model.MM
```

```
pred.MM <- predict_ff_regression(object = model.MM, Xnew = X.test)
# Compute mean squared errors for the test sample
round(mean((Y.test - pred.classical)^2), 4) # 1.5705 (classical method)
round(mean((Y.test - pred.MM)^2), 4) # 0.8166 (MM method)
```

5 Data analysis

We consider the MaryRiverFlow dataset available in the **robflreg** package to present the superiority of the robust functional principal component regression models over the classical model when the dataset includes outliers. The MaryRiverFlow dataset consists of hourly river-flow measurements obtained from January 2009 to December 2014 (6 years in total) in the Mary River, Australia. River-flow time series varies throughout the years due to the variation of the seasons and the amount of rainfall received at particular catchments. This problem still needs to be solved in the hydrological domain to be addressed appropriately using theoretical models.

Our first aim with the MaryRiverFlow dataset is to assess how the previous river flow curve time series, $\mathcal{X}_i(s)$, affects the current day's maximum river flow, Y_i. Here, the observations of predictor are assumed to be functions of hours, i.e., there are 2188 functional observations $\mathcal{X}_i(t)(1 \leq t \leq 24, i = 1, \ldots, 2188)$ while the response is a scalar predictor. A graphical display of the response and predictor is presented in Figure 7. From this figure, both the scalar response and functional predictor include clear outliers, which motivates us to apply the robust functional principal component regression models to better model this dataset.

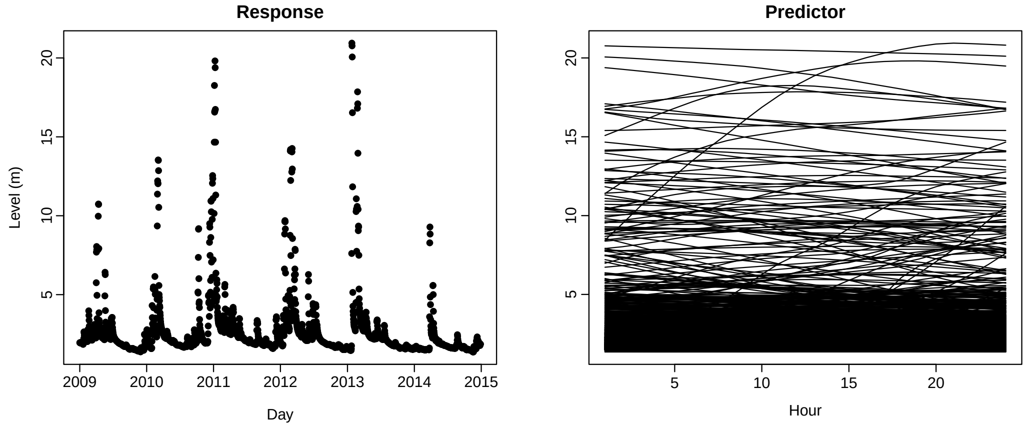


Figure 7: Plot of the daily maximum river flow measurement (left panel) and the functional time series plot (right panel) of the flow level in the Mary River.

Figure 7 can be produced by the following code:

```
library(robflreg)
library(fda.usc)
data("MaryRiverFlow")
X <- MaryRiverFlow[1:2188,]
Y <- apply(MaryRiverFlow[2:2189,], 1, max)
Day <- seq(as.Date("2009/01/01"), as.Date("2014/12/28"), by="days")
plot(Day, Y, type = "p", pch = 16, ylab = "Level (m)", main = "Response")
X <- fdata(X, argvals = 1:24)
plot(X, lty = 1, ylab = "", xlab = "Hour", col = "black",
     main = "Predictor", mgp = c(2, 0.5, 0))
```

We assume the SFLRM, $Y_i = \beta_0 + \int \mathcal{X}_i(s)\beta(s)ds$, where Y_i is the maximum river flow measurement for the current day and $\mathcal{X}_i(s)$ is the true river flow measurements recorded in the previous day. An expanding-window approach is considered to compare the predictive performance of the robust methods with the classical one. While doing so, The entire data are divided into two parts: a training sample containing the days from 01/01/2009 to 20/12/2014 and a test sample including days from 21/12/2014 to 30/12/2014. First, the models are constructed using the entire training data to forecast the maximum river flow measurement on 21/12/2014. Then, the maximum river flow measurement is forecasted by increasing the training samples by one day. This procedure is repeated until the training samples cover the entire dataset. The MSPE is computed for each method, and the computed mean MSPEs ($\times 10^{-3}$) along with standard errors ($\times 10^{-3}$ given in bracket) are 7.559 (5.674), 1.635 (1.303), 1.671 (1.376), 1.617 (1.274), and 1.617 (1.274) for the classical, LTS, MM, S, and tau-estimator based SFLRM, respectively. From the results, all the robust methods produce significantly smaller MSPE values than the classical method. These results can be obtained by the following code:

```
library(robflreg)
data("MaryRiverFlow")
MaryRiverFlow <- as.matrix(MaryRiverFlow)
X <- list(MaryRiverFlow[1:2188,])
Y <- apply(MaryRiverFlow[2:2189,], 1, max)
gp <- rep(list(1:24), 1)

MSPE <- matrix(, ncol = 5, nrow = 10)
colnames(MSPE) <- c("classical","LTS","MM","S","tau")
starting_value <- 2178

for(i in 1:10){

# Divide the data into training and test samples
    Y.train <- Y[1:starting_value]
    Y.test <- Y[(starting_value+1)]

    X.train <- list(X[[1]][1:starting_value,])
    X.test <- list(matrix(X[[1]][(starting_value+1),], nrow = 1))

    # Perform classical and robust functional principal component regression models
    model.classical <- rob.sf.reg(Y.train, X.train, emodel = "classical", gp = gp)
   model.LTS <- rob.sf.reg(Y.train, X.train, emodel = "robust", fmodel = "LTS", gp = gp)
    model.MM <- rob.sf.reg(Y.train, X.train, emodel = "robust", fmodel = "MM", gp = gp)
    model.S <- rob.sf.reg(Y.train, X.train, emodel = "robust", fmodel = "S", gp = gp)
   model.tau <- rob.sf.reg(Y.train, X.train, emodel = "robust", fmodel = "tau", gp = gp)

    # Predict the maximum river flow measurement of the current day
    pred.classical <- predict_sf_regression(object = model.classical, Xnew = X.test)
    pred.LST <- predict_sf_regression(object = model.LTS, Xnew = X.test)
    pred.MM <- predict_sf_regression(object = model.MM, Xnew = X.test)
    pred.S <- predict_sf_regression(object = model.tau, Xnew = X.test)
    pred.tau <- predict_sf_regression(object = model.tau, Xnew = X.test)

    # Record the MSPE values
    MSPE[i,1] <- (Y.test - pred.classical)^2
    MSPE[i,2] <- (Y.test - pred.LST)^2
    MSPE[i,3] <- (Y.test - pred.MM)^2
    MSPE[i,4] <- (Y.test - pred.S)^2
    MSPE[i,5] <- (Y.test - pred.tau)^2
    starting_value <- starting_value + 1
}

apply(MSPE, 2, mean); apply(MSPE, 2, sd)
```

Our second aim with the MaryRiverFlow dataset is to assess how the previous river flow curve time series, $\mathcal{X}_i(s)$ affects the current day's river flow curve time series, $\mathcal{Y}_i(t)$. In this case, the elements of both the response and predictor are assumed to be functions of hours. Here, we assume the FFLRM, $\mathcal{Y}_i(t) = \beta_0(t) + \int \mathcal{X}_i(s)\beta(s,t)dsdt$. The similar expanding-window approach used in the SFLRM example is considered, i.e., the entire dataset is divided into two parts: a training sample containing the days from 01/01/2009 to 20/12/2014 and a test sample including days from 21/12/2014 to 30/12/2014. The functional principal component regression models are constructed using the entire training data to forecast the river flow curve time series on 21/12/2014. Then, the river flow curve time series is forecasted by increasing the training samples by one day. This procedure is repeated until the training samples cover the entire dataset. The MSPE is computed for each method, and the computed mean MSPEs ($\times 10^{-3}$) along with standard errors ($\times 10^{-3}$ given in bracket) are 5.721 (3.884), 1.565 (0.965), 1.377 (0.847), 1.546 (0.949), 1.511 (0.920), and 1.377 (0.849) for the classical, MCD, MLTS, MM, S, and tau-estimator based FFLRMs, respectively. From the results, the robust methods produce improved MSPEs over the classical FFLRM, i.e., the robust method models the MaryRiverFlow data better than the classical functional principal component regression model. These results can be obtained by the following code:

```
library(robflreg)
data("MaryRiverFlow")
MaryRiverFlow <- as.matrix(MaryRiverFlow)

# The following function is used to obtain the functional response and predictor
var_fun = function(data,order){
  n = dim(data)[1]
  y = data[((order+1):n),]
  x=list()
  a=1
  b=order
  for(i in 1:order){
    x[[i]] = data[(a:(n-b)),]
    a = a+1
    b = b-1
  }
  return(list(x=x,y=y))
}

# Grid points for the functional response and predictor
gpY <- 1:24 # grid points of Y
gpX <- rep(list(1:24), 1) # grid points of Xs

MSPE <- matrix(, ncol = 6, nrow = 10)
colnames(MSPE) <- c("classical","MCD","MLTS","MM","S","tau")
starting_value <- 2179
# In two cases (when i = 3 and i = 6) the covariance is not decomposed.
# Thus, try() is used to ignore these two cases.
for(i in 1:10){
  try({
  data.i <- MaryRiverFlow[1:starting_value,]
  # Obtain the functional response and predictor
  XY = var_fun(data=data.i, order=1)
  X = XY$x
  Y = XY$y

  # Perform classical and robust functional principal component regression models
  model.classical <- rob.ff.reg(Y = Y, X = X, model = "full", emodel = "classical",
                          gpY = gpY, gpX = gpX)
  model.MCD <- rob.ff.reg(Y = Y, X = X, model = "full", emodel = "robust",
                          fmodel = "MCD", gpY = gpY, gpX = gpX)
  model.MLTS <- rob.ff.reg(Y = Y, X = X, model = "full", emodel = "robust",
                          fmodel = "MLTS", gpY = gpY, gpX = gpX)
  model.MM <- rob.ff.reg(Y = Y, X = X, model = "full", emodel = "robust",
                          fmodel = "MM", gpY = gpY, gpX = gpX)
  model.S <- rob.ff.reg(Y = Y, X = X, model = "full", emodel = "robust",
                          fmodel = "S", gpY = gpY, gpX = gpX)
  model.tau <- rob.ff.reg(Y = Y, X = X, model = "full", emodel = "robust",
                          fmodel = "tau", gpY = gpY, gpX = gpX)
  Xnew = list(t(as.matrix(Y[dim(Y)[1],])))

  # Predict the maximum river flow measurement of the current day
  predict.classical <- predict_ff_regression(object = model.classical, Xnew = Xnew)
  predict.MCD <- predict_ff_regression(object = model.MCD, Xnew = Xnew)
  predict.MLTS <- predict_ff_regression(object = model.MLTS, Xnew = Xnew)
  predict.MM <- predict_ff_regression(object = model.MM, Xnew = Xnew)
  predict.S <- predict_ff_regression(object = model.S, Xnew = Xnew)
  predict.tau <- predict_ff_regression(object = model.tau, Xnew = Xnew)

  # Record the MSPE values
  MSPE[i,1] <- mean((predict.classical - MaryRiverFlow[starting_value+1,])^2)
  MSPE[i,2] <- mean((predict.MCD - MaryRiverFlow[starting_value+1,])^2)
  MSPE[i,3] <- mean((predict.MLTS - MaryRiverFlow[starting_value+1,])^2)
  MSPE[i,4] <- mean((predict.MM - MaryRiverFlow[starting_value+1,])^2)
  MSPE[i,5] <- mean((predict.S - MaryRiverFlow[starting_value+1,])^2)
  MSPE[i,6] <- mean((predict.tau - MaryRiverFlow[starting_value+1,])^2)
  starting_value <- starting_value +1
  },silent=T)
}

apply(MSPE, 2, mean, na.rm=TRUE); apply(MSPE, 2, sd, na.rm=TRUE)
```

6 Conclusion

The R package **robflreg** provides an implementation of functional principal component regression model based on several robust procedures to fit and predict SFLRM and FFLRM. These methods are centered on the RFPCA of Bali et al. (2011), a popular robust dimension reduction technique in functional data, and several robust regression parameter estimators. In addition, the package **robflreg** allows us to fit and predict SFLRM and FFLRM via the classical FPCA and least-squares estimator. Several simulation examples and empirical data analysis show that the robust procedures provide better inference for functional linear regression models when outliers are present in the response and predictor variables. The **robflreg** package code can be found at: `https://github.com/cran/robflreg`.

The aspects that the current version of the R package **robflreg** that may be improved upon in the future are listed below.

1) In the current version, the scalar predictors are allowed in the SFLRM only. In the next versions of the package, it is planned to improve the functions `rob.ff.reg` and `predict_ff_regression` to include scalar predictors (whose effects are constant and/or vary along the continuum of the functional predictor).
2) The current package version does not allow for modeling the function-on-scalar regression model, where the response consists of random functions, and the predictors include scalar observations. The robust procedures used in the FFLRM are planned to extend the function-on-scalar regression model in the subsequent versions of the package.
3) In the current version, the observations of the functional variables are assumed to be densely observed. In the next versions of the package, the robust methods are planned to extend the models where the elements of functional variables can also be observed over irregular and curve-specific grids.

Acknowledgement

We thank three reviewers for their careful reading of our manuscript and valuable suggestions and comments, which have helped us produce an improved version of our manuscript and the R package. The first author was supported by The Scientific and Technological Research Council of Turkey (TUBITAK) (grant no: 120F270). The second author was partially supported by an Australian Research Council Discovery Project (grant no: DP230102250). This study is dedicated to the people who lost their lives in the earthquake that occurred in Turkey on February 6, 2023.

Bibliography

M. K. Ahn, J. D. Tucker, W. Wu, and A. Srivastava. Regression models using shapes of functions as predictors. *Computational Statistics and Data Analysis*, 151:107017, 2020. doi: https://doi.org/10.1016/j.csda.2020.107017. [p212]

A. Arribas-Gil and J. Romo. Shape outlier detection and visualization for functional data: the outliergram. *Biostatistics*, 15(4):603–619, 2014. doi: https://doi.org/10.1093/biostatistics/kxu006. [p223]

J. L. Bali, G. Boente, D. E. Tyler, and J.-L. Wang. The multivariate least-trimmed squares estimator. *Journal of Multivariate Analysis*, 99(3):311–338, 2008. doi: https://doi.org/10.1016/j.jmva.2006.06.005. [p212]

J. L. Bali, G. Boente, D. E. Tyler, and J.-L. Wang. Robust functional principal components: A projection-pursuit approach. *The Annals of Statistics*, 39(6):2852–2882, 2011. doi: https://doi.org/10.1214/11-AOS923. [p212, 217, 218, 219, 221, 230]

M. G. Ben, E. Martinez, and V. J. Yohai. Robust estimation for the multivariate linear model based on a τ scale. *Journal of Multivariate Analysis*, 97(7):1600–1622, 2006. doi: https://doi.org/10.1016/j.jmva.2005.08.007. [p212]

U. Beyaztas and H. L. Shang. On function-on-function regression: Partial least squares approach. *Environmental and Ecological Statistics*, 27(1):95–114, 2020. doi: https://doi.org/10.1007/s10651-019-00436-1. [p212]

U. Beyaztas and H. L. Shang. A partial least squares approach for function-on-function interaction regression. *Computational Statistics*, 36(2):911–939, 2021. doi: https://doi.org/10.1080/03610926.2022.2065018. [p221]

U. Beyaztas and H. L. Shang. A robust functional partial least squares for scalar-on-multiple-function regression. *Journal of Chemometrics*, 36(4):e3394, 2022. doi: https://doi.org/10.1002/cem.3394. [p212]

M. Bilodeau and P. Duchesne. Robust estimation of the sur model. *The Canadian Journal of Statistics*, 28 (2):277–288, 2000. doi: https://doi.org/10.2307/3315978. [p212]

G. Boente, M. Salibian-Barrera, and P. Vena. Robust estimation for semi-functional linear regression models. *Computational Statistics & Data Analysis*, 152:107041, 2020. doi: https://doi.org/10.1016/j.csda.2020.107041. [p212]

H. Cardot, F. Ferraty, and P. Sarda. Functional linear model. *Statistics and Probability Letters*, 45(1): 11–22, 1991. doi: https://doi.org/10.1016/S0167-7152(99)00036-X. [p212]

H. Cardot, F. Ferraty, and P. Sarda. Spline estimators for the functional linearmodel. *Statistica Sinica*, 13(3):571–591, 2003. [p212]

D. Chen, P. Hall, and H.-G. Müller. Single and multiple index functional regression models with nonparametric link. *The Annals of Statistics*, 39(3):1720–1747, 2011. doi: https://doi.org/10.1214/11-AOS882. [p212]

J. M. Chiou, Y. F. Yang, and Y. T. Chen. Multivariate functional linear regression and prediction. *Journal of Multivariate Analysis*, 146:301–312, 2016. doi: https://doi.org/10.1016/j.jmva.2015.10.003. [p212]

C. Croux and A. Ruiz-Gazen. High breakdown estimators for principal components: the projection-pursuit approach revisited. *Journal of Multivariate Analysis*, 95(1):206–226, 1996. doi: https://doi.org/10.1016/j.jmva.2004.08.002. [p212]

D. Şentürk and H.-G. Müller. Generalized varying coefficient models for longitudinal data. *Biometrika*, 95(3):653–666, 2008. doi: https://doi.org/10.1093/biomet/asn006. [p212]

A. Cuaves, M. Febrero, and R. Fraiman. Robust estimation and classification for functional data via projection-based depth notions. *Computational Statistics*, 22(3):481–496, 2007. doi: https://doi.org/10.1007/s00180-007-0053-0. [p223]

A. Cuevas. A partial overview of the theory of statistics with functional data. *Journal of Statistical Planning and Inference*, 147:1–23, 2014. doi: https://doi.org/10.1016/j.jspi.2013.04.002. [p212]

W. Dai and M. G. Genton. Multivariate functional data visualization and outlier detection. *Journal of Computational and Graphical Statistics*, 27(4):923–934, 2018. doi: https://doi.org/10.1080/10618600.2018.1473781. [p223]

W. W. Dou, D. Pollard, and H. H. Zhou. Estimation in functional regression for general exponential families. *The Annals of Statistics*, 40(5):2421–2451, 2012. doi: https://doi.org/10.1214/12-AOS1027. [p212]

I. L. Dryden and K. V. Mardia. *Statistical Shape Analysis, with Applications in R*. Wiley Series in Probability and Statistics, New York, 2016. [p212]

M. Febrero-Bande, P. Galeano, and W. Gonzalez-Mantelga. Outlier detection in functional data by depth measures, with application to identify abnormal NO$_x$ levels. *Environmetrics*, 19(4):331–345, 2008. doi: https://doi.org/10.1002/env.878. [p223]

F. Ferraty and P. Vieu. *Nonparametric Functional Data Analysis*. Springer, New York, 2006. [p212]

J. Goldsmith, J. Bobb, C. M. Crainiceanu, B. Caffo, and D. Reich. Penalized functional regression. *Journal of Computational and Graphical Statistics*, 20(4):830–851, 2011. doi: https://doi.org/10.1198/jcgs.2010.10007. [p212]

J. Goldsmith, F. Scheipl, L. Huang, J. Wrobel, C. Di, J. Gellar, J. Harezlak, M. W. McLean, B. Swihart, L. Xiao, C. Crainiceanu, and P. T. Reiss. *refund: Regression with Functional Data*, 2022. URL https://CRAN.R-project.org/package=refund. R package version 0.1-28. [p212]

J. Harezlak, B. A. Coull, N. M. Laird, S. R. Magari, and D. C. Christiani. Penalized solutions to functional regression problems. *Computational Statistics and Data Analysiss*, 51(10):4911–4925, 2007. doi: https://doi.org/10.1016/j.csda.2006.09.034. [p212]

L. Horváth and P. Kokoszka. *Inference for Functional Data with Applications*. Springer, New York, 2012. [p212]

T. Hsing and R. Eubank. *Theoretical Foundations of Functional Data Analysis, with an Introduction to Linear Operators*. John Wiley & Sons, Chennai, India, 2015. [p212]

H. Hullait, D. S. Leslie, N. G. Pavlidis, and S. King. Robust function-on-function regression. *Technometrics*, 63(3):396–409, 2021. doi: https://doi.org/10.1080/00401706.2020.1802350. [p212]

A. E. Ivanescu, A. M. Staicu, F. Scheipl, and S. Greven. Penalized function-on-function regression. *Computational Statistics*, 30(2):539–568, 2015. doi: https://doi.org/10.1007/s00180-014-0548-4. [p212]

G. M. James. Generalized linear models with functional predictors. *Journal of Royal Statistical Society, Series B*, 64(3):411–432, 2002. doi: https://doi.org/10.1111/1467-9868.00342. [p212]

C. Jiang and J. L. Wang. Functional single index models for longitudinal data. *The Annals of Statistics*, 39(1):362–388, 2011. doi: https://doi.org/10.1214/10-AOS845. [p212]

I. Kalogridis and S. V. Aelst. Robust functional regression based on principal components. *Journal of Multivariate Analysis*, 173:393–415, 2019. doi: https://doi.org/10.1016/j.jmva.2019.04.003. [p212]

P. Kokoszka and M. Reimherr. *Introduction to Functional Data Analysis*. CRC Press, Boca Raton, 2017. [p212]

M. Koller and W. A. Stahel. Sharpening wald-type inference in robust regression for small samples. *Computational Statistics & Data Analysis*, 55(8):2504–2515, 2011. doi: https://doi.org/10.1016/j.csda.2011.02.014. [p212]

N. L. Kudraszow and R. A. Moronna. Estimates of mm type for the multivariate linear model. *Journal of Multivariate Analysis*, 102(9):1280–1292, 2011. doi: https://doi.org/10.1016/j.jmva.2011.04.011. [p212]

R. A. Maronna and V. J. Yohai. Robust functional linear regression based on splines. *Computational Statistics and Data Analysis*, 65:46–55, 2013. doi: https://doi.org/10.1016/j.csda.2011.11.014. [p212]

J. S. Marron, J. O. Ramsay, L. M. Sangalli, and A. Srivastava. Functional data analysis of amplitude and phase variation. *Statistical Science*, 30(4):468–484, 2015. doi: https://doi.org/10.1214/15-STS524. [p212]

H. Matsui, S. Kawano, and S. Konishi. Regularized functional regression modeling for functional response and predictors. *Journal of Math-for-Industry*, 1(A3):17–25, 2009. [p212]

F. A. Ocana, A. M. Aguilera, and M. Escabias. Computational considerations in functional principal component analysis. *Computational Statistics*, 22(3):449–465, 2007. doi: https://doi.org/10.1007/s00180-007-0051-2. [p218]

P. Raña, G. Aneiros, and J. M. Vilar. Detection of outliers in functional time series. *Environmetrics*, 26(3):178–191, 2015. doi: https://doi.org/10.1002/env.2327. [p212]

J. O. Ramsay and C. J. Dalzell. Some tools for functional data analysis. *Journal of the Royal Statistical Society, Series B*, 53(3):539–572, 1991. doi: https://doi.org/10.1111/j.2517-6161.1991.tb01844.x. [p212]

J. O. Ramsay and B. W. Silverman. *Applied Functional Data Analysis*. Springer, New York, 2002. [p212]

J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer, New York, 2006. [p212, 213]

J. O. Ramsay, S. Graves, and G. Hooker. *fda: Functional Data Analysis*, 2022. URL https://CRAN.R-project.org/package=fda. R package version 6.0.5. [p212]

P. T. Reiss and T. R. Ogden. Functional principal component regression and functional partial least squares. *Journal of the American Statistical Association: Theory and Methods*, 102(479):984–996, 2007. doi: https://doi.org/10.1198/016214507000000527. [p212]

P. J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association: Theory and Methods*, 79(388):871–881, 1984. doi: https://doi.org/10.1080/01621459.1984.10477105. [p212]

P. J. Rousseeuw, K. V. Driessen, S. V. Aelst, and J. Agullo. Robust multivariate regression. *Technometrics*, 46(3):293–305, 2004. doi: https://doi.org/10.1198/004017004000000329. [p212]

M. Salibian-Barrera, G. Willems, and R. Zamar. The fast-tau estimator for regression. *Journal of Computational and Graphical Statistics*, 17(3):659–682, 2008. doi: https://doi.org/10.1198/106186008X343785. [p212]

F. Scheipl, A.-M. Staicu, and S. Greven. Functional additive mixed models. *Journal of Computational and Graphical Statistics*, 24(2):477–501, 2015. doi: https://doi.org/10.1080/10618600.2014.901914. [p212]

H. Shin and S. Lee. An RKHS approach to robust functional linear regression. *Statistica Sinica*, 26: 255–272, 2016. [p212]

A. Srivastava and E. P. Klassen. *Functional and Shape Data Analysis*. Springer, New York, 2016. [p212]

Y. Sun and M. G. Genton. Functional boxplots. *Journal of Computational and Graphical Statistics*, 20(2): 316–334, 2011. doi: https://doi.org/10.1198/jcgs.2011.09224. [p223]

J. D. Tucker, J. R. Lewis, and A. Srivastava. Elastic functional principal component regression. *Statistical Analysis and Data Mining*, 12(2):101–115, 2019. doi: https://doi.org/10.1002/sam.11399. [p212]

F. Yao, H.-G. Müller, and J.-L. Wang. Functional linear regression analysis for longitudinal data. *The Annals of Statistics*, 33(6):2873–2903, 2005. doi: https://doi.org/10.1214/009053605000000660. [p212]

V. J. Yohai. High breakdown-point and high efficiency estimates for regression. *The Annals of Statistics*, 15(2):642–665, 1987. doi: https://doi.org/10.1214/aos/1176350366. [p212]

H. Zhu, P. J. Brown, and J. S. Morris. Robust, adaptive functional regression in functional mixed model framework. *Journal of the American Statistical Association*, 106(1):1167–1179, 2011. doi: https://doi.org/10.1198/jasa.2011.tm10370. [p212]

Ufuk Beyaztas
Marmara University
Department of Statistics, Goztepe Campus, 34722, Kadikoy, Istanbul
Turkey
(ORCID 0000-0002-5208-4950)
ufuk.beyaztas@marmara.edu.tr

Han Lin Shang
Macquarie University
Department of Actuarial Studies and Business Analytics, Level 7, 4 Eastern Road, Sydney, NSW 2109
Australia
(ORCID 0000-0003-1769-6430)
hanlin.shang@mq.edu.au

Fairness Audits and Debiasing Using mlr3fairness

by Florian Pfisterer, Siyi Wei, Sebastian Vollmer, Michel Lang, and Bernd Bischl

Abstract Given an increase in data-driven automated decision-making based on machine learning (ML) models, it is imperative that, along with tools to develop and improve such models, there are sufficient capabilities to analyze and assess models with respect to potential biases. We present the package **mlr3fairness**, a collection of metrics and methods that allow for the assessment of bias in machine learning models. Our package implements a variety of widely used fairness metrics that can be used to audit models for potential biases, along with a set of visualizations that can help to provide additional insights into such biases. **mlr3fairness** furthermore integrates bias mitigation methods for machine learning models through data pre-processing or post-processing of predictions. These allow practitioners to trade off performance and fairness metrics that are appropriate for their use case.

1 Introduction

Humans are increasingly subject to data-driven automated decision-making. Those automated procedures, such as credit risk assessments, are often applied using predictive models (Kozodoi, Jacob, and Lessmann 2022; Galindo and Tamayo 2000), often profoundly affecting individual's lives. It is therefore important that, along with tools to develop and improve such models, we also develop sufficient capabilities to analyze and assess models not only with respect to their robustness and predictive performance but also with respect to potential biases. This is highlighted by the European General Data Protection Regulation (GDPR) which requires data to be processed fairly. Popular modelling frameworks for the R language (R Core Team 2021) such as **caret** (Kuhn 2021), **tidymodels** (Kuhn and Wickham 2020), **SuperLearner** (Polley et al. 2021), or **mlr** (Bischl et al. 2016) implement a plethora of metrics to measure performance, but fairness metrics are widely missing. This lack of availability can be detrimental to obtaining fair and unbiased models if the result is to forgo bias audits due to the considerable complexity of implementing such metrics. Consequently, there exists a considerable necessity for R packages to (a) implement such metrics, and (b) to connect these metrics to existing ML frameworks. If biases are detected and need to be mitigated, we might furthermore want to employ bias mitigation techniques that tightly integrate with the fitting and evaluation of the resulting models in order to obtain trade-offs between a model's fairness and utility (e.g., predictive accuracy).

In this article, we present the **mlr3fairness** package which builds upon the ML framework **mlr3** (Lang et al. 2019). Our extension contains fairness metrics, fairness visualizations, and model-agnostic pre- and post-processing operators that aim to reduce biases in ML models. Additionally, **mlr3fairness** comes with reporting functionality that assists the user in documenting data and ML models, as well as in performing fairness audits.

In the remainder of the article, we first provide an introduction to fairness in ML to raise awareness of biases that can arise due to the use of ML models. Next, we introduce the **mlr3fairness** package, followed by an extensive case study, showcasing the capabilities of **mlr3fairness**. We conclude with a summary.

2 Fairness in Machine Learning

Studies have found that data-driven automated decision-making systems often improve over human expertise (Dawes, Faust, and Meehl (1989)) and high-stakes decisions can therefore be enhanced using data-driven systems. This often does not only improve predictions, but can also make decisions more efficient through automation. Such systems, often without human oversight, are now ubiquitous in everyday life (O'neil 2016; Eubanks 2018; Noble 2018). To provide further examples, ML-driven systems are used for highly influential decisions such as loan accommodations (Chen 2018; Turner and McBurnett 2019), job applications (Schumann et al. 2020), healthcare (Topol 2019), and criminal sentencing (Angwin et al. 2016; Corbett-Davies et al. 2017; Berk et al. 2018). With this proliferation, such decisions have become subject to scrutiny as a result of prominent inadequacies or failures, for example in the case of the COMPAS recidivism prediction system (Angwin et al. 2016).

Without proper auditing, those models can unintentionally result in negative consequences for individuals, often from underprivileged groups (Barocas, Hardt, and Narayanan 2019). Several sources of such biases are worth mentioning in this context: Data often contains **historical biases** such as gender or racial stereotypes, that – if picked up by the model – will be replicated into the future. Similarly, unprivileged populations are often not represented in data due to **sampling biases** leading to models that perform well in groups sufficiently represented in the data but worse on others (Buolamwini and Gebru 2018) – this includes a higher rate of missing data. Other biases include biases in how *labels* and *data* are measured (Bao et al. 2021) as well as **feedback** loops where repeated decisions affect the population subject to such decisions. For an in-depth discussion and further sources of biases, the interested reader is referred to available surveys of the field (Barocas, Hardt, and Narayanan 2019; Mehrabi et al. 2021; S. Mitchell et al. 2021).

Quantifying fairness

We now turn to the question of how we can detect whether disparities exist in a model and if so, how they can be quantified. What constitutes a fair model depends on a society's ethical values and which normative position we take, resulting in different metrics that are applied to a problem at hand. In this article, we focus on a subgroup of these, so-called *statistical group fairness* metrics. First, the observations are grouped by a sensitive attribute A ($A = 0$ vs. $A = 1$), which, e.g., is an identifier for a person's race or a person's gender. For the sake of simplicity, we consider a *binary classification* scenario and a *binary sensitive attribute*. Each observation has an associated label $Y, Y \in \{0,1\}$, and we aim to predict, e.g., whether a defendant was caught re-offending. A system then makes a prediction $\hat{Y}, \hat{Y} \in \{0,1\}$, with the goal to predict whether an individual might re-offend. We assume that $Y = 1$ is the favored outcome in the following exposition. While we do not describe them in detail, the concepts discussed in the following often extend naturally to more complex scenarios including multi-class classification, regression or survival analysis. Similarly, metrics can be extended to settings that require consideration of multiple possibly intersecting sensitive attributes. We now provide and discuss groups of metrics that require either *Separation* or *Independence* (Barocas, Hardt, and Narayanan 2019) to provide further intuition regarding core concepts and possible applications.

Separation

One group of widely used fairness notions requires **Separation**: $\hat{Y} \perp A|Y$. In order for separation to hold, the prediction $\hat{Y}$ has to be independent of A given the true label Y. This essentially requires that some notion of model error, e.g., accuracy or false positive rate, is equal across groups A. From this notion, we can derive several metrics that come with different implications. It is important to note that those metrics can only meaningfully identify biases under the assumption that no disparities exist in the data or that they are legally justified. For example, if societal biases lead to disparate measurements of an observed quantity (e.g. SAT scores) for individuals with the same underlying ability, *separation* based metrics might not identify existing biases. For this reason, Wachter, Mittelstadt, and Russell (2020) refer to those metrics as *bias-preserving* metrics since underlying disparities are not addressed.

Equalized Odds A predictor $\hat{Y}$ satisfies *equalized odds* with respect to a sensitive attribute A and observed outcome Y, if $\hat{Y}$ and A are conditionally independent given Y:

$$\mathbb{P}\left(\hat{Y}=1 \mid A=0, Y=y\right)=\mathbb{P}\left(\hat{Y}=1 \mid A=1, Y=y\right), \quad y \in\{0,1\}. \tag{1}$$

In short, we require that the true positive rates (TPR) and false positive rates (FPR) across both groups $A = 0$ and $A = 1$ are equal. This intuitively requires, e.g., in the case of university admission, independent of the sensitive attribute, equal chances for qualified individuals to be accepted and unqualified individuals to be rejected. Similar measures have been proposed based on equalized false positive rates (Chouldechova 2017) and false omission rates (Berk et al. 2018), depending on the scenario and societal context.

Equality of Opportunity A predictor $\hat{Y}$ satisfies *equality of opportunity* with respect to a sensitive attribute A and observed outcome Y, if $\hat{Y}$ and A are conditionally independent for $Y = 1$. This is a relaxation of the aforementioned *equalized odds* essentially only requiring equal TPRs:

$$\mathbb{P}\left(\hat{Y}=1 \mid A=0, Y=1\right)=\mathbb{P}\left(\hat{Y}=1 \mid A=1, Y=1\right). \tag{2}$$

Intuitively, this only requires that, independent of the sensitive attribute, qualified individuals have the same chance of being accepted.

Performance Parity A more general formulation can be applied when we require parity of some performance metric across groups. To provide an example, Buolamwini and Gebru (2018) compare accuracy across intersectional subgroups, essentially arguing that model performance should be equal across groups:

$$\mathbb{P}\left(\hat{Y} = Y \mid A = 0\right) = \mathbb{P}\left(\hat{Y} = Y \mid A = 1\right). \tag{3}$$

This intuitively requires that the model should work equally well for all groups, i.e., individuals are correctly accepted or denied at the same rate, independent of the predicted attribute. This notion can be extended across supervised learning settings and performance metrics, leading to considerations of equal mean squared error, e.g., in a regression setting.

Independence

The second group of fairness metrics is given by so-called *bias-transforming* metrics (Wachter, Mittelstadt, and Russell 2020). They require that decision rates, such as the positive rate, are equal across groups. This notion can identify biases, e.g., those which arise from societal biases, that manifest in different base rates across groups. At the same time, employing such notions poses a considerable risk, as blindly optimizing for demographic parity might result in predictors that, for example jail innocent people from an advantaged group in order to achieve parity across both groups (Dwork et al. 2012; Berk et al. 2018). A predictor $\hat{Y}$ satisfies *demographic parity* (Calders and Verwer 2010) with respect to a sensitive attribute A and observed outcome Y, if $\hat{Y}$ and A are conditionally independent:

$$\mathbb{P}\left(\hat{Y} = 1 \mid A = 0\right) = \mathbb{P}\left(\hat{Y} = 1 \mid A = 1\right). \tag{4}$$

In contrast to the previous definitions, this requires that the chance of being accepted is equal across groups.

Fairness metrics

In order to encode the requirements in equations (1) - (4) into a fairness metric, we encode differences between measured quantities in two groups. For a performance metric M, e.g., the true positive rate (TPR), we calculate the difference in the metric across the two groups:

$$\Delta_{\mathrm{M}} = \mathrm{M}_{A=0} - \mathrm{M}_{A=1}.$$

When Δ_{M} significantly deviates from 0, this indicates a fairness violation with respect to the fairness notion described in M. To provide an example, with $\mathbb{P}\left(\hat{Y} = 1 \mid A = \star, Y = 1\right)$ denoted with $\mathrm{TPR}_{A=\star}$, we calculate the difference in TPR between the two groups:

$$\Delta_{\mathrm{TPR}} = \mathrm{TPR}_{A=0} - \mathrm{TPR}_{A=1}.$$

When Δ_{TPR} now significantly deviates from 0, the prediction $\hat{Y}$ violates the requirement for *equality of opportunity* formulated above.

It is important to note that in practice, we might not be able to perfectly satisfy a given metric, e.g., due to stochasticity in data and labels. Instead, to provide a binary conclusion regarding fairness, a model could be considered fair if $|\Delta_{\mathrm{TPR}}| < \epsilon$ for a given threshold $\epsilon > 0$, e.g., $\epsilon = 0.05$. This allows for small deviations from perfect fairness due to variance in the estimation of $\mathrm{TPR}_{A=\star}$ or additional sources of bias. However, choosing appropriate thresholds is difficult, and widely used values for ϵ such as 0.05 are arbitrary and do not translate to legal doctrines, e.g., disparate impact (Watkins, McKenna, and Chen 2022). A more in-depth treatment of metrics is given by (Barocas, Hardt, and Narayanan 2019; Saleiro et al. 2018; Kim, Chen, and Talwalkar 2020; Mehrabi et al. 2021; Wachter, Mittelstadt, and Russell 2020).

Selecting fairness metrics While the aforementioned metrics are conceptually similar, they encode different beliefs of what constitutes *fair* in a given scenario. Wachter, Mittelstadt, and Russell (2020) differentiate between *bias-preserving* and *bias-transforming* metrics: Bias-preserving metrics such as equalized odds and equality of opportunity require that errors made by a model are equal across groups. This can help to detect biases stemming, from imbalances in the sampling or under- and overfitting in ML models, but might be problematic in cases where labels are biased. To provide an example, police enforcement and subsequent arrests of violent re-offenders might be different across ZIP code areas, a proxy for race. This might lead to situations where observed labels Y suffer from differential measurement bias strongly correlated with race (Bao et al. 2021). Bias-preserving metrics do not take such disparities into account and might, therefore (wrongly) lead to the conclusion that a given model is fair.

Bias-transforming methods, in contrast, do not depend on labels and might therefore not suffer from this problem. They can help detect biases arising from different base rates across populations, arising, e.g., from aforementioned biases in the labelling or as a consequence of structural discrimination. Deciding which metrics to use constitutes a value judgement and requires careful assessment of the societal context a decision-making system is deployed in. A discussion of different metrics and their applicability can be found in the Aequitas Fairness Toolkit (Saleiro et al. 2018) which also provides guidance towards selecting a metric via the Aequitas Fairness Tree. Wachter, Mittelstadt, and Russell (2020) recommend using *bias-transforming* metrics and provide a checklist that can guide the choice of fairness metric. Corbett-Davies and Goel (2018), on the other hand, point out several limitations of available metrics and argue for grounding decisions in real-world quantities in addition to abstract fairness metrics. Similarly, Friedler, Scheidegger, and Venkatasubramanian (2016) emphasize the need to differentiate between constructs we aim to measure (e.g., job-related knowledge) and the observed quantity that can be measured in practice (e.g., years in a job) when trying to automate decisions, since disparities in how constructs translate to observed quantities might suffer from bias. To provide an example, individuals with similar abilities might exhibit different measured quantities (grades) due to structural bias, e.g., differential access to after-school tutoring programs.

The dangers of fairness metrics We want to stress that overly trusting in metrics can be dangerous and that fairness metrics cannot and should not be used to *prove* or *guarantee* fairness. Whether a selected fairness notion (and a corresponding numerical value) is actually fair depends on the societal context in which a decision is made and which action should be derived from a given prediction. Therefore, selecting the correct fairness metric requires a thorough understanding of the societal context of a decision, as well as the possible implications of such decisions. To provide an example, in some cases discrepancies in positive predictions might be justified or even desired, as they, for example, allow for a more nuanced, gender-specific diagnosis (Cirillo et al. 2020). Furthermore, fairness metrics might not detect biases in more fine-grained subgroups, e.g., at the intersection of multiple sensitive attributes. It is also important to note that fairness metrics merely provide a reduction of the aforementioned fairness notions into mathematical objectives. As such, they require a variety of abstraction steps that might invalidate the metric (Watkins, McKenna, and Chen 2022), as they, for example, require that the data is a large enough and representative sample of exactly the population that we aim to investigate. Furthermore, practitioners need to look beyond the model, and also at the data used for training and the process of data and label acquisition. If the data for example exhibit disparate measurement errors in the features or labels, valid fairness assessments can become impossible. Similarly, feedback loops might arise from a prediction leading to changes in the data collected in the future. Even an *initially fair* model might then lead to adverse effects in the long term (Schwöbel and Remmers 2022).

Note that the fairness definitions presented above serve a dual purpose (Wachter, Mittelstadt, and Russell 2020): First, as a *diagnostic tool* to detect disparities. This allows for assessing whether a model has inherited biases, e.g., from historical disparities reflected in the data. The second purpose is as a basis for *model selection* and making fair decisions in practice. In this setting, fairness notions are employed to audit ML models or to select which model should be used in practice. In this setting, it is important to note that fairness metrics should not be used as the sole basis for making decisions about whether to employ a given ML model or to assess whether a given system is fair. We therefore explicitly encourage using the presented metrics for exploratory purposes.

Other notions of fairness In addition to *statistical group fairness notions* introduced above, several additional fairness notions exist. The notion of *individual fairness* was proposed by Dwork et al. (2012). Its core idea comes from the principle of *treating similar cases similarly and different cases differently*. In contrast to statistical group fairness notions, this notion allows assessing *fairness* at an individual level and would therefore allow determining whether an individual is treated fairly. A more in-depth treatment of individual fairness notions is, given in Binns (2020). Similarly, a variety of *causal* fairness notions exist (c.f. Kilbertus et al. (2017)). They argue that assessing fairness requires incorporating causal relationships in the data and propose a variety of causal fairness metrics based on a *directed acyclic graph* describing relationships in the data.

Fairness constraints Statistical group fairness notions suffer from two further problems in practice: First, it might be hard to exactly satisfy the required fairness notions, e.g., due to a limited amount of data available for evaluation. Secondly, only requiring fairness might lead to degenerate solutions (Corbett-Davies and Goel 2018) or models that have low utility, e.g., in separating *good* and *bad* credit risk. One approach to take this into account is to employ models which maximize utility but satisfy some maximum constraint on potential unfairness. This can be achieved via constraints on the employed fairness measure, e.g. $|\Delta_M| \leq \epsilon$ requiring that the absolute difference in a metric M

between groups is smaller than a chosen value ϵ. In the following, we denote the fairness metric we want to minimize with Δ_M and a performance metric with ρ.

$$\rho_{|\Delta_M| \leq \epsilon} = \begin{cases} \rho & |\Delta_M| \leq \epsilon \\ -|\Delta_M| & \text{else.} \end{cases}$$

Note that this assumes that the fairness metric ρ is strictly positive and should be maximized. This approach is similar in spirit to the approach of Perrone et al., (2021) who optimize the constrained expected improvement $cEI = \mathbb{P}(|\Delta_M| \leq \epsilon) \cdot \rho$.

However, it is not immediately clear how the constraint ϵ should be chosen. An alternative, therefore, is to employ *multi-objective optimization* to investigate available trade-offs between performance and accuracy metrics. This can be done via **mlr3tuning** (Becker, Lang, et al. 2023) which contains functionality to tune models for multiple metrics, described in more detail in the mlr3book (Bernd Bischl 2024). The result of multi-objective optimization then is the *Pareto-set*: A list of models which optimally trade off the specified objectives.

Bias mitigation

If biases are detected in a model, we might now be interested in improving models in order to potentially mitigate such biases. Bias in models might arise from a variety of sources, so a careful understanding of the data, data quality and distribution might lead to approaches that can help in decreasing biases, e.g. through the collection of better or additional data or a better balancing of sensitive groups. Similarly, biases might arise from the model, through under- or overfitting and more careful tuning of model hyperparameters might help with improving fairness. Especially if the goal is to satisfy *bias-transforming* metrics, a better solution might often be to address fairness problems in the real world instead of relying on algorithmic interventions to solve fairness. This might lead to more robust, long-term solutions instead of temporarily addressing issues via algorithmic interventions. In addition, a variety of algorithmic bias mitigation techniques, that might help with obtaining fairer models have been proposed. Their goal is to reduce measured gaps in fairness, either via data pre-processing, employing models that incorporate fairness, or by applying post-processing techniques to a model's predictions. Popular examples of such techniques include computing instance weights before training (Kamiran and Calders 2012), where each observation is weighted proportional to the inverse frequency of its label and sensitive attribute. Other methods work by directly learning fair models that incorporate fairness constraints into the fitting procedure (Zafar et al. 2017) or by adapting model predictions, e.g., Hardt, Price, and Srebro (2016) propose to randomly flip a small fraction of predictions in each group given by $\hat{Y}$ and A, such that fairness metrics are satisfied in expectation. Since bias mitigation techniques are often tailored towards a particular fairness metric, the optimal choice is often not trivial and a combination of algorithms and bias mitigation techniques determined via tuning might result in an optimal model.

Bias-mitigation techniques, as proposed above, have the goal of mitigating fairness issues, as measured by fairness metrics. In practice, this usually comes with several drawbacks: First, bias-mitigation strategies often lead to a decrease in a classifier's predictive performance (Corbett-Davies and Goel 2018). In addition, processing schemes can worsen interpretability or introduce stochasticity during prediction (see, e.g., Hardt, Price, and Srebro (2016)). Furthermore, we want to caution against favouring bias-mitigation techniques over policy interventions that tackle biases at their root cause. A different set of risks is posed by *fairwashing* (Aivodji et al. 2019), i.e., finding fair explanations or satisfying fairness metrics for otherwise unfair models. If biases are only addressed at a given moment and without regard for downstream effects, they might simultaneously lead to a decrease in predictive performance in the near term and to negative consequences for the sensitive group in the long term (Schwöbel and Remmers 2022).

3 The mlr3fairness package

In this section, we first give an overview of related software. Next, we give a very brief introduction to the **mlr3** ecosystem of packages. Finally, the implemented extensions for fairness are presented.

Related software

Several R packages provide similar capabilities to our software, but mostly focus on fairness metrics and visualization. The **fairness** package (Kozodoi and V. Varga 2021) allows for the calculation of a variety of fairness metrics, while **aif360** (Bellamy et al. 2019) wraps the Python **aif360** module allowing for the computation of fairness metrics and several bias mitigation techniques, but has only limited interoperability with R objects such as `data.frames`. The **fairmodels** (Wiśniewski and Biecek 2022) package again allows for the computation of fairness metrics for classification and regression settings as well as several bias mitigation techniques. It tightly integrates with **DALEX** (Biecek 2018) to gain further insight using interpretability techniques.

Outside R, in Python, the **fairlearn** module (Bird et al. 2020) provides ample functionality to study a wide variety of metrics, bias mitigation with respect to a variety of pre-, in- and post-processing methods as well as to visualize differences. It furthermore provides a *fairlearn dashboard* providing a comprehensive fairness report. The **aif360** (Bellamy et al. 2019) module similarly provides metrics as well as bias mitigation techniques while the **aequitas** fairness toolkit (Saleiro et al. 2018) provides similar capabilities. Interoperability with the **scikit-learn** (Pedregosa et al. 2011) ML framework allows for bias mitigation for a wide variety of ML models in all aforementioned systems. Similar capabilities are also available in Julia's **Fairness.jl** (Agrawal, Chen, et al. 2020) library.

The mlr3 ecosystem

mlr3fairness is tightly integrated into the ecosystem of packages around the ML framework **mlr3** (Lang et al. 2019). **mlr3** provides the infrastructure to fit, resample, and evaluate over 100 ML algorithms using a unified API. Packages from the ecosystem can be installed and updated via the **mlr3verse** (Lang and Schratz 2023) package. Multiple extension packages bring numerous additional advantages and extra functionality. In the context of fairness, the following extension packages deserve special mention:

- **mlr3pipelines** (Binder et al. 2021) for pre- and postprocessing via pipelining. This allows composing bias mitigation techniques with arbitrary ML algorithms shipped with **mlr3** as well as fusing ML algorithms with pre-processing steps such as imputation or class balancing. It furthermore integrates with **mcboost** (Pfisterer et al. 2021), which implements additional bias mitigation methods. We present an example in the supplementary material.
- **mlr3tuning** and **bbotk** (Becker, Richter, et al. 2023) for its extensive tuning capabilities.
- **mlr3proba** (Sonabend et al. 2021) for survival analysis.
- **mlr3benchmark** for post-hoc analysis of benchmarked approaches.
- **mlr3oml** as a connector to OpenML (Vanschoren et al. 2014), an online scientific platform for collaborative ML.

In order to provide the required understanding for **mlr3**, we briefly introduce some terminology and syntax. A full introduction can be found in the mlr3 book (Bernd Bischl 2024).

A `Task` in **mlr3** is a basic building block holding the data, storing covariates and the target variable along with some meta-information. The shorthand constructor function `tsk()` can be used to quickly access example tasks shipped with **mlr3** or **mlr3fairness**. In the following chunk, we retrieve the binary classification task with id `"adult_train"` from the package. It contains a part of the `Adult` data set (Dua and Graff 2017). The task is to predict whether an individual earns more than $50.000 per year. The column `"sex"` is set as a binary sensitive attribute with levels `"Female"` and `"Male"`.

```
library("mlr3verse")
library("mlr3fairness")

task = tsk("adult_train")
print(task)
```

```
#> <TaskClassif:adult_train> (30718 x 13)
#> * Target: target
#> * Properties: twoclass
#> * Features (12):
#>   - fct (7): education, marital_status, occupation, race, relationship,
#>     sex, workclass
#>   - int (5): age, capital_gain, capital_loss, education_num,
#>     hours_per_week
#> * Protected attribute: sex
```

The second building block is the `Learner`. It is a wrapper around an ML algorithm, e.g., an implementation of logistic regression or a decision tree. It can be trained on a `Task` and used for obtaining a `Prediction` on an independent test set which can subsequently be scored using a `Measure` to get an estimate for the predictive performance on new data. The shorthand constructors `lrn()` and msr() allow for the instantiation of implemented Learners and Measures, respectively. In the following example, we will first instantiate a learner, then split our data into a train and test set, afterwards train it on the train set of the dataset and finally evaluate predictions on held-out test data. The train-test split in this case is given by row indices, here stored in the idx variable.

```
learner = lrn("classif.rpart", predict_type = "prob")
idx = partition(task)
learner$train(task, idx$train)
prediction = learner$predict(task, idx$test)
```

We then employ the classif.acc measure which measures the accuracy of a prediction compared to the true label:

```
measure = msr("classif.acc")
prediction$score(measure)

#> classif.acc
#>      0.8382
```

In the example above, we obtain an accuracy score of 0.8382, meaning our ML model correctly classifies roughly 84 % of the samples in the test data. As the split into training set and test set is stochastic, the procedure should be repeated multiple times for smaller datasets (Bischl et al. 2012) and the resulting performance values should be aggregated. This process is called resampling, and can easily be performed with the resample() function, yielding a ResampleResult object. In the following, we employ 10-fold cross-validation as a resampling strategy:

```
resampling = rsmp("cv", folds = 10)
rr = resample(task, learner, resampling)
```

We can call the aggregate method on the ResampleResult to obtain the accuracy aggregated across all 10 replications.

```
rr$aggregate(measure)

#> classif.acc
#>      0.8408
```

Here, we obtain an accuracy of 0.8408, so slightly higher than previous scores, due to using a larger fraction of the data. Furthermore, this estimate has a lower variance (as it is an aggregate) at the cost of additional computation time. To properly compare competing modelling approaches, candidates can be benchmarked against each other using the benchmark() function (yielding a BenchmarkResult). In the following, we compare the decision tree from above to a logistic regression model. To do this, we use the benchmark_grid function to compare the two Learners across the same Task and resampling procedure. Finally, we aggregate the measured scores each learner obtains on each cross-validation split using the $aggregate() function.

```
learner2 = lrn("classif.log_reg", predict_type = "prob")

grid = benchmark_grid(task, list(learner, learner2), resampling)
bmr = benchmark(grid)

bmr$aggregate(measure)[, .(learner_id, classif.acc)]

#>         learner_id classif.acc
#> 1:   classif.rpart      0.8408
#> 2: classif.log_reg      0.8467
```

After running the benchmark, we can again call .$aggregate to obtain aggregated scores. The mlr3viz package comes with several ready-made visualizations for objects from mlr3 via ggplot2's (Wickham 2016) autoplot function. For a BenchmarkResult, the autoplot function provides a Box-plot comparison of performances across the cross-validation folds for each Learner. Figure 1 contains the box-plot comparison. We can see that log_reg has higher accuracy and lower interquartile range across the 10 folds, and we might therefore want to prefer the log_reg model.

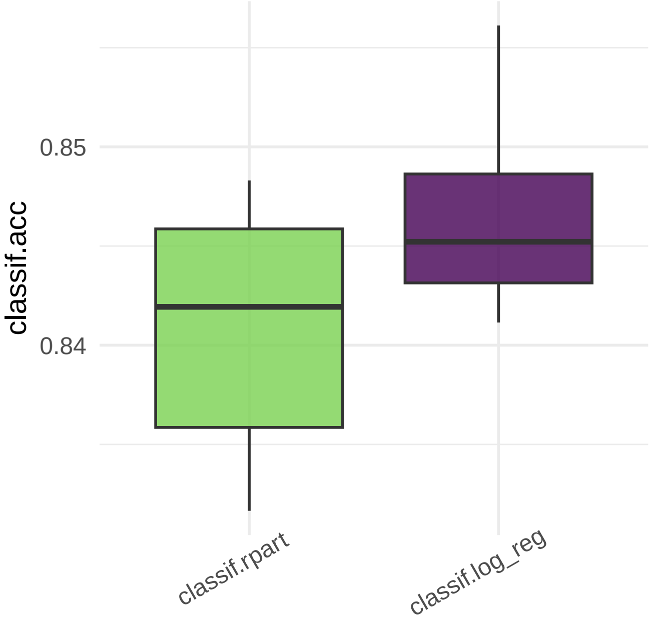


Figure 1: Model comparison based on accuracy for decision trees (rpart) and logistic regression (log_reg) across resampling splits.

Selecting the sensitive attribute

For a given task, we can select one or multiple sensitive attributes. In **mlr3**, the sensitive attribute is identified by the column role pta and can be set as follows:

```
task$set_col_roles("marital_status", add_to = "pta")
```

In the example above, we add the "marital_status" as an additional sensitive attribute. This information is then automatically passed on when the task is used, e.g., when computing fairness metrics. If more than one sensitive attribute is specified, metrics will be computed based on intersecting groups formed by the columns.

Quantifying fairness

With the **mlr3fairness** package loaded, fairness measures can be constructed via msr() like any other measure in **mlr3**. They are listed with prefix *fairness*, and simply calling msr() without any arguments will return a list of all available measures. Table 1 provides an overview of some popular fairness measures which are readily available.

Table 1: Overview of fairness metrics available with mlr3fairness.

key	description
fairness.acc	Accuracy equality (Buolamwini and Gebru, 2018)
fairness.mse	Mean squared error equality (Regression)
fairness.eod	Equalized odds (Hardt et al., 2016)
fairness.tpr	True positive rate equality / Equality of opportunity (Hardt et al., 2016)
fairness.fpr	False positive rate equality / Predictive equality (Chouldechova, 2017)
fairness.tnr	True negative rate equality
fairness.fnr	False negative rate equality (Berk et al., 2018)
fairness.fomr	False omission rate equality (Berk et al., 2018)
fairness.tnr	Negative predictive value equality
fairness.tnr	Positive predictive value equality
fairness.cv	Demographic parity / Equalized positive rates (Calders and Verwer, 2010)
fairness.pp	Predictive parity / Equalized precision (Chouldechova, 2017)
fairness.{tp, fp, tn, fn}	Equal true positives, false positives, true negatives, false negatives
fairness.acc_eod=.05	Accuracy under equalized odds constraint (Perrone et al., 2021)

key	description
fairness.acc_ppv=.05	Accuracy under ppv constraint (Perrone et al., 2021)

Furthermore, new custom fairness measures can be easily implemented, either by implementing them directly or by composing them from existing metrics. This process is extensively documented in an accompanying measures vignette available with the package.

Here we choose the binary accuracy measure "classif.acc" and the equalized odds metric from above using "fairness.eod": The constructed list of measures can then be used to score a Prediction, a ResampleResult or BenchmarkResult, e.g.

```
measures = list(msr("classif.acc"), msr("fairness.eod"))
rr$aggregate(measures)
```

```
#>              classif.acc fairness.equalized_odds
#>                  0.84078                 0.07939
```

We can clearly see a comparatively large difference in equalized odds at around 0.08. This means, that in total, the false positive rates (FPR) and true positive rates (TPR) on average differ by ~0.08, indicating that our model might exhibit a bias. Looking at the individual components yields a clearer picture. Here, we are looking at the confusion matrices of the combined predictions of the 10 folds, grouped by sensitive attribute:

```
fairness_tensor(rr)
```

```
#> $Male
#>         truth
#> response   <=50K    >50K
#>    <=50K 0.43030 0.10033
#>    >50K  0.03408 0.11202
#>
#> $Female
#>         truth
#> response    <=50K     >50K
#>    <=50K 0.282668 0.020900
#>    >50K  0.003907 0.015789
```

Plotting the prediction density or comparing measures graphically often provides additional insights: For example, in Figure 2, we can see that Females are more often predicted to earn below $50.000. Similarly, we can see that both equality in FPR and TPR differ considerably.

```
fairness_prediction_density(prediction, task)
compare_metrics(prediction, msrs(c("fairness.fpr", "fairness.tpr", "fairness.eod")), task)
```

Bias mitigation

As mentioned above, several ways to improve a model's fairness exist. While non-technical interventions, such as e.g. collecting more data should be preferred, **mlr3fairness** provides several bias mitigation techniques that can be used together with a Learner to obtain fairer models. Table 2 provides an overview of implemented bias mitigation techniques. They are implemented as PipeOps from the **mlr3pipelines** package and can be combined with arbitrary learners using the %>>% operator to build a pipeline that can later be trained. In the following example, we show how to combine a learner with a reweighing scheme (reweighing_wts) or alternatively how to post-process predictions using the equalized odds debiasing (EOd) strategy. An introduction to **mlr3pipelines** is available in the corresponding mlr3book chapter (Bernd Bischl 2024).

```
po("reweighing_wts") %>>% lrn("classif.glmnet")
po("learner_cv", lrn("classif.glmnet")) %>>% po("EOd")
```

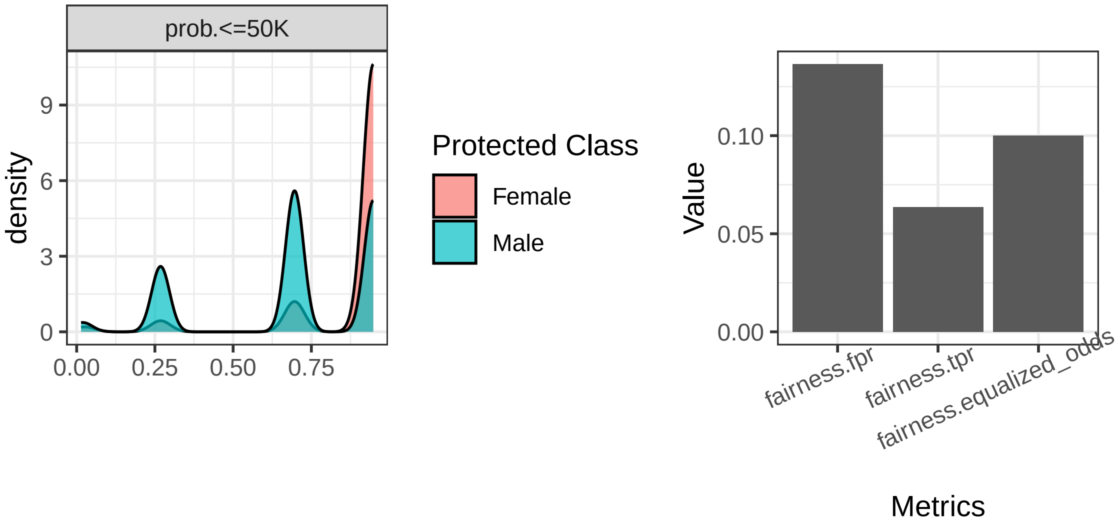


Figure 2: Visualizing predictions of the decision tree model. Left: Prediction densities for the negative class for Female and Male. Right: Fairness metrics comparison for FPR, TPR, EOd metrics. Plots show a higher likelihood for the '<50k' class for females resulting in fairness metrics different from 0.

Table 2: Overview of bias mitigation techniques available in mlr3fairness.

Key	Description	Type	Reference
EOd	Equalized-Odds Debiasing	Postprocessing	Hardt, Price, and Srebro (2016)
reweighing_os	Reweighing (Oversampling)	Preprocessing	Kamiran and Calders (2012)
reweighing_wts	Reweighing (Instance weights)	Preprocessing	Kamiran and Calders (2012)

It is simple for users or package developers to extend **mlr3fairness** with additional bias mitigation methods – as an example, the **mcboost** package adds further post-processing methods that can improve fairness. Along with pipeline operators, **mlr3fairness** contains several machine learning algorithms listed in table 3 that can directly incorporate fairness constraints. They can similarly be constructed using the `lrn()` shorthand.

Table 3: Overview of fair ML algorithms available with mlr3fairness.

Key	Package	Reference
regr.fairfrrm	fairml	Scutari, Panero, and Proissl (2021)
classif.fairfgrrm	fairml	Scutari, Panero, and Proissl (2021)
regr.fairzlm	fairml	Zafar et al. (2017)
classif.fairzlrm	fairml	Zafar et al. (2017)
regr.fairnclm	fairml	Komiyama et al. (2018)

Reports

Because fairness aspects can not always be investigated based on the fairness definitions above (e.g., due to biased sampling or labelling procedures), it is important to document data collection and the resulting data as well as the models resulting from this data. Informing auditors about those aspects of a deployed model can lead to better assessments of a model's fairness. Questionnaires for ML models (M. Mitchell et al. 2019) and data sets (Gebru et al. 2021) have been proposed in literature. We further add automated report templates using R markdown (Xie, Dervieux, and Riederer 2020) for data sets and ML models. In addition, we provide a template for a *fairness report* which includes many fairness metrics and visualizations to provide a good starting point for generating a fairness report inspired by the *Aequitas Toolkit* (Saleiro et al. 2018). A preview for the different reports can be obtained from the Reports vignette in the package documentation.

Table 4: Overview of reports generated by mlr3fairness.

Report	Description	Reference
`report_modelcard()`	Modelcard for ML models	M. Mitchell et al. (2019)
`report_datasheet()`	Datasheet for data sets	Gebru et al. (2021)
`report_fairness()`	Fairness Report	–

4 Case study

In order to demonstrate a full workflow, we conduct full bias assessment and bias mitigation on the popular adult data set (Dua and Graff 2017). The goal is to predict whether an individual's income is larger than $50.000 with the sensitive attribute being *gender*. The data set is included with **mlr3fairness**, separated into a *train* and *test* task and can be instantiated using `tsk("adult_train")` and `tsk("adult_test")`, respectively. As a fairness metric, we consider *true positive parity* which calls for equality in the true positive rates across groups, in this case the sex variable. We furthermore are interested in the model's utility, here measured as its classification accuracy.

```
library("mlr3verse")
library("mlr3fairness")

task = tsk("adult_train")
print(task)

#> <TaskClassif:adult_train> (30718 x 13)
#> * Target: target
#> * Properties: twoclass
#> * Features (12):
#>   - fct (7): education, marital_status, occupation, race, relationship,
#>     sex, workclass
#>   - int (5): age, capital_gain, capital_loss, education_num,
#>     hours_per_week
#> * Protected attribute: sex

measures = msrs(c("fairness.tpr", "classif.acc"))
```

In order to get an initial perspective, we benchmark three models using 3-fold cross-validation each:

- a classification tree from the **rpart** package,
- a penalized logistic regression from the **glmnet** package and
- a penalized logistic regression from the **glmnet** package, but with reweighing pre- processing.

The logistic regression in the latter two approaches does not support operating on factor features natively, therefore we pre-process the data with a feature encoder from **mlr3pipelines**. To achieve this, we connect the feature encoder `po("encode")` with the learner using the `%>>%` operator. This encodes factor variables into integers using dummy encoding. We then evaluate all three learners on the `adult_train` data using 3-fold cross-validation by building up a grid of experiments we want to run using `benchmark_grid`. This grid is then executed using the `benchmark` function, and we can aggregate the performance and fairness metric scores via the `$aggregate()` function.

```
set.seed(4321)
learners = list(
    lrn("classif.rpart"),
    po("encode") %>>% lrn("classif.glmnet"),
    po("encode") %>>% po("reweighing_wts") %>>% lrn("classif.glmnet")
)

grid = benchmark_grid(
  tasks = tsks("adult_train"),
  learners = learners,
  resamplings = rsmp("cv", folds = 3)
)
```

```
bmr1 = benchmark(grid)
bmr1$aggregate(measures)[, c(4, 7, 8)]
```

```
#>                            learner_id fairness.tpr classif.acc
#> 1:                      classif.rpart     0.059767      0.8408
#> 2:              encode.classif.glmnet     0.070781      0.8411
#> 3: encode.reweighing_wts.classif.glmnet     0.004732      0.8351
```

The pre-processing step of reweighing already improved the fairness while sacrificing only a tiny bit of performance. To see if we can further improve, we use **mlr3tuning** to jointly tune all hyperparameters of the *glmnet* model as well as our reweighing hyperparameter. In order to do this, we use an AutoTuner from **mlr3tuning**; a model that tunes its own hyperparameters during training. The full code for setting up this model can be found in the appendix. An AutoTuner requires a specific metric to tune for. Here, we define a fairness-thresholded accuracy metric. We set $\epsilon = 0.01$ as a threshold:

$$if\ |\Delta_{EOd}| \leq \epsilon : \text{accuracy}\ \ else:\ -|\Delta_{EOd}|.$$

```
metric = msr("fairness.constraint",
    performance_measure = msr("classif.acc"),
    fairness_measure = msr("fairness.eod"),
    epsilon = 0.01
)
```

We then design the pipeline and the hyperparameters we want to tune over. In the following example, we choose tuning_iters = 3 and set a small range for the hyperparameters in vals to shorten the run time of the tuning procedure. In real settings, this parameter would be set to a larger number, such as 100. To construct a self-tuning learner, we construct an AutoTuner that takes as input a learner, the resampling procedure and metric used for tuning as well as the tuning strategy along with a termination criterion (here how many tuning iterations should be run). In addition, we provide a new id for the learner to beautify subsequent printing and visualization. We can then use this self-tuning learner like any other learner and benchmark it using benchmark as described above.

```
tuning_iters = 3
at = AutoTuner$new(lrn, rsmp("holdout"),
    metric,
    tuner = mlr3tuning::tnr("random_search"),
    terminator = trm("evals", n_evals = tuning_iters)
)
at$id = "glmnet_weighted_tuned"

grd = benchmark_grid(
  tasks = tsks("adult_train"),
  learners = list(at),
  resamplings = rsmp("cv", folds = 3)
)

bmr2 = benchmark(grd, store_models = TRUE)
bmr2$aggregate(measures)[, c(4, 7, 8)]
```

```
#>               learner_id fairness.tpr classif.acc
#> 1: glmnet_weighted_tuned     0.009486      0.8385
```

The result improves w.r.t. accuracy while only slightly decreasing the measured fairness. Note that the generalization error is estimated using a holdout strategy during training and slight violations of the desired threshold ϵ should therefore be considered (Feurer et al. 2023). The results of both benchmark experiments can then be collected and jointly visualized in Figure 3 visualizing accuracy and fairness of models in our benchmark. In addition to aggregate scores (denoted by a cross) individual iterations of the 3-fold Cross-Validation (represented by points) are shown to visualize variations in the individual results.

```
bmr$aggregate(measures)[, c(4, 7, 8)]
```

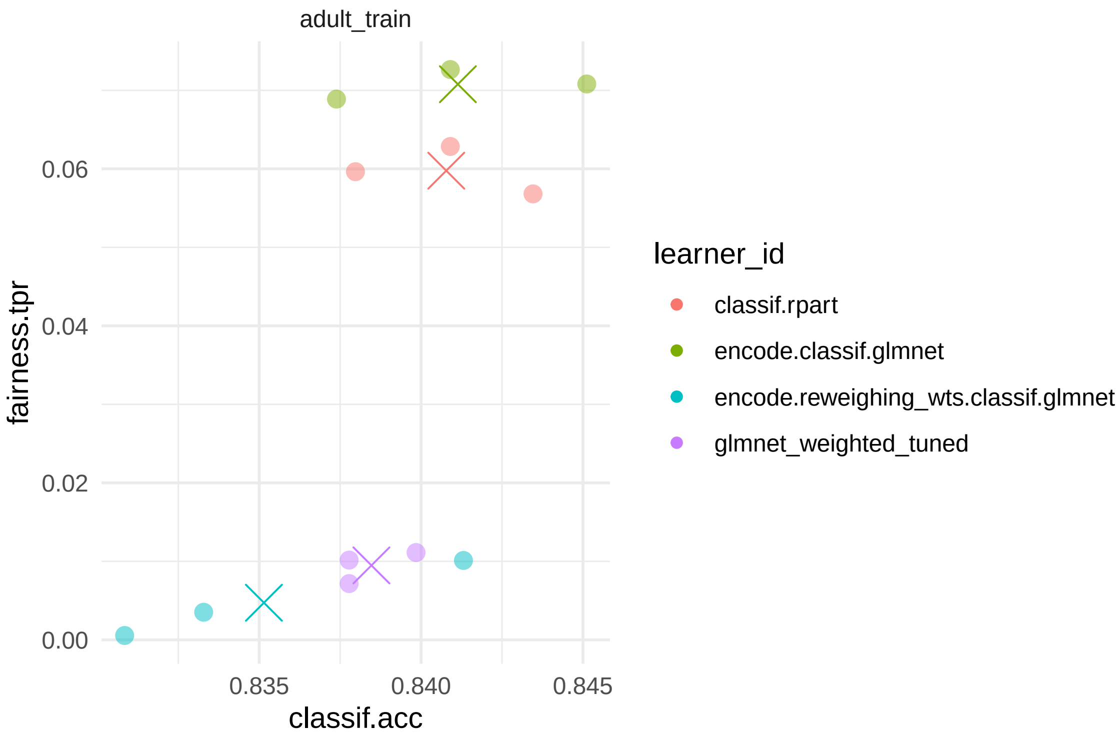


Figure 3: Fairness-Accuracy tradeoff for 3-fold CV on the adult train set.

```
#>                               learner_id fairness.tpr classif.acc
#> 1:                         classif.rpart     0.059767      0.8408
#> 2:                 encode.classif.glmnet     0.070781      0.8411
#> 3: encode.reweighing_wts.classif.glmnet     0.004732      0.8351
#> 4:                 glmnet_weighted_tuned     0.009486      0.8385
```

Especially when considering optimizing accuracy while still retaining a fair model, tuning can be helpful and further improve upon available trade-offs. In this example, the AutoTuner improves w.r.t. the fairness metric while offering accuracy comparable with the simple glmnet model. This can be observed from the fairness accuracy tradeoff shown in Figure 3. Whether the achieved accuracy is sufficient needs to be determined, e.g. from a business context. For now, we assume that the model obtained from the AutoTuner is the model we might want to use going forward. Having decided on a final model, we can now train the final model on the full training data

```
at_lrn = bmr$learners$learner[[4]]
at_lrn$train(tsk("adult_train"))
```

and predict on the held out *test* set available for the Adult dataset to obtain a final estimate. This is important since estimating fairness metrics often incurs significant variance (Agrawal, Pfisterer, et al. 2020) and evaluation of the test-set provides us with an unbiased estimate of model performance after the previous model selection step.

```
test = tsk("adult_test")
at_lrn$predict(test)$score(measures, test)
```

```
#> fairness.tpr  classif.acc
#>      0.07141      0.84375
```

On the held-out test set, the fairness constraint is slightly violated which can happen due to the comparatively large variance in the estimation of fairness metrics.

5 Summary

The large-scale availability and use of automated decision making systems have resulted in growing concerns for a lack of fairness in the decisions made by such systems. As a result, fairness auditing methods, that allow for investigating (un-)fairness in such systems are an important step towards improving the auditability of deployed systems. For ease of use, it is especially important, that they provide interoperability with machine learning toolkits that allows for ease of use and integration into model evaluation and tuning. In future work we plan on implementing several tools that further support the user w.r.t. pinpointing potential fairness issues in the data, especially through the help of interpretability tools, such as the **iml** (Molnar, Bischl, and Casalicchio 2018) package. We furthermore aim to implement additional fairness metrics from the realm of 'individual fairness' (Dwork et al. 2012) and 'conditional demographic parity' (Wachter, Mittelstadt, and Russell 2020).

6 Appendix

Tuning the ML pipeline

We include the full code to construct the AutoTuner with additional details and comments below. We first load all required packages and use **mlr3**'s interaction with the **future** (Bengtsson 2021) package to automatically distribute the tuning to all available cores in parallel by setting a plan. See the documentation of **future** for platform-specific hints regarding parallelization.

```
library(mlr3misc)
library(mlr3)
library(mlr3pipelines)
library(mlr3fairness)
library(mlr3tuning)

# Enable parallelization utilizing all cores
future::plan("multicore")
```

We then instantiate an ML pipeline using **mlr3pipelines**. This connects several modelling steps, in our case **categorical encoding**, **reweighing** and a final **learner** using the %>>% (double caret) operator, ultimately forming a new learner. This learner can then subsequently be fit on a Task. We use the po(<key>) shorthand to construct a new pipeline operator from a dictionary of implemented operators. We conduct **categorical encoding** because **glmnet** can not naturally handle categorical variables, and we therefore have to encode them (in our case using one-hot encoding).

```
# Define the learner pipeline.
lrn = as_learner(po("encode") %>>% po("reweighing_wts") %>>%
  po("learner", lrn("classif.glmnet")))
```

In addition, we have to specify the hyperparameter space our Tuner should tune over. We do this by defining a list of values with a to_tune() token specifying the range. Note, that hyperparameter names are prefixed with the respective operation's id.

```
# Define the parameter space to optimize over
vals = list(
  reweighing_wts.alpha = to_tune(0.75, 1),
  classif.glmnet.alpha = to_tune(0.5, 1),
  classif.glmnet.s = to_tune(1e-4, 1e-2, logscale = TRUE)
)

# Add search space to the learner
lrn$param_set$values = insert_named(lrn$param_set$values, vals)
```

Before we now train the model, we again specify a metric we aim to satisfy, here we would like the equalized odds difference to be smaller than 0.1. In this case, we set a constraint on the *equalized odds difference* comprised of the differences in true positive rate (TPR) and false positive rate (FPR):

$$\Delta_{EOd} = \frac{|\text{TPR}_{sex=M} - \text{TPR}_{sex=F}| + |\text{FPR}_{sex=M} - \text{FPR}_{sex=F}|}{2}.$$

This can be done using the fairness.constraint measure.

```
metric = msr("fairness.constraint",
    performance_measure = msr("classif.acc"),
    fairness_measure = msr("fairness.eod"),
    epsilon = 0.1
)
```

We can now instantiate a new AutoTuner using lrn defined above by additionally providing arguments specifying the tuning strategy, in our case random search, the measure to optimize for as well as the number of tuning steps.

```
metric = msr("fairness.constraint",
    performance_measure = msr("classif.acc"),
    fairness_measure = msr("fairness.eod"),
    epsilon = 0.1
)

at = AutoTuner$new(
  learner = lrn, # The learner
  resampling = rsmp("holdout"), # inner resampling strategy
  measure = metric, # the metric to optimize for
  tuner = mlr3tuning::tnr("random_search"), # tuning strategy
  terminator = trm("evals", n_evals = 30) # number of tuning steps
)
```

The so-constructed AutoTuner can now be used on any classification Task! Additional information regarding the AutoTuner is again available in the corresponding mlr3book chapter. In the following example, we will apply it to the Adult task and train our model. This will perform a tuning loop for the specified number of evaluations and automatically retrain the best found parameters on the full data.

```
at$train(tsk("adult_train"))
```

After training, we can look at the best models found, here ordered by our metric. Note, that our metric reports the negative constraint violation if the constraint is violated and the accuracy in case the constraint is satisfied.

```
head(at$archive$data[order(fairness.acc_equalized_odds_cstrt), 1:4])

#>    reweighing_wts.alpha classif.glmnet.alpha classif.glmnet.s
#> 1:               0.9503               0.5138           -4.654
#> 2:               0.8160               0.9289           -5.314
#> 3:               0.7507               0.6784           -5.786
#> 4:               0.9694               0.9219           -6.198
#> 5:               0.8254               0.8826           -7.241
#> 6:               0.8808               0.7362           -7.311
#>    fairness.acc_equalized_odds_cstrt
#> 1:                            0.8411
#> 2:                            0.8445
#> 3:                            0.8447
#> 4:                            0.8448
#> 5:                            0.8448
#> 6:                            0.8448
```

We can then use the tuned model to assess our metric on the held out data:

```
prd = at$predict(tsk("adult_test"))
prd$score(c(metric, msr("classif.acc"), msr("fairness.eod")),  tsk("adult_test"))

#> fairness.acc_equalized_odds_cstrt                       classif.acc
#>                           0.83976                           0.83976
#>           fairness.equalized_odds
#>                           0.07512
```

So our tuned model manages to obtain an accuracy of ~0.84 while satisfying the specified constraint of $\Delta_{EOd} < 0.1$. So to summarize, we have tuned a model to optimize accuracy with respect to a constraint on a selected fairness metric using an AutoTuner.

References

Agrawal, Ashrya, Jiahao Chen, Sebastian Vollmer, and Anthony Blaom. 2020. "Fairness.jl." Zenodo. https://doi.org/10.5281/zenodo.3977197.

Agrawal, Ashrya, Florian Pfisterer, Bernd Bischl, Jiahao Chen, Srijan Sood, Sameena Shah, Francois Buet-Golfouse, Bilal A Mateen, and Sebastian Vollmer. 2020. "Debiasing Classifiers: Is Reality at Variance with Expectation?" *arXiv:2011.02407*.

Aivodji, Ulrich, Hiromi Arai, Olivier Fortineau, Sébastien Gambs, Satoshi Hara, and Alain Tapp. 2019. "Fairwashing: The Risk of Rationalization." In *International Conference on Machine Learning*, 97:161–70. PMLR.

Angwin, Julia, Jeff Larson, Surya Mattu, and Lauren Kichner. 2016. "Machine Bias." ProPublica. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

Bao, Michelle, Angela Zhou, Samantha A Zottola, Brian Brubach, Sarah Desmarais, Aaron Seth Horowitz, Kristian Lum, and Suresh Venkatasubramanian. 2021. "It's COMPASlicated: The Messy Relationship Between RAI Datasets and Algorithmic Fairness Benchmarks." In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Barocas, Solon, Moritz Hardt, and Arvind Narayanan. 2019. *Fairness and Machine Learning*. fairmlbook.org. https://fairmlbook.org/.

Becker, Marc, Michel Lang, Jakob Richter, Bernd Bischl, and Daniel Schalk. 2023. *Mlr3tuning: Hyperparameter Optimization for 'Mlr3'*.

Becker, Marc, Jakob Richter, Michel Lang, Bernd Bischl, and Martin Binder. 2023. *Bbotk: Black-Box Optimization Toolkit*.

Bellamy, Rachel KE, Kuntal Dey, Michael Hind, Samuel C Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, et al. 2019. "AI Fairness 360: An Extensible Toolkit for Detecting and Mitigating Algorithmic Bias." *IBM Journal of Research and Development* 63 (4/5): 4–1. https://aif360.mybluemix.net/.

Bengtsson, Henrik. 2021. "A Unifying Framework for Parallel and Distributed Processing in R Using Futures." *The R Journal* 13 (2): 208–27. https://doi.org/10.32614/RJ-2021-048.

Berk, Richard, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. 2018. "Fairness in Criminal Justice Risk Assessments: The State of the Art." *Sociological Methods & Research*, August. https://doi.org/10.1177/0049124118782533.

Bernd Bischl, Lars Kotthoff, Raphael Sonabend, ed. 2024. *Applied Machine Learning Using Mlr3 in R*. CRC Press. https://mlr3book.mlr-org.com.

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.

Binder, Martin, Florian Pfisterer, Michel Lang, Lennart Schneider, Lars Kotthoff, and Bernd Bischl. 2021. "mlr3pipelines - Flexible Machine Learning Pipelines in R." *Journal of Machine Learning Research* 22 (184): 1–7. https://jmlr.org/papers/v22/21-0281.html.

Binns, Reuben. 2020. "On the Apparent Conflict Between Individual and Group Fairness." In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 514–24. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3351095.3372864.

Bird, Sarah, Miro Dudík, Richard Edgar, Brandon Horn, Roman Lutz, Vanessa Milan, Mehrnoosh Sameki, Hanna Wallach, and Kathleen Walker. 2020. "Fairlearn: A Toolkit for Assessing and Improving Fairness in AI." MSR-TR-2020-32. Microsoft. https://www.microsoft.com/en-us/research/publication/fairlearn-a-toolkit-for-assessing-and-improving-fairness-in-ai/.

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.

Bischl, Bernd, Olaf Mersmann, Heike Trautmann, and Claus Weihs. 2012. "Resampling Methods for Meta-Model Validation with Recommendations for Evolutionary Computation." *Evolutionary Computation* 20 (2): 249–75.

Buolamwini, Joy, and Timnit Gebru. 2018. "Gender shades: Intersectional accuracy disparities in commercial gender classification." In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 77–91. PMLR.

Calders, Toon, and Sicco Verwer. 2010. "Three naive Bayes approaches for discrimination-free classification." *Data Mining and Knowledge Discovery* 21 (2): 277–92. https://doi.org/10.1007/s10618-010-0190-x.

Chen, Jiahao. 2018. "Fair Lending Needs Explainable Models for Responsible Recommendation." In *Proceedings of the 2nd FATREC Workshop on Responsible Recommendation*.

Chouldechova, Alexandra. 2017. "Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments." *Big Data* 5 (2): 153–63. https://doi.org/10.1089/big.2016.0047.

Cirillo, Davide, Silvina Catuara-Solarz, Czuee Morey, Emre Guney, Laia Subirats, Simona Mellino, Annalisa Gigante, et al. 2020. "Sex and Gender Differences and Biases in Artificial Intelligence for Biomedicine and Healthcare." *NPJ Digital Medicine* 3 (1): 1–11. https://doi.org/10.1038/s41746-020-0288-5.

Corbett-Davies, Sam, and Sharad Goel. 2018. "The Measure and Mismeasure of Fairness: A Critical Review of Fair Machine Learning." *arXiv:1808.00023*.

Corbett-Davies, Sam, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. 2017. "Algorithmic Decision Making and the Cost of Fairness." In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 797–806. KDD '17. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3097983.3098095.

Dawes, Robyn M, David Faust, and Paul E Meehl. 1989. "Clinical Versus Actuarial Judgment." *Science* 243 (4899): 1668–74. https://doi.org/10.1126/science.2648573.

Dua, Dheeru, and Casey Graff. 2017. "UCI Machine Learning Repository." University of California, Irvine, School of Information; Computer Sciences. http://archive.ics.uci.edu/ml.

Dwork, Cynthia, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. "Fairness Through Awareness." In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 214–26.

Eubanks, Virginia. 2018. *Automating Inequality: How High-Tech Tools Profile, Police, and Punish the Poor*. St. Martin's Press.

Feurer, Matthias, Katharina Eggensperger, Edward Bergman, Florian Pfisterer, Bernd Bischl, and Frank Hutter. 2023. "Mind the Gap: Measuring Generalization Performance Across Multiple Objectives." In *Advances in Intelligent Data Analysis XXI*, 130–42.

Friedler, Sorelle A., Carlos Scheidegger, and Suresh Venkatasubramanian. 2016. "On the (Im)possibility of Fairness." *arXiv:1609.07236*.

Galindo, Jorge, and Pablo Tamayo. 2000. "Credit Risk Assessment Using Statistical and Machine Learning: Basic Methodology and Risk Modeling Applications." *Computational Economics* 15 (1): 107–43.

Gebru, Timnit, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. 2021. "Datasheets for Datasets." *Communications of the ACM* 64 (12): 86–92.

Hardt, Moritz, Eric Price, and Nati Srebro. 2016. "Equality of Opportunity in Supervised Learning." *Advances in Neural Information Processing Systems* 29: 3315–23.

Kamiran, Faisal, and Toon Calders. 2012. "Data Preprocessing Techniques for Classification Without Discrimination." *Knowledge and Information Systems* 33 (1): 1–33.

Kilbertus, Niki, Mateo Rojas Carulla, Giambattista Parascandolo, Moritz Hardt, Dominik Janzing, and Bernhard Schölkopf. 2017. "Avoiding Discrimination Through Causal Reasoning." *Advances in Neural Information Processing Systems* 30.

Kim, Joon Sik, Jiahao Chen, and Ameet Talwalkar. 2020. "Fact: A Diagnostic for Group Fairness Trade-Offs." In *International Conference on Machine Learning*, 5264–74. PMLR.

Komiyama, Junpei, Akiko Takeda, Junya Honda, and Hajime Shimao. 2018. "Nonconvex Optimization for Regression with Fairness Constraints." In *International Conference on Machine Learning*, 2737–46. PMLR.

Kozodoi, Nikita, Johannes Jacob, and Stefan Lessmann. 2022. "Fairness in Credit Scoring: Assessment, Implementation and Profit Implications." *European Journal of Operational Research* 297 (3): 1083–94. https://doi.org/10.1016/j.ejor.2021.06.023.

Kozodoi, Nikita, and Tibor V. Varga. 2021. *Fairness: Algorithmic Fairness Metrics*. https://CRAN.R-project.org/package=fairness.

Kuhn, Max. 2021. *Caret: Classification and Regression Training*. https://CRAN.R-project.org/package=caret.

Kuhn, Max, and Hadley Wickham. 2020. *Tidymodels: A Collection of Packages for Modeling and Machine Learning Using Tidyverse Principles.* https://www.tidymodels.org.

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.

Lang, Michel, and Patrick Schratz. 2023. *Mlr3verse: Easily Install and Load the 'Mlr3' Package Family*.

Mehrabi, Ninareh, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. "A Survey on Bias and Fairness in Machine Learning." *ACM Computing Surveys (CSUR)* 54 (6): 1–35.

Mitchell, Margaret, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. "Model Cards for Model Reporting." In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 220–29. New York, NY, USA: PMLR; Association for Computing Machinery. https://doi.org/10.1145/3287560.3287596.

Mitchell, Shira, Eric Potash, Solon Barocas, Alexander D'Amour, and Kristian Lum. 2021. "Algorithmic fairness: Choices, assumptions, and definitions." *Annual Review of Statistics and Its Application* 8: 141–63. https://doi.org/10.1146/annurev-statistics-042720-125902.

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.

Noble, Safiya Umoja. 2018. *Algorithms of Oppression*. New York University Press.

O'neil, Cathy. 2016. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30.

Perrone, Valerio, Michele Donini, Muhammad Bilal Zafar, Robin Schmucker, Krishnaram Kenthapadi, and Cédric Archambeau. 2021. "Fair Bayesian Optimization." In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 854–63.

Pfisterer, Florian, Christoph Kern, Susanne Dandl, Matthew Sun, Michael P. Kim, and Bernd Bischl. 2021. "Mcboost: Multi-Calibration Boosting for R." *Journal of Open Source Software* 6 (64): 3453. https://doi.org/10.21105/joss.03453.

Polley, Eric, Erin LeDell, Chris Kennedy, and Mark van der Laan. 2021. *SuperLearner: Super Learner Prediction*. https://CRAN.R-project.org/package=SuperLearner.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. http://www.R-project.org/.

Saleiro, Pedro, Benedict Kuester, Loren Hinkson, Jesse London, Abby Stevens, Ari Anisfeld, Kit T Rodolfa, and Rayid Ghani. 2018. "Aequitas: A Bias and Fairness Audit Toolkit." *arXiv:1811.05577*.

Schumann, Candice, Jeffrey S. Foster, Nicholas Mattei, and John P. Dickerson. 2020. "We Need Fairness and Explainability in Algorithmic Hiring." In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 1716–20. AAMAS '20. Auckland, New Zealand: International Foundation for Autonomous Agents; Multiagent Systems.

Schwöbel, P., and P. Remmers. 2022. "The Long Arc of Fairness: Formalisations and Ethical Discourse." In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2179–88. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3531146.3534635.

Scutari, Marco, Francesca Panero, and Manuel Proissl. 2021. "Achieving Fairness with a Simple Ridge Penalty." *arXiv:2105.13817*.

Sonabend, Raphael, Franz J Király, Andreas Bender, Bernd Bischl, and Michel Lang. 2021. "mlr3proba: An R Package for Machine Learning in Survival Analysis." *Bioinformatics*, February. https://doi.org/10.1093/bioinformatics/btab039.

Topol, Eric J. 2019. "High-Performance Medicine: The Convergence of Human and Artificial Intelligence." *Nature Medicine* 25 (1): 44–56. https://doi.org/10.1038/s41591-018-0300-7.

Turner, Matthew, and Michael McBurnett. 2019. "Predictive Models with Explanatory Concepts: A General Framework for Explaining Machine Learning Credit Risk Models That Simultaneously Increases Predictive Power." In *Proceedings of the 15th Credit Scoring and Credit Control Conference*. https://crc.business-school.ed.ac.uk/wp-content/uploads/sites/55/2019/07/C12-Predictive-Models-with-Explanatory-Concepts-McBurnett.pdf.

Vanschoren, Joaquin, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2014. "OpenML." *ACM SIGKDD Explorations Newsletter* 15 (2): 49–60. https://doi.org/10.1145/2641190.2641198.

Wachter, S., B. Mittelstadt, and C. Russell. 2020. "Bias Preservation in Machine Learning: The Legality of Fairness Metrics Under EU Non-Discrimination Law." *West Virginia Law Review* 123.

Watkins, Elizabeth Anne, Michael McKenna, and Jiahao Chen. 2022. "The Four-Fifths Rule Is Not Disparate Impact: A Woeful Tale of Epistemic Trespassing in Algorithmic Fairness." *arXiv:2202.09519*.

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

Wiśniewski, Jakub, and Przemysław Biecek. 2022. "Fairmodels: A Flexible Tool for Bias Detection, Visualization, and Mitigation in Binary Classification Models." *The R Journal* 14: 227–43. https://rj.urbanek.nz/articles/RJ-2022-019/.

Xie, Yihui, Christophe Dervieux, and Emily Riederer. 2020. *R Markdown Cookbook*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook.

Zafar, Muhammad Bilal, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P. Gummadi. 2017. "Fairness beyond Disparate treatment & Disparate Impact." In *Proceedings of the 26th International Conference on World Wide Web*, 1171–80. Geneva, Switzerland: International World Wide Web Conferences Steering Committee. https://doi.org/10.1145/3038912.3052660.

Bibliography

R. Berk, H. Heidari, S. Jabbari, M. Kearns, and A. Roth. Fairness in Criminal Justice Risk Assessments: The State of the Art. *Sociological Methods & Research*, Aug. 2018. doi: 10.1177/0049124118782533. [p241]

J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 77–91. PMLR, 2018. [p241]

T. Calders and S. Verwer. Three naive Bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, 21(2):277–292, 2010. doi: 10.1007/s10618-010-0190-x. [p241]

A. Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data*, 5(2):153–163, June 2017. doi: 10.1089/big.2016.0047. [p241]

M. Hardt, E. Price, and N. Srebro. Equality of opportunity in supervised learning. *Advances in Neural Information Processing Systems*, 29:3315–3323, 2016. [p241]

V. Perrone, M. Donini, M. B. Zafar, R. Schmucker, K. Kenthapadi, and C. Archambeau. Fair Bayesian Optimization. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 854–863, 2021. [p241, 242]

Florian Pfisterer
Ludwig-Maximilians-Universität München
Munich, Germany
Munich Center for Machine Learning
Munich, Germany
ORCiD: 0000-0001-8867-762X
florian.pfisterer@stat.uni-muenchen.de

Siyi Wei
University of Toronto
Toronto, Canada
weisiyi2@gmail.com

Sebastian Vollmer
Deutsches Forschungszentrum für Künstliche Intelligenz
Kaiserslautern, Germany
University of Kaiserslautern
Kaiserslautern, Germany
ORCiD: 0000-0002-9025-0753
Sebastian.vollmer@dfki.de

Michel Lang
Research Center Trustworthy Data Science and Security
Dortmund, Germany
TU Dortmund University
Dortmund, Germany
ORCiD: 0000-0001-9754-0393
michel.lang@stat.uni-muenchen.de

Bernd Bischl
Ludwig-Maximilians-Universität München
Munich, Germany
Munich Center for Machine Learning
Munich, Germany
ORCiD: 0000-0001-6002-6980
bernd.bischl@stat.uni-muenchen.de

ClusROC: An R Package for ROC Analysis in Three-Class Classification Problems for Clustered Data

by Duc-Khanh To, Gianfranco Adimari, and Monica Chiogna

Abstract This paper introduces an R package for ROC analysis in three-class classification problems, for clustered data in the presence of covariates, named ClusROC. The clustered data that we address have some hierarchical structure, i.e., dependent data deriving, for example, from longitudinal studies or repeated measurements. This package implements point and interval covariate-specific estimation of the true class fractions at a fixed pair of thresholds, the ROC surface, the volume under the ROC surface, and the optimal pairs of thresholds. We illustrate the usage of the implemented functions through two practical examples from different fields of research.

1 Introduction

In clinical studies, receiver operating characteristic (ROC) surface analysis is widely used to evaluate the accuracy of a diagnostic test (or biomarker) when there are three ordinal disease classes (or diagnostic groups). See Nakas (2014) for a comprehensive review. Although the clinical context is where ROC surface analysis finds its natural application, other contexts, such as economics and engineering, very often face the problem of evaluating the accuracy of a classifier.

Within the ROC surface analysis framework, the following quantities are typically objects of interest: ROC surface, VUS (volume under ROC surface), and optimal pair of thresholds. Statistical methods for evaluating such quantities have been widely discussed in the statistical literature. We cite, among others, papers by Nakas and Yiannoutsos (2004); Xiong et al. (2006); Nakas et al. (2010); Attwood et al. (2014); Bantis et al. (2017); To Duc et al. (2016). Moreover, on the Comprehensive R Archive Network (CRAN), there are several packages implementing estimation methods for ROC surface analysis, for instance, **trinROC** (Noll and Reinhard, 2022), **ThresholdROC** (Sara et al., 2021) and **bcROCsurface** (To, 2021).

Most of the existing methods for ROC surface analysis have focused on a standard setting, in which measurements on statistical units are realizations of independent random variables, and the diagnostic test, or, more broadly, the classifier, is not influenced by any covariate. In some studies, however, not only the classifier can be affected by some covariates that characterize the units themselves (e.g. age, gender), but statistical units can be enrolled in clusters (e.g., families, genotype, communities, etc.). When statistical units are drawn from clusters, they can no longer be treated as independent. Indeed, units from the same cluster are typically more similar to each other than they will be to statistical units from other clusters. Therefore, unobserved variables may induce statistical dependence between observations within clusters that may be uncaptured by covariates. For such kinds of clustered data, which have a hierarchical structure and are dependent, Xiong et al. (2018) proposed the use of a standard linear mixed-effects model (McCulloch and Searle, 2001) to account for clusters and covariates' effects on the classifier. Then, the authors developed an approach to estimate the VUS, under an assumption of normality. Based on the model in Xiong et al. (2018), To et al. (2022) developed an estimation procedure for the ROC surface and methods for choosing the optimal pair of thresholds. The authors also discussed a variant of their approach, based on the Box-Cox transformation, useful when the normality assumption is (not heavily) violated.

In this paper, we introduce our R package for ROC surface analysis with clustered data, named **ClusROC**. The package (with related details) is available on CRAN at `http://CRAN.Rproject.org/package=ClusROC`. In the package, we implement procedures for estimating the parameters of the models (with and without the Box-Cox transformation), and for making inferences about the ROC surface, and the optimal pair of thresholds, by following methods outlined in To et al. (2022). In addition, we also implement a procedure for estimating the VUS, as discussed in Xiong et al. (2018).

In the following sections, we first briefly present the reference model and the inferential procedures for ROC surface analysis with clustered data. Then, we describe the **ClusROC** package and illustrate its use through two real datasets. The last section provides a brief conclusion.

2 The ROC surface analysis for clustered data

Here we briefly review methods proposed by Xiong et al. (2018) and To et al. (2022). Details and theoretical results can be found in the original articles. For convenience, as in the quoted papers, we refer to some clinical studies in the presentation and use appropriate language for that area.

The models

Let Y be the diagnostic test result, on a continuous scale, and let Y_1, Y_2, Y_3 be the test results for subjects in classes 1, 2, and 3, respectively. We assume that higher values of test results are associated with higher severity of the disease, and the severity of the disease grows with the class (i.e., class 3 is the worst). Let $X_1, \ldots, X_p$ be p covariates, possibly associated with the test Y.

Let c be the total number of clusters, randomly selected from the population. For the k-th cluster, $k = 1, \ldots, c$, let n_{ki} be the total number of subjects belonging to class i, $i = 1, 2, 3$ and let $n_k = n_{k1} + n_{k2} + n_{k3}$ be the total sample size within the cluster. Note that n_{ki} might be equal to 0 for some clusters. The linear mixed-effects model for the clustering effect on the test result Y, as well as for covariates' effects, is written as follows (Xiong et al., 2018; To et al., 2022):

$$\begin{aligned} Y_1 &= \alpha_{k_1} + z_1^\top \boldsymbol{\beta}_1 + \varepsilon_1, \\ Y_2 &= \alpha_{k_2} + z_2^\top \boldsymbol{\beta}_2 + \varepsilon_2, \\ Y_3 &= \alpha_{k_3} + z_3^\top \boldsymbol{\beta}_3 + \varepsilon_3, \end{aligned} \tag{1}$$

where (Y_1, Y_2, Y_3) is a triplet of test scores from three randomly sampled subjects from the three disease classes, (k_1, k_2, k_3), $k_i \in \{1, \ldots, c\}$, are cluster memberships indicating the clusters from which Y_1, Y_2, Y_3 are observed, $z_i = (1, x_{1i}, \ldots, x_{pi})^\top$ are fixed (i.e., not random) covariates values, and $\boldsymbol{\beta}_i = (\beta_{0i}, \beta_{1i}, \ldots, \beta_{pi})^\top$, $i = 1, 2, 3$, are vectors of parameters representing covariates effects. In model (1), α_k are random effects accounting for the presence of clusters, and ε_i are subject-level random errors. We assume that: (i) the random effects α_k and the subject-level random errors ε_i follow a normal distribution, i.e., $\alpha_k \sim \mathcal{N}(0, \sigma_c^2)$ and $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$ with $i = 1, 2, 3$; (ii) $\alpha_1, \alpha_2, \ldots, \alpha_c$ and $\varepsilon_1, \varepsilon_2, \varepsilon_3$ are all independent (see also, McCulloch and Searle, 2001).

Let $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^\top, \boldsymbol{\beta}_2^\top, \boldsymbol{\beta}_3^\top)^\top$ with $\boldsymbol{\beta}_i = (\beta_{0i}, \beta_{1i}, \ldots, \beta_{pi})^\top$, and $\boldsymbol{\theta} = (\sigma_c, \sigma_1, \sigma_2, \sigma_3)^\top$ be the unknown parameters in model (1). By using a restricted (or residual) maximum likelihood (REML) estimation approach, a consistent estimator $\widehat{\boldsymbol{\gamma}} = (\widehat{\boldsymbol{\beta}}^\top, \widehat{\boldsymbol{\theta}}^\top)^\top$ of $\boldsymbol{\gamma} = (\boldsymbol{\beta}^\top, \boldsymbol{\theta}^\top)^\top$ can be obtained. Under some regularity conditions, we have $\widehat{\boldsymbol{\gamma}} \dot{\sim} \mathcal{N}(\boldsymbol{\gamma}, \boldsymbol{\Lambda})$. The asymptotic covariance matrix $\boldsymbol{\Lambda}$ can be consistently estimated by using the sandwich formula (Liang and Zeger, 1986; Kauermann and Carroll, 2001; Mancl and DeRouen, 2001).

In some practical situations, data distributions may be skewed and the normality assumption might be violated. For such cases, To et al. (2022) considered the application of Box-Cox transformation for linear mixed-effects models (Lipsitz et al., 2000; Gurka et al., 2006):

$$\begin{aligned} Y_1^{(\lambda)} &= \alpha_{k_1} + z_1^\top \boldsymbol{\beta}_1 + \varepsilon_1, \\ Y_2^{(\lambda)} &= \alpha_{k_2} + z_2^\top \boldsymbol{\beta}_2 + \varepsilon_2, \\ Y_3^{(\lambda)} &= \alpha_{k_3} + z_3^\top \boldsymbol{\beta}_3 + \varepsilon_3, \end{aligned} \tag{2}$$

where $Y_i^{(\lambda)}$ is the Box-Cox transformed response, $Y_i^{(\lambda)} = (Y_i^\lambda - 1)/\lambda$ if $\lambda \neq 0$ and $Y_i^{(\lambda)} = \log(Y_i)$ if $\lambda = 0$, with $i = 1, 2, 3$, $Y_i > 0$, and λ is the transformation parameter (Box and Cox, 1964). Assumptions about the random effects α_k and the subject-level random errors ε_i are the same as in model (1). To obtain $\widehat{\lambda}$ and the REML estimator $\widehat{\boldsymbol{\gamma}}$, To et al. (2022) applied the method proposed by Gurka and Edwards (2011) which is based on the scaled Box-Cox transformation model (Gurka et al., 2006). The estimator of the variance-covariance matrix of the REML estimator $\widehat{\boldsymbol{\gamma}}$ is obtained again by applying the sandwich formula.

ROC surface analysis

According to the model (1), at a given vector z of covariates' values, $Y_i \sim \mathcal{N}(z^\top \boldsymbol{\beta}_i, \sigma_c^2 + \sigma_i^2)$ with $z = (1, x_1, \ldots, x_p)^\top$ and $i = 1, 2, 3$. To et al. (2022) further assume that $z^\top \boldsymbol{\beta}_1 < z^\top \boldsymbol{\beta}_2 < z^\top \boldsymbol{\beta}_3$, i.e., that the stochastic dominance for the three classes holds at z.

For given thresholds t_1 and t_2 ($t_1 < t_2$), the covariate-specific true class fractions (TCFs) are written as $\mathrm{TCF}_1(t_1;z) = \Phi\left(\frac{t_1 - z^\top \boldsymbol{\beta}_1}{\sqrt{\sigma_c^2+\sigma_1^2}}\right)$, $\mathrm{TCF}_2(t_1,t_2;z) = \Phi\left(\frac{t_2 - z^\top \boldsymbol{\beta}_2}{\sqrt{\sigma_c^2+\sigma_2^2}}\right) - \Phi\left(\frac{t_1 - z^\top \boldsymbol{\beta}_2}{\sqrt{\sigma_c^2+\sigma_2^2}}\right)$ and $\mathrm{TCF}_3(t_2;z) = 1 - \Phi\left(\frac{t_2 - z^\top \boldsymbol{\beta}_3}{\sqrt{\sigma_c^2+\sigma_3^2}}\right)$, where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. Moreover, by setting $\mathrm{TCF}_1(t_1;z) = p_1$ and $\mathrm{TCF}_3(t_2;z) = p_3$, the covariate-specific ROC surface can be defined as a function of (p_1, p_3), i.e.,

$$\begin{aligned}
\mathrm{ROCs}(p_1,p_3;z) \quad = \quad & \Phi\left(\frac{\Phi^{-1}(1-p_3)\sqrt{\sigma_c^2+\sigma_3^2} + z^\top \boldsymbol{\beta}_3 - z^\top \boldsymbol{\beta}_2}{\sqrt{\sigma_c^2+\sigma_2^2}}\right) \\
& - \Phi\left(\frac{\Phi^{-1}(p_1)\sqrt{\sigma_c^2+\sigma_1^2} + z^\top \boldsymbol{\beta}_1 - z^\top \boldsymbol{\beta}_2}{\sqrt{\sigma_c^2+\sigma_2^2}}\right),
\end{aligned} \tag{3}$$

if $\Phi^{-1}(p_1) < \frac{\Phi^{-1}(1-p_3)\sqrt{\sigma_c^2+\sigma_3^2} + z^\top \boldsymbol{\beta}_3 - z^\top \boldsymbol{\beta}_1}{\sqrt{\sigma_c^2+\sigma_1^2}}$; otherwise, $\mathrm{ROCs}(p_1,p_3;z) = 0$.

Based on the above expressions, for clustered data, To et al. (2022) considered covariate-specific estimation of the TCFs at a fixed pair of thresholds (t_1, t_2) and estimation of the ROC surface for each pair (p_1, p_3). Moreover, the authors proposed methods to estimate covariate-specific optimal pairs of thresholds (t_1^+, t_2^+) by considering three different criteria: (i) maximization of the covariate-specific generalized Youden index (GYI); (ii) minimization of the covariate-specific Euclidean distance between the ideal point $(1,1,1)$ and the point $(\mathrm{TCF}_1(t_1;z), \mathrm{TCF}_2(t_1,t_2;z), \mathrm{TCF}_3(t_2;z))$ (CtP); (iii) maximization of the covariate-specific volume of the cuboid under the covariate-specific ROC surface (MV). Resulting estimators are shown to be consistent and asymptotically normal, with asymptotic covariance matrices estimable by using the plug-in method. The normal approximation results can be used to construct suitable (joint) confidence regions.

Starting from model (1), covariate-specific inference on the VUS is discussed in Xiong et al. (2018), where maximum likelihood (ML) methods for point and interval estimation are proposed. In particular, Xiong et al. (2018) showed that covariate-specific ML VUS estimator $\hat{\theta}(z)$ is the summation of five components: $\hat{\theta}_1(z) = \widehat{\Pr}(Y_1 < Y_2 < Y_3, k_1 = k_2 = k_3|z)$, $\hat{\theta}_2(z) = \widehat{\Pr}(Y_1 < Y_2 < Y_3, k_1 = k_2 \neq k_3|z)$, $\hat{\theta}_3(z) = \widehat{\Pr}(Y_1 < Y_2 < Y_3, k_1 = k_3 \neq k_2|z)$, $\hat{\theta}_4(z) = \widehat{\Pr}(Y_1 < Y_2 < Y_3, k_1 \neq k_2 = k_3|z)$, and $\hat{\theta}_5(z) = \widehat{\Pr}(Y_1 < Y_2 < Y_3, k_1 \neq k_2 \neq k_3|z)$.

In cases where the assumption of normality for Y_1, Y_2 and Y_3 is unrealistic, inferential procedures for covariate-specific TCFs, ROC surface, optimal pair of thresholds (and VUS) can be obtained starting from model (2); see Section 3.3 in To et al. (2022).

3 Overview of R package ClusROC

The **ClusROC** package implements techniques for ROC surface analysis, in case of clustered data and the presence of covariates. The package comprises five major functions:

- `clus_lme()`: This function fits the linear mixed-effects model (1) under the normality assumption, or the model (2) when the Box-Cox transformation is used.
- `clus_roc_surface()`: This function estimates and makes a 3D plot of covariate-specific ROC surfaces.
- `clus_opt_thres3()`: This function estimates the covariate-specific optimal pair of thresholds.
- `clus_vus()`: This function estimates the covariate-specific VUS.
- `clus_tcfs()`: This function estimates the covariate-specific TFCs at a specified pair of thresholds.

The **ClusROC** package can be installed directly from CRAN by using the code below:

```
install.packages("ClusROC")
```

Also, it can be installed from GitHub ("toduckhanh/ClusROC"), by using the function `install_github()` from **devtools** package:

```
library(devtools)
install_github("toduckhanh/ClusROC")
```

Description

The REML estimation in function clus_lme() is based on the function lme() of package **nlme** (Pinheiro et al., 2022). The function clus_lme() needs, firstly, specification of a fixed_formula which is a two-sided linear formula object describing the fixed-effects part of the model for three classes, with the response on the left of ~ operator and the terms, separated by + operators, on the right. Secondly, the arguments name_class and name_clust are needed to specify the name of the variables indicating the disease classes (or diagnostic groups) and the clusters in the data, respectively. To enable the Box-Cox transformation, users need to set the argument boxcox as TRUE. The Box-Cox parameter λ is estimated by a grid search on the interval $(-2, 2)$. This interval is suggested by Gurka and Edwards (2011), but users can change this range by setting the argument interval_lambda. Before fitting the model, clus_lme() determines the ordering of the disease groups based on the average values of test results in each disease group. If an ordering is provided by the user via the argument levl_class, the ordering of the mean values is still obtained to confirm the input ordering. In case of disagreement between the two orderings, the one based on the averages of test results is adopted. The function plot() provides three diagnostic plots for the model fitted by clus_lme(), namely, a Q-Q plot for residuals, a Fitted vs. Residuals plot, and a Q-Q plot for cluster effects. These plots exploit the **ggplot2** package (Hadley et al., 2022).

The functions clus_roc_surface(), clus_opt_thres3(), clus_vus() and clus_tcfs() are the main functions to perform the ROC surface analysis for clustered data. All of them require the output of clus_lme() as an argument. When one of the above functions is called, a check on the monotone ordering assumption is performed. That is, for a given value of the covariates, say z, the three predicted mean values of the test results in the three diagnostic groups, i.e., $z^\top \widehat{\boldsymbol{\beta}}_1$, $z^\top \widehat{\boldsymbol{\beta}}_2$ and $z^\top \widehat{\boldsymbol{\beta}}_3$, are computed and compared. If the assumption ($z^\top \widehat{\boldsymbol{\beta}}_1 < z^\top \widehat{\boldsymbol{\beta}}_2 < z^\top \widehat{\boldsymbol{\beta}}_3$) is not met, the ROC surface analysis is not performed at z.

The function clus_roc_surface() estimates a covariate-specific ROC surface at a single point for covariates and makes a 3D plot to display the estimated covariate-specific ROC surface by using **rgl** package (Duncan et al., 2021). This function also allows plotting an ellipsoidal confidence region for TCFs at a given pair of thresholds, in the ROC surface space. If the constructed confidence region is outside the unit cube, a probit transformation (Bantis et al., 2017) is automatically applied to obtain an appropriate confidence region, which is inside the unit cube.

The function clus_opt_thres3() gives the estimated covariate-specific optimal pair of thresholds as defined by the criteria GYI, CtP, and MV at multiple points of the covariates. The optimization is done by using the function optim() with the "L-BFGS-B" method; however, users can select other optimization methods, such as, "BFGS" or "Nelder-Mead". The function returns also the estimated asymptotic variance-covariance matrix of the (estimated) covariate-specific optimal pair of thresholds. Under the normality assumption, the asymptotic variance-covariance matrix is estimated by the Delta method. If the Box-Cox transformation is applied, a nonparametric bootstrap procedure for clustered data is used to estimate the asymptotic variance-covariance matrix. To speed up the bootstrap procedure, users can set the argument parallel as TRUE to enable parallel computing support. Users can also select the number of CPUs needed for the computation. After calling clus_opt_thres3(), the function plot() can be used to display confidence regions (and point estimates) of covariate-specific optimal pairs of thresholds.

The function clus_vus() estimates the covariate-specific VUS at multiple points of the covariates by using the integrate() routine. This function also performs the statistical test for the null hypothesis $H_0 : VUS = 1/6$ versus the alternative $H_A : VUS > 1/6$. This statistical test is a formal assessment of the adequacy of a diagnostic test in a three-class classification problem via its VUS at given covariates values. After calling clus_vus(), users can apply ci_clus_vus() to obtain confidence intervals for covariate-specific VUS. Three types of confidence intervals are computed: normal approximation-based; after logit and probit transformations.

The function clus_tcfs() estimates covariate-specific TCFs at a specified pair of thresholds, given one or multiple points for the covariates. This function also implements the Delta method to estimate the asymptotic variance-covariance matrix of the estimated covariate-specific TCFs. Note that, if the Box-Cox transformation is applied for the linear mixed-effects model, the pair of threshold values must be provided in the original scale.

Applications

To illustrate the use of the **ClusROC** package, we provide two examples with the MouseNeurons dataset and EnergyEthiopia, which are included in the package.

The glutamatergic neurons

To et al. (2022) used the MouseNeurons dataset to evaluate the ability of the Lysosomal Associated Membrane Protein Family Member 5 (Lamp5) gene to discriminate three types of glutamatergic neurons, namely Layer 2/3 Intratelencephalic (L2/3 IT), Layer 4 (L4) and Layer 5 Pyramidal Tract (L5 PT) neurons. A full version of the data is publicly available at http://portal.brain-map.org/atlases-and-data/rnaseq/mouse-v1-and-alm-smart-seq.

This dataset includes 860 observations (brain cells) and the following variables: the expression of the Lamp5 gene (diagnostic test/biomarker), the mouse genotype (which yields 23 clusters), the class labels (L2/3 IT, L4, and L5 PT), and the sex and age (in days) of the mouse. Below, we illustrate the use of the **ClusROC** package using this data.

Step 1: Load library and data

```
> library(ClusROC)
> data("MouseNeurons")
```

The above code loads the **ClusROC** package and the MouseNeurons data into R.

Step 2: Model fitting

In this step, we fit a linear mixed-effects model with Lamp5_cpm as a response and age_days as a covariate, under the Box-Cox transformation, by using function clus_lme():

```
> out_md <- clus_lme(fixed_formula = Lamp5_cpm ~ age_days,
+                    name_class = "subclass_label", name_clust = "genotype_id",
+                    data = MouseNeurons, boxcox = TRUE)
The ordered levels of classes are specified by the order of
 averages of the test values for each class:
L4 < L5 PT < L2/3 IT
```

As, in this case, the rank-ordered nature of the biomarker concerning the classes is not given, the monotone ordering was specified by ordering the classes according to the rank of the biomarker's sample means in the three groups. The results are shown as follows.

```
> print(out_md)

CALL: clus_lme(fixed_formula = Lamp5_cpm ~ age_days, name_class = "subclass_label",
    name_clust = "genotype_id", data = MouseNeurons, boxcox = TRUE)

Coefficients:
                                Est. Std.Error z-value p-value
subclass_labelL4             0.78770   5.95328   0.132 0.89474
subclass_labelL4:age_days    0.45039   0.18193   2.476 0.01330 *
subclass_labelL5 PT         34.30543  12.27946   2.794 0.00521 **
subclass_labelL5 PT:age_days 0.20995   0.10652   1.971 0.04872 *
subclass_labelL2/3 IT       49.34642  20.88162   2.363 0.01812 *
subclass_labelL2/3 IT:age_days 0.08991 0.08306   1.082 0.27907
sigma_c                      6.78582   4.15973      --      --
sigma_1                     15.02492   7.12012      --      --
sigma_2                     11.24066   5.70949      --      --
sigma_3                     11.14321   6.07808      --      --
lambda                       0.44565        --      --      --
ICC                          0.22848        --      --      --
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of observations: 860
Number of clusters: 23
Sample size within cluster:
     Min      Max  Average
  1.0000 330.0000  37.3913
Box-Cox transformation: TRUE
```

The diagnostic plots for the fitted model are obtained by the following command (see Figure 1):

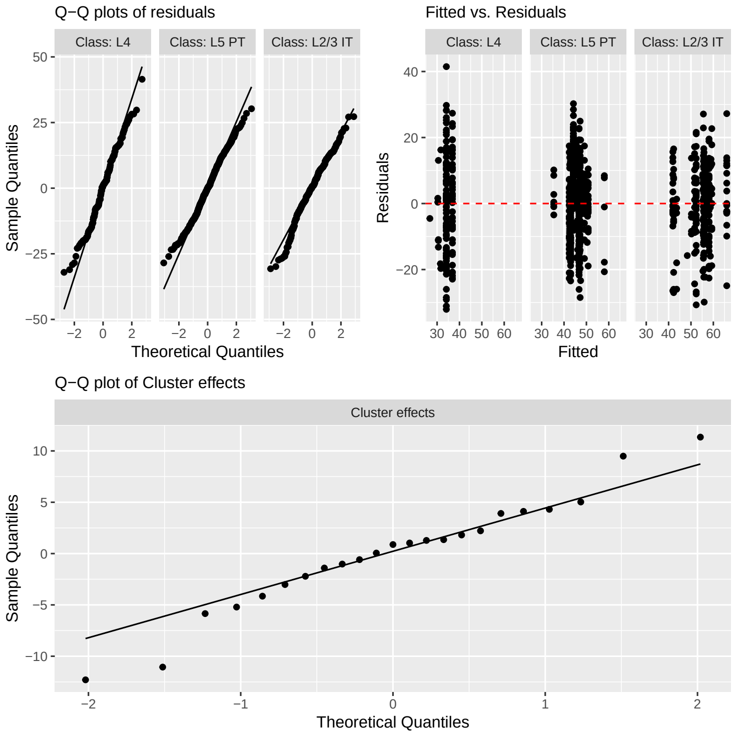


Figure 1: The diagnostic plots for the linear mixed-effects model.

```
> plot(out_md)
```

The plots suggest that the assumptions of normality and homogeneity of variances are satisfied. To help the reader evaluate the need for transforming the data, we also fitted the linear mixed-effects model without the Box-Cox transformation. The results are reported in Appendix, showing that the assumptions are violated.

Step 3: ROC surface analysis

After fitting the linear mixed-effects models of interest and verifying the normality assumption and the homogeneity of variances assumption, the ROC surface analysis can be performed. First, let us start with the estimation for covariate-specific VUS.

```
> out_vus <- clus_vus(out_clus_lme = out_md,
+                      newdata = data.frame(age_days = c(54, 60, 66)))
> print(out_vus)

CALL: clus_vus(out_clus_lme = out_md, newdata = data.frame(age_days = c(54,
    60, 66)))

Covariate-specific VUS:
 Covariates Values  Est. Std.Error z-value p-value
                54 0.541    0.0505    7.42  <0.001 ***
                60 0.514    0.0535    6.49  <0.001 ***
                66 0.485    0.0582    5.47  <0.001 ***
---
Signif. codes:   0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
z-value and p-value are for testing the null hypothesis H0: VUS = 1/6 vs HA: VUS > 1/6
```

In this case, we are interested in computing the (estimated) covariate-specific VUS at three different ages of mice, i.e., 54, 60, and 66 days. The 95% confidence intervals for covariate-specific VUS are obtained by:

```
> ci_clus_vus(out_vus)

The 95% confidence intervals for covariate-specific VUS:
   Covariates Values  Normal approximation  Logit transformation  Probit transformation
```

```
                  54        (0.442, 0.640)        (0.442, 0.637)         (0.442, 0.638)
                  60        (0.409, 0.618)        (0.410, 0.616)         (0.410, 0.617)
                  66        (0.371, 0.599)        (0.374, 0.598)         (0.373, 0.599)
```

As shown in the results, we can see three types of confidence intervals: normal approximation-based; after logit and probit transformations.

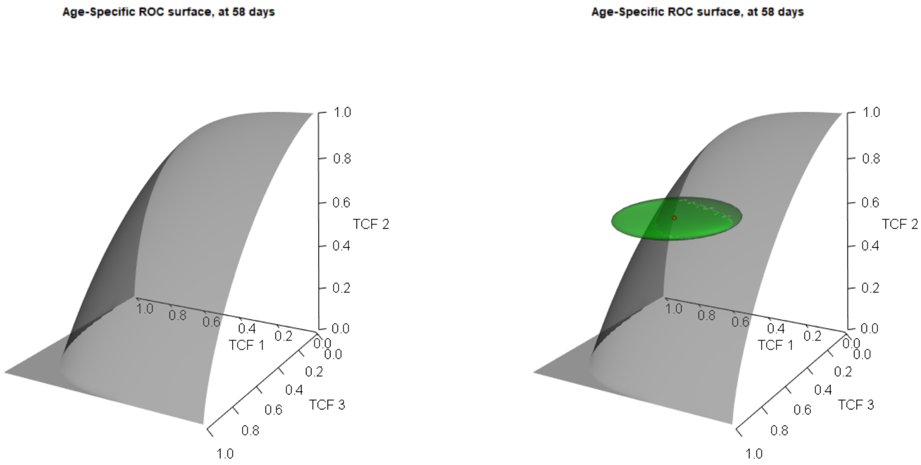
One obtains an estimate and a plot of the covariate-specific ROC surface, at, for example, the age of 58 days, as follows:

```
> clus_roc_surface(out_clus_lme = out_md, newdata = data.frame(age_days = 58),
+                  main = "Age-Specific ROC surface, at 58 days")
```

The result is displayed in Figure 2(a). A plot of a 95% ellipsoidal confidence region for TCFs at a fixed pair of threshold, for instance, $(t_1, t_2) = (350, 1350)$ is obtained as follows:

```
> clus_roc_surface(out_clus_lme = out_md, newdata = data.frame(age_days = 58),
+                  main = "Age-Specific ROC surface, at 58 days",
+                  ellips = TRUE, thresholds = c(350, 1350))
```

The 95% ellipsoidal confidence region for TCFs and covariate-specific ROC surface are displayed in 2(b), together.



(a) (b)

Figure 2: The plots of covariate-specific ROC surface at Age as 58 days: (a) without an ellipsoidal confidence region; (b) with an ellipsoidal confidence region for TCFs at $t_1 = 350$ and $t_2 = 1350$.

Estimates of covariate-specific TCFs at several values of the covariate, for a fixed pair of thresholds, are obtained by:

```
> clus_tcfs(out_clus_lme = out_md, newdata = data.frame(age_days = c(54, 58, 62)),
+           thresholds = c(350, 1350), ap_var = TRUE)

CALL: clus_tcfs(out_clus_lme = out_md, newdata = data.frame(age_days = c(54,
    58, 62)), thresholds = c(350, 1350), apVar = TRUE)

Covariate-specific TCFs at (350,1350) :
 Covariate(s) Values TCF 1 TCF 2 TCF 3 Se.TCF 1 Se.TCF 2 Se.TCF 3
                  54 0.576 0.632 0.522    0.113   0.0242   0.0630
                  58 0.533 0.620 0.533    0.119   0.0242   0.0604
                  62 0.490 0.607 0.544    0.124   0.0244   0.0592
```

The results consist of three-point estimates of covariate-specific TCFs at the fixed pair of thresholds (350, 1350), corresponding to three different values of age, 54, 58, and 62 (days), and the associated standard errors (Se). Note that, for the function `roc_surface()` and `clus_tcfs()`, if the Box-Cox transformation is applied, the pair of thresholds values must be provided in the original scale.

The following call performs the estimation of the covariate-specific optimal pair of thresholds, and the corresponding asymptotic variance-covariance matrices, at three different ages of mice: 55, 65 and 75 days. Here, we consider all criteria, i.e., GYI, CtP and MV.

```
> out_thresh <- clus_opt_thres3(method = c("GYI", "CtP", "MV"), out_clus_lme = out_md,
+                               newdata = data.frame(age_days = c(55, 65, 75)),
+                               ap_var = TRUE,
+                               control = list(n_boot = 1000, parallel = TRUE,
+                                              ncpus = 8))
```

In this case, a nonparametric bootstrap procedure for clustered data estimates the asymptotic variance-covariance matrix of the (estimated) covariate-specific optimal thresholds. Hence, we used a parallel computation with 8 CPUs (Windows 10 Pro 64-bit, Intel(R) Core(TM) i7-7700, 3.6 GHz) to speed up this process (the computation time for the parallel bootstrap process is about 15 minutes). As a general recommendation, we advise the user to switch to parallel computation not only in cases similar to the one tackled in this example but whenever the computational times appear to be too slow. The results are shown below.

```
> print(out_thresh)

CALL: clus_opt_thres3(method = c("GYI", "CtP", "MV"), out_clus_lme = out_md,
    newdata = data.frame(age_days = c(55, 65, 75)), ap_var = TRUE,
    control = list(n_boot = 1000, parallel = TRUE, ncpus = 8))

Covariate-specific optimal pair of thresholds:
 Covariate(s) Values                   Method Threshold 1 Threshold 2 TCF 1 TCF 2 TCF 3
                  55 Generalized Youden Index         530        1170 0.706 0.429 0.630
                  55    Closest to Perfection         446        1260 0.647 0.534 0.575
                  55               Max Volume         460        1260 0.658 0.525 0.575
                  65 Generalized Youden Index         627        1240 0.671 0.395 0.613
                  65    Closest to Perfection         538        1350 0.612 0.508 0.550
                  65               Max Volume         548        1350 0.619 0.502 0.550
                  75 Generalized Youden Index         731        1320 0.632 0.361 0.597
                  75    Closest to Perfection         639        1450 0.575 0.482 0.524
                  75               Max Volume         644        1450 0.578 0.480 0.523

Standard errors of Covariate-specific optimal pair of thresholds:
 Covariate(s) Values                   Method SE. Threshold 1 SE. Threshold 2
                  55 Generalized Youden Index            61.1           130.0
                  55    Closest to Perfection            73.8            85.7
                  55               Max Volume            66.7            85.8
                  65 Generalized Youden Index            98.6           120.0
                  65    Closest to Perfection           115.0            87.1
                  65               Max Volume            98.7            84.3
                  75 Generalized Youden Index           133.0           147.0
                  75    Closest to Perfection           146.0           140.0
                  75               Max Volume           126.0           121.0
```

The following call is needed to plot 95% confidence regions for the covariate-specific optimal pairs of thresholds, and the result is shown in Figure 3.

```
> plot(out_thresh, colors = c("forestgreen", "blue", "red"),
+      xlims = c(250, 1250), ylims = c(750, 1750), names.labels = "Age in days:",
+      size.point = 0.9)
```

House cooking fuel choice

We use the dataset EnergyEthiopia to illustrate the use of the package for evaluating a classifier in a three-class setting with panel data. The EnergyEthiopia data, included in the package, is a subset of the panel dataset discussed in Alem et al. (2016). The full dataset is collected in 4 cities of Ethiopia (Addis Ababa, Awassa, Dessie and Mekelle) within three different years: 2000, 2004 and 2009. Each household participated in the study at most three times (three years). For each time, the information on household energy choice and some covariates such as expenditure, demographic indicators, household size and educational status, are recorded. Alem et al. (2016) used the full dataset to investigate the determinants of household fuel cooking choice and energy transition in urban Ethiopia, based on a random-effects multinomial logistic model.

The dataset, named EnergyEthiopia, includes 2088 observations from 1123 households living in Addis Ababa and the following variables:

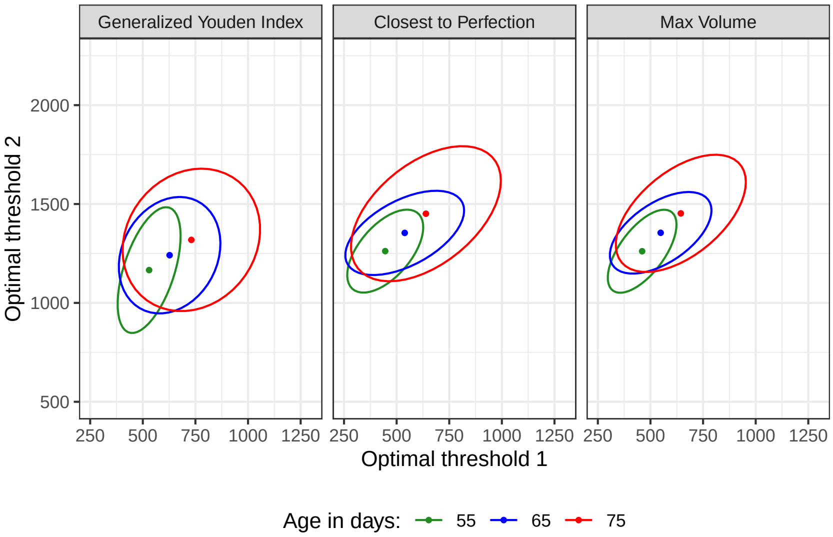


Figure 3: The 95% confidence regions for the age-specific optimal pairs of thresholds for three values of age: 55, 65, and 75.

- the cooking energy states of a household (energy2) with three categories: clean fuel only (denoted as 1), a mix of clean and biomass fuel (2) and biomass fuel only (1);
- log real consumption per adult equivalent units (lrconsaeu);
- the household size (hhs);
- the log of firewood price (lfirewood_pr);
- the log of charcoal price (lcharcol_pr);
- the log of kerosene price (lkerosene_pr);
- the log of electricity price (lelectric_pr).

We are interested in evaluating the ability of the *log real consumption per adult equivalent units* to discriminate three cooking energy states for a household, given the information provided by some covariates, such as the household size, the prices of firewood, charcoal and kerosene. Above, the term "clean fuel only" refers to clean energy sources such as electricity, gas and kerosene; whereas, "biomass fuel only" refers to energy sources such as firewood, charcoal, dung and crop residues. In our analysis, the 1123 households make up the clusters.

```
> library(ClusROC)
> data("EnergyEthiopia")
```

As the first step, we fit a linear mixed-effects model for the response variable lrconsaeu with the covariates: hhs_ft, lfirewood_pr, lcharcol_pr and lkerosene_pr. Here hhs_ft is a factor representing four levels of the household size: small ($1 \leq$ hhs ≤ 4), medium ($5 \leq$ hhs ≤ 8), large ($9 \leq$ hhs ≤ 12) and very large (hhs ≥ 13).

```
> out_md_enery <- clus_lme(
+   fixed_formula = lrconsaeu ~ hhs_ft + lfirewood_pr + lcharcol_pr + lkerosene_pr,
+   name_class = "energy2", name_clust = "uqid",
+   data = EnergyEthiopia, boxcox = FALSE
+ )
The ordered levels of classes are specified by the order of
 averages of the test values for each class:
3 < 2 < 1
```

As in the first application, we still have no information about the rank-ordered nature of the classifier (lrconsaeu) concerning the classes, so the monotone ordering was specified by ordering the classes according to the rank of the sample means of lrconsaeu inside the three groups. The results are shown as follows.

```
> print(out_md_enery)
```

```
CALL: clus_lme(fixed_formula = lrconsaeu ~ hhs_ft + lfirewood_pr +
    lcharcol_pr + lkerosene_pr, name_class = "energy2", name_clust = "uqid",
    data = EnergyEthiopia, boxcox = FALSE)

Coefficients:
                             Est. Std.Error z-value  p-value
energy23                  4.93925   0.11856  41.661  < 2e-16 ***
energy23:hhs_ftmedium    -0.48145   0.09126  -5.275 1.33e-07 ***
energy23:hhs_ftlarge     -0.77942   0.12139  -6.421 1.36e-10 ***
energy23:hhs_ftvery large -0.92139   0.14911  -6.179 6.44e-10 ***
energy23:lfirewood_pr    -0.09812   0.04806  -2.042  0.04119 *
energy23:lcharcol_pr     -0.01146   0.08398  -0.136  0.89150
energy23:lkerosene_pr    -0.13069   0.09731  -1.343  0.17927
energy22                  5.14705   0.08235  62.502  < 2e-16 ***
energy22:hhs_ftmedium    -0.39801   0.06730  -5.914 3.33e-09 ***
energy22:hhs_ftlarge     -0.60084   0.09475  -6.341 2.28e-10 ***
energy22:hhs_ftvery large -0.63877   0.18434  -3.465  0.00053 ***
energy22:lfirewood_pr     0.02983   0.04203   0.710  0.47793
energy22:lcharcol_pr     -0.10345   0.05570  -1.857  0.06329 .
energy22:lkerosene_pr    -0.24888   0.08953  -2.780  0.00544 **
energy21                  5.05581   0.05928  85.283  < 2e-16 ***
energy21:hhs_ftmedium    -0.40857   0.04612  -8.859  < 2e-16 ***
energy21:hhs_ftlarge     -0.57306   0.06944  -8.252  < 2e-16 ***
energy21:hhs_ftvery large -0.74813   0.17193  -4.351 1.35e-05 ***
energy21:lfirewood_pr     0.05490   0.02776   1.978  0.04793 *
energy21:lcharcol_pr     -0.11632   0.03957  -2.940  0.00328 **
energy21:lkerosene_pr     0.13104   0.06246   2.098  0.03590 *
sigma_c                   0.47835   0.02062      --       --
sigma_1                   0.55878   0.03409      --       --
sigma_2                   0.49135   0.02568      --       --
sigma_3                   0.57106   0.02103      --       --
ICC                       0.43933        --      --       --
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Number of observations: 2088
Number of clusters: 1123
Sample size within cluster:
     Min      Max  Average
1.000000 3.000000 1.859305
Box-Cox transformation: FALSE
```

The diagnostic plots shown in Figure 4 seem to suggest that the normality assumption and the homogeneity of variances are reliable.

```
> plot(out_md_enery)
```

We estimate the covariate-specific VUS at four different combinations of the covariates, when the value of hhs_ft changes from "small" to "very large", the values of lfirewood_pr, lcharcol_pr and lkerosene_pr are fixed as 1, -1 and 2, respectively. Here, the values "1" and "2" of lfirewood_pr and lkerosene_pr refer to the very high price of the firewood, and the kerosene, respectively, while the value "-1" of lcharcol_pr refers to the very low price of the charcoal. The results of the covariate-specific VUS estimation procedure are displayed below.

```
> out_vus_enery <- clus_vus(
+   out_clus_lme = out_md_enery,
+   newdata = data.frame(hhs_ft = c("small", "medium", "large", "very large"),
+                        lfirewood_pr = c(1, 1, 1, 1),
+                        lcharcol_pr = c(-1, -1, -1, -1),
+                        lkerosene_pr = c(2, 2, 2, 2))
+ )
> print(out_vus_enery)

CALL: clus_vus(out_clus_lme = out_md_enery, newdata = data.frame(hhs_ft = c("small",
    "medium", "large", "very large"), lfirewood_pr = c(1, 1,
    1, 1), lcharcol_pr = c(-1, -1, -1, -1), lkerosene_pr = c(2,
```

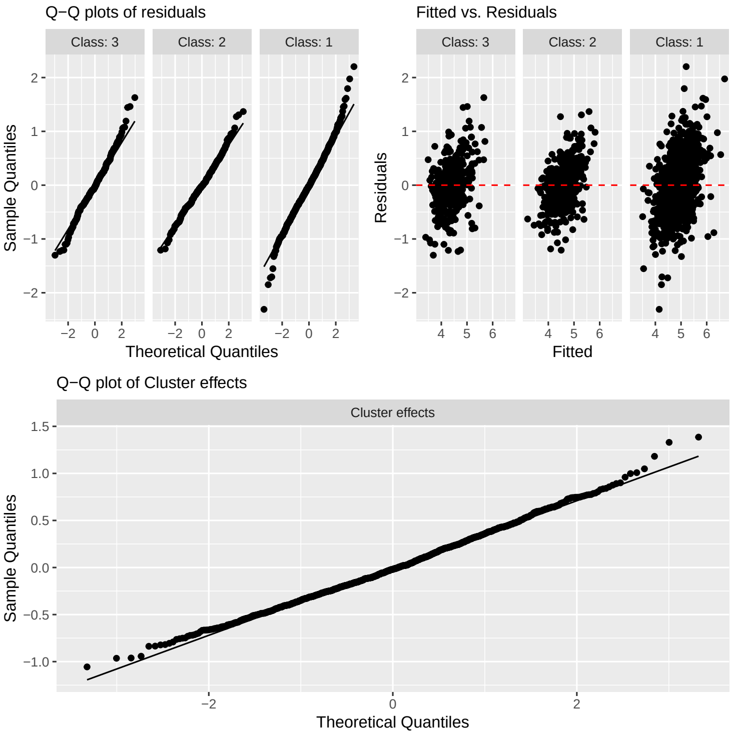


Figure 4: The diagnostic plots for the linear mixed-effects model (EnergyEthiopia data).

```
   2, 2, 2)))

Covariate-specific VUS:
      Covariates Values  Est. Std.Error z-value p-value
      (small, 1, -1, 2) 0.380    0.0674    3.17  <0.001 ***
     (medium, 1, -1, 2) 0.404    0.0639    3.72  <0.001 ***
      (large, 1, -1, 2) 0.443    0.0693    3.99  <0.001 ***
 (very large, 1, -1, 2) 0.440    0.0862    3.17  <0.001 ***
---
Signif. codes:    0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
z-value and p-value are for testing the null hypothesis H0: VUS = 1/6 vs HA: VUS > 1/6
```

The estimated covariate-specific VUSs suggest that the accuracy of `lrconsaeu` increases as the household size increases while the other covariates are fixed. The inference results for covariate-specific VUS indicate that the `lrconsaeu` can be considered as a classifier for distinguishing a household using biomass fuel only from others using either clean fuel only or a mixed fuel. However, its accuracy is not so high. We also compute the 95% confidence intervals for the covariate-specific VUS at four different points.

```
> ci_clus_vus(out_vus_enery)

The 95\% confidence intervals for covariate-specific VUS:
   Covariates Values  Normal approximation  Logit transformation  Probit transformation
    (small, 1, -1, 2)       (0.248, 0.512)       (0.259, 0.518)         (0.257, 0.517)
   (medium, 1, -1, 2)       (0.279, 0.529)       (0.287, 0.533)         (0.286, 0.532)
    (large, 1, -1, 2)       (0.307, 0.579)       (0.314, 0.580)         (0.313, 0.580)
(very large, 1, -1, 2)      (0.271, 0.609)       (0.284, 0.609)         (0.281, 0.609)
```

Since the covariate point `("large", 1, -1, 2)` gives the highest covariate-specific VUS estimate among the considered points, we plot the covariate-specific ROC surface at this point, to visualize the covariate-specific TCFs at all possible pairs of thresholds. The result is displayed in Figure 5.

```
> clus_roc_surface(
+   out_clus_lme = out_md_enery,
+   newdata = data.frame(hhs_ft = "large", lfirewood_pr = 1,
+                        lcharcol_pr = -1, lkerosene_pr = 2),
+   file_name = "ROCS_Energy_ex1.png"
+ )
```

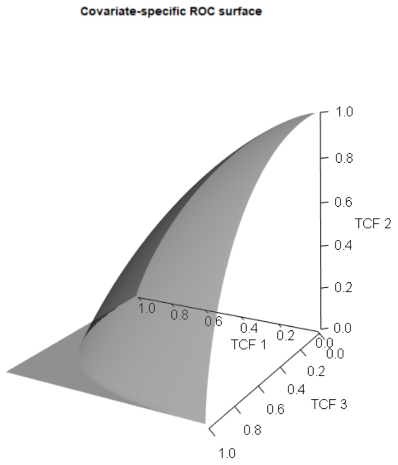


Figure 5: The plot of covariate-specific ROC surface at the covariate point ("large", 1, -1, 2).

Now, suppose we consider a pair of thresholds as $(t_1, t_2) = (3.75, 4.75)$. The results of the estimation of the covariate-specific TCFs at that pair, associated with four different covariate points, are obtained by:

```
> clus_tcfs(out_clus_lme = out_md_enery,
+           newdata = data.frame(
+             hhs_ft = c("small", "medium", "large", "very large"),
+             lfirewood_pr = c(1, 1, 1, 1), lcharcol_pr = c(-1, -1, -1, -1),
+             lkerosene_pr = c(2, 2, 2, 2)),
+           thresholds = c(3.75, 4.75), ap_var = TRUE)

CALL: clus_tcfs(out_clus_lme = out_md_enery, newdata = data.frame(hhs_ft = c("small",
    "medium", "large", "very large"), lfirewood_pr = c(1, 1,
    1, 1), lcharcol_pr = c(-1, -1, -1, -1), lkerosene_pr = c(2,
    2, 2, 2)), thresholds = c(3.75, 4.75), ap_var = TRUE)

Covariate-specific TCFs at (3.75,4.75) :
    Covariate(s) Values TCF 1 TCF 2 TCF 3 Se.TCF 1 Se.TCF 2 Se.TCF 3
      (small, 1, -1, 2) 0.126 0.415 0.839   0.0569   0.0617   0.0366
     (medium, 1, -1, 2) 0.312 0.526 0.671   0.0906   0.0247   0.0532
      (large, 1, -1, 2) 0.467 0.532 0.588   0.1100   0.0147   0.0612
 (very large, 1, -1, 2) 0.543 0.529 0.495   0.1230   0.0247   0.1080
```

Finally, we obtain the covariate-specific optimal pairs of thresholds at four different covariate points, based on all criteria, i.e., GYI, CtP and MV.

```
> out_thresh_enery <- clus_opt_thres3(
+   method = c("GYI", "CtP", "MV"), out_clus_lme = out_md_enery,
+   newdata = data.frame(hhs_ft = c("small", "medium", "large", "very large"),
+                        lfirewood_pr = c(1, 1, 1, 1),
+                        lcharcol_pr = c(-1, -1, -1, -1),
+                        lkerosene_pr = c(2, 2, 2, 2)),
+   ap_var = TRUE
+ )
> print(out_thresh_enery)

CALL: clus_opt_thres3(method = c("GYI", "CtP", "MV"), out_clus_lme = out_md_enery,
    newdata = data.frame(hhs_ft = c("small", "medium", "large",
        "very large"), lfirewood_pr = c(1, 1, 1, 1), lcharcol_pr = c(-1,
        -1, -1, -1), lkerosene_pr = c(2, 2, 2, 2)), ap_var = TRUE)

Covariate-specific optimal pair of thresholds:
    Covariate(s) Values                   Method Threshold 1 Threshold 2 TCF 1 TCF 2 TCF 3
      (small, 1, -1, 2) Generalized Youden Index        4.52        5.18 0.460 0.370 0.661
      (small, 1, -1, 2)    Closest to Perfection        4.51        5.35 0.455 0.453 0.572
      (small, 1, -1, 2)               Max Volume        4.51        5.35 0.454 0.452 0.575
     (medium, 1, -1, 2) Generalized Youden Index        4.13        4.78 0.509 0.363 0.657
     (medium, 1, -1, 2)    Closest to Perfection        4.08        4.94 0.481 0.465 0.574
     (medium, 1, -1, 2)               Max Volume        4.07        4.93 0.480 0.464 0.578
      (large, 1, -1, 2) Generalized Youden Index        3.91        4.59 0.553 0.379 0.669
      (large, 1, -1, 2)    Closest to Perfection        3.84        4.74 0.514 0.486 0.591
      (large, 1, -1, 2)               Max Volume        3.83        4.73 0.512 0.483 0.597
 (very large, 1, -1, 2) Generalized Youden Index        3.84        4.50 0.592 0.369 0.627
 (very large, 1, -1, 2)    Closest to Perfection        3.74        4.63 0.535 0.485 0.560
 (very large, 1, -1, 2)               Max Volume        3.74        4.63 0.535 0.483 0.562

Standard errors of Covariate-specific optimal pair of thresholds:
    Covariate(s) Values                   Method SE. Threshold 1 SE. Threshold 2
      (small, 1, -1, 2) Generalized Youden Index           0.234          0.1040
      (small, 1, -1, 2)    Closest to Perfection           0.112          0.0914
      (small, 1, -1, 2)               Max Volume           0.114          0.0932
     (medium, 1, -1, 2) Generalized Youden Index           0.166          0.0990
     (medium, 1, -1, 2)    Closest to Perfection           0.110          0.0868
     (medium, 1, -1, 2)               Max Volume           0.111          0.0891
      (large, 1, -1, 2) Generalized Youden Index           0.158          0.1090
      (large, 1, -1, 2)    Closest to Perfection           0.122          0.0954
      (large, 1, -1, 2)               Max Volume           0.123          0.0988
 (very large, 1, -1, 2) Generalized Youden Index           0.192          0.1640
 (very large, 1, -1, 2)    Closest to Perfection           0.154          0.1390
 (very large, 1, -1, 2)               Max Volume           0.154          0.1410
```

The 95

```
> plot(out_thresh_enery, colors = c("orange", "red", "blue", "forestgreen"),
+      file_name = "optThres_energy_ellip.pdf", nrow_legend = 2,
+      width = 6, height = 4)
```

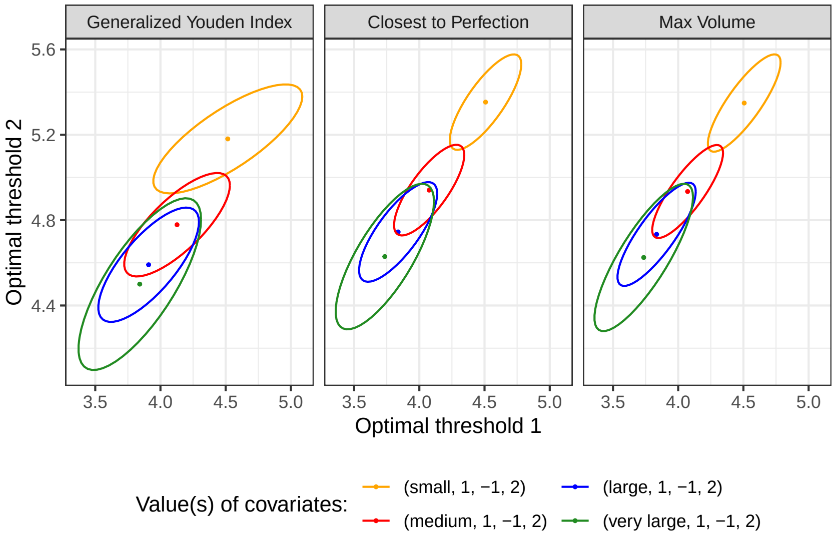


Figure 6: The 95% confidence regions for the age-specific optimal pairs of thresholds for four covariate points (in the house cooking fuel choice example).

4 Conclusion

This paper introduces the **ClusROC** package, the first R package for ROC surface analysis in three-class classification problems, for clustered data and in the presence of covariates. The package allows obtaining point estimates and confidence regions for true class fractions, ROC surface estimates and plots, point and interval estimates of VUS, and point estimates and confidence regions for optimal pair of thresholds. In the last case, three different criteria can be used: GYI, CtP, and MV. The package is available on Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=ClusROC`. The functions in the package are described and implemented using, as examples, the real datasets `MouseNeurons` and `EnergyEthiopia` provided in the package itself.

The linear mixed-effects model relies on the assumptions of normality and homoscedasticity. When such assumptions do not hold, the Box-Cox transformation can be employed. In these cases, a bootstrap procedure is needed to estimate the elliptical confidence regions for the optimal thresholds. This procedure can take a long computation time depending on the size of the data (either the number of clusters or the sample size within the clusters). For this reason, we recommend users enable the parallel computation option, which is already implemented within the function `clus_opt_thres3()`.

5 Acknowledgments

This research was supported by the Ministero dell'Istruzione, dell'Università e della Ricerca-Italy (grant number DIFO_ECCELLENZA18_01). The authors acknowledge the Editor and two anonymous reviewers whose valuable suggestions contributed to improving the presentation of the contents.

Bibliography

Y. Alem, A. D. Beyene, G. Köhlin, and A. Mekonnen. Modeling household cooking fuel choice: A panel multinomial logit approach. *Energy Economics*, 59:129–137, 2016. URL `https://doi.org/10.1016/j.eneco.2016.06.025`. [p261]

K. Attwood, L. Tian, and C. Xiong. Diagnostic thresholds with three ordinal groups. *Journal of Biopharmaceutical Statistics*, 24(3):608–633, 2014. URL `https://doi.org/10.1080/10543406.2014.888437`. [p254]

L. E. Bantis, C. T. Nakas, B. Reiser, D. Myall, and J. C. Dalrymple-Alford. Construction of joint confidence regions for the optimal true class fractions of Receiver Operating Characteristic (ROC) surfaces and manifolds. *Statistical Methods in Medical Research*, 26(3):1429–1442, 2017. URL `https://doi.org/10.1177/0962280215581694`. [p254, 257]

G. E. Box and D. R. Cox. An analysis of transformations (with discussion). *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–252, 1964. URL `https://doi.org/10.1111/j.2517-6161.1964.tb00553.x`. [p255]

M. Duncan, A. Daniel, N. Oleg, U. Simon, C. Ming, G. Albrecht, B. Ben, C. Gabor, S. Adam, S. Alexander, T. R. C. Team, E. Dirk, T. authors of Shiny, T. authors of knitr, O. Jeroen, D. Yohann, U. Joshua, i. M. Xavier, Fernandez, H. George, K. Ivan, and S. Michael. *rgl: 3D Visualization Using OpenGL*, 2021. URL `https://CRAN.R-project.org/package=rgl`. R package version 0.108.3. [p257]

M. J. Gurka and L. J. Edwards. Estimating variance components and random effects using the box-cox transformation in the linear mixed model. *Communications in Statistics—Theory and Methods*, 40(3):515–531, 2011. URL `https://doi.org/10.1080/03610920903411192`. [p255, 257]

M. J. Gurka, L. J. Edwards, K. E. Muller, and L. L. Kupper. Extending the Box–Cox transformation to the linear mixed model. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 169(2):273–288, 2006. URL `https://doi.org/10.1111/j.1467-985X.2005.00391.x`. [p255]

W. Hadley, C. Winston, H. Lionel, L. P. Thomas, T. Kohske, W. Claus, W. Kara, Y. Hiroaki, D. Dewey, and RStudio. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2022. URL `https://CRAN.R-project.org/package=ggplot2`. R package version 3.3.6. [p257]

G. Kauermann and R. J. Carroll. A note on the efficiency of sandwich covariance matrix estimation. *Journal of the American Statistical Association*, 96(456):1387–1396, 2001. URL `https://doi.org/10.1198/016214501753382309`. [p255]

K.-Y. Liang and S. L. Zeger. Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1):13–22, 1986. URL `https://doi.org/10.1093/biomet/73.1.13`. [p255]

S. R. Lipsitz, J. Ibrahim, and G. Molenberghs. Using a Box–Cox transformation in the analysis of longitudinal data with incomplete responses. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 49(3):287–296, 2000. URL `https://doi.org/10.1111/1467-9876.00192`. [p255]

L. A. Mancl and T. A. DeRouen. A covariance estimator for GEE with improved small-sample properties. *Biometrics*, 57(1):126–134, 2001. URL https://doi.org/10.1111/j.0006-341X.2001.00126.x. [p255]

C. E. McCulloch and S. R. Searle. *Generalized, linear, and mixed Models*. USA: John Wiley & Sons, 2001. ISBN 0-471-19364-X. [p254, 255]

C. T. Nakas. Developments in ROC surface analysis and assessment of diagnostic markers in three-class classification problems. *REVSTAT-Statistical Journal*, 12(1):43–65, 2014. URL https://doi.org/10.57805/revstat.v12i1.143. [p254]

C. T. Nakas and C. T. Yiannoutsos. Ordered multiple-class ROC analysis with continuous measurements. *Statistics in Medicine*, 23(22):3437–3449, 2004. URL https://doi.org/10.1002/sim.1917. [p254]

C. T. Nakas, T. A. Alonzo, and C. T. Yiannoutsos. Accuracy and cut-off point selection in three-class classification problems using a generalization of the Youden index. *Statistics in Medicine*, 29(28): 2946–2955, 2010. URL https://doi.org/10.1002/sim.4044. [p254]

S. Noll and F. Reinhard. *trinROC: Statistical Tests for Assessing Trinormal ROC Data*, 2022. URL https://CRAN.R-project.org/package=trinROC. R package version 0.6. [p254]

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 3.1-159. [p257]

P.-J. Sara, P. Natalia, and S. Konstantina. *ThresholdROC: Optimum Threshold Estimation*, 2021. URL https://CRAN.R-project.org/package=ThresholdROC. R package version 2.9.0. [p254]

D.-K. To. *bcROCsurface: Bias-Corrected Methods for Estimating the ROC Surface of Continuous Diagnostic Tests*, 2021. URL https://CRAN.R-project.org/package=bcROCsurface. R package version 1.0-5. [p254]

D.-K. To, G. Adimari, M. Chiogna, and D. Risso. Receiver operating characteristic estimation and threshold selection criteria in three-class classification problems for clustered data. *Statistical Methods in Medical Research*, 31(7):1325–1341, 2022. URL https://doi.org/10.1177/09622802221089029. [p254, 255, 256, 258]

K. To Duc, M. Chiogna, and G. Adimari. Bias–corrected methods for estimating the receiver operating characteristic surface of continuous diagnostic tests. *Electronic Journal of Statistics*, 10(2):3063–3113, 2016. URL https://doi.org/10.1214/16-EJS1202. [p254]

C. Xiong, G. van Belle, J. P. Miller, and J. C. Morris. Measuring and estimating diagnostic accuracy when there are three ordinal diagnostic groups. *Statistics in Medicine*, 25(7):1251–1273, 2006. URL https://doi.org/10.1002/sim.2433. [p254]

C. Xiong, J. Luo, L. Chen, F. Gao, J. Liu, G. Wang, R. Bateman, and J. C. Morris. Estimating diagnostic accuracy for clustered ordinal diagnostic groups in the three-class case–Application to the early diagnosis of Alzheimer disease. *Statistical Methods in Medical Research*, 27(3):701–714, 2018. URL https://doi.org/10.1177/0962280217742539. [p254, 255, 256]

6 Appendix

To help the reader evaluate the need for transforming the data, we report here the results of the analysis based on the linear mixed-effects model without the Box-Cox transformation. The results show that the assumptions of normality and homoscedasticity are violated.

```
> out_md_0 <- clus_lme(fixed_formula = Lamp5_cpm ~ age_days,
+                       name_class = "subclass_label", name_clust = "genotype_id",
+                       data = MouseNeurons)
The ordered levels of classes are specified by the order of
 averages of the test values for each class:
L4 < L5 PT < L2/3 IT
```

```
> print(out_md_0)

CALL: clus_lme(fixed_formula = Lamp5_cpm ~ age_days, name_class = "subclass_label",
    name_clust = "genotype_id", data = MouseNeurons)

Coefficients:
                                   Est. Std.Error z-value  p-value
subclass_labelL4              -378.9740  208.6613  -1.816   0.0693 .
subclass_labelL4:age_days       11.3966    1.4166   8.045 8.62e-16 ***
subclass_labelL5 PT            558.0245   89.6372   6.225 4.80e-10 ***
subclass_labelL5 PT:age_days     8.6915    0.6581  13.207  < 2e-16 ***
subclass_labelL2/3 IT         1227.4463  251.7121   4.876 1.08e-06 ***
subclass_labelL2/3 IT:age_days   5.0032    3.4950   1.432   0.1523
sigma_c                        330.1655   68.9777      --       --
sigma_1                        529.2924    9.1052      --       --
sigma_2                        514.0238   12.5033      --       --
sigma_3                        611.5288   30.7857      --       --
ICC                              0.2638        --      --       --
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Number of observations: 860
Number of clusters: 23
Sample size within cluster:
     Min      Max  Average
  1.0000 330.0000  37.3913
Box-Cox transformation: FALSE

> plot(out_md_0)
```

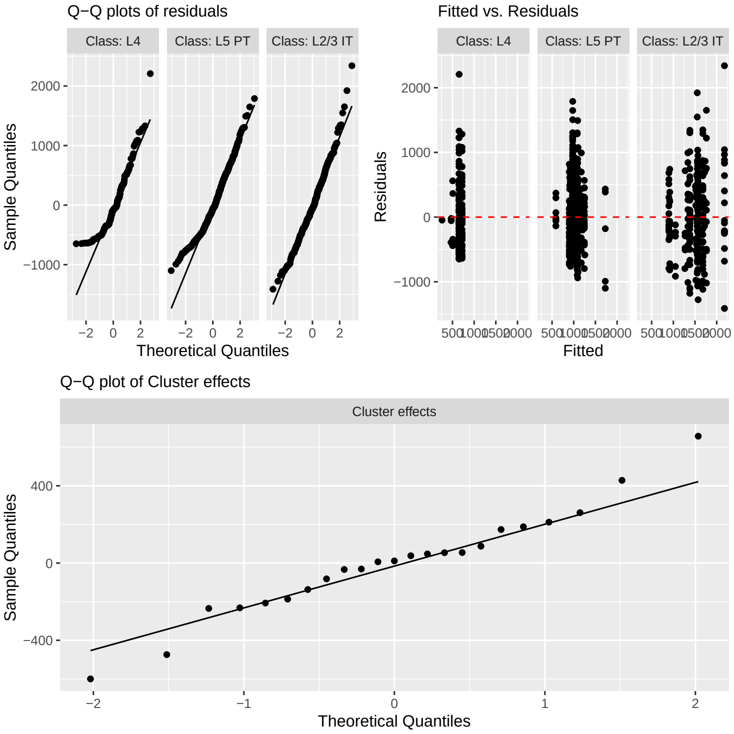


Figure 7: The diagnostic plots for the linear mixed-effects model for MouseNeurons data, without Box-Cox transformation.

Duc-Khanh To
University of Padova
Department of Statistical Sciences
Via C. Battisti, 241; I-35121 Padova, Italy

and
Department of Information and Engineering
Via Gradenigo, 6/b; 35131 Padova, Italy
ORCiD: 0000-0002-4641-0764
duckhanh.to@unipd.it

Gianfranco Adimari
University of Padova
Department of Statistical Sciences
Via C. Battisti, 241; I-35121 Padova, Italy
ORCiD: 0000-0002-7811-912X
gianfranco.adimari@unipd.it

Monica Chiogna
University of Bologna
Department of Statistical Sciences "Paolo Fortunati' '
Via Belle Arti, 41; 40126 Bologna, Italy
ORCiD: 0000-0002-7238-3739
monica.chiogna2@unibo.it

Resampling Fuzzy Numbers with Statistical Applications: FuzzyResampling Package

by Maciej Romaniuk, Przemysław Grzegorzewski

Abstract The classical bootstrap has proven its usefulness in many areas of statistical inference. However, some shortcomings of this method are also known. Therefore, various bootstrap modifications and other resampling algorithms have been introduced, especially for real-valued data. Recently, bootstrap methods have become popular in statistical reasoning based on imprecise data often modeled by fuzzy numbers. One of the challenges faced there is to create bootstrap samples of fuzzy numbers which are similar to initial fuzzy samples but different in some way at the same time. These methods are implemented in **FuzzyResampling** package and applied in different statistical functions like single-sample or two-sample tests for the mean. Besides describing the aforementioned functions, some examples of their applications as well as numerical comparisons of the classical bootstrap with the new resampling algorithms are provided in this contribution.

1 Introduction

The bootstrap introduced by Efron (1979) is one of the most important contemporary achievements of computational statistics. It has proven its usefulness in various fields, e.g., in estimating standard errors, computing confidence intervals, or hypothesis testing, especially if the actual population distribution is unknown or the desired statistical procedures are not tractable analytically, and when the sample size is not large enough to apply asymptotic methods (e.g, Efron and Tibshirani (1993)). Then Giné and Zinn (1990) developed a bootstrapped approximation to the Central Limit Theorem for generalized random elements, which allows to apply bootstrap for fuzzy random variables used for modeling imprecise outputs of random experiments. This application is of extreme importance because the absence of suitable models for the distribution of fuzzy random variables makes the statistical reasoning difficult. Practice has shown that the use of the bootstrap to handle fuzzy data has been very effective (e.g., Gil et al. (2006); González-Rodríguez et al. (2006); Montenegro et al. (2004); Lubiano et al. (2016, 2017)).

An important disadvantage of the classical bootstrap, easily spotted especially for small sample sizes, is that the bootstrap samples almost always contain repeated values and hence their variability is lower when compared to the initial sample. This is more troublesome when the random distribution of the assumed statistical model is continuous, since this feature is not properly reflected with the classical bootstrap where each value has the same non-zero probability of appearing in the bootstrap sample. To overcome this problem many modifications of Efron's approach were proposed in the literature, like the smoothed bootstrap (Silverman and Young, 1987). The same shortcomings and remarks apply to bootstrap methods applied in a fuzzy environment. Here it would also be desirable to generate samples consisting of fuzzy observations that are almost identical to those contained in the original sample, but nevertheless different from them to some extent. Several methods have been proposed to deal with this problem (Grzegorzewski and Romaniuk, 2022b). One can divide them into two main groups. The first one group shares the generation of new fuzzy numbers which keep some "characteristic points" of the membership functions describing initial fuzzy values. For the second group, we are interested in the preservation of some basic characteristics of the initial values, like their canonical representation (Delgado et al., 1998). Both approaches seem to be fruitful in many aspects of the statistical inference (see Grzegorzewski et al. (2019, 2020b,a); Grzegorzewski and Romaniuk (2022b); Romaniuk (2019); Romaniuk and Hryniewicz (2019)) and may be helpful in improving the results of some real-life data analyses.

To allow potential users access to the aforementioned new bootstrap methods, several resampling algorithms have been implemented in **FuzzyResampling** package (Romaniuk et al., 2022). They have been complemented by some additional procedures which might be useful for the generation of samples of trapezoidal fuzzy numbers, for estimation of the standard error (or the MSE), and for testing hypotheses about the fuzzy mean (in the single or two samples problem), etc. The proposed package can be applied together with other tools designed for data analysis. **FuzzyResampling** is, to our knowledge, the first package in R (R Core Team, 2023) which provides a few resampling methods for trapezoidal fuzzy numbers together with some statistical functions based on them.

In the following, we briefly compare **FuzzyResampling** with the existing packages, introduce a necessary notation and describe the new resampling methods. Functions implemented in the package are illustrated with examples. Finally, the proposed resampling methods are compared with the classical bootstrap in some important statistical problems (both theoretical and utilizing real-life data).

Related packages

There is an abundance of packages oriented toward the classic bootstrap and its statistical applications, like **boot** (Canty and Ripley, 2022), with many functions and data originated from Davison and Hinkley (1997) including the parametric and nonparametric resampling approaches, or **bootstrap** (Tibshirani and Leisch, 2019), which contains functions and data related to Efron and Tibshirani (1993), e.g. cross-validation or jackknife procedures.

There are also many R packages intended to handle fuzzy numbers that may be useful in statistical reasoning with fuzzy data. Researchers can utilize **FuzzyNumbers** (Gagolewski and Caha, 2021) for computing arithmetic operators, designing approximations, and finding possibility and necessity values for arbitrary fuzzy numbers or some of their special types. **FuzzySTs** (Berkachy and Donze, 2020) might be helpful for fuzzification, calculation of different distances, numerical estimations of fuzzy statistical measures and bootstrap distribution of the likelihood ratio, etc. Such packages as **SAFD** (Trutschnig et al., 2013) or **Sim.PLFN** (Parchami, 2017) are devoted to simulation on fuzzy numbers (see also Coroianu et al. (2013) for additional details).

However, to our knowledge, **FuzzyResampling** is the only package that provides the new resampling methods, different from Efron's classical approach, for trapezoidal fuzzy numbers and lets users apply them to various problems of statistical inference. In particular, **SAFD** provides procedures for the bootstrapped versions of the one- or two-sample tests for the mean of trapezoidal fuzzy numbers (which are related to the C-test mentioned in this paper), but they are limited to the classical bootstrap. Although it contains a function that generates stochastically perturbated new values based on the input data frame, it only adds some random noise without keeping track of important characteristics of the input fuzzy numbers (as is done in the resampling methods implemented in **FuzzyResampling**).

Fuzzy numbers – notation

For a more detailed introduction to the theory of fuzzy numbers, we refer the reader to Ban et al. (2015). In the following, we provide only some necessary concepts and notation.

A fuzzy number A is characterized by its membership function $A(x)$ which represents the degree of membership of x into A. We assume that $0 \leqslant A(x) \leqslant 1$, where $A(x) = 1$ indicates that x surely belongs to A, while $A(x) = 0$ means that x surely does not belong to A. If $A(x) \in (0,1)$ then x partially belongs to A.

Perhaps the most often used fuzzy numbers are known as the **trapezoidal fuzzy numbers** (TPFNs) with their membership functions given by

$$A(x) = \begin{cases} \frac{x-a_1}{a_2-a_1} & \text{if } a_1 < x \leqslant a_2, \\ 1 & \text{if } a_2 \leqslant x \leqslant a_3, \\ \frac{a_4-x}{a_4-a_3} & \text{if } a_3 \leqslant x < a_4, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $a_1, a_2, a_3, a_4 \in \mathbb{R}$, and $a_1 \leqslant a_2 \leqslant a_3 \leqslant a_4$. Here, the interval $[a_2, a_3]$, called the core, contains all values which are fully compatible with the concept described by the fuzzy number A, while the interval $[a_1, a_4]$, called the support, indicates all real values that are compatible to some extent with A. If A is a TPFN it can be described completely using only four real values a_1, a_2, a_3, a_4, so we can state $A = (a_1, a_2, a_3, a_4)$. If $a_2 = a_3$ then we have a **triangular fuzzy number** (TRFN), and $A = (a_1, a_2, a_4)$.

Another parametrization of a trapezoidal fuzzy number, which is especially useful for generating fuzzy data and simulation studies, is given by

$$c = \frac{a_2 + a_3}{2},\ s = \frac{a_3 - a_2}{2},\ l = a_2 - a_1,\ r = a_4 - a_3, \tag{2}$$

where c indicates the center of the core of a fuzzy number, s is equal to its core radius (the spread of the core), and l and r stand for the spread of the left and the right arm of the membership function $A(x)$, respectively. Of course, we have $c \in \mathbb{R}$, and $s, l, r \geqslant 0$.

To introduce basic arithmetic operations (such as addition, subtraction, etc.) for the fuzzy numbers, the Zadeh extension principle should be applied (see Ban et al. (2015)). However, in the case of TPFNs, some operations are easier to conduct, e.g., for $A = (a_1, a_2, a_3, a_4)$ and $B = (b_1, b_2, b_3, b_4)$, we have

$$A + B = (a_1 + b_1, a_2 + b_2, a_3 + b_3, a_4 + b_4), \tag{3}$$

so the result is also a TPFN.

Resampling approaches for fuzzy data

Let $\widetilde{\mathbb{X}} = (\tilde{X}_1, \ldots, \tilde{X}_n)$ be a fuzzy random sample (Puri and Ralescu, 1986) and $\widetilde{\mathbb{x}} = (\tilde{x}_1, \ldots, \tilde{x}_n)$ – its actual realization. This sample will be called the **primary sample** or the **initial sample**. For the classical Effron's bootstrap, the bootstrap samples (or **secondary samples**) are drawn randomly with replacement from the primary sample. In this way we can create b bootstrap samples and the i-th element of the j-th secondary sample is denoted by $\tilde{x}^*_{ij}$.

Because of the previously mentioned shortcomings of Efron's approach, several modifications were proposed in the case of real-valued data. New resampling methods for fuzzy data were introduced recently in Grzegorzewski et al. (2019, 2020b,a); Romaniuk (2019); Romaniuk and Hryniewicz (2019). All of them are implemented in **FuzzyResampling** package. We summarize them briefly below. For a more detailed review, we refer the reader to Grzegorzewski and Romaniuk (2022b).

The proposed approaches can be divided into two groups. In the first one, following (2), each TPFN $\tilde{x}_i$ is described by its "characteristic points", like the left bound of the core $lc_i = a_2 = c_i - s_i$, the core width $wc_i = a_3 - a_2 = 2s_i$, and the spreads l_i and r_i. Then, given a primary sample $\widetilde{\mathbb{x}}$, four sets are created: $LC = \{lc_1, \ldots, lc_n,\}$, $WC = \{wc_1, \ldots, wc_n,\}$, $L = \{l_1, \ldots, l_n,\}$ and $R = \{r_1, \ldots, r_n,\}$. If the **d-method** (Romaniuk and Hryniewicz, 2019) is applied then four numbers lc^*, wc^*, l^*, r^* from the respective sets LC, WC, L, R are drawn to construct a new bootstrapped element $\tilde{x}^*$. This procedure is repeated n times to obtain a whole secondary sample. For the **w-method** (Romaniuk and Hryniewicz, 2019), instead of the equal probabilities on the discrete sets, the special density $w(x)$ is used to construct x^*_i. This leads to a more diversified secondary sample than for the d-method. Both methods were also applied in Romaniuk and Hryniewicz (2021) for interval-valued fuzzy numbers.

The second group of methods contains the following flexible bootstrap algorithms: **VA-method** (Grzegorzewski et al., 2019, 2020a), **EW-method** (Grzegorzewski et al., 2020a), **VAF-method** (Grzegorzewski et al., 2020b) and **VAA-method** (Grzegorzewski and Romaniuk, 2022b). Contrary to **d-method** or **w-method**, flexible approaches can be applied to primary samples of any fuzzy numbers, but the generated outputs consist of TPFNs. However, the generated secondary samples preserve the so-called canonical representations of the primary observations. Our flexible methods differ in the type of preserved representation, which is summarized in Table 1.

Each of the flexible methods consists of four steps. Firstly, depending on the particular method, some characteristics of fuzzy numbers – like the value $\mathrm{Val}(\tilde{x}_i)$, ambiguity $\mathrm{Amb}(\tilde{x}_i)$ (Delgado et al., 1998), expected value $\mathrm{EV}(\tilde{x}_i)$ (i.e. the middle point of the expected interval of a fuzzy number (Dubois and Prade, 1987; Heilpern, 1992)), width $\mathrm{w}(\tilde{x}_i)$ (Chanas, 2001), fuzziness $\mathrm{Fuzz}(\tilde{x}_i)$ (Delgado et al., 1998), left- and right-hand ambiguities $\mathrm{Amb}_L(\tilde{x}_i), \mathrm{Amb}_U(\tilde{x}_i)$ – are calculated for each observation $\tilde{x}_i$ of the primary sample. What values are determined depends on the user's decision on which values should be preserved during the generation of the bootstrap samples. While some characteristics will be kept, others will be randomly generated from the uniform distribution. Finally, the remaining values describing a TPFN are directly calculated based on the formulas depending on the resampling approach. In Table 1 we list some flexible bootstrap methods along with an indication of which parameters are preserved, which are randomly generated, and which are directly calculated.

Method	Preserved	Randomly generated	Directly calculated
VA-method	$\mathrm{Val}, \mathrm{Amb}$	s, c	l, r
EW-method	EV, w	s, c	l, r
VAF-method	$\mathrm{Val}, \mathrm{Amb}, \mathrm{Fuzz}$	c	s, l, r
VAA-method	$\mathrm{Val}, \mathrm{Amb}_L, \mathrm{Amb}_U$	s	c, l, r

Table 1: Brief characteristic of the flexible bootstrap methods.

2 Overview of FuzzyResampling package

Below we briefly discuss functions implemented in **FuzzyResampling** package (version 0.6.0, available via CRAN). All further examples in R can be self-repeated using a supplementary file.

Generation of the initial sample

When the bootstrap is applied to solve any real-life problem the provided data is automatically treated as the initial sample. However, to perform any simulation study many synthetic samples are necessary, so effective data generators from various distributions are then strongly desirable. In the case of reasoning with imprecise data, we need an algorithm which generates fuzzy numbers used for modeling such data. In **FuzzyResampling** package two specialized functions to generate randomly trapezoidal fuzzy numbers are available:

```
GeneratorNU (n, mu, sigma, a, b, increases = FALSE, ...)
GeneratorNExpUU (n, mu, sigma, lambda, b, c, increases = FALSE, ...)
```

The third function, `GeneratorFuzzyNumbers`, can be used for various random distributions describing the "true origin" of TPFN, the increases of its core and the support. In this case, the respective functions (e.g., `rnorm`) from **stats** package (R Core Team, 2023), together with their parameters' names, can be directly applied.

In both cases, to generate a TPFN, the parametrization described by (2) is used. In particular, to obtain a single TPFN four values c, s, l, r are generated independently using random distributions described in Table 2, where N(`mu`,`sigma`) stands for the normal distribution with the mean `mu` and standard deviation `sigma`, U (0,a) denotes the uniform distribution on the interval (0, a), Exp (`lambda`) stands for the exponential distribution with the parameter `lambda`, etc. More details concerning various types of simulated fuzzy numbers can be found in, e.g., Grzegorzewski et al. (2020b); Grzegorzewski and Romaniuk (2022a).

Function	c	s	l	r
GeneratorNU	N(mu,sigma)	U (0,a)	U(0,b)	U(0,b)
GeneratorNExpUU	N(mu,sigma)	Exp (lambda)	U(0,b)	U(0,c)

Table 2: Distributions used to simulate fuzzy numbers.

If we generate a sample of `n` fuzzy numbers using any of the considered functions we receive the output as a matrix with `n` rows and four columns corresponding to values a_1, a_2, a_3, a_4 defined in (1) (abbreviated further as "the ends") if the parameter `increases` is set to `FALSE`, otherwise producing values $l, c - s, c + s, r$ described in (2) (abbreviated further as "the increments"). It is worth noting that the parameter `increases` plays the same role of indicating the form of the output (or input) fuzzy numbers in all functions in this package.

As an example consider the following code generating a sample of $n = 10$ trapezoidal fuzzy numbers in the "ends" mode, i.e. a_1, a_2, a_3, a_4:

```
# seed PRNG
> set.seed(1234)

> sample1 <- GeneratorNU(10, mu = 0, sigma = 1, a = 0.2, b = 0.6)
> head(sample1,2)
           [,1]       [,2]       [,3]       [,4]
[1,] -1.6023884 -1.2703882 -1.1158475 -1.0715795
[2,] -0.1709531  0.2168906  0.3304666  0.5162785
```

Resampling methods

To perform the classical Efron's bootstrap, the `ClassicalBootstrap` function should be applied. Functions related to other methods (VA, EW, etc.) have the common form `XXXMethod`, where `XXX` stands for the respective method abbreviation. For instance, we write `EWMethod` in the case of the EW approach. A general structure of these functions is rather intuitive:

```
XXXMethod(initialSample, b = n, increases = FALSE)
```

where `initialSample` denotes the name of a matrix of input fuzzy numbers, `b` indicates the desired size of the bootstrap sample (if `b` is not specified we assume it is equal to the initial sample n, i.e. the number of rows in the input matrix), while with `increases` we set the mode ("ends" or "increments") used for describing fuzzy numbers both in the input and output matrices.

To generate the bootstrapped sample of $b = n = 10$ values using the classical Efron's approach based on the previously created initial sample `sample1`:

```
> set.seed(1234)
>
> sampleEfron <- ClassicalBootstrap(sample1)
>
> head(sampleEfron,2)
           [,1]       [,2]       [,3]       [,4]
[1,] -1.3584678 -0.8991919 -0.7285674 -0.2195319
[2,]  0.0427377  0.3439362  0.6579900  0.9603501
```

Similarly, to perform the VA method for $b = 20$:

```
> set.seed(1234)
>
> sampleVA <- VAMethod(sample1, 20)
>
> head(sampleVA,2)
             [,1]       [,2]       [,3]        [,4]
[1,] -1.091883230 -1.0324841 -0.8074509 -0.06176483
[2,]  0.009038746  0.3607857  0.4974204  1.28148926
```

Calculation of the mid/spread distance

A distance between two fuzzy numbers often used in the statistical context is the so-called mid/spread distance D_θ^λ proposed in Bertoluzza et al. (1995); Trutschnig et al. (2009). It can be designated by:

```
BertoluzzaDistance(fuzzyNumber1, fuzzyNumber2, theta = 1/3, increases = FALSE)
```

where `fuzzyNumber1` and `fuzzyNumber2` denote vectors containing the respective fuzzy numbers, `theta` is the weight of the impact of the distance between the mid-points of α-cuts in contrast to the distance between their spreads (usually `theta=1/3` or `theta=1`).

To find the mid/spread distance between two fuzzy numbers from the sample `sample1` we apply:

```
> BertoluzzaDistance(sample1[1,],sample1[2,])
[1] 1.488457
```

Estimation of SE/MSE

Bootstrap is often applied to estimate the standard error (SE) or the mean squared error (MSE) of the considered estimator. If the resulting estimator of the mean is given by a trapezoidal fuzzy number both SE and MSE can be obtained with the following function:

```
SEResamplingMean(initialSample, resamplingMethod = "ClassicalBootstrap",
  repetitions = 100, trueMean = NA, theta = 1/3, increases = FALSE)
```

where `initialSample` indicates a matrix containing the initial sample, `resamplingMethod` denotes a resampling method, and `repetitions` stands for the number of the bootstrap samples. If `trueMean` is not set then SE is estimated by the formula

$$\widehat{\text{SE}} = \sqrt{\frac{1}{\text{repetitions} - 1} \sum_{k=1}^{\text{repetitions}} D_\theta^2\left(\bar{X}^{*i}, \bar{X}^*\right)}, \tag{4}$$

where D_θ is the mid/spread distance, while

$$\bar{X}^{*i} = \frac{1}{n} \sum_{j=1}^{n} X^{*ij}, \qquad \bar{X}^* = \frac{1}{\text{repetitions}} \sum_{i=1}^{\text{repetitions}} \bar{X}^{*i}, \tag{5}$$

and where X^{*ij} is the j-th fuzzy value in the i-th bootstrap sample. Otherwise, MSE is estimated as follows

$$\widehat{\text{MSE}} = \frac{1}{\text{repetitions} - 1} \sum_{k=1}^{\text{repetitions}} D_\theta^2\left(\bar{X}^{*i}, \mathbb{E}X\right), \tag{6}$$

where $\mathbb{E}X$ is the Aumann-type mean (Puri and Ralescu, 1986) of the fuzzy number given by `trueMean` expressed by the four values (depending on the mode “ends” or “increments” specified in `increases` parameter).

To estimate SE using the classical bootstrap and the previously generated `sample1` we apply:

```
> set.seed(1234)
> SEResamplingMean(sample1)
$mean
[1] -0.70650791 -0.40296248 -0.22308095  0.05157294

$SE
[1] 0.05715056
```

Here, mean is equal to $\bar{X}^*$ while SE gives $\widehat{\mathrm{SE}}$.

On the other hand, the MSE based on the VA method can be estimated as follows:

```
> set.seed(1234)
>
> SEResamplingMean(sample1,resamplingMethod = "VAMethod",trueMean = c(-0.4,-0.1,0.1,0.4))
$mean
[1] -0.4 -0.1  0.1  0.4

$SE
[1] 0.04421334
```

where SE actually gives $\widehat{\mathrm{MSE}}$.

Bootstrapped one-sample test for the mean

Bootstrap methods are often used in hypothesis testing with fuzzy data. One of such tests, denoted further as the one-sample C-test (Colubi, 2009; Lubiano et al., 2016), is designed to verify

$$H_0 : \mathbb{E}X = \mu_0 \quad \text{vs.} \quad H_1 : \mathbb{E}X \neq \mu_0, \tag{7}$$

where μ_0 is a fixed fuzzy number assumed as the true fuzzy mean. This test can be invoked with the following function:

```
OneSampleCTest(initialSample,mu_0,numberOfSamples = 100,theta = 1/3,
  resamplingMethod = "ClassicalBootstrap",increases = FALSE)
```

where initialSample is a matrix describing the initial sample, mu_0 is the hypothetical mean μ_0, numberOfSamples is the number of bootstrap samples and resamplingMethod specifies the resampling method. As the output, we receive the estimated *p-value* which can be used to make a final decision (i.e. to reject or not reject the null hypothesis H_0).

To check whether the true mean of the population represented by sample1 is a TPFN mu_0 parametrized by $(-0.4, -0.1, 0.1, 0.4)$ we can apply the C-test based on the classical bootstrap as follows:

```
> set.seed(1234)
> OneSampleCTest(sample1, mu_0 = c(-0.4,-0.1,0.1,0.4))
[1] 0.31
```

To perform this test based on the VA method we apply:

```
> set.seed(1234)
>
> OneSampleCTest(sample1, mu_0 = c(-0.4,-0.1,0.1,0.4),
+                numberOfSamples = 1000, resamplingMethod = "VAMethod")
[1] 0.253
```

In both cases, there is no reason to reject the null hypothesis because the estimated p-values are greater than any reasonable significance level (like the traditional 0.05).

Bootstrapped two-sample test for the mean

A similar procedure can be applied to check whether there is a significant difference between the means of the two independent fuzzy samples. More precisely, we consider the two-sample C-test (Lubiano et al., 2016) to verify

$$H_0 : \mathbb{E}X_1 = \mathbb{E}X_2 \quad \text{vs.} \quad H_1 : \mathbb{E}X_1 \neq \mathbb{E}X_2, \tag{8}$$

where $\mathbb{E}X_1$ and $\mathbb{E}X_2$ are the Aumann-type means of the first and second sample, respectively. Here the desired p-value can be calculated by the following function

```
TwoSampleCTest(initialSample1,initialSample2,numberOfSamples = 100,
 theta = 1/3,resamplingMethod = "ClassicalBootstrap",increases = FALSE)
```

where initialSample1 and initialSample2 are matrices corresponding to the first and the second initial sample, respectively, while other parameters remain identical as for OneSampleCTest.

Given previously generated sample1 and sample2 simulated with a slight shift (i.e. for mu = 0.2) we verify (8) using the two-sample C-test based on the classical bootstrap. Thus we type the following commands:

```
> set.seed(1234)
> sample2 <- GeneratorNU(10, mu = 0.2, sigma = 1, a = 0.2, b = 0.6)
>
> TwoSampleCTest(sample1, sample2,numberOfSamples = 1000)
[1] 0.678
```

Thus obtaining such a high p-value there is no reason to reject H_0.

3 Comparison of the resampling methods with FuzzyResampling package

Now we numerically compare the introduced resampling methods with the classical bootstrap using various statistical approaches like the standard error estimation or the power analysis.

Method comparison with SE/MSE

To compare the statistical properties of the considered resampling methods we firstly use SE/MSE for the mean. In some papers, it is stated that the proposed new resampling methods usually have smaller errors than the classical bootstrap (Grzegorzewski et al., 2020a,b; Grzegorzewski and Romaniuk, 2022b). To check this, one may use the following SEResamplingMean function

```
ComparisonSEMean (generator, sampleSize = 10, numberOfSamples = 100,
  repetitions = 100,trueMean = NA, theta = 1/3, ...)
```

where generator indicates the algorithm applied to simulate fuzzy numbers (see Tab. 2), ... specifies some additional parameters required by generator and numberOfSamples stands for the number of experiment repetitions performed to minimize the undesired influence of random fluctuations.

To estimate SE for a small (e.g. $n = 10$) or moderate (e.g. $n = 50$) initial sample size we proceed as follows:

```
> set.seed(1234567)

> outputSEsample3 <- ComparisonSEMean(generator = "GeneratorNExpUU",sampleSize = 10,
+ numberOfSamples = 10000,repetitions = 100,mu = 0,sigma = 4,lambda = 4,b = 0.4,c = 0.8)

> round(outputSEsample3, digits = 5)

ClassicalBootstrap  VAMethod  EWMethod  VAFMethod  DMethod  WMethod VAAMethod
           0.10756   0.10757   0.10753    0.10751  0.10763  0.10664   0.10764
```

As it can be seen the estimated SE for the new resampling methods may be significantly lower (about 1%) than for the classical bootstrap. These results confirm the remarks given in Grzegorzewski and Romaniuk (2022b) that "the w-method is usually the best approach for small sample sizes, while the VAF-method is the best for moderate and big samples. In general, when considering various models, it appears that the VAF-method and EW-method behave in a rather stable manner, while the behavior of the VAA-method depends strongly on the type of data." Some other examples can be found in the supplemental file.

A similar comparison of the bootstrap methods can be done using the MSE:

```
> set.seed(1234567)

> outputMSEsample2 <- ComparisonSEMean(generator = "GeneratorNExpUU",sampleSize = 10,
+ numberOfSamples = 10000, repetitions = 100, trueMean = c(-0.45,-0.25,0.25,0.65),
+ mu = 0, sigma = 4, lambda = 4, b = 0.4, c = 0.8)
```

```
> round(outputMSEsample1, digits = 5)

ClassicalBootstrap  VAMethod  EWMethod  VAFMethod  DMethod  WMethod  VAAMethod
           0.16495   0.16499   0.16482    0.16470  0.16496   0.16315  0.16519
```

Here the relative difference is about 1.1% if the w-method is compared with the classical bootstrap.

Test size estimation

We can also compare the empirical size (i.e. the maximal percentage of rejections if the null hypothesis holds) of the one-sample C-test based on different resampling approaches (see also Grzegorzewski et al. (2020b)). To proceed with such a study, the following special function based on `OneSampleCTest` is available:

```
ComparisonOneSampleCTest (generator,mu_0,shift=0,sampleSize = 10,numberOfSamples = 10,
 initialSamples = 100,theta = 1/3,significance = 0.05, ...)
```

It generates a sequence of initial samples (their number is given in `initialSamples` and the size is determined by `sampleSize`) for fuzzy numbers of the type specified by `generator`. Then some deterministic shift of the size `shift` is added to each fuzzy observation in these samples. Next, the function `OneSampleCTest` is executed to calculate the p-value for each combination of the initial sample and resampling method. Then, by comparing the p-value with the assumed significance level `significance` we make a decision whether to reject the null hypothesis H_0 specified in (7) or not. The output of this procedure is the percentage of rejections in the sequence of experiments. To estimate the test size one should set `shift=0`. All other parameters of this function are passed to `OneSampleCTest` or the respective `generator`.

Let us consider the following simulation experiment:

```
> set.seed(1234567)

> outputCSizetest1 <- ComparisonOneSampleCTest(generator="GeneratorNU",
+ mu_0 = c(-0.4,-0.1,0.1,0.4), sampleSize = 10,numberOfSamples = 100,
+ initialSamples = 1000,mu = 0, sigma = 1,a = 0.2,b = 0.6)

> round(outputCSizetest1, digits = 4)

ClassicalBootstrap VAMethod EWMethod VAFMethod DMethod WMethod  VAAMethod
             0.038    0.035    0.040     0.035   0.037   0.040   0.034

# check the distance to the "true value" 0.05
> round(outputCSizetest1-0.05, digits = 4)

ClassicalBootstrap VAMethod EWMethod VAFMethod DMethod WMethod  VAAMethod
            -0.012   -0.015   -0.010    -0.015  -0.013  -0.010   -0.016
```

It can be seen that the empirical sizes of the C-test combined with the new resampling methods might be closer to the assumed significance level of 0.05 than obtained for Efron's bootstrap. Some other examples can be found in the supplemental file. The plot showing the estimated percentage of rejections as a function of the significance level is shown in Fig. 1. Because the distance between these two values (i.e. the percentage of rejections and the significance level) is usually low, the respective differences are additionally plotted in Fig. 2 for all the resampling methods.

Our experiments confirm the conclusion stated in (Grzegorzewski et al., 2020a) that the classical bootstrap was never the winner in such comparisons. New resampling methods usually behave much better – especially the d-method and the VA-method. The EW-method behaves always in a relatively stable manner and never drops below the third place.

Test power analysis

Our package also enables the power comparison for the one-sample C-test combined with different resampling methods. This is possible by using the `ComparisonOneSampleCTest` function, which determines the percentage of the null hypothesis rejection under increasing shift added to the initial sample:

```
ComparePowerOneSampleCTest (generator,mu_0,shiftVector,sampleSize = 10,
 numberOfSamples = 10,initialSamples = 100,theta = 1/3,significance = 0.05, ...)
```

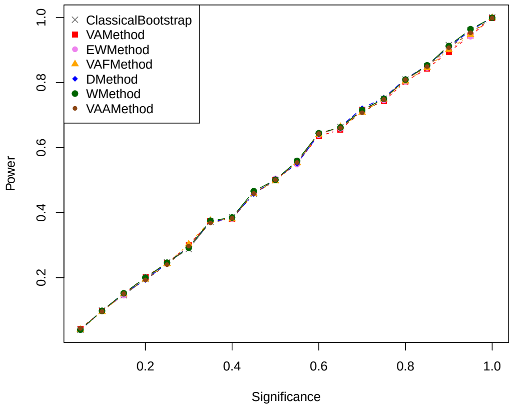


Figure 1: Test size depending on the significance level for the one-sample C-test based on the samples generated with `GeneratorNU`.

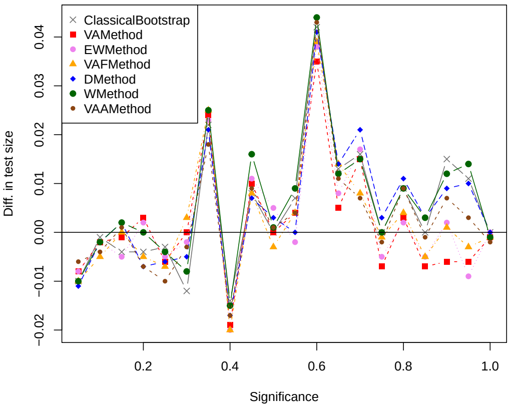


Figure 2: Differences in test size between the resampling methods and the significance level.

This procedure produces a matrix with the successive shift values, set in the vector `shiftVector`, specified in the rows and the corresponding percentages of rejections for the respective resampling methods given in columns.

Below we show how to apply this function to comparisons based on very small samples ($n = 5$). To minimize undesired random effects the following experiment was repeated 10000 times:

```
# prepare vector of shifts
> shifts <- seq(0.1,1,by = 0.05)

> set.seed(1234567)

> outputCPowerTest1 <- ComparePowerOneSampleCTest(generator="GeneratorNU",
+ mu_0 = c(-0.4,-0.1,0.1,0.4),shiftVector = shifts,sampleSize = 5,numberOfSamples = 100,
+ initialSamples = 10000,mu = 0,sigma = 1,a = 0.2,b = 0.6)

> head(round(outputCPowerTest1, digits = 4),4)

     ClassicalBootstrap VAMethod EWMethod VAFMethod DMethod WMethod VAAMethod
0.1              0.0274   0.0302   0.0305    0.0314  0.0288  0.0276    0.0278
0.15             0.0278   0.0321   0.0321    0.0305  0.0304  0.0293    0.0297
0.2              0.0315   0.0357   0.0337    0.0325  0.0339  0.0326    0.0302
0.25             0.0386   0.0393   0.0386    0.0391  0.0395  0.0397    0.0366
```

The results given both in the output matrix and in Fig. 3 show that the new resampling methods have greater power than the classical bootstrap approach. To better highlight the results, differences between the power curves for each of the new resampling methods and Efron's bootstrap are plotted in Fig. 4. Observing these results we can agree with the conclusions of simulations reported in (Grzegorzewski and Romaniuk, 2022b), that "In general (i.e. including other models) this method *(VA-method)* was usually one of the best, especially for smaller sample sizes or a lower number of bootstrap replications (e.g. $b = 100$). The VAA-method or the EW-method were usually just after VA-method; the d-method, w- and the VAA-method were neither too good nor too bad, but the classical bootstrap (as compared with other resampling methods) was usually the worse method in hypotheses testing."

Two-sample test – a real-life case

Finally, we compare the resampling methods combined with the two-sample C-test applied to some real-life imprecise data. Such data may appear naturally wherever we deal with expert opinions expressed in everyday language. In our study, we consider the opinions of the experts related to the Gamonedo cheese quality (a kind of blue cheese produced in Asturias, Spain). The inherently imprecise assessments were modeled by the TPFNs (Ramos-Guajardo et al., 2019).

To check whether two given experts, say A and C, differ significantly in their opinions, we verify the null hypothesis (8) of no difference (Grzegorzewski et al., 2020a). We can perform the desired test using the following function based on `TwoSampleCTest`:

```
CompareRealCaseTwoSample(numberOfSamples, numberOfIterations)
```

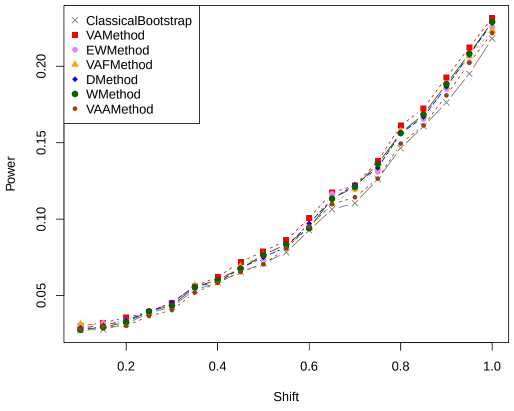


Figure 3: Power curves of the one-sample C-test based on the samples generated with `GeneratorNU`.

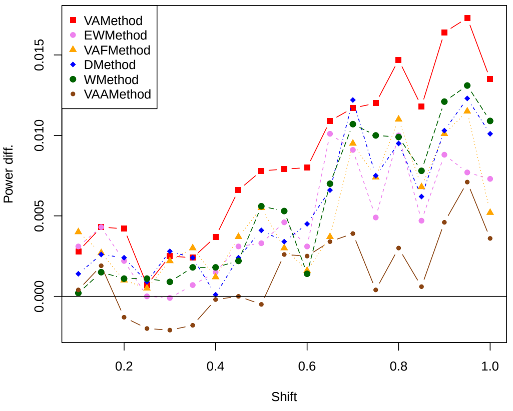


Figure 4: Differences in power curves between the new resampling methods and the classical approach.

which delivers p-values of the two-sample C-test for all resampling methods. To eliminate undesired randomness each experiment (i.e. the bootstrap procedure for the single test) is repeated `numberOfIterations` times and then the obtained p-values are averaged. The parameter `numberOfSamples` is passed directly to `TwoSampleCTest`.

Then the following function is applied:

```
> set.seed(1234567)
> realCaseP <- CompareRealCaseTwoSample(numberOfSamples=100,numberOfIterations=10000)
>
> round(realCaseP,5)
     ClassicalBootstrap VAMethod EWMethod VAFMethod DMethod WMethod VAAMethod
     0.82055            0.91213  0.88388  0.85542   0.82179 0.82212 0.85455
```

Assuming any reasonable significance level, the null hypothesis H_0 is not rejected. Thus, whichever resampling method is applied, we may conclude that the given two experts do not differ significantly in their opinions. However, the new resampling methods usually lead to greater p-values than the classical bootstrap (i.e. we are "more sure" that H_0 can be accepted).

4 Conclusions

The main goal of the proposed resampling algorithms for fuzzy samples of trapezoidal fuzzy numbers is to overcome the shortcomings typical to the classical bootstrap. New approaches are implemented in **FuzzyResampling** package and are supplemented with some ready-to-use functions useful in statistical inference based on imprecise data. It is shown that, in some cases analyzed in this paper and related literature, the suggested bootstrap algorithms are more effective than the classical one.

Obviously, many problems are still open. New methods and algorithms that would be helpful in solving many problems faced by practitioners are still needed. In particular, an in-depth study on the so-called "epistemic bootstrap" (Grzegorzewski and Romaniuk, 2021) would be useful to handle "epistemic data" instead of the "ontic data" (Couso and Dubois, 2014) discussed in this paper.

Bibliography

A. Ban, L. Coroianu, and P. Grzegorzewski. *Fuzzy Numbers: Approximations, Ranking and Applications*. Polish Academy of Sciences, Warsaw, 2015. [p272]

R. Berkachy and L. Donze. *FuzzySTs: Fuzzy Statistical Tools*, 2020. URL `https://CRAN.R-project.org/package=FuzzySTs`. R package version 0.2. [p272]

C. Bertoluzza, N. Corral, and A. Salas. On a new class of distances between fuzzy numbers. *Mathware and Soft Computing*, 2(2):71–84, 1995. URL `https://eudml.org/doc/39054`. [p275]

A. Canty and B. D. Ripley. *boot: Bootstrap R (S-Plus) Functions*, 2022. R package version 1.3-28.1. [p272]

S. Chanas. On the interval approximation of a fuzzy number. *Fuzzy Sets and Systems*, 122:353–356, 2001. URL https://doi.org/10.1016/S0165-0114(00)00080-4. [p273]

A. Colubi. Statistical inference about the means of fuzzy random variables: Applications to the analysis of fuzzy- and real-valued data. *Fuzzy Sets and Systems*, 160:344–356, 2009. URL https://doi.org/10.1016/j.fss.2007.12.019. [p276]

L. Coroianu, M. Gagolewski, and P. Grzegorzewski. Nearest piecewise linear approximation of fuzzy numbers. *Fuzzy Sets and Systems*, 233:26–51, 2013. ISSN 0165-0114. URL https://doi.org/10.1016/j.fss.2013.02.005. [p272]

I. Couso and D. Dubois. Statistical reasoning with set-valued information: Ontic vs. epistemic views. *International Journal of Approximate Reasoning*, 55:1502–1518, 2014. URL https://doi.org/10.1016/j.ijar.2013.07.002. [p280]

A. C. Davison and D. V. Hinkley. *Bootstrap Methods and their Application*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997. URL https://doi.org/10.1017/CBO9780511802843. [p272]

M. Delgado, M. Vila, and W. Voxman. On a canonical representation of a fuzzy number. *Fuzzy Sets and Systems*, 93:125–135, 1998. URL https://doi.org/10.1016/S0165-0114(96)00144-3. [p271, 273]

D. Dubois and H. Prade. The mean value of a fuzzy number. *Fuzzy Sets and Systems*, 24:279–300, 1987. URL https://doi.org/10.1016/0165-0114(87)90028-5. [p273]

B. Efron. Bootstrap methods: Another look at the jackknife. *Annals of Statistics*, 7:1–26, 1979. URL https://doi.org/10.1214/aos/1176344552. [p271]

B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Number 57 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, Florida, USA, 1993. [p271, 272]

M. Gagolewski and J. Caha. *FuzzyNumbers Package: Tools to Deal with Fuzzy Numbers in R*, 2021. URL https://github.com/gagolews/FuzzyNumbers/. [p272]

M. Gil, M. Montenegro, G. González-Rodríguez, A. Colubi, and M. Casals. Bootstrap approach to the multi-sample test of means with imprecise data. *Computational Statistics and Data Analysis*, 51: 148–162, 2006. URL https://doi.org/10.1016/j.csda.2006.04.018. [p271]

E. Giné and J. Zinn. Bootstrapping general empirical measures. *Annals of Probability*, 18(2):851–869, 1990. URL https://doi.org/10.1214/aop/1176990862. [p271]

G. González-Rodríguez, M. Montenegro, A. Colubi, and M. Gil. Bootstrap techniques and fuzzy random variables: Synergy in hypothesis testing with fuzzy data. *Fuzzy Sets and Systems*, 157: 2608–2613, 2006. URL https://doi.org/10.1016/j.fss.2003.11.021. [p271]

P. Grzegorzewski and M. Romaniuk. Epistemic bootstrap for fuzzy data. In *Joint Proceedings of IFSA-EUSFLAT-AGOP 2021 Conferences*, pages 538–545. Atlantis Press, 2021. URL https://doi.org/10.2991/asum.k.210827.071. [p280]

P. Grzegorzewski and M. Romaniuk. Bootstrapped Kolmogorov-Smirnov test for epistemic fuzzy data. In D. Ciucci, I. Couso, J. Medina, D. Ślęzak, D. Petturiti, B. Bouchon-Meunier, and R. R. Yager, editors, *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 494–507, Cham, 2022a. Springer International Publishing. ISBN 978-3-031-08974-9. URL https://doi.org/10.1007/978-3-031-08974-9_39. [p274]

P. Grzegorzewski and M. Romaniuk. Bootstrap methods for fuzzy data. In K. T. Atanassov, V. Atanassova, J. Kacprzyk, A. Kałuszko, M. Krawczak, J. W. Owsiński, S. S. Sotirov, E. Sotirova, E. Szmidt, and S. Zadrożny, editors, *Uncertainty and Imprecision in Decision Making and Decision Support: New Advances, Challenges, and Perspectives*, pages 28–47, Cham, 2022b. Springer International Publishing. ISBN 978-3-030-95929-6. URL https://doi.org/10.1007/978-3-030-95929-6_3. [p271, 273, 277, 279]

P. Grzegorzewski, O. Hryniewicz, and M. Romaniuk. Flexible bootstrap based on the canonical representation of fuzzy numbers. In *Proceedings of EUSFLAT 2019*. Atlantis Press, 2019. ISBN 978-94-6252-770-6. URL https://doi.org/10.2991/eusflat-19.2019.68. [p271, 273]

P. Grzegorzewski, O. Hryniewicz, and M. Romaniuk. Flexible resampling for fuzzy data. *International Journal of Applied Mathematics and Computer Science*, 30(2):281–297, 2020a. URL https://doi.org/10.34768/amcs-2020-0022. [p271, 273, 277, 278, 279]

P. Grzegorzewski, O. Hryniewicz, and M. Romaniuk. Flexible bootstrap for fuzzy data based on the canonical representation. *International Journal of Computational Intelligence Systems*, 13:1650–1662, 2020b. URL https://doi.org/10.2991/ijcis.d.201012.003. [p271, 273, 274, 277, 278]

S. Heilpern. The expected value of a fuzzy number. *Fuzzy Sets and Systems*, 47:81–86, 1992. URL https://doi.org/10.1016/0165-0114(92)90062-9. [p273]

M. A. Lubiano, M. Montenegro, B. Sinova, S. de la Rosa de Sáa, and M. A. Gil. Hypothesis testing for means in connection with fuzzy rating scale-based data: algorithms and applications. *European Journal of Operational Research*, 251:918–929, 2016. URL https://doi.org/10.1016/j.ejor.2015.11.016. [p271, 276]

M. A. Lubiano, A. Salas, C. Carleos, and M. A. de la Rosa de Sáa, S.and Gil. Hypothesis testing-based comparative analysis between rating scales for intrinsically imprecise data. *International Journal of Approximate Reasoning*, 88:128–147, 2017. URL https://doi.org/10.1016/j.ijar.2017.05.007. [p271]

M. Montenegro, A. Colubi, M. Casals, and M. Gil. Asymptotic and bootstrap techniques for testing the expected value of a fuzzy random variable. *Metrika*, 59:31–49, 2004. URL https://doi.org/10.1007/s001840300270. [p271]

A. Parchami. *Sim.PLFN: Simulation of Piecewise Linear Fuzzy Numbers*, 2017. URL https://CRAN.R-project.org/package=Sim.PLFN. R package version 1.0. [p272]

M. Puri and D. Ralescu. Fuzzy random variables. *Journal of the Mathematical Analysis and Applications*, 114:409–422, 1986. URL https://doi.org/10.1016/0022-247X(86)90093-4. [p273, 275]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2023. URL https://www.R-project.org/. [p271, 274]

A. Ramos-Guajardo, A. Blanco-Fernández, and G. González-Rodríguez. Applying statistical methods with imprecise data to quality control in cheese manufacturing. In P. Grzegorzewski, A. Kochanski, and J. Kacprzyk, editors, *Soft Modeling in Industrial Manufacturing*, pages 127–147. Springer, 2019. URL https://doi.org/10.1007/978-3-030-03201-2_8. [p279]

M. Romaniuk. On some applications of simulations in estimation of maintenance costs and in statistical tests for fuzzy settings. In A. Steland, E. Rafajłowicz, and O. Okhrin, editors, *Stochastic Models, Statistics and Their Applications*, pages 437–448. Springer International Publishing, 2019. URL https://doi.org/10.1007/978-3-030-28665-1_33. [p271, 273]

M. Romaniuk and O. Hryniewicz. Interval-based, nonparametric approach for resampling of fuzzy numbers. *Soft Computing*, 23:5883–5903, 2019. URL https://doi.org/10.1007/s00500-018-3251-5. [p271, 273]

M. Romaniuk and O. Hryniewicz. Discrete and smoothed resampling methods for interval-valued fuzzy numbers. *IEEE Transactions on Fuzzy Systems*, 29:599–611, 2021. ISSN 1941-0034. URL https://doi.org/10.1109/TFUZZ.2019.2957253. [p273]

M. Romaniuk, P. Grzegorzewski, and O. Hryniewicz. *FuzzyResampling: Resampling Methods for Triangular and Trapezoidal Fuzzy Numbers*, 2022. URL https://CRAN.R-project.org/package=FuzzyResampling. R package version 0.6.0. [p271]

B. W. Silverman and G. A. Young. The bootstrap: To smooth or not to smooth? *Biometrika*, 74(3): 469–479, 1987. URL https://doi.org/10.2307/2336686. [p271]

R. Tibshirani and F. Leisch. *bootstrap: Functions for the Book "An Introduction to the Bootstrap"*, 2019. URL https://CRAN.R-project.org/package=bootstrap. R package version 2019.6. [p272]

W. Trutschnig, G. González-Rodríguez, A. Colubi, and M. Gil. A new family of metrics for compact, convex (fuzzy) sets based on a generalized concept of mid and spread. *Information Sciences*, 179: 3964–3972, 2009. URL https://doi.org/10.1016/j.ins.2009.06.023. [p275]

W. Trutschnig, M. A. Lubiano, and J. Lastra. *SAFD — An R Package for Statistical Analysis of Fuzzy Data*, pages 107–118. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. URL https://doi.org/10.1007/978-3-642-30278-7_10. [p272]

Maciej Romaniuk
Systems Research Institute, Polish Academy of Sciences
Newelska 6, 01-447 Warsaw
Poland
(0000-0001-9649-396X)
mroman@ibspan.waw.pl

Przemysław Grzegorzewski
Faculty of Mathematics and Information Science, Warsaw University of Technology
Koszykowa 75, 00-662 Warsaw
Poland
Systems Research Institute, Polish Academy of Sciences
Newelska 6, 01-447 Warsaw
Poland
(0000-0002-5191-4123)
przemyslaw.grzegorzewski@pw.edu.pl

combinIT: An R Package for Combining Interaction Tests for Unreplicated Two-Way Tables

by Mahmood Kharrati-Kopaei, Zahra Shenavari, Hossein Haghbin

Abstract Several new tests have been proposed for testing interaction in unreplicated two-way analysis of variance models. Unfortunately, each test is powerful for detecting a pattern of interaction. Therefore, it is reasonable to combine multiple interaction tests to increase the power of detection for significant interactions. We introduce the package **combinIT** that provides researchers the results of six existing recommended interaction tests, including: the value of test statistics, exact Monte Carlo p-values, approximated or adjusted p-values, the results of four combined tests and explanations of interaction types if the discussed tests are significant. The software **combinIT** is a more comprehensive R package in comparison with the two existing packages. In addition, the software is executed quickly to obtain the exact Monte Carlo p-values, even for large Monte Carlo runs, in contrast to existing packages.

1 Introduction

Suppose that there are two factors A and B with a and b levels, respectively. To investigate the effect of the factors on a response variable y, an unreplicated two-way analysis of variance (ANOVA) model is sometimes used to reduce the experiment cost. Formally, this model is

$$y_{ij} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ij},\ i = 1, \ldots, a;\ \ j = 1, \ldots, b, \tag{1}$$

where y_{ij} denotes the i-th and the j-th observation of the response variable, μ is the grand mean, α_i and β_j denote the main effects of the factors, γ_{ij}s are interaction terms, and ϵ_{ij}s are random errors which are independent and identically normal variables with mean zero and variance σ^2. As noted by Franck and Osborne (2016), the unreplicated two-way ANOVA models arise in the context of (a) completely randomized two-factor experiments, (b) randomized complete block experiments that block on a source of nuisance variability, and (c) observational studies with two factors. The main problem in an unreplicated two-way ANOVA model is that all observations are used to estimate the model parameters and no observations remain to estimate σ^2; see Shenavari and Kharrati-Kopaei (2018) and Kharrati-Kopaei and Sadooghi-Alvandi (2007). Therefore, neither F-test nor t-test can be used to make inferences on the effects (including the interaction effect). The textbook approach is to assume that the interaction terms are zero (i.e. the model is additive). In this case, $(a-1)\,(b-1)$ degrees of freedom are left to estimate σ^2 and the usual F-tests can be used to make inferences about the main effects. However, if the interaction effects are present (i.e. the model is non-additive), the main effects may be masked by the interaction effect; see Montgomery (2017). Therefore, it is important to implement some method to test if there exists a significant interaction in an unreplicated ANOVA model.

The existing approaches to test interaction in unreplicated two-way ANOVA models can be categorized into two main groups: functional-based-form (FBF) and non-functional-based-form (NFBF). In FBF approaches, a functional form is assumed for the interaction terms. This group includes interaction tests proposed by Tukey (1949), Mandel (1961, 1971), Johnson and Graybill (1972), and Corsten and Eijnsbergen (1972, 1974); see Milliken and Johnson (1989, pp 2–63) for a discussion of some of these interaction tests. Simecek and Simeckova (2013) also modified Tukey's test. It is known that these tests are powerful for detecting interaction only when the specified functional form for the interaction terms is appropriate; see Boik (1993a), Alin and Kurt (2006), Simecek and Simeckova (2013), and Shenavari and Kharrati-Kopaei (2018). In NFBF approaches, a specific functional form is not assumed for the interaction terms. The NFBF group includes interaction tests proposed by Milliken and Rasmuson (1977), Tusell (1990), Boik (1993a), Piepho (1994), Kharrati-Kopaei and Sadooghi-Alvandi (2007), Franck et al. (2013), Malik et al. (2016) and Kharrati-Kopaei and Miller (2016). Kharrati-Kopaei and Sadooghi-Alvandi (2007) formally showed that there does not exist the best interaction test in an unreplicated two-way ANOVA model in the sense that there is no test that can detect all patterns of interaction with high power. This means that each test is powerful for detecting certain types of interaction. Therefore, it is meaningful to combine multiple interaction tests to provide researchers with a testing approach that leverages many existing methods to detect different patterns of non-additivity. We note that the interaction tests are dependent and the main problem is how to combine multiple dependent tests into a single test procedure for testing additivity such that it controls the Type I error rate and has an acceptable power. Shenavari and Kharrati-Kopaei (2018) evaluated the performances of different combination methods and ultimately recommended combining the interaction tests by the Bonferroni method, a Beta (Sidak) approximation, the Jacobi polynomial expansion, and the Gaussian copula method. These methods are abbreviated as Bon, Sidak, JPE, and GC, respectively, throughout this paper.

Despite the application of unreplicated two-way ANOVA models in industry, biology, agriculture, and medicine, the newly proposed interaction tests have not been discussed in statistical packages; as similarly noted by Franck and Osborne (2016). Recently, Osborne et al. (2016) released the package **hiddenf** which is available from the Comprehensive R Archive Network (CRAN). As the name of the package **hiddenf** implies, this package mainly focuses on detecting a hidden non-additivity structure proposed by Franck et al. (2013). This package also reports the p-values of tests for non-additivity developed by Tukey (1949), Mandel (1961), Kharrati-Kopaei and Sadooghi-Alvandi (2007), and Malik et al. (2016). However, this package neither provides the result of NFBF interaction tests proposed in Boik (1993a), Piepho (1994), and Kharrati-Kopaei and Miller (2016) nor the result of the combined interaction test. In addition, this package uses a Monte Carlo procedure to report the p-value of Malik et al. (2016); however, the applied procedure is rather time-consuming. Further, the package **hiddenf** provides only adjusted Bonferroni p-values of the tests proposed by Kharrati-Kopaei and Sadooghi-Alvandi (2007) and Franck et al. (2013) instead of the exact Monte Carlo p-values. Although Franck and Osborne (2016) mentioned that the Bonferroni adjustment is not overly conservative for $a \leq 7$, one might be interested in knowing the exact p-values. We note that Simeckova et al. (2014) also developed the **additivityTests** package that provides test statistics, critical values, and binary reject/fail-to-reject decisions for tests proposed by Tukey (1949), Mandel (1961), Boik (1993a), Tusell (1990), Johnson and Graybill (1972), and Simecek and Simeckova (2013). However, the **additivityTests** package does not provide the p-values of tests; see Franck and Osborne (2016).

In this paper, we introduce the R package **combinIT**, which is available from CRAN (Shenavari et al., 2022). This package reports both exact Monte Carlo and adjusted or approximate (if available) p-values of the six NFBF interaction tests developed by Boik (1993a), Piepho (1994), Kharrati-Kopaei and Sadooghi-Alvandi (2007), Franck et al. (2013), Malik et al. (2016), and Kharrati-Kopaei and Miller (2016). We use abbreviations Boik, Piepho, KKSA, Franck, Malik, and KKM, respectively, for these NFBF tests. In addition, this package provides the results of four combined interaction tests that are based on the Bon, Sidak, JPE, and GC methods. Furthermore, if a significant interaction is detected by a combined test, the package **combinIT** gives some explanations of the interaction type or pattern. Note that FBF tests have not been considered in **combinIT** because Boik (1993b) showed that the Boik test is never less powerful and is sometimes much more powerful than the test proposed by Johnson and Graybill (1972), which can be regarded as an extension of FBF tests proposed by Tukey (1949) and Mandel (1961, 1971). In addition, Boik (1993b) compared the Boik test with the NFBF test developed by Tusell (1990) and recommended the Boik test. Furthermore, the Piepho test is a modified version of the NFBF test proposed by Milliken and Rasmuson (1977) hence this test has not been considered in **combinIT**, either. In summary, the **combinIT** package reports the results of all recommended existing interaction tests in unreplicated two-way ANOVA models and, in this view, can be regarded as a more comprehensive package than **hiddenf** for testing interaction. In terms of code execution speed, nearly 25% of the code has been written in C++; see `https://github.com/haghbinh/combinIT`. We used the **Rcpp** package (Eddelbuettel and François, 2011; Eddelbuettel, 2013; Eddelbuettel and Balamuta, 2018) for writing some parts of the codes in C++. Thus, Monte Carlo simulations for calculating the p-value of the Malik test are not as time-consuming as those in the **hiddenf** package. In addition, using the **Rcpp** package allows us to calculate the exact Monte Carlo p-values of the Franck and KKSA tests in a reasonable time, in contrast to the **hiddenf** package, which provides only adjusted Bonferroni p-values; see Franck and Osborne (2016).

The rest of the paper is organized as follows. In the next section, the six NFBF interaction tests and the four combination methods that are utilized in the **combinIT** package are briefly reviewed. Then, technical details of the **combinIT** package are provided. After that, use of the package **combinIT** is illustrated with an example. The execution speed of the code is also discussed in this section. Finally, the performances of the combined interaction tests in terms of controlling the Type I error rate are discussed via a simulation study by using the **hiddenf** and **combinIT** packages.

2 Six NFBF interaction tests and four combination methods

This section has two subsections. We first review the Boik, Piepho, KKSA, Franck, Malik, and KKM interaction tests. We also discuss what patterns of interactions these tests are powerful for detecting. We then review the Bon, Beta, JPE, and GC combination methods. These tests and combination methods are utilized in the **combinIT** package. Throughout this section, the null hypothesis of interest is H_0 : There is no interaction or, equivalently, $H_0 : \gamma_{ij} = 0$ for all i and j and $r_{ij} = y_{ij} - \bar{y}_{i.} - \bar{y}_{.j} + \bar{y}_{..}$ (in usual notation) denote the residuals of the model 1 under H_0.

Six NFBF interaction tests

Boik. Let $p = \min\{a-1, b-1\}$, $q = \max\{a-1, b-1\}$ and $\mathrm{tr}\,(\boldsymbol{X})$ denote the trace of matrix $\boldsymbol{X}$. Boik (1993a) proposed a locally best invariant (LBI) test and H_0 is rejected when

$$T_{\mathrm{Boik}} = \mathrm{tr}^2\left(\boldsymbol{R}'\boldsymbol{R}\right) / \left(p\mathrm{tr}\left(\left(\boldsymbol{R}'\boldsymbol{R}\right)^2\right)\right)$$

is small, where $\boldsymbol{R} = [r_{ij}]$ denotes the residual matrix of the model 1. Boik (1993a) provided the exact distribution of the test statistic under the null hypothesis H_0 when $p = 2$ and its asymptotic distribution when q is large. Note that the p-value of the Boik test is always 1 when $p = 1$. Note that the Boik (1993a) test is powerful for detecting interaction when the matrix of interaction terms $[\gamma_{ij}]$ has small rank and one singular value dominates the remaining singular values; see Boik (1993b). For example, the Boik test is powerful for detecting the multiplicative form of interaction. The package **combinIT** provides the value of test statistic T_{Boik}, an asymptotic p-value, and an exact p-value of the Boik test based on a Monte Carlo simulation procedure.

Piepho. Piepho (1994) discussed and modified an interaction test procedure proposed by Milliken and Rasmuson (1977) by using the estimator of variance for each level of one factor; i.e.,

$$\hat{\sigma}_i^2 = \frac{a\,(a-1)\,W_i - \sum_{k=1}^{a} W_k}{(a-1)\,(a-2)\,(b-1)},$$

where $W_i = \sum_{j=1}^{b} r_{ij}^2$. Piepho (1994) proposed three test statistics. The **combinIT** package utilizes the third one, which detects a significant interaction when

$$T_{\mathrm{Piepho}} = -\,(a-1)\,(a-2)\,(b-1)\log(U)/2$$

is large, where $U = 2a\sum_{i<j}\hat{\sigma}_i^2\hat{\sigma}_j^2 / \left((a-1)\left(\sum_{i=1}^{a}\hat{\sigma}_i^2\right)^2\right)$. Piepho (1994) provided only an asymptotic chi-square distribution of T_{Piepho} under H_0. By construction, the Piepho test is powerful for detecting interactions when the estimators of variances are heterogeneous across the levels of one factor. The package **combinIT** provides the value of test statistic T_{Piepho}, an asymptotic chi-square p-value, and an exact p-value of the Piepho test based on a Monte Carlo simulation procedure.

KKSA. Suppose that $a \geq b$ and $b \geq 4$. Split the data table into two sub-tables, obtained by putting a_1 $(2 \leq a_1 \leq a-2)$ rows in the first sub-table and the remaining a_2 rows in the second sub-table $(a_1 + a_2 = a)$. The number of all possible divisions of the data table is $\mathrm{APD} = 2^{(a-1)} - a - 1$. For the l-th division, let RSS1 and RSS2 denote the residual sum of squares for the two sub-tables. Let $F_l^* = \max\{F_l, 1/F_l\}$ where $F_l = (a_2-1)\,RSS1/\left((a_1-1)\,RSS2\right)$ and P_l denote the corresponding p-value. Kharrati-Kopaei and Sadooghi-Alvandi (2007) proposed $\mathrm{minP} = \min_{1\leq l\leq \mathrm{APD}} P_l$ as a test statistic and the hypothesis of no interaction is rejected for a small value of minP. The KKSA test is powerful for detecting interaction when the magnitude of interaction effects is heteroscedastic across the sub-tables of observations. The package **combinIT** provides the value of the test statistic and an exact p-value of the KKSA test based on a Monte Carlo simulation procedure. In addition, it provides an approximate p-value based on the Bonferroni adjustment. Note that the KKSA test is not applicable when both a and b are less than 4.

Franck. Franck et al. (2013) defined the hidden additivity structure as "the levels of one factor belong in two or more groups such that within each group the effects of the two factors are additive but the groups may interact with the ungrouped factor". Based on this concept, Franck et al. (2013) proposed a test statistic by dividing the table of data into two sub-tables and developing an interaction F-test. Then, they considered all possible configurations of data and used the maximum of the interaction F-tests as a test statistic; the all-configurations and maximum-interaction F-test (ACMIF). The hypothesis of no interaction is rejected when ACMIF is large. It is clear that the Franck test is powerful when there is a hidden additivity structure in the data set. The package **combinIT** provides the value of the test statistic (ACMIF), an exact p-value of the Franck test based on a Monte Carlo simulation procedure, and an approximate p-value based on the Bonferroni adjustment.

Malik. Malik et al. (2016) used the idea that some cells may be involved in a significant interaction pattern and those cells might produce large positive or negative residuals. Therefore, Malik et al. (2016) proposed to partition the residuals into three clusters using a suitable clustering method like k-means clustering. The hypothesis of no interaction can be interpreted as the effect of the three clusters being equal. In more detail, Malik et al. (2016) proposed the model $y_{ij} = \mu^* + \alpha_i^* + \beta_j^* + \xi_{k(ij)} + \epsilon_{ij}$ where ξ_k is the cluster effect for the k-th cluster, $k = 1, 2, 3$, and $k\,(ij)$ denotes the cluster k to which the residual of the (i, j)-th cell is assigned by the clustering procedure. Malik et al. (2016) considered the hypothesis of no interaction as $H_0 : \xi_1 = \xi_2 = \xi_3$. Let $\mathrm{SSE}\,(\mathrm{cluster}) = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{3}\left(y_{ij} - \hat{\mu}^* - \hat{\alpha}_i^* - \hat{\beta}_j^* - \hat{\xi}_{k(ij)}\right)^2$ where the hat notation denotes the estimates of effects. Malik et al. (2016) proposed the following test statistic to test H_0

$$T_{\text{Malik}} = \frac{\left(\sum_{i=1}^{a}\sum_{j=1}^{b} r_{ij}^2 - \text{SEE}\,(\text{cluster})\right)/2}{\text{SSE}\,(\text{cluster})\,/\,((a-1)\,(b-1)-2)}.$$

The hypothesis H_0 is rejected for large values of T_{Malik}. Note that the result of the Malik test may depend on the method of clustering. In the **combinIT** package, clustering is done by *kmeans* function in **RcppArmadillo**. The `speed_mode` parameter on the kmeans clustering was set as `static_subset`. Note that the Malik test performs well when there are some cells that produce large negative or positive residuals due to significant interaction. The package **combinIT** provides the value of test statistic T_{Malik} and an exact p-value of the Malik test based on a Monte Carlo simulation procedure.

KKM. Let $\mu_{ij} = \mu + \alpha_i + \beta_j + \gamma_{ij}$ denote the (i, j)-th cell mean and let $\eta_{ij} = \mu_{ij} - \mu_{i'j} - \mu_{ij'} + \mu_{i'j'}, i \neq i'; j \neq j'$, denote pairwise interaction contrasts. Kharrati-Kopaei and Miller (2016) used the idea that if a significant interaction is present, its corresponding interaction contrast shall be significant. To describe this method, let $\mathbf{Z}$ be an $n \times 1$ vector with elements $y_{ij} - y_{i'j} - y_{ij'} + y_{i'j'}$ where $n = ab\,(a-1)\,(b-1)\,/4$ is the total number of estimated pairwise interaction contrasts. Kharrati-Kopaei and Miller (2016) proposed as a test statistic, where Z_k is the k-th element of $\mathbf{Z}$, PSE $=$ median$\{|Z_k|; |Z_k| \leq 5S_0\}$ in which $S_0 = \text{median}\{|Z_k|; k = 1, \ldots, n\}/c_0$ where c_0 is an unbiasing constant obtained by a Monte Carlo simulation. H_0 is rejected when T_{KKM} is large. This test procedure is powerful for detecting significant interactions when they are caused by some cells; i.e. some values of $|Z_k|$ are large. The package **combinIT** provides the value of test statistic T_{KKM} and the exact p-value of the KKM test based on a Monte Carlo simulation procedure.

Four methods for combining interaction tests

Since there is no single best interaction test in an unreplicated two-way ANOVA model, it is reasonable to combine multiple interaction tests to provide researchers with a testing approach that leverages many existing methods to detect different patterns of non-additivity. In this section, we review four combination methods that were proposed by Shenavari and Kharrati-Kopaei (2018). Throughout the section, we assume that $K \geq 2$ tests, whose test statistics are dependent and whose distribution functions are continuous, have been performed for a null hypothesis H_0. Therefore, there are K dependent p-values $P_1, \ldots, P_K$ corresponding to the interaction tests. We are going to combine these p-values into a single p-value using one of the following four test procedures for testing H_0.

Bon. Following the Bonferroni inequality, H_0 can be rejected at level α when $Kp_{\min} < \alpha$ where $p_{\min}$ is the observed value of $P_{\min} = \min_{1\leq i\leq K}\{P_i\}$. Note that the Bonferroni method leads to a conservative method of combination; i.e. its Type I error rate is always less than α; $\Pr_{H_0}\{KP_{\min} < \alpha\} \leq \alpha$.

Sidak. The distribution function of $P_{\min}$ under H_0 would be a beta distribution with parameters 1 and K if $P_i, i = 1, \ldots, K$, were independent; i.e. $\Pr_{H_0}\{P_{\min} \leq x\} = 1 - (1-x)^K$. When P_is are dependent, it can be shown that there is an $x^* \in (0,1)$ such that $\Pr_{H_0}\{P_{\min} \leq x\} \leq 1 - (1-x)^K$ for all $x < x^*$; see Sadooghi-Alvandi and Kharrati-Kopaei (2015). Therefore, H_0 would be rejected at level α when $1 - (1-p_{\min})^K < \alpha$.

JPE. The distribution of $P_{\min}$ under H_0 can be approximated by the Jacobi polynomial expansion. In this expansion, the distribution of $P_{\min}$ is approximated by the weighted sum of several Beta distribution functions with parameters $p = 1$ and $q = 1/E_1 - 1$ where E_1 denotes the first moment of $P_{\min}$; see Shenavari and Kharrati-Kopaei (2018). Up to the second term in the Jacobi polynomial expansion, the distribution of $P_{\min}$ is approximated by a Beta distribution with parameters 1 and q. In practice, the parameter q can be estimated as $\hat{q} = \min\{\max\{1, 1/P_{\min} - 1\}, K - 1\}$; see Shenavari and Kharrati-Kopaei (2018). By this approximation, the null hypothesis H_0 is rejected at level α when $1 - (1-p_{\min})^{\hat{q}} < \alpha$.

GC. Let $Z_i = \Phi^{-1}(P_i)$, $i = 1, \ldots, K$, where $\Phi^{-1}(.)$ is the quantile function of the standard normal distribution. Although it is well known that Z_is are marginally distributed as the standard normal distribution under the null hypothesis H_0, the joint distribution of the Z_i is unknown in practice. One can use the Gaussian copula (GC) family to model the joint distribution of Z_is. Therefore, one can assume that the random vector $(Z_1, \ldots, Z_K)$ has a K-variate normal distribution under the null hypothesis H_0 with zero mean vector and variance-covariance matrix $\mathbf{\Sigma}$. In this case, the null hypothesis H_0 is rejected at level α when

$$\Pr_{H_0}\{P_{\min} \leq p_{\min}\} = 1 - \Pr_{H_0}\{Z_1 > \Phi^{-1}(p_{\min}), \ldots, Z_K > \Phi^{-1}(p_{\min})\} < \alpha.$$

This probability is calculated by using **mvtnorm** in the package **combinIT**. Note that $\mathbf{\Sigma}$ can be estimated by $\hat{\mathbf{\Sigma}} = (1-\hat{\rho})\,\boldsymbol{I}_K + \hat{\rho}\mathbf{1}_K\mathbf{1}_K^{'}$ where $\boldsymbol{I}_K$ is a $K \times K$ identity matrix, $\mathbf{1}_K$ is a K-dimensional vector of ones, and $\hat{\rho} = \max\{-1/(K-1), 1 - \sum_{i=1}^{K}(Z_i - \bar{Z})^2/(K-1)\}$; see Shenavari and Kharrati-Kopaei (2018).

Note 1. If the null hypothesis H_0 is rejected, the combined tests can suggest a pattern of non-additivity by examining the interaction test for which $P_{\min} = \min_{1 \le i \le K}\{P_i\}$ occurs.

Note 2. The inequality $Kp_{\min} \le 1 - (1 - p_{\min})^K \le 1 - (1 - p_{\min})^{\hat{q}}$ guarantees that the JPE method provides an interaction test that is never less powerful and is sometimes much more powerful than the Bon and Sidak counterparts.

3 Technical details about the combinIT package

The **combinIT** package stands for "Combined Interaction Test". This package allows users to perform the six NFBF interaction tests individually and to perform a combined test of the six NFBF interaction tests. It also has a function to produce an interaction plot and four data sets. The available functions in the **combinIT** package are described in Table 1. This package produces two S3-class objects: (i) ITtest, produced by functions that perform a single test and (ii) combtest, produced by the function that performs the combined test. These returned objects are lists with the following components:

(i) `ITtest` class: For the return object of the functions `Malik_test`, `Franck_test`, `Boik_test`, `KKSA_test`, `KKM_test`, and `Piepho_test`. An object of this class consists of a list of six components, including:

- `pvalue_exact`: The calculated exact Monte Carlo p-value. For `Boik_test`, this is an exact Monte Carlo p-value when $p > 2$ and it is an exact p-value for $p = 2$.
- `pvalue_appro`: For `Piepho_test` this is an asymptotic p-value. This component is the Bonferroni-adjusted p-value for `Franck.test` and `KKSA_test`. This component is not available for `Malik_test` and `KKM_test`.
- `nsim`: The number of Monte Carlo samples used to calculate the exact Monte Carlo p-values.
- `statistic`: The value of the test statistic.
- `data_name`: The name of the input data set.
- `test`: The name of the test.

(ii) `combtest` class: For the return object of the function `CI_test`. An object of this class consists of a list of 19 components, including:

- `nsim`: The number of Monte Carlo samples used to calculate the exact Monte Carlo p-values.
- `Piepho_pvalue`: The p-value of Piepho's (1994) test.
- `Piepho_Stat`: The value of Piepho's (1994) test statistic.
- `Boik_pvalue`: The p-value of Boik test.
- `Boik_Stat`: The value of Boik's (1993) test statistic.
- `Malik_pvalue`: The p-value of Malik, et al.'s (2016) test.
- `Malik_Stat`: The value of Malik, et al.'s (2016) test statistic.
- `KKM_pvalue`: The p-value of Kharrati-Kopaei and Miller's (2016) test.
- `KKM_Stat`: The value of Kharrati-Kopaei and Miller's (2016) test statistic.
- `KKSA_pvalue`: The p-value of Kharrati-Kopaei and Sadooghi-Alvandi's (2007) test.
- `KKSA_Stat`: The value of Kharrati-Kopaei and Sadooghi-Alvandi's (2007) test statistic.
- `Franck_pvalue`: The p-value of Franck, et al.'s (2013) test.
- `Franck_Stat`: The value of Franck, et al.'s (2013) test statistic.
- `Bonferroni`: The combined p-value by using the Bonferroni method.
- `Sidak`: The combined p-value by using the Sidak method.
- `Jacobi`: The combined p-value by using the Jacobi method.
- `GC`: The combined p-value by using the Gaussian copula.
- `data_name`: The name of the input data set.
- `test`: The name of the test.

The `print` method is developed for a brief displaying of these objects. One of the key features of this package is to make it possible for users to obtain accurate results for various tests within a reasonable time. This feature was achieved by assigning the time-consuming parts of these tests to several C++ functions that are called from R using **Rcpp**.

Function	Descriptions	Inputs	Main outputs
`Boik_test`	Performs the Boik test	The data matrix, the number of Monte Carlo samples for calculating the exact p-value.	The value of test statistic, an asymptotic p-value, the exact Monte Carlo p-value.
`Piepho_test`	Performs the Piepho test	The data matrix, the number of Monte Carlo samples for calculating an exact p-value.	The value of test statistic, an asymptotic p-value, the exact Monte Carlo p-value.
`KKSA_test`	Performs the KKSA test	The data matrix, the number of Monte Carlo samples for calculating the exact p-value, logical Elapsed_time for printing the progress.	The value of test statistic, an adjusted p-value, the exact Monte Carlo p-value.
`Franck_test`	Performs the Franck test	The data matrix, the number of Monte Carlo samples for calculating the exact p-value, logical Elapsed_time for printing the progress.	The value of test statistic, an adjusted p-value, the exact Monte Carlo p-value.
`Malik_test`	Performs the Malik test	The data matrix, the number of Monte Carlo samples for calculating the exact p-value, logical Elapsed_time for printing the progress.	The value of test statistic and the exact Monte Carlo p-value.
`KKM_test`	Performs the KKM test	The data matrix, the number of Monte Carlo samples for calculating the exact p-value and the unbiasing constant.	The value of test statistic and the exact Monte Carlo p-value.
`CI_test`	Performs the six NFBF tests and their combined tests	The data matrix, the number of Monte Carlo samples for calculating the exact p-value and the unbiasing constant, logical Elapsed_time for printing the progress.	The value of six test statistics, the exact Monte Carlo p-values of tests, and the result of four combined tests.
interaction_plot	Plots the interaction plot	The matrix of data.	Interaction plot.

Table 1: A summary of the functions in the **combinIT** package .

4 Using the package combinIT

Here, we illustrate the usage of the **combinIT** package. We also compare the new package with **hiddenf** in terms of execution speed. For reference, the functions of **combinIT** have been shown in Table 1. This table also presents some information about the inputs and outputs of functions.

We consider the copy number variation (CNV) data set that contains CNV values for normal and tumor tissue samples among six dogs. In this data set, the value of CNV was measured as a signal intensity obtained from a comparative genomic hybridization (CGH) array, with higher signals corresponding to higher copy numbers; see Franck et al. (2013) and Osborne et al. (2016). The following code loads the data and plots the interaction plot. It can be seen from Figure 1 that a hidden structure might exist (rows 4 and 5 are parallel and they interact with rows 1, 2, 3, and 6).

```
> library(combinIT)
> data(CNV)
> interaction_plot(CNV)
```

Now, we use `CI_test` to combine the result of the six interaction tests to see if there exists any significant interaction. We note that the exact Monte Carlo p-values are obtained based on 10000 Monte Carlo samples and hence the absolute error of Monte Carlo approximation is at most 0.0082 with the 0.95 confidence coefficient (following the Central Limit Theorem). In addition, the p-value of the Boik test will be 1 since $p = 1$. We also set $\alpha = 0.05$ (as default to test at the 5% level) and `report=TRUE` (to provide a report on the test), and `opvalue=0.6187` as the p-value of the Tukey's test (that is, we are going to combine the p-value of the Tukey's test in addition to the six interaction tests).

```
> CI_test(CNV, nsim = 10000, alpha = 0.05, report = TRUE, nc0 = 10000,
opvalue = 0.6187, Elapsed_time = FALSE)
Test:   Combined interaction Test
Data:   CNV
Piepho Test: Statistic =  1.73949 , Pvalue =  0.8743
```

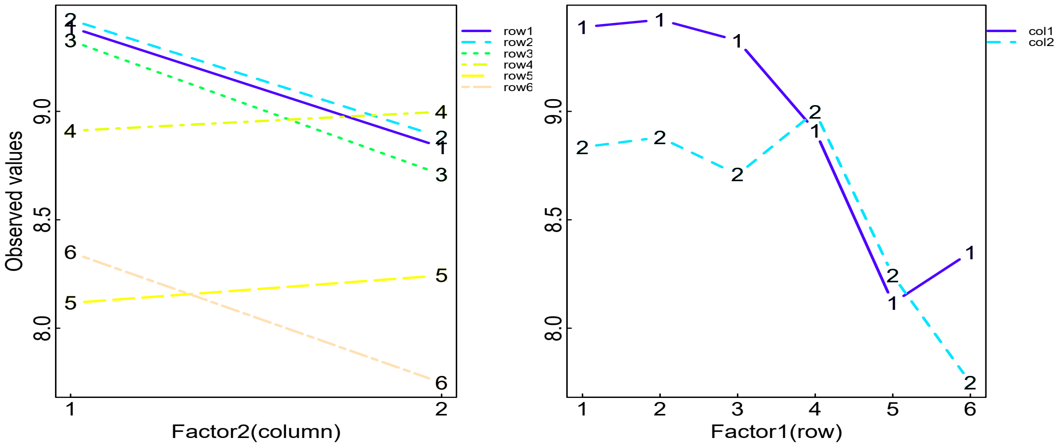


Figure 1: The interaction plot of the CNV data.

```
Boik Test: Statistic =  1 , Pvalue =  1
Malik Test: Statistic =  526.66767 , Pvalue =  0.0289
KKM Test: Statistic =  1.18173 , Pvalue =  0.9994
KKSA Test: Statistic =  0.0112 , Pvalue =  0.2092
Franck Test: Statistic =  526.66767 , Pvalue =  7e-04
Bonferroni method: Pvalue = 0.0049
Sidak method: Pvalue = 0.00489
Jacobi method: Pvalue = 0.00419
Gaussian copula: Pvalue = 0.00491
------------------------------------------------
A report on the combined interaction test:
A  significant  hidden  structure  exists  at  the 5% level. The first
group  includes  rows:  4, 5. The second group includes rows: 1, 2, 3,
6.  The  estimated  critical  value of the Franck_test at the 5% level
with 10000 Monte Carlo samples is 57.327.
```

The CI_test function has detected a significant interaction at the 5% level and the detected interaction type is the hidden structure as we expect. All combined tests report that there is a significant interaction. It is worth mentioning that a significant interaction would not be detected if only Boik or Piepho or KKM or KKSA test was used. However, the combined tests provide researchers with much more opportunity to detect a significant interaction. One can use the Franck_test to analyze the hidden structure in the CNV data set in more detail. The argument plot was set as TRUE to plot the interaction plot in the Franck_test function. This interaction plot in Figure 2 shows that rows 4 and 5 are parallel (plotted in blue and solid line) and that they interact with rows 1, 2, 3, and 6 (plotted in red and broken line).

```
> Franck_test(CNV, nsim = 10000, alpha = 0.05, report = TRUE, plot = TRUE,
Elapsed_time = FALSE)
Test:   Franck Test
Data:   CNV
Statistic =  526.668
Exact Monte Carlo P-value =  0
Approximate P-value =  0.001
Nsim =  10000
---------------------------------------
A report on the test:
A  significant  hidden  structure  exists  at  the 5% level. The first
group  includes  rows:  4, 5. The second group includes rows: 1, 2, 3,
6.  The  estimated  critical  value of the Franck_test at the 5% level
with 10000 Monte Carlo samples is 56.7246.
```

Although the other five interaction tests can be used individually to test interaction, one should take care that a Type I error has not resulted from the multiple tests that have been performed. Code for each test is shown below.

```
> Boik_test(CNV, nsim = 10000, alpha = 0.05, report = TRUE)
Test:   Boik Test
```

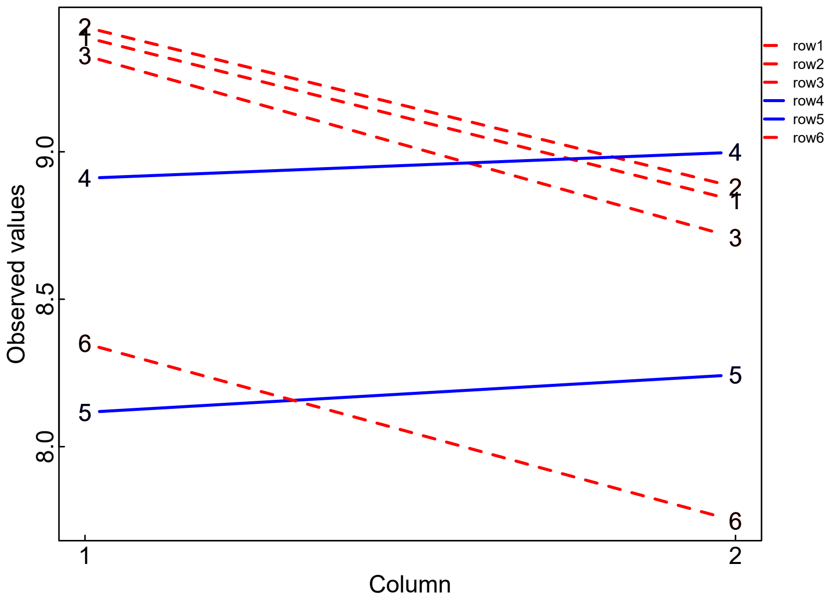


Figure 2: The interaction plot of the CNV data to see the hidden structure of interaction.

```
Data:   CNV
Statistic =  1
Exact Monte Carlo P-value =  1
Approximate P-value =  1
Nsim =  10000
---------------------------------------
A report on the test:
The  Boik_test  could not detect any significant interaction at the 5%
level. The exact critical value of the Boik_test is 1.
> Piepho_test(CNV, nsim = 10000, alpha =0.05, report = TRUE)
Test:   Piepho Test
Data:   CNV
Statistic =  1.739
Exact Monte Carlo P-value =  0.878
Approximate P-value =  0.884
Nsim =  10000
---------------------------------------
 A report on the test:
 The  Piepho_test  could  not detect any significant interaction at the
 5%  level.  The  estimated critical value of the Piepho_test at the 5%
 level with 10000 Monte Carlo samples is 12.5169.
> KKSA_test(CNV, nsim = 10000, alpha = 0.05, report = TRUE, plot = FALSE,
Elapsed_time = FALSE)
Test:   KKSA Test
Data:   CNV
Statistic =  0.011
Exact Monte Carlo P-value =  0.206
Approximate P-value =  0.28
Nsim =  10000
---------------------------------------
 A report on the test:
 The  KKSA_test  could not detect any significant interaction at the 5%
 level.  The  estimated critical value of the KKSA_test at the 5% level
 with 10000 Monte Carlo samples is 0.0023.
> Malik_test(CNV, nsim = 10000, alpha = 0.05, report = TRUE, Elapsed_time = FALSE)
Test:   Malik Test
Data:   CNV
Statistic =  526.668
Exact Monte Carlo P-value =  0.026
```

```
Approximate P-value =  NULL
Nsim =  10000
---------------------------------------
A report on the test:
There   exists   a   significant   interaction  at  the  5%  level.The
significant  interaction  might due to the some outliers in residuals;
some cells produce large negative or positive residuals:
The cell with row=5 and column=1 produces a large negative residual
The cell with row=5 and column=2 produces a large positive residual
The  estimated  critical  value of the Malik_test at the 5% level with
10000 Monte Carlo samples is 362.0247.
> KKM_test(CNV, nsim = 10000, alpha = 0.05, report = TRUE, nc0 = 10000)
Test:   KKM Test
Data:   CNV
Statistic =  1.182
Exact Monte Carlo P-value =  0.999
Approximate P-value =  NULL
Nsim =  10000
---------------------------------------
 A report on the test:
 The  KKM_test  could  not detect any significant interaction at the 5%
 level.  The  estimated  critical value of the KKM_test at the 5% level
 with 10000 Monte Carlo samples is 4.2359.
```

As we see, Boik_test, Piepho_test, KKSA_test, and KKM_test could not detect any significant interaction at the 5% level. However, Malik_test is significant and the significant interaction is due to the cells $(5, 1)$ and $(5, 2)$ that produce large negative and positive residuals, respectively. Note that the KKSA_test has an argument plot that if plot is TRUE an interaction plot will be plotted. Color and line type are used to display which levels of row factor are assigned to which sub-tables based on the minimum p-values among all possible configurations. The values of the Monte Carlo p-value and the adjusted p-value of the Franck test differ little (the presented values were rounded up to three decimal digits and the actual value of the Monte Carlo p-value is around 0.00069); on the other hand, they differ for the KKSA test. This shows the importance of calculating the exact Monte Carlo p-values, while the package **hiddenf** provides only the adjusted p-values.

We now compare the execution speed of the Malik_test function in the **combinIT** package with the MalikPavlue function in the **hiddenf** package. To do this, we consider the CNV data again. We use the command system.time in R to record the elapsed time of execution in seconds for comparing the speed of execution. The elapsed time for 500 (the default of the **hiddenf** package), 1000, 5000, 10000 (the default of the **combinIT** package), 50000, and 100000 Monte Carlo samples are shown in Figure 3. All computations were done on a laptop with Windows 10 operating system, Intel(R) Core(TM) i5-4200U CPU,1.60GHz 2.30 GHz processor, and 8G of RAM. The following shows the R code used and the output for 500 Monte Carlo samples.

```
> library(hiddenf)
> CNV.mtx<-HiddenF(CNV)
> system.time(
+ Malik_test(CNV,nsim=500,Elapsed_time = FALSE)
+ )
user  system elapsed
0.15    0.13    0.22
> system.time(
+ MalikPvalue(CNV.mtx,N=500)
+ )
(Pvalue from Malik's test estimated with N=500 Monte Carlo datasets)
user  system elapsed
4.05    0.00    4.08
```

It is seen that the run time of the Malik_test function is much shorter than that of the MalikPvalue function. Actually, the run speed of the Malik_test function is at least 18.5 times faster than the run speed of the MalikPvalue function.

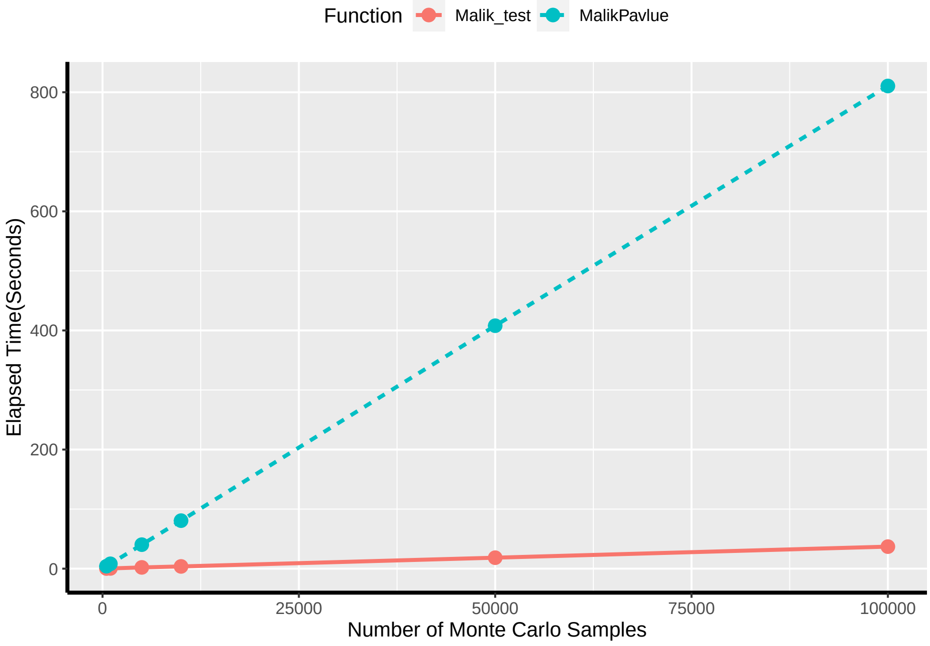


Figure 3: The elapsed time of executing `MalikPvalue` and `Malik_test` functions for the CNV data.

5 Simulation study

In this section, we examine the performance of the Bon, Sidak, JPE, and GC methods in terms of controlling the Type I error rate using the **combinIT** and **hiddenf** packages. Note that Shenavari and Kharrati-Kopaei (2018) evaluated the performance of these methods for combining the p-values of the Boik, Piepho, KKSA, Franck, KKM, and Malik tests. Here, in addition to the six p-values, we consider the p-values of Tukey's test and Mandel's test (Tukey, 1949; Mandel, 1961) as the two other tests that may apply. We used the following procedure to estimate the Type I error rate of the combination methods. For given a and b, an $a \times b$ data table is generated from the standard normal distribution. We also took $\sigma^2 = 1$ without loss of generality (since the tests are scale invariant). For the computed interaction effects" changes to "For the generated data table, the tests are done and their p-values are recorded. For each method of combination, these p-values are combined and the result of the combined test's rejection of the null hypothesis of no interaction is recorded. This process is repeated $N1$ times and the fraction of times that the combined test rejects the null hypothesis is calculated as an estimate of the Type I error rate of the combining method.

The results of simulation for $\alpha = 0.10, 0.05, 0.01$, $N1 = 10,000$, and different values of a and b are shown in Table 2. The exact Monte Carlo p-values of tests were calculated based on $N2 = 100,000$ Monte Carlo samples. In this table, the estimated absolute errors (EAE) in estimating the Type I error rate with confidence coefficient 0.95 are shown in parentheses. The value of EAE with confidence coefficient 0.95 is $1.96\sqrt{\hat{\alpha}\,(1-\hat{\alpha})\,/N1}$ where $\hat{\alpha}$ denotes the estimated Type I error rate.

It is seen from Table 2 that the combining methods control the Type I error rate. This shows that these methods can be used for combining the p-values of the other tests successfully that may be introduced in the future. Among these four combination methods, the JPE method performs better than the others since the values of its estimated Type I error are fairly close to the nominal Type I error. We did not examine the performance of methods in terms of detecting a significant interaction power, because the considered methods do not have the same estimated Type I error rate (The JPE method would have the highest power, because its estimated Type I error rate is higher than the other methods; see also **Note 2**).

6 Summary

We reviewed six recommended interaction tests in the context of unreplicated two-way ANOVA models and four combination methods for combining dependent interaction tests. Then, we introduced the **combinIT** package and demonstrated that the advantages of this package over the two existing packages are: (i) it reports test statistics, estimated critical value, approximate p-values, and exact

		Methods			
α	$a \times b$	Bon	Sidak	JPE	GC
0.10	4×3	0.0681(0.0049)	0.0716(0.0051)	0.0822(0.0054)	0.0770(0.0052)
	5×3	0.0738(0.0051)	0.0771(0.0052)	0.0876(0.0055)	0.0784(0.0054)
	5×5	0.0752(0.0052)	0.0792(0.0053)	0.0879(0.0055)	0.0799(0.0053)
	6×6	0.0723(0.0051)	0.0749(0.0052)	0.0849(0.0055)	0.0764(0.0052)
0.05	4×3	0.0372(0.0037)	0.0377(0.0037)	0.0423(0.0039)	0.0388(0.0038)
	5×3	0.0384(0.0038)	0.0394(0.0038)	0.0441(0.0040)	0.0396(0.0038)
	5×5	0.0399(0.0038)	0.0411(0.0038)	0.0463(0.0041)	0.0406(0.0039)
	6×6	0.0395(0.0038)	0.397(0.0038)	0.0448(0.0041)	0.0402(0.0038)
0.01	4×3	0.0090(0.0019)	0.0090(0.0019)	0.0103(0.0020)	0.0091(0.0019)
	5×3	0.0078(0.0017)	0.0078(0.0017)	0.0088(0.0018)	0.0079(0.0017)
	5×5	0.0090(0.0019)	0.0092(0.0019)	0.0101(0.0020)	0.0092(0.0019)
	6×6	0.0087(0.0018)	0.0087(0.0018)	0.0096(0.0019)	0.0086(0.0018)

Table 2: The estimated Type I error rates of the combined tests and their EAE (the values in parentheses).

Monte Carlo p-values of six recommended NFBF interaction tests, (ii) it provides the results of the four combined interaction tests in addition to some descriptions of detected significant interaction patterns, and (iii) its execution is fast enough to make it feasible to calculate the exact Monte Carlo p-values. Using the **combinIT** package, we evaluated the performance of the four combined tests in terms of controlling the Type I error rate. Simulation results show that the combined tests control the Type I error rate, and therefore can be used as an interaction test that leverages existing methods to detect different patterns of non-additivity.

We note that the **combinIT** package can handle data sets as large as 15 × 15, although performing the KKSA and Franck tests may be time-consuming when the size of the data set is larger. We will focus on solving this problem in the next version of the package.

7 Acknowledgment

The first author would like to acknowledge the Research Council of Shiraz University.

Bibliography

A. Alin and S. Kurt. Testing non-additivity (interaction) in two-way anova tables with no replication. *Statistical Methods in Medical Research*, 15(1):63–85, 2006. URL https://doi.org/10.1191/0962280206sm426oa. PMID: 16477949. [p284]

R. J. Boik. Testing additivity in two-way classifications with no replications:the locally best invariant test. *Journal of Applied Statistics*, 20(1):41–55, 1993a. URL https://doi.org/10.1080/02664769300000004. [p284, 285, 286]

R. J. Boik. A comparison of three invariant tests of additivity in two-way classifications with no replications. *Computational Statistics and Data Analysis*, 15:411–424, 1993b. URL https://doi.org/10.1016/0167-9473(93)90173-Q. [p285, 286]

L. C. A. Corsten and A. C. v. Eijnsbergen. Multiplicative effects in two-way analysis of variance. *Statistica Neerlandica*, 26(3):61–68, 1972. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9574.1972.tb00173.x. [p284]

L. C. A. Corsten and A. C. v. Eijnsbergen. Addendum to "multiplicative effects in two-way analysis of variance". *Statistica Neerlandica*, 28(1):51–51, 1974. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9574.1974.tb00231.x. [p284]

D. Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer, New York, 2013. URL https://doi.org/10.1007/978-1-4614-6868-4. ISBN 978-1-4614-6867-7. [p285]

D. Eddelbuettel and J. J. Balamuta. Extending extitR with extitC++: A Brief Introduction to extitRcpp. *The American Statistician*, 72(1):28–36, 2018. URL https://doi.org/10.1080/00031305.2017.1375990. [p285]

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL https://doi.org/10.18637/jss.v040.i08. [p285]

C. T. Franck and J. A. Osborne. Exploring interaction effects in two-factor studies using the hiddenf package in r. *The R Journal*, 8(1):159–172, 2016. URL https://doi.org/10.32614/RJ-2016-011. [p284, 285]

C. T. Franck, D. M. Nielsen, and J. A. Osborne. A method for detecting hidden additivity in two-factor unreplicated experiments. *Computational Statistics and Data Analysis*, 67:95–104, Nov 2013. ISSN 0167-9473. URL http://dx.doi.org/10.1016/j.csda.2013.05.002. [p284, 285, 286, 289]

D. E. Johnson and F. A. Graybill. An analysis of a two-way model with interaction and no replication. *Journal of the American Statistical Association*, 67(340):862–868, 1972. ISSN 01621459. URL http://www.jstor.org/stable/2284651. [p284, 285]

M. Kharrati-Kopaei and A. Miller. A method for testing interaction in unreplicated two-way tables: using all pairwise interaction contrasts. *Journal of Statistical Computation and Simulation*, 86(6): 1203–1215, 2016. URL https://doi.org/10.1080/00949655.2015.1057821. [p284, 285, 287]

M. Kharrati-Kopaei and S. M. Sadooghi-Alvandi. A new method for testing interaction in unreplicated two-way analysis of variance. *Communications in Statistics - Theory and Methods*, 36:2787 – 2803, 2007. URL https://doi.org/10.1080/03610920701386851. [p284, 285, 286]

W. A. Malik, J. Möhring, and H. P. Piepho. A clustering-based test for nonadditivity in an unreplicated two-way layout. *Communications in Statistics - Simulation and Computation*, 45(2):660–670, 2016. URL https://doi.org/10.1080/03610918.2013.870196. [p284, 285, 286]

J. Mandel. Non-additivity in two-way analysis of variance. *Journal of the American Statistical Association*, 56(296):878–888, 1961. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1961.10482132. [p284, 285, 293]

J. Mandel. A new analysis of variance model for non-additive data. *Technometrics*, 13(1):1–18, 1971. URL https://www.tandfonline.com/doi/abs/10.1080/00401706.1971.10488751. [p284, 285]

G. A. Milliken and D. E. Johnson. *Analysis of Messy Data, Volume II: Nonreplicated Experiments*. Chapman and Hall/CRC, 1st ed edition, 1989. URL https://doi.org/10.1201/9781315172194. [p284]

G. A. Milliken and D. M. Rasmuson. A heuristic technique for testing for the presence of interaction in nonreplicated factorial experiments. *Australian and New Zealand Journal of Statistics*, 19:32–38, 1977. URL https://onlinelibrary.wiley.com/doi/10.1111/j.1467-842X.1994.tb00889.x. [p284, 285, 286]

D. C. Montgomery. *Design and Analysis of Experiments*. John Wiley and Sons, Inc., Hoboken, NJ, USA, 2017. ISBN 9781119113478. [p284]

J. A. Osborne, C. T. Franck, and B. Choi. *hiddenf: The All-Configurations, Maximum-Interaction F-Test for Hidden Additivity*, 2016. URL https://CRAN.R-project.org/package=hiddenf. R package version 2.0. [p285, 289]

H.-P. Piepho. On tests for interaction in a nonreplicated two-way layout. *Australian Journal of Statistics*, 36(3):363–369, 1994. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-842X.1994.tb00889.x. [p284, 285, 286]

S. M. Sadooghi-Alvandi and M. Kharrati-Kopaei. Testing three-factor interaction in unreplicated three-way layouts. *Communications in Statistics - Theory and Methods*, 44(3):587–606, 2015. URL https://doi.org/10.1080/03610926.2012.748915. [p287]

Z. Shenavari and M. Kharrati-Kopaei. A method for testing additivity in unreplicated two-way layouts based on combining multiple interaction tests. *International Statistical Review*, 86(3):469–487, 2018. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/insr.12262. [p284, 287, 288, 293]

Z. Shenavari, H. Haghbin, M. Kharrati-Kopaei, and S. M. Najibi. *combinIT: A Combined Interaction Test for Unreplicated Two-Way Tables*, 2022. URL https://CRAN.R-project.org/package=combinIT. R package version 2.0.0. [p285]

P. Simecek and M. Simeckova. Modification of tukey's additivity test. *Journal of Statistical Planning and Inference*, 143:197–201, 2013. URL https://doi.org/10.1016/j.jspi.2012.07.002. [p284, 285]

M. Simeckova, T. Rusch, and P. Simecek. *additivityTests: Additivity Tests in the Two Way Anova with Single Sub-class Numbers*, 2014. URL https://CRAN.R-project.org/package=additivityTests. R package version 1.1-4. [p285]

J. W. Tukey. One degree of freedom for non-additivity. *Biometrics*, 5(3):232–242, 1949. ISSN 0006341X, 15410420. URL http://www.jstor.org/stable/3001938. [p284, 285, 293]

F. Tusell. Testing for interaction in two-way anova tables with no replication. *Computational Statistics and Data Analysis*, 10(1):29–45, 1990. ISSN 0167-9473. URL https://www.sciencedirect.com/science/article/pii/016794739090101M. [p284, 285]

Mahmood Kharrati-Kopaei
Department of Statistics, College of Science, Shiraz University
Shiraz
Iran
(ORCiD 0000-0001-5555-253X)
mkharati@shirazu.ac.ir

Zahra Shenavari
Department of Mathematics, Faculty of Sciences, Shiraz Branch, Islamic Azad University
Shiraz
Iran
(ORCiD 0000-0001-8347-0590)
zshenavari@yahoo.com

Hossein Haghbin
Department of Statistics, Faculty of Intelligent Systems Engineering and Data Science, Persian Gulf University
Boushehr
Iran
(ORCiD 0000-0001-8416-2354)
https://github.com/haghbinh
haghbin@pgu.ac.ir

Estimating Causal Effects using Bayesian Methods with the R Package BayesCACE

by Jincheng Zhou, Jinhui Yang, James S. Hodges, Lifeng Lin, and Haitao Chu

Abstract Noncompliance, a common problem in randomized clinical trials (RCTs), complicates the analysis of the causal treatment effect, especially in meta-analysis of RCTs. The complier average causal effect (CACE) measures the effect of an intervention in the latent subgroup of the population that complies with its assigned treatment (the compliers). Recently, Bayesian hierarchical approaches have been proposed to estimate the CACE in a single RCT and a meta-analysis of RCTs. We develop an R package, BayesCACE, to provide user-friendly functions for implementing CACE analysis for binary outcomes based on the flexible Bayesian hierarchical framework. This package includes functions for analyzing data from a single study and for performing a meta-analysis with either complete or incomplete compliance data. The package also provides various functions for generating forest, trace, posterior density, and auto-correlation plots, which can be useful to review noncompliance rates, visually assess the model, and obtain study-specific and overall CACEs.

1 Introduction

Noncompliance in randomized clinical trials and causal effect

Randomized clinical trials (RCTs) are often used to evaluate healthcare-related interventions. An RCT typically compares an experimental treatment to a standard treatment or to a placebo. A common problem in RCTs is that not all patients fully comply with the allocated treatments. Although RCT investigators control the randomization process, the actual treatments received by study participants may not follow the randomization allocation; this is called noncompliance. For example, in trials of a therapist-led intervention, noncompliance occurs when individuals randomized to the intervention fail to take the intervention (e.g., due to severe adverse events), or when some patients assigned to the control figure out a way to take the intervention. In some cases, investigators can collect outcome data on all of these patients, regardless of whether they followed interventions. When compliance status is incompletely observed, it is more complicated to evaluate the causal treatment effect.

Conventionally, researchers use the intention-to-treat (ITT) analysis, in which data are analyzed based on treatments originally allocated rather than treatments actually received. The ITT method estimates the effect of being offered the intervention, namely, the overall effect in the real world in which the intervention is made available. However, our interest may lie in a different question, namely the causal effect of actually receiving the treatment. When using ITT, the treatment effect tends to be diluted by including people who do not receive the treatment to which they were randomly allocated (Freedman 1990).

To identify a treatment's causal effect, the principal stratification framework (Frangakis and Rubin 2002) is proposed, which stratifies subjects on the joint potential post-randomization variables. This causal inference method is widely used in handling various intercurrent events (also called an intermediate variable) in areas like vaccine effect (Hudgens and Halloran 2006; J. Zhou et al. 2016), pain relief use (Baccini, Mattei, and Mealli 2017), surrogate endpoint evaluation (Gilbert et al. 2015), noncompliance (J. Zhou et al. 2019), etc. An estimator called the "complier average causal effect" (CACE) has been proposed, in which patients are classified into different principal strata (compliers, never-takers, always-takers, and defiers) based on their *potential* behavior after assignment to both the treatment and control arms. *Compliers* are patients who receive the treatment as assigned; *never-takers* are those who do not receive treatment, regardless of treatment assignment; *always-takers* are those who receive treatment regardless of treatment assignment; and patients who always do the opposite of their treatment assignment are called *defiers*. The CACE is the causal effect of the intervention estimated from compliers. Because patients are assumed to be compliers (or not) before the randomization, the CACE retains the benefit of the randomization. Specifically, CACE is an unbiased estimate of the difference in outcomes for compliers in the intervention group compared to those in the control group, who would have engaged with treatment had they been randomized to the intervention group.

The biggest challenge in estimating the CACE is that we cannot actually identify which participants are compliers. Some of those receiving the treatment in the intervention group are compliers, but the rest are always-takers. Similarly, some of those not receiving the treatment in the control arm are compliers, but others are never-takers. Several R packages are available to perform CACE analysis in a single study. For example, the **noncomplyR** package (Coggeshall 2017) provides convenient functions for using Bayesian methods to perform inferences on the CACE. The package **eefAnalytics** (Kasim et al.

2017) provides tools for exploratory CACE analysis of simple randomized trials, cluster randomized trials, and multi-site trials with a focus on education trials. Besides the CACE analysis, another method commonly used to account for noncompliance is the instrumental variable (IV) method estimating the treatment effect with two-staged least squares (2SLS) regression (White 1982); the archived R package **ivpack** (Jiang and Small 2014) performs this type of analysis.

CACE in meta-analysis

All of the above methods are framed in a single study setting. However, for analyzing multiple trials in the presence of noncompliance, no software is available for causal effect analysis, specifically for meta-analysis. When noncompliance data are reported in each trial, one could intuitively implement a two-step approach by first estimating CACE for each study and then combining the study-specific estimates using a fixed-effect or random-effects model to estimate the population-averaged CACE. Recently, J. Zhou et al. (2019) proposed a Bayesian hierarchical model to estimate the CACE in a meta-analysis of randomized trials where compliance may be heterogeneous between studies. It is also common that noncompliance data are not available for some trials. Simply excluding trials with incomplete noncompliance data from a meta-analysis can be inefficient and potentially biased. J. Zhou, Hodges, and Chu (2021) proposed an improved flexible Bayesian hierarchical CACE framework to account simultaneously for heterogeneous noncompliance and incomplete noncompliance data. More recently, T. Zhou et al. (2021) used a generalized linear latent and mixed model to estimate CACE, which accounts for between-study heterogeneity with random effects. The package **BayesCACE** focuses on providing user-friendly functions to estimate CACE in either a single study or meta-analysis using models based on J. Zhou et al. (2019), Baker (2020), J. Zhou, Hodges, and Chu (2020) and J. Zhou, Hodges, and Chu (2021).

This article introduces the R package **BayesCACE**, which performs CACE analysis for binary outcomes in a single study, and meta-analysis with either complete or incomplete noncompliance information. The package **BayesCACE** is available from the Comprehensive R Archive Network (CRAN). It uses Markov chain Monte Carlo (MCMC) methods on the R platform through JAGS. JAGS is a program for analyzing Bayesian hierarchical models using MCMC simulation, which is available for diverse computer platforms including Windows and Mac OS X. Convergence of the MCMC routine can be assessed by the function outputs. The package also provides functions to generate posterior trace plots, density plots, and auto-correlation plots. For meta-analysis, the package provides a forest plot of study-specific CACE estimates with 95% credible intervals as well as the overall CACE estimate, to visually display the causal treatment effect comparisons.

This article is organized as follows. The next section defines CACE in mathematical notation that will be used throughout the paper. We also describe the assumptions needed to make the CACE a valid causal effect estimator. Following that, we present an overview of the Bayesian hierarchical models for CACE implemented in the **BayesCACE** package. Then, we illustrate use of the package with a case study example and discuss the output structures. Finally, we provide a brief discussion with potential future improvements.

Assumptions and definition of CACE

The CACE is a measure of the causal effect of a treatment or intervention on patients who received it as intended by the original group allocation. It is an unbiased causal effect estimate based on five standard assumptions commonly used in causal inference research. First, it assumes that potential outcomes for each participant are independent of the potential outcomes for other participants, known as the *Stable Unit Treatment Value Assumption (SUTVA)*. Second, it assumes that assignment to treatment is random, so that the proportion of compliers should be the same in the intervention and control groups, thus allowing the estimation of one of the core unobserved parameters needed to derive a CACE estimate. Third, it assumes that treatment assignment has an effect on the outcome only if it changes the actual treatment taken, an assumption known as *exclusion restriction*. For never-takers, for instance, it assumes that simply being assigned to treatment does not affect their outcomes, as they do not actually receive the treatment assigned to them. Fourth, it assumes that assigning the study treatment to participants in the intervention group induces at least some participants to receive the treatment, so the compliance rate is not zero. Finally, it assumes that there is a monotonic relationship between treatment assignment and treatment receipt, which implies that there are no individuals for whom assignment to treatment actually reduces the likelihood of receiving treatment (i.e., no defiers). This assumption reduces the number of compliance types for which estimates are derived, permitting a properly identified model.

We follow J. Zhou et al. (2019) and introduce notation both on the individual level and on the study level. Suppose a meta-analysis reviews I two-armed RCTs, and N_i is the number of subjects in the i-th trial for $i \in \{1, \ldots, I\}$. If the data include a single study only, then $I = 1$ and we can remove the subscript i from all notation.

On the individual level, notation is defined as follows for subject j in trial i.

1. Let $R_{ij} = r$ index the randomization assignment with $r = 0$ for those randomized to control and $r = 1$ for those randomized to the intervention.
2. Let $T^r_{ij} = t \in \{0, 1\}$ be the indicator of whether the individual received the intervention. This is a *potential* outcome under the randomization assignment $r \in \{0, 1\}$, i.e., what the value of treatment t would be for individual (i, j) if $r = 0$ or $r = 1$, respectively.
3. Let $Y^{r,t}_{ij} = o \in \{0, 1\}$ be the *potential* binary outcome under randomization assignment r and treatment received t. Note that the *exclusion restriction* assumption allows us to define $Y^t_{ij} \equiv Y^{r,t}_{ij}$.
4. The sets of $\{Y^{r,t}_{ij}\}$ and $\{T^r_{ij}\}$ are the *potential* outcome and treatment-received status respectively under possible r and t, but for each subject in a trial, only one of the possible values of each set can be observed. Therefore, we denote the observed response and received treatment variables as Y_{ij} and T_{ij}.
5. We allow $T_{ij} = *$ if the actual received treatment is not recorded. Then let $M_{ij} = m$ be the missing indicator corresponding to whether subject j has actual treatment received status on record ($m = 0$) or missing ($m = 1$).
6. Using these potential outcomes, we can define the compliers and the CACE. Let C_{ij} be the latent compliance class of individual j in trial i, defined as follows:

$$C_{ij} = \begin{cases} 0, & \text{for never-taker with } (T^0_{ij}, T^1_{ij}) = (0, 0) \\ 1, & \text{for complier with } (T^0_{ij}, T^1_{ij}) = (0, 1) \\ 2, & \text{for always-taker with } (T^0_{ij}, T^1_{ij}) = (1, 1) \\ 3, & \text{for defier with } (T^0_{ij}, T^1_{ij}) = (1, 0) \end{cases}.$$

A subject's compliance status C_{ij} is not observable because in a two-arm trial, only one of T^1_{ij} and T^0_{ij} can be observed. Based on the observed randomization group and actual treatment received, the compliance classes can be only partially identified.

Now, the complier average causal effect of the i-th trial is the average difference between potential outcomes for compliers. In this case, the CACE in study i is $\theta^{\text{CACE}}_i = E(Y^1_{ij} - Y^0_{ij} | C_{ij} = 1)$, where the patients for whom $C_{ij} = 1$ are the compliers.

On the study level, n_{irto} denotes the observed number of individuals in study i, randomization group r, actual received treatment group t, and outcome o. If the compliance status of individual j in trial i is not on record, $T_{ij} = t = *$ so the corresponding count is n_{ir*o}, which is the sum of the two unobserved counts n_{ir0o} and n_{ir1o}.

2 Estimating CACE

This section briefly describes the Bayesian hierarchical models used to estimate CACE. These models form the basis of the framework proposed by J. Zhou et al. (2019) and underlie the **BayesCACE** package. In addition to the notation defined in the previous section, we define the following parameters for study i.

1. Let π_{ia} and π_{in} be the probabilities of being an always-taker and a never-taker, respectively. Because defiers are ruled out by the monotonicity assumption, each trial has at most only three compliance classes. Thus the probability of being a complier in study i is $\pi_{ic} = 1 - \pi_{ia} - \pi_{in}$.
2. Define these response probabilities: u_{i1} for a complier randomized to the treatment group; v_{i1} for a complier randomized to the control/placebo group; s_{i1} for a never-taker; and b_{i1} for an always-taker. Thus for study i, the parameters included in the model are $\boldsymbol{\beta}_i = (\pi_{ia}, \pi_{in}, u_{i1}, v_{i1}, s_{i1}, b_{i1})$.

As the outcome is binary, the expected difference between outcomes from the two treatment groups among compliers is just the risk difference between u_{i1} and v_{i1}. Therefore, the CACE can be written as $\theta^{\text{CACE}}_i = E(Y^1_{ij} - Y^0_{ij} | C_{ij} = 1) = u_{i1} - v_{i1}$.

CACE for a single trial with noncompliance

Consider first a single trial with noncompliance, i.e., $I = 1$, so all notation and parameters defined earlier are reduced to the version without subscript i. According to J. Zhou et al. (2019), each observed n_{rto} has a corresponding probability that can be written in terms of parameters defined in $\boldsymbol{\beta} = (\pi_a, \pi_n, u_1, v_1, s_1, b_1)$, thus the vector $(n_{000}, n_{001}, n_{010}, n_{011}, n_{100}, n_{101}, n_{110}, n_{111})$ follows a multinomial distribution. The likelihood is available in the Supplemental Materials.
The CACE for a single study is $u_1 - v_1$, so the posterior of θ^{CACE} is the posterior of $u_1 - v_1$.

CACE for a meta-analysis with complete compliance information

This section introduces two methods for performing a meta-analysis of the CACE when noncompliance data are reported in each trial.

The two-step approach

As described in the previous section, using the observed data n_{irto}, θ_i^{CACE} is identified for study~i. Therefore, to estimate the population-average CACE in a meta-analysis, we propose combining the study-specific estimates and standard errors using a standard meta-analysis method such as the fixed-effect (Laird and Mosteller 1990) or random-effects model (Hedges and Vevea 1998; Hedges and Olkin 1985). We call this a "two-step" approach. As the CACE measure is a risk difference, a transformation may be necessary to ensure that the normal distribution assumption is approximately true. Building upon the well-developed R package **metafor**, various estimators suggested in the literature can be estimated to account for potential between-study heterogeneity in the CACE, e.g., the Hunter–Schmidt estimator, the Hedges estimator, the DerSimonian–Laird estimator, the maximum-likelihood or restricted maximum-likelihood estimator, or the empirical Bayes estimator (Viechtbauer 2010).

The Bayesian hierarchical model

In a meta-analysis, the CACE can also be estimated using the joint likelihood from the Bayesian hierarchical model. This method is systematically introduced in J. Zhou et al. (2019). The log likelihood contribution of trial i is denoted by adding a subscript i to each parameter. Then the log likelihood for all trials in the meta-analysis is $\log \mathcal{L}(\boldsymbol{\beta}) = \sum_i \log L_i(\boldsymbol{\beta}_i)$. Because the studies are probably not exactly identical in their eligibility criteria, measurement techniques, study quality, etc., differences in methods and sample characteristics may introduce heterogeneity to the meta-analysis. One way to model the heterogeneity is to use a random-effects model.

To guarantee the desired properties of study i's latent compliance classes and to account for possible between-study heterogeneity in the compliance class and response probabilities, we use these transformations:

1. $\pi_{in} = \frac{\exp(n_i)}{1+\exp(n_i)+\exp(a_i)}, \pi_{ia} = \frac{\exp(a_i)}{1+\exp(n_i)+\exp(a_i)}$, where $n_i = \alpha_n + \delta_{in}, a_i = \alpha_a + \delta_{ia}$, and $(\delta_{in}, \delta_{ia})^\top \sim N(0,\ \boldsymbol{\Sigma}_{ps}), \boldsymbol{\Sigma}_{ps} = \left(\begin{smallmatrix} \sigma_n^2 & \rho\sigma_n\sigma_a \\ \rho\sigma_n\sigma_a & \sigma_a^2 \end{smallmatrix}\right)$.
2. We also define random effect models on the transformed scale of each response probability $s_{i1}, b_{i1}, u_{i1}, v_{i1}$: $g(s_{i1}) = \alpha_s + \delta_{is}$, $g(b_{i1}) = \alpha_b + \delta_{ib}$, $g(u_{i1}) = \alpha_u + \delta_{iu}$, $g(v_{i1}) = \alpha_v + \delta_{iv}$, where $g(\cdot)$ is a link function such as the logit or probit, $\delta_{is} \sim N(0, \sigma_s^2)$, $\delta_{ib} \sim N(0, \sigma_b^2)$, $\delta_{iu} \sim N(0, \sigma_u^2)$, $\delta_{iv} \sim N(0, \sigma_v^2)$.

Here we allow correlation between n_i and a_i, and assign random effect variables to all parameters. However, if a parameter does not vary between trials, it can be modeled as a fixed effect. Let $f(\boldsymbol{\beta}_i|\boldsymbol{\beta}_0, \boldsymbol{\Sigma}_0)$ be the distributions described above of all parameters $\boldsymbol{\beta}_i = (\pi_{ia}, \pi_{in}, s_{i1}, b_{i1}, u_{i1}, v_{i1})$, where $\boldsymbol{\beta}_0$ is the vector of mean hyper-parameters $(\alpha_n, \alpha_a, \alpha_s, \alpha_b, \alpha_u, \alpha_v)$, and $\boldsymbol{\Sigma}_0$ is the diagonal covariance matrix containing $\boldsymbol{\Sigma}_{ps}, \sigma_s^2, \sigma_b^2, \sigma_u^2$ and σ_v^2.
If we specify $f(\boldsymbol{\beta}_0)$ and $f(\boldsymbol{\Sigma}_0)$ as the prior distributions for the hyper-parameters, then the joint posterior distribution is proportional to the likelihood multiplied by the priors, i.e., $\prod_i L_i(\boldsymbol{\beta}_i) f(\boldsymbol{\beta}_i|\boldsymbol{\beta}_0, \boldsymbol{\Sigma}_0) f(\boldsymbol{\beta}_0) f(\boldsymbol{\Sigma}_0)$.

As stated earlier, $\theta_i^{\text{CACE}} = u_{i1} - v_{i1}$ for study i, so for the meta-analysis, the overall CACE is $\theta^{\text{CACE}} = E(\theta_i^{\text{CACE}}) = E(u_{i1}) - E(v_{i1})$. When a random effect δ_{iu} or δ_{iv} is not assigned in the model, $E(u_{i1}) = g^{-1}(\alpha_u)$ and $E(v_{i1}) = g^{-1}(\alpha_v)$. Otherwise, $E(u_{i1})$ and $E(v_{i1})$ can be estimated by integrating out the random effects, e.g., $E(u_{i1}) = \int_{-\infty}^{+\infty} g^{-1}(\alpha_u + t)\sigma_u^{-1}\phi(\frac{t}{\sigma_u})dt$, where $\phi(\cdot)$ is the standard Gaussian density. If the function $g(\cdot)$ is the probit link, this expectation has a closed form:

$E(u_{i1}) = \Phi(\frac{\alpha_u}{\sqrt{1+\sigma_u^2}})$. If the link function $g(\cdot)$ is logit, a well-established approximation $E(u_{i1}) \approx \text{logit}^{-1}(\frac{\alpha_u}{\sqrt{1+C^2\sigma_u^2}})$ can be used, where $C = \frac{16\sqrt{3}}{15\pi}$ (Zeger, Liang, and Albert 1988). The above formulas also apply to $E(v_{i1})$, the expected response rate of a complier in the control group.

The two-step approach, stated by Lin and Zeng (2010), can be viewed as asymptotically equivalent to the model using the joint likelihood. However, as the two-step approach requires the whole set of parameters to be estimated independently for each study, the total number of effective parameters tends to be larger than the Bayesian hierarchical model, so estimates using our method are likely to be more efficient.

CACE for meta-analysis with incomplete compliance information

Another advantage of the Bayesian hierarchical model is that it can include trials with incomplete compliance data. Commonly, some trials do not report noncompliance data because study investigators do not collect actual received treatment status for some subjects or simply do not report compliance. The two-step approach needs counts for all of the groups defined by randomized assignment, treatment received, and outcome in order to estimate the study-specific θ_i^{CACE}. Thus, by using this method, trials with incomplete compliance data are simply excluded, making estimation less efficient and potentially biased.

J. Zhou, Hodges, and Chu (2021) proposed a comprehensive framework to incorporate both heterogeneous and incomplete noncompliance data for estimating the CACE in a meta-analysis of RCTs. Here we present the data structure needed for binary outcomes. For study i, randomization group $r \in \{0,1\}$ and output $o \in \{0,1\}$, if the compliance information is reported, then values of n_{ir0o} and n_{ir1o} are reported, so we assign the marginal count $n_{ir*o} = 0$. Otherwise, we do not have data on outcomes for groups defined by actually received treatment, so only the marginal n_{ir*o} is observed, where n_{ir*o} is the number of patients randomized to treatment arm r who had outcome o. In this situation, the two unobserved counts n_{ir0o} and n_{ir1o} are assigned as 0. In the Supplemental Materials, a table for the observed counts data with corresponding probabilities is presented. The log likelihood is also obtained from the multinomial distribution. The CACE for this meta-analysis incorporating incomplete compliance data is $\theta^{\text{CACE}} = E(\theta_i^{\text{CACE}}) = E(u_{i1}) - E(v_{i1}) = \Phi(\frac{\alpha_u}{\sqrt{1+\sigma_u^2}}) - \Phi(\frac{\alpha_v}{\sqrt{1+\sigma_v^2}})$ if the probit link function is used for u_{i1} and v_{i1}.

3 Using the R package BayesCACE

The primary objective of the **BayesCACE** package is to provide a user-friendly implementation of the Bayesian method for estimating the CACE. The package is now available to download and install via CRAN at `https://CRAN.R-project.org/package=BayesCACE`. It can be installed within R using the command `install.packages("BayesCACE")`. The latest version of the package is 1.2.3.

The **BayesCACE** package depends on the R packages **rjags** (Plummer 2018), **coda** (Plummer et al. 2006), and **forestplot** (Gordon and Lumley 2017). Users need to install JAGS separately from its homepage `http://mcmc-jags.sourceforge.net` as the **BayesCACE** package does not include a copy of the JAGS library. The current version of JAGS is 4.3.0, which is the version of the package that **BayesCACE** requires; earlier versions of JAGS may not guarantee exactly reproducible results.

Data structure for estimating the CACE

We introduce the data structures through the illustrative example included in the package **BayesCACE**: `epidural_c` and `epidural_ic`. These two data sets were obtained from Bannister-Tyrrell et al. (2015), who conducted an exploratory meta-analysis of the association between using epidural analgesia in labor and the risk of cesarean section. The dataset `epidural_c` contains 10 trials with full compliance information; each trial has 8 observed counts, denoted by n_{irto} and presented in columns `nirto` for $i = 1, \ldots, 10$ and $r, t, o \in \{0,1\}$. These data were re-analyzed by J. Zhou et al. (2019) in a meta-analysis using their proposed Bayesian hierarchical model to estimate the CACE. The function `cace.meta.c()` performs this analysis. The column `study.id` contains IDs for the 10 studies, and `study.name` labels each study by its first author's surname and its publication year.

The data can be loaded and printed using these commands:

```
library("BayesCACE")
data("epidural_c", package = "BayesCACE")
epidural_c
```

```
#>    study.id     study.name n000 n001 n010 n011 n100 n101 n110 n111
#> 1         1   Bofill, 1997   37    2   11    1    2    0   42    5
#> 2         2    Clark, 1998   72    6   68   16    7    2  134   13
#> 3         3  Halpern, 2004   62    5   44    7    0    0  112   12
#> 4         4     Head, 2002   51    7    2    0    3    0   43   10
#> 5         5     Jain, 2003   72   11    0    0    0    2   36    7
#> 6         6   Nafisi, 2006  179   19    0    0    0    0  173   24
#> 7         7  Nikkola, 1997    6    0    4    0    0    0   10    0
#> 8         8    Ramin, 1995  546   17   95    8  230    2  393   39
#> 9         9   Sharma, 1997  336   16    5    0  114    1  231   12
#> 10       10 Volmanen, 2008   23    1    3    0    1    0   23    1
```

The other dataset epidural_ic represents the situation in which not all trials report complete compliance data. It contains 27 studies, only 10 of which have full compliance information and are included in epidural_c. This dataset is also drawn from Bannister-Tyrrell et al. (2015), and represents studies with incomplete compliance information when estimating the CACE. The function cace.meta.ic() performs this analysis.

Each study is represented by one row in the dataset; the columns study.id and study.name have the same meanings as in the dataset epidural_c. Each study's data are summarized in 12 numbers (columns) denoted by n_{irto} and n_{ir*o}. For a particular randomization group $r \in \{0, 1\}$, the observed counts are presented either as n_{irto} or n_{ir*o} depending on whether the compliance information is available; values for other columns are denoted by 0. The corresponding column names in the dataset are nirto and nirso, respectively.

The first 6 rows of the dataset epidural_ic are printed below.

```
data("epidural_ic", package = "BayesCACE")
head(epidural_ic)
```

```
#>   study.id       study.name n000 n001 n010 n011 n0s0 n0s1 n100 n101 n110 n111
#> 1        1     Bofill, 1997   37    2   11    1    0    0    2    0   42    5
#> 2        2      Clark, 1998   72    6   68   16    0    0    7    2  134   13
#> 3        3  Dickinson, 2002    0    0    0    0  428   71    0    0    0    0
#> 4        4      Evron, 2008   40    4    0    0    0    0    0    0    0    0
#> 5        5 El Kerdawy, 2010    0    0    0    0   12    3    0    0    0    0
#> 6        6   Gambling, 1998    0    0    0    0  573   34  206   10  371   29
#>   n1s0 n1s1
#> 1    0    0
#> 2    0    0
#> 3  408   85
#> 4  129   19
#> 5   11    4
#> 6    0    0
```

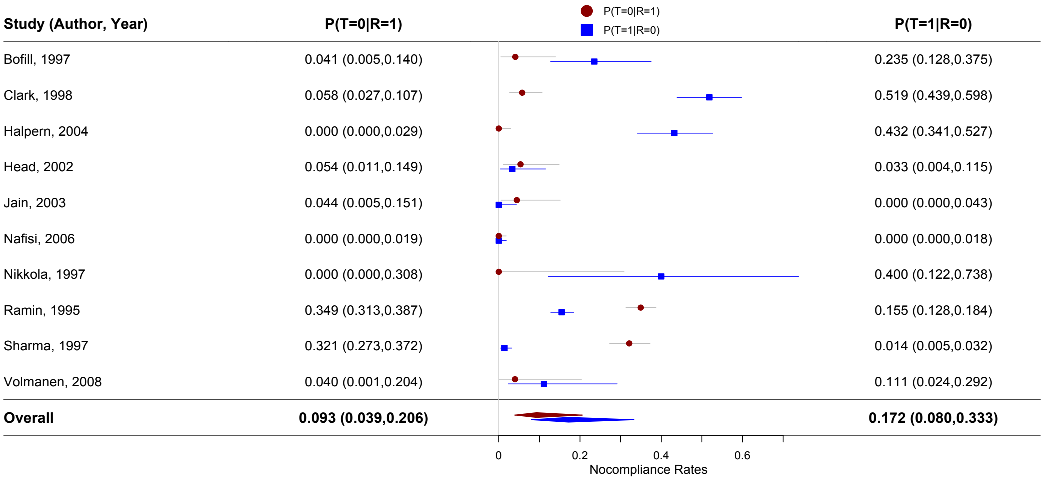
Note that NA is not allowed in a dataset for the package BayesCACE, but some trials may have 0 events or 0 noncompliance rates.

Plotting noncompliance rates

Before performing the CACE analysis, one might want a visual overview of study-specific noncompliance rates in both randomization arms. The function plt.noncomp provides a forest plot of noncompliance rates in an R plot window. The function can be simply called as

```
plt.noncomp(data, overall = TRUE)
```

where data is a dataset with structure like epidural_c or epidural_ic. Specifically, the dataset should contain the following columns: study.id, study.name, and 8 or 12 columns of data represented by n_{irto}, or n_{irto} and n_{ir*o} (see previous section for more details). Each row corresponds to one study. Only studies with full compliance information are included in this plot because noncompliance rates cannot be calculated without compliance data. Figure 1 shows the resulting plot, where the red dot with its horizontal line shows the study-specific noncompliance rate with its 95% exact confidence interval for the patients randomized to the treatment arm, and the blue square with its horizontal line represents that rate and interval for those in the control arm. The confidence intervals are calculated



Study (Author, Year)	P(T=0|R=1)	P(T=1|R=0)
Bofill, 1997	0.041 (0.005,0.140)	0.235 (0.128,0.375)
Clark, 1998	0.058 (0.027,0.107)	0.519 (0.439,0.598)
Halpern, 2004	0.000 (0.000,0.029)	0.432 (0.341,0.527)
Head, 2002	0.054 (0.011,0.149)	0.033 (0.004,0.115)
Jain, 2003	0.044 (0.005,0.151)	0.000 (0.000,0.043)
Nafisi, 2006	0.000 (0.000,0.019)	0.000 (0.000,0.018)
Nikkola, 1997	0.000 (0.000,0.308)	0.400 (0.122,0.738)
Ramin, 1995	0.349 (0.313,0.387)	0.155 (0.128,0.184)
Sharma, 1997	0.321 (0.273,0.372)	0.014 (0.005,0.032)
Volmanen, 2008	0.040 (0.001,0.204)	0.111 (0.024,0.292)
Overall	**0.093 (0.039,0.206)**	**0.172 (0.080,0.333)**

Figure 1: Noncompliance rates plot generated by the function plt.noncomp(). The red dots and lines show the study-specific noncompliance rate with its 95% confidence interval randomized to the treatment arm, and the blue squares and lines refer to those in the control arm.

by the Clopper–Pearson exact method (Clopper and Pearson 1934), which is based on the cumulative distribution function of the binomial distribution. Using the default `overall = TRUE`, the figure also gives a summary estimate of the compliance rates per randomization group. This overall rate is estimated using a logit generalized linear mixed model. Otherwise, if the argument `overall` is `FALSE`, the plot shows only study-specific noncompliance rates. Any additional parameters passed to the function will be automatically used in the `forestplot` function from the **forestplot** package.

CACE analysis for a single study or in a meta-analysis

The major functions in **BayesCACE** are `cace.study()`, `cace.meta.c()`, and `cace.meta.ic()`, which implement the models introduced earlier to perform Bayesian CACE analysis for different data structures. In particular, `cace.study()` performs CACE analysis for a single study. The function `cace.meta.c()` performs CACE analysis for a meta-analysis when each trial reports noncompliance information. Users can choose to do the analysis either by the two-step approach or using the Bayesian hierarchical model. When some trials do not report noncompliance data, the function `cace.meta.ic()` can be applied to perform a CACE meta-analysis using the likelihood provided in the Supplemental Materials. Each function may take 1–15 minutes to run. Generally the two-step approach using the function `cace.meta.c()` takes longer because MCMC chains are run on the studies one by one. The actual run time depends on the amount of data and the user's processor.

Function `cace.study()` for a study-specific analysis or a two-step meta-analysis

For the default interface, the arguments of the function `cace.study()` are

```
cace.study(data, param = c("CACE", "u1", "v1", "s1", "b1", "pi.c", "pi.n",
  "pi.a"), re.values = list(), model.code = '', digits = 3, n.adapt = 1000,
  n.iter = 100000, n.burnin = floor(n.iter/2), n.chains = 3, n.thin =
  max(1,floor((n.iter-n.burnin)/1e+05)), conv.diag = FALSE, mcmc.samples =
  FALSE, two.step = FALSE, method = "REML")
```

where users need to input `data` with the same structure as `epidural_c`, containing either one row of observations for a single study, or multiple rows referring to multiple studies in a meta-analysis. This function fits a model for a single study. If the data includes more than one study, the study-specific CACEs will be estimated by retrieving data row by row.

The argument `param` is a character string vector indicating the parameters to be tracked and estimated. By default all parameters are included: θ^{CACE} (`CACE`), u_1 (`u1`), v_1 (`v1`), s_1 (`s1`), b_1 (`b1`), π_a (`pi.a`), π_n (`pi.n`), and $\pi_c = 1 - \pi_a - \pi_n$ (`pi.c`). Users can modify the string vector to only include parameters of interest besides θ^{CACE}. Users can specify the prior distributions (mean and standard deviation) of $n, a, \alpha_s, \alpha_b, \alpha_u, \alpha_v$ with the `re.values` parameter. By default, the `re.values` list is empty, and they are assigned to the transformed scale of the following parameters: $\pi_n = \frac{\exp(n)}{1+\exp(n)+\exp(a)}$, $\pi_a = \frac{\exp(a)}{1+\exp(n)+\exp(a)}$, $\text{logit}(s_1) = \alpha_s$, $\text{logit}(b_1) = \alpha_b$, $\text{probit}(u_1) = \alpha_u$, and $\text{probit}(v_1) = \alpha_v$, where $n, a \sim N(0, 2.5^2)$ and $\alpha_s, \alpha_b, \alpha_u, \alpha_v \sim N(0, 2^2)$. With these settings, a 95% prior probability interval for any of the probabilities π_{in}, π_{ia}, and π_{ic} ranges from about 0.001 to 0.91, and a 95% prior interval for the probabilities s_1, b_1, u_1, and v_1 ranges approximately from 0.01 to 0.98. The prior parameters are passed into the `model.study` function to get the model code, which first calls the `prior.study` to get the custom prior distribution. Here we give an example output of `prior.study` when assigning $N(0, 10^{-2})$ to every parameter:

```
  out.string <-
    "# priors
    n ~ dnorm(0, 0.01)
    a ~ dnorm(0, 0.01)
    alpha.s ~ dnorm(0, 0.01)
    alpha.b ~ dnorm(0, 0.01)
    alpha.u ~ dnorm(0, 0.01)
    alpha.v ~ dnorm(0, 0.01)
    "
```

To customize the model fully, the user can pass their complete model string to the `cace.study()` function with the parameter `model.code`. The arguments `n.adapt`, `n.iter`, `n.burnin`, `n.chains`, and `n.thin` control the MCMC algorithm run by the R package **rjags** (Plummer 2018). The argument `n.adapt` is the number of iterations for adaptation; it is used to maximize the sampling efficiency, and the default is set as 1,000. The argument `n.chains` determines the number of MCMC chains (the default is 3); `n.iter` is the number of iterations of each MCMC chain; `n.burnin` is the number of burn-in iterations to be discarded at the beginning of each chain; `n.thin` is the thinning rate for MCMC chains, which is used to avoid potential high auto-correlation and to save computer memory when `n.iter` is large. The default of `n.thin` is set as `1` or the largest integer not greater than `((n.iter - n.burnin)/1e+05))`, whichever is larger. The argument `conv.diag` specifies whether to compute the Gelman and Rubin convergence statistic ($\hat{R}$) of each parameter as a convergence diagnostic (Brooks and Gelman 1998; Gelman and Rubin 1992). The chains are considered well-mixed and converged to the target distribution if $\hat{R} \leq 1.1$. If the argument `mcmc.samples = TRUE`, the function saves each chain's MCMC samples for all parameters, which can be used to produce trace, posterior density, and auto-correlation plots by calling the functions `plt.trace`, `plt.density`, and `plt.acf`.

By default, the function `cace.study()` returns a list including posterior estimates (posterior mean, standard deviation, median, and a 95% credible interval with 2.5% and 97.5% quantiles as the lower and upper bounds), and the deviance information criterion (DIC) statistic (Spiegelhalter et al. 2002) for each study. The argument `two.step` is a logical value indicating whether to conduct a two-step meta-analysis. If `two.step = TRUE`, the posterior mean and standard deviation of study-specific θ_i^{CACE} are used to perform a standard meta-analysis, using the R package **metafor**. The default estimation method is the REML (restricted maximum-likelihood estimator) method for the random-effects model (Harville 1977). Users can change the argument `method` to obtain different meta-analysis estimators from either a random-effects model or a fixed-effect model, e.g., `method = "DL"` refers to the DerSimonian–Laird estimator, `method = "HE"` returns the Hedges estimator, and `method = "HS"` gives the Hunter–Schmidt estimator. More details are available from the documentation of the function `metafor::rma` (Viechtbauer 2010). If the input data include only one study, the meta-analysis result is the same as the result from the single study.

Here is an example to demonstrate the function's usage. We call the function `cace.study()` on the dataset `epidural_c` as follows:

```
data("epidural_c", package = "BayesCACE")
set.seed(123)
out.study <- cace.study(data = epidural_c, conv.diag = TRUE,
                        mcmc.samples = TRUE, two.step = TRUE)
```

The following messages are output as the code runs:

```
% NA is not allowed in the input data set;
% the rows containing NA are removed.
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
```

```
Graph information:
   Observed stochastic nodes: 2
   Unobserved stochastic nodes: 6
   Total graph size: 44

Initializing model

  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100%
  |**************************************************| 100%
  |**************************************************| 100%
MCMC convergence diagnostic statistics are calculated and saved in conv.out
```

If the dataset contains more than one study, e.g., the `epidural_c` dataset has 10 trials, then once the JAGS model compiles for the first study, it automatically continues to run on the next study's data. The results are saved in the object `out.study`, a list containing the model name, posterior information for each monitored parameter, and DIC of each study. We can use parameter names to display the corresponding estimates. The argument `digits` in the function `cace.study()` can be used to change the number of significant digits to the right of the decimal point. Here, we used the default setting `digits = 3`. For example, the estimates of θ^{CACE} for each single study (posterior mean and standard deviation, posterior median, 95% credible interval, and time-series standard error) can be displayed as:

```
out.study$CACE
```

```
#>           Mean     SD     2.5%       50%  97.5% Naive SE Time-series SE
#>  [1,]  0.04980 0.0797 -0.09510  4.46e-02 0.2180 1.45e-04       2.51e-04
#>  [2,] -0.02490 0.0489 -0.12200 -2.23e-02 0.0785 8.94e-05       1.48e-04
#>  [3,] -0.02210 0.0606 -0.12700 -2.90e-02 0.1120 1.11e-04       1.94e-04
#>  [4,]  0.07180 0.0758 -0.07550  7.10e-02 0.2230 1.38e-04       2.01e-04
#>  [5,]  0.08250 0.0768 -0.06260  8.11e-02 0.2370 1.40e-04       2.51e-04
#>  [6,]  0.02600 0.0319 -0.03650  2.59e-02 0.0891 5.83e-05       7.55e-05
#>  [7,]  0.01430 0.1580 -0.28200  2.11e-04 0.4050 2.89e-04       4.21e-04
#>  [8,]  0.05030 0.0248  0.00176  5.02e-02 0.0993 4.54e-05       7.34e-05
#>  [9,] -0.01100 0.0234 -0.05740 -1.09e-02 0.0350 4.27e-05       6.22e-05
#> [10,]  0.00145 0.0655 -0.13400 -4.36e-06 0.1460 1.20e-04       1.56e-04
```

If the argument `conv.diag` is specified as `TRUE`, the output list contains a sub-list `conv.out`, which outputs the point estimates of the "potential scale reduction factor" (the Gelman and Rubin convergence statistic, labeled `Point est.`) calculated for each parameter from each single study, and their upper confidence limits (labeled `Upper C.I.`). Approximate convergence is diagnosed when the upper limit is close to 1 (Brooks and Gelman 1998; Gelman and Rubin 1992). For example, the first sub-list from `conv.out` is:

```
out.study$conv.out[[1]]
```

```
#>      Point est. Upper C.I.
#> CACE   1.000025   1.000046
#> b1     1.000041   1.000129
#> pi.a   1.000025   1.000094
#> pi.c   1.000036   1.000134
#> pi.n   1.000029   1.000067
#> s1     1.000014   1.000018
#> u1     1.000016   1.000033
#> v1     1.000077   1.000185
```

In this example, we included `mcmc.samples = TRUE` in the argument, so the output list `out.study` includes each chain's MCMC samples for all parameters. They can be used with our plotting functions to generate the trace, posterior density, and auto-correlation plots for further model diagnostics.

If the dataset used by the function `cace.study()` has more than one study, specifying the argument `two.step = TRUE` causes the two-step meta-analysis for θ^{CACE} to be done. The outcomes are saved as a sub-list object `meta`. Note that users can obtain different meta-analysis estimators by changing the `method` argument as described earlier.

```
out.study$meta
```

```
#>
#> Random-Effects Model (k = 10; tau^2 estimator: REML)
#>
#> tau^2 (estimated amount of total heterogeneity): 0.0002 (SE = 0.0008)
#> tau (square root of estimated tau^2 value):      0.0131
#> I^2 (total heterogeneity / total variability):   8.10%
#> H^2 (total variability / sampling variability):  1.09
#>
#> Test for Heterogeneity:
#> Q(df = 9) = 5.9353, p-val = 0.7464
#>
#> Model Results:
#>
#> estimate      se    zval    pval    ci.lb   ci.ub
#>   0.0182  0.0143  1.2758  0.2020  -0.0098  0.0462
#>
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Function `cace.meta.c()` for meta-analysis with complete compliance data

The function `cace.meta.c()` performs the Bayesian hierarchical model method for meta-analysis when the dataset has complete compliance information for all studies. The function's default arguments are as shown:

```
cace.meta.c(data, param = c("CACE", "u1out", "v1out", "s1out", "b1out",
  "pic", "pin", "pia"), random.effects = list(), re.values = list(),
  model.code = '', digits = 3, n.adapt = 1000, n.iter = 100000,
  n.burnin = floor(n.iter/2), n.chains = 3, n.thin =
  max(1,floor((n.iter-n.burnin)/100000)), conv.diag = FALSE,
  mcmc.samples = FALSE, study.specific = FALSE)
```

The arguments controlling the MCMC algorithm are mostly similar to those of `cace.study()`. One major difference is that users need to specify parameters that are modeled as random effects. Earlier, we showed how to specify random effects for each parameter on the transformed scales, namely δ_{in}, δ_{ia}, δ_{iu}, δ_{iv}, δ_{is}, and δ_{ib}, and allowed a non-zero correlation ρ between δ_{in} and δ_{ia}. The model with all of these random effects as well as the correlation ρ is considered the full model. However, this function is flexible, allowing users to choose which random effects to include by specifying the `random.effects` argument. By default, the list is empty and all of the list values are set to `TRUE`. Users can customize that by setting `delta.n`, `delta.a`, `delta.u`, `delta.v`, `delta.s`, `delta.b`, and/or `cor` to `FALSE`. Note that ρ (`cor`) can only be included when both δ_{in} (`delta.n`) and δ_{ia} (`delta.a`) are set to `TRUE`. Otherwise, a warning is shown and the model continues running by forcing `delta.n = TRUE` and `delta.a = TRUE`. The default parameters to be monitored depend on which parameters are modeled as random effects. For example, `u1out` refers to $E(u_{i1})$, where for the probit link, $E(u_{i1}) = \Phi(\alpha_u)$ if δ_u is not specified in the model, and $E(u_{i1}) = \Phi(\frac{\alpha_u}{\sqrt{1+\sigma_u^2}})$ when the random effect δ_u is included.

Users can use the `re.values` parameter to customize the prior distribution. Like the function `cace.study()`, by default, weakly informative priors $\alpha_n, \alpha_a \sim N(0, 2.5^2)$ and α_s, α_b, α_u, $\alpha_v \sim N(0, 2^2)$ are assigned to the means of these transformed parameters: $\pi_{in} = \frac{\exp(n_i)}{1+\exp(n_i)+\exp(a_i)}$, $\pi_{ia} = \frac{\exp(a_i)}{1+\exp(n_i)+\exp(a_i)}$, where $n_i = \alpha_n + \delta_{in}$, $a_i = \alpha_a + \delta_{ia}$, $\text{logit}(s_{i1}) = \alpha_s + \delta_{is}$, $\text{logit}(b_{i1}) = \alpha_b + \delta_{ib}$, $\text{probit}(u_{i1}) = \alpha_u + \delta_{iu}$, and $\text{probit}(v_{i1}) = \alpha_v + \delta_{iv}$. For the random effects, we have $\delta_{is} \sim N(0, \sigma_s^2)$, $\delta_{ib} \sim N(0, \sigma_b^2)$, $\delta_{iu} \sim N(0, \sigma_u^2)$, and $\delta_{iv} \sim N(0, \sigma_v^2)$, as response rates are assumed to be independent between latent classes. A $Gamma(2, 2)$ hyper-prior distribution is assigned to the precision parameters σ_s^{-2}, σ_b^{-2}, σ_u^{-2} and σ_v^{-2}, which corresponds to a 95% interval of $(0.6, 2.9)$ for the corresponding standard deviations, allowing moderate heterogeneity in the response rates. In a reduced model with one of δ_{in} or δ_{ia} set to 0, the prior of the other precision parameter is also assumed to be $Gamma(2, 2)$, which gives moderate heterogeneity for latent compliance class probabilities, whereas for the full model, $(\delta_{in}, \delta_{ia})^\top \sim N(0,\ \mathbf{\Sigma}_{ps})$, the prior for the variance-covariance matrix $\mathbf{\Sigma}_{ps}$ is $InvWishart(\mathbf{I}, 3)$, where $\mathbf{I}$ is the identity matrix.

Similar to `cace.study()`, to customize the model fully, the user can pass their complete model string with the parameter `model.code`. Because the function `cace.meta.c()` is more complicated depending on the choice of random effects, we show an example of the customized prior distributions when assigning delta.n = TRUE, delta.a = TRUE, delta.u = TRUE, delta.v = FALSE, delta.s = TRUE, delta.b = TRUE, and cor = TRUE while keeping default values for re.values.

```
string <-
"# priors
alpha.n ~  dnorm(0, 0.16)
alpha.a ~ dnorm(0, 0.16)
alpha.s ~  dnorm(0, 0.25)
alpha.b ~  dnorm(0, 0.25)
alpha.u ~  dnorm(0, 0.25)
alpha.v ~  dnorm(0, 0.25)

II[1,1] <- 1
II[2,2] <- 1
II[1,2] <- 0
II[2,1] <- 0

Omega.rho ~  dwish (II[,], 3)
Sigma.rho <- inverse(Omega.rho)
sigma.n <- Sigma.rho[1, 1]
sigma.a <- Sigma.rho[2, 2]
rho <- Sigma.rho[1, 2]
u1out <- phi(alpha.u/sqrt(1+sigma.u^2))
tau.u ~ dgamma(2, 2)
sigma.u <- 1/sqrt(tau.u)
v1out <- phi(alpha.v)
CACE <- u1out-v1out
s1out <- ilogit(alpha.s/sqrt(1 + (16^2*3/(15^2*pi^2))*sigma.s^2))
tau.s ~ dgamma(2, 2)
sigma.s <- 1/sqrt(tau.s)
b1out <- ilogit(alpha.b/sqrt(1 + (16^2*3/(15^2*pi^2))*sigma.b^2))
tau.b ~ dgamma(2, 2)
sigma.b <- 1/sqrt(tau.b)
"
```

The epidural_c dataset is used as a real-study example:

```
data("epidural_c", package = "BayesCACE")
set.seed(123)
out.meta.c <- cace.meta.c(data = epidural_c, conv.diag = TRUE,
                          mcmc.samples = TRUE, study.specific = TRUE)
```

The usage of arguments conv.diag and mcmc.samples is the same as for the function cace.study. When the argument study.specific is specified as TRUE, the model will first check the logical status of arguments delta.u and delta.v. If both are FALSE, meaning that neither response rate u_{i1} or v_{i1} is modeled with a random effect, then the study-specific θ_i^{CACE} is the same across studies. The function gives a warning and continues by making study.specific = FALSE. Otherwise, the study-specific θ_i^{CACE} are estimated and saved as the parameter cacei.

In this example, by calling the object smry from the output list out.meta.c, posterior estimates (posterior mean, standard deviation, posterior median, 95% credible interval, and time-series standard error) are displayed.

```
out.meta.c$smry
```

```
#>                Mean     SD     2.5%       50%  97.5% Naive SE Time-series SE
#> CACE       0.020200 0.0627 -0.10200  0.018900 0.1490 1.14e-04       7.69e-04
#> b1out      0.128000 0.0459  0.05970  0.121000 0.2370 8.39e-05       4.07e-04
#> cacei[1]   0.043900 0.0679 -0.08130  0.040700 0.1870 1.24e-04       2.35e-04
#> cacei[2]  -0.023100 0.0489 -0.11500 -0.025000 0.0822 8.94e-05       1.89e-04
#> cacei[3]  -0.007630 0.0569 -0.11000 -0.011800 0.1130 1.04e-04       2.16e-04
#> cacei[4]   0.065000 0.0678 -0.06620  0.064300 0.2010 1.24e-04       1.62e-04
#> cacei[5]   0.054000 0.0686 -0.07380  0.051500 0.1960 1.25e-04       2.45e-04
#> cacei[6]   0.026300 0.0309 -0.03390  0.026200 0.0875 5.64e-05       6.87e-05
#> cacei[7]   0.002770 0.0931 -0.18900 -0.000142 0.2100 1.70e-04       3.71e-04
#> cacei[8]   0.048300 0.0239  0.00171  0.048200 0.0956 4.36e-05       6.38e-05
#> cacei[9]  -0.010600 0.0224 -0.05520 -0.010500 0.0331 4.09e-05       5.49e-05
#> cacei[10]  0.000228 0.0600 -0.12000 -0.001200 0.1280 1.10e-04       2.15e-04
#> pia        0.114000 0.0742  0.02460  0.098200 0.3010 1.35e-04       3.92e-03
#> pic        0.821000 0.0842  0.60500  0.838000 0.9350 1.54e-04       4.50e-03
#> pin        0.064200 0.0397  0.01540  0.056700 0.1590 7.25e-05       2.11e-03
#> s1out      0.184000 0.1040  0.04560  0.161000 0.4450 1.91e-04       9.03e-04
#> u1out      0.127000 0.0473  0.05520  0.120000 0.2390 8.64e-05       6.10e-04
#> v1out      0.107000 0.0406  0.04700  0.100000 0.2040 7.41e-05       4.55e-04
```

The posterior estimates of θ_i^{CACE} can be used to make a forest plot by calling the function `plt.forest`.

Users can manually do model selection procedures by including different random effects and comparing DIC from the outputs. DIC and its two components are saved as an object `DIC` in the output list.

```
out.meta.c$DIC
```

```
#>
#> D.bar 204.40801
#> pD     44.74788
#> DIC   249.15590
```

`DIC` is the penalized deviance, calculated as the sum of `D.bar` and `pD`, where `D.bar` is the posterior expectation of the deviance, reflecting the model fit, and `pD` reflects the effective number of parameters in the model. `D.bar` is usually lower when more parameters are included in the model, but complex models may lead to overfitting. Thus `DIC` balances the model's fit against the effective number of parameters. Generally a model with smaller DIC is preferred. However, it is difficult to conclude what constitutes an important improvement in DIC. Following Lunn et al. (2012), we suggest that a reduction of less than 5 is not a substantial improvement. When fitting models to a particular dataset, it is usually uncertain which random effect variables should be included in the model. The function `cace.meta.c()` allows users to specify candidate models with different random effects, and thus to conduct a forward/backward/stepwise model selection procedure to choose the best fitting model.

Function `cace.meta.ic()` for meta-analysis with incomplete compliance information

Another major function in the package **BayesCACE** is `cace.meta.ic()`. It also estimates θ^{CACE} using the Bayesian hierarchical model but can accommodate studies with incomplete compliance data. The arguments of this function are:

```
cace.meta.ic(data, param = c("CACE", "u1out", "v1out", "s1out", "b1out",
  "pic", "pin", "pia"), random.effects = list(), re.values = list(),
  model.code = '', digits = 3, n.adapt = 1000, n.iter = 100000,
  n.burnin = floor(n.iter/2), n.chains = 3, n.thin =
  max(1,floor((n.iter-n.burnin)/100000)), conv.diag = FALSE,
  mcmc.samples = FALSE, study.specific = FALSE)
```

The arguments of `cace.meta.ic()` are mostly similar to those of `cace.meta.c()`, although the function `cace.meta.ic()` calls a different built-in model file from the package **BayesCACE**. The major difference in using this function is that users need to create a dataset with the same structure as `epidural_ic`. As for `cace.meta.c()`, users can set their customized prior distributions. Here we use the `epidural_ic` dataset as an example:

```
data("epidural_ic", package = "BayesCACE")
set.seed(123)
out.meta.ic <- cace.meta.ic(data = epidural_ic, conv.diag = TRUE,
                            mcmc.samples = TRUE, study.specific = TRUE)
```

The results are saved in the object `out.meta.ic`, a list containing posterior estimates for monitored parameters, DIC, convergence diagnostic statistics, and MCMC samples. In this example, the argument `study.specific` is `TRUE`, so the summary for each study-specific θ_i^{CACE} is displayed in the object `out.meta.ic$smry` together with other parameters.

Note that when compiling the JAGS model, the warning “adaptation incomplete” may occasionally occur, indicating that the number of iterations for the adaptation process is not sufficient. The default value of `n.adapt` (the number of iterations for adaptation) is 1,000. This is an initial sampling phase during which the samplers adapt their behavior to maximize their efficiency (e.g., a Metropolis–Hastings random walk algorithm may change its step size) (Plummer 2018). The “adaptation incomplete” warning indicates that the MCMC algorithm may not achieve maximum efficiency, but it generally has little impact on the posterior estimates of the treatment effects. To avoid this warning, users may increase `n.adapt`.

Plotting the trace plot, posterior density, and auto-correlation

When compiling the JAGS models, it is helpful to assess the performance of the MCMC algorithm. The functions `plt.trace`, `plt.density`, and `plt.acf` provide diagnostic plots for the MCMC, namely trace plots, kernel density estimation plots, and auto-correlation plots. Both trace plots and auto-correlation plots can be used to examine whether the MCMC chains appear to be drawn from stationary distributions. A posterior density plot for a parameter visually shows the posterior distribution. Users can simply call this function on objects produced by `cace.study()`, `cace.meta.c()`, or `cace.meta.ic()`.

The arguments of this plot function are:

```
plt.trace(obj, param = c("CACE"), trialnumber = 1, ...)
plt.density(obj, param = c("CACE"), trialnumber = 1, ...)
plt.acf(obj, param = c("CACE"), trialnumber = 1, ...)
```

We use the objects list obtained from fitting the Bayesian hierarchical model `cace.meta.ic()` as an example to generate the three plots. To avoid lengthy output we just illustrate how these plots are produced for θ^{CACE}. The relevant code is:

```
plt.trace(obj = out.meta.ic)
plt.density(obj = out.meta.ic)
plt.acf(obj = out.meta.ic)
```

The produced plots are shown in Figures 2–4. The trace plots in Figure 2 show the parameter values sampled at each iteration versus the iteration number. Each chain is drawn as a separate trace plot to avoid overlay. Here we used the default `n.chains = 3`, so three trace plots are drawn. These plots show evidence that the posterior samples of θ^{CACE} are drawn from the stationary distribution.

The density plot in Figure 3 is smoothed using the R function `density()`. It shows that the kernel-smoothed posterior of θ^{CACE} is almost symmetric. The posterior mean is not far from 0, indicating that the complier average causal effect of using epidural analgesia in labor on cesarean section is likely not significant.

The auto-correlation plot in Figure 4 is a bar plot displaying the auto-correlation for different lags. At lag 0, the value of the chain has perfect auto-correlation with itself. As the lag becomes greater, the values become less correlated. After a lag of about 50, the auto-correlation drops below 0.1. If the plot shows high auto-correlation, users can run the chain longer or can choose a larger `n.thin`, e.g., `n.thin = 10` would keep only 1 out of every 10 iterations, so that the thinned out chain is expected to have the auto-correlation drop quickly. Any additional parameters passed to the 3 plotting function will be automatically used in the `plot` function for `plt.trace` and `plt.density`, and in the `acf` function for `plt.acf`.

Plotting the study-specific CACE in a forest plot

A graphical overview of the results can be obtained by creating a forest plot (Lewis and Clarke 2001). The function `plt.forest()` draws a forest plot for θ^{CACE} estimated from the meta-analysis. Users can call this function for the objects from `cace.meta.c()` or `cace.meta.ic()`. Here is an example using the object `out.meta.ic`:

```
plt.forest(data = epidural_ic, obj = out.meta.ic)
```

Note that in addition to the object `out.meta.ic`, users also need to specify the dataset used to compute that object, from which the `plt.forest()` function extracts the study names and publication years for the figure.

Figure 5 is a forest plot of θ_i^{CACE} for each study individually, using the Bayesian method with full random effects and default priors. The summary estimate based on the model `cace.meta.ic()`

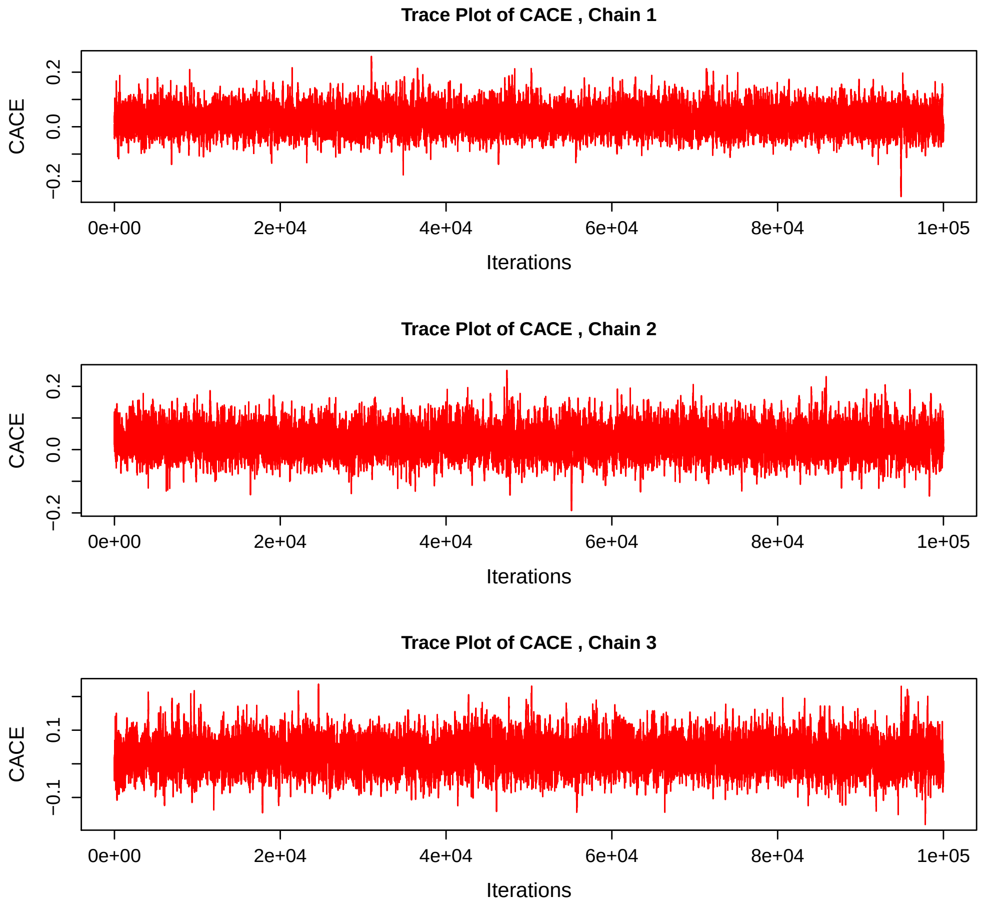


Figure 2: Trace plots for θ^{CACE} from the epidural_ic dataset fit using cace.meta.ic() for a sample of 3 chains. Because there are no strong patterns and the variability is relatively constant, we can conclude that the posterior means are drawn from a stationary distribution.

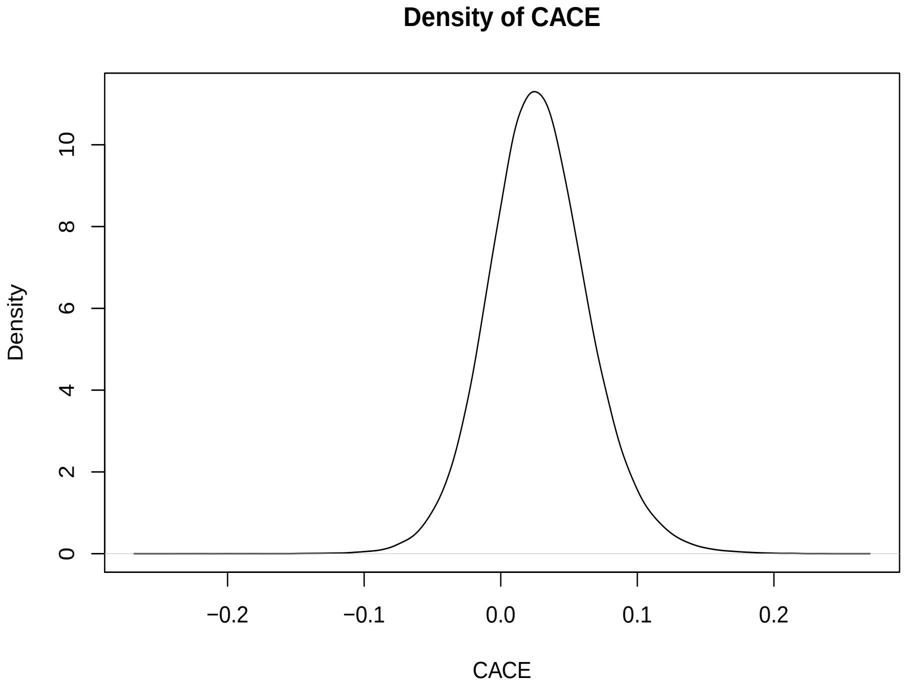


Figure 3: The kernel smoothed density for θ^{CACE} from the function cace.meta.ic() applied to the epidural analgesia in labor meta-analysis. The posterior mean is close to 0, indicating that the complier average causal effect may not be significant in this case.

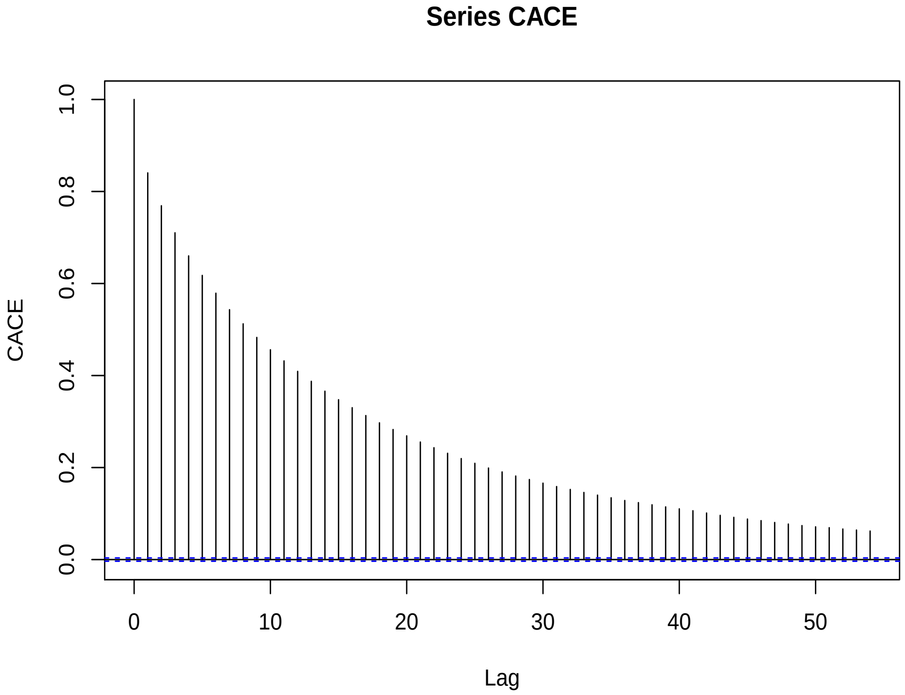
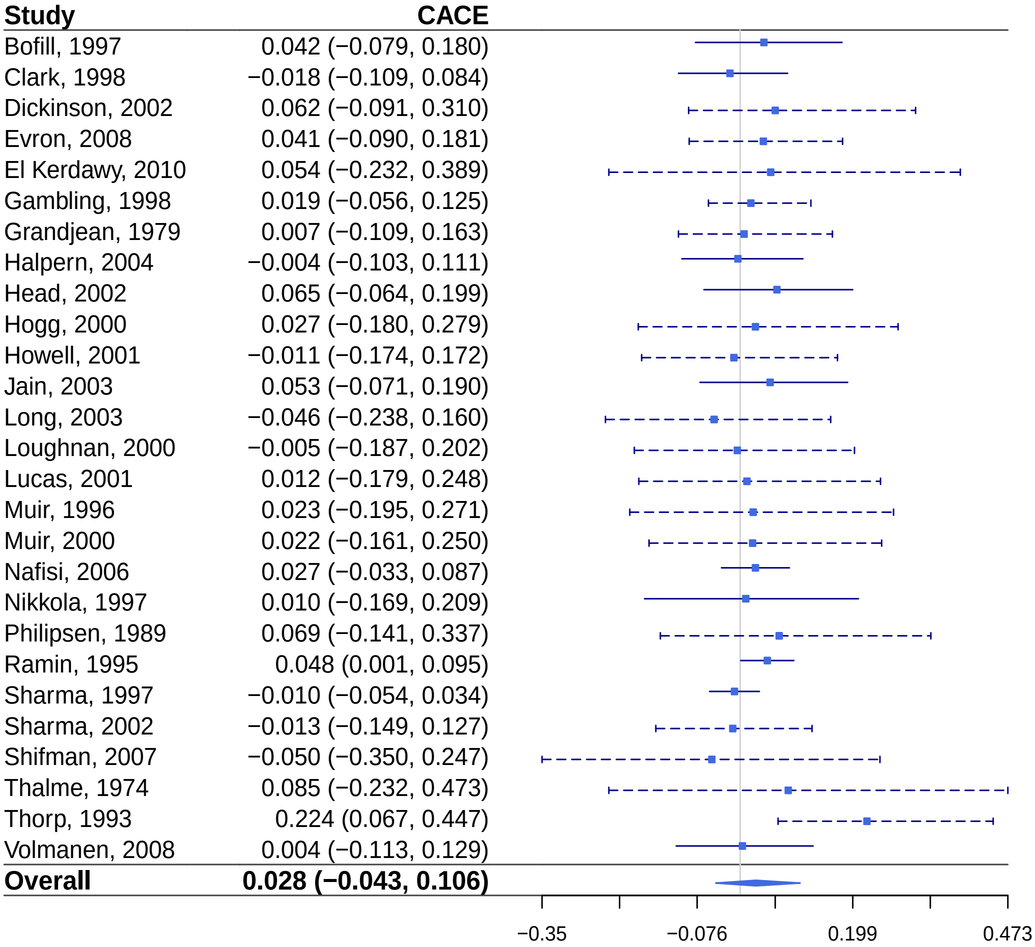


Figure 4: Auto-correlation plot of θ^{CACE} from the model cace.meta.ic() fit to the epidural_ic dataset. As the lag increases, the values become less correlated. Users can choose to address high auto-correlation with a longer chain or a larger n.thin.



Study	CACE
Bofill, 1997	0.042 (−0.079, 0.180)
Clark, 1998	−0.018 (−0.109, 0.084)
Dickinson, 2002	0.062 (−0.091, 0.310)
Evron, 2008	0.041 (−0.090, 0.181)
El Kerdawy, 2010	0.054 (−0.232, 0.389)
Gambling, 1998	0.019 (−0.056, 0.125)
Grandjean, 1979	0.007 (−0.109, 0.163)
Halpern, 2004	−0.004 (−0.103, 0.111)
Head, 2002	0.065 (−0.064, 0.199)
Hogg, 2000	0.027 (−0.180, 0.279)
Howell, 2001	−0.011 (−0.174, 0.172)
Jain, 2003	0.053 (−0.071, 0.190)
Long, 2003	−0.046 (−0.238, 0.160)
Loughnan, 2000	−0.005 (−0.187, 0.202)
Lucas, 2001	0.012 (−0.179, 0.248)
Muir, 1996	0.023 (−0.195, 0.271)
Muir, 2000	0.022 (−0.161, 0.250)
Nafisi, 2006	0.027 (−0.033, 0.087)
Nikkola, 1997	0.010 (−0.169, 0.209)
Philipsen, 1989	0.069 (−0.141, 0.337)
Ramin, 1995	0.048 (0.001, 0.095)
Sharma, 1997	−0.010 (−0.054, 0.034)
Sharma, 2002	−0.013 (−0.149, 0.127)
Shifman, 2007	−0.050 (−0.350, 0.247)
Thalme, 1974	0.085 (−0.232, 0.473)
Thorp, 1993	0.224 (0.067, 0.447)
Volmanen, 2008	0.004 (−0.113, 0.129)
Overall	**0.028 (−0.043, 0.106)**

Figure 5: Forest plot of study-specific θ^{CACE} from the model cace.meta.ic() with full random effects fit to the epidural_ic dataset. The summary estimate and confidence interval limits based on the model cace.meta.ic() are included in the figure, both in terms of written values and the squares and lines on the right. Overall, it shows that the study-specific θ_i^{CACE} vary from negative to positive in individual studies, while most of the 95% credible intervals cover zero.

is automatically added to the figure, with the outer edges of the polygon indicating the confidence interval limits. The 95% credible interval of the summary θ^{CACE} covers zero, indicating a non-significant complier average causal effect estimate for using epidural analgesia in labor on the risk of cesarean section for the meta-analysis with 27 trials. For a study with incomplete data on compliance status, a dashed horizontal line in the forest plot is used to represent the posterior 95% credible interval of θ_i^{CACE} from the Bayesian hierarchical model fit. The study-specific θ_i^{CACE} vary from negative to positive in individual studies, while most of the 95% credible intervals cover zero. As the θ_i^{CACE} for a trial without complete compliance data is not estimable using only data from that single trial, dashed lines tend to have longer credible intervals than those with complete data (solid lines).

4 Discussion

This article provides an overview of the **BayesCACE** package for conducting CACE analysis with R. Bayesian hierarchical models estimating the CACE in individual studies and in meta-analysis are introduced to demonstrate the underlying methods of the functions. Practical usage of various functions is illustrated using real meta-analysis datasets `epidural_c` and `epidural_ic`. The package provides several plots for model outputs and model diagnosis.

It is important to note that the two-step approach for meta-analysis is included in the package **BayesCACE** because by using the full observed data from a single study i, θ_i^{CACE} is identifiable, making it possible to pool the estimated posterior means and standard deviations of the θ_i^{CACE} in a meta-analysis. However, the Bayesian hierarchical-model meta-analysis method for estimating the overall CACE is preferred for two reasons: the conventional two-step approach requires the whole set of parameters to be estimated for each trial, giving a greater total number of parameters than the random effect model, so the estimate of the CACE can be less efficient. Also, when study i does not report complete compliance data, it must be excluded from the two-step approach because θ_i^{CACE} is no longer directly estimable by simply using the incomplete data from this individual study, while the function `cace.meta.ic()` can use the incomplete information and thus help improve the efficacy in estimation.

The Gelman and Rubin convergence statistics, time-series standard errors, trace plots, and autocorrelation plots are provided by the package **BayesCACE** to examine whether the MCMC chains are drawn from stationary distributions. However, in practice, any sample is finite, thus there is no guaranteed way to prove that the sampler has converged (Kass et al. 1998; Cowles and Carlin 1996). Additional techniques may be required to determine the effective sample size for adequate convergence (Robert and Casella 2004). For example, the well-developed R package **mcmcse** (Flegal et al. 2017) can be used to assess whether MCMC has been run for enough iterations (sufficient chain lengths). To call the functions in **mcmcse**, users can specify the argument `mcmc.samples = TRUE` in `cace.study()`, `cace.meta.c()`, and `cace.meta.ic()`, so the MCMC posterior samples of monitored parameters are saved in the output objects.

The current version of **BayesCACE** only applies to binary outcomes. However, the Bayesian hierarchical model can be extended to handle ordinal outcomes $o \in \{1, \ldots, O\}$.
By selecting weighting scores $\{W_1, W_2, \ldots, W_O\}$ to reflect distances between outcome categories $\{1, \ldots, O\}$, θ_i^{CACE} is defined as $E(Y_{ij}^1 - Y_{ij}^0 | C_{ij} = 1) = \sum_o (W_o \times u_{io}) - \sum_o (W_o \times v_{io})$ (J. Zhou, Hodges, and Chu 2021; J. Zhou et al. 2019). Equally spaced scores $\{1, 2, ..., O\}$, their linear transforms, and midranks are reasonable weight choices (Agresti 2013). Future work will add CACE meta-analysis functions for ordinal outcomes, and allow users to choose their preferred weights $\{W_1, W_2, \ldots, W_O\}$. Note that ordinal outcomes lead to more complex correlation structures in the parameters related to response rates, so multivariate prior distributions are necessary to analyze such outcomes.

References

Agresti, Alan. 2013. *Categorical Data Analysis*. Third Edition. Hoboken, NJ: John Wiley & Sons.

Baccini, Michela, Alessandra Mattei, and Fabrizia Mealli. 2017. "Bayesian Inference for Causal Mechanisms with Application to a Randomized Study for Postoperative Pain Control." *Biostatistics* 18 (4): 605–17.

Baker, Stuart G. 2020. "CACE and Meta-Analysis (Letter to the Editor)." *Biometrics* 76 (4): 1383–84.

Bannister-Tyrrell, Melanie, Branko Miladinovic, Christine L Roberts, and Jane B Ford. 2015. "Adjustment for Compliance Behavior in Trials of Epidural Analgesia in Labor Using Instrumental Variable Meta-Analysis." Journal Article. *Journal of Clinical Epidemiology* 68 (5): 525–33.

Brooks, Stephen P, and Andrew Gelman. 1998. "General Methods for Monitoring Convergence of Iterative Simulations." Journal Article. *Journal of Computational and Graphical Statistics* 7 (4): 434–55.

Clopper, Charles J, and Egon S Pearson. 1934. "The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial." *Biometrika* 26 (4): 404–13.

Coggeshall, Scott. 2017. **noncomplyR***: Bayesian Analysis of Randomized Experiments with Non-Compliance*. `https://CRAN.R-project.org/package=noncomplyR`.

Cowles, Mary Kathryn, and Bradley P Carlin. 1996. "Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review." *Journal of the American Statistical Association* 91 (434): 883–904.

Flegal, James M, John Hughes, Dootika Vats, and N Dai. 2017. **mcmcse***: Monte Carlo Standard Errors for MCMC*. `https://CRAN.R-project.org/package=mcmcse`.

Frangakis, Constantine E, and Donald B Rubin. 2002. "Principal Stratification in Causal Inference." *Biometrics* 58 (1): 21–29.

Freedman, Laurence S. 1990. "The Effect of Partial Noncompliance on the Power of a Clinical Trial." *Controlled Clinical Trials* 11 (3): 157–68.

Gelman, Andrew, and Donald B Rubin. 1992. "Inference from Iterative Simulation Using Multiple Sequences." Journal Article. *Statistical Science* 7 (4): 457–72.

Gilbert, Peter B, Erin E Gabriel, Ying Huang, and Ivan SF Chan. 2015. "Surrogate Endpoint Evaluation: Principal Stratification Criteria and the Prentice Definition." *Journal of Causal Inference* 3 (2): 157–75.

Gordon, Max, and Thomas Lumley. 2017. **forestplot***: Advanced Forest Plot Using "Grid" Graphics*. `https://CRAN.R-project.org/package=forestplot`.

Harville, David A. 1977. "Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems." *Journal of the American Statistical Association* 72 (358): 320–38.

Hedges, Larry V, and Ingram Olkin. 1985. *Statistical Methods for Meta-Analysis*. Orlando, FL: Academic Press.

Hedges, Larry V, and Jack L Vevea. 1998. "Fixed- and Random-Effects Models in Meta-Analysis." *Psychological Methods* 3 (4): 486–504.

Hudgens, Michael G, and M Elizabeth Halloran. 2006. "Causal Vaccine Effects on Binary Postinfection Outcomes." *Journal of the American Statistical Association* 101 (473): 51–64.

Jiang, Yang, and Dylan Small. 2014. **ivpack***: Instrumental Variable Estimation*. `https://CRAN.R-project.org/package=ivpack`.

Kasim, Adetayo, ZhiMin Xiao, Steve Higgings, and Ewoud De Troyer. 2017. **eefAnalytics***: Analysing Education Trials*. `https://CRAN.R-project.org/package=eefAnalytics`.

Kass, Robert E, Bradley P Carlin, Andrew Gelman, and Radford M Neal. 1998. "Markov Chain Monte Carlo in Practice: A Roundtable Discussion." *The American Statistician* 52 (2): 93–100.

Laird, Nan M, and Frederick Mosteller. 1990. "Some Statistical Methods for Combining Experimental Results." *International Journal of Technology Assessment in Health Care* 6 (1): 5–30.

Lewis, Steff, and Mike Clarke. 2001. "Forest Plots: Trying to See the Wood and the Trees." *BMJ* 322 (7300): 1479–80.

Lin, D Y, and D Zeng. 2010. "On the Relative Efficiency of Using Summary Statistics Versus Individual-Level Data in Meta-Analysis." *Biometrika* 97 (2): 321–32.

Lunn, David, Chris Jackson, Nicky Best, David Spiegelhalter, and Andrew Thomas. 2012. *The BUGS Book: A Practical Introduction to Bayesian Analysis*. New York, NY: Chapman; Hall/CRC.

Plummer, Martyn. 2018. **rjags***: Bayesian Graphical Models Using MCMC*. `https://CRAN.R-project.org/package=rjags`.

Plummer, Martyn, Nicky Best, Kate Cowles, and Karen Vines. 2006. "CODA: Convergence Diagnosis and Output Analysis for MCMC." *R News* 6 (1): 7–11.

Robert, Christian, and George Casella. 2004. *Monte Carlo Statistical Methods*. New York, NY: Springer Science & Business Media.

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

Viechtbauer, Wolfgang. 2010. "Conducting Meta-Analyses in r with the **metafor** Package." *Journal of Statistical Software* 36 (3): 1–48.

White, Halbert. 1982. "Instrumental Variables Regression with Independent Observations." *Econometrica* 50 (2): 483–99.

Zeger, Scott L, Kung-Yee Liang, and Paul S Albert. 1988. "Models for Longitudinal Data: A Generalized Estimating Equation Approach." *Biometrics* 44 (4): 1049–60.

Zhou, Jincheng, Haitao Chu, Michael G Hudgens, and M Elizabeth Halloran. 2016. "A Bayesian Approach to Estimating Causal Vaccine Effects on Binary Post-Infection Outcomes." *Statistics in Medicine* 35 (1): 53–64.

Zhou, Jincheng, James S Hodges, and Haitao Chu. 2020. "Rejoinder to 'CACE and Meta-Analysis (Letter to the Editor)' by Stuart Baker." *Biometrics* 76 (4): 1385.

———. 2021. "A Bayesian Hierarchical CACE Model Accounting for Incomplete Noncompliance with Application to a Meta-Analysis of Epidural Analgesia on Cesarean Section." *Journal of the American Statistical Association* 116 (536): 1700–1712.

Zhou, Jincheng, James S. Hodges, M. Fareed K. Suri, and Haitao Chu. 2019. "A Bayesian Hierarchical Model Estimating CACE in Meta-Analysis of Randomized Clinical Trials with Noncompliance." *Biometrics* 75 (3): 978–87.

Zhou, Ting, Jincheng Zhou, James S Hodges, Lifeng Lin, Yong Chen, Stephen R Cole, and Haitao Chu. 2021. "Estimating the Complier Average Causal Effect in a Meta-Analysis of Randomized Clinical Trials with Binary Outcomes Accounting for Noncompliance: A Generalized Linear Latent and Mixed Model Approach." *American Journal of Epidemiology* 191 (1): 220–29.

Jincheng Zhou
Gilead Inc.
Clinical Data Science
Foster City, CA 94404, USA
jeni.zhou9@gilead.com

Jinhui Yang
University of Minnesota Twin Cities
Department of Computer Science and Engineering
Minneapolis, MN 55455, USA
yang7004@umn.edu

James S. Hodges
University of Minnesota Twin Cities
Division of Biostatistics
School of Public Health
Minneapolis, MN 55455, USA
hodge003@umn.edu

Lifeng Lin
University of Arizona
Department of Epidemiology and Biostatistics
Mel and Enid Zuckerman College of Public Health
Tucson, AZ 85724, USA
lifenglin@arizona.edu

Haitao Chu
(Affliation 1) Pfizer Inc. (Affliation 2) University of Minnesota Twin Cities
(1) Statistical Research and Data Science Center
New York, NY 10017, USA
(2) Division of Biostatistics
School of Public Health
Minneapolis, MN 55455, USA
(1) Haitao.Chu@Pfizer.com *(2)* chux0051@umn.edu

A Framework for Producing Small Area Estimates Based on Area-Level Models in R

by Sylvia Harmening, Ann-Kristin Kreutzmann, Sören Schmidt, Nicola Salvati and Timo Schmid

Abstract The R package **emdi** facilitates the estimation of regionally disaggregated indicators using small area estimation methods and provides tools for model building, diagnostics, presenting, and exporting the results. The package version 1.1.7 includes unit-level small area models that rely on access to micro data. The area-level model by Fay and Herriot (1979) and various extensions have been added to the package since the release of version 2.0.0. These extensions include (a) area-level models with back-transformations, (b) spatial and robust extensions, (c) adjusted variance estimation methods, and (d) area-level models that account for measurement errors. Corresponding mean squared error estimators are implemented for assessing the uncertainty. User-friendly tools like a stepwise variable selection, model diagnostics, benchmarking options, high quality maps and results exportation options enable a complete analysis procedure. The functionality of the package is illustrated by examples based on synthetic data for Austrian districts.

1 Introduction

Small area estimation (SAE) enables better insight at smaller scales, for which it has gained importance in both academic and applied research. Among others, SAE is used for estimating socio-economic measures like income, poverty and health or indicators for agriculture (Datta et al., 1991; Tzavidis et al., 2012; Zhang et al., 2015; Pratesi, 2016). Economic or political decision-makers and official statistics practitioners especially benefit from reliable estimation of disaggregated indicators and thus SAE methods. Existing surveys were often not planned for analysis at disaggregated levels and show only small sample sizes, which leads to a low precision of the estimates. SAE methods can be employed to avoid expensive and time-consuming enlargements of the sample size of surveys. The idea is to combine data sources with model-based approaches. Existing survey data will be enriched by auxiliary information, e.g., from census or register data, to improve the accuracy of the indicator estimation on an area- or domain- level. The terms area and domain are used interchangeably and refer either to a geographic area or to any subpopulation of a population of interest, like socio-demographic groups. Among others, Pfeffermann (2013), Rao and Molina (2015), Tzavidis et al. (2018) and Jiang and Rao (2020) give comprehensive overviews of SAE methods.

The main goal of the package **emdi** is the simplification of estimating these regionally disaggregated indicators. The package version 1.1.7 contains direct estimation based exclusively on survey data and model-based estimation using the unit-level empirical best predictor (EBP) method (Molina and Rao, 2010). The EBP approach is powerful since it enables the simultaneous estimation of various indicators. For this, it relies on unit-level information, i.e. information about each unit in each domain. Though survey data often provides unit-level information, access to census or register data at unit-level is less likely. Hence, area-level models provide a valuable alternative, with the following benefits: First, only area-level aggregates are needed to estimate the regional indicators. Second, area-level models can consider the survey design by integrating the sampling weights. Third, the computation is faster compared to the computational intensive EBP approach.

Various R packages that employ different area-level models are available on the Comprehensive R Archive Network (CRAN). The package **smallarea** (Nandy, 2015) offers several variance estimation methods for the standard Fay-Herriot (FH) model: maximum likelihood (ML), residual maximum likelihood (REML), and both Prasad-Rao and Fay-Herriot method-of-moment. Estimation of unknown sampling variances is also offered." The ability to estimate unit- and area-level models under heteroscedasticity is implemented by the **JoSAE** package (Breidenbach, 2018). Robust estimation of area-level models with spatial and/or temporal structures in the random effects is supported by package **saeRobust** (Warnholz, 2022). The **mcmcsae** package (Boonstra, 2021) also takes spatial and temporal correlation of the random effects into account, but fits unit- and area-level models via Markov Chain Monte Carlo simulation. Estimation of univariate and multivariate FH models is possible with package **msae** (Permatasari and Ubaidillah, 2022). The package **hbsae** (Boonstra, 2022) allows for the fitting of unit- and area-level models by frequentist or hierarchical Bayesian approaches. The possibility of estimating FH models and some of its extensions in a Bayesian framework is also given by the **BayesSAE** package (Developer, 2018). The **tipsae** package (De Nicolò and Gardini, 2022) provides estimation and mapping tools within a Bayesian setting for proportions that are defined

	Package								
Area-level model	***smallarea***	***JoSAE***	***sae***	***saeRobust***	***msae***	***hbsae***	***BayesSAE***	***saeME***	***emdi***
Standard variance estimation	√				√	√			√
Adjusted variance estimation									√
Unknown sampling variances	√								
Heteroscedasticity		√							
Spatial correlation			√						√
Spatio-temporal correlation			√						
Robust				√					√
Robust, spatial correlation				√					√
Robust, (spatio-)temporal correlation				√					
Multivariate					√				
Bayesian formulation						√	√		
Measurement error								√	√
Transformation (log, arcsin)									√

Table 1: Overview of selected implemented area-level models in R packages available on CRAN.

on the unit interval. The **mme** package (Lopez-Vizcaino et al., 2019) implements Gaussian area-level multinomial mixed-effects models in the SAE context. The **saeME** package (Mubarak and Ubaidillah, 2022) can fit an area-level model when the auxiliary variables are measured with error. The **NSAE** package (Chandra et al., 2022) can fit stationary and nonstationary FH models. One of the commonly used packages is the **sae** package (Molina and Marhuenda, 2015). It includes a wide range of area-level models (the standard FH model with REML, ML and FH method-of-moment model fitting and a spatial and a spatio-temporal extension of the FH model) and unit-level models (the nested error linear regression model of Battese et al. (1988) and the EBP approach). Table 1 gives an overview of selected packages and the implemented methodology.

Besides packages that include particular area-level models, the packages **saeMSPE** (Xiao et al., 2022) and **SAEval** (Fasulo, 2022) offer different analytical- and resampling-based MSE estimators and tools for diagnostics and graphical evaluation of SAE models, respectively.

The latest version of package **emdi** 2.1.3 combines a wide range of SAE models with several tools that enable a complete analysis, and therefore adds to the space of useful packages, for the following reasons:

- None of the existing packages contains such a variety of different area-level models.
- In addition to models that are already available in existing R packages, **emdi** includes: adjusted variance estimation methods and transformation options for the standard FH model. Adjusted variance estimation methods are of particular importance when working in a non-Bayesian framework. In a Bayesian context, the variance will always be estimated as strictly positive, so packages providing a Bayesian approach do not need adjusted variance estimation methods.
- Package **emdi** offers user-friendly tools that go beyond model estimation: diagnostic tools with both summary and graphical results, benchmarking options, high-quality geographical visualization of results, and export of results to Excel and OpenDocument Spreadsheet formats.
- Plus a stepwise variable selection algorithm for area-level models is included in **emdi** to allow the user to build a model based on information criteria.

Thus, since package version 2.0.0, version 1.1.7 has been extended by various area-level models, but stays in line with the user-friendly orientation of the existing version.

The structure of the paper can be described as follows. The section Statistical methodology introduces the statistical methods implemented in the package. The example data sets included in the package are presented in the section Data sets. The section Functionality and case studies provides an illustrative description of the functions using the example data sets. The section Estimation procedure for the standard Fay-Herriot model guides the reader from model-building to diagnostics of a standard FH model and creating maps of the results. The section Estimation of the extended area-level models follows with short descriptions of how to build the different extended area-level models. Finally, the section Conclusion and outlook summarizes our contributions and gives an outlook.

2 Statistical methodology

Area-level models for the estimation of indicators like means, totals or shares have been added to the package since the release of version 2.0.0. These comprise the area-level model by Fay and Herriot (1979) and several extensions of this standard model which account for issues that may come up in real data applications. To measure the precision of those models, MSE estimators have been integrated following the literature.

Standard Fay-Herriot model

Throughout the paper, a finite population U is assumed that consists of N units that are subdivided into D domains or areas of specific sizes $N_1, ..., N_D$. Then a random sample of size n can be drawn from U and partitioned into D areas with $n_1, ..., n_D$ observations per domain.

The FH model links area-level direct estimators based on survey data to covariates aggregated on an area level that may stem from administrative data (e.g. register or census) or alternative data sources (e.g. satellite, social media or mobile phone data). The FH model is composed of two levels. The first level is the sampling model

$$\hat{\theta}_i^{\text{Dir}} = \theta_i + e_i, \quad i = 1, \dots, D.$$

$\hat{\theta}_i^{\text{Dir}}$ is an unbiased direct estimator for a population indicator of interest θ_i, for instance, a mean or a ratio. e_i represents independent and normally distributed sampling errors with $e_i \overset{ind}{\sim} N\left(0, \sigma_{e_i}^2\right)$. Though the model assumes known sampling variances, in practical applications $\sigma_{e_i}^2$ are usually unknown, and have to be estimated from the unit-level sample data (Rivest and Vandal, 2003; Wang and Fuller, 2003; You and Chapman, 2006). Package **emdi** provides a non-parametric bootstrap for estimating the variances of the direct estimator (Alfons and Templ, 2013). To allow for complex survey designs, sampling weights (w) can be considered in the direct estimation (Horvitz and Thompson, 1952). For example, an estimator for the population mean θ_i of a continuous variable of interest y for each area i is estimated by

$$\hat{\theta}_i^{\text{Dir}} = \frac{\sum_{j=1}^{n_i} w_{ij} y_{ij}}{\sum_{j=1}^{n_i} w_{ij}},$$

where the index j indicates an individual with $j = 1, ..., n_i$ in the i-th area. The second FH level links the target indicator θ_i linearly to area-specific covariates $\boldsymbol{x}_i$,

$$\theta_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + u_i,$$

where $\boldsymbol{\beta}$ is a vector of unknown fixed-effect parameters, and u_i is an independent and identically normally distributed random effect with $u_i \overset{iid}{\sim} N\left(0, \sigma_u^2\right)$.
The combination of the sampling and the linking model leads to a special linear mixed model

$$\hat{\theta}_i^{\text{Dir}} = \boldsymbol{x}_i^\top \boldsymbol{\beta} + u_i + e_i, \quad i = 1, \dots, D. \tag{1}$$

The empirical best linear unbiased estimators $\hat{\boldsymbol{\beta}}$ are computed by weighted least square theory. The empirical best linear unbiased predictor (EBLUP) of θ_i is obtained by substituting the variance parameter σ_u^2 with an estimate. The resulting estimator can then be written as

$$\begin{aligned} \hat{\theta}_i^{\text{FH}} &= \boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}} + \hat{u}_i \\ &= \hat{\gamma}_i \hat{\theta}_i^{\text{Dir}} + (1 - \hat{\gamma}_i)\, \boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}}. \end{aligned} \tag{2}$$

The EBLUP/FH estimator can be understood as a weighted average of the direct estimator $\hat{\theta}_i^{\text{Dir}}$ and a regression-synthetic part $\boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}}$. The estimated shrinkage factor $\hat{\gamma}_i = \frac{\hat{\sigma}_u^2}{\hat{\sigma}_u^2 + \sigma_{e_i}^2}$ puts more weight on the direct estimator when the sampling variance is small and vice versa. Areas for which no direct estimation results exist, because the sample size is zero or the results may not be published, are called out-of-sample domains. For those domains, the prediction reduces to the regression-synthetic component $\hat{\theta}_{i,\text{out}}^{\text{FH}} = \boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}}$ (Rao and Molina, 2015).

Estimation methods for σ_u^2
The variance of the random effects has to be estimated. Commonly used approaches are the FH method-of-moment estimator (Fay and Herriot, 1979), the ML, and the REML estimators (Rao and Molina, 2015). The likelihood methods are known to perform more efficiently than the methods of moments (Rao and Molina, 2015). The commonly used methods can produce negative variance

estimates that should be strictly positive. In the estimation methods mentioned above, negative variance estimates are set to zero ($\hat{\sigma}_u^2 = \max\left(\tilde{\sigma}_u^2, 0\right)$) resulting in zero estimates of the shrinkage factor γ_i. Therefore, no weight is put on the direct estimator, ignoring its possible reliability. This poses a problem, especially when the number of areas is small. To avoid this so-called over-shrinkage problem, Li and Lahiri (2010) and Yoshimori and Lahiri (2014) proposed methods that adjust the respective likelihoods of the standard ML and REML approaches by some factor:

$$L_{\text{adj}}\left(\sigma_u^2\right) = A \times L\left(\sigma_u^2\right),$$

where A denotes the adjustment factor and $L(\sigma_u^2)$ the given likelihood function. The proposed adjustment factors are:

- by Li and Lahiri (2010): $A = \sigma_u^2$,
- by Yoshimori and Lahiri (2014): $A = \left(\tan^{-1}\left(\sum_{i=1}^{D} \gamma_i\right)\right)^{1/D}$.

Simulation studies conducted by Yoshimori and Lahiri (2014) showed that the adjusted Yoshimori-Lahiri methods are preferable when the variance of the random effect is small relative to the sampling variance. Otherwise, the adjusted Li-Lahiri methods are recommended. Package **emdi** offers six different variance estimation methods: standard ML (`ml`) and REML (`reml`), and adjusted ML and REML following either Li and Lahiri (2010) (`amrl`, `ampl`) or Yoshimori and Lahiri (2014) (`amrl_yl`, `ampl_yl`).

Extended area-level models

In real data applications, problems might occur that were not theoretically expected. There may also be some violation of the assumptions of the standard FH model, e.g., normality and independence of the error terms. The following section outlines the extensions of the standard FH model that are implemented in the package **emdi** to address these issues.

Transformations

When working with right-skewed data like income, wealth or business data, the assumptions of a linear relation between the response and the explanatory variables and normality of both error terms (u_i and e_i) of the FH model may be violated. Applying a log-transformation could be a reasonable solution to meet these model assumptions (Neves et al., 2013; Kreutzmann et al., 2022). In the **emdi** package, the direct estimates and their variances are transformed following Neves et al. (2013):

$$\hat{\theta}_i^{\text{Dir*log}} = \log\left(\hat{\theta}_i^{\text{Dir}}\right),$$
$$\text{var}\left(\hat{\theta}_i^{\text{Dir*log}}\right) = \left(\hat{\theta}_i^{\text{Dir}}\right)^{-2} \text{var}\left(\hat{\theta}_i^{\text{Dir}}\right),$$

where the *log notation stands for the logarithmic transformed scale. To obtain the FH estimator on the transformed scale $\hat{\theta}_i^{\text{FH*log}}$, $\hat{\theta}_i^{\text{Dir}}$ is substituted by $\hat{\theta}_i^{\text{Dir*log}}$ and $\text{var}(\hat{\theta}_i^{\text{Dir*log}})$ serves as estimate for the sampling variances ($\sigma_{e_i}^2$) in Equation 2. Since the logarithm is a nonlinear transformation, the final FH estimates on the original scale require a bias-corrected back-transformation (Slud and Maiti, 2006; Sugawasa and Kubokawa, 2017). The **emdi** package provides two options:

1. A *crude* method (`bc_crude`) that takes the properties of the log-normal distribution into account:
$$\hat{\theta}_i^{\text{FH, crude}} = \exp\left\{\hat{\theta}_i^{\text{FH*log}} + 0.5\text{MSE}\left(\hat{\theta}_i^{\text{FH*log}}\right)\right\}.$$
2. A bias correction suggested by Slud and Maiti (2006) (`bc_sm`) that further regards the bias due to the random effects:
$$\hat{\theta}_i^{\text{FH, Slud-Maiti}} = \exp\left\{\hat{\theta}_i^{\text{FH*log}} + 0.5\hat{\sigma}_u^2\left(1 - \hat{\gamma}_i^{\text{*log}}\right)\right\}.$$

The FH estimator on the transformed scale is denoted by $\hat{\theta}_i^{\text{FH*log}}$ and, accordingly $\text{MSE}(\hat{\theta}_i^{\text{FH*log}})$ stands for a MSE estimator on the transformed scale, e.g., the Prasad-Rao or Datta-Lahiri MSE (cf. following subsection). The Slud-Maiti back-transformation is derived for the ML variance estimation of the random effect and is implemented for in-sample domains in **emdi**. In the presence of out-of-sample domains, the *crude* method can be applied, which allows to use also other variance estimation methods.

Another transformation provided by the **emdi** package is the arcsin transformation, which is widely used when the direct estimator of the FH model is a ratio (Casas-Cordero et al., 2016; Schmid et al., 2017). The **emdi** package automatically transforms the direct estimates and the sampling variances as suggested by Jiang et al. (2001):

$$\hat{\theta}_i^{\text{Dir*arcsin}} = \sin^{-1}\left(\sqrt{\left(\hat{\theta}_i^{\text{Dir}}\right)}\right),$$
$$\operatorname{var}\left(\hat{\theta}_i^{\text{Dir*arcsin}}\right) = 1/\left(4\tilde{n}_i\right),$$

where the *arcsin denotes the arcsin transformed scale, and $\tilde{n}_i$ the effective sample size: the sample size adjusted by the sampling design (Jiang et al., 2001). The FH model is estimated using Equation 2 and, if necessary, the results are truncated to the interval $[0, \pi/2]$ to ensure final results between 0 and 1. To obtain final estimates on the original scale, the final estimation results must be subjected to a back-transformation. Two different back-transformations are available in **emdi**:

1. A *naive* back-transformation (`naive`):

$$\hat{\theta}_i^{\text{FH, naive}} = \sin^2\left(\hat{\theta}_i^{\text{FH*arcsin}}\right).$$

2. A back-transformation with bias-correction (`bc`) following Sugawasa and Kubokawa (2017) and Hadam et al. (2020):

$$\hat{\theta}_i^{\text{FH, bc}} = \int_{-\infty}^{\infty} \sin^2(t) \frac{1}{2\pi\frac{\hat{\sigma}_u^2\sigma_{e_i}^2}{\hat{\sigma}_u^2+\sigma_{e_i}^2}} \exp\left(-\frac{\left(t-\hat{\theta}_i^{\text{FH*arcsin}}\right)^2}{2\frac{\hat{\sigma}_u^2\sigma_{e_i}^2}{\hat{\sigma}_u^2+\sigma_{e_i}^2}}\right) dt.$$

Spatial FH model

The standard FH model assumes independence of the random effects. However, when working with geographical areas, assuming correlated random effects to incorporate a certain neighbouring structure can be valuable. The **emdi** package contains the spatial FH model introduced by Petrucci and Salvati (2006) that considers a simultaneously autoregressive process of order one, SAR(1). Compared to the standard model, the estimation differs mainly by discarding the assumptions of independent random effects and estimating a spatial autoregressive coefficient (ρ) which takes values between -1 and 1. Greater absolute values of (ρ) indicate a stronger relationship with the neighboring areas. The random effect u_i in Equation 1 is replaced by

$$\boldsymbol{u} = \rho_1 \boldsymbol{W}\boldsymbol{u} + \boldsymbol{\epsilon},\ \boldsymbol{\epsilon} \sim N\left(\boldsymbol{0}_D, \sigma_1^2 \boldsymbol{I}_D\right), \tag{3}$$

with $\boldsymbol{W}$ being the $D \times D$ row standardized proximity matrix that describes the neighbourhood structure of the areas, $\boldsymbol{0}_D$ a vector of zeros and $\boldsymbol{I}_D$ the $D \times D$ identity matrix. The random effects $\boldsymbol{u}$ of Equation 3 follow a SAR(1). When normality of the random effects is assumed, the model can be fitted by ML and REML. The application of spatial FH models should be considered when no geographic auxiliary variables are available to capture the spatial relation, or when ρ_1 is larger than 0.5 (Bertarelli et al., 2021). Even before estimating the model, the **emdi** package enables testing for spatial correlation by Moran's I and Geary's C statistics (Cliff and Ord, 1981; Pratesi and Salvati, 2008). While Moran's I mimics a typical correlation coefficient whose values range from -1 and 1, Geary's C takes values between 0 and 2 (0: positive, 1: no, 2: negative spatial autocorrelation). The two statistics behave inversely to each other.

Robust area-level models

In the case of influential outlying observations, the **emdi** package allows for robust versions of the standard and the spatial FH model. The theory is extensively studied in Warnholz (2016), wherein the robust estimation procedure for linear mixed models suggested by Sinha and Rao (2009) was extended to area-level models. The model fitting can be understood as a robustified ML version that also contains an influence function with a tuning constant `k`. 1.345 is recommended as an initial value for the tuning constant (Sinha and Rao, 2009). When non-symmetric outliers are expected to influence the robust estimation, a bias correction should be involved. This correction can be controlled by a multiplier constant (`mult_constant`). For further details, we also refer to Chambers et al. (2014) and Schmid et al. (2016).

Measurement error model

The standard FH model is based on the assumption that the covariates are measured without error (Fay and Herriot, 1979). This characteristic is typically assumed because census or register data are used as auxiliary information. However, when the covariate information stems from larger

Model	Type of MSE	Reference
Standard FH (depending on variance estimation of σ_u^2)		
`ml/ampl_yl`	Analytical	Datta and Lahiri (2000)
`reml/amrl_yl`	Analytical	Prasad and Rao (1990)
`ampl/amrl`	Analytical	Li and Lahiri (2010)
`ml/reml` (out-of-sample)	Analytical	Rao and Molina (2015)
Transformations		
`log` (depending on back-transformation)		
`bc_crude`	Analytical	Rao and Molina (2015)
`bc_sm`	Analytical	Slud and Maiti (2006)
`arcsin` (depending on back-transformation)		
`naive`	Jackknife	Jiang et al. (2001)
	Weighted Jackknife	Jiang et al. (2001); Chen and Lahiri (2002)
	Parametric bootstrap	Hadam et al. (2020)
`bc`	Parametric bootstrap	Hadam et al. (2020)
Spatial FH (depending on variance estimation)		
`ml/reml`	Analytical	Singh et al. (2005)
`ml/reml`	Parametric bootstrap	Molina et al. (2009)
`reml`	Nonparametric bootstrap	Molina et al. (2009)
Robust FH		
	Pseudolinear	Warnholz (2016)
	Parametric bootstrap	Warnholz (2016)
FH with ME		
	Jackknife	Jiang et al. (2002)

Table 2: Overview of the MSE estimation options of the `fh` function.

surveys or alternative data sources, this assumption can be violated. The **emdi** package includes an implementation of the measurement error (ME) model developed by Ybarra and Lohr (2008). To account for the ME in the covariates $\boldsymbol{x}_i$, they modified the shrinkage factor as follows:

$$\gamma_i = \frac{\sigma_u^2 + \boldsymbol{\beta}^\top \boldsymbol{C}_i \boldsymbol{\beta}}{\sigma_u^2 + \boldsymbol{\beta}^\top \boldsymbol{C}_i \boldsymbol{\beta} + \sigma_{e_i}^2},$$

where the $\boldsymbol{C}_i$ stands for the variance-covariance matrix of the covariates, which is a required prerequisite for the model. The modified shrinkage factor pulls more weight on the direct estimator when the variances of the covariates are large. A modified weighted least squares method and a moment estimator were used to estimate $\boldsymbol{\beta}$s and the σ_u^2, respectively. Additional details are available in Ybarra and Lohr (2008).

Mean squared error estimation

To evaluate the accuracy of the EBLUP estimates, the MSE is the most common measure used in SAE (Rao and Molina, 2015). The **emdi** package offers a variety of MSE estimators stemming from both analytical determination and resampling strategies, like the bootstrap and jackknife methods. Table 2 gives an overview of the included MSE approaches. For each area-level model presented in the previous sections, the provided MSE types are shown. The quoted references detail extensive formulas and derivations. As an additional measure of variability of the direct and FH estimates, within various functions and methods of the **emdi** package, the coefficient of variation (CV) is provided: $CV = \sqrt{\widehat{\text{MSE}}(\hat{\theta}_i)}/\hat{\theta}_i$, where $\hat{\theta}_i$ either stands for $\hat{\theta}_i^{\text{Dir}}$ or $\hat{\theta}_i^{\text{FH}}$.

3 Data sets

The **emdi** package version 1.1.7 contains a sample 'eusilcA_smp' and a population data set 'eusilcA_pop' at a household level. The generation process for both data sets is extensively described in Kreutzmann et al. (2019). Our process is nearly equivalent, but we do not produce out-of-sample domains for the

Variable	Meaning
Sample data set	
`Domain`	Austrian districts
`Mean`	Mean of the equivalized household income
`MTMED`	Share of households who earn more than the national median income
`Cash`	Mean employee cash or near cash income
`Var_Mean`	Variance of equivalized household income
`Var_MTMED`	Variance of share of households who earn more than the national median income
`Var_Cash`	Variance of employee cash or near cash income
`n`	Effective sample sizes
Population data set	
`Domain`	Austrian districts
`eqsize`	Equivalized household size according to the modified OECD scale
`cash`	Employee cash or near cash income
`self_empl`	Cash benefits or losses from self-employment (net)
`unempl_ben`	Unemployment benefits (net)
`age_ben`	Old-age benefits (net)
`surv_ben`	Survivor's benefits (net)
`sick_ben`	Sickness benefits (net)
`dis_ben`	Disability benefits (net)
`rent`	Income from rental of a property or land (net)
`fam_allow`	Family/children related allowances (net)
`house_allow`	Housing allowances (net)
`cap_inv`	Interest, dividends, profit from capital investments in unincorporated business (net)
`tax_adj`	Repayments/receipts for tax adjustment (net)
`ratio_n`	Ratios of the population size per area and the total population size

Table 3: Variables of the aggregated data sets. The `Domain` variables are factors, the rest of the variables are numeric. Except for the variables `Domain` and `ratio_n`, the observations of all variables of the population data set consist of the mean values per district.

area-level version of the data sets. The Austrian European Union Statistics on Income and Living Conditions (EU-SILC) synthetic 2006 data set (`eusilcP`) sourced from the **simFrame** package (Alfons et al., 2010) serves as basis for our data sets. The lowest regional level in the `eusilcP` data set consists of the nine Austrian states. Based on certain population size and income criteria, households were allocated to 94 Austrian districts resulting in the synthetic population data set 'eusilcA_pop'. For the 'eusilcA_smp' data set, a sample was drawn following a stratified random sampling process using the districts as strata. To show the usage of the FH model and its extensions, area-level data is required. The area-level survey and population data sets, 'eusilcA_smpAgg' and 'eusilcA_popAgg', are obtained by aggregation on the district level with the help of the `direct` function defined by the **emdi** package. The direct estimates in 'eusilcA_smpAgg' are the weighted mean equivalized household income `Mean`, the ratio of households that earn more than the national median income (`MTMED`) and their variances. These are based on the equivalized household income `eqIncome` in 'eusilcA_smp', defined as the total income of a household divided by the size of the household, with household size equalized by the modified equivalence scale of the Organisation for Economic Co-operation and Development (OECD) (Hagenaars et al., 1994). Additionally, the mean of the variable `cash`, its variance and the sample sizes are included in 'eusilcA_smpAgg', being required by the model extensions. The population data set 'eusilcA_popAgg' contains a variety of variables that describe different income sources of households, and a variable `ratio_n` that describes the ratios of the population sizes per area and the total population size. The variable `Domain` exists in both data sets and identifies the different districts. Both data sets have an observation for each of the 94 Austrian districts, with the sample data set 'eusilcA_smpAgg' containing eight variables and the population data set 'eusilcA_popAgg' containing fifteen. Table 3 provides an overview of all included variables of the sample and population data sets. For the creation of the proximity matrix used in the spatial FH model and the usage of the `map_plot` function, a shape file is needed. A shape file 'shape_austria_dis' (`.rda` format, `"SpatialPolygonsDataFrame"`) for the 94 districts of Austria is provided. This file was sourced from the SynerGIS website (Bundesamt für Eich- und Vermessungswesen, 2017). The data set 'eusilcA_prox', an example proximity matrix, has also been added to the **emdi** package. The creation of 'eusilcA_prox' is described in the following section.

4 Functionality and case studies

While the theoretical background of the implemented area-level models has been introduced in the section Statistical methodology, the focus of this section lies on the functionality and the workflow of their usage in R. All of the contained area-level models can be applied by one function: `fh`. Table 4 gives an overview of the 20 input arguments of function `fh`, together with a short description and any default settings. Not every argument needs a specification for every estimated model. Depending on the area-level model, different arguments have to be determined (see Table 6 in Appendix A). Figure 1 demonstrates the estimation possibilities of a standard FH model (for the extended area-level models see Figure 6 in Appendix A). In line with the `direct` and `ebp` functions of package version 1.1.7, the S3 object system is used for function `fh` (Chambers and Hastie, 1992). All three return objects of class `"emdi"`. The application of function `direct` leads to a `"direct"` object, and of functions `ebp` and `fh` to objects of classes `"ebp"` and `"fh"`, respectively. Though all of the returned objects contain ten components, not every component is available for each estimation method. In these cases they are indicated as `NULL` (see Table 5). Furthermore, the `model` component differs for the two classes `"ebp"` and `"fh"`. The components of objects of class `"fh"` are provided in Table 7 in Appendix B. Not all of

Argument	Description	Default
`fixed`	Formula of fixed-effects part of linear mixed model	
`vardir`	Domain-specific sampling variances of the direct estimates	
`combined_data`	Combined sample and census data set	
`domains`	Domain indentifier for `combined_data`	NULL
`method`	Model fitting method	reml
`interval`	Lower and upper limit for the variance estimation	NULL
`k`	Tuning constant for robust estimation	1.345
`mult_constant`	Bias correction multiplier constant for robust estimation	1
`transformation`	Type of transformation	no
`backtransformation`	Type of back-transformation	NULL
`eff_smpsize`	Effective sample sizes for the arcsin transformation	NULL
`correlation`	Correlation of random effects	no
`corMatrix`	Proximity matrix for the spatial model	NULL
`Ci`	Array of the variance-covariance matrix of the explanatory variables for each area for the ME model	NULL
`tol`	Tolerance value for the variance estimation	0.0001
`maxit`	Maximum number of iterations for the variance estimation	100
`MSE`	MSE estimation	FALSE
`mse_type`	Type of MSE estimator	analytical
`B`	Numbers of bootstrap iterations for computation of a bootstrap MSE and information criteria by Marhuenda et al. (2014)	c(50,0)
`seed`	Seed for random number generator	123

Table 4: Input arguments of function `fh`.

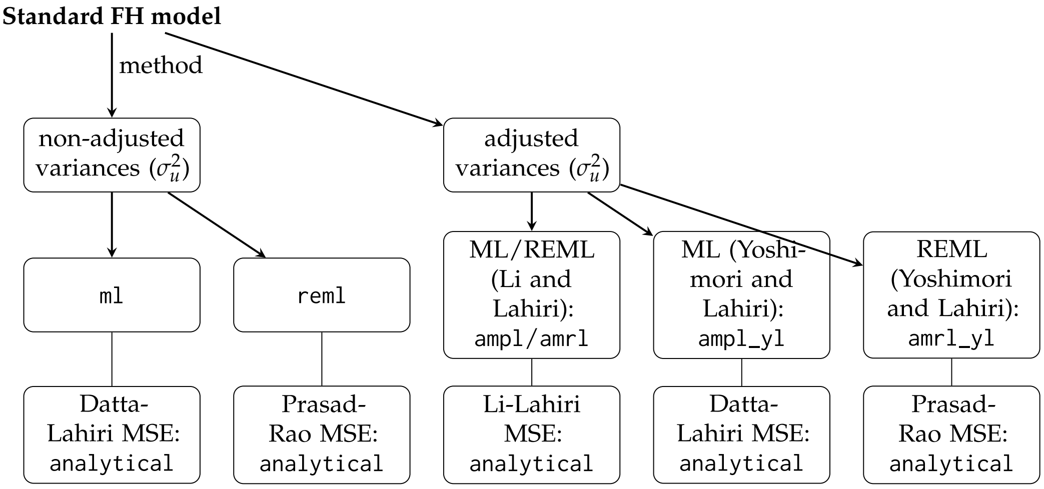

Standard FH model
method
non-adjusted variances (σ_u^2)
adjusted variances (σ_u^2)
ml
reml
ML/REML (Li and Lahiri): ampl/amrl
ML (Yoshimori and Lahiri): ampl_yl
REML (Yoshimori and Lahiri): amrl_yl
Datta-Lahiri MSE: analytical
Prasad-Rao MSE: analytical
Li-Lahiri MSE: analytical
Datta-Lahiri MSE: analytical
Prasad-Rao MSE: analytical

Figure 1: Overview of the standard FH model and adjusted variance estimation methods.

the components are available for every area-level model, e.g., the shrinkage factors per domain are not provided for the spatial and robust model extensions, as they do not have an intuitive interpretation in those cases. Due to the consistent structure, all functions and methods of **emdi** version 1.1.7 can be applied to objects of class "fh". Additionally, new functions and methods are available for the area-level models. Furthermore, a variety of methods that are available in base R and used by other model fitting R packages are included in the latest package version 2.1.3 for the different "emdi" objects. Two examples of the new generic functions used are coef and logLik. Figure 2 demonstrates the steps of a full data analysis procedure and the respective functions, from model building and diagnostics to presenting the results. The section Estimation procedure for the standard Fay-Herriot model demonstrates the procedure shown in Figure 2 by applying the standard FH model to the Austrian EU-SILC data described in the section Data sets. To demonstrate how the different extended area-level models are fitted with function fh, the section Estimation of the extended area-level models follows.

	Name	Description	Available for		
			direct	ebp	fh
1	ind	Point estimates per area	√	√	√
2	MSE	Variance/MSE estimates per area	√	√	√
3	transform_param	Transformation and shift parameters		√	
4	model	Fitted model		√	√
5	framework	List for data description	√	√	√
6	transformation	Type of transformation		√	√
7	method	Estimation method		√	√
8	fixed	Formula of fixed effects		√	√
9	call	Function call	√	√	√
10	successful_bootstraps	Number of successful bootstraps	√		√

Table 5: The ten "emdi" object components distinguished in "direct", "ebp" and "fh". More detailed information is provided by the package documentation.

Estimation procedure for the standard Fay-Herriot model

The aim of this example is to estimate the equivalized income for the 94 Austrian districts. The package and the example data sets are loaded as follows:

```
> library("emdi")
> data("eusilcA_popAgg")
> data("eusilcA_smpAgg")
```

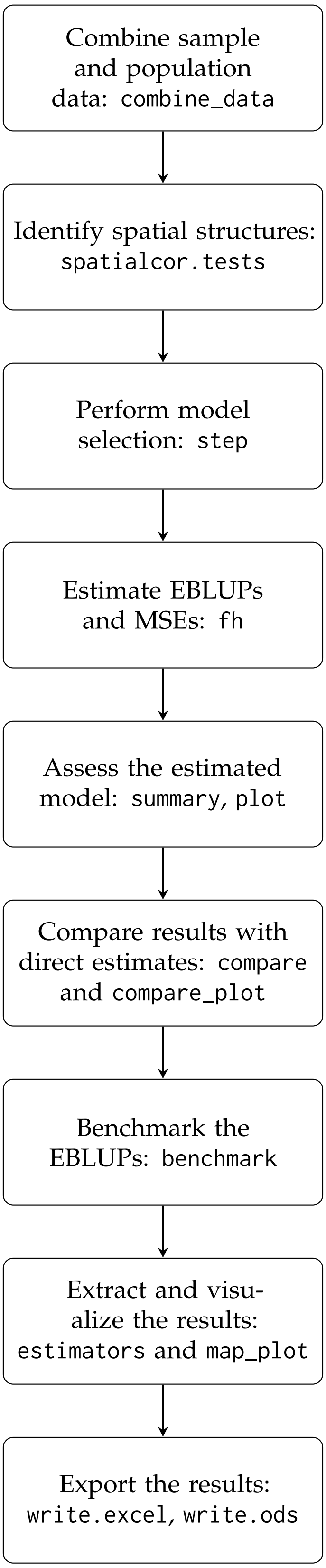

Combine sample and population data: combine_data
Identify spatial structures: spatialcor.tests
Perform model selection: step
Estimate EBLUPs and MSEs: fh
Assess the estimated model: summary, plot
Compare results with direct estimates: compare and compare_plot
Benchmark the EBLUPs: benchmark
Extract and visualize the results: estimators and map_plot
Export the results: write.excel, write.ods

Figure 2: Estimation procedure for area-level models.

Combine input data
The function fh requires one data set (argument combined_data) that comprises the sample and population data. Thus, the data set must contain all variables of the formula object fixed, the variances of the direct estimates and, optionally, a domain identifier. For cases where the sample and population data are only available separately, a merging function combine_data is provided. The necessary arguments are the two data sets and characters specifying the domain indicator for the respective data sets.

```
> combined_data <- combine_data(
+   pop_data = eusilcA_popAgg, pop_domains = "Domain",
+   smp_data = eusilcA_smpAgg, smp_domains = "Domain")
```

Identify spatial structures
With the help of a proximity matrix, Moran's I and Geary's C test statistics can be computed to identify spatial structures by the spatialcor.tests command. For the creation of the proximity matrix, the shapefile must be loaded. We load the Austrian shapefile that is provided and merge it to the sample data set by using the respective domain identifiers with the help of the merge method from the **sp** package (Pebesma and Bivand, 2005). Before merging, we sort the Austrian shapefile by the domains in the sample data.

```
> library("sp")
> load_shapeaustria()
> shape_austria_dis <- shape_austria_dis[
+   order(shape_austria_dis\$PB),]
> austria_shape <- merge(shape_austria_dis,
+   eusilcA_smpAgg, by.x = "PB", by.y = "Domain",
+   all.x = F)
```

Then the poly2nb and nb2mat functions of the **spdep** package (Bivand and Wong, 2018) are used. While poly2nb generates a list of neighbours that share joint boundaries, nb2mat computes a weights matrix. The style argument has to be set to W, as a row standardized proximity matrix is required.

```
> library("spdep")
> rel <- poly2nb(austria_shape,
+   row.names = austria_shape$PB)
> eusilcA_prox <- nb2mat(rel, style = "W",
+   zero.policy = TRUE)
```

Thus, a row standardized proximity matrix is generated that initially had weights of one if an area shares a boundary with another area and zero if not. Function spatialcor.tests makes use of the moran.test and geary.test functions with their respective default settings, from the **spdep** package. The input arguments are the created matrix and the direct estimates.

```
> spatialcor.tests(direct = combined_data$Mean,
+   corMatrix = eusilcA_prox)
```

```
  Statistics     Value      p.value
1 Moran's I 0.2453677 5.607958e-05
2 Geary's C 0.6238681 2.473294e-03
```

Since the output indicates only a weak positive spatial autocorrelation, the following estimation procedure does not consider the integration of a correlation structure for the random effects.

Perform model selection

Besides theoretical considerations on which auxiliary variables should be part of the model, the decision for the best model should be based on information criteria like the Akaike or Bayesian information criterion (AIC, BIC). Many applications use selection techniques based on linear regression (Casas-Cordero et al., 2016; Schmid et al., 2017). Instead, the **emdi** package provides the AIC, BIC, Kullback information criterion (KIC) and their bootstrap and bias corrected versions (AICc, AICb1, AICb2, KICc, KICb1, KICb2) especially developed for FH models by Marhuenda et al. (2014). These criteria are also included in the **sae** package, but the **emdi** package enables a stepwise variable selection procedure based on the chosen information criteria, comparable to the `step` function for `lm` models of package **stats**. The most important input arguments are an object of class `"fh"` and the direction of the stepwise search (`both`, `backward`, `forward`). In this example, the default setting `backward` and the `KICb2` information criterion is used. In the `fixed` argument of the `fh` function, the variables employee cash (`cash`), cash benefits from self-employment (`self_empl`) and unemployment benefits (`unempl_ben`) are included. For a valid comparison of models based on information criteria, the model fitting method must be `ml`. To activate the estimation of the information criteria by Marhuenda et al. (2014), we set the number of bootstrap iterations to 50. The output shows the stepwise removal of variables until the lowest KICb2 is reached, the function call and an overview of the estimated coefficients of the final recommended model.

```
> fh_std <- fh(fixed = Mean ~ cash + self_empl + unempl_ben, vardir = "Var_Mean",
+   combined_data = combined_data, domains = "Domain", method = "ml", B = c(0,50))
> step(fh_std, criteria = "KICb2")

Start: KICb2 = 1709.42
Mean ~ cash + self_empl + unempl_ben

            df  KICb2
- unempl_ben 1 1708.3
<none>         1709.4
- self_empl  1 1763.0
- cash       1 1808.6

Step: KICb2 = 1708.33
Mean ~ cash + self_empl

            df  KICb2
<none>         1708.3
- self_empl  1 1765.3
- cash       1 1816.1


Call:
fh(fixed = Mean ~ cash + self_empl, vardir = "Var_Mean",
    combined_data = combined_data,
    domains = "Domain", method = "ml", B = c(0, 50))

Coefficients:
             coefficients  std.error t.value   p.value
(Intercept)   3070.51231  635.94290  4.8283 1.377e-06 ***
cash             1.05939    0.07049 15.0288 < 2.2e-16 ***
self_empl        1.74564    0.22017  7.9284 2.219e-15 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

KICb2 is minimised when the variable `unempl_ben` is removed. Therefore, the formula `Mean` ∼ `cash + self_empl` is used in the following.

Estimate EBLUPs and MSEs

The standard FH model is built. In addition to the `fixed` part, required arguments are `vardir` and `combined_data`. We specify the `domains` (if the `domains` argument is set to `NULL`, the domains are numbered consecutively) and activate the estimation of the MSE and of the information criteria by Marhuenda et al. (2014).

```
> fh_std <- fh(fixed = Mean ~ cash + self_empl, vardir = "Var_Mean", combined_data =
+   combined_data, domains = "Domain", method = "ml", MSE = TRUE, B = c(0,50))
```

Assess the estimated model

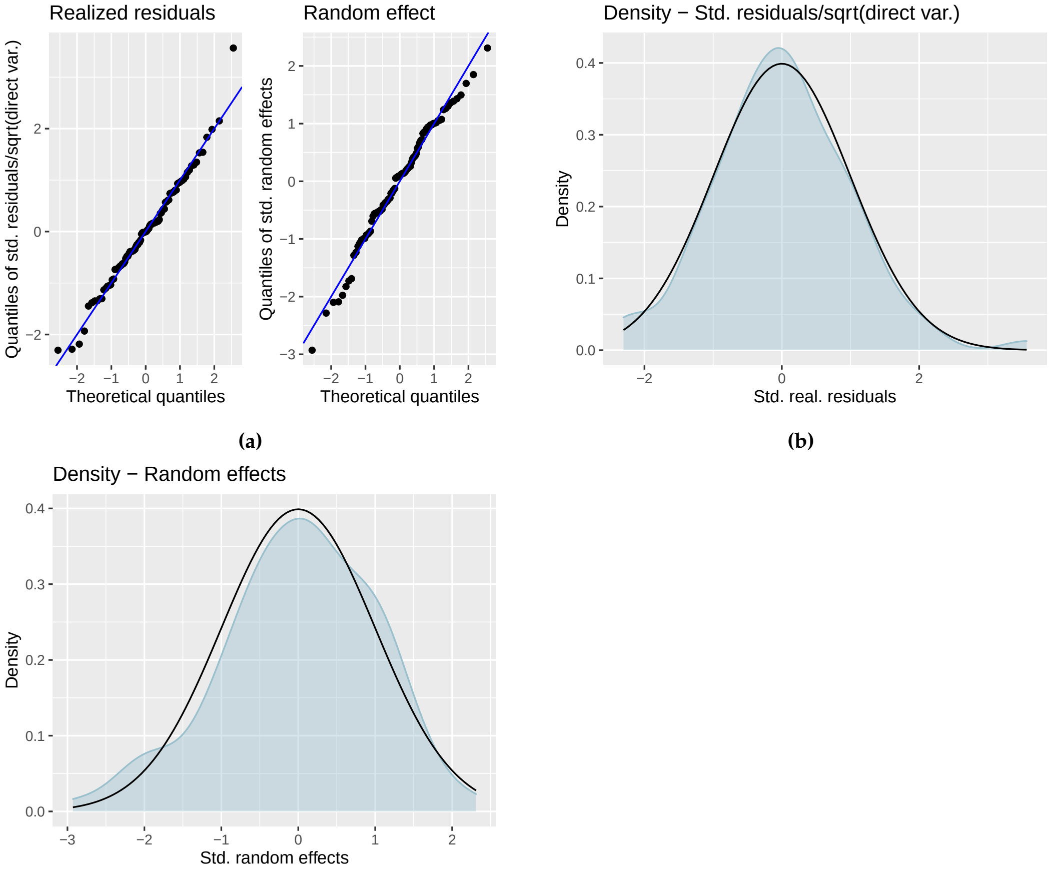
In many publications using FH models, model diagnostics are discussed only briefly, if at all. One reason for this might be the lack of an existing implementation of desired diagnostic measures in R or other statistical software. The summary method of **emdi** provides additional information about the data and model components, in particular the chosen estimation methods, the number of domains, the log-likelihood, the information criteria by Marhuenda et al. (2014), the adjusted R^2 of a standard linear model and the adjusted R^2 especially for FH models proposed by Lahiri and Suntornchost (2015). Additionally, measures to validate model assumptions about the standardized realized residuals and the random effects are provided: skewness and kurtosis (skewness and kurtosis of package **moments**, Komsta and Novomestky, 2015) of the standardized realized residuals and the random effects and the test statistics with corresponding p value of the Shapiro-Wilks-test for normality of both error terms. As the introduced area-level models do not assume a homoscedastic sampling distribution, the realized residuals ($\hat{e}_i$) are standardized by $\hat{e}_i^{\text{std}} = \hat{e}_i / \sigma_{e_i}$ for the summary and plot methods. The summary output differs slightly for the different implemented area-level models. For example, log-likehoods and thus information criteria are not available in theory for the robust and the ME model.

```
> summary(fh_std)

Call:
fh(fixed = Mean ~ cash + self_empl, vardir = "Var_Mean",
    combined_data = combined_data,
    domains = "Domain", method = "ml", MSE = TRUE, B = c(0, 50))

Out-of-sample domains: 0
In-sample domains: 94

Variance and MSE estimation:
Variance estimation method: ml
Estimated variance component(s): 1371195
MSE method: datta-lahiri

Coefficients:
           coefficients std.error  t.value   p.value
(Intercept)  3070.51231 635.94290   4.8283 1.377e-06 ***
cash            1.05939   0.07049  15.0288 < 2.2e-16 ***
self_empl       1.74564   0.22017   7.9284 2.219e-15 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Explanatory measures:
    loglike      AIC     AICc     AICb1     AICb2      BIC      KIC
1 -847.8303 1703.661 1703.91 1715.758 1703.461 1713.834 1707.661
      KICc    KICb1    KICb2     AdjR2     FH_R2
1 1708.783 1720.632 1708.335 0.9212817 0.9482498

Residual diagnostics:
                         Skewness Kurtosis Shapiro_W Shapiro_p
Standardized_Residuals  0.3004662 3.971216 0.9840810 0.3119346
Random_effects         -0.4113238 3.086048 0.9839858 0.3072834

Transformation: No transformation
```

The output of the example shows that all domains have survey information and the variance of σ_u^2 amounts to 1371195. Further, all of the included auxiliary variables are quite significant and their explanatory power is large with an adjusted R^2 (for FH models) of around 0.95. The results of the Shapiro-Wilk-test indicate that normality is not rejected for both errors. Graphical residual diagnostics are possible by the plot method.

```
> plot(fh_std)
```

Figure 3 shows normal quantile-quantile (Q-Q) plots of the standardized realized residuals and random effects (Figure 3a) as well as plots of the kernel densities of the distribution of both error terms and, for comparison, a standard normal distribution (Figure 3b and 3c). Like in **emdi** version 1.1.7, the user is free to modify the interface of the plots. The label and color arguments are easy to edit. Additionally, the overall appearance of the plots are changeable by the gg_theme argument, as the plots are built with the **ggplot2** package (Wickham, 2016). We refer to the package documentation for a detailed description of how to customize the `plot` arguments. Figure 3 supports the results of the normality tests provided in the `summary` output, the distribution of the standardized random effects may be slightly skewed (Figure 3c). If one is not be satisfied with the results, applying a log-transformation could improve the distribution of the error terms.



(a)

(b)

(c)

Figure 3: Output of `plot(fh_std)`: (a) normal quantile-quantile (Q-Q) plots of the standardized realized residuals and random effects, (b) and (c): kernel densities of the distribution of the standardized realized residuals and random effects (blue) in comparison to a standard normal distribution (black).

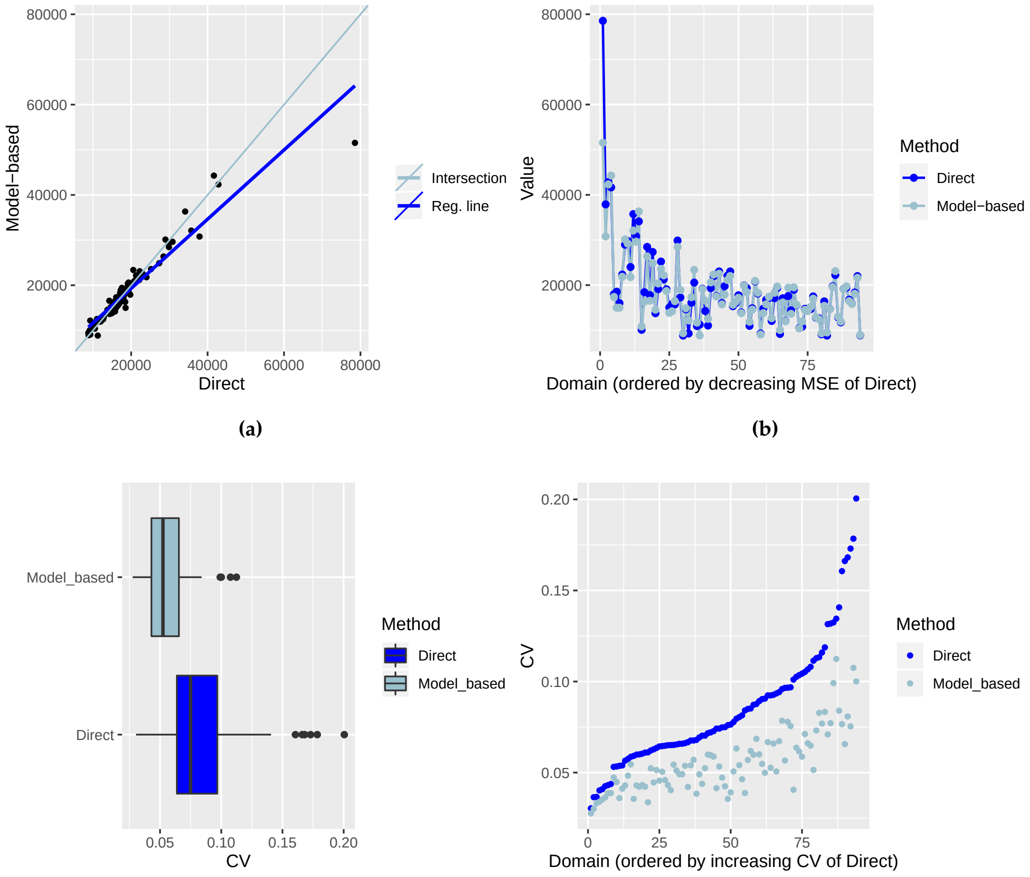
Compare results with direct estimates

The FH results should be consistent with the direct estimates for domains with a small direct MSE and/or large sample sizes. Further, the precision of the direct estimates should be improved by using auxiliary information. The comparison of the direct and model-based (FH) estimates can be done graphically by the generic function `compare_plot`. For the `fh` method the required input argument is an object of class `"fh"`. When the default settings of the command are used, the output consists of two plots: a scatter plot proposed by Brown et al. (2001) and a line plot. Besides the direct and FH estimates, the plot contains the fitted regression and the identity line. These two lines should not differ too much. Preferably, the model-based (FH) estimates should track the direct estimates within the line plot especially for domains with a large sample size or small MSE of the direct estimator. The points are ordered by decreasing MSE of the direct estimates. In addition, the input arguments `MSE` and `CV` can be set to `TRUE` leading to two extra plots, respectively. The MSE/CV estimates of the direct and model-based (FH) estimates are compared first via boxplots and second via ordered scatter plots (ordered by increasing CV of the direct estimates). Like for the `plot` command, a variety of customization options are offered, e.g., the label options (`label`), the format of the points (`shape`) and the style of the line (`line_type`).

```
> compare_plot(fh_std, CV = TRUE, label = "no_title")
```

Except one high value, the fitted regression and identity line of the scatter plot (Figure 4a) are relatively close. Note that the high value corresponds to the domain Eisenstadt (Stadt) with a very small sample size of 10 and the highest MSE of the direct estimates, so the direct estimator is very uncertain. Also the direct estimates are well tracked by the model-based (FH) estimates within the line plot (Figure 4b).

The boxplot (Figure 4c) and the ordered scatter plot (Figure 4d) show that the precision of the direct estimates could be improved by the usage of the FH model in terms of CVs. Additionally, all of the CV values are less than 20% which is a common rule of the UK Office for National Statistics in order to determine whether estimation results should be published (Miltiadou, 2020).



(a) (b)

(c) (d)

Figure 4: Output of `compare_plot(fh_std)`: (a) and (b) scatter and line plots of direct and model-based point estimates, (c) and (d) boxplot and scatter plots of the CV estimates of the direct and model-based (FH) estimates.

Further on, the function `compare` enables the user to compute a goodness of fit diagnostic (Brown et al., 2001) and a correlation coefficient of the direct estimates and the estimates of the regression-synthetic part of the FH model (Chandra et al., 2015). Following Brown et al. (2001), the difference between the model-based estimates and the direct estimates should not be significant (null hypothesis). The Wald test statistic is specified as

$$W\left(\hat{\theta}_i^{\text{FH}}\right) = \sum_{i=1}^{D} \frac{\left(\hat{\theta}_i^{\text{Dir}} - \hat{\theta}_i^{\text{FH}}\right)^2}{\widehat{\text{var}}\left(\hat{\theta}_i^{\text{Dir}}\right) + \widehat{\text{MSE}}\left(\hat{\theta}_i^{\text{FH}}\right)}$$

and is approximately χ^2-distributed with D degrees of freedom. When working with out-of-sample domains, those are not taken into account, because the direct estimates and their variances are missing. The input argument of function `compare` is an `"fh"` object.

```
> compare(fh_std)

Brown test

Null hypothesis: EBLUP estimates do not differ significantly from the
      direct estimates

  W.value Df p.value
 46.97181 94 0.9999874
```

```
Correlation between synthetic part and direct estimator: 0.94
```

The results of the goodness of fit statistic and the correlation coefficient confirm what the scatter and the line plot already indicated. In the example the null hypothesis is not rejected and the correlation coefficient indicates a strong positive correlation (0.94) between the direct and model-based (FH) estimates.

Benchmarking for consistent estimates

The idea of benchmarking is that the aggregated FH estimates should sum up to estimates of a higher regional level (τ):

$$\sum_{i=1}^{D} \xi_i \hat{\theta}_i^{\text{FH, bench}} = \tau,$$

where ξ_i stands for the share of the population size of each area in the total population size (N_i/N). In our example, the EBLUP estimates could get aggregated on a national level and then compared to or benchmarked with the Austrian mean equivalized income. The **emdi** package contains a benchmark function that allows the user to select three different options suggested by Datta et al. (2011). A general estimator of the three options can be written as follows:

$$\hat{\theta}_i^{\text{FH, bench}} = \hat{\theta}_i^{\text{FH}} + \left(\sum_{i=1}^{D} \frac{\xi_i^2}{\phi_i}\right)^{-1} \left(\tau - \sum_{i=1}^{D} \xi_i \hat{\theta}_i^{\text{FH}}\right) \frac{\xi_i}{\phi_i}.$$

Depending on the weight ϕ_i, the formula leads to different benchmarking options. If ϕ_i equals ξ_i, all FH estimates are adjusted by the same value (`raking`). A ratio adjustment (`ratio`) is being conducted if $\phi_i = \xi_i / \hat{\theta}_i^{\text{FH}}$. For the last option (`MSE_adj`), $\phi_i = \xi_i / \widehat{\text{MSE}}\left(\hat{\theta}_i^{\text{FH}}\right)$. While the first option is a relatively naive approach, the latter two conduct larger adjustments for the areas with larger FH and MSE estimates, respectively. Thus, for the benchmark function the following arguments have to be specified: an object of class `"fh"`, a `benchmark` value, a vector containing the ξ_is (`share`) and the `type` of benchmarking. The output is a data frame with an extra column `FH_Bench` for the benchmarked EBLUP values. If the optional argument `overwrite` is set to `TRUE`, the benchmarked results are added to the `"fh"` object and the MSE estimates of the non benchmarked FH estimates are set to `NULL`. For the used example, the benchmark value is calculated by taking the mean of the variable `eqIncome` of the 'eusilcA_smp' data frame. The ξ_is can be found in 'eusilcA_popAgg' as `ratio_n`.

```
> fh_bench <- benchmark(fh_std, benchmark = 20140.09, share = eusilcA_popAgg$ratio_n,
+   type = "ratio")
> head(fh_bench)

              Domain   Direct       FH FH_Bench Out
1          Amstetten 14768.57 14242.04 14480.61   0
2              Baden 21995.72 21616.40 21978.49   0
3            Bludenz 12069.59 12680.38 12892.79   0
4     Braunau am Inn 10770.53 11925.82 12125.59   0
5            Bregenz 35731.20 32101.69 32639.43   0
6 Bruck-Mürzzuschlag 23027.37 22523.50 22900.79   0
```

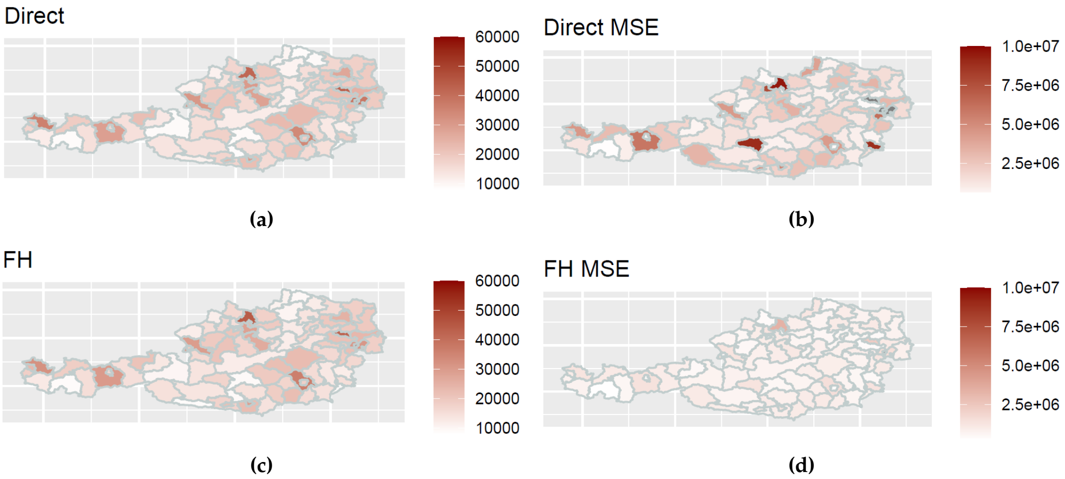
It is recognizable that for the first six Austrian districts the original estimates are slightly modified by the benchmarking.

Extract and visualize the results

To gain an overview of the point, MSE and CV results of the direct estimates compared to the model-based (FH) results the generic function `estimators` (Kreutzmann et al., 2019) can be used, but differences among areas or hotspots of special interest are usually easier to detect on maps. With function `map_plot`, the **emdi** package offers a user-friendly way to produce maps since creating maps can often become a time consuming task. The input arguments mainly consist of an object of class `"emdi"` and a spatial polygon of a shape file. The only issue that might come up is if domain identifiers in the data do not match to the respective identifiers of the shape file. In those cases, the input argument `map_tab` is required, which is a data frame that contains the matching of the domain indentifiers of the population and the shape file data sets. For detailed instructions, we refer to Kreutzmann et al. (2019) and to the help page of function `map_plot`.

For producing maps of the 94 Austrian districts, the Austrian shape file has to be loaded. In addition to the `"emdi"` object, the `"SpatialPolygonsDataFrame"` object (`map_obj`) and a domain indicator (`map_dom_id`) have to be specified. The `map_tab` argument is not necessary since the identifiers match in our example. To allow for an easier comparison of the results, we adjust the scales of the maps using the scale_points argument.

```
> load_shapeaustria()
> map_plot(object = fh_std, MSE = TRUE, map_obj = shape_austria_dis,
+   map_dom_id = "PB", scale_points = list(Direct = list(
+   ind = c(8000, 60000), MSE = c(200000, 10000000)), FH = list(
+   ind = c(8000, 60000), MSE = c(200000, 10000000))))
```



(a) (b) (c) (d)

Figure 5: Output of map_plot: Maps of the direct and FH estimates ((a) and (c)) with corresponding MSE estimates ((b) and (d)).

Figures 5a and 5c show the distribution of the estimated (direct vs. model-based) equivalized income across Austria. It is striking that white and light red tones dominate the map, indicating relatively low mean incomes of the districts. But in contrast, districts Eisenstadt (Stadt), Urfahr-Umgebung and Mödling stand out having the largest incomes. Urfahr-Umgebung is also eye-catching when having a look at the MSE estimates (Figures 5b and 5d). The MSE of the direct and the FH estimates are quite high. Probably a single wealthy household raised the mean income and also the variance. Figure 5b contains some districts with MSEs larger than the customized scaling (gray areas). Without the scaling it would have been hard to identify any differences in Figure 5d.

Export the results

Some users might have an interest to store the results separately or to use them for presentations. Excel and OpenDocument Spreadsheets provide many opportunities for that. In contrast to some existing R packages, the **emdi** functions write.excel/write.ods do not only export the estimation results, but also the output of summary. Usage of the functions is comprehensively described in Kreutzmann et al. (2019).

Estimation of the extended area-level models

FH model with transformation

If the indicator of interest needs a transformation, either log or arcsin, in addition to the function used in the previous subsection, the arguments transformation and backtransformation must be specified. If, for example, the share of households per area that earn more than the national median income (MTMED) is the indicator of interest, an arcsin transformation can be used. The bias-corrected back-transformation bc is chosen in the example. Two more arguments are needed when using an arcsin transformation: the name of the variable describing the effective sample sizes (eff_smpsize) which needs to be contained in the combined_data frame. Because of having chosen the bias-corrected back-transformation, the only possible mse_type is boot, if the MSE estimation is activated.

```
> fh_arcsin <- fh(fixed = MTMED ~ cash + age_ben + rent + house_allow,
+   vardir = "Var_MTMED", combined_data = combined_data, domains = "Domain",
+   transformation = "arcsin", backtransformation = "bc", eff_smpsize = "n",
+   MSE = TRUE, mse_type = "boot")
```

Spatial FH model

If the spatial correlation tests indicated a spatial correlation of the domains, a spatial FH model for incorporating the spatial structure in the model could be used. For that, the correlation has to be set to `spatial` and the example proximity matrix has to be given to the model within the `corMatrix` argument. The possible variance estimation methods are `ml` and `reml`.

```
> fh_spatial <- fh(fixed = Mean ~ cash + self_empl, vardir = "Var_Mean",
+   combined_data = combined_data, domains = "Domain", correlation = "spatial",
+   corMatrix = eusilcA_prox, MSE = TRUE)
```

Robust FH model

If extreme values could influence the estimation, the application of a robust model might be appropriate. Within the robust framework, package **emdi** allows the user to choose between a standard and a spatial model (defaults to `correlation = "no"`). The estimation method must be `reblup` or `reblupbc` which includes a bias correction that can be modified by the argument `mult_constant`. Further, the tuning constant `k` defaults to 1.345 as proposed by Sinha and Rao (2009) and Warnholz (2016) and can be changed if desired. The functions of the package **saeRobust** are utilized for the robust extensions. An exemplary call with pseudolinear MSE estimation looks like this:

```
> fh_robust <- fh(fixed = Mean ~ cash + self_empl, vardir = "Var_Mean",
+   combined_data = combined_data, domains = "Domain", method = "reblup",
+   MSE = TRUE, mse_type = "pseudo")
```

Measurement error model

If other data sources than register data, e.g., data from larger surveys or big data sources are used as auxiliary information, the ME model should be applied. For the estimation of the ME model, the model fitting method must be set to `me` and the only possible MSE estimation method is `jackknife`. The most complex input argument consists of the creation of the MSE array `Ci`. The variability of the auxiliary variables that is taken into account by the ME model is expressed by the variance-covariance matrices per domain (`Ci`). For example, for three covariates `a`, `b` and `c` the array should look like

$$\boldsymbol{C}_i = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \mathrm{var}_i(a) & \mathrm{cov}_i(a,b) & \mathrm{cov}_i(a,c) \\ 0 & \mathrm{cov}_i(a,b) & \mathrm{var}_i(b) & \mathrm{cov}_i(b,c) \\ 0 & \mathrm{cov}_i(a,c) & \mathrm{cov}_i(b,c) & \mathrm{var}_i(c) \end{pmatrix}, \; i = 1, ..., D.$$

The first row and column contain zeros, because the intercept is considered. The variances and covariances can be computed by standard approaches like, for example, the Horvitz-Thompson estimator.

For the Austrian EUSILC data example, the equalized income can also be explained by a variable of the sample data set. The code below demonstrates how the MSE array `Ci` is created for one covariate (variable `Cash` and its variance `Var_Cash`) and how the final ME model is built.

```
> P <- 1
> M <- 94
> Ci_array <- array(data = 0, dim = c(P + 1, P + 1, M))
> Ci_array[2,2, ] <- eusilcA_smpAgg$Var_Cash

> fh_yl <- fh(fixed = Mean ~ Cash, vardir = "Var_Mean",
+   combined_data = eusilcA_smpAgg, domains = "Domain", method = "me",
+   Ci = Ci_array, MSE = TRUE, mse_type = "jackknife")
```

5 Conclusion and outlook

In this paper, we have presented how the **emdi** package version 1.1.7 has been extended with various area-level models. Along with the well-known FH model, adjusted variance estimation methods and transformation options are offered to the user. In addition, spatial, robust, and ME model extensions of the standard model allow the user to address various issues that arise in practical data applications. All of these methods can be estimated conveniently by using a single function that provides EBLUP and the respective MSE estimates to measure their precision. Especially in the section Functionality and case studies, it is clear that the package does not only contain tools for estimation of the different SAE models. Instead, it additionally provides user-friendly tools to enable a whole data analysis procedure: 1. starting with model building and estimation, moving on to 2. model assessment and diagnostics, 3. presentation of the results, and finishing with 4. exporting the results to Excel or OpenDocument Spreadsheet.

For future package versions, it is planned to expand the options in the field of area-level models. In some practical applications, the incorporation of random effects is redundant. Therefore, an area-level estimator that considers a preliminary testing for the random effects following Molina et al. (2015)

will be included. Since version 2.0.0 **emdi** accounts for spatial structures of the random effects. Future developments may also account for out-of-sample EBLUP and MSE estimation for the spatial model proposed by Saei and Chambers (2005) and for temporal and spatio-temporal extensions (Rao and Yu, 1994; Marhuenda et al., 2013). For the existing ME model, a bootstrap MSE estimation option may be added to the package since the Jackknife MSE estimator may produce negative MSE estimates (Marchetti et al., 2015). Furthermore, cross-validation options additional to the model assessment via information criteria and the R^2 will be investigated.

6 Acknowledgments

The work of Kreutzmann and Schmid has been supported by the German Research Foundation within the project QUESSAMI (281573942) and by the MIUR-DAAD Joint Mobility Program (57265468). The numerical results are not official estimates and are only produced for illustrating the methods. The authors are indebted to the Editor-in-Chief, Associate Editor and the referees for comments that significantly improved the article.

Bibliography

A. Alfons and M. Templ. Estimation of social exclusion indicators from complex surveys: The R package laeken. *Journal of Statistical Software*, 54(15):1–25, 2013. URL https://doi.org/10.18637/jss.v054.i15. [p3]

A. Alfons, M. Templ, and P. Filzmoser. An object-oriented framework for statistical simulation: The R package simFrame. *Journal of Statistical Software*, 37(3):1–36, 2010. URL https://doi.org/10.18637/jss.v037.i03. [p7]

G. E. Battese, R. M. Harter, and W. A. Fuller. An error-components model for prediction of county crop areas using survey and satellite data. *Journal of the American Statistical Association*, 83(401):28–36, 1988. URL https://doi.org/10.1080/01621459.1988.10478561. [p2]

G. Bertarelli, F. Schirripa Spagnolo, N. Salvati, and M. Pratesi. Small area estimation of agricultural data. In *Spatial Econometric Methods in Agricultural Economics Using R*. CRC book, 2021. [p5]

R. S. Bivand and D. W. S. Wong. Comparing implementations of global and local indicators of spatial association. *TEST*, 27(3):716–748, 2018. URL https://doi.org/10.1007/s11749-018-0599-x. [p10]

H. J. Boonstra. *mcmcsae: Markov Chain Monte Carlo Small Area Estimation*, 2021. URL https://CRAN.R-project.org/package=mcmcsae. R package version 0.7.0. [p1]

H. J. Boonstra. *hbsae: Hierarchical Bayesian Small Area Estimation*, 2022. URL https://CRAN.R-project.org/package=hbsae. R package version 1.2. [p1]

J. Breidenbach. *JoSAE: Unit-Level and Area-Level Small Area Estimation*, 2018. URL https://CRAN.R-project.org/package=JoSAE. R package version 0.3.0. [p1]

G. Brown, R. Chambers, P. Heady, and D. Heasman. Evaluation of small area estimation methods - an application to unemployment estimates from the UK LFS. In *Proceedings of Statistics Canada Symposium*, 2001. [p13, 14]

Bundesamt für Eich- und Vermessungswesen. Verwaltungsgrenzen (VGD) - 1:250.000 Bezirksgrenzen, Daten vom 01.04.2017 von SynerGIS, 2017. URL http://data-synergis.opendata.arcgis.com/datasets/bb4acc011100469185d2e59fa4cae5fc_0. [accessed: 07.02.2018]. [p8]

C. Casas-Cordero, J. Encina, and P. Lahiri. Poverty mapping for the chilean comunas. In *Analysis of Poverty by Small Area Estimation*, pages 379–403. John Wiley & Sons, 2016. URL https://doi.org/10.1002/9781118814963.ch20. [p5, 11]

J. Chambers and T. Hastie, editors. *Statistical Models in S*. Chapman & Hall, London, 1992. [p8]

R. Chambers, H. Chandra, N. Salvati, and N. Tzavidis. Outlier robust small area estimation. *Journal of the Royal Statistical Society B*, 76(1):47–69, 2014. URL https://doi.org/10.1111/rssb.12019. [p5]

H. Chandra, N. Salvati, and R. Chambers. A spatially nonstationary Fay-Herriot model for small area estimation. *Journal of the Survey Statistics and Methodology*, 3(2):109–135, 2015. URL https://doi.org/10.1093/jssam/smu026. [p14]

H. Chandra, N. Salvati, R. Chambers, and S. Guha. *NSAE: Nonstationary Small Area Estimation*, 2022. URL https://CRAN.R-project.org/package=NSAE. R package version 0.4.0. [p2]

S. Chen and P. Lahiri. A weighted jackknife mspe estimator in small-area estimation. In *Proceeding of the Section on Survey Research Methods*, pages 473–477, 2002. American Statistical Association. [p6]

A. Cliff and J. Ord. *Spatial Processes: Models and Applications*. Pion, London, 1981. [p5]

G. Datta, M. Ghosh, R. Steorts, and J. Maples. Bayesian benchmarking with applications to small area estimation. *TEST*, 20(3):574–588, 2011. URL https://doi.org/10.1007/s11749-010-0218-y. [p15]

G. S. Datta and P. Lahiri. A unified measure of uncertainty of estimated best linear unbiased predictors in small area estimation problems. *Statistica Sinica*, 10(2):613–627, 2000. URL http://www.jstor.com/stable/24306735. [p6]

G. S. Datta, R. E. Fay, and M. Ghosh. Hierarchical and empirical bayes multivariate analysis in small area estimation. In *Proceedings of Bureau of the Census 1991 Annual Research Conference*, pages 63–79. US Bureau of the Census, 1991. [p1]

S. De Nicolò and A. Gardini. *tipsae: Tools for Handling Indices and Proportions in Small Area Estimation*, 2022. URL https://CRAN.R-project.org/package=tipsae. R package version 0.0.6. [p1]

C. S. Developer. *BayesSAE: Bayesian Analysis of Small Area Estimation*, 2018. URL https://CRAN.R-project.org/package=BayesSAE. R package version 1.0-2. [p1]

A. Fasulo. *SAEval: Small Area Estimation Evaluation*, 2022. URL https://CRAN.R-project.org/package=SAEval. R package version 0.1.5. [p2]

R. E. Fay and R. A. Herriot. Estimates of income for small places: An application of James-Stein procedures to census data. *Journal of the American Statistical Association*, 74(366):269–277, 1979. URL https://doi.org/10.1080/01621459.1979.10482505. [p1, 3, 5]

S. Hadam, N. Würz, and A.-K. Kreutzmann. Estimating regional unemployment with mobile network data for functional urban areas in Germany. *Refubium - Freie Universität Berlin Repository*, pages 1–28, 2020. URL https://doi.org/10.17169/refubium-26791. [p5, 6]

A. Hagenaars, K. de Vos, and M. Zaidi. *Poverty Statistics in the Late 1980s: Research Based on Mirco-data*. Office for the Official Publications of the European Communities, 1994. [p7]

D. Horvitz and D. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260):663–685, 1952. URL https://doi.org/10.1080/01621459.1952.10483446. [p3]

J. Jiang and J. S. Rao. Robust small area estimation: An overview. *Annual Review of Statistics and Its Application*, 7:337–360, 2020. URL https://doi.org/10.1146/annurev-statistics-031219-041212. [p1]

J. Jiang, P. Lahiri, S.-M. Wan, and C.-H. Wu. Jackknifing in the Fay-Herriot model with an example. In *Proceedings of the Seminar on Funding Opportunity in Survey Research Council of Professional Associations on Federal Statistics*, pages 75–97, 2001. [p5, 6]

J. Jiang, P. Lahiri, and S.-M. Wan. A unified jackknife theory for empirical best prediction with m-estimation. *The Annals of Statistics*, 30(6):1782–1810, 2002. URL https://doi.org/10.1214/aos/1043351257. [p6]

L. Komsta and F. Novomestky. *moments: Moments, cumulants, skewness, kurtosis and related tests*, 2015. URL https://CRAN.R-project.org/package=moments. R package version 0.14. [p12]

A.-K. Kreutzmann, S. Pannier, N. Rojas-Perilla, T. Schmid, M. Templ, and N. Tzavidis. The R package emdi for estimating and mapping regionally disaggregated indicators. *Journal of Statistical Software*, 91(7):1–33, 2019. URL https://doi.org/10.18637/jss.v091.i07. [p6, 15, 16]

A.-K. Kreutzmann, P. Marek, M. Runge, N. Salvati, and T. Schmid. The Fay-Herriot model for multiply imputed data with an application to regional wealth estimation in Germany. *Journal of Applied Statistics*, 49(13):3278–3299, 2022. URL https://doi.org/10.1080/02664763.2021.1941805. [p4]

P. Lahiri and J. Suntornchost. Variable selection for linear mixed models with applications in small area estimation. *The Indian Journal of Statistics*, 77-B(2):312–320, 2015. URL https://www.jstor.org/stable/43694416. [p12]

H. Li and P. Lahiri. An adjusted maximum likelihood method for solving small area estimation problems. *Journal of Multivariate Analyis*, 101(4):882–902, 2010. URL https://doi.org/10.1016/j.jmva.2009.10.009. [p4, 6]

E. Lopez-Vizcaino, M. Lombardia, and D. Morales. *mme: Multinomial Mixed Effects Models*, 2019. URL https://CRAN.R-project.org/package=mme. R package version 0.1-6. [p2]

S. Marchetti, C. Giusti, M. Pratesi, N. Salvati, F. Giannotti, D. Pedreschi, S. Rinzivillo, L. Pappalardo, and L. Gabrielli. Small area model-based estimators using big data sources. *Journal of Official Statistics*, 31(2):263–281, 2015. URL https://doi.org/10.1515/jos-2015-0017. [p18]

Y. Marhuenda, I. Molina, and D. Morales. Small area estimation with spatio-temporal Fay-Herriot models. *Computational Statistics and Data Analysis*, 58:308–325, 2013. URL https://doi.org/10.1016/j.csda.2012.09.002. [p18]

Y. Marhuenda, D. Morales, and M. del Camen Pardo. Information criteria for Fay-Herriot model selection. *Computational Statistics and Data Analysis*, 70:268–280, 2014. URL https://doi.org/10.1016/j.csda.2013.09.016. [p8, 11, 12, 24]

M. Miltiadou. Measuring and reporting reliability of labour force survey and annual population survey estimates force survey and annual population survey estimates, 2020. URL https://www.ons.gov.uk/employmentandlabourmarket/peopleinwork/employmentandemployeetypes/methodologies/measuringandreportingreliabilityoflabourforcesurveyandannualpopulationsurveyestimates. UK Office for National Statistics, [accessed: 05.06.2020]. [p14]

I. Molina and Y. Marhuenda. sae: An R package for small area estimation. *The R Journal*, 7(1):81–98, 2015. URL https://doi.org/10.32614/rj-2015-007. [p2]

I. Molina and J. Rao. Small area estimation of poverty indicators. *The Canadian Journal of Statistics*, 38 (3):369–385, 2010. URL https://doi.org/10.1002/cjs.10051. [p1]

I. Molina, N. Salvati, and M. Pratesi. Bootstrap for estimating the mse of the spatial eblup. *Computational Statistics*, 24:441–458, 2009. URL https://doi.org/10.1007/s00180-008-0138-4. [p6]

I. Molina, J. Rao, and G. Datta. Small area estimation under a Fay-Herriot model with preliminary testing for the presence of random area effects. *Survey Methodology*, 41(1):1–19, 2015. URL https://www150.statcan.gc.ca/n1/pub/12-001-x/2015001/article/14161-eng.htm. [p17]

M. Mubarak and A. Ubaidillah. *saeME: Small Area Estimation with Measurement Error*, 2022. URL https://CRAN.R-project.org/package=saeME. R package version 1.3. [p2]

A. Neves, D. Silva, and S. Correa. Small domain estimation for the Brazilian service sector survey. *ESTADÍSTICA*, 65(185):13–37, 2013. [p4]

E. J. Pebesma and R. S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, November 2005. URL https://CRAN.R-project.org/doc/Rnews/. [p10]

N. Permatasari and A. Ubaidillah. *msae: Multivariate Fay Herriot Models for Small Area Estimation*, 2022. URL https://CRAN.R-project.org/package=msae. R package version 0.1.5. [p1]

A. Petrucci and N. Salvati. Small area estimation for spatial correlation in watershed erosion assessment. *Journal of Agricultural, Biological and Environmental Statistics*, 11(2):169–182, 2006. URL https://doi.org/10.1198/108571106X110531. [p5]

D. Pfeffermann. New important developments in small area estimation. *Statistical Science*, 28(1):40–68, 2013. URL https://doi.org/10.1214/12-STS395. [p1]

N. Prasad and J. Rao. The estimation of the mean squared error of small-area estimation. *Journal of the American Statistical Association*, 85(409):163–171, 1990. URL https://doi.org/10.1080/01621459.1990.10475320. [p6]

M. Pratesi, editor. *Analysis of Poverty Data by Small Area Estimation*. John Wiley & Sons, 2016. URL https://doi.org/10.1002/9781118814963. [p1]

M. Pratesi and N. Salvati. Small area estimation: The eblup estimator based on spatially correlated random area effects. *Statistical Methods and Applications*, 17(1):113–141, 2008. URL https://doi.org/10.1007/s10260-007-0061-9. [p5]

J. N. K. Rao and I. Molina. *Small Area Estimation*. John Wiley & Sons, 2015. URL https://doi.org/10.1002/9781118735855. [p1, 3, 6]

J. N. K. Rao and M. Yu. Small-area estimation by combining time-series and cross-sectional data. *The Canadian Journal of Statistics*, 22(4):511–528, 1994. URL https://doi.org/10.2307/3315407. [p18]

L.-P. Rivest and N. Vandal. Mean squared error estimation for small areas when the small area variances are estimated. In *Proceedings of International Conference of Recent Advanced Survey Sampling*, pages 197–206, 2003. [p3]

A. Saei and R. Chambers. Out of sample estimation for small areas using area level data. *Southampton Statistical Sciences Research Institute Methodology Working Paper*, M05/11, 2005. URL http://eprints.soton.ac.uk/id/eprint/14327. Southampton Statistical Sciences Research Institute, UK. [p18]

T. Schmid, N. Tzavidis, R. Münnich, and R. Chambers. Outlier robust small area estimation under spatial correlation. *Scandinavian Journal of Statistics*, 43(3):806–826, 2016. URL https://doi.org/10.1111/sjos.12205. [p5]

T. Schmid, F. Bruckschen, N. Salvati, and T. Zbiranski. Constructing sociodemographic indicators for national statistical institutes using mobile phone data: Estimating literacy rates in Senegal. *Journal of the Royal Statistical Society A*, 180(4):1163–1190, 2017. URL https://doi.org/10.1111/rssa.12305. [p5, 11]

B. B. Singh, K. Shukla, and D. Kundu. Spatio-temporal models in small area estimation. *Survey Methodology*, 31(2):183–195, 2005. URL https://www150.statcan.gc.ca/n1/en/catalogue/12-001-X20050029053. [p6]

S. Sinha and J. Rao. Robust small area estimation. *The Canadian Journal of Statistics*, 37(3):381–399, 2009. URL https://doi.org/10.1002/cjs.10029. [p5, 17]

E. Slud and T. Maiti. Mean-squared error estimation in transformed Fay-Herriot models. *Journal of the Royal Statistical Society B*, 68(2):239–257, 2006. URL https://doi.org/10.1111/j.1467-9868.2006.00542.x. [p4, 6]

S. Sugawasa and T. Kubokawa. Transforming response values in small area prediction. *Computational Statistics and Data Analysis*, 114:47–60, 2017. URL https://doi.org/10.1016/j.csda.2017.03.017. [p4, 5]

N. Tzavidis, R. Chambers, N. Salvati, and H. Chandra. Small area estimation in practice an application to agricultural business survey data. *Journal of the Indian Society of Agricultural Statistics*, 66(1): 213–228, 2012. URL https://ro.uow.edu.au/eispapers/758/. [p1]

N. Tzavidis, L.-C. Zhang, A. Luna Hernandez, T. Schmid, and N. Rojas-Perilla. From start to finish: A framework for the production of small area official statistics. *Journal of the Royal Statistical Society A*, 181(4):927–979, 2018. URL https://doi.org/10.1111/rssa.12364. [p1]

K. Ushey, J. McPherson, J. Cheng, A. Atkins, and J. Allaire. *packrat: A Dependency Management System for Projects and their R Package Dependencies*, 2022. URL https://CRAN.R-project.org/package=packrat. R package version 0.8.0. [p25]

J. Wang and W. A. Fuller. The mean squared error of small area predictors constructed with estimated area variances. *Journal of the American Statistical Association*, 98:716–723, 2003. URL https://doi.org/10.1198/016214503000000620. [p3]

S. Warnholz. *Small Area Estimation Using Robust Extensions to Area Level Models*. PhD thesis, Freie Universität Berlin, 2016. URL https://doi.org/10.17169/refubium-13904. [p5, 6, 17]

S. Warnholz. *saeRobust: Robust Small Area Estimation*, 2022. URL https://CRAN.R-project.org/package=saeRobust. R package version 0.3.0. [p1]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. URL https://ggplot2.tidyverse.org. [p12]

P. Xiao, X. Liu, Y. Liu, and S. Liu. *saeMSPE: Compute MSPE Estimates for the Fay Herriot Model and Nested Error Regression Model*, 2022. URL https://CRAN.R-project.org/package=saeMSPE. R package version 1.0. [p2]

L. M. R. Ybarra and S. L. Lohr. Small area estimation when auxiliary information is measured with error. *Biometrika*, 95(4):919–931, 2008. URL https://doi.org/10.1093/biomet/asn048. [p6]

M. Yoshimori and P. Lahiri. A new adjusted maximum likelihood method for the Fay-Herriot small area model. *Journal of Multivariate Analysis*, 124:281–294, 2014. URL https://doi.org/10.1016/j.jmva.2013.10.012. [p4]

Y. You and B. Chapman. Small area estimation using area level models and estimated sampling variances. *Survey Methodology*, 32(1):97–103, 2006. URL https://www150.statcan.gc.ca/n1/en/catalogue/12-001-X20060019263. [p3]

X. Zhang, J. Holt, S. Yun, H. Lu, K. Greenlund, and J. Croft. Validation of multilevel regression and poststratification methodology for small area estimation of health indicators from the behavioral risk factor surveillance system. *American Journal of Epidemiology*, 182(2):127–137, 2015. URL https://doi.org/10.1093/aje/kwv002. [p1]

7 Appendix A: Area-level model options and corresponding input arguments

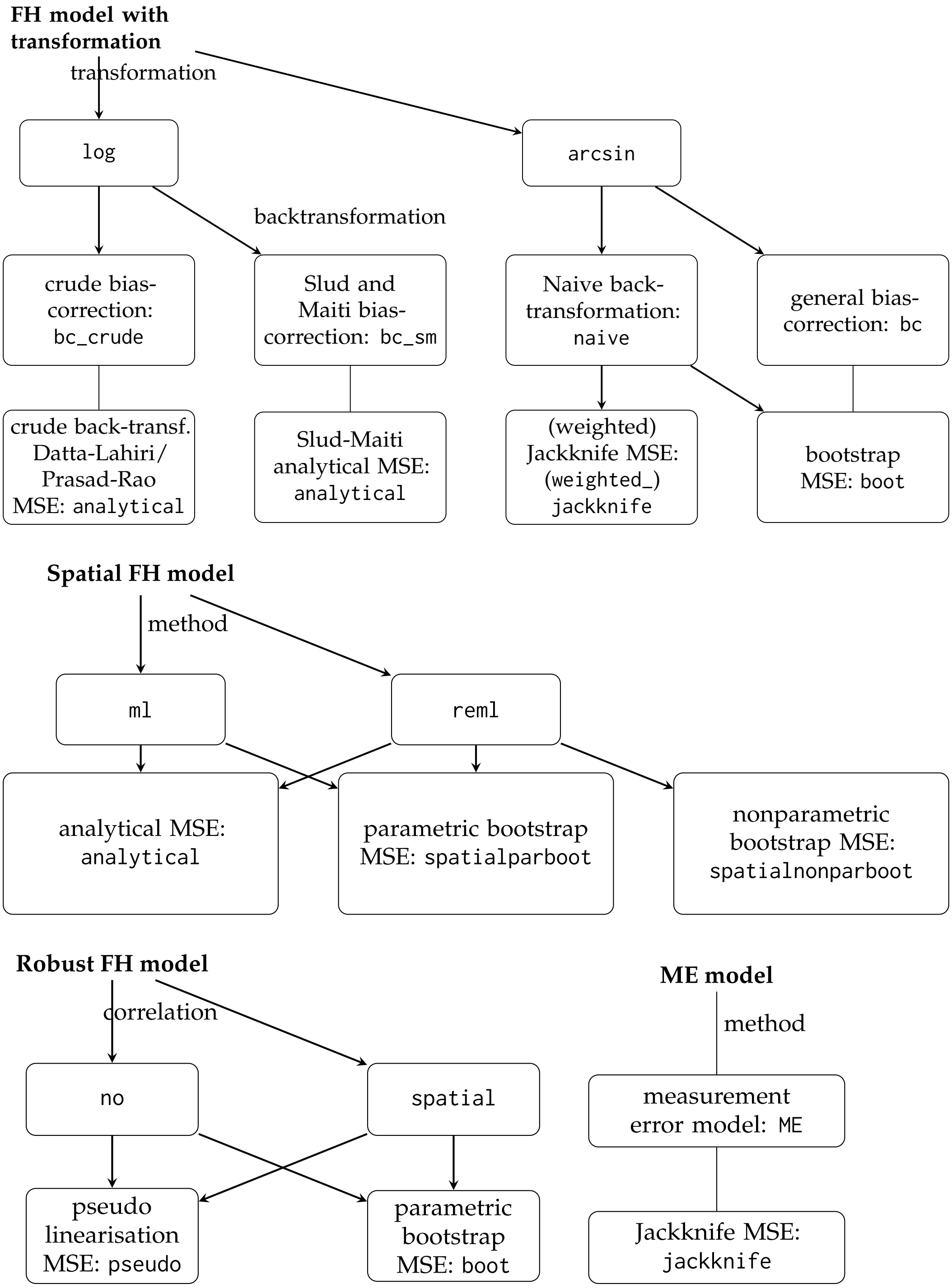

FH model with transformation
transformation
log
arcsin
backtransformation
crude bias-correction: bc_crude
Slud and Maiti bias-correction: bc_sm
Naive back-transformation: naive
general bias-correction: bc
crude back-transf. Datta-Lahiri/ Prasad-Rao MSE: analytical
Slud-Maiti analytical MSE: analytical
(weighted) Jackknife MSE: (weighted_) jackknife
bootstrap MSE: boot
Spatial FH model
method
ml
reml
analytical MSE: analytical
parametric bootstrap MSE: spatialparboot
nonparametric bootstrap MSE: spatialnonparboot
Robust FH model
correlation
no
spatial
pseudo linearisation MSE: pseudo
parametric bootstrap MSE: boot
ME model
method
measurement error model: ME
Jackknife MSE: jackknife

Figure 6: Overview of extended area-level models and combinations of estimation methods.

Argument			FH model		
	Standard	Transformed	Spatial	Robust	ME
`fixed`	√	√	√	√	√
`vardir`	√	√	√	√	√
`combined_data`	√	√	√	√	√
`domains`	(√)	(√)	(√)	(√)	(√)
`method`	√	√	√	√	√
`interval`	(√)	(√)			
`k`				√	
`mult_constant`				√	
`transformation`	√	√	√	√	√
`backtransformation`		√			
`eff_smpsize` (only if `transformation = "arcsin"`)		√			
`correlation`	√	√	√	√	√
`corMatrix` (only if `correlation = "spatial"`)			√	√	
`Ci`					√
`tol`			√	√	√
`maxit`			√	√	√
`MSE`	√	√	√	√	√
`mse_type` (only if `MSE = TRUE`)	√	√	√	√	√
`B`	(√)	√	√	√	
`seed`	(√)	(√)	(√)	(√)	

Table 6: Required √ and optional (√) input arguments of function `fh` for the different area-levels models. `B`: Only if bootstrap MSE is chosen. When the standard FH model is applied, `B` is required for the computation of the information criteria by Marhuenda et al. (2014) (optionally).

8 Appendix B: Output of the model component

Name	Short description			Available for		
		Standard	Transformed	Spatial	Robust	ME
`coefficients`	Estimated regression coefficients	√	√	√	√	√
`variance`	Estimated variance of the random effects/ estimated spatial correlation parameter	√	√	√	√	√
`random_effects`	Random effects per domain	√	√	√	√	√
`real_residuals`	Realized residuals per domain	√	√	√	√	√
`std_real_residuals`	Standardized realized residuals per domain	√	√	√	√	√
`gamma`	Shrinkage factors per domain	√	√			√
`model_select`	Model selection and accuracy criteria	√	√	√		
`correlation`	Selected correlation structure of the random effects	√	√	√	√	√
`k`	Tuning constant				√	
`mult_constant`	Multiplier constant for bias correction				√	
`seed`	Seed of the random number generator	√	√	√	√	

Table 7: Components of the output component `model` for models of class `"fh"`.

9 Reproducibility

For the computation of the results in this paper we worked with R version 4.2.2 on a 64-bit platform under Microsoft Windows 10 with the installed packages listed in Table 8. Using the package **packrat** (Ushey et al., 2022) a snapshot of the corresponding repository was created that is available from the GitHub folder (`https://github.com/SoerenPannier/emdi.git`). We suggest the following steps:

- Install Git.
- Create a new project in RStudio.
- Choose checkout from version control and select Git.
- Insert the repository URL: `https://github.com/SoerenPannier/emdi.git`.
- Let **packrat** complete the initialization process.
- Restart RStudio.
- Enter the R command `packrat::restore()`.
- After finishing the installation process all packages are installed as provided in Table 8.

Sylvia Harmening
Institute for Statistics and Econometrics, School of Business & Economics, Freie Universität Berlin
Garystr. 21, 14195 Berlin
Germany
`sylvia.harmening@fu-berlin.de`

Ann-Kristin Kreutzmann
Institute for Statistics and Econometrics, School of Business & Economics, Freie Universität Berlin
Garystr. 21, 14195 Berlin
Germany
`ann-kristin.kreutzmann@fu-berlin.de`

Sören Schmidt
Institute for Statistics and Econometrics, School of Business & Economics, Freie Universität Berlin
Garystr. 21, 14195 Berlin
Germany
`soeren.pannier@fu-berlin.de`

Nicola Salvati
Department of Economics and Management, University of Pisa
Via C. Ridolfi, 10 56124 Pisa
Italy
`nicola.salvati@unipi.it`

Timo Schmid
Institute of Statistics, Otto-Friedrich-Universität Bamberg
Feldkirchenstr. 21, 96052 Bamberg
Germany
`timo.schmid@uni-bamberg.de`

Package	Version	Package	Version	Package	Version
aoos	0.5.0	highr	0.9	RColorBrewer	1.1-3
assertthat	0.2.1	HLMdiag	0.5.0	Rcpp	1.0.9
backports	1.4.1	hms	1.1.1	RcppArmadillo	0.11.2.0.0
BBmisc	1.12	isoband	0.2.5	readODS	1.7.0
bit	4.0.4	janitor	2.1.0	readr	2.1.2
bit64	4.0.5	jsonlite	1.8.0	rematch	1.0.1
boot	1.3-28	knitr	1.39	rematch2	2.1.2
brew	1.0-7	labeling	0.4.2	reshape2	1.4.4
brio	1.1.3	laeken	0.5.2	rgeos	0.5-9
cachem	1.0.6	lifecycle	1.0.1	rlang	1.0.4
callr	3.7.1	lubridate	1.8.0	roxygen2	7.2.1
cellranger	1.1.0	magrittr	2.0.3	rprojroot	2.0.3
checkmate	2.1.0	maptools	1.1-4	s2	1.1.0
classInt	0.4-7	MASS	7.3-58	saeRobust	0.3.0
cli	3.3.0	memoise	2.0.1	scales	1.2.0
clipr	0.8.0	modules	0.10.0	sf	1.0-8
colorspace	2.0-3	moments	0.14.1	simFrame	0.5.4
commonmark	1.8.0	MuMIn	1.47.1	snakecase	0.11.0
cpp11	0.4.2	munsell	0.5.0	sp	1.5-0
crayon	1.5.1	nlme	3.1-158	spData	2.0.1
data.table	1.14.2	openxlsx	4.2.5	spdep	1.2-4
DBI	1.1.3	operator.tools	1.6.3	stringi	1.7.8
deldir	1.0-6	packrat	0.8.1	stringr	1.4.0
desc	1.4.1	parallelMap	1.5.1	terra	1.5-34
diagonals	6.4.0	pbapply	1.5-0	testthat	3.1.4
diffobj	0.3.5	pillar	1.8.0	tibble	3.1.8
digest	0.6.29	pkgconfig	2.0.3	tidyr	1.2.0
dplyr	1.0.9	pkgload	1.3.0	tidyselect	1.1.2
e1071	1.7-11	plyr	1.8.7	tzdb	0.3.0
ellipsis	0.3.2	praise	1.0.0	units	0.8-0
emdi	2.1.3	prettyunits	1.1.1	utf8	1.2.2
evaluate	0.15	processx	3.7.0	vctrs	0.4.1
fansi	1.0.3	progress	1.2.2	viridisLite	0.4.0
farver	2.1.1	proxy	0.4-27	vroom	1.5.7
fastmap	1.1.0	ps	1.7.1	waldo	0.4.0
formula.tools	1.7.1	purrr	0.3.4	withr	2.5.0
fs	1.5.2	R.cache	0.16.0	wk	0.6.0
generics	0.1.3	R.methodsS3	1.8.2	xfun	0.31
ggplot2	3.3.6	R.oo	1.25.0	xml2	1.3.3
ggrepel	0.9.1	R.rsp	0.45.0	yaml	2.3.5
glue	1.6.2	R.utils	2.12.0	zip	2.2.0
gridExtra	2.3	R6	2.5.1		
gtable	0.3.0	raster	3.5-21		

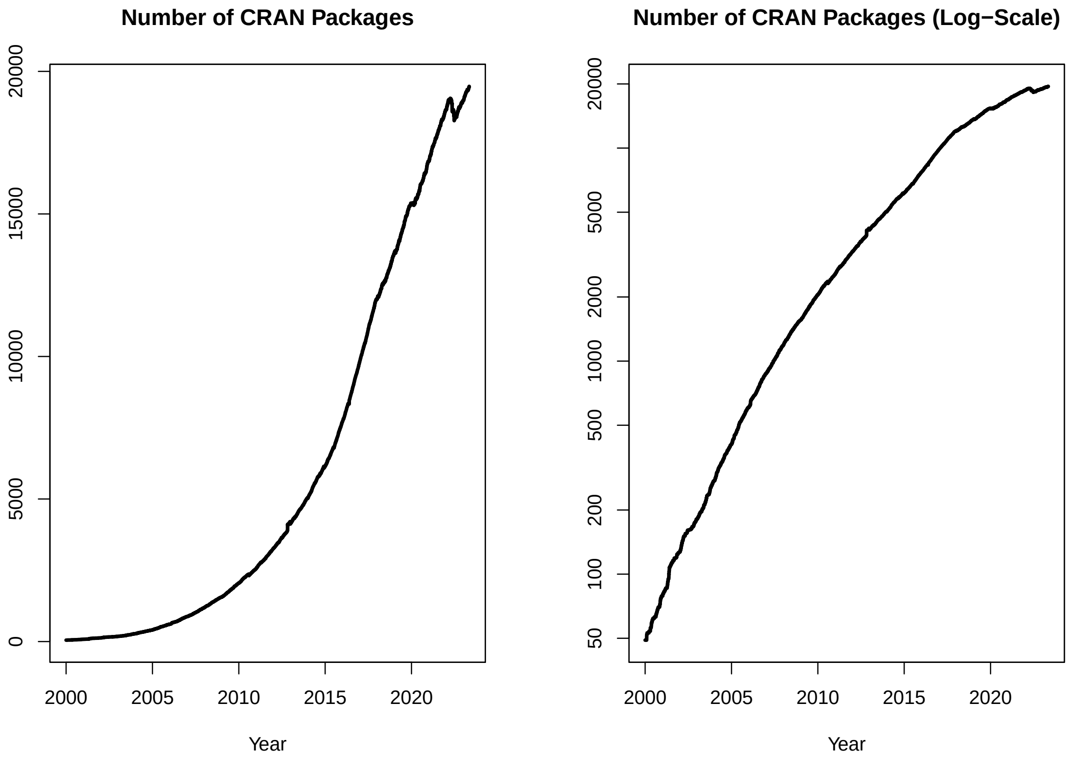
Table 8: Installed packages for the computation of the results in this paper.

Changes on CRAN

2023-02-01 to 2023-04-30

by Kurt Hornik, Uwe Ligges, and Achim Zeileis

In the past 3 months, 516 new packages were added to the CRAN package repository. 206 packages were unarchived, 416 were archived and 4 had to be removed. The following shows the growth of the number of active packages in the CRAN package repository:



On 2023-04-30, the number of active packages was around 19439.

CRAN package submissions

From February 2023 to April 2023 CRAN received 7905 package submissions. For these, 13597 actions took place of which 9071 (67%) were auto processed actions and 4526 (33%) manual actions.

Minus some special cases, a summary of the auto-processed and manually triggered actions follows:

	archive	inspect	newbies	pending	pretest	publish	recheck	waiting
auto	2276	1725	1370	0	0	2300	787	613
manual	1809	94	361	175	73	1497	406	111

These include the final decisions for the submissions which were

	archive	publish
auto	2126 (27.6%)	1977 (25.7%)
manual	1788 (23.2%)	1808 (23.5%)

where we only count those as *auto* processed whose publication or rejection happened automatically in all steps.

CRAN mirror security

Currently, there are 100 official CRAN mirrors, 80 of which provide both secure downloads via '`https`' *and* use secure mirroring from the CRAN master (via rsync through ssh tunnels). Since the R 3.4.0 release, `chooseCRANmirror()` offers these mirrors in preference to the others which are not fully secured (yet).

CRAN Task View Initiative

There is one new task view:

- Genomics, Proteomics, Metabolomics, Transcriptomics, and Other Omics: Maintained by Julie Aubert, Toby Dylan Hocking, Nathalie Vialaneix.

Currently there are 43 task views (see `https://cran.r-project.org/web/views/`), with median and mean numbers of CRAN packages covered 106 and 120, respectively. Overall, these task views cover 4344 CRAN packages, which is about 22% of all active CRAN packages.

Kurt Hornik
WU Wirtschaftsuniversität Wien
Austria
ORCiD: 0000-0003-4198-9911
`Kurt.Hornik@R-project.org`

Uwe Ligges
TU Dortmund
Germany
ORCiD: 0000-0001-5875-6167
`Uwe.Ligges@R-project.org`

Achim Zeileis
Universität Innsbruck
Austria
ORCiD: 0000-0003-0918-3766
`Achim.Zeileis@R-project.org`

R Foundation News

by Torsten Hothorn

1 Donations and members

Membership fees and donations received between 2023-02-20 and 2023-05-09.

Donations

Gilberto Camara (Brazil) Korea R User Group (Korea, Republic of) Nickalus Redell (United States) Scrub the web (Poland) Rav Vaid (United States)

Supporting institutions

Alfred Mueller Analytic Services, München (Germany) Department of Clinical Research, University of Basel, Switzerland, Basel (Switzerland) Ef-prime, Inc., Tokyo (Japan)

Supporting members

Ashanka Beligaswatte (Australia) Frederic Bertrand (France) Alistair Cullum (United States) Guenter Faes (Germany) Bernd Fröhlich (Germany) Dejan Gregor (United Kingdom) Heidi Imker (United States) Knut Helge Jensen (Norway) Christian Kampichler (Netherlands) Katharina Kesy (Germany) Ziyad Knio (United States) Sebastian Koehler (Germany) Sebastian Krantz (Germany) Chris Kuty (United States) Luca La Rocca (Italy) Teemu Daniel Laajala (Finland) Thierry Lecerf (Switzerland) Eric Lim (United Kingdom) Michal Majka (Austria) Myriam Maumy (France) Ernst Molitor (Germany) David Monterde (Spain) Stefan Moog (Germany) Steffen Moritz (Germany) Maciej Nasinski (Poland) Jens Oehlschlägel (Germany) Harald Sterly (Germany) Robert van den Berg (Austria) Thomas van der Vaart (Netherlands) Fredrik Wartenberg (Sweden) Jason Wyse (Ireland)

Torsten Hothorn
Universität Zürich
Switzerland
ORCiD: 0000-0001-8301-0471
Torsten.Hothorn@R-project.org