The R Journal

Volume 17/3, September 2025

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

Contents

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 Rob J Hyndman

In this issue

On behalf of the editorial board, I am pleased to present Volume 17 Issue 3 of the R Journal. This issue features eleven research articles, plus news from CRAN and the R Foundation.

Article 2025-022 by Abhishek Ulayil, Dianne Cook, Heather Turner, Mitchell O'Hara-Wild, and Christophe Dervieux describes an R Journal initiative to convert legacy articles into R markdown to allow for better accessibility for print-disabled readers. The associated R markdown format provided by the **rjtools** package also encourages better reproducibility and enables interactivity for future submissions. We continue to use the tooling described in this article to convert LaTeX submissions to the journal into R markdown.

A review of image processing packages in R is provided by article 2025-030, covering 38 packages for image analysis published between 2006 and 2024.

The remaining nine articles in this issue describe various R packages developed by members of the R community, and available on CRAN. Each article provides an overview of the package, its functionality, and examples of its use.

Supplementary material for each article, with fully reproducible code, is available for download from the Journal website.

Rob J Hyndman
Monash University

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

lsirm12pl: An R Package for the Latent Space Item Response Model

Dongyoung Go, Gwanghee Kim, Jina Park, Junyong Park, Minjeong Jeon, Ick Hoon Jin

Abstract The item response model in latent space (LSIRM; Jeon et al. (2021)) uncovers unobserved interactions between respondents and items in the item response data by embedding both in a shared latent metric space. The R package lsirm12pl implements Bayesian estimation of the LSIRM and its extensions for various response types, base model specifications, and missing data handling. Furthermore, the lsirm12pl package provides methods to improve model utilization and interpretation, such as clustering item positions on an estimated interaction map. The package also offers convenient summary and plotting options to evaluate and process the estimated results. In this paper, we provide an overview of the LSIRM's methodological foundation and describe several extensions included in the package. We then demonstrate the use of the package with real data examples contained within it.

1 Introduction

Item response theory (IRT) models are a widely-used statistical approach to analyze assessment data in various fields, e.g., medical, educational, psychological, health, and marketing research (An and Yung, 2014; Zanon et al., 2016). IRT models are designed to establish a relationship between observed item response data and unobserved person characteristics, commonly referred to as latent traits, e.g., competencies, attitudes, or personality (de Ayala, 2009; Brzezińska, 2018). IRT models can predict the probability of a correct (or positive) response as a function of the respondents' latent traits and item features, such as item difficulty and discrimination. Additional technical details of IRT models are provided in the subsequent section.

Several R packages are available for estimating IRT models. The **ltm** package (Rizopoulos, 2006) is available to analyze dichotomous and polytomous item response data, including a one-parameter logistic (1PL) model (or the Rasch model), a two-parameter logistic (2PL) model, a three-parameter model (Rasch, 1960; Birnbaum, 1968) and a graded response model (Samejima, 1968). The **eRm** package (Mair and Hatzinger, 2007) estimates various extensions of the Rasch model, such as the rating scale model (RSM) (Andrich, 1978), partial credit model (PCM) (Masters, 1982), linear logistic test model (LLTM) (Scheiblechner, 1972), the linear rating scale model (LRSM) (Fischer and Parzer, 1991), and the linear partial credit model (LPCM) (Glas and Verhelst, 1989; Fischer and Ponocny, 1994). The **mirt** package (Chalmers, 2012) can estimate a wide range of IRT models, including exploratory and confirmatory multidimensional item response models. The **pcIRT** package (Hohensinn, 2018) provides functions for estimating IRT models for polytomous (nominal) and continuous data, including the multidimensional polytomous Rasch model (Andersen, 1973) and the continuous rating scale model (Müller, 1987), using conditional maximum likelihood (CML) estimation (Baker and Kim, 2023).

Conventional IRT models are typically based on two assumptions: conditional independence and homogeneity. Conditional independence assumes that item responses are independent of each other conditional on respondents' latent traits. The homogeneity assumption is, e.g., that respondents with the same ability have the same probability of answering a question correctly and that respondents have the same probability of answering a question with the same item difficulty. However, these assumptions are often violated in practice due to unobserved interactions between respondents and items, e.g., when particular items are more similar to each other (e.g., testlets) or when particular respondents show different probabilities of giving correct responses to certain items compared to other respondents with similar ability (e.g., differential item functioning). Violations of these assumptions can lead to biased parameter estimates and inferences (Chen and Wang, 2007;

Braeken, 2010; Myszkowski and Storme, 2024). While some existing methods can address known violations prior to data analysis, there is currently no approach that enables the detection or management of unknown sources of violations in item response analysis, to the best of our knowledge.

Jeon et al. (2021) proposed a latent space item response model (LSIRM) that addresses such limitations of conventional IRT models. The LSIRM aims to estimate inherent interactions between respondents and items, alongside the latent traits of both. A key feature is its visual representation of these interactions in a low-dimensional latent space, called an *interaction map* in the form of distances between them. Interaction maps offer a clear and intuitive interpretation of complex respondent-item relationships, allowing users to identify patterns and clusters based on spatial proximity on the map. Additional details of the LSIRM are provided in a later section.

This paper presents the **lsirm12pl** package in R that offers Bayesian estimation of the LSIRM and its extensions. Jeon et al. (2021) focused on the response data for binary items and the Rasch model as the base model, and currently, no package is available to estimate the LSIRMs. To broaden the applicability of latent space item response modeling, **lsirm12pl** enables: (1) modeling continuous item responses; (2) missing data handling under different missing mechanisms assumptions (Rubin, 1976); and (3) an extended base model specification using the 2PL model both for binary and continuous item response data. The package also offers options to cluster latent positions of items in the estimated interaction map using spectral clustering and the Neyman-Scott process model. The **lsirm12pl** supplies convenient summary and plotting options for interaction maps, model assessment, diagnosis, and result process and interpretation, aiming to improve the utilization of the LSIRM in practice.

The subsequent sections of the paper are structured as follows. To begin, we provide a concise overview of the 1PL and 2PL IRT models. We then delve into the LSIRM for binary response data and demonstrate how to fit the model with the package functions using a real dataset. Next, we extend the LSIRM to accommodate continuous data and provide guidance on fitting this extended model using the **lsirm12pl** package. In the end, we conclude the paper with final remarks and a discussion of future developments.

2 Item response theory models

IRT models are essential for analyzing item response data, which comprises respondents' answers to items on tests, surveys, or questionnaires. This section briefly discusses two widely utilized IRT models for dichotomous item response data.

2.1 1PL IRT Model

The 1PL model, also known as the Rasch model (Rasch, 1961), is a classic IRT model for analyzing dichotomous item response data. Suppose $\boldsymbol{Y} = \{y_{k,i}\} \in \{0,1\}^{N\times P}$ is the $N \times P$ binary item response matrix under analysis, where $y_{k,i} = 1$ indicates a correct (or positive) response of the respondent k to the item i. In the Rasch model, the probability of the correct response for item i given by respondent k is given as follows:

$$\text{logit}(\mathbb{P}(y_{k,i} = 1|\theta_k, \beta_i)) = \theta_k + \beta_i, \;\; \theta_k \sim N(0, \sigma^2),$$

where $\theta_k \in \mathbb{R}$ is the respondent intercept parameter of respondent k, $\beta_i \in \mathbb{R}$ is the item intercept parameter of item i. The person intercept parameter represents the latent trait of interest, such as cognitive ability. The item intercept parameter represents the item easiness (or minus difficulty). As the respondent's ability increases, her/his likelihood of giving correct responses increases. On the other hand, the likelihood of correct responses decreases as the difficulty of the item increases. Note that based on the model, the probability of a respondent's giving a correct response to an item is a function of the two parameters – the person's ability and the item's difficulty. This means that respondents with the same level of ability are assumed to have the same probability of giving a correct response to an item.

Similarly, items with the same level of difficulty are assumed to have the same probability of being correctly responded to. In other words, interactions between persons and items are not allowed in this model. However, in reality, respondents with the same level of ability and items with the same level of difficulty may show a different success probability due to unobserved characteristics they may have, violating the assumptions.

2.2 2PL IRT Model

The 2PL model (Birnbaum, 1968) extends the 1PL model by incorporating a discrimination parameter for each item. This parameter reflects the ability of the item to differentiate among respondents with similar abilities. The item discrimination parameters are also considered as item slopes, indicating how quickly the probability of correct responses increases as the respondent's abilities increase (An and Yung, 2014). A larger discrimination parameter indicates that the item is better at distinguishing between respondents with similar ability levels. In the 2PL model, the probability of person k giving a correct response to item i is given as follows:

$$\text{logit}(\mathbb{P}(y_{k,i} = 1|\theta_k, \alpha_i, \beta_i)) = \alpha_i\theta_k + \beta_i, \quad \theta_k \sim N(0, \sigma^2), \tag{1}$$

where $\alpha_i \in \mathbb{R}$ is the discrimination parameter for the item i. For model identifiability, one of the item slope parameters is fixed at 1, e.g., $\alpha_1 = 1$. With the term $\alpha_i\theta_k$, the 2PL model allows for an interaction between respondents and items to some degree. However, it is not likely to capture all interactions between respondents and items that might not depend on respondents' ability (Jeon et al., 2021).

3 Standard lsirm: 1pl lsirm for dichotomous data

The LSIRM (Jeon et al., 2021) has been proposed as an extension of the conventional IRT models. The key idea of the LSIRM is to place respondents and items in a low-dimensional, shared metric space, called an interaction map, so that their unobserved interactions can be captured in the form of distances between them. Below we provide a brief overview of the LSIRM in its original form, presented for dichotomized data.

3.1 Statistical framework

To capture unobserved interactions between respondents and items, the original LSIRM (Jeon et al., 2021) embeds items and respondents in an interaction map, i.e., a D-dimensional latent Euclidean space. Note that the interaction map is used as a tool to represent pairwise interactions between respondents and items; thus, the dimensions of an interaction map do not represent any specific quantity or have a substantive meaning. The original LSIRM is built on the 1PL IRT model; thus, we refer to the original LSIRM as the 1PL LSIRM.

The 1PL LSIRM assumes that the probability of giving a correct answer to item i by the respondent k is determined by a linear combination of the main effect of the respondent k, the main effect of the item i, and the pairwise distance between the latent position of item i and the latent position of respondent k. Then, the model is given by:

$$\text{logit}(\mathbb{P}(y_{k,i} = 1|\theta_k, \beta_i, \gamma, \boldsymbol{z_k}, \boldsymbol{w_i})) = \theta_k + \beta_i - \gamma d(\boldsymbol{z_k}, \boldsymbol{w_i}), \tag{2}$$

where $\theta_k \in \mathbb{R}$ and $\beta_i \in \mathbb{R}$ represent the respondent's latent trait and the item's easiness, respectively, same as in the conventional 1PL model. These two terms can also be seen as the main effects of respondents and items. The third term, $-\gamma d(\boldsymbol{z_k}, \boldsymbol{w_i})$, captures the interactions between respondents and items, where $\boldsymbol{z_k} \in \mathbb{R}^D$ and $\boldsymbol{w_i} \in \mathbb{R}^D$ are the latent position of the respondent k, and the latent position of the item i, respectively, where $\gamma \geq 0$ is the weight of the distance term $d(z_k, w_i)$. For a distance function $d : \mathbb{R}^D \times \mathbb{R}^D \mapsto [0, \infty)$, we use a Euclidean norm (that is, $d(z_k, w_i) = ||z_k - w_i||$) as a distance function in `lsirm1pl`

to maintain simplicity and enhance the interpretability of the model (Hoff et al., 2002). The weight of the distance term $\gamma \geq 0$ indicates the amount of interactions between respondents and items in the data, such that large γ implies stronger evidence for respondent by item interactions in the data. In contrast, near zero γ implies little evidence of respondent-item interactions in the data, thus suggesting that one may go with the conventional IRT model without the interaction term.

The likelihood function of the observed data with the 1PL LSIRM is

$$\mathbb{L}\left(\boldsymbol{Y}=\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{\beta},\gamma,\boldsymbol{Z},\boldsymbol{W}\right)=\prod_{K=1}^{N}\prod_{i=1}^{P}\mathbb{P}(Y_{k,i}=y_{k,i}|\theta_k,\beta_i,\gamma,\boldsymbol{z_k},\boldsymbol{w_i}),$$

where $\boldsymbol{\theta}=(\theta_1,\ldots,\theta_N)$, $\boldsymbol{\beta}=(\beta_1,\ldots,\beta_P)$, $\boldsymbol{Z}=(\boldsymbol{z_1},\ldots,\boldsymbol{z_N})$ and $\boldsymbol{W}=(\boldsymbol{w_1},\ldots,\boldsymbol{w_P})$. Here, item responses are assumed to be independent conditional on the positions of respondents and items in an interaction map, as well as the main effects of respondent and item. This means that the traditional conditional independence assumption is alleviated with the model by accounting for respondent-item interactions. The detailed parameter estimations are described in the following Section.

3.2 Parameter Estimation

The R package **lsirm12pl** applies a fully Bayesian approach using the Markov chain Monte Carlo (MCMC) for estimation of the LSIRM. Following is the posterior distribution of the 1PL LSIRM.

$$\begin{aligned}\pi\left(\boldsymbol{\theta},\boldsymbol{\beta},\gamma,\boldsymbol{Z},\boldsymbol{W}|\boldsymbol{Y}=\boldsymbol{y}\right)\propto&\prod_{k=1}^{N}\prod_{i=1}^{P}\mathbb{P}(Y_{k,i}=y_{k,i}|\theta_k,\beta_i,\gamma,\boldsymbol{z_k},\boldsymbol{w_i})\\&\times\prod_{k=1}^{N}\pi(\theta_k)\times\prod_{i=1}^{P}\pi(\beta_i)\times\pi(\gamma)\times\prod_{k=1}^{N}\pi(\boldsymbol{z_k})\prod_{i=1}^{P}\pi(\boldsymbol{w_i})\end{aligned}$$

The priors for the model parameters are specified as follows:

$$\begin{aligned}\theta_k\mid\sigma^2&\sim N(0,\sigma^2),\ \sigma^2>0\\\beta_i\mid\tau_\beta^2&\sim N(0,\tau_\beta^2),\ \tau_\beta^2>0\\\log\gamma\mid\mu_\gamma,\tau_\gamma^2&\sim N(\mu_\gamma,\tau_\gamma^2),\ \mu_\gamma\in\mathbb{R},\ \tau_\gamma^2>0\\\sigma^2\mid a_\sigma,b_\sigma&\sim\text{Inv-Gamma}(a_\sigma,b_\sigma),a_\sigma>0,b_\sigma>0\\\boldsymbol{z_k}&\sim\text{MVN}_D(\boldsymbol{0},\boldsymbol{I_D})\\\boldsymbol{w_i}&\sim\text{MVN}_D(\boldsymbol{0},\boldsymbol{I_D}).\end{aligned}$$

where $\boldsymbol{0}$ is a D-vector of zeros and $\boldsymbol{I_D}$ is $D\times D$ identity matrix. The argument names and default values for the prior specifications in the **lsirm12pl** are described in our Github site [1].

To generate posterior samples for $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, $\boldsymbol{Z}$ and $\boldsymbol{W}$, we use the Metropolis-Hastings-within-Gibbs sampler (Chib and Greenberg, 1995). The conditional posterior distribution for each parameter is given in Equations in the Github site. We list the arguments, the default values for the jumping rules, and the standard deviations of the Gaussian proposal distributions in the Github site.

The log-odds of the probability of giving correct responses depend on the latent positions through distances, as discussed in Jeon et al. (2021). Because distances are invariant to translations, reflections, and rotations of the positions of respondents and items, the likelihood function is invariant under these transformations. To resolve the identifiability issue of latent positions, the **lsirm12pl** package applies Procrustes transformation (Gower, 1975) as a post-processing of posterior samples, which is a standard procedure in the latent space modeling literature (Hoff et al., 2002; Sewell and Chen, 2015; Jeon et al., 2021).

[1] https://github.com/jiniuslab/lsirm12pl

3.3 An Illustrated Example

Here, we demonstrate how to apply the 1PL LSIRM to real datasets using the **lsirm12pl** package. To this end, we use the Inductive Reasoning Developmental Test (TDRI) dataset (Golino, 2016) from the package, which contains item responses from 1,803 Brazilians (52.5% female) of ages ranging from 5 to 85 years (M = 15.75; SD = 12.21). TDRI is a pencil-and-paper test consisting of 56 items that are designed to assess developmentally sequenced and hierarchically organized inductive reasoning.

The **lsirm12pl** package provides several functions for fitting the LSIRM, which requires setting hyperparameter values for prior distributions and/or tuning parameters for MCMC chains. By default, the `lsirm` function runs with the default settings unless otherwise specified by the user. The default MCMC run setting includes 15,000 iterations, 2,500 burn-ins, and 5 thinning. The base function for running the LSIRM is

```
lsirm(A ~ <term 1>(<term 2>,<term 3>,...))
```

where `A` is an item response matrix to be analyzed, `<term1>` is for the model option – either 'lsirm1pl' or 'lsirm2pl', while `<term 2>` and `<term 3>` are other specific options for the chosen model, which are detailed in the documentation of the **lsirm12pl** package. The following is an example of how to fit the 1PL LSIRM to the TDRI dataset (with no missing) with a default estimation setting with 4 MCMC chains using 2 multi-core processors. Additional details of other default values can be found in the documentation of the **lsirm12pl** package.

```
R > library("lsirm12pl")
R > data <- lsirm12pl::TDRI
R > data <- data[complete.cases(data),]
R > head(data)
  i1 i2 i3 i4 ... i54 i55 i56
1  1  1  1  1  ... 0   0   0
2  1  1  1  1  ... 0   0   0
3  1  1  1  1  ... 0   0   0
4  1  1  1  1  ... 0   0   0
5  1  1  1  1  ... 0   0   0

R > lsirm_result <- lsirm(data ~ lsirm1pl(chains = 4, multicore = 2, seed = 2025))
```

The estimation results of the model parameters $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, γ, $\boldsymbol{Z}$, and $\boldsymbol{W}$ are stored in individual lists per chain, while all results across the chains are stored in a single list.

The `summary()` function generates a summary of the first chain by default, but users can obtain summaries of other chains by setting the `chain.idx` option. The function provides posterior estimates of the "covariate coefficients" (model parameters such as β), allowing the users to select the `"mean"`, `"median"`, or `"mode"` through the `estimate` option. It also provides the highest posterior density interval (HPD) for the model parameters with the `CI` option that allows the users to set different significance levels. Additionally, the function supplies the Bayesian information criterion (BIC) and the maximum log posterior value. When the column names are available in the input data, the `summary()` function uses these names in the summary of the results.

```
R > summary(lsirm_result, chain.idx = 1, estimate = 'mean', CI = 0.95)
==========================
Summary of model
==========================

Call:         lsirm.formula(formula = data ~ lsirm1pl(chains = 4, multicore = 2, seed = 2025))
Model:         lpl LSIRM
Data type:         binary
```

```
Variable Selection:         FALSE
Missing:          NA
MCMC sample of size 15000, after burnin of 2500 iteration

Covariate coefficients posterior means of chain 1 :

    Estimate     2.5%   97.5%
i1   6.42354  5.82468  7.0642
...
i56  -1.01604 -2.58822  0.8133

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

Overall BIC (Smaller is better) : 43661.52

Maximum Log-posterior Iteration:
      value iter
[1,] -11487 137
```

The `diagnostic()` function checks the convergence of MCMC for each parameter using various diagnostic tools, such as trace plots, posterior density distributions, autocorrelation functions (ACF), and Gelman-Rubin-Brooks plots. The `diagnostic()` function has options: `draw.item`, and `gelman.diag`. The `draw.item` option in the `diagnostic()` function specifies the names and indexes of the parameters to diagnose. The `draw.item` option is set to a list where a key represents each parameter such as ``beta'',``theta'',``gamma'',``alpha'',``sigma'', and ``sigma_sd'', and the values indicate the indices of these parameters. The indexes can be expressed as vectors. In the following example code, the `draw.item` option is set as `list(``beta'' = c(1))` to check the diagnostic of the first index of the beta parameter, i.e. $\beta_1$. With `gelman.diag = TRUE`, the Gelman-Rubin convergence diagnostic, known as potential scale reduction factors (PSRF), is obtained.

```
R > diagnostic(lsirm_result,
           draw.item = list("beta" = c(1)),
           gelman.diag = TRUE)

Potential scale reduction factors:

          Point est. Upper C.I.
beta [i1]       1.01       1.04
```

Figure 1 displays the output of `diagnostic()` for β_1: trace plot (top left), posterior density plot (top right), autocorrelation plot (bottom left), and Gelman-Rubin-Brooks plot (bottom right). Different colors indicate different MCMC chains. Trace plots visualize the mixing of the MCMC chains. In a well-converged model, the trace plots should show that chains fluctuate consistently around a constant value, which indicates the parameter space is thoroughly explored. Density plots display the distribution of the sampled parameter values. In a converged model, the density plots should exhibit smooth, overlapping curves across different chains, which demonstrates that the samples are drawn from the same posterior distribution. Autocorrelation plots measure the correlation between samples at different lags. For a converged model, the autocorrelation should decrease rapidly as the lag increases, which suggests the samples are independent and the parameter space has been effectively explored. Gelman-Rubin-Brooks plots show a shrink factor, known as the potential scale reduction, which compares the variance within each chain to the variance between chains. A shrink factor value close to 1 indicates the within-chain variance is similar to the between-chain variance, providing evidence of good convergence. By examining

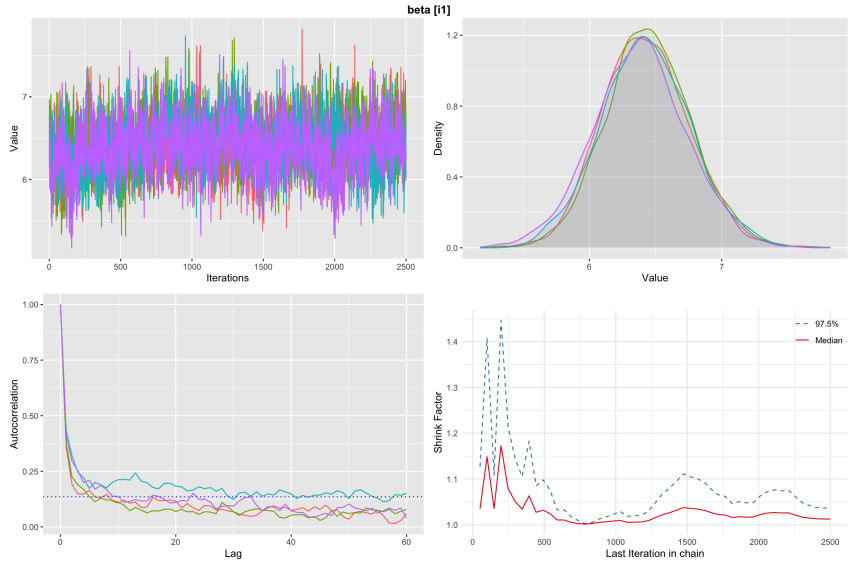


Figure 1: Diagnostics of β_1 using the `diagnostic()` function on the results of the 1PL LSIRM fitted to the TDRI dataset. The top left is the trace plot, the top right is the posterior density plot, the bottom left is the autocorrelation plot, and the bottom right is the Gelman-Rubin-Brooks plot. Different colors represent different MCMC chains.

these diagnostics collectively, the convergence of the model parameters from the LSIRM can be ensured.

The **lsirm12pl** package supplies the `gof()` function that assesses the goodness of fit of the LSIRM to the item response data being analyzed. A main diagnostic tool is a boxplot that displays the average posterior 'predicted' response value for each item (item-wise means) with the average 'observed' response value for each item indicated by red dots. When the model fits well, the red dot is close to the midline of the boxplot. The `gof()` function includes a `chain.idx` option to choose a specific chain for assessing the goodness of fit. By default, the first MCMC chain is selected (i.e. `chain.idx=1`). When the `diagnostic()` function shows good convergence for all parameters, it does not matter which chain is chosen because the parameter estimates must be consistent across all chains. For illustration, we use the first MCMC chain by setting `chain.idx=1`.

For binary data, the `gof()` function additionally offers a receiver operating characteristic (ROC) curve. The ROC curve is a graphical representation that assesses the performance of a binary classifier. The area under the curve (AUC) quantifies the overall ability of the model to distinguish between classes, with values ranging from 0.5 to 1. A higher AUC and a curve close to the top-left corner indicate better performance.

```
R > gof(lsirm_result, chain.idx = 1)
```

Figure 2 presents the output of the `gof()` function for the TDRI dataset, showing the results for the first MCMC chain, including both the boxplot (left) and the ROC curve (right). In this example, all of the red dots are closely located at the midline of the boxplot and the area under the curve (AUC) of the ROC curve approaches 1, suggesting that the fit of the LSIRM to the data is satisfactory.

The main outcome of the LSIRM fitting is the estimates of the respondent and item main effects and the interaction map. For a visual summary of these outputs, the user can use the `plot()` function with the `option` argument.

```
R > plot(lsirm_result, option = "beta", chain.idx = 1)
R > plot(lsirm_result, option = "theta", chain.idx = 1)
```

Figures 3 display the `plot()` results for summarizing β_i and θ_k by setting the `option` argument as the `"beta"` and `"theta"`, respectively. `option="beta"` generates the boxplots

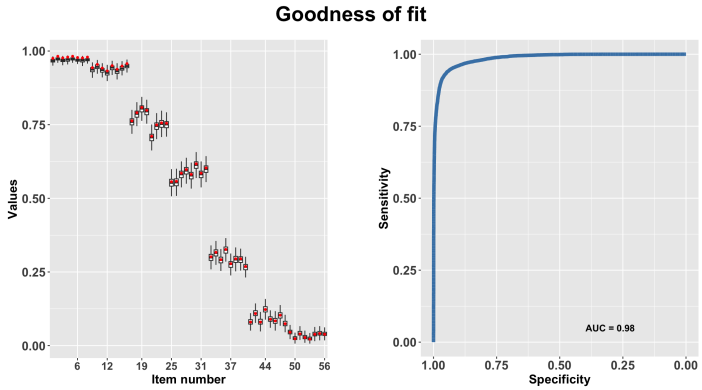
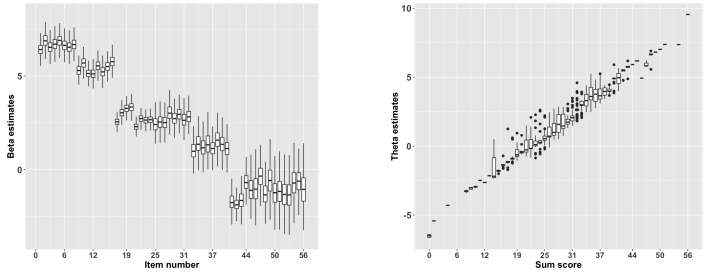


Figure 2: Goodness of fit of the 1PL LSIRM for the TDRI data using the `gof()` function. The box plots of the average posterior predicted response values for items (item-wise means) with the average observed response value for each item indicated by red dots (left) and the receiver operating characteristic (ROC) curve (right).



(a) β_i **(b)** θ_k

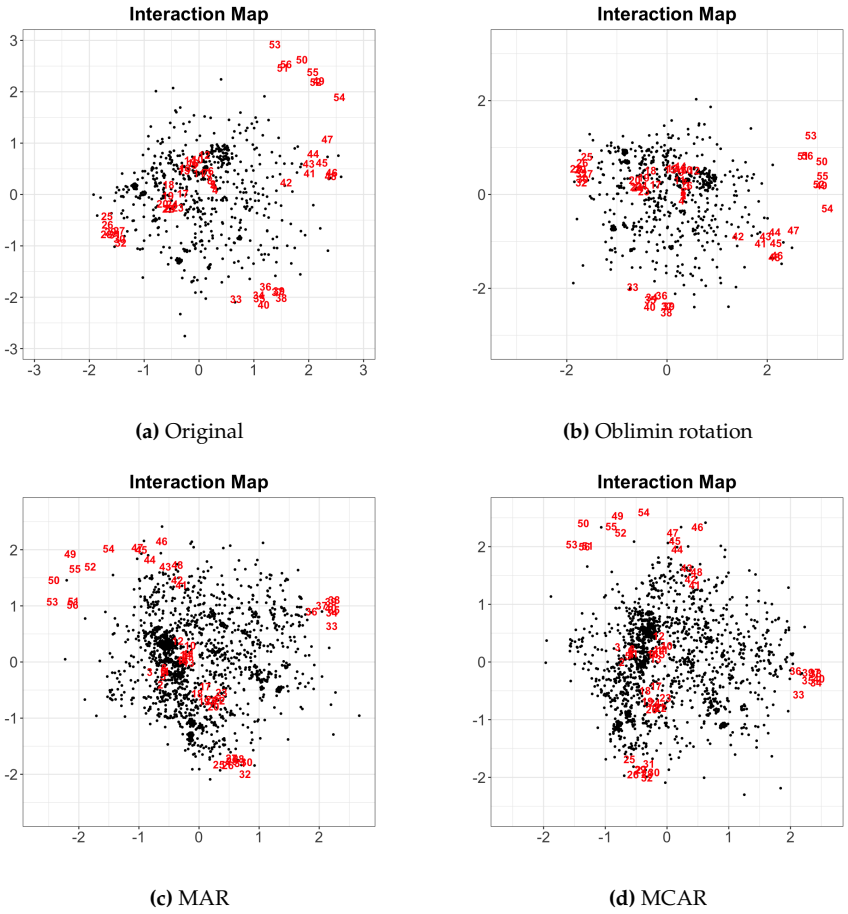
Figure 3: Visual summaries of the parameter estimate of β_i and θ_k using the `plot()` function based on the 1PL LSIRM fitted to the TDRI dataset. (a) Boxplots of the posterior samples for β_i; outliers are suppressed in the boxplots for the sake of simplicity. (b) Boxplots of the point estimate for θ_k as a function of the total sum scores of positive responses.

of the posterior samples for β_i, while `option="theta"` generates the boxplots of the point estimates for θ_k, plotted against the total sum scores of positive responses (ranging from 0 to P, where P is the number of items). In Figure 3a, as the item number increases (from left to right on the x axis), the β_k estimates decrease, indicating that earlier items are easier. In Figure 3b, individuals who correctly answer more items (i.e., higher sum scores) have higher θ_i estimates. It is sensible that those who answer more items correctly have higher θ_i values, as θ_i represents their ability levels.

The `plot()` also returns the interaction map with `option="interaction"`. The interaction map is created based on the estimated latent position of the item and respondent, $\boldsymbol{w}_i$ and $\boldsymbol{z}_k$, respectively. Note that the primary and unique advantage of the LSIRM lies in deriving intuitive information from the interaction map, based on the distances between respondents and items, between respondents, and between items. In the interaction map, a shorter distance between the latent position of the item i ($\boldsymbol{w_i}$) and the latent position of the respondent k ($\boldsymbol{z_k}$) indicates a stronger dependence (or interactions), which implies that the respondent k is more likely to respond correctly (or positively) to the item i, given the person's ability. In the **lsirm12pl** package, the interaction map is set to a two-dimensional space, as in Jeon et al. (2021), for parsimony and interpretability.

```
R > plot(lsirm_result, option = "interaction", chain.idx = 1)
```

Figure 4a (top left) displays the interaction map based on the 1PL LSIRM for the TDRI data. The latent positions of items are represented as red numbers and respondents as black dots on the map. Note that the two coordinates (dimensions) of the map do not



(a) Original **(b)** Oblimin rotation

(c) MAR **(d)** MCAR

Figure 4: The estimated interaction maps based on the 1PL LSIRM fitted to the TDRI dataset. (a) The original interaction map. (b) The rotated interaction map using oblimin rotation. (c) The interaction map based on the MAR assumption. (d) The interaction map based on the MCAR assumption which has been reflected across the y-axis to facilitate comparisons with the other interaction maps. Red numbers represent item positions and black dots represent respondent positions in all plots.

represent specific quantities or substantive meaning, as the interaction map is simply a tool to represent respondent-by-item interactions (Jeon et al., 2021). In Figure 4a, we observe that respondents are widely spread around the center of the interaction map, while the items are separated by some clusters. For example, items located in the far north, such as items 50, 52, 53, 54, and 55 (red numbers), are far from most respondents (black dots) on the map. This suggests that most respondents are less likely to respond correctly to those specific items, regardless of their ability levels. In other words, those items located in the south are difficult items, which is also confirmed in Figure 3(a).

```
R > plot(lsirm_result, option = "interaction", rotation = TRUE, chain.idx = 1)
```

If desired, one can rotate the interaction map to improve the interpretability of the coordinates of the map with `rotation=TRUE`.

The **lsirm12pl** package offers the oblimin rotation (Jennrich, 2002) using the **GPArotation** package in R (Bernaards and Jennrich, 2005). Figure 4b (top left) shows the rotated version of the original map shown in Figure 4a (top right). In this example, the original and rotated maps appear pretty similar, although on the right there seems to be slight clockwise rotation compared to the one on the left; after rotation, the x- and y- coordinates may be interpreted based on the items that are closely placed to the respective coordinates.

Further, the **lsirm12pl** package offers an option to cluster latent positions of items. Two types of clustering methods are available: spectral clustering (Ng et al., 2001; von

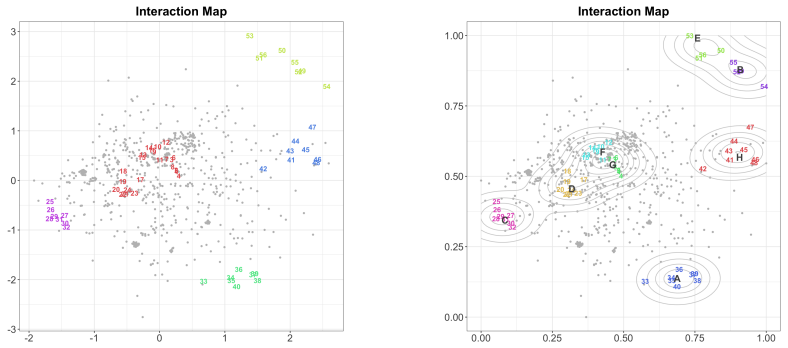
Luxburg, 2007) and the Neyman-Scott process modeling approach (Thomas, 1949; Neyman and Scott, 1952; Yi et al., 2024), with `cluster` option as `spectral` and `neyman`, respectively. Spectral clustering is a technique that clusters points based on the eigenvalues of a similarity matrix, using the spectral properties of the data to identify clusters. The implementation of spectral clustering is based on the **kernlab** package (Karatzoglou et al., 2004) in R where the number of clusters is determined using the average silhouette width (Batool and Hennig, 2021). The Neyman-Scott point process modeling approach is a method to cluster points in time or space. Parent points (cluster center) are generated using a Poisson process, with offspring points clustered around each parent based on a defined probability distribution. The **lsirm12pl** package implements this method directly; it applies the MCMC algorithm a number of times independently to determine the distribution of the cluster number and cluster centers. Then, the mode of the cluster number distribution is chosen as the optimal cluster number. With the optimal cluster number, the cluster center that minimizes the Bayesian information criterion is selected, and items are assigned to the nearest cluster based on Euclidean distances.

```
R > plot(lsirm_result, cluster = "spectral", chain.idx = 1)

Clustering result (Spectral Clustering):
  group                                      item
      A              49, 50, 51, 52, 53, 54, 55, 56
      B              33, 34, 35, 36, 37, 38, 39, 40
      C              41, 42, 43, 44, 45, 46, 47, 48
      D              1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                     11, 12, 13, 14, 15, 16, 17, 18,
                     19, 20, 21, 22, 23, 24
      E              25, 26, 27, 28, 29, 30, 31, 32

R > plot(lsirm_result, cluster = "neyman", chain.idx = 1)

|================================================| 100%
Clustering result (Neyman-Scott process):
  group                                      item
      A             33, 34, 35, 36, 37, 38, 39, 40
      B             49, 52, 54, 55
      C             25, 26, 27, 28, 29, 30, 31, 32
      D             17, 18, 19, 20, 21, 22, 23, 24
      E             50, 51, 53, 56
      F             9, 10, 11, 12, 13, 14, 15, 16
      G             1, 2, 3, 4, 5, 6, 7, 8
      H             41, 42, 43, 44, 45, 46, 47, 48
```

Figures 5a and 5b are item clustering results based on spectral clustering and the Neyman-Scott process model, respectively. In both plots, the gray dots indicate respondents, where numbers in colors indicate items with different cluster memberships. The Neyman-Scott process modeling approach additionally displays the center of the item cluster (alphabets A to H) and a contour for each item cluster. In this example, the spectral clustering method identifies five clusters, while the Neyman-Scott process modeling approach identifies eight clusters. Overall, the Neyman-Scott process modeling approach further split the items near the center (red items on the left) and in the north (light green items on the left) compared to the spectral clustering method.

3.4 Flexible Modeling Options

The **lsirm12pl** package provides a range of flexible modeling options for the LSIRM.



(a) Spectral Clustering **(b)** Neyman-Scott Process Model

Figure 5: The interaction map with item clustering results based on the 1PL LSIRM fitted to the TDRI dataset, using (a) spectral clustering and the (b) Neyman-Scott process modeling approach. In both plots, the gray dots indicate respondents, while numbers in colors indicate items with different cluster memberships. The Neyman-Scott process approach additionally displays the center of the cluster (alphabets A to H) and a contour for each cluster.

First, with `fixed_gamma` option, one can fix the distance weight γ to a constant value of 1. By doing so, the scale of the latent space can be standardized, easing comparison between different interaction maps. The default value of `fixed_gamma` option is `FALSE` (i.e., γ is freely estimated).

```
R > lsirm_result <- lsirm(data ~ lsirm1pl(fixed_gamma = TRUE))
```

Second, with `spikenslab` option, one can apply a model selection with the spike-and-slab prior for the distance weight γ (Ishwaran and Rao, 2005). The spike-and-slab prior is a mixture of two log-normal priors: one is densely concentrated near zero, while another is more broadly spread across positive values. With this option, one can determine whether γ is zero or not, which in turn determines whether the distance term is needed for the data being analyzed. If γ is not zero, it is evidence for item-by-respondent interactions in the data being analyzed, and thus it would be useful to investigate the interactions between respondents and items with the LSIRM approach. The default value of `spikenslab` option is `FALSE`.

The posterior probability of γ being non-zero is indicated by the return value of `pi_estimate`. A `pi_estimate` value greater than 0.5 suggests that γ is likely non zero. Note that the two options `fixed_gamma` and `spikenslab` cannot be used simultaneously.

```
R > lsirm_result <- lsirm(data ~ lsirm1pl(spikenslab = TRUE))
R > lsirm_result$pi_estimate
[1] 0.9984
```

Third, the package offers options for handling missing data. In the context of missing data, three key assumptions are considered (Rubin, 1976): missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). MCAR occurs when the probability of missing data on a variable is unrelated to any other measured variables or the variable itself, making the missingness entirely random. MAR happens when the probability of missing data on a variable is related to some of the observed data but not the missing data itself, meaning the missingness can be explained by other observed variables. Unlike MCAR and MAR, MNAR assumes that the missing data mechanism depends on the unobserved data itself, making it challenging to estimate the model without strong assumptions or additional information about the missing data. Therefore, we focus on two types of missing data, MCAR and MAR, in the **lsirm12pl** package with the `missing_data` option.

With `missing_data = "mcar"`, the missing data are assumed to follow MCAR and the parameters are estimated solely based on the observed elements of the dataset being

analyzed. On the other hand, with `missing_data="mar"`, the missing data are assumed to be MAR, and the data augmentation algorithm (Tanner and Wong, 1987) is applied to impute missing values. With `missing_data="mar"`, the function returns the posterior samples of the imputed responses with `imp` and the probability of a correct response with `imp_estimate`. Imputed values are listed in the order of respondents. Note that all missing values should be recoded via `missing.val`. The percentage of missing data in the TDRI dataset is 30%, and missing values are replaced with 99 which is the default coding of missing data. The `missing_data` option can be used in combination with other options such as `(spikenslab = TRUE,missing_data = "mar")`.

```
R > data <- lsirm12pl::TDRI
R > data[is.na(data)] <- 99
R > lsirm_result <- lsirm(data ~ lsirm1pl(missing_data = "mcar"))
R > lsirm_result <- lsirm(data ~ lsirm1pl(missing_data = "mar"))
R > lsirm_result$imp_estimate
[1] 0.9900 0.9868 0.9768 0.9884 0.9716 0.9716
...
[997] 0.9860 0.9788 0.8240 0.9924
 [ reached getOption("max.print") -- omitted 32327 entries ]
R > plot(lsirm_result, option = "interaction")
```

Figures 4(c) and 4(d) show the resulting interaction maps with MAR and MCAR assumptions, respectively. Note that a reflection is applied to the interaction map with MCAR assumptions (c) across the y-axis to ease comparisons with the other interaction plots. Reflection or rotation does not change the interpretations of the interaction map as interpretations are based on the distances, not the positions themselves. If the interaction maps of two missing assumptions are considerably different, further investigation is necessary to determine which assumption would be more suitable for the analyzed data. In this example, the interaction maps with the missing data options are similar to the original map presented in Figure 4a. Note that the original interaction map (a) is based on the complete item response data with no missing values; that is, the data includes only respondents who answered all items. In contrast, the interaction maps with the missing data option are based on MAR and MCAR assumptions. The similarity between these maps suggests that the observed only or imputed item response data provides a reasonable representation of the original item response data.

4 2pl lsirm for dichotomous data

The two-parameter LSIRM (2PL LSIRM) extends the 1PL LSIRM with item discrimination parameters.

4.1 Statistical framework

The 2PL LSIRMs the probability of a correct response by respondent k to item i as follows:

$$\text{logit}\Big(\mathbb{P}(Y_{k,i}=1|\theta_k,\alpha_i,\beta_i,\gamma,\boldsymbol{z_k},\boldsymbol{w_i})\Big)=\alpha_i\theta_k+\beta_i-\gamma d(\boldsymbol{z_k},\boldsymbol{w_i}),\quad \theta_k\sim N(0,\sigma^2),$$

where θ_k, β_i, $\mathbf{z}_k$, $\mathbf{w}_i$ and γ have similar interpretations to Equation (2), while α_i represents the item discrimination parameters and one of the α_i parameters is fixed at 1, e.g., $\alpha_1=1$ to ensure identifiability.

The observed data likelihood function under the 2PL LSIRM is given as

$$\mathbb{L}\left(\boldsymbol{Y}=\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{\alpha},\boldsymbol{\beta},\gamma,\boldsymbol{Z},\boldsymbol{W}\right)=\prod_{k=1}^{N}\prod_{i=1}^{P}\mathbb{P}(Y_{k,i}=y_{k,i}|\theta_k,\alpha_i,\beta_i,\gamma,\boldsymbol{z_k},\boldsymbol{w_i}).$$

4.2 Parameter Estimation

Following is the posterior distribution of the 2PL LSIRM.

$$\pi(\boldsymbol{\theta}, \boldsymbol{\beta}, \gamma, \boldsymbol{Z}, \boldsymbol{W} | \boldsymbol{Y} = \boldsymbol{y}) \propto \prod_{k=1}^{N} \prod_{i=1}^{P} \mathbb{P}(Y_{k,i} = y_{k,i} | \theta_k, \alpha_i, \beta_i, \gamma, \boldsymbol{z_k}, \boldsymbol{w_i}) \times \prod_{k=1}^{N} \pi(\theta_k) \times \prod_{i=1}^{P} \pi(\alpha_i) \times \prod_{i=1}^{P} \pi(\beta_i) \times \pi(\gamma) \times \prod_{k=1}^{N} \pi(\boldsymbol{z_k}) \prod_{i=1}^{P} \pi(\boldsymbol{w_i})$$

The prior distributions for θ_k, β_i, $\mathbf{z}_k$, $\mathbf{w}_i$, and γ are identical for the 1PL LSIRM. For item discrimination parameters α, a log-normal distribution is used with a mean of μ_α and a variance of τ_α^2, as α is typically assumed to be positive. The argument names and default values for the prior specification are shown in the Github site. For the item discrimination parameters α, the arguments and default values are `pr_mean_alpha = 0.5,pr_sd_alpha = 1`.

The conditional posterior distribution for each parameter follows the same form as the Equation in the Github site. The jumping rule for each parameter is given in the Github site. The default jumping rule for α is `jump_alpha = 1`.

4.3 An Illustrated Example

We apply the 2PL LSIRM to the TDRI data. The default settings of the 2PL LSIRM are the same as the 1PL LSIRM except for α. The base function for the 2PL LSIRM is

```
lsirm(A ~ lsirm2pl())
```

The 2PL LSRIRM for dichotomous data was fitted with 4 MCMC chains using 2 multicore processors, as demonstrated in the following example code. Similarly to the `lsirm1pl` results, the estimation results for the model parameters $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, $\boldsymbol{\alpha}$, γ, $\boldsymbol{Z}$, and $\boldsymbol{W}$ for each chain are provided in individual lists.

```
R > library("lsirm12pl")
R > data <- lsirm12pl::TDRI
R > data <- data[complete.cases(data),]
R > lsirm_result <- lsirm(data ~ lsirm2pl(chains = 4, multicore = 2, seed = 2025))
```

To diagnose the results of the 2PL LSIRM analysis, we can use the `diagnostic()` function. Similarly to the diagnostic of the 1PL LSIRM, the convergence of MCMC for each parameter can be checked using various diagnostic tools, such as trace plots, posterior density distributions, autocorrelation functions (ACF), and Gelman-Rubin-Brooks plots. By setting the `draw.item` option to `list('beta'= c(1))`, we perform diagnostics for the β_i parameter of the first item ($i = 1$).

```
R > diagnostic(lsirm_result,
               draw.item = list(beta = c(1)),
               gelman.diag = T)

Potential scale reduction factors:

          Point est. Upper C.I.
beta [i1]          1          1
```

Figure 6 displays the diagnostic results for β_1 of the results of the 2PL LSIRM fitted to the TDRI dataset. These plots help us assess the convergence of MCMC for β_1. The interpretation of each plot from the `diagnostic()` function is the same as mentioned for the

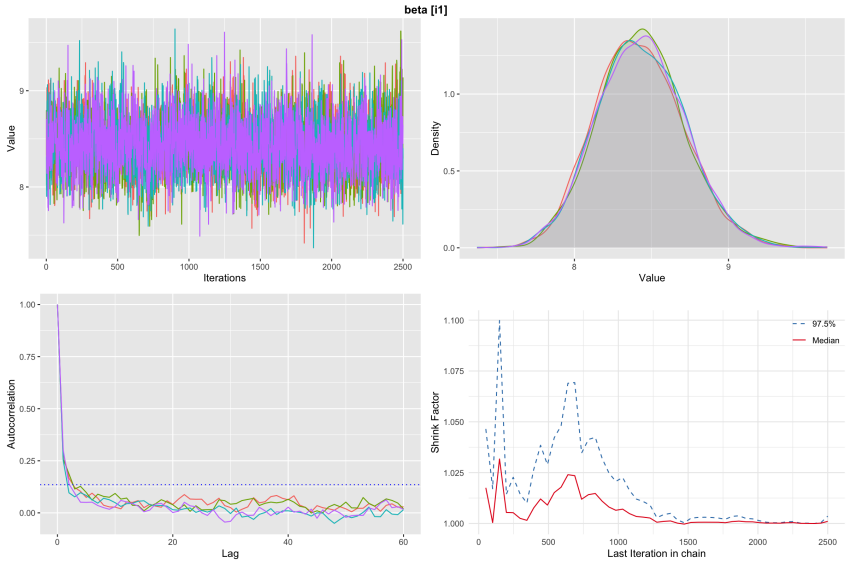


Figure 6: Diagnostics of β_1 using the `diagnostic()` function on the results of the 2PL LSIRM fitted to the TDRI dataset. The top left is the trace plot, the top right is the posterior density plot, the bottom left is the autocorrelation plot, and the bottom right is the Gelman-Rubin-Brooks plot. Different colors represent different MCMC chains.

1PL LSIRM in the previous Section. In Figure 6, the convergence of β_1 was confirmed by various diagnostic tools.

The `gof()` function assesses the goodness-of-fit of the LSIRM. Figure 7 visualizes the box plots of the posterior predicted response values (item-wise means) compared with the observed item-wise means and the ROC curve to check the performance of the 2PL LSIRM fitted to the TDRI dataset. In the figure, most of the red dots are located close to the midline of the boxplots and the AUC is 0.97, indicating the fit of the 2PL LSIRM to the TDRI dataset is satisfactory.

```
R > gof(lsirm_result, chain.idx = 1)
```

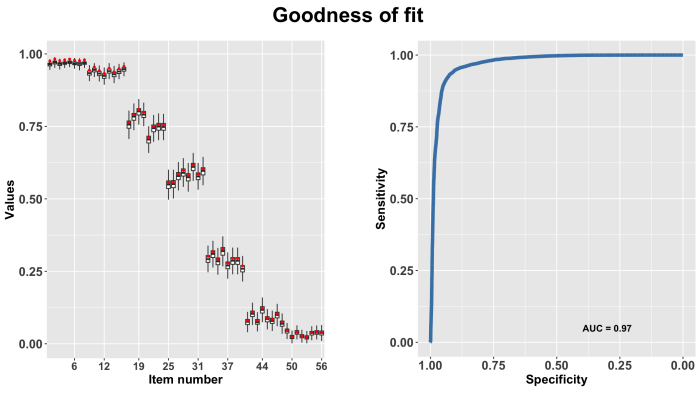
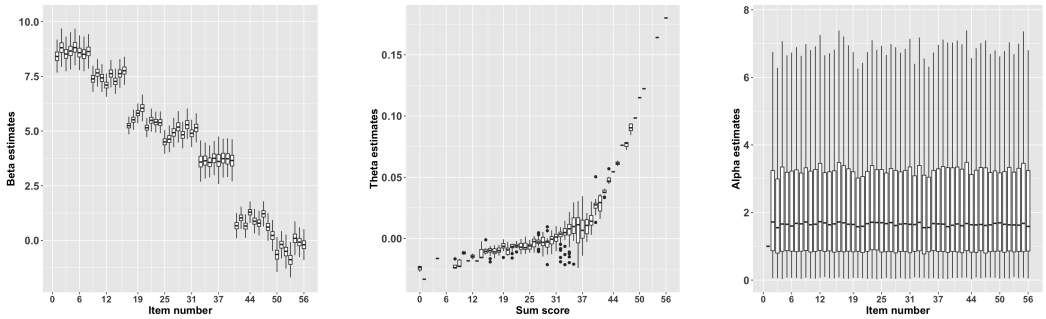


Figure 7: Goodness of fit of the 2PL LSIRM for the TDRI data using the `gof()` function. The box plots of the average posterior predicted response values for items against the average observed response value for each item indicated by red dots (left) and the receiver operating characteristic (ROC) curve (right).

Similarly to the 1PL LSIRM, the `plot()` function can generate different graphs for the 2PL LSIRM results with the `option` argument:

```
R > plot(lsirm_result, option = "beta", chain.idx = 1)
R > plot(lsirm_result, option = "theta", chain.idx = 1)
R > plot(lsirm_result, option = "alpha", chain.idx = 1)
```



(a) β_i **(b)** θ_k **(c)** α_i

Figure 8: Summarizing β_i and θ_k using the `plot()` function on the results of the 2PL LSIRM fitted to the TDRI dataset. (a) Boxplots of the posterior samples for β_i. (b) Boxplots of the point estimate for θ_k as a function of the total sum scores of positive responses. (c) Box plots of the posterior samples for α_i.

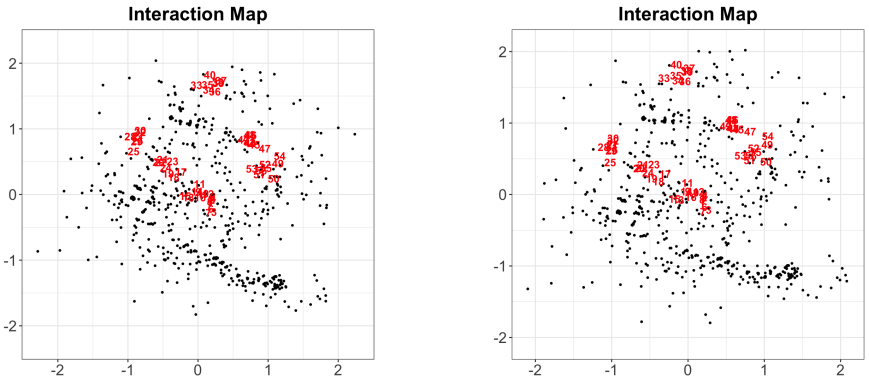
Figure 8 illustrates the results of the `plot()` function with the `"beta"`, `"theta"`, and `"alpha"` options for the 2PL LSIRM, respectively. Similarly to the 1PL LSIRM, β_i decreases for later items, indicating that later items are more difficult than earlier items. In addition, respondents with more correctly answered items (that is, higher sum scores) have higher θ_k estimates. Lastly, the distribution of the posterior samples for α_i is similar across all items.

The `plot()` function can be used to create a visualization of the interaction map based on the 2PL LSIRM by setting `option` argument as `"interaction"`. Figures 9a and 9b show the original and rotated interaction maps, respectively. In the interaction maps based on the 1PL and 2PL LSIRM, we notice that although the clustering of items is similar, the distances between item groups appear smaller in the interaction map of the 2PL LSIRM. This difference arises because the 2PL LSRIM takes item discrimination (α) into account, which explains some degree of item-by-person interactions. The interaction map with the oblimin rotation (b) shows a slight clockwise rotation compared to the original interaction map (a), similarly to the 1PL LSIRM's case.

```
R > plot(lsirm_result, option = "interaction", chain.idx = 1)
R > plot(lsirm_result, option = "interaction", rotation = TRUE, chain.idx = 1)
```

The `plot()` function visualizes the cluster of the latent positions of items by using spectral clustering and the Neyman-Scott process modeling approach, achieved by setting the `cluster` option to `spectral` and `neyman`, respectively.

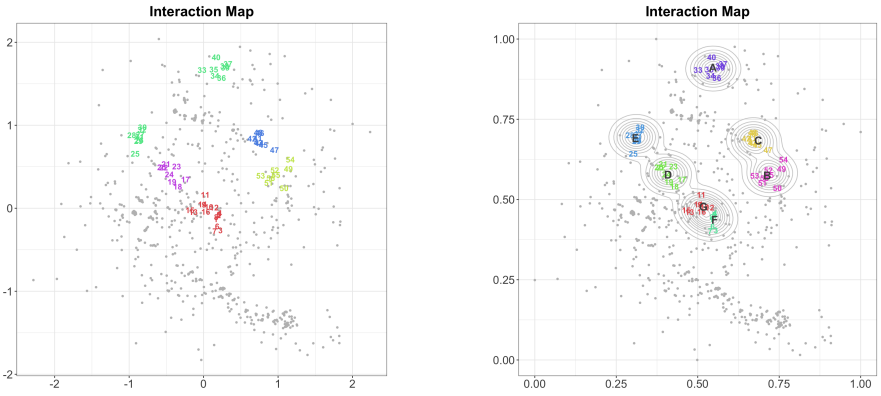
```
R > plot(lsirm_result, cluster = "spectral", chain.idx = 1)

Clustering result (Spectral Clustering):
  group                                item
      A         49, 50, 51, 52, 53, 54, 55, 56
      B         25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
                35, 36, 37, 38, 39, 40
      C         41, 42, 43, 44, 45, 46, 47, 48
      D         1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                11, 12, 13, 14, 15, 16
      E         17, 18, 19, 20, 21, 22, 23, 24

 R > plot(lsirm_result, cluster = "neyman", chain.idx = 1)

|================================================| 100%
```



(a) Original **(b)** Oblimin rotation

Figure 9: The interaction map based on the 2PL LSIRM fitted to the TDRI dataset. (a) Visualization of the interaction map based on the 2PL LSIRM. (b) A rotated interaction map using oblimin rotation. Red numbers and black dots represent the latent positions for items and respondents, respectively, in both plots.

```
Clustering result (Neyman-Scott process):
  group                                item
      A         33, 34, 35, 36, 37, 38, 39, 40
      B         49, 50, 51, 52, 53, 54, 55, 56
      C         41, 42, 43, 44, 45, 46, 47, 48
      D         17, 18, 19, 20, 21, 22, 23, 24
      E         25, 26, 27, 28, 29, 30, 31, 32
      F         1, 2, 3, 4, 5, 6, 7, 8
      G         9, 10, 11, 12, 13, 14, 15, 16
```



(a) Spectral Clustering **(b)** Neyman-Scott Process Model

Figure 10: The interaction map with the item clustering results based on the 2PL LSIRM fitted to the TDRI dataset, using (a) spectral clustering and (b) the Neyman-Scott process modeling approach. In both plots, the gray dots indicate respondents, where numbers in colors indicate items with different cluster memberships. The Neyman-Scott process approach additionally displays the center of the cluster (alphabets) and a contour for each cluster.

Figure 10a and 10b display the result of spectral clustering and the Neyman-Scott process approach, respectively. The interpretation of gray dots, numbers, alphabets, and contours is the same as the 1PL LSIRM case. Spectral clustering resulted in 5 clusters, whereas the Neyman-Scott process approach resulted in 7 clusters. That is, the Neyman-Scott process approach identified more item groups than the spectral clustering. The overall clustering results are similar to the 1PL LSIRM case.

The flexible modeling options discussed with the 1PL LSIRM can be applied to the 2PL LSIRM. That is, `fixed_gamma` fixes the distance weight γ to 1 and `spikenslab` assigns a spike and slab prior to γ. Additionally, the two missing data options, `missing_data = "mcar"` and `missing_data = "mar"`, are available for the 2PL LSIRM.

```
R > lsirm_result <- lsirm(data ~ lsirm2pl(fixed_gamma = TRUE))
R > lsirm_result <- lsirm(data ~ lsirm2pl(spikenslab = TRUE))
R > lsirm_result <- lsirm(data ~ lsirm2pl(missing_data = "mcar"))
R > lsirm_result <- lsirm(data ~ lsirm2pl(missing_data = "mar"))
```

5 Lsirm for continuous item responses data

We consider an extension of the LSIRM for continuous item response data (LSIRM-continuous), which is done by using an appropriate link function following the generalized linear model framework (McCullagh and Nelder, 2019).

5.1 Statistical framework

Consider the continuous item response data consisting of the $N \times P$ matrix $\boldsymbol{Y} = \{y_{k,i}\} \in \mathbb{R}^{N \times P}$. By choosing the identity link function for continuous item responses, the 1PL LSIRM-continuous is given as follows:

$$y_{k,i} \mid \boldsymbol{\theta}, \boldsymbol{\beta}, \gamma, \boldsymbol{Z}, \boldsymbol{W}, \sigma_\epsilon^2 = \theta_k + \beta_i - \gamma d(\boldsymbol{z_k}, \boldsymbol{w_i}) + \epsilon_{k,i}, \;\; \theta_k \sim \mathrm{N}(0, \sigma^2), \;\; \epsilon_{k,i} \sim \mathrm{N}(0, \sigma_\epsilon^2).$$

where θ_k and β_i are the main effects of the respondent k and the item i, respectively. $\boldsymbol{z_k}$ and $\boldsymbol{w_i}$ are the latent positions of the respondent k and the item i, respectively. An additional error term $\epsilon_{k,i} \sim N(0, \sigma_\epsilon^2)$ explains residuals unexplained by the 1PL LSIRM-continuous. A shorter distance between $\boldsymbol{w_i}$ and $\boldsymbol{z_k}$ indicates that the respondent k is likely to give a higher response value to item i given the main effects of the respondent and the item.

It is straightforward to extend the 1PL LSIRM-continuous to the 2PL version by adding item discrimination (or slope) parameters α_i. The 2PL LSIRM-continuous for $y_{k,i}$ is given as follows:

$$y_{k,i} \mid \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma, \boldsymbol{Z}, \boldsymbol{W}, \sigma_\epsilon^2 = \alpha_i \theta_k + \beta_i - \gamma d(\boldsymbol{z_k}, \boldsymbol{w_i}) + \epsilon_{k,i}, \;\; \theta_k \sim \mathrm{N}(0, \sigma^2), \;\; \epsilon_{k,i} \sim \mathrm{N}(0, \sigma_\epsilon^2).$$

The interpretations of the model parameters in the 2PL LSIRM-continuous are similar to the case of the 1PL LSIRM-continuous and the 2PL LSIRM. For model identifiability, one of the item slopes is fixed to 1, e.g., $\alpha_1 = 1$.

The likelihood function of the 1PL LSIRM-continuous is given as

$$\begin{aligned} \mathbf{L}\left(\boldsymbol{Y} = \boldsymbol{y} | \boldsymbol{\theta}, \boldsymbol{\beta}, \gamma, \boldsymbol{Z}, \boldsymbol{W}, \sigma_\epsilon^2\right) &= \prod_{k=1}^{N} \prod_{i=1}^{P} \mathbf{L}(Y_{k,i} = y_{k,i} | \theta_k, \beta_i, \gamma, \boldsymbol{z_k}, \boldsymbol{w_i}, \sigma_\epsilon^2), \\ &= \prod_{k=1}^{N} \prod_{i=1}^{P} N(y_{k,i}; \theta_k + \beta_i - \gamma ||\boldsymbol{z_k} - \boldsymbol{w_i}||, \sigma_\epsilon^2), \end{aligned}$$

and the likelihood function of the 2PL LSIRM-continuous is given as

$$\mathbf{L}\left(\boldsymbol{Y} = \boldsymbol{y} | \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma, \boldsymbol{Z}, \boldsymbol{W}, \sigma_\epsilon^2\right) = \prod_{k=1}^{N} \prod_{i=1}^{P} N(y_{j,i}; \alpha_i \theta_k + \beta_i - \gamma ||\boldsymbol{z_k} - \boldsymbol{w_i}||, \sigma_\epsilon^2).$$

5.2 Parameter Estimation

Following is the posterior distribution of LSRIM-continuous.

$$\pi\left(\boldsymbol{\theta}, \boldsymbol{\beta}, \gamma, \boldsymbol{Z}, \boldsymbol{W}, \sigma_\epsilon^2 | \boldsymbol{Y} = \boldsymbol{y}\right) \propto \prod_{k=1}^{N}\prod_{i=1}^{P} N(y_{k,i,}; \theta_k + \beta_i - \gamma||\boldsymbol{z_k} - \boldsymbol{w_i}||, \sigma_\epsilon^2) \times \prod_{k=1}^{N} \pi(\theta_k) \times \prod_{i=1}^{P} \pi(\beta_i) \times \pi(\gamma) \times \prod_{k=1}^{N} \pi(\boldsymbol{z_k}) \times \prod_{i=1}^{P} \pi(\boldsymbol{w_i}) \times \pi(\sigma_\epsilon^2)$$

The priors of the model parameters in the LSIRM-continuous are set the same as the priors for the LSIRM. For the error term σ_ϵ^2, we set the prior as follows:

$$\sigma_\epsilon^2 | a_{\sigma_\epsilon}, b_{\sigma_\epsilon} \sim \text{Inv-Gamma}(a_{\sigma_\epsilon}, b_{\sigma_\epsilon}),\ a_{\sigma_\epsilon} > 0,\ b_{\sigma_\epsilon} > 0.$$

The conditional posterior distributions of the LSIRM-continuous are similar to the LSIRM, which are given in the Github site. The jumping rule defaults remain the same as in the binary case (shown in the Github site).

5.3 An Illustrated Example

We demonstrate the application of the LSIRM-continuous using the Big Five Personality Test (FPT) dataset (Goldberg, 1992), which is included in the package. Data were collected from 1,015,342 respondents using an interactive online personality test from 2016 to 2018. The "Big-Five Factor Markers" from the international personality item pool were used in the test, which consists of 50 questions with response categories 1 = disagree, 3 = neutral, and 5 = agree based on a five-point Likert scale. Negatively worded items were reverse-coded. For illustration purposes, we randomly selected a sample of 3,000 respondents from the original data. We treated the ordinal item responses as continuous data, which is a common practice in applied research and is generally considered acceptable for other models, such as factor analysis.

The main fitting function for running the 1PL and 2PL LSIRM-continuous is identical to the 1PL and 2PL LSIRM for binary data. The function automatically identifies the data type and applies the appropriate models. Following is the code for data pre-processing and 1PL LSIRM-continuous fitting.

```
R > data <- lsirm12pl::BFPT
R > data[(data==0)|(data==6)] = NA
R > reverse <- c(2, 4, 6, 8, 10, 11, 13, 15, 16, 17,
                 18, 19, 20, 21, 23, 25, 27, 32, 34,
                 36, 42, 44, 46)
R > data[, reverse] <- 6 - data[, reverse]
R > data <- data[complete.cases(data),]
R > head(data)
    EXT1 EXT2 EXT3 ... OPN8 OPN9 OPN10
1    2    3    2   ...  2    4    4
2    1    3    3   ...  4    3    4
3    4    3    3   ...  2    4    4
5    1    4    3   ...  3    5    3
6    1    3    2   ...  3    5    3
8    3    5    4   ...  4    3    4

R > lsirm_result <- lsirm(data ~ lsirm1pl(niter = 25000, nburn = 5000, nthin = 10,
                                          jump_beta = 0.08, jump_theta = 0.3,
                                          jump_gamma  = 1.0,
                                          chains = 4, multicore = 2, seed = 2025))
```

To ensure convergence of the parameters, a different number of iterations and jumping rules of parameters are applied in this example. The estimated results for each chain are returned in the list containing estimated information on $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, γ, $\boldsymbol{Z}$, $\boldsymbol{W}$, σ, and σ_ϵ. The functions `summary()`, `diagnostics()`, and `gof()` described for the 1PL LSIRM can be applied similarly to obtain a summary, diagnosis, and goodness-of-fit results, respectively.

```
R > diagnostic(lsirm_result, draw.item = list(beta = c("AGR1")))
```

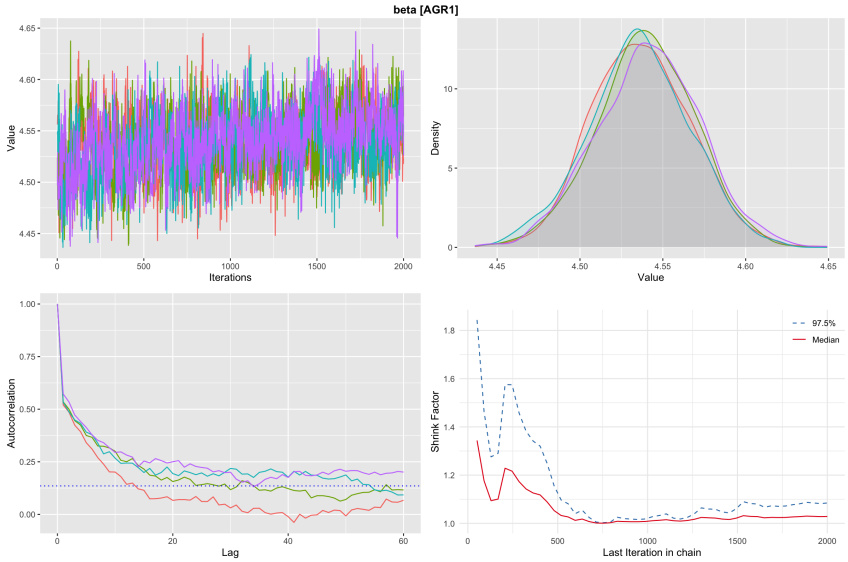


Figure 11: Diagnostics of β_{AGE_1} using the `diagnostic()` function on the results of the 1PL LSIRM-continuous fitted to the FPT dataset. The top left is the trace plot, the top right is the posterior density plot, the bottom left is the autocorrelation plot, and the bottom right is the Gelman-Rubin-Brooks plot. Different colors represent different MCMC chains.

Figure 11 displays the diagnostic results for β_{AGE_1} obtained using the `diagnostic()` function. The results suggest the convergence of β_{AGE_1} is achieved for this model.

Figure 12 shows the result of the goodness of fit assessment for the continuous example data. Unlike binary data, ROC is not available for continuous data, so the results of the `gof()` function include only the boxplots of the average predicted response values (item-wise means) against the observed item-wise means (marked with red dots). In this example, the red dots are located close to the midlines of the boxplots, implying a satisfactory model fit.

```
R > gof(lsirm_result, chain.idx = 1)
```

Same as before, the `plot()` function can be used to summarize the parameter estimate of β_i and θ_k.

```
R > plot(lsirm_result, option = "beta", chain.idx = 1)
R > plot(lsirm_result, option = "theta", chain.idx = 1)
```

Figure 13a shows the boxplots of the posterior samples for β_i. The parameter estimate of β_{35} is the smallest , while the estimates for β_{41} to β_{50} are relatively high. Figure 13b shows the boxplots of the point estimates of θ_k as a function of the sum scores of the observed responses binned into 10 groups to prevent overlap and improve readability on the x-axis. As expected, higher sum scores are aligned with higher θ_k values (indicating socially desirable characteristics).

Figure 14a illustrates the interaction map derived from the 1PL LSIRM-continuous. In the interaction map, the latent positions of the respondents are positioned around the center, while the items are scattered in a triangular pattern showing roughly three clusters (in the

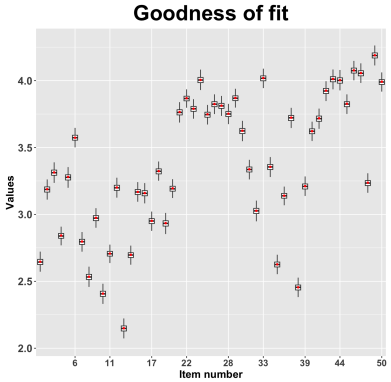
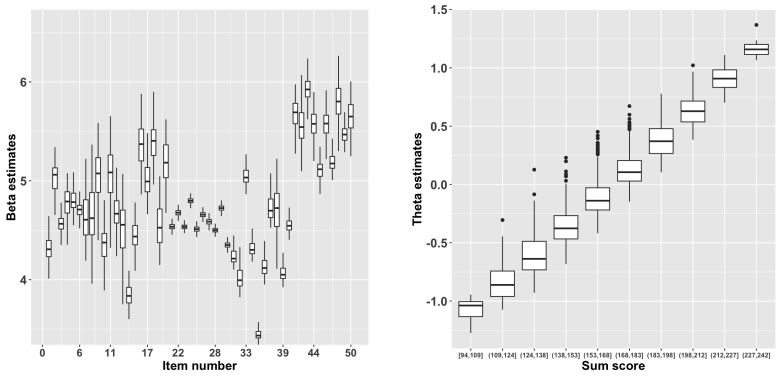


Figure 12: Goodness of fit of the 1PL LSIRM-continuous for the BFPT dataset using the `gof()` function. The box plots of the average posterior predicted response value for items against the average observed response value for each item indicated by red dots.



(a) β_i **(b)** θ_k

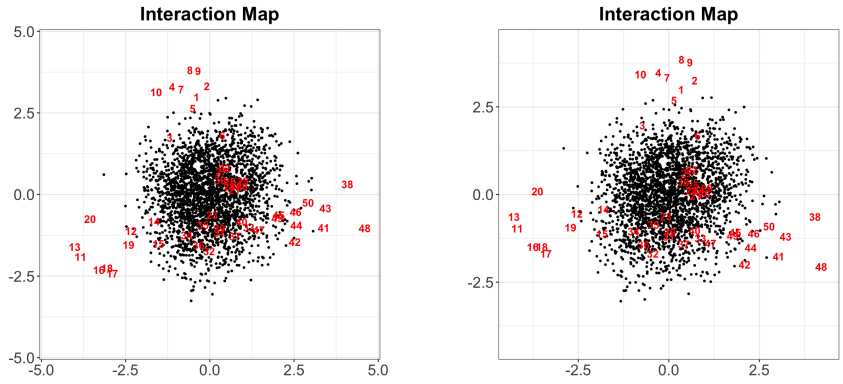
Figure 13: Visual summaries of β_i and θ_k using the `plot()` function on the 1PL LSIRM-continuous fitted to the FPT dataset. (a) Boxplots of the posterior samples for β_i. (b) Boxplots of the point estimates of θ_k as a function of the total sum scores of the responses binned into 10 groups

north, west, and east) in addition to the cluster around the center. The original interaction map is slightly rotated counterclockwise in the rotated interaction map, so that the items are more closely placed to the two coordinates.

```
R > plot(lsirm_result, chain.idx = 1)
R > plot(lsirm_result, rotation = TRUE, chain.idx = 1)
```

Figure 15a and Figure 15b depict the result of spectral clustering and the Neyman-Scott process modeling approach for the BFPT example dataset, respectively. Gray dots, numbers with colors, alphabets, and contours have the same interpretations as the clustering results presented earlier with the binary LSIRM models. In Figure 15b, the items in cluster C, highlighted in purple and located near the center of the interaction map, are closer to the latent positions of most people compared to other items. This implies that these items are more likely to receive higher response values. Conversely, items in clusters that are positioned farther from the center, such as clusters A, B, and E, are distant from many (but different groups of) respondents, indicating they are likely to receive lower response values by those who are far away from the corresponding item clusters.

```
R > plot(lsirm_result, cluster = "spectral", chain.idx = 1)

Clustering result (Spectral Clustering):
  group                                        item
```



(a) Original **(b)** Oblimin rotation

Figure 14: The interaction map based on the 1PL LSIRM-continuous fitted to the FPT dataset. (a) Visualization of the interaction map based on the 1PL LSIRM-continuous. (b) A rotated interaction map of the 1PL LSIRM-continuous using oblimin rotation. Red numbers and black dots represent the latent positions for items and respondents, respectively.

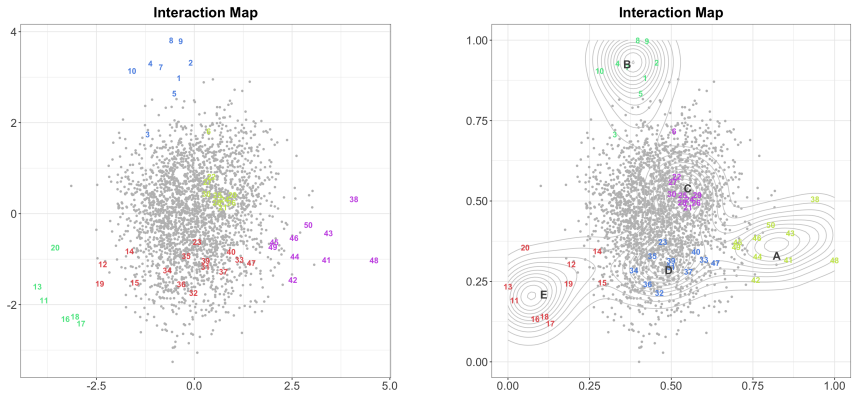
```
      A          11, 13, 16, 17, 18, 20
      B          38, 41, 42, 43, 44, 45, 46, 48, 49, 50
      C          1, 2, 3, 4, 5, 7, 8, 9, 10
      D          6, 21, 22, 24, 25, 26, 27, 28, 29, 30
      E          12, 14, 15, 19, 23, 31, 32, 33, 34, 35,
                 36, 37, 39, 40, 47

R > plot(lsirm_result, cluster = "neyman", chain.idx = 1)

|=================================================| 100%
Clustering result (Neyman-Scott process):
  group                                        item
      A          38, 41, 42, 43, 44, 45, 46, 48, 49, 50
      B          1, 2, 3, 4, 5, 7, 8, 9, 10
      C          6, 21, 22, 24, 25, 26, 27, 28, 29, 30
      D          23, 31, 32, 33, 34, 35, 36, 37, 39, 40, 47
      E          11, 12, 13, 14, 15, 16, 17, 18, 19, 20
```

The flexible modeling options discussed with the binary LSIRM can be applied to the functions for the LSIRM-continuous.

```
R > lsirm_result <- lsirm(data ~ lsirm1pl(fixed_gamma = TRUE))
R > lsirm_result <- lsirm(data ~ lsirm1pl(spikenslab = TRUE))
R > lsirm_result <- lsirm(data ~ lsirm1pl(missing_data = "mcar"))
R > lsirm_result <- lsirm(data ~ lsirm1pl(missing_data = "mar"))
```

6 Conclusion

In this paper, we introduced the R package **lsirm12pl** for estimating the LSIRM (Jeon et al., 2021) and its extensions. The LSIRM is a powerful extension of conventional IRT models that allow for the estimation and visualization of potential interactions between respondents and items in an interaction map, a low dimensional Euclidean space. The original LSIRM framework was proposed for binary item response data based on the 1PL IRT base model. To broaden its applicability, we extended the original LSIRM to cover different response types and model specifications. Further, we added several useful options, e.g., for handling missing data, item clustering, and model assessment.



(a) Spectral Clustering **(b)** Neyman-Scott Process Model

Figure 15: The interaction map with item clustering results based on the 1PL LSIRM model-continuous fitted to the FPT dataset, using (a) spectral clustering and (b) Neyman-Scott process approach. In both plots, the gray dots indicate respondents, where numbers in colors indicate items with different cluster memberships. The Neyman-Scott process approach additionally displays the center of the cluster (alphabets) and a contour for each cluster.

It is worth noting that one may observe some variability in clustering results across multiple chains. Such variability stems from at least three factors: First, clustering is an unsupervised method, meaning that results are not uniquely determined and may naturally vary across Markov chains. Second, we apply the Neyman-Scott Process clustering to the posterior distributions of the LSIRM model, and the clustering outcomes are therefore directly influenced by the estimated LSIRM results. Third, Bayesian models inherently incorporate uncertainty, reflected in the posterior distributions and subsequently in the clustering outcomes. Having said that, we noted in our empirical analysis that the overall clustering structure was pretty stable across multiple chains. For critical applications, we recommend implementing a consensus clustering approach by aggregating results across multiple chains to identify the most robust and stable groupings.

A fully Bayesian approach was used for model estimation with a Metropolis-Hastings-within-Gibbs sampler. The **lsirm12pl** package offers default estimation settings for priors, jumping rules, number of iterations, burn-ins, and thinning. The default estimation setting works reasonably well in a broad range of situations, but users can manually revise the estimation settings if desired. In addition, the package **lsirm12pl** offers convenient supplemental functions to evaluate, summarize, visualize, diagnose, and interpret the estimated results. We provided detailed illustrations of the package with real data examples available in the package. We hope that these illustrations guide researchers in using the **lsirm12pl** package for the analysis of their own datasets.

The R package **lsirm12pl** is our first step in making the LSIRM approach more applicable and usable in practice. The code is written using **Rcpp** (Eddelbuettel and François, 2011; Eddelbuettel, 2013; Eddelbuettel and Balamuta, 2018) and **RcppArmadillo** (Eddelbuettel and Sanderson, 2014) in R for efficient computation. For example, in our first data example with 726 respondents and 56 test items, the computation time for one chain using a single core `lsirm1pl` was 1.93 minutes on an Apple M1 laptop. We provide additional details of our package implementation in the package GitHub site (https://github.com/jiniuslab/lsirm12pl). We will continue to update the package by incorporating additional modeling, data analysis, and visualization options to make the **lsirm12pl** package more useful in a wider range of situations. For example, we are currently engaged in the development of other extensions, such as for ordinal and longitudinal data. As advancements are made, we will systematically include them in the software package. This ongoing development will further enhance the utility and applicability of the LSIRM in various practical scenarios.

7 Acknowledgement

We thank the Editor, Associate Editor, and reviewers for their constructive comments. We also thank Dr. Won Chang for constructive comments on our work. This work was partially supported by the National Research Foundation of Korea [grant number NRF-2020R1A2C1A01009881, RS-2023-00217705, RS-2024-00333701; Basic Science Research Program awarded to IHJ], [grant number NRF-2025S1A5C3A0200632411; awarded to IHJ and MJ], and the ICAN (ICT Challenge and Advanced Network of HRD) support program [grant number RS-2023-00259934], supervised by the IITP (Institute of Information & Communications Technology Planning & Evaluation). Correspondence should be addressed to Ick Hoon Jin, Department of Applied Statistics, Department of Statistics and Data Science, Yonsei University, Seoul, Republic of Korea. Go, Kim, and Park are co-first authors and listed in alphabetical order.

References

X. An and Y. F. Yung. Item response theory: What it is and how you can use the IRT procedure to apply it. *SAS Institute Inc*, 10(4), 2014. [p4, 6]

E. B. Andersen. Conditional inference for multiple-choice questionnaires. *British Journal of Mathematical and Statistical Psychology*, 26(1):31–44, 1973. doi: 10.1111/j.2044-8317.1973.tb00504.x. [p4]

D. Andrich. A rating formulation for ordered response categories. *Psychometrika*, 43(4):561–573, 1978. doi: 10.1007/bf02293814. [p4]

F. B. Baker and S.-H. Kim, editors. *Item response theory*. Statistics: A Series of Textbooks and Monographs. Taylor & Francis, London, England, 2 edition, 2023. [p4]

F. Batool and C. Hennig. Clustering with the average silhouette width. *Computational Statistics & Data Analysis*, 158:107190, 2021. doi: 10.1016/j.csda.2021.107190. [p13]

C. A. Bernaards and R. I. Jennrich. Gradient projection algorithms and software for arbitrary Rotation criteria in factor analysis. *Educational and Psychological Measurement*, 65(5):676–696, Oct. 2005. doi: 10.1177/0013164404272507. [p12]

A. L. Birnbaum. Some latent trait models and their use in inferring an examinee's ability. *Statistical theories of mental test scores*, 1968. [p4, 6]

J. Braeken. A boundary mixture approach to violations of conditional independence. *Psychometrika*, 76(1):57–76, 2010. ISSN 1860-0980. doi: 10.1007/s11336-010-9190-4. URL http://dx.doi.org/10.1007/s11336-010-9190-4. [p5]

J. Brzezińska. Item response theory models in the measurement theory with the use of ltm package in R. *Econometrics*, 22(1):11–25, 2018. doi: doi:10.15611/eada.2018.1.01. [p4]

R. P. Chalmers. mirt: A multidimensional item response theory package for the R environment. *Journal of Statistical Software*, 48(6):1–29, 2012. doi: 10.18637/jss.v048.i06. [p4]

C.-T. Chen and W.-C. Wang. Effects of ignoring item interaction on item parameter estimation and detection of interacting items. *Applied Psychological Measurement*, 31(5):388–411, 2007. ISSN 1552-3497. doi: 10.1177/0146621606297309. URL http://dx.doi.org/10.1177/0146621606297309. [p4]

S. Chib and E. Greenberg. Understanding the Metropolis–Hastings algorithm. *The American Statistician*, 49:327–335, 1995. doi: https://doi.org/10.2307/2684568. [p7]

R. J. de Ayala. *The theory and practice of item response theory*. Methodology in the Social Sciences. Guilford Publications, New York, NY, 2009. [p4]

D. Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer, New York, 2013. ISBN 978-1-4614-6867-7. [p25]

D. Eddelbuettel and J. J. Balamuta. Extending R with C++: A brief introduction to Rcpp. *The American Statistician*, 72(1):28–36, 2018. doi: 10.1080/00031305.2017.1375990. [p25]

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. doi: 10.18637/jss.v040.i08. [p25]

D. Eddelbuettel and C. Sanderson. Rcpparmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063, March 2014. doi: 10.1016/j.csda.2013.02.005. [p25]

G. H. Fischer and P. Parzer. An extension of the rating scale model with an application to the measurement of change. *Psychometrika*, 56(4):637–651, 1991. doi: 10.1007/bf02294496. [p4]

G. H. Fischer and I. Ponocny. An extension of the partial credit model with an application to the measurement of change. *Psychometrika*, 59(2):177–192, 1994. doi: 10.1007/bf02295182. [p4]

C. A. W. Glas and N. D. Verhelst. Extensions of the partial credit model. *Psychometrika*, 54(4): 635–659, 1989. doi: 10.1007/bf02296401. [p4]

L. R. Goldberg. The development of markers for the big-five factor structure. *Psychological Assessment*, 4(1):26–42, 1992. ISSN 1040-3590. doi: 10.1037/1040-3590.4.1.26. URL http://dx.doi.org/10.1037/1040-3590.4.1.26. [p21]

H. Golino. TDRI dataset, 2016. URL https://figshare.com/articles/dataset/TDRI_dataset_csv/3142321/1. [p8]

J. C. Gower. Generalized Procrustes analysis. *Psychometrik*, 40:33–51, 1975. doi: 10.1007/bf02291478. [p7]

P. Hoff, A. Raftery, and M. S. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97:1090–1098, 2002. doi: 10.1198/016214502388618906. [p7]

C. Hohensinn. pcIRT: An R package for polytomous and continuous Rasch models. *Journal of Statistical Software, Code Snippets*, 84(2):1–14, 2018. doi: 10.18637/jss.v084.c02. [p4]

H. Ishwaran and J. S. Rao. Spike and slab variable selection: Frequentist and Bayesian strategies. *The Annals of Statistics*, 33(2), Apr. 2005. doi: 10.1214/009053604000001147. URL https://doi.org/10.1214/009053604000001147. [p14]

R. I. Jennrich. A simple general method for oblique rotation. *Psychometrika*, 67(1):7–19, Mar. 2002. doi: 10.1007/bf02294706. [p12]

M. Jeon, I. H. Jin, M. Schweinberger, and S. Baugh. Mapping unobserved item–respondent interactions: A latent space item response model with interaction map. *Psychometrika*, pages 1–26, 2021. doi: 10.1007/s11336-021-09762-5. [p5, 6, 7, 11, 12, 24]

A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004. doi: 10.18637/jss.v011.i09. [p13]

P. Mair and R. Hatzinger. Extended Rasch modeling: The eRm package for the application of IRT models in R. *Journal of Statistical Software*, 20(9):1–20, 2007. doi: 10.18637/jss.v020.i09. [p4]

G. N. Masters. A Rasch model for partial credit scoring. *Psychometrika*, 47(2):149–174, 1982. doi: 10.1007/bf02296272. [p4]

P. McCullagh and J. A. Nelder. *Generalized linear models*. Routledge, 2019. [p20]

H. Müller. A Rasch model for continuous ratings. *Psychometrika*, 52(2):165–181, 1987. doi: 10.1007/bf02294232. [p4]

N. Myszkowski and M. Storme. Modeling sequential dependencies in progressive matrices: An auto-regressive item response theory (AR-IRT) approach. *Journal of Intelligence*, 12(1):7, 2024. ISSN 2079-3200. doi: 10.3390/jintelligence12010007. URL http://dx.doi.org/10.3390/jintelligence12010007. [p5]

J. Neyman and E. L. Scott. A theory of the spatial distribution of galaxies. *The Astrophysical Journal*, 116:144 –163, 1952. doi: 10.1086/145599. [p13]

A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, page 849–856, Cambridge, MA, USA, 2001. MIT Press. [p12]

G. Rasch. *Probabilistic models for some intelligence and attainment tests*. Danish Institute for Educational Research, Copenhagen, 1960. [p4]

G. Rasch. On general laws and the meaning of measurement in psychology. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 4: Contributions to Biology and Problems of Medicine*, pages 321–333, Berkeley, Calif., 1961. University of California Press. URL https://projecteuclid.org/euclid.bsmsp/1200512895. [p5]

D. Rizopoulos. ltm: An R package for latent variable modeling and item response analysis. *Journal of Statistical Software*, 17(5):1–25, 2006. doi: 10.18637/jss.v017.i05. [p4]

D. B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976. [p5, 14]

F. Samejima. Estimation of latent ability using a response pattern of graded scores. *ETS Research Bulletin Series*, 1968(1):i–169, 1968. doi: 10.1002/j.2333-8504.1968.tb00153.x. [p4]

H. Scheiblechner. Das Lernen und Lösen komplexer Denkaufgaben. *Zeitschrift für experimentelle und angewandte Psychologie*, 19:476–506, 1972. [p4]

D. K. Sewell and Y. Chen. Latent space models for dynamic networks. *Journal of the American Statistical Association*, 110:1646–1657, 2015. [p7]

M. Tanner and W. Wong. The calculation of posterior distributions by data augmentation (with discussion). *Journal of the American Statistical Association*, 82:528–550, 1987. [p15]

M. Thomas. A generalization of Poisson's binomial limit for use in ecology. *Biometrika*, 36 (1/2):18–25, 1949. doi: 10.2307/2332526. [p13]

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007. doi: 10.1007/s11222-007-9033-z. [p12]

S. Yi, M. Kim, J. Park, M. Jeon, and I. H. Jin. Impacts of innovation school system in Korea: A latent space item response model with Neyman-Scott point process, 2024. URL https://arxiv.org/abs/2306.02106. [p13]

C. Zanon, C. S. Hutz, H. Yoo, and R. K. Hambleton. An application of item response theory to psychological test development. *Psicologia: Reflexão e Crítica*, 29(1), 2016. doi: 10.1186/s41155-016-0040-x. [p4]

Dongyoung Go
Department of Statistics and Data Science
Department of Applied Statistics
Yonsei University
50 Yonsei-ro, Seodaemun, Seoul, Republic of Korea
dongyoung.gr@yonsei.ac.kr

Gwanghee Kim
Department of Statistics and Data Science
Yonsei University
50 Yonsei-ro, Seodaemun, Seoul, Republic of Korea
musagh08@yonesi.ac.kr

Jina Park
Department of Statistics and Data Science
Department of Applied Statistics
Yonsei University
50 Yonsei-ro, Seodaemun, Seoul, Republic of Korea
pja070707@yonesi.ac.kr

Junyong Park
Samsung Electronics, Formerly at Yonsei University
1 Samsung Electronics-ro, Hwaseong-si, Gyeonggi-do, Republic of Korea
jun94.park@samsung.com

Minjeong Jeon
School of Education and Information Studies
University of California, Los Angeles
405 Hilgard Avenue, Los Angeles, CA 90095
mjjeon@ucla.edu

Ick Hoon Jin
Department of Statistics and Data Science
Department of Applied Statistics
Yonsei University
50 Yonsei-ro, Seodaemun, Seoul, Republic of Korea
ijin@yonsei.ac.kr

Converting LaTeX Legacy R Journal Articles into R Markdown Articles using texor and rebib

by Abhishek Ulayil, Dianne Cook, Heather Turner, Mitchell O'Hara-Wild, and Christophe Dervieux

Abstract In 2021 the R Journal made a change of templates for article writing to R Markdown instead of LaTeX. The reasons were to encourage better reproducibility of articles using dynamic documents, enable interactivity in articles, and to make the articles more accessible for print-disabled readers. A resulting challenge was to explore whether legacy articles might be suitably converted into HTML output. This paper describes the process of converting an R Journal article from LaTeX to HTML format via R Markdown, and the two new R packages, texor and rebib, that can be used to achieve the conversion.

1 Introduction

The R Journal is the primary open-access outlet for publications produced by the R community. It was started in 2009, evolving from the R News newsletter that ran from 2001, to become a more formal publication that encourages and provides credit for the documentation of statistical computing research.

The tooling behind the production of the R Journal is regularly updated. Early articles were typeset using LaTeX (The LaTeX Project, 2023), based on a specific, but changing, template. Using LaTeX requires that code is separated from the documentation, and there is a chance that code chunks in the paper don't reproduce the results reported. With the emergence of dynamic document systems such as R Markdown (Xie et al., 2018), a tight coupling of code and documentation is possible. Code chunks are dynamically executed when the document is typeset using a system like **knitr** (Xie, 2015), making reporting of computing research more reproducible.

In 2019, with the help of funding from the R Consortium, work began to update operations to adopt an R Markdown template, that would produce both HTML and PDF versions of the paper. In mind, was also to develop, adopt and encourage good standards for R Journal articles.

There are numerous benefits of HTML format:

1. Articles can include interactive graphics (to allow readers to query, re-focus, and watch an animation, for example) and interactive tables (enabling re-sorting in different ways to better digest the information).
2. It is the best format for accessibility (The Daisy Consortium, 2023). Screen readers and other assistive technologies can deliver the content to print-disabled readers. (See Byrne-Haber et al. (2023) for current guidelines.)
3. HTML provides a more comfortable reading experience on mobile devices, which are increasingly used as researchers work on the move and share work via social media.
4. Search engines can easily access the full text of articles, facilitating the discovery of published articles.

These provide the motivations for converting all of the legacy R Journal articles into HTML.

A key decision when designing the conversion software was whether to convert the LaTeX source to HTML; the PDF output to HTML; or the LaTeX source to R Markdown, which would then be converted to HTML using the current journal tools. The latter approach was decided to be the most versatile and useful. If an article can be converted from LaTeX to R Markdown, it would help authors make the transition to reproducible publishing, beyond

what the R Journal needed. Once an article is in R Markdown format it can be adapted to include the code for dynamic execution.

In addition to the article format, changes to the website structure were important for delivering the publication. Website architectures are also constantly evolving, and the emergence of **distill** (Dervieux et al., 2022) allows the journal website to optimally deliver R Markdown articles.

The **rjtools** was developed to create articles using R Markdown for the R Journal and to embed them into the journal website. The packages described here, **texor** and **rebib**, describe software to convert legacy LaTeX format articles into R Markdown, so that they can be rendered in HTML in the new website.

The paper is organised as follows. Section 2 gives an overview of the conversion process, which includes pre-processing using regular expressions, post-processing using Lua filters, and handling of figures, tables and equations. Section 3 describes the **texor** package that handles most of the conversion. Section 4 describes tools for special handling of bibliography files. The supplementary materials have folders containing specific examples that can be used for understanding how the conversions are done.

2 The internals of converting from LaTeX to R Markdown

The decision to convert to R Markdown format means that the final output to PDF and HTML will depend on Pandoc (MacFarlane et al., 2023). Pandoc is a versatile document conversion program written in Haskell that is core to numerous documentation systems, including R Markdown and Quarto. Pandoc first converts a document into an abstract syntax tree. From this, it can convert to a different format, including custom ones.

Pandoc can be used to do the conversion from LaTeX to R Markdown also. However, some pre-processing is necessary to handle special R Journal LaTeX styling. And further post-processing is necessary to handle specific R Journal R Markdown styling. The **texor** package contains functionality to handle this pre- and post-processing of the document, in a workflow illustrated in Figure 1.

2.1 Pre-processing using regular expressions

LaTeX is a very descriptive language, that allows authors substantial freedom for customization. Markdown (Gruber, 2002), on which R Markdown is based, is more restrictive and was designed to make it easier to create web pages without the distraction of a gazillion HTML tags. The beauty of Markdown is that it allows the author to focus on writing, without format cluttering the text. The drawback is that it is simple typesetting, optimized for web delivery.

While Pandoc can do most of the heavy lifting, it cannot cope with all the freedom with which LaTeX documents are written. An example of this is with formatting of code. Pandoc only handles the `verbatim` environment, but there are many ways to format code in LaTeX, and the R Journal template has a special `\code{}` command. If the code environment is not verbatim, then Pandoc will also try to process the actual code content as LaTeX commands and will likely lose details. It is better to convert these synonyms into `verbatim` environments prior to passing the document to Pandoc.

The functions in **texor** that handle the pre-processing using regular expressions are:

- `stream_editor()`: operates like the `sed` function in unix (Ritchie and Thompson, 1978) and allows generic text pattern matching and replacement.
- `patch_code_env()`: replaces the common code environments, `code`, `example`, `Sin`, `Sout`, `Scode`, `Sinput`, `smallverbatim`, `boxedverbatim`, `smallexample` with `verbatim`.
- `patch_equations()`: handles various equation environments.
- `patch_figure_env()`: handles various figure environments.
- `patch_table_env()`: handles various table environments.

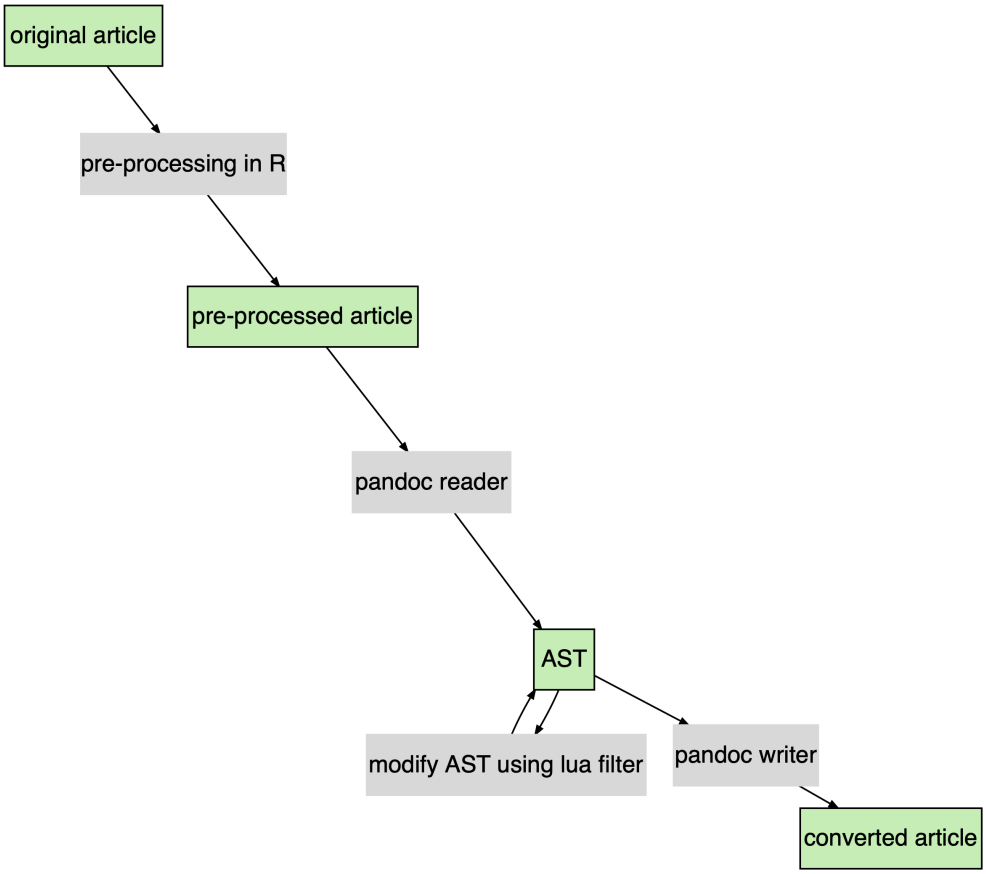


Figure 1: Workflow of the document conversion conducted by texor. (Note: AST is pandoc's abstract syntax tree.)

These functions are verbose and describe all the changes being made. They also create a backup of the original file before making the changes.

2.2 Post-processing using Lua filters

Lua (Ierusalimschy et al., 1996) is a programming language, that is lightweight and fast: ideal for procedural operations. It is embedded in many other applications to allow custom scripting for extensibility. Pandoc allows users to provide custom Lua filters to produce custom output formats. The **texor** package handles the post-processing of the R Markdown document into the special format for the R Journal using a suite of Lua filters.

Here is an example of a Lua filter available in **texor**:

```
function Div(el)
    if el.classes[1] == 'thebibliography' then
        return { }
    end
end
```

This filter reads the abstract syntax tree, selecting all the Div elements. Then it looks for the class "thebibliography." This Div element contains the LaTeX bibliographic records, that appear at the very end of papers. It should not be in the document when using the "RJ-web-article" layout, because it is added from meta-data when the R Markdown is knitted. So the Lua filter removes this section.

2.3 Figures

Standard, single figure

Figure definitions in LaTeX are many and varied! The standard, single-figure definition with the `figure` environment and raster image format such as PNG or JPG, is handled by Pandoc. It will convert:

```
\begin{figure}[htbp]
  \centering
  \includegraphics[width=0.35\textwidth]{Rlogo-5.png}
  \caption{The logo of R.}
  \label{figure:rlogo}
\end{figure}
```

to

```
<figure id="figure:rlogo">
<img src="Rlogo-5.png" style="width:35.0%" />
<figcaption>Figure 1: The logo of R.</figcaption>
</figure>
```

PDF format images

Images in PDF format are converted to PNG, in the pre-processing of the LaTeX document, and then post-processed using Pandoc as described above.

Multiple figures

Multiple figures are supported with the latest versions of Pandoc, so definitions like:

```
\begin{figure*}[htbp]
  \centering
  \includegraphics[width=0.45\textwidth]{Rlogo-5.png}
  \includegraphics[width=0.45\textwidth]{normal}
  \caption{Images side by side}
  \label{fig:twoimages}
\end{figure*}
```

will be converted to:

```
<figure id="fig:twoimages">
<p><img src="Rlogo-5.png" style="width:45.0%" alt="image" /><img
src="normal.png" style="width:45.0%" alt="image" /></p>
<figcaption>Figure 3: Images side by side</figcaption>
</figure>
```

`tikz` format images

Some legacy articles define images using `tikz` commands, such as:

```
\begin{figure}

%% Generated Image will included as a PNG above automatically
  \centering
\tikzstyle{process} = [rectangle, rounded corners,
minimum width=3cm,
minimum height=1cm,
text centered,
draw=black]
\tikzstyle{arrow} = [thick,->,>=stealth]
\begin{tikzpicture}[node distance=4cm]
%Nodes
...
```

This is handled by pre-processing the LaTeX to create the image, as both PDF, and then PNG, for inclusion in the R Markdown document using:

```
<figure id="fig:tikz">
<img src="tikz/figtikz.png" style="width:100.0%" />

<figcaption>Figure 5: Tikz Image example</figcaption>
</figure>
```

Algorithm2e graphics as figures

Algorithm2e graphics using the `algorithm2e` package in LaTeX are supported, and the following description will yield the result in Figure 2.

```
\begin{algorithm}[htbp]
\SetAlgoLined
\KwData{this text}
\KwResult{how to write algorithm with \LaTeX2e }
initialization\;
\While{not at end of this document}{
read current\;
\eIf{understand}{
```

Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while *not at end of this document* **do**
  read current;
  if *understand* **then**
    go to next section;
    current section becomes this one;
  else
    go back to the beginning of current section;
  end
end

Figure 2: How to write algorithms.

```
go to next section\;
current section becomes this one\;
}{
go back to the beginning of current section\;
}
}
\caption{How to write algorithms}
  \label{alg:how}
\end{algorithm}

<figure id="alg:how">
<img src="alghow.png" style="width:100.0%" />

<figcaption>Algorithm 1: How to write algorithms</figcaption>
</figure>
```

2.4 Equations

Math is handled primarily by Pandoc. The inline math and equation descriptions are unchanged between LaTeX and R Markdown.

The HTML output renders math using MathJax. This does mean that some functionality, like `\bm`, `\boldmath` and `\mathbbm` are not supported, and special definitions can only be handled in a limited capacity.

The numbering of equations is a bit trickier. LaTeX automatically numbers equations, unless specifically instructed not to. Equation numbering in R Markdown requires specific labeling using `(\#eq:xx)` as described in Xie (2023). The **texor** helps by adding the labeling using a Lua filter to convert the existing `\label{..}` to `(\#eq:xx)`.

2.5 Tables

Tables form one of the biggest challenges in migrating from LaTeX to R Markdown, because the sophistication is not completely replicated. However, there have been many improvements in table definitions for R Markdown that are increasingly producing the beautifully crafted tables possible in LaTeX. The conversion in **texor** can mostly handle the simple tables, and for producing more complex tables it may be necessary to manually edit the resulting `Rmd` file to make conditional tables, one to render specifically for HTML output using packages such as **kableExtra** (Zhu, 2021), **gt** (Iannone et al., 2023), **htmlTable** (Gordon et al., 2022), **tableHTML** (Boutaris et al., 2023), **tables** (Murdoch, 2023) or **DT** (Xie et al., 2023). A benefit of using conditional markup is that it can take advantage of HTML-specific features, such as sortable columns or paged tables.

Generic tables

Simple LaTeX tables are converted into traditional markdown format tables by Pandoc. So this table definition:

```
\begin{table}[htbp]
\centering
\begin{tabular}{l | llll }
 \hline
 Graphics Format & LaTeX & Markdown & Rmarkdown & HTML \\
 \hline
 PNG       & Yes & Yes & Yes & Yes \\
 JPG       & Yes & Yes & Yes & Yes \\
 PDF       & Yes & No & No & No \\
 SVG       & No & Yes & Yes & Yes \\
 Tikz      & Yes & No & Yes & No \\
 Algorithm & Yes & No & No & No \\
\hline
\end{tabular}
\caption{Image Format support in various Markup/Typesetting Languages}
\label{table:1}
\end{table}
```

will be converted to:

```
::: {#table:1}
  ------------------------------------------------------
  Graphics Format   LaTeX   Markdown   Rmarkdown   HTML
  ----------------- ------- ---------- ----------- ------
  PNG               Yes     Yes        Yes         Yes

  JPG               Yes     Yes        Yes         Yes

  PDF               Yes     No         No          No

  SVG               No      Yes        Yes         Yes

  Tikz              Yes     No         Yes         No

  Algorithm         Yes     No         No          No
  ------------------------------------------------------

  : Table 1: Image Format support in various Markup/Typesetting
  Languages
:::
```

Multicolumn tables

A multicolumn table requires:

1. The stream editor modifies the `\multicolumn{..}` to `\multicolumnx{..}`.
2. A LaTeX macro is used to redefine the `\multicolumnx{..}` to `\multicolumn{---}` (which is accepted by pandoc).
3. Pandoc reads the table and transforms it to markdown.

EXAMPLE	*X*		*Y*	
	1	2	1	2
EX1	X11	X12	Y11	Y12
EX2	X21	X22	Y21	Y22
EX3	X31	X32	Y31	Y32
EX4	X41	X42	Y41	Y42
EX5	X51	X52	Y51	Y52

Table 1: An example multicolumn table.

Also note that the stream editor is used to rename `table*` environment to `table` environment because the HTML format is single column, so the asterisk indicating that the table should be drawn over the full width of the page is redundant in this case.

Other tables

Tables with images, math, code or links in the cells are generally handled. Also `widetable` tables that allow for specific width or wrapping of tables into blocks are also partially handled.

3 Using texor

The package `texor` can be installed from CRAN and the development version from https://github.com/Abhi-1U/texor. The website for the package, https://abhi-1u.github.io/texor, has vignettes documenting usage.

Note that you will need to use Pandoc Version > 3.0.0 (if possible latest) for the best results. You can check your version with:

```
rmarkdown::pandoc_version()
```

The only function that a user will typically require is `latex_to_web()`. This creates the R Journal style R Markdown file from a given R Journal style LaTeX file. This is achieved by several sequential steps: `convert_to_markdown()`, `generate_rmd()`, and `produce_html()`.

For converting the 14 years of legacy R Journal articles, batch processing of issues was conducted.

For individuals who are interested in submitting their paper to the R Journal but have written their article using the legacy LaTeX format and wish to convert it to the current R Markdown format, you can use the `latex_to_web()` function on your paper directory. This will get you about 80% of the way to an R Markdown version of your paper. You will then want to:

- Edit the lines where figures are included. The conversion will create HTML code to define the image. This should be changed into Markdown description `![Caption](Image)` or using `knitr::include_graphic(Image)` in an R code chunk, in order for both HTML and PDF versions of the paper to be created when the `rjtools` article template is knitted.
- Edit the tables for aesthetics. The conversion will create Markdown tables, which will convert appropriately to HTML and PDF. However, to have more control and to create more elegant tables using `knitr::kable` provides a finer level of control.
- Include your R code to dynamically do the computing described in your article. If any code block is time-consuming to complete, saving intermediate output, or caching would be recommended.

4 Managing the bibliography using rebib

Typically bibliographies are generated during the processing of a LaTeX article using the BibTeX software (Feder, 2006) operating on a `.bib` list of references. The current R Journal template requires the inclusion of the `.bib` file. But LaTeX actually uses a `.bbl` format for references, which is what BibTeX generates as an intermediate format during the article processing.

The instructions for R Journal authors have changed over time. Initially, authors were instructed to run BibTeX on their article and to include the `.bbl` formatted references directly in the `.tex` file. This was to avoid clashes in citation keys between multiple articles in an issue. Later on, authors were instructed to provide a `.bib` file instead. During the conversion of legacy articles, it was discovered that some papers had references in both the `.tex` file and in separate `.bbl` or `.bib` files. Usually, the different files were equivalent, but sometimes references were only found in one source and in some cases the files contained conflicting information! While LaTeX can technically handle either `.bbl` or `.bib` formatted references, R Markdown can only handle `.bib`.

The **rebib** package was developed to handle these tricky situations. It converts embedded LaTeX bibliographies into a close BibTeX equivalent. The features of the package are:

- extracting embedded bibliographic entries from a `.tex` file;
- creating the mandatory title and author fields;
- creating the optional URL, ISBN, publisher, pages and year fields, when available;
- storing remaining information in `"journal"` (internally) and `"publisher"` (when writing BibTeX file);
- ignoring commented LaTeX code;
- tracking citations included in the document, and
- aggregating references from embedded bibliographic entries with references from supplementary `.bbl` and `.bib` files.

The package `rebib` can be installed from CRAN and the development version from `https://github.com/Abhi-1U/rebib`. The website for the package, `https://abhi-1u.github.io/rebib`, has vignettes documenting usage.

5 The process of converting all the legacy articles

The R Journal has been operating since 2009 and has published 682 articles through 2022 in addition to numerous editorials and news items. These LaTeX articles are the main focus of the conversions made with **texor** and **rebib**.

5.1 Batch processing

The code is designed to process all the articles in an issue at a time, by operating on all the files in the directory housing the issue. Each processing function, from pre-processing the text in the LaTeX file to post-processing accepts a directory name as the argument. Generally, if a problem is encountered with one article, the processing will skip to the next. The vignette Introduction to texor provides more details.

5.2 Known problems requiring manual fixes

The problems encountered with some articles are many and varied, and include:

- The table format is often not as neat in the HTML as in the LaTeX. It can be made close to the original with extra effort in the R Markdown description.
- Some of the figure files are missing - so the figure may need to be extracted from the pdf file and included with `knitr::include_graphics()`.

- Figures and equations may need some resizing. Sometimes the figure or equation might extend outside the text region in the HTML, which can be fixed by specifying the preferred size, or breaks in the equation lines.
- Alignment in multi-line equations may need adjusting.
- Alignment of columns in a table may not be the same as the original, for example, the text might be center-aligned, instead of the original left-aligned.
- Specialty LaTeX definitions included by an author for their article may not be recognized by Pandoc, leaving the original command as is, in the article.

5.3 Plan for release

The HTML versions of all successfully converted legacy articles will be on the R Journal website late in 2023. The final step in the integration requires the R Markdown file to be knitted in place to capture the metadata for each article.

5.4 Reporting problems

As a consequence of this project, the journal website has been furnished with an additional link titled "CONTRIBUTE". Visiting this link will give you details on how to report a problem with an article that you are reading, or even to make a suggestion on improving operations.

6 Summary

The original motivation for the **texor** and **rebib** packages was to convert legacy LaTeX articles into HTML format for accessibility. Thus it only creates an `.Rmd` (R Markdown) file which is used to produce only an HTML version. The original PDF remains as published. The goal is to use the packages for future articles where the authors have chosen to submit only in the legacy LaTeX template.

However, the software can also be used by authors themselves to convert to dynamic documents, by providing an initial R Markdown version of their LaTeX, that with some modification, such as including their R code for computations directly in the document, will produce both PDF and HTML versions of their submitted paper.

Adopting the new R Markdown template also provides better standards for R Journal articles. It encourages authors to be more conscious that their work is reproducible. There is more standardization of file names and file structure, and this helps the editors and reviewers check papers. This in turn makes it easier to build the issues, eliminating some common errors that authors make that cause compilation failure.

There are many benefits of providing good standards, though, and an important one is that this can have positive spillover effects for the R community. Authors can become more aware of good practice. For example, one of the useful elements when an article is in HTML format is that figures can be made richer with an `alt-text` (alternative text) element. In an R Markdown code chunk generating a figure, this can be set using `fig.alt`. Alt-text provides a textual alternative to non-text content in HTML documents, and serves various purposes: screen readers can deliver the content to print-disabled people, it is displayed in place of the image if it fails to load, and can serve to assist search engines. It is a work in progress: to encourage adoption by authors of new article submissions, and it would be useful to edit into the converted legacy articles at a later date.

There are many other journals that have a similar backlog of legacy articles. A current and ongoing project is being conducted by arXiv Accessibility Project (2023) to convert their collection of articles from LaTeX (shared as PDF) to HTML, utilizing LaTeXML (Miller and Ginev, 2023). The work conducted for the R Journal might be of interest to similarly small journals that operate without paid staff. Converting legacy articles provides accessibility to a wealth of educational material.

After the change was made to use R Markdown, a new development for the open-source scientific and technical publishing community has emerged: quarto (Allaire et al., 2024). Although R Markdown would suggest a focus on R, it was always possible to include code chunks written in other languages. But Quarto makes this cleaner, and thus more appealing for non-R developers. It also provides a cleaner typesetting. At some point, the R Journal will likely shift to a Quarto template, which is reasonably straightforward, but for the present R Markdown is a suitable dynamic document delivery system for the R Journal.

Acknowledgments

The authors wish to thank the Google Summer of Code program for financially supporting Abhishek Ulayil's work on this project, and the R Project organization for their support. Heather Turner was supported by the Engineering and Physical Sciences Research Council [EPSRC EP/V052128/1] during this project. Also in the initial stages of the development, Zola Batsaikhan carefully checked the conversion of articles, listing and fixing the problems after the conversion. This helped to refine the software to catch these problems and fix in the automatic conversion.

Supplementary materials

The supplementary materials has example folders containing LaTeX documents that allow the reader to see how different common patterns in the legacy documents are handled with the conversion. These include:

- `code-env`: Explains how different code environments defined by the R Journal style are handled, and additional details such as code in figure environments, and code in table environments.
- `math-env`: Examples of inline math, display math, and how equation numbering is handled by a Lua filter to convert from LaTeX labeling to R Markdown labeling.
- `figure-env`: Explains how the variety of figure definitions are handled in the conversion, including different image formats, numbering, captions, labeling, multiple images, and `tikz` images with an example adapted from (Cassidy, 2013).
- `table-env`: Examples of how a variety of table types are converted, including multicolumn, complex and wide tables.
- `lua-filters`: Overview and lots of small examples of Lua filters to handle the custom output needed for the R Markdown format.
- `metadata`: This has a collection of additional format handling including extracting metadata like author names and affiliations, article identifiers used in the review process, and handling citations, footnotes and links.
- `bibliography`: The bibliography was handled differently over the years of the journal, and this details how to use the `rebib` functionality to handle `bbl` files, embedded `bbl`, to convert into the standard `.bib` format.

In each of these folders there is a `RJwrapper.tex`, and `.tex` file, with the extra template files `RJournal.sty` and `Rlogo-5.png` and `.bib` files. These match the legacy template file structure, from which the `RJwrapper.pdf` file is created. To test the conversion for each of these examples, set the path directory to one of the folders and use the `latex_to_web()` function as follows:

```
article_dir <- "path-to-this supplementary folder"
texor::latex_to_web(article_dir)
```

This will create an `.Rmd` and `.html` files in the same directory, that demonstrate the converted R Markdown version and the HTML output format.

You'll need to ensure that you have the latest versions of **texor** and **rebib**, and Pandoc (at least later than version 3.0.0).

Source materials

The **texor** and **rebib** source code and materials to reproduce this paper are available at:

- **texor**: Version 1.3.0 https://abhi-1u.github.io/texor
- **rebib**: Version 0.3.2 https://abhi-1u.github.io/rebib/
- This paper: https://github.com/Abhi-1U/texor-rjarticle
- More details on **rjtools** are at https://rjournal.github.io/rjtools/

References

J. Allaire, C. Teague, C. Scheidegger, Y. Xie, C. Dervieux, and G. Woodhull. Quarto, Nov. 2024. URL https://github.com/quarto-dev/quarto-cli. [p40]

arXiv Accessibility Project. HTML as an accessible format for papers, 2023. URL https://info.arxiv.org/about/accessible_HTML.html. [p39]

T. Boutaris, C. Zauchner, and D. Jomar. *tableHTML: A Tool to Create HTML Tables*, 2023. URL https://CRAN.R-project.org/package=tableHTML. R package version 2.1.2. [p35]

S. Byrne-Haber, D. Fazio, C. LaPierre, and J. Sajka. W3C accessibility maturity model, 2023. URL https://www.w3.org/TR/maturity-model/. [p30]

J. Cassidy. *LaTeX Graphics using TikZ: A Tutorial for Beginners (Part 3)—Creating Flowcharts*. Overleaf tutorials, 2013. URL https://www.overleaf.com/learn/latex/. [p40]

C. Dervieux, J. Allaire, R. Iannone, A. P. Hill, and Y. Xie. *distill: 'R Markdown' Format for Scientific and Technical Writing*, 2022. URL https://CRAN.R-project.org/package=distill. R package version 1.5. [p31]

A. Feder. BibTeX: Reference management software. https://www.bibtex.org, 2006. [p38]

M. Gordon, S. Gragg, and P. Konings. *htmlTable: Advanced Tables for Markdown/HTML*, 2022. URL https://CRAN.R-project.org/package=htmlTable. R package version 2.4.1. [p35]

J. Gruber. Markdown. https://daringfireball.net/projects/markdown/, 2002. [p31]

R. Iannone, J. Cheng, B. Schloerke, E. Hughes, A. Lauer, and J. Seo. *gt: Easily Create Presentation-Ready Display Tables*, 2023. URL https://CRAN.R-project.org/package=gt. R package version 0.9.0. [p35]

R. Ierusalimschy, L. H. de Figueiredo, and W. C. Filho. Lua—an extensible extension language. *Software: Practice and Experience*, 26(6):635–652, 1996. [p33]

J. MacFarlane, A. Krewinkel, and J. Rosenthal. Pandoc. https://github.com/jgm/pandoc, 2023. [p31]

B. Miller and D. Ginev. LaTeXML: A LaTeX to XML/HTML/MathML converter, 2023. URL https://math.nist.gov/~BMiller/LaTeXML/. [p39]

D. Murdoch. *tables: Formula-Driven Table Generation*, 2023. URL https://CRAN.R-project.org/package=tables. R package version 0.9.17. [p35]

O. M. Ritchie and K. Thompson. The UNIX time-sharing system. *The Bell System Technical Journal*, 57(6):1905–1929, 1978. doi: 10.1002/j.1538-7305.1978.tb02136.x. [p31]

The Daisy Consortium. Creating the best way to read and publish, 2023. URL https://daisy.org. [p30]

The LaTeX Project. LaTeX – a document preparation system. https://www.latex-project.org, 2023. [p30]

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. [p30]

Y. Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*. Chapman and Hall/CRC, Boca Raton, Florida, 2023. URL https://bookdown.org/yihui/bookdown/markdown-extensions-by-bookdown.html. [p35]

Y. Xie, J. Allaire, and G. Grolemund. *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida, 2018. ISBN 9781138359338. URL https://bookdown.org/yihui/rmarkdown. [p30]

Y. Xie, J. Cheng, and X. Tan. *DT: A Wrapper of the JavaScript Library 'DataTables'*, 2023. URL https://CRAN.R-project.org/package=DT. R package version 0.28. [p35]

H. Zhu. *kableExtra: Construct Complex Table with 'kable' and Pipe Syntax*, 2021. URL https://CRAN.R-project.org/package=kableExtra. R package version 1.3.4. [p35]

Abhishek Ulayil
Institute of Actuaries of India (student)
Mumbai, India
ORCiD: 0009-0000-6935-8690
abhishek.ulayil.m@gmail.com

Dianne Cook
Monash University
Melbourne, Australia
ORCiD: 0000-0002-3813-7155
dicook@monash.edu

Heather Turner
University of Warwick
Coventry, United Kingdom
ORCiD: 0000-0002-1256-3375
ht@heatherturner.net

Mitchell O'Hara-Wild
Monash University
Melbourne, Australia
ORCiD: 0000-0001-6729-7695
mail@mitchelloharawild.com

Christophe Dervieux
Posit PBC
Paris, France
ORCiD: 0000-0003-4474-2498
christophe.dervieux@gmail.com

Feature-Based Time-Series Analysis in R using the Theft Ecosystem

by Trent Henderson and Ben D. Fulcher

Abstract Time series are measured and analyzed across the sciences. One method of quantifying the structure of time series is by calculating a set of summary statistics or 'features', and then representing a time series in terms of its properties as a feature vector. The resulting feature space is interpretable and informative, and enables conventional statistical learning approaches, including clustering, regression, and classification, to be applied to time-series datasets. Many open-source software packages for computing sets of time-series features exist across multiple programming languages, including 'catch22' (22 features: Matlab, R, Python, Julia), 'feasts' (43 features: R), 'tsfeatures' (62 features: R), 'Kats' (40 features: Python), 'tsfresh' (783 features: Python), and 'TSFEL' (156 features: Python). However, there are several issues: (i) a singular access point to these packages is not currently available; (ii) to access all feature sets, users must be fluent in multiple languages; and (iii) these feature-extraction packages lack extensive accompanying methodological pipelines for performing feature-based time-series analysis, such as applications to time-series classification. Here we introduce a solution to these issues in the form of two complementary statistical software packages for R called 'theft': Tools for Handling Extraction of Features from Time series and 'theftdlc': theft 'downloadable content'. 'theft' is a unified and extendable framework for computing features from the six open-source time-series feature sets listed above as well as custom user-specified features. 'theftdlc' is an extension package to 'theft' which includes a suite of functions for processing and interpreting the performance of extracted features, including extensive data-visualization templates, low-dimensional projections, and time-series classification. With an increasing volume and complexity of large time-series datasets in the sciences and industry, 'theft' and 'theftdlc' provide a standardized framework for comprehensively quantifying and interpreting informative structure in time series.

1 Introduction

Taking repeated measurements of some quantity through time, forming a time series, is common across the sciences and industry. The types of time series commonly analyzed are diverse, ranging from time-varying signals of an electroencephalogram (West et al., 1999), CO_2 concentration in the atmosphere (Kodra et al., 2011), light-curves from distant stars (Barbara et al., 2022), and the daily number of clicks on a webpage (Kao et al., 2021). We can ask many different questions about such data, for example: (i) "can we distinguish the dynamics of brain disorders from neurotypical brain function?"; (ii) "can we classify different geospatial regions based on their temporal CO_2 concentration?"; or (iii) "can we classify new stars based on their light curves?". One approach to answering such questions is to capture properties of each time series and use that information to train a classification algorithm. This can be achieved by extracting from each time series a set of interpretable summary statistics or 'features'. Using this procedure, a collection of univariate time series can be represented as a time series $\times$ feature matrix which can be used as the basis for a range of conventional statistical learning procedures (Fulcher and Jones, 2017; Fulcher, 2018).

The range of time-series analysis methods that can be used to define time-series features is vast, including properties of the distribution, autocorrelation function, stationarity, entropy, methods from the physics nonlinear time-series analysis literature (Fulcher et al., 2013). Because features are real-valued scalar outputs of a mathematical operation, and are often tightly linked to underlying theory (e.g., Fourier analysis or information theory), they can yield interpretable understanding of patterns in time series and the processes that produce them—information that can guide further investigation. The first work to organize these methods from across the interdisciplinary literature encoded thousands of diverse

time-series analysis methods as features and compared their behavior on a wide range of time series (Fulcher et al., 2013). The resulting interdisciplinary library of thousands of time-series features has enabled new ways of doing time-series analysis, including the ability to discover high-performing methods for a given problem in a systematic, data-driven way through large-scale comparison (overcoming the subjective and time-consuming task of selecting methods manually) (Fulcher and Jones, 2014). This approach has been termed 'highly comparative time-series analysis' and has been implemented in the Matlab software `hctsa`, which computes > 7700 time-series features (Fulcher and Jones, 2017). The approach of automated discovery provided by `hctsa` has been applied successfully to many scientific problems, such as classifying zebra finch motifs across different social contexts (Paul et al., 2021), classifying cord pH from fetal heart-rate dynamics (Fulcher et al., 2012), and classifying changes in cortical dynamics from manipulating the firing of excitatory and inhibitory neurons (Markicevic et al., 2020). While `hctsa` is comprehensive in its coverage of time-series analysis methods, calculating all of its features on a given dataset is computationally expensive and it requires access to the proprietary Matlab software, limiting its broader use.

The past decade has seen the development of multiple software libraries that implement different sets of time-series features across a range of open-source programming languages. Here, we focus on the following six libraries:

- `catch22` (C, Matlab, R, Python, Julia) computes a representative subset of 22 features from `hctsa` (Lubba et al., 2019). The > 7700 features in `hctsa` were applied to 93 time-series classification tasks to retain the smallest number of features that maintained high performance on these tasks while also being minimally redundant with each other, yielding the `catch22` set. `catch22` was coded in C for computational efficiency, with wrappers for Matlab, and packages for: R, as **Rcatch22** (Henderson, 2021); Julia, as `Catch22.jl` (Harris, 2021); and Python, as `pycatch22`. `catch22` is also commonly extended to include mean and variance to form `"catch24"` in order to achieve competitive performance on tasks where the dataset has not been standardized (Henderson et al., 2023).

- **tsfeatures** (R) is the most prominent package for computing time-series features in R (Hyndman et al., 2020). The 62 features in **tsfeatures** include techniques commonly used by econometricians and forecasters, such as crossing points, seasonal and trend decomposition using Loess (Cleveland et al., 1990), autoregressive conditional heteroscedasticity (ARCH) models, unit-root tests, and sliding windows. **tsfeatures** also includes sixteen features from `hctsa` that were previously used to organize tens of thousands of time series in the *CompEngine* time-series database (Fulcher et al., 2020).

- **feasts** (R) shares a subset of the same features as **tsfeatures**, computing a total of 43 features (O'Hara-Wild et al., 2021). However, the scope of **feasts** as a software package is larger: it is a vehicle to incorporate time-series features into the software ecosystem known as the **tidyverts**[1]—a collection of packages for time series that follow tidy data principles (Wickham, 2014). This ensures alignment with the broader and popular **tidyverse** collection of packages for data wrangling, summarization, and statistical graphics (Wickham et al., 2019). **feasts** also includes functions for producing graphics, but these are largely focused on exploring quantities of interest in econometrics, such as autocorrelation, seasonality, and Seasonal and Trend decomposition using Loess (STL).

- `tsfresh` (Python) includes 783 features that measure properties of the autocorrelation function, entropy, quantiles, fast Fourier transforms, and distributional characteristics (Christ et al., 2018). `tsfresh` also includes a built-in feature filtering procedure, FeatuRe Extraction based on Scalable Hypothesis tests (FRESH), that uses a hypothesis-testing process to control the percentage of irrelevant extracted features (Christ et al., 2017). `tsfresh` has been used widely to solve time-series problems, such as anomaly

[1] https://tidyverts.org

detection in Internet-of-Things streaming data (Yang et al., 2021) and sensor-fault classification (Liu et al., 2020). `tsfresh` is also commonly accessed through the `FreshPRINCE` functionality within the popular `sktime` Python library (Löning et al., 2019).

- `TSFEL` (Python) contains 156 features that measure properties associated with distributional characteristics, the autocorrelation function, spectral quantities, and wavelets (Barandas et al., 2020). `TSFEL` was initially designed to support feature extraction of inertial data—such as data produced by human wearables—for the purpose of activity detection and rehabilitation.
- `Kats` (Python), developed by Facebook Research, contains a broad range of time-series functionality, including operations for forecasting, outlier and property detection, and feature calculation (Jiang et al., 2022). The feature-calculation module of `Kats` is called **tsfeatures** and includes 40 features (30 of which mirror R's **tsfeatures** package). `Kats` includes features associated with crossing points, STL decomposition, sliding windows, autocorrelation and partial autocorrelation, and Holt–Winters methods for detecting linear trends.

The six feature sets vary over several orders of magnitude in their computation time, and exhibit large differences in both within-set feature redundancy—how correlated features are within a given set—and between-set feature redundancy—how correlated, on average, features are between different pairwise comparisons of sets (Henderson and Fulcher, 2021). While each set contains a range of features that could be used to tackle time-series analysis problems, there are currently no guidelines for selecting an appropriate feature set for a given problem, nor methods for combining the different strengths of all sets. Performance on a given time-series analysis task depends on the choice of the features that are used to represent the time series, highlighting the importance of being able to easily compute many different features from across different feature sets. Furthermore, following feature extraction, there exists no set of visualization and analysis templates for common feature-based problem classes, such as feature-based time-series classification—like the tools provided in `hctsa` (Fulcher and Jones, 2017). Here we present a solution for these challenges in the form of two connected open-source packages for R called **theft**: Tools for Handling Extraction of Features from Time series (which unifies the six disparate feature sets and provides a consistent interface for general feature extraction) (Henderson, 2025b); and **theftdlc**: theft downloadable content (which handles the subsequent processing, analysis, and visualization of time-series features) (Henderson, 2025c). Together, we refer to these packages as the **theft** 'ecosystem'.

2 The theft ecosystem for R

theft unifies the six free and open-source feature sets described in Section 1, thus overcoming barriers in using diverse feature sets developed in different software environments and the differences in their syntax and input-output structures. The package also enables users to manually specify the functions for any number of features they wish to extract on top of the six pre-existing sets. **theftdlc** builds upon the foundation of **theft** by providing an extensive analytical pipeline as well as statistical data visualization templates for understanding feature behavior and performance. To our knowledge, such pipelines and templates do not currently exist in the free and open-source setting, making **theftdlc** a useful tool for both computing and understanding features. While there is some software support for computing features in a consistent setting (such as in `tsflex` (Van Der Donckt et al., 2022), which also provides sliding window extraction capability), such software is limited to only specifying the functional form of individual time-series features rather than automatically accessing features contained in existing feature sets. We partition the analytical capabilities of **theft** and **theftdlc** into two separate packages for two reasons: (i) it reduces dependencies if users wish only to extract features and conduct analysis themselves; and (ii) having a separate analysis package means additional functionality can be continuously added without exhausting R package dependency limits.

An overview of the broad functionality of the **theft** ecosystem is presented in Figure 1. The workflow begins in **theft** with automated installation of Python libraries (if the three Python-based feature sets are required) through the function `install_python_pkgs` which takes the name of the virtual environment to create (`venv`) and the path to the Python interpreter to use (`python`) as arguments (Fig.1A). The Python environment containing the installed software is then instantiated within the R session using `init_theft` (Fig.1B). Time-series data (Fig.1C) is loaded into the environment and converted to a `tsibble::tbl_ts` data structure if required (Fig.1D) (Wang et al., 2020). Time-series features are then extracted in **theft** using the desired pre-existing sets or user-supplied features (Fig.1E). The workflow transitions to **theftdlc** where the user can pass the extracted features into a range of statistical and visualization functions to derive interpretable understanding of the patterns in their dataset (Fig.1F–M). A variety of plot types are readily available, including heatmaps of the time-series $\times$ feature matrix (Fig.1F) and feature $\times$ feature matrix (Fig.1G), and violin plots of feature distributions (Fig.1H). Basic feature selection functionality is available through the `shrink` function (Fig.1I) which implements penalized maximum likelihood generalized linear models using a backend to the R package **glmnet**. Low-dimensional projection functionality is provided by the `project` function (Fig.1J). Time-series classification operations are accessible via the `classify` function (Fig.1K). Distributional summaries of time-series feature and time-series classification values are available through the `interval` function (Fig.1L). Finally, basic cluster analysis is possible through the `cluster` function (Fig.1M). Importantly, **theft** and **theftdlc** use R's `S3` object-oriented programming system, meaning classes and their methods are defined to ensure easy usage with R generic functions, such as `plot`. Classes are defined for feature-calculation objects (Fig.1E; `feature_calculations`), low dimensional projection objects (Fig.1I; `feature_projection`), interval calculation objects (Fig.1K; `interval_calculations`), and cluster objects (Fig.1L; `feature_clusters`). The individual functions of both **theft** and **theftdlc** are discussed in detail in the following sections.

In this paper, we demonstrate how the **theft** ecosystem can be used to tackle a time-series classification problem, using the Bonn University electroencephalogram (EEG) dataset as a case study (Andrzejak et al., 2001). The dataset contains 500 time series, each of length $T = 4097$, with 100 time series each from five labeled classes: (i) awake with eyes open (labeled `eyesOpen`); (ii) awake with eyes closed (`eyesClosed`); (iii) epileptogenic zone (`epileptogenic`); (iv) hippocampal formation of the opposite hemisphere of the brain (`hippocampus`); and (v) seizure activity (`seizure`). Note that classes (i) and (ii) are from healthy volunteers, while classes (iii), (iv), and (v) are from a presurgical diagnosis archive. The time series are comprised of EEG segments 23.6 seconds in duration that were cut out of continuous multichannel recordings, converted from analog to digital using 12-bit conversion, and then written onto a computer at a sampling rate of 173.614Hz. Further trimming of start and end discontinuities from the original 4396 samples was then performed, resulting in a final time series of length T = 4097 samples. This dataset was chosen as a demonstrative example because it has been widely studied as a time-series classification problem, and prior studies have focused on properties of the dynamics that accurately distinguish the classes—which is well-suited to the feature-based approach. For example, prior analysis (using `hctsa`) revealed that seizure recordings are characterized most notably by higher variance, as well as lower entropy, lower long-range scaling exponents, and many other differences (Fulcher et al., 2013). We can easily read the data file in from its zipper format online and convert to a `tsibble` ready for use:

2.1 System requirements

In order to access the features from `tsfresh`, `TSFEL`, and `Kats` in **theft**, Python >=3.10 is required (Python 3.10 is recommended). To use all other functionality across both **theft** and **theftdlc**, only R >=3.5.0 is required.

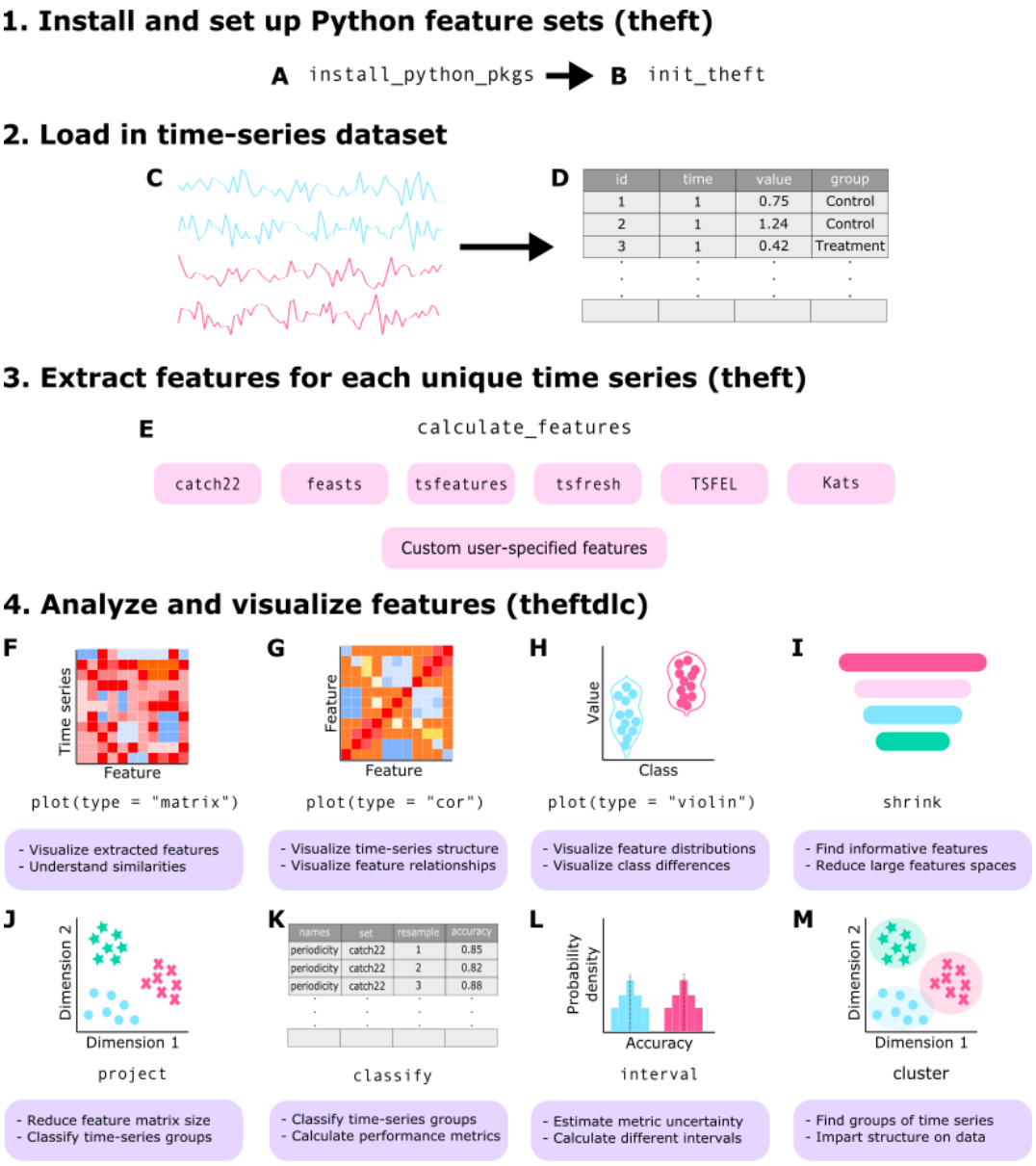


Figure 1: theft and theftdlc together implement a workflow for extracting features from univariate time series and processing and analyzing the results. First, the user can install the relevant Python libraries (A) within R and point it to the correctly installed versions (B). Next, a time-series dataset (C) is converted into a tsibble (D) with key variables, a time index variable, and a measured variable. One or more feature sets or custom user-supplied features are then computed on the dataset (E). A range of statistical analysis and data visualization functionality is available in theftdlc on the resulting feature data, including: (F) normalized time series x feature matrix visualization; (G) normalized feature x feature correlation matrix visualization; (H) violin plots of feature distributions (including by group/class where applicable); (I) basic feature selection using penalized maximum likelihood generalized linear models; (J) low-dimensional projections of the feature space; (K) time-series classification procedures (a common application of feature-based time-series analysis); (L) evaluating the uncertainty intervals of the resulting performance metrics through a range of distributional summary methods; and uncovering hidden structure in time-series data through cluster analysis in the feature space.

2.2 Installing Python libraries

Prior to calculating features, the requisite Python feature sets need to be installed. The `install_python_pkgs` function in **theft** handles this operation entirely within R—all that needs to be supplied is `venv` (a string specifying the name of the new virtual environment to create) and `python` (a string specifying the filepath to the Python interpreter to use). Note that the filepath to the Python interpreter will differ depending on the user's operating system and will require correct specification. For example, on Windows it might be a path similar to `"C:/Users/YourName/AppData/Local/Programs/Python/Python310/python.exe"`, whereas on Linux or MacOS (which was used for the present work) it could be a path similar to `"/usr/local/bin/python3.10"` or `"/usr/bin/python3"`.

An example call which installs the Python libraries into a new virtual environment and initiates the virtual environment within R is shown in the code below. Note that `init_theft` only needs to be run once per session.

```
install_python_pkgs(venv = "theft-eco-py", python = "/usr/local/bin/python3.10")
init_theft("theft-eco-py")
```

2.3 Extracting features

In feature-based time-series analysis, each univariate time series in a dataset is represented as a feature vector, such that the dataset can be represented as a time series $\times$ feature data matrix. Any single feature set, or combination of multiple feature sets, can be computed for a given time-series dataset with the **theft** function `calculate_features`. `calculate_features` takes a `tbl_ts` as input, using the data structure defined by the **tsibble** package for R (Wang et al., 2020). This ensures consistency with the broader `tidyverts` collection of R packages for conducting time-series analysis. A `tbl_ts` is a temporal data structure which is defined by a `key` which identifies each unique time series and an `index` which identifies the time indices. Other columns are treated as measured variables. Since `theft` is a univariate tool, `calculate_features` only accepts inputs that have one measured variable. Since much of the functionality in `theftdlc` is associated with time-series classification problems, if multiple keys are defined, the first is treated as an "ID" variable, while the second is treated as the "grouping" variable for classification.

Users can control various aspects of the feature extraction process through modification of optional arguments. The `catch22` feature set can be expanded to form 'catch24' with included mean and standard deviation by setting `catch24 = TRUE`. Features from the `"compengine"` subset of **tsfeatures** can be calculated by setting `use_compengine = TRUE` which substantially increases computation time for the addition of 16 features. The in-built algorithm in `tsfresh` for selecting relevant features can be used for that set by specifying `tsfresh_cleanup = TRUE` (Christ et al., 2017). In addition, it is a common practice in feature-based time-series analysis to quantify the relative performance of features. For situations where differences in mean and variance are not of interest, z-scoring can be used to standardize each time series prior to the calculation of time-series features. The argument `z_score` enables automatic normalization of each time series within `calculate_features` prior to computing features. Further, users may always wish to disable package warnings when computing features. This can be achieved by setting `warn = FALSE`. Last, parallel processing can be engaged by setting the `n_jobs` argument to a value ≥ 2 (`calculate_features` defaults to serial processing). Currently, parallelization has only been implemented for **tsfeatures**, `tsfresh`, and `tsfel`.

The output of `calculate_features` is an S3 object of class `feature_calculations`, which in this example is stored in the R environment as `all_features`. Within this object is a data frame which contains five columns if the dataset is labeled (as in time-series classification), and four otherwise: `id` (unique identifier for each time series), `names` (feature name), `values` (feature value), `feature_set` (feature set name), and `group` (class label, if applicable). This output structure ensures that, regardless of the feature set selected, the resulting object is

always of the same format and can be used with the rest of the **theftdlc** functions without manual data reshaping.

For the Bonn EEG dataset of 500 time series, each of length $T = 4097$ samples, calculating features for all sets in **theft** took $\approx$ 5.7 hours on a 2019 MacBook Pro with an Intel Core i7 2.6 GHz 6-Core CPU. The extensive computation time was largely driven by **tsfeatures**, noting that the other five sets ranged from just seconds (`catch22`) to several minutes. With the sixteen `"compengine"` features enabled, computation time increased to $\approx$ 11.8 hours. Previous work provides a comprehensive discussion of computation speed between the sets and the scalability with time-series length (Henderson and Fulcher, 2021). An example call which extracts features from all six sets for the Bonn EEG dataset is shown in the code below.

```
all_features <- calculate_features(
  data = bonn_eeg,
  feature_set = c("catch22", "feasts", "tsfeatures",
                  "tsfresh", "tsfel", "kats"),
  use_compengine = FALSE, catch24 = TRUE)
```

theft is also set up to enable users to calculate their own custom features. For example, we could specify two new features such as the mean and standard deviation by adding the requisite functions and the names for those features in a list to the additional `features` argument, as demonstrated in the code below. User-supplied functions must take a vector input and return a numeric scalar value to be a valid time-series feature.

```
all_features_msd <- calculate_features(
                        data = bonn_eeg,
                        feature_set = c("catch22", "feasts", "tsfeatures",
                                        "tsfresh", "tsfel", "kats"),
                        features = list("mean" = mean, "sd" = sd),
                        use_compengine = FALSE)
```

In addition, previous work highlighted that it can be helpful to provide a simple benchmark of performance for the more comprehensive feature sets (Henderson et al., 2023). As such, two simple feature sets—"`quantiles`" and "`moments`"—are also available in `calculate_features` to enable the quick computation of a set of quantiles and the first four moments of the distribution to serve as a baseline for more sophisticated feature sets. "`quantiles`" and "`moments`" are both able to be specified directly as values to the `feature_set` argument of `calculate_features`. By default, the `"quantiles"` feature set includes a collection of 100 quantiles from the range 0.01 to 1.

Users who wish to explore the computed results efficiently can also download the pre-computed features and classifiers used in this paper:

```
files <- c("all_features", "mf_results", "feature_classifiers")

for(f in files){
  temp <- tempfile()
 download.file(paste0("https://github.com/hendersontrent/bonn-eeg-data/raw/refs/heads/main/",
                      f, ".Rda"), temp)
  load(temp)
  unlink(temp)
}
```

2.4 Normalizing features

Different features vary over very different ranges; e.g., features that estimate p-values from a hypothesis test vary over the unit interval, whereas a feature that computes the

length of a time series takes (often large) positive integer values. These differences in scale can complicate the visualization of feature behavior and the construction of statistical learning algorithms involving diverse features. To overcome these limitations, a common pre-processing step involves scaling all features. Several of **theftdlc**'s internal functions utilize rescaling functionality—specifically, providing the user the choice of five methods for converting a set of raw feature values, $\mathbf{x}$, to a normalized version, $\mathbf{z}$:

1. z-score (`"zScore"`): $z_i = \frac{x_i - \mu}{\sigma}$,
2. linear scaling to unit interval (`"MinMax"`): $z_i = \frac{x_i - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}$,
3. maximum absolute scaling (`"MaxAbs"`): $z_i = \frac{x_i}{|\max(\mathbf{x})|}$
4. sigmoid (`"Sigmoid"`): $z_i = \left[1 + \exp(-\frac{x_i - \mu}{\sigma})\right]^{-1}$,
5. and outlier-robust sigmoid (`"RobustSigmoid"`): $z_i = \left[1 + \exp\left(-\frac{x_i - \text{median}(\mathbf{x})}{\text{IQR}(\mathbf{x})/1.35}\right)\right]^{-1}$,

where μ is the mean, σ is the standard deviation, and $\text{IQR}(\mathbf{x})$ is the interquartile range of $\mathbf{x}$. All four transformations end with a linear rescaling to the unit interval. The outlier-robust sigmoid transformation, introduced in (Fulcher et al., 2013), can be helpful in normalizing feature-value distributions with large outliers. Feature normalization is implemented in the R package **normaliseR** which is a key dependency for **theftdlc** (Henderson, 2024).

2.5 Visualizing the feature matrix

A hallmark of large-scale feature extraction is the ability to visualize the intricate patterns of how different time-series analysis algorithms behave across a time-series dataset. This can be achieved in **theftdlc** by specifying `type = "matrix"` when calling `plot` on a `feature_calculations` object to produce a heatmap of the time series (rows) $\times$ feature matrix (columns) which organizes the rows and columns to help reveal interesting patterns in the data. The plot of the combination of all six open feature sets for the Bonn EEG dataset is shown in Figure 2. We can see some informative structure in this graphic, including many groups of features with similar behavior on this dataset (i.e., columns with similar patterns), indicating substantial redundancy across the joint set of features (Henderson and Fulcher, 2021). The bottom block of 100 rows, which visually have the most distinctive properties, was found to correspond to time series from the `seizure` class, indicating the ability of this large combination of time-series features to meaningfully structure the dataset.

```
plot(all_features,
     type = "matrix",
     norm_method = "RobustSigmoid",
     clust_method = "average")
```

In matrix plots in **theftdlc**, hierarchical clustering is used to reorder rows and columns so that time series (rows) with similar properties are placed close to each other and features (columns) with similar behavior across the dataset are placed close to each other—where similarity in behavior is quantified using Euclidean distance in both cases (Day and Edelsbrunner, 1984). In Figure 2, we specify the usage of average (i.e., unweighted pair group method with arithmetic mean) agglomeration. Default settings within `plot` enable users to easily generate outputs in a single line of code, but more advanced users may seek to tweak the optional arguments. For example, different linkage algorithms for hierarchical clustering can be controlled by supplying the argument to `clust_method`, which uses average agglomeration as a default, and the different rescaling methods defined earlier can be supplied to the `norm_method` argument, which defaults to `"zScore"`.

2.6 Projecting low-dimensional feature-spaces

Low-dimensional projections are a useful tool for visualizing the structure of high-dimensional datasets in low-dimensional spaces. Here we are interested in representing a

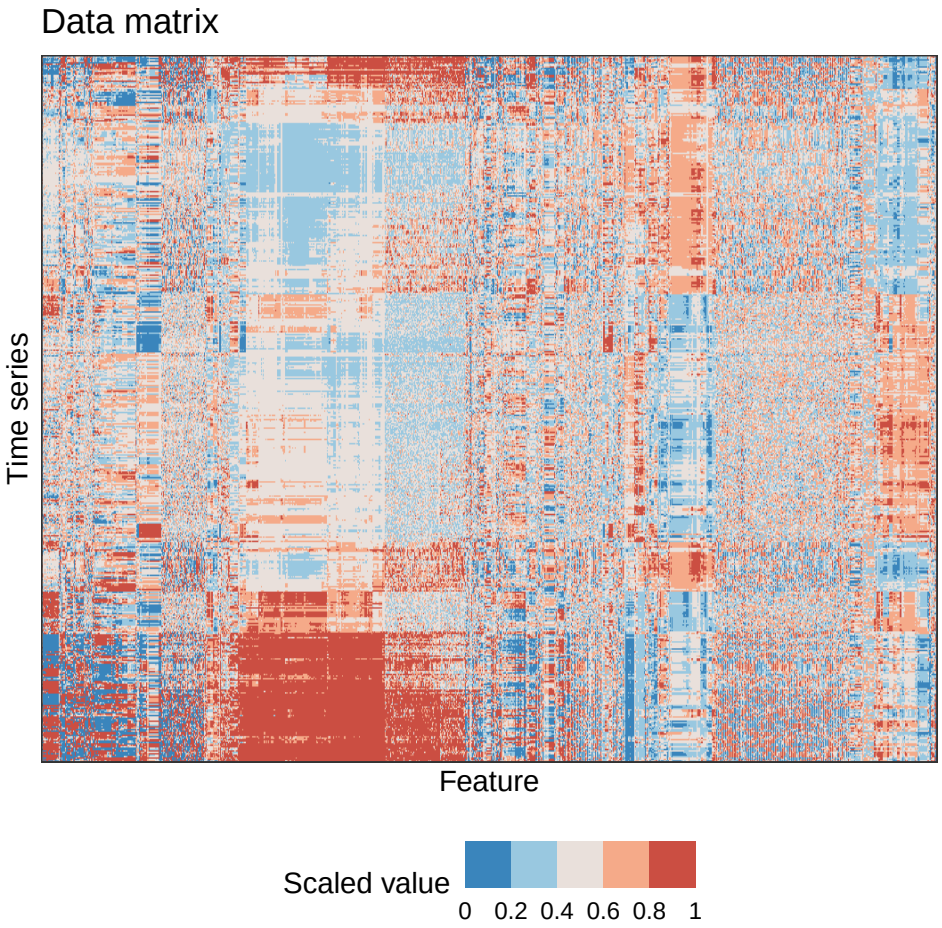


Figure 2: A time series by feature matrix heatmap produced by generating a matrix plot on the feature calculations object. Extracted feature vectors for each time series (500) in the Bonn EEG dataset using all six feature sets in theft (1005 features in total, after filtering out 85 features with NaN values) are represented as a heatmap. Similar features (columns) and time series (rows) are positioned close to each other using (average) hierarchical clustering. Each tile is a normalized value for a given time series and feature.

time-series dataset in a two-dimensional projection of the feature space, which can reveal structure in the dataset, including how different labeled classes are organized. For linear dimensionality reduction techniques, such as principal components analysis (PCA) (Jolliffe, 2002), the results can be visualized in two dimensions as a scatterplot, where the principal component (PC) that explains the most variance in the data is positioned on the horizontal axis and the second PC on the vertical axis, and each time series is represented as a point (colored by its group label in the case of a labeled dataset). When the structure of a dataset in the low-dimensional feature space matches known aspects of the dataset (such as class labels), it suggests that the combination of diverse time-series features can capture relevant dynamical properties that differ between the classes. It can also reveal new types of structure in the dataset, like clear sub-clusters within a labeled class, that can guide new understanding of the dataset. Low-dimensional projections of time-series features have been shown to meaningfully structure time-series datasets: revealing sex and day/night differences in *Drosophila* (Fulcher and Jones, 2017), distinguishing types of stars based on their light curves (Barbara et al., 2022), and categorizing sleep epochs (Decat et al., 2022).

Several algorithms for projecting feature spaces are available in **theftdlc**:

- Principal components analysis (PCA)—`"PCA"`
- *t*-Stochastic Neighbor Embedding (*t*-SNE)—`"tSNE"`
- Classical multidimensional scaling (MDS)—`"ClassicalMDS"`
- Kruskal's non-metric multidimensional scaling—`"KruskalMDS"`
- Sammon's non-linear mapping non-metric multidimensional scaling—`"SammonMDS"`
- Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP)—`"UMAP"`

Time-series datasets can be projected in low-dimensional feature spaces in **theftdlc** using the `project` function. Users can control algorithm hyperparameters by supplying additional arguments to the `...` argument of `project`—the function then passes these arguments to the requisite dimension reduction algorithm internally. The `project` function returns an `S3` object of class `feature_projection`, which can then be passed to `plot` which will automatically draw the appropriate two-dimensional scatterplot. The `feature_projection` class is a list object which contains elements representing the user-supplied feature data frame, the model data, the two-dimensional projected data frame, and the model fit object itself. The plot method for this object contains only one other argument (`show_covariance`) which specifies whether to draw covariance ellipses for each group in the scatterplot (if a grouping variable is detected).

The low-dimensional projection plot for the Bonn EEG dataset (using *t*-SNE and all > 1000 non-`NaN` features across the six feature sets included in **theft**) is shown in Figure 3 with perplexity 15, as produced by the code below. The low-dimensional projection meaningfully structures the labeled classes of the dataset. Specifically, two of the presurgical diagnosis classes—`epileptogenic` (epileptogenic zone) and `hippocampus` (hippocampal formation of the opposite hemisphere of the brain)—appear to exhibit considerable overlap in the projected space, while the two healthy volunteer classes `eyesOpen` (awake state with eyes open) and `eyesClosed` (awake state with eyes closed) occupy space further away from the other classes but closer to each other. The `seizure` class occupies a space largely separate from the other four classes in the projection, consistent with its distinctive periodic dynamics (Fulcher et al., 2013).

```
low_dim_calc <- project(all_features,
                        method = "MinMax",
                        low_dim_method = "tSNE",
                        perplexity = 15)

plot(low_dim_calc)
```

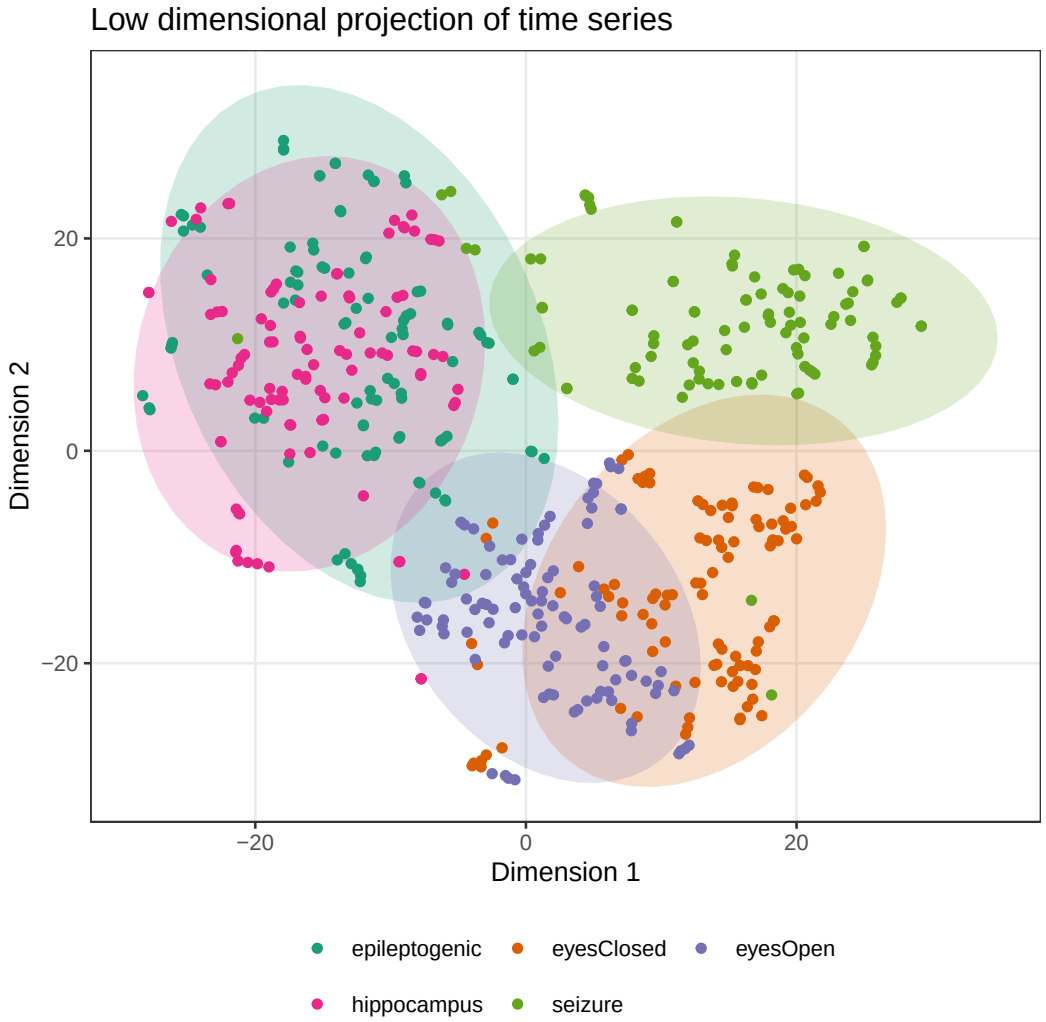


Figure 3: Low-dimensional projection of the Bonn EEG dataset using theft. Using t-SNE with perplexity 15, the high-dimensional feature space of >1000 features is projected into two dimensions. Each point represents a time series which is colored according to its class label. Time series that are located close in this space have similar properties, as measured by the six feature sets in theft.

2.7 Constructing classifiers with multiple features

Combinations of complementary, discriminative features can often be used to construct accurate time-series classifiers (Fulcher and Jones, 2014). Drawing on computed time-series features (that may derive from one or more existing feature sets), **theftdlc** can fit and evaluate classifiers using the `classify` function. This allows users to evaluate the relative performance of each feature set, of the combination of all sets, or any other combination of features. Providing easy access to a range of classification algorithms and accompanying inferential tools (such as null permutation testing to obtain p-values and performance distributions through the resampling-based algorithm) through `classify` allows users to compare sets of features to better understand the most accurate feature sets for a given time-series classification problem. The code presented below provides an example usage for the Bonn EEG dataset with a linear support vector machine (SVM) classifier (which is the default and so does not require explicit specification).

```
mf_results <- classify(
                data = all_features,
                by_set = TRUE,
                train_size = 0.8,
                n_resamples = 100,
                use_null = TRUE)
```

The `classify` function is flexible in that users can supply any function that can be used with R's native `stats::predict` generic. For example, a user could easily use a radial basis function SVM instead:

```
rbfClassifier <- function(formula, data){
  mod <- e1071::svm(formula, data = data, kernel = "radial", scale = FALSE,
                    probability = TRUE)
}

mf_results_rbf <- classify(
                    data = all_features,
                    classifier = rbfClassifier,
                    by_set = TRUE,
                    train_size = 0.8,
                    n_resamples = 100,
                    use_null = TRUE)
```

In the above code, we specified that we want `classify` to fit separate classifiers for each feature set, using a training set size that is 80% of the input data size, with 100 resamples for each feature set as a way to incorporate uncertainty. We also enabled null model fitting for permutation testing. In applications involving small datasets, or when small effects are expected, it is useful to quantify how different the observed classification performance is from a null setting in which data are classified randomly (i.e., could the same results have been obtained by chance?). One method for inferring test statistics is to use permutation testing—a procedure that samples a null process many times to form a distribution against which a value of importance (i.e., the classification accuracy result from a model) can be compared to estimate a p-value (Ojala and Garriga, 2009). In **theft**, permutation testing is implemented for evaluating classification performance in `classify` through the `use_null` argument. When set to `TRUE`, `classify` will compute results for `n_resamples` models where the class labels match the data, but also `n_resamples` models where the class labels are shuffled, thus severing the input-output relationship. In the absence of any data errors, we would expect the mean classification accuracy for the empirical null distribution to approximate chance for the problem. This provides a useful comparison point for the main (non-shuffled) models—if the main models statistically outperform the empirical null models, then the result, quantified by the p-value, is likely not due to chance, thus

indicating that the set of time-series features represent quantities that can meaningfully capture differences between classes.

The resampling procedure begins by partitioning the input data into `n_samples` number of seeded train-test splits for reproducibility, using `train_size` to govern the size of the training set for each split. Importantly, the procedure tracks the class representation in the first resample and preserves these proportions for all subsequent resamples. The feature data for each resample is then normalized as a z-score by computing the mean and standard deviation of each feature in the training set, and using these values to rescale both the training and test sets. This ensures that test data are completely unseen. The procedure then iterates over each resample and fits the classification algorithm for each. By default, `classify` calculates the following accuracy metrics, where C is the number of classes, TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives:

- Accuracy (`"accuracy"`): $\frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C}(TP_i+FP_i+FN_i)}$
- Mean precision (`"precision"`): $\frac{1}{C}\sum_{i=1}^{C}\frac{\mathrm{TP}_i}{\mathrm{TP}_i+\mathrm{FP}_i}$
- Mean recall (`"recall"`): $\frac{1}{C}\sum_{i=1}^{C}\frac{\mathrm{TP}_i}{\mathrm{TP}_i+\mathrm{FN}_i}$
- F1 score (`"f1"`): $2\frac{\mathrm{Precision}\cdot\mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}}$

`classify` returns a list with two elements: (i) the train and test sizes; and (ii) a data frame of classification results. While the raw results are useful, **theftdlc** enables automated analysis of them. The `theftdlc::interval` and `compare_features` functions were designed to enable fast and intuitive comparisons between individual features and entire sets. `theftdlc::interval` produces summaries of classification results with uncertainty using three distinct methods: (i) standard deviation—`"sd"`; (ii) confidence interval based off the t-distribution—`"se"`; and (iii) quantile summary—`"quantile"`.

Users can compute different interval summaries by modifying the additional arguments:

- `metric`—the classification performance metric to calculate intervals for. Can be one of `"accuracy"`, `"precision"`, `"recall"`, or `"f1"`
- `by_set`—whether to compute intervals for each feature set. If `FALSE`, the function will instead calculate intervals for each individual feature
- `type`—whether to calculate a $\pm SD$ interval with `"sd"`, confidence interval based off the t-distribution with `"se"`, or a quantile with `"quantile"`
- `interval`—the width of the interval to calculate. Defaults to 1 if `type = "sd"` to produce a $\pm 1SD$ interval. Defaults to `0.95` if `type = "se"` or `type = "quantile"` for a 95% interval
- `model_type`—whether to calculate intervals for main models with `"main"` or null models with `"null"` if the `use_null` argument of `classify` was `use_null = TRUE`

Below we calculate the mean $\pm 1SD$ for each feature set. For the Bonn EEG dataset, we find that the set of all features (`All features`) produced the highest mean classification accuracy (91.2%) and `catch22` (the smallest feature set) produced the lowest mean accuracy (80.1%). The best performing individual feature set was the largest—`tsfresh`—with a mean classification accuracy of 90.8% which was marginally below the set of all features.

```
set_intervals <- theftdlc::interval(
  mf_results,
  metric = "accuracy",
  by_set = TRUE,
  type = "sd",
  model_type = "main"
  )

set_intervals
```

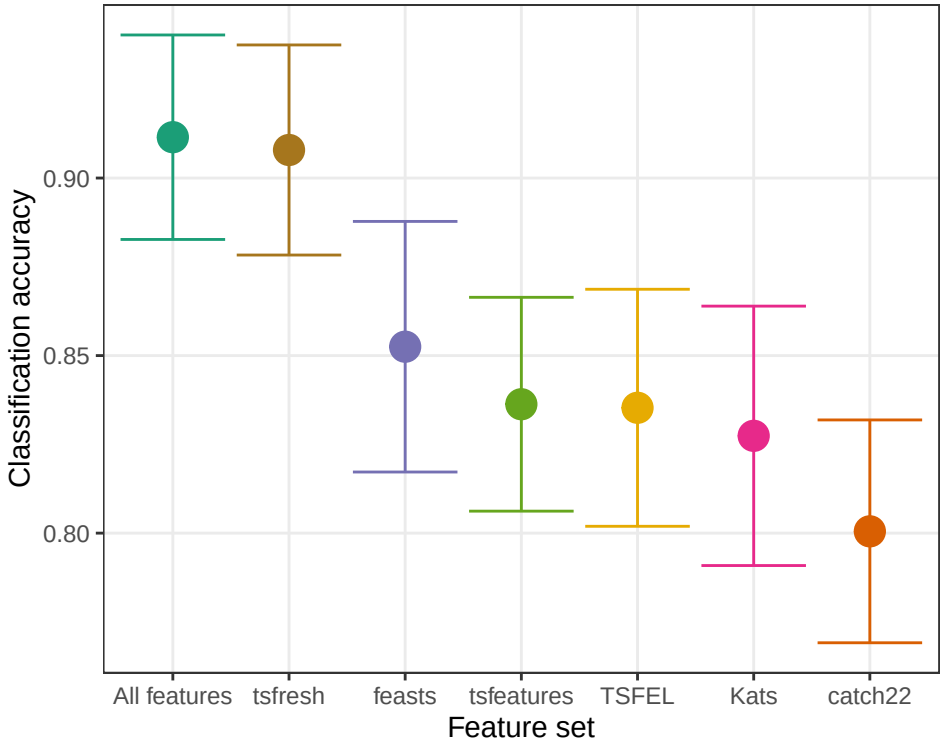


Figure 4: Comparison of mean classification accuracy between feature sets in theft for the five-class Bonn EEG classification task. Classification accuracy using a linear SVM is presented for each of the six feature sets in theft as well as the combination of all their features. The number of features retained for analysis after filtering is displayed in parentheses after the feature set name on the horizontal axis which has been sorted from highest to lowest mean accuracy. Mean classification accuracy across the same 100 resamples is displayed as colored points for each set with one standard deviation error bars.

```
#>    feature_set  .mean    .lower    .upper
#> 1 All features 0.9115 0.8827069 0.9402931
#> 2         Kats 0.8274 0.7908680 0.8639320
#> 3        TSFEL 0.8353 0.8019226 0.8686774
#> 4      catch22 0.8005 0.7691217 0.8318783
#> 5       feasts 0.8525 0.8172126 0.8877874
#> 6   tsfeatures 0.8363 0.8061940 0.8664060
#> 7      tsfresh 0.9079 0.8783138 0.9374862
```

`theftdlc::interval` returns an `S3` object of class `interval_calculations` which is a data frame that can be used with the `plot` generic through **theftdlc**. The resulting plot is presented in Figure 4 where we see the considerable overlap in mean $\pm 1SD$ classification accuracy between the feature sets.

```
plot(set_intervals)
```

While the visual aid is useful, we often want to understand if a given feature set has performed better than we might expect due to chance. Or can we determine if a given feature set has statistically outperformed another (such as the set of all features compared to `tsfresh`), therefore providing guidance on which features to retain for subsequent analysis? How can we estimate if the performance of the smallest and fastest-to-compute set – `catch22` – is meaningfully different from the larger feature sets with close average classification accuracy values (i.e., `Kats`, `TSFEL`, and `tsfeatures`)? The `compare_features` function in **theftdlc** enables pairwise comparisons between either individual features or entire feature sets, or to their own respective empirical null distributions (i.e., the 'chance' distribution). It does so through usage of the resampled t-test which accounts for the correlation between

samples in the calculation of the test statistic (Nadeau and Bengio, 2003). The resampled t-test is implemented through the **correctR** R package (Henderson, 2025a). `compare_features` returns a data frame with columns for the hypothesis that was tested, the test statistic, p-value, adjusted p-value (if specified), and the means of each group for each hypothesis test. `compare_features` can operate on any of the metrics computed in `classify`. The function only takes a small number of arguments in addition to the output of `classify`:

- `metric`—the classification performance metric to calculate intervals for. Can be one of `"accuracy"`, `"precision"`, `"recall"`, or `"f1"`
- `by_set`—whether to compare entire feature sets (`TRUE`) or individual features (`FALSE`)
- `hypothesis`—whether to compare each entire feature set or individual feature to its own empirical null distribution (`"null"`, if permutation testing was used in `classify`) or to each other (`"pairwise"`)
- `p_adj`—method for adjusting p-values for multiple comparisons. Defaults to `"none"` for no adjustments, but can take any valid option received by `stats::p.adjust`

For example, we can statistically compare the performance of each feature set against their empirical null distributions (formed by the distribution of performance on shuffled class label data) using the code presented below. When specifying `hypothesis = "null"`, **theftdlc** automatically applies a one-tailed test, since the hypothesis is that the main models should outperform their null counterparts. We find that, for the full Bonn EEG dataset with 100 time series per class and strong differences between signals, mean classification accuracy for each feature set is far higher than chance level (20%) over 100 resamples. Further, we obtain extremely small p-values close to zero (displayed in the table below as zero due to rounding for visual clarity), providing support for the low probability of obtaining classification accuracy results at least as extreme as what we observed under the null hypothesis. This confirms that time-series features can effectively distinguish between classes in the dataset. Note that we drop some of the columns reported by **theftdlc** here for spatial reasons.

```
compare_features(mf_results,
                 metric = "accuracy",
                 by_set = TRUE,
                 hypothesis = "null",
                 p_adj = "none")
```

hypothesis	set_mean	null_mean	t_statistic	p.value
All features != own null	0.911	0.192	12.496	0
Kats != own null	0.827	0.188	9.701	0
TSFEL != own null	0.835	0.197	10.638	0
catch22 != own null	0.800	0.203	8.699	0
feasts != own null	0.853	0.192	10.003	0
tsfeatures != own null	0.836	0.197	11.157	0
tsfresh != own null	0.908	0.192	13.787	0

We can then compare the feature sets to one another to provide statistical evidence for any differences in the performance ranges visualized in Figure 4 using the code below. For `hypothesis = "pairwise"`, **theftdlc** applies a two-tailed test. We find that the set of all features outperforms all individual sets at $\alpha < 0.05$ except for `tsfresh` ($p = 0.820$), `feasts` ($p = 0.094$), and `tsfeatures` ($p = 0.077$).

```
compare_features(mf_results,
                 metric = "accuracy",
                 by_set = TRUE,
                 hypothesis = "pairwise",
                 p_adj = "none")
```

hypothesis	set_a_mean	set_b_mean	t_statistic	p.value
All features != catch22	0.911	0.800	3.026	0.003
All features != feasts	0.911	0.853	1.713	0.090
All features != Kats	0.911	0.827	2.306	0.023
All features != tsfeatures	0.911	0.836	2.066	0.041
All features != TSFEL	0.911	0.835	2.058	0.042
All features != tsfresh	0.911	0.908	0.198	0.843
catch22 != feasts	0.800	0.853	-1.249	0.215
catch22 != Kats	0.800	0.827	-0.634	0.527
catch22 != tsfeatures	0.800	0.836	-0.871	0.386
catch22 != TSFEL	0.800	0.835	-0.873	0.385
catch22 != tsfresh	0.800	0.908	-2.776	0.007
feasts != Kats	0.853	0.827	0.579	0.564
feasts != tsfeatures	0.853	0.836	0.414	0.680
feasts != TSFEL	0.853	0.835	0.419	0.676
feasts != tsfresh	0.853	0.908	-1.436	0.154
Kats != tsfeatures	0.827	0.836	-0.198	0.843
Kats != TSFEL	0.827	0.835	-0.183	0.855
Kats != tsfresh	0.827	0.908	-2.006	0.048
tsfeatures != TSFEL	0.836	0.835	0.025	0.980
tsfeatures != tsfresh	0.836	0.908	-1.796	0.076
TSFEL != tsfresh	0.835	0.908	-1.913	0.059

2.8 Finding and understanding informative individual features

Fitting models which use multiple features as inputs is often useful for predicting class labels. However, users are also typically interested in understanding patterns in their dataset, such as interpreting the types of time-series analysis methods that best separate different classes, and the relationships between these top-performing features. This can be achieved using mass univariate statistical testing of individual features, quantifying their performance either relative to an empirical null distribution or each other. **theftdlc** implements the ability to identify top-performing features in the `compare_features` function by setting `by_set = FALSE`, with an example usage for the Bonn EEG dataset (using features from all six packages) shown in the code below. We implement parallel processing to speed up computation time.

```
feature_classifiers <- classify(data = all_features,
                                by_set = FALSE,
                                train_size = 0.8,
                                n_resamples = 100,
                                use_null = TRUE)

feature_vs_null <- compare_features(feature_classifiers,
                                    by_set = FALSE,
                                    hypothesis = "null",
                                    n_workers = 6)
```

Straightforward **dplyr** syntax can then be used to identify the top n features (Wickham et al., 2023). We have the choice of either mean classification accuracy or p-values relative to the empirical null to determine informative features. For illustrative purposes, here we have used mean classification accuracy to find the top $n = 40$ features. We show the top 20 features below for spatial reasons. We see that the 17 best-performing individual features achieve $> 50\%$ accuracy (which far exceeds the chance probability for a five-class problem of 20%).

```
top_40 <- feature_vs_null |>
  dplyr::slice_max(feature_mean, n = 40)

top_40 |>
  top_n(feature_mean, n = 20)
```

hypothesis	feature_mean	null_mean	t_statistic	p.value
tsfresh_values__ autocorrelation__lag_6 != own null	0.537	0.138	4.054	0.000
tsfresh_values__ autocorrelation__lag_7 != own null	0.537	0.136	4.244	0.000
tsfresh_values__ autocorrelation__lag_8 != own null	0.527	0.138	4.513	0.000
tsfresh_values__ change_quantiles__f_agg_"mean"__ isabs_True__qh_1.0__ql_0.4 != own null	0.524	0.142	7.373	0.000
tsfresh_values__ change_quantiles__f_agg_"mean"__ isabs_True__qh_1.0__ql_0.2 != own null	0.521	0.146	6.819	0.000
tsfresh_values__ change_quantiles__f_agg_"mean"__ isabs_True__qh_0.8__ql_0.4 != own null	0.520	0.148	7.016	0.000
TSFEL_0_Wavelet energy_25.0Hz != own null	0.515	0.145	7.903	0.000
TSFEL_0_Wavelet standard deviation_25.0Hz != own null	0.515	0.145	7.903	0.000
tsfresh_values__ change_quantiles__f_agg_"mean"__ isabs_True__qh_0.8__ql_0.2 != own null	0.512	0.148	6.362	0.000
tsfresh_values__ change_quantiles__f_agg_"mean"__ isabs_True__qh_0.6__ql_0.2 != own null	0.511	0.141	7.350	0.000
TSFEL_0_Median absolute diff != own null	0.504	0.147	6.645	0.000
TSFEL_0_Mean absolute diff != own null	0.504	0.146	6.298	0.000
TSFEL_0_Sum absolute diff != own null	0.504	0.146	6.298	0.000
tsfresh_values__ absolute_sum_of_changes != own null	0.504	0.146	6.298	0.000
tsfresh_values__ change_quantiles__f_agg_"mean"__ isabs_True__qh_1.0__ql_0.0 != own null	0.504	0.146	6.298	0.000
tsfresh_values__ mean_abs_change != own null	0.504	0.146	6.298	0.000
TSFEL_0_Signal distance != own null	0.501	0.146	6.136	0.000
tsfresh_values__ change_quantiles__f_agg_"mean"__ isabs_True__qh_0.8__ql_0.0 != own null	0.500	0.144	7.167	0.000
tsfresh_values__ autocorrelation__lag_5 != own null	0.500	0.132	3.280	0.001
tsfresh_values__ augmented_dickey_fuller__attr_ "teststat"__autolag_"AIC" != own null	0.497	0.140	5.100	0.000

Understanding what each feature within the table measures can provide insight into the types of features relevant for the classification problem. For example, we see features that measure properties associated with autocorrelation (e.g., `tsfresh_values__autocorrelation_lag_6` which measures the value of the autocorrelation function at lag 6) and signal peaks (e.g., `TSFEL_0_Median absolute diff` which measures the median value of all absolute differences along the signal), among others. However, interpreting this table is challenging as the relationships between the features are unknown—are all the 40 features behaving differently, or are they all highly correlated to each other and essentially proxy metrics for the same underlying time-series property? We can better understand these relationships by visualizing the pairwise feature $\times$ feature correlation matrix.

To achieve this, we can filter the original feature data to only include the top features, assign it to a `feature_calculations` object type, and make use of the `plot` function in **theftdlc** to visualize pairwise absolute correlations between the top performing features. This is presented visually in Figure 5. The plot reveals two main groups of highly correlated ($|\rho| \gtrapprox 0.8$) features: in the bottom left and upper right of the plot. The cluster in the bottom left contains features that capture different types of autocorrelation structure in the time series, including linear autocorrelation coefficients (e.g., `tsfresh_values__autocorrelation_lag_6`) and the variance of means over sliding windows (e.g., `tsfeatures_stability`). The large cluster in the top right (containing features from `tsfresh` and `TSFEL` exclusively) contains features sensitive to variance—including change quantiles (e.g., `tsfresh_values__change_quantiles__f_agg_"mean"__isabs_True__qh_1.0__ql_0.4` which measures the mean of the absolute change of the time series values inside quantiles $0.4 - 1.0$), wavelet variance (e.g., `TSFEL_0_Wavelet standard deviation_25.0Hz` which measures the variance of coefficients from Ricker wavelets with widths $1 - 10$ at the lower quarter of the assumed sampling frequency of 100Hz), and the 'distance traveled' by the signal (e.g., `TSFEL_0_Signal distance` which measures the sum of square root squared differences). While the differences between classes—as identified through the list of top features—in this case were simple (i.e., autocorrelation and variance), other, more complex features may perform the strongest on other problems, or even different pairs of classes within the five-class dataset investigated here. Identifying when simple features perform well is important as it can provide interpretable benchmarks for assessing relative performance gains achieved by more complex and/or less interpretable alternative classifiers (Henderson et al., 2023).

```
feature_matrix_filt <- all_features |>
  dplyr::filter(feature_set %in% top_40$feature_set &
                  names %in% top_40$original_names) |>
  structure(class = c("feature_calculations", "data.frame"))

plot(feature_matrix_filt, type = "cor")
```

Having identified the discriminative features, it can be important to understand how they differ amongst the labeled classes of a dataset. This can be achieved by visualizing the distribution of values for each class for each of the features. In **theftdlc**, a violin plot can be produced in `plot` by setting `type = "violin"`, where each time series is represented as a point organized and colored by its class label. Note that a boxplot alternative (which highlights univariate outliers as points) is also possible through specifying `type = "box"`. Here, for visual clarity, we show violin plots for a selected feature from the variance cluster of features from Figure 5: `0_Signal distance` from `TSFEL` (mean classification accuracy 50.1% over 100 resamples); and a selected feature from the autocorrelation-sensitive cluster of features: `values_autocorrelation_lag_6` from `tsfresh` (mean classification accuracy 53.7% over 100 resamples). The outputs are shown in Figure 6. Consistent with their high classification scores relative to chance (20%), both features are individually informative of class differences. The plot shows that with regard to autocorrelation structure, we see that `eyesClosed` exhibits typically weak to moderate negative coefficient values at lag

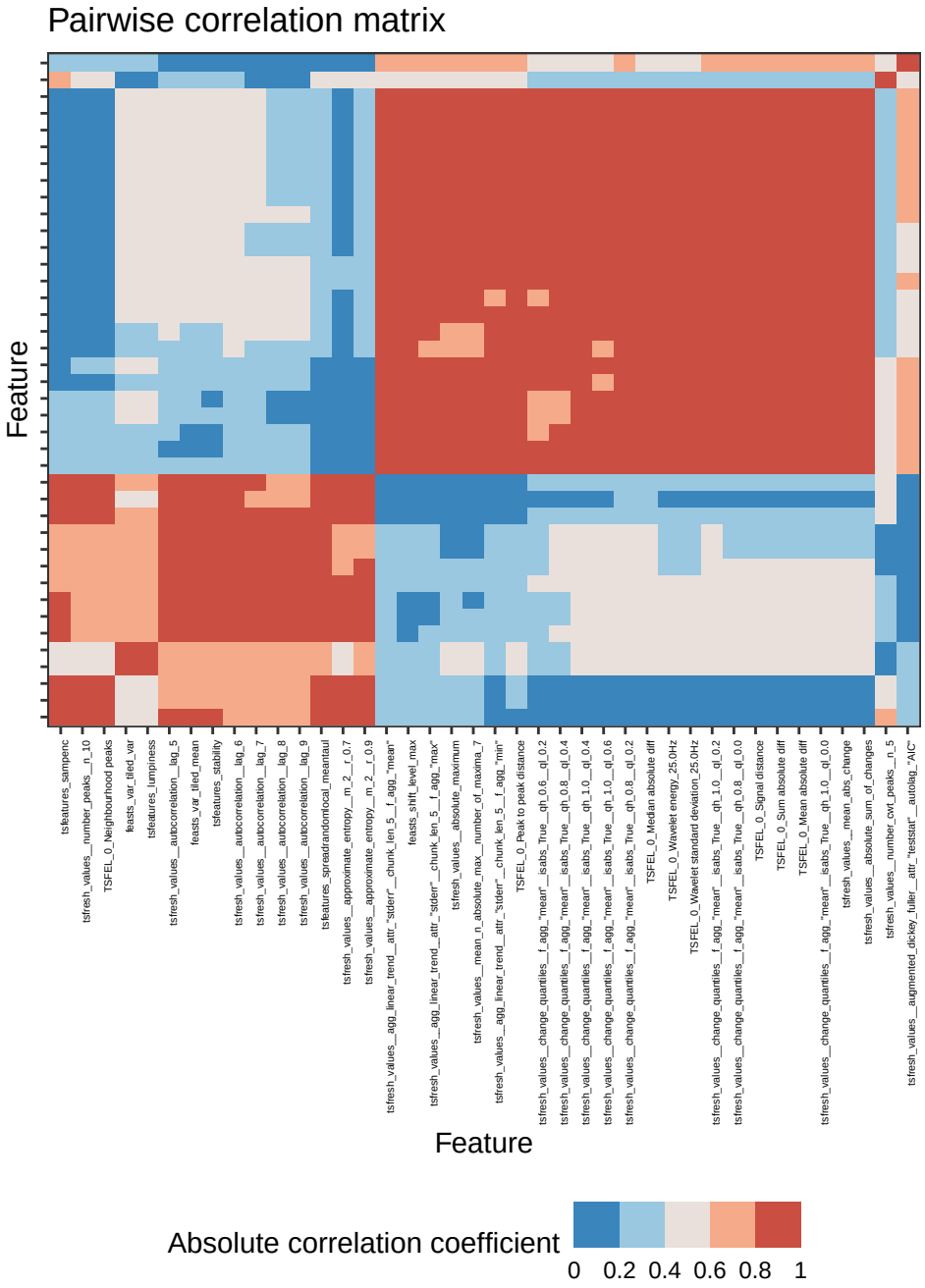


Figure 5: A group of change quantile and difference-associated features and a group of autocorrelation-sensitive features perform the best at distinguishing between the five classes in the Bonn EEG dataset using the absolute Spearman correlation coefficient to capture feature-feature similarity. To aid the identification of similarly performing features, the matrix of correlation coefficients between features were then organized using hierarchical clustering (on Euclidean distances with average linkage) along rows and columns to order the heatmap graphic.

6, while hippocampus and epileptogenic exhibit typically moderate positive coefficients. eyesClosed is characterized by weak to moderate coefficient values largely within the $0 - 0.5$ range, while seizure exhibits a substantially wider distribution of values than the other classes—a distribution which almost spans the entire range of coefficients exhibited by the others. This defining lack of temporal predictability in short-range autocorrelation coefficients for seizure time series is consistent with prior work and is characteristic of the more erratic nature of seizure state brain activity (Fulcher et al., 2013).

The plot also shows that with regard to signal distance (i.e., $\sum_{t=1}^{T-1}\sqrt{1+(x_{t+1}-x_t)}$ where T is the length of the time series and x is the vector of values), all classes except seizure exhibit similar values with subtle differences in the mean and variance of their feature value distributions. This is consistent with the lower classification performance of this feature compared to values_autocorrelation_lag_6. For seizure, we again see a wide distribution of feature values definitive of this class. Practically, since signal distance measures the total distance traveled by the signal between time points, it can be inferred that seizure state brain activity (as measured by an EEG) fluctuates far more than healthy brain activity and measurements from the epileptogenic zone (i.e., the difference between values at any two consecutive time points is, on average, larger), consistent with known dynamics of seizure states (Andrzejak et al., 2001).

Together, these two feature case studies reinforce the interpretative benefits to a feature-based approach to time-series analysis. Features can not only organize time-series data, reveal structure, and predict class membership, but they can also provide insight into the underlying generative properties that distinguish different time series—such as the difference in brain activity between seizure state and regular brain function.

```
plot(feature_matrix_filt,
     type = "violin",
     feature_names = c("values__autocorrelation__lag_6",
                       "0_Signal distance")) +
  theme(strip.text = element_text(size = 6))
```

2.9 Additional functionality

In addition to the functionality demonstrated here, **theft** and **theftdlc** include a collection of other functions, not demonstrated in this article for brevity, including cluster analysis (through the cluster function in **theftdlc**), simple feature selection using penalized maximum likelihood generalized linear models (through the shrink function in **theftdlc**), and the processing of hctsa-formatted Matlab files in **theft**. Readers are encouraged to explore this additional functionality in the detailed vignettes included with the packages.

3 Discussion

Feature-based time-series analysis is a powerful computational tool for tackling statistical learning problems using sequential (typically time-ordered) data. We have introduced the **theft** and **theftdlc** packages for R which implement the extraction, processing, visualization, and statistical analysis of time-series features. The value of time-series features stems from their interpretability and strong connection to theory that can be used to understand the empirical properties of their dynamics. **theft** provides a unified interface to extracting features from six open-source packages—catch22, **feasts**, **tsfeatures**, Kats, tsfresh, and TSFEL—while **theftdlc** provides a comprehensive range of analyses to leverage the combined contributions from all of these packages. For the first time in the free and open-source software setting, the **theft** ecosystem provides a full workflow for conducting feature-based time-series analysis, taking the analyst from feature extraction through to generating interpretable insights about their data. **theftdlc** introduces a set of simply named functions that make analysis of time-series features calculated in **theft** intuitive and streamlined:

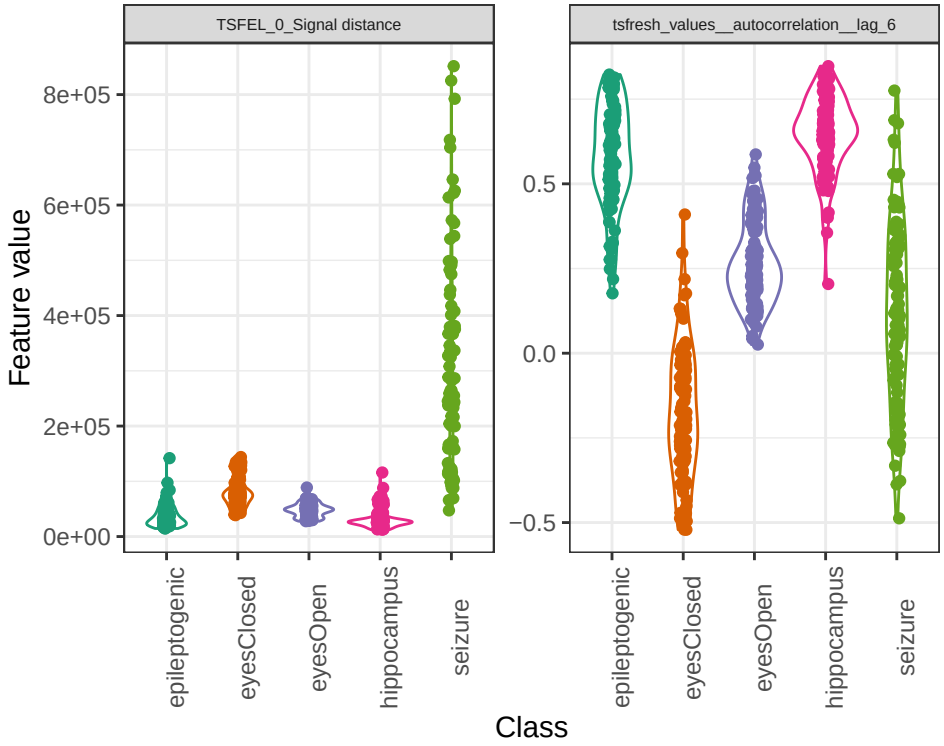


Figure 6: Violin plots (on original feature value scale) of a sample of two of the top 40 features of all six feature sets in theft for classifying Bonn EEG groups. Classes differ in their variance and autocorrelation properties.

`classify`, `project`, `cluster`, `interval`, `plot`, and `compare_features`. **theft** and **theftdlc** reduce the need to construct complex, bespoke workflows with multiple software libraries that were not designed to work together—the **theft** ecosystem provides an extensive suite of functions, but also presents a set of templates for advanced users to alter and adapt as their research requires.

We demonstrated the **theft** ecosystem on the five-class Bonn EEG time-series classification problem (Andrzejak et al., 2001), in which the full feature-based classification analysis pipeline—from feature extraction to normalization, classification, and interpretation of individual features—was achieved using a small number of key functions in **theft** and **theftdlc**. We showed that this intuitive pipeline could be used to derive insights about the temporal patterns which distinguish different classes of EEG time series and produce high-performing results in a simple statistical learning classification context. In other settings, emphasis may be placed on the classification procedure, where more complex classifiers—such as Gaussian processes or generalized additive models—may yield strong results. In others, users may not have a labeled dataset and instead seek to uncover structure in their data. For such cases, the `cluster` functionality of **theftdlc** may prove valuable in deriving scientific understanding. Regardless of the feature-based time-series analysis context, **theft** and **theftdlc** enable consistent, end-to-end analytical pipelines.

As new and more powerful features (and feature sets) are developed in the future, they can be incorporated into **theft** to enable ongoing assessments of the types of problems they are best placed to solve. In addition to the analysis templates provided through functions in **theftdlc**, there is much flexibility for users to adapt them or build new functionality for their own use-cases, such as applying different types of statistical learning algorithms on extracted feature matrices (e.g., feature selection), or to adapt the results to different applications such as extrinsic regression (Tan et al., 2021) or forecasting (Montero-Manso et al., 2020). Future work could also aim to reduce redundancy from across the combined features towards a new reduced feature set that combines the most generically informative and unique features from across the available feature-extraction packages (following the

aims of the `catch22` feature set, selected from a library of > 7700 candidate features in `hctsa` (Lubba et al., 2019)).

References

R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C. E. Elger. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, 64(6 Pt 1):061907, Dec. 2001. ISSN 1539-3755. doi: 10.1103/PhysRevE.64.061907. [p46, 63, 64]

M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa. TSFEL: Time Series Feature Extraction Library. *SoftwareX*, 11:100456, Jan. 2020. ISSN 2352-7110. doi: 10.1016/j.softx.2020.100456. [p45]

N. H. Barbara, T. R. Bedding, B. D. Fulcher, S. J. Murphy, and T. Van Reeth. Classifying Kepler light curves for 12,000 A and F stars using supervised feature-based machine learning. *Monthly Notices of the Royal Astronomical Society*, page stac1515, June 2022. ISSN 0035-8711. doi: 10.1093/mnras/stac1515. [p43, 52]

M. Christ, A. W. Kempa-Liehr, and M. Feindt. Distributed and parallel time series feature extraction for industrial big data applications, May 2017. [p44, 48]

M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – a Python package). *Neurocomputing*, 307: 72–77, Sept. 2018. ISSN 0925-2312. doi: 10.1016/j.neucom.2018.03.067. [p44]

R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning. STL: A seasonal-trend decomposition procedure based on loess (with discussion). *Journal of Official Statistics*, 6: 3–73, 1990. [p44]

W. H. E. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1(1):7–24, Dec. 1984. ISSN 1432-1343. doi: 10.1007/BF01890115. [p50]

N. Decat, J. Walter, Z. H. Koh, P. Sribanditmongkol, B. D. Fulcher, J. M. Windt, T. Andrillon, and N. Tsuchiya. Beyond traditional sleep scoring: Massive feature extraction and data-driven clustering of sleep time series. *Sleep Medicine*, 98:39–52, Oct. 2022. ISSN 1389-9457. doi: 10.1016/j.sleep.2022.06.013. [p52]

B. D. Fulcher. Feature-based time-series analysis. In *Feature Engineering for Machine Learning and Data Analytics*. CRC Press, 2018. ISBN 978-1-315-18108-0. [p43]

B. D. Fulcher and N. S. Jones. Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3026–3037, Dec. 2014. ISSN 1041-4347, 1558-2191, 2326-3865. doi: 10.1109/TKDE.2014.2316504. [p44, 54]

B. D. Fulcher and N. S. Jones. hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. *Cell Systems*, 5(5):527–531.e3, Nov. 2017. ISSN 2405-4712. doi: 10.1016/j.cels.2017.10.001. [p43, 44, 45, 52]

B. D. Fulcher, A. E. Georgieva, C. W. G. Redman, and N. S. Jones. Highly comparative fetal heart rate analysis. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3135–3138, Aug. 2012. doi: 10.1109/EMBC.2012.6346629. [p44]

B. D. Fulcher, M. A. Little, and N. S. Jones. Highly comparative time-series analysis: The empirical structure of time series and their methods. *Journal of The Royal Society Interface*, 10(83):20130048, June 2013. doi: 10.1098/rsif.2013.0048. [p43, 44, 46, 50, 52, 63]

B. D. Fulcher, C. H. Lubba, S. S. Sethi, and N. S. Jones. A self-organizing, living library of time-series data. *Scientific Data*, 7(1):213, July 2020. ISSN 2052-4463. doi: 10.1038/s41597-020-0553-0. [p44]

B. J. Harris. *Catch22.jl*, 2021. v0.2.1. [p44]

T. Henderson. *Rcatch22: Calculation of 22 CAnonical Time-Series CHaracteristics*, 2021. URL https://CRAN.R-project.org/package=Rcatch22. R package version 0.2.3. [p44]

T. Henderson. *normaliseR: Re-Scale Vectors and Time-Series Features*, 2024. URL https://CRAN.R-project.org/package=normaliseR. R package version 0.1.2. [p50]

T. Henderson. *correctR: Corrected Test Statistics for Comparing Machine Learning Models on Correlated Samples*, 2025a. URL https://CRAN.R-project.org/package=correctR. R package version 0.3.1. [p57]

T. Henderson. *theft: Tools for Handling Extraction of Features from Time Series*, 2025b. URL https://CRAN.R-project.org/package=theft. R package version 0.8.2. [p45]

T. Henderson. *theftdlc: Analyse and Interpret Time Series Features*, 2025c. URL https://CRAN.R-project.org/package=theftdlc. R package version 0.2.1. [p45]

T. Henderson and B. D. Fulcher. An Empirical Evaluation of Time-Series Feature Sets. In *2021 International Conference on Data Mining Workshops (ICDMW)*, pages 1032–1038, Dec. 2021. doi: 10.1109/ICDMW53433.2021.00134. [p45, 49, 50]

T. Henderson, A. G. Bryant, and B. D. Fulcher. Never a dull moment: Distributional properties as a baseline for time-series classification, 2023. URL https://arxiv.org/abs/2303.17809. [p44, 49, 61]

R. Hyndman, Y. Kang, P. Montero-Manso, T. Talagala, E. Wang, Y. Yang, and M. O'Hara-Wild. *tsfeatures: Time Series Feature Extraction*, 2020. URL https://CRAN.R-project.org/package=tsfeatures. R package version 1.1.1. [p44]

X. Jiang, S. Srivastava, S. Chatterjee, Y. Yu, J. Handler, P. Zhang, R. Bopardikar, D. Li, Y. Lin, U. Thakore, M. Brundage, G. Holt, C. Komurlu, R. Nagalla, Z. Wang, H. Sun, P. Gao, W. Cheung, J. Gao, Q. Wang, M. Guerard, M. Kazemi, Y. Chen, C. Zhou, S. Lee, N. Laptev, T. Levendovszky, J. Taylor, H. Qian, J. Zhang, A. Shoydokova, T. Singh, C. Zhu, Z. Baz, C. Bergmeir, D. Yu, A. Koylan, K. Jiang, P. Temiyasathit, and E. Yurtbay. Kats, 3 2022. URL https://github.com/facebookresearch/Kats. [p45]

I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer-Verlag, New York, 2002. ISBN 978-0-387-95442-4. doi: 10.1007/b98835. [p52]

L.-J. Kao, C.-C. Chiu, H.-J. Wang, and C. Y. Ko. Prediction of remaining time on site for e-commerce users: A SOM and long short-term memory study. *Journal of Forecasting*, 40(7): 1274–1290, 2021. doi: https://doi.org/10.1002/for.2771. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/for.2771. [p43]

E. Kodra, S. Chatterjee, and A. R. Ganguly. Exploring Granger causality between global average observed time series of carbon dioxide and temperature. *Theoretical and Applied Climatology*, 104(3):325–335, June 2011. ISSN 1434-4483. doi: 10.1007/s00704-010-0342-3. [p43]

G. Liu, L. Li, L. Zhang, Q. Li, and S. S. Law. Sensor faults classification for SHM systems using deep learning-based method with Tsfresh features. *Smart Materials and Structures*, 29(7):075005, May 2020. ISSN 0964-1726. doi: 10.1088/1361-665X/ab85a6. [p45]

M. Löning, A. Bagnall, S. Ganesh, V. Kazakov, J. Lines, and F. J. Király. sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872*, 2019. [p45]

C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones. Catch22: CAnonical Time-series CHaracteristics. *Data Mining and Knowledge Discovery*, 33(6):1821–1852, Nov. 2019. ISSN 1573-756X. doi: 10.1007/s10618-019-00647-x. [p44, 65]

M. Markicevic, B. D. Fulcher, C. Lewis, F. Helmchen, M. Rudin, V. Zerbi, and N. Wenderoth. Cortical Excitation:Inhibition Imbalance Causes Abnormal Brain Network Dynamics as Observed in Neurodevelopmental Disorders. *Cerebral Cortex*, 30(9):4922–4937, 2020. ISSN 1047-3211. doi: 10.1093/cercor/bhaa084. [p44]

P. Montero-Manso, G. Athanasopoulos, R. J. Hyndman, and T. S. Talagala. FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting*, 36(1):86–92, Jan. 2020. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2019.02.011. [p64]

C. Nadeau and Y. Bengio. Inference for the generalization error. *Machine Learning*, 52:239, 2003. [p57]

M. O'Hara-Wild, R. Hyndman, and E. Wang. *feasts: Feature Extraction and Statistics for Time Series*, 2021. URL https://CRAN.R-project.org/package=feasts. R package version 0.4.1. [p44]

M. Ojala and G. C. Garriga. Permutation Tests for Studying Classifier Performance. In *2009 Ninth IEEE International Conference on Data Mining*, pages 908–913, Miami Beach, FL, USA, Dec. 2009. IEEE. ISBN 978-1-4244-5242-2. doi: 10.1109/ICDM.2009.108. [p54]

A. Paul, H. McLendon, V. Rally, J. T. Sakata, and S. C. Woolley. Behavioral discrimination and time-series phenotyping of birdsong performance. *PLOS Computational Biology*, 17(4): e1008820, Apr. 2021. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1008820. [p44]

C. W. Tan, C. Bergmeir, F. Petitjean, and G. I. Webb. Time series extrinsic regression. *Data Mining and Knowledge Discovery*, 35(3):1032–1060, May 2021. ISSN 1573-756X. doi: 10.1007/s10618-021-00745-9. [p64]

J. Van Der Donckt, J. Van Der Donckt, E. Deprost, and S. Van Hoecke. tsflex: Flexible time series processing & feature extraction. *SoftwareX*, 17:100971, Jan. 2022. ISSN 2352-7110. doi: 10.1016/j.softx.2021.100971. [p45]

E. Wang, D. Cook, and R. J. Hyndman. A new tidy data structure to support exploration and modeling of temporal data. *Journal of Computational and Graphical Statistics*, 29(3):466–478, 2020. doi: 10.1080/10618600.2019.1695624. URL https://doi.org/10.1080/10618600.2019.1695624. [p46, 48]

M. West, R. Prado, and A. D. Krystal. Evaluation and Comparison of EEG Traces: Latent Structure in Nonstationary Time Series. *Journal of the American Statistical Association*, 94 (446):375–387, June 1999. ISSN 0162-1459. doi: 10.1080/01621459.1999.10474128. [p43]

H. Wickham. Tidy data. *Journal of Statistical Software*, 59(1):1–23, Sept. 2014. ISSN 1548-7660. doi: 10.18637/jss.v059.i10. [p44]

H. Wickham, M. Averick, J. Bryan, W. Chang, L. D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani. Welcome to the Tidyverse. *Journal of Open Source Software*, 4(43):1686, Nov. 2019. ISSN 2475-9066. doi: 10.21105/joss.01686. [p44]

H. Wickham, R. François, L. Henry, K. Müller, and D. Vaughan. *dplyr: A Grammar of Data Manipulation*, 2023. URL https://CRAN.R-project.org/package=dplyr. R package version 1.1.4. [p58]

Z. Yang, I. A. Abbasi, E. E. Mustafa, S. Ali, and M. Zhang. An anomaly detection algorithm selection service for IoT stream data based on tsfresh tool and genetic algorithm. *Security and Communication Networks*, 2021:6677027, Feb. 2021. ISSN 1939-0114. doi: 10.1155/2021/6677027. [p45]

Trent Henderson
The University of Sydney
School of Physics
Sydney, Australia
ORCiD: 0009-0005-5467-9914
then6675@uni.sydney.edu.au

Ben D. Fulcher
The University of Sydney
School of Physics
Sydney, Australia
ORCiD: 0000-0002-3003-4055
ben.fulcher@sydney.edu.au

QTE.RD: An R Package for Quantile Treatment Effects in Regression-Discontinuity Designs

by Zhongjun Qu and Jungmo Yoon

Abstract The QTE.RD package provides methods to test, estimate, and conduct uniform inference on quantile treatment effects in sharp regression discontinuity designs, allowing for covariates, and implementing robust bias correction. The package offers four main functions for estimating quantile treatment effects and uniform confidence bands, testing hypotheses related to treatment effects, selecting bandwidths using cross-validation or mean squared error criteria, and visualizing the estimated effects and confidence bands. This note includes an empirical illustration of the package's functionality using data on the impact of tracking on student achievement.

1 Introduction

The regression discontinuity (RD) design (Thistlethwaite and Campbell, 1960) has become an important methodology for identifying and estimating causal effects from observational data. Under a sharp RD design, the assignment to a treatment is fully determined by whether the value of a covariate, known as the running variable, surpasses a fixed cutoff. The randomization near the cutoff allows for the estimation of treatment effects by comparing individuals above the threshold with those below it. To date, the majority of studies in the RD literature have focused on the average treatment effect (ATE).

Treatment effects are often heterogeneous, and the concept of quantile treatment effects (QTE; Lehmann (1975), and Doksum (1974)) offers a flexible framework for documenting the heterogeneity. Four key issues often arise in such contexts: 1) Constructing a uniform confidence band that covers the quantile treatment effects at a given confidence level; 2) Testing the statistical significance of the treatment effect within a given quantile range (Treatment Significance); 3) Assessing whether the treatment effects are equal across all quantiles (Treatment Homogeneity); 4) Determining if the effects are uniformly positive or uniformly negative within this quantile range (Treatment Unambiguity).

Furthermore, if heterogeneity is detected by examining the above issues, utilizing covariates can help pinpoint the source of the heterogeneity. For instance, consider a sample comprising both males and females. If the quantile treatment effects are equal within each gender group but differ between groups, introducing a gender dummy into the model will reveal homogeneous quantile treatment effects for both groups. In this context, QTE estimates plotted as a function of the quantile index should show that the effects are identical within each group but differ between them. However, if the QTE demonstrates a non-zero slope in quantile for any subgroup, it indicates that treatment heterogeneity persists even after accounting for initial covariates. This then indicates the need for further analysis with additional explanatory variables to fully understand the underlying heterogeneity.

These considerations motivated the study of Qu and Yoon (2019) and Qu et al. (2024). The former study develops methods for conducting uniform inference on QTEs for sharp RD designs without covariates, while the latter paper extends the methods to allow for covariates. The proposed package implements their methods in an easy-to-use fashion. Four functions are provided:

1. `rd.qte()`. This function provides point estimates of QTEs over a range of quantiles and a uniform confidence band that covers these effects at a given confidence level. The estimation is based on local-linear regressions. The user can specify whether or not to include any covariates and how many of them to include in the regression.

The function provides results either without any bias correction or with robust bias corrections (Qu et al. (2024)).

2. `rdq.test()`. This function provides testing results for three hypotheses on the treatment effects outlined above: Treatment Significance, Homogeneity, and Unambiguity. The user can choose whether to allow for covariates, and whether to conduct robust bias correction. The critical values are obtained via simulations when implementing these tests.

3. `rdq.bandwidth()`. This function implements two bandwidth selection rules: the cross-validation bandwidth and the MSE optimal bandwidth. In practice, one can apply both methods to examine result sensitivity.

4. `plot.qte()`. This function generates figures summarizing the QTE estimates and their uniform confidence bands, helping users visualize the results from testing and estimation.

For the validity of these methods, the covariates must be balanced at the cutoff, meaning that their distribution does not change discontinuously at the RD threshold. Correlations with the running variable are permitted. Bias correction is applied to ensure that the tests and confidence intervals achieve correct asymptotic coverage. This is necessary because the underlying nonparametric functions are approximated using local linear methods, and omitted terms can distort inference if not properly accounted for—a well-known issue in the nonparametric estimation literature. Our proposed procedure not only estimates the bias but also accounts for the uncertainty in that estimation, which is why we refer to it as robust bias correction.

When implementing these methods, users need to supply the following input: the outcome variable in `y`, treatment status in `d` (0 or 1), independent variables in `x`, and a quantile range $\mathcal{T}$. The first column of `x` is a scalar running variable, and the remaining columns in `x` include additional covariates, which can be discrete or continuous. When there are no covariates, `x` is just a column vector of the running variable.

Let x_0 denote the cutoff and z_0 the value of the remaining covariates at which to evaluate the effects (e.g., if a female dummy is included, $z_0 = 1$ indicates the female subgroup). The main objects of the analysis are the conditional quantile functions on the right and left sides of the cutoff: $Q(\tau|x_0^+, z_0)$ and $Q(\tau|x_0^-, z_0)$, and the QTE at this cutoff: $Q(\tau|x_0^+, z_0) - Q(\tau|x_0^-, z_0)$.

The above functions can also be used to analyze data from a randomized controlled trial (RCT). In this case, the two sides of the cutoff are replaced by observations from the control and treatment groups, respectively. Let $Q_1(\tau|x, z)$ be the conditional quantile function of the treatment ($d = 1$) group and $Q_0(\tau|x, z)$ be that of the control ($d = 0$) group, then the QTE at (x, z) is defined as $Q_1(\tau|x, z) - Q_0(\tau|x, z)$. If we have $x = x_0$ for some x_0, the estimate we provide will correspond to the local treatment effect near the chosen x_0, placing no restriction on the effects away from x_0.

Unlike in the RD setting, the choice of x in the RCT setting involves making a modeling decision: x typically represents a baseline variable that is highly predictive of the outcome, while z is used to examine additional treatment heterogeneity. This approach allows quantile treatment effects to vary nonparametrically with the baseline variable and linearly with additional covariates. In our empirical example, we use the baseline test score as x. Because it is highly predictive of the outcome (the endline test score), it is natural to examine the effects of tracking separately for students at different points in the initial performance distribution. To capture further heterogeneity, we use student and teacher characteristics as covariates z, which enables us to examine how treatment effects vary linearly with observable student and classroom factors.

There are several *R* and *Stata* packages available for estimating QTE. Table 1 provides a comparison of these packages with **QTE.RD**. As indicated in the table, to our knowledge, there is currently no *R* package for estimating QTEs under RD designs, even for the simplest

setting without covariates. There are, however, *Stata* functions **rddqte** and **rdqte** that can be used in RD designs by applying methods in Frandsen et al. (2012) and Chiang et al. (2019). These *Stata* functions are particularly useful for fuzzy designs. But they do not offer methods allowing heterogeneous effects by groups or continuous covariates. So if the goal is to explore the heterogeneity in full generality for sharp RD designs, it is suitable to use **QTE.RD**.

Additionally, some *R* and *Stata* packages are available for estimating QTE under alternative identification strategies. The R packages **qte** and **quantreg.nonpar** rely on the conditional independence (or unconfoundedness) assumption between the potential outcomes and the selection variable. The Stata package **ivqte** is based on the availability of instrumental variables. While they are useful for these alternative research designs, they are not directly applicable to RD designs.

Table 1: Comparisons of R/Stata packages and functions

Package Name	Statistical Platform	Research Design	Hetero. Effects	Conf. Band	Bias Correction	Regression Model	Estimation Method
qte	R	CI	×	pointw	×	QR	Linear
ivqte	Stata	IV	×	pointw	×	DR	Linear
quantreg.nonpar	R	CI	×	uniform	✓	QR	Series
rddqte	Stata	RDD	×	pointw	×	DR	Local
rdqte	Stata	RDD	×	uniform	✓	DR	Local
QTE.RD	R	RDD	✓	uniform	✓	QR	Local

Note: 'CI' = conditional independence (unconfoundedness); 'IV' = instrumental variable; 'RDD' = regression discontinuity design; 'pointw' = confidence band is pointwise w.r.t. the quantile level τ; 'uniform' = confidence band is uniform in τ. 'QR' = quantile regression; 'DR' = distributional regression; 'Linear' = linear regression; 'Series' = series estimation; 'Local' = local polynomial regression. The symbol ✓ means that the indicated feature is available.

We apply the functions of this package to study the impact of tracking (assigning students into separate classes by prior achievement) on student achievement using the dataset of Duflo et al. (2011). Their experimental data includes 121 primary schools in Kenya which received funds in 2005 to hire a new teacher and split their single first-grade class into two sections. The schools were randomly divided into the treated group, 61 tracking schools, and the control group, 60 non-tracking schools. In tracking schools, students were assigned to sections based on baseline test scores. In non-tracking schools, students were randomly assigned.

The experimental design has rich random variations, featuring elements of both randomized controlled trials (RCT) and RD. By comparing tracking and non-tracking schools, that is, by exploiting the RCT structure, one can study the effect of tracking on all students. Additionally, by analyzing median students within tracking schools, that is, by exploiting the RD structure, one can study the effect of tracking on marginal students who barely made or missed the opportunity of being assigned to a high ability section. Both structures were exploited in Duflo et al. (2011), though they focused on average effects instead of quantile effects.

The experiment lasted for 18 months. The main outcome variable is the sum of math and language scores on the endline tests administered in all schools at the end of the program. Duflo et al. (2011) also examined the long-run effect using a follow-up test which was given one year after the tracking ended. We analyze the short-term effect by focusing on the endline test. To study the heterogeneity in effects, we use as covariates the baseline test score, students' gender and age (at the endline test), and whether teachers are civil servants or contract teachers.

From the RD design, we find no evidence that marginal students assigned to the lower section (i.e., those falling just below the median of the initial achievement distribution) performed any worse than those assigned to the upper section, despite the fact that the latter group had higher achieving peers. This finding, while confirming Duflo et al. (2011), is more definitive in that it documents the lack of effect, not only on average, but also at any points of the outcome distribution. Examining heterogeneity across groups by covariates,

we find that the effects of assigning to an upper section are negative for girls but positive for boys across quantiles. In particular, female students who were at the bottom of the endline scores may fare worse if they were assigned to the upper section. However, these effects are mostly statistically insignificant, suggesting that a larger dataset is needed to draw more precise conclusions regarding this heterogeneity.

From the RCT design, we find uniformly positive effects of tracking. Test scores were higher in tracking schools than in non-tracking schools up to 0.351 standard deviations. The null hypothesis of no effect is firmly rejected. The biggest effects can be found for students who were at the middle of the baseline test distribution, male and younger students, and those taught by contract teachers. Our findings from the RCT support that tracking may be beneficial to all students, not just to those assigned to high achievement sections. If the peer effect is the dominant factor in the effect of tracking, the marginal students who are assigned to the lower section may have much to lose. But our findings from the RDD indicate that even this group did not suffer at all. This can be explained if, as Duflo et al. (2011) argued, tracking allows teachers to closely match their instruction to the need of students and benefits all students.

Below, we first review the statistical methods implemented in the package, and after that we will provide details on the implementation in the context of the empirical application.

2 Methods for quantile treatment effects with and without covariates

This section presents materials in the following order: the model and the issues of interest, the main estimation steps along with the bandwidth selection methods, the uniform confidence bands with and without bias correction, and finally, the computation of tests related to treatment significance, homogeneity, and unambiguity. We also highlight that the methods can be used to estimate local effects in RCTs in addition to RD designs.

2.1 Model

Let y represent the outcome variable, x be the running variable, and z ($q \times 1$) be a set of covariates. We focus on the sharp RD design in which the treatment status shifts at $x = x_0$: no one below x_0 is treated, and everyone above x_0 is treated. For theoretical analysis, define two local neighborhoods of x_0: the left neighborhood $\mathbb{B}^-(x_0) = [x_0 - \delta, x_0)$ and the right neighborhood $\mathbb{B}^+(x_0) = [x_0, x_0 + \delta]$ with δ a small positive constant. With $Q(\tau|x,z)$ denoting the τ-th conditional quantile of y given x and z, the model we assume to hold is given by:

$$\begin{aligned} Q(\tau|x,z) &= g_1(x,\tau) + z'\beta_1(\tau) + xz'\gamma_1(\tau) \text{ over } \tau \in \mathcal{T} \text{ for any } x \in \mathbb{B}^-(x_0), \\ Q(\tau|x,z) &= g_2(x,\tau) + z'\beta_2(\tau) + xz'\gamma_2(\tau) \text{ over } \tau \in \mathcal{T} \text{ for any } x \in \mathbb{B}^+(x_0), \end{aligned}$$

where $g_1(x,\tau)$ and $g_2(x,\tau)$ are continuous nonparametric functions of x and τ. For a given z, the QTE at the τ-th quantile is defined as

$$\delta(\tau|z) = \delta(\tau|x_0,z) = \lim_{x \downarrow x_0} Q(\tau|x,z) - \lim_{x \uparrow x_0} Q(\tau|x,z),$$

where $\lim_{x\downarrow x_0}$ denotes the value of the function as x approaches the limit from the right side of the cutoff, and $\lim_{x\uparrow x_0}$ from the left side of the cutoff. Explicit conditions on the model are stated in Assumptions 1–4 of Qu et al. (2024). These allow for correlations between the running variable and the covariates.

We will denote the two right hand side limiting expressions by $Q(\tau|x_0^+,z)$ and $Q(\tau|x_0^-,z)$, respectively. The methods in this package primarily address the following issues:

1. Uniform Confidence Band. For any given z and coverage level p, obtain a band $[L_p(\tau|z), U_p(\tau|z)]$ such that $\Pr\{\delta(\tau|z) \in [L_p(\tau|z), U_p(\tau|z)] \text{ for all } \tau \in \mathcal{T}\} \geq p$ holds asymptotically.

2. Treatment Significance. Test $H_0 : \delta(\tau|z) = 0$ for all $\tau \in \mathcal{T}$ against $H_1 : \delta(\tau|z) \neq 0$ for some $\tau \in \mathcal{T}$.

3. Treatment Homogeneity. Test $H_0 : \delta(\tau|z)$ is constant over $\mathcal{T}$ against $H_1 : \delta(\tau|z) \neq \delta(s|z)$ for some $\tau, s \in \mathcal{T}$.

4. Treatment Unambiguity. Test $H_0 : \delta(\tau|z) \geq 0$ over $\mathcal{T}$ against $H_1 : \delta(\tau|z) < 0$ for some $\tau \in \mathcal{T}$. Alternatively, test $\delta(\tau|z) \leq 0$ over $\mathcal{T}$ against $\delta(\tau|z) > 0$ for some $\tau \in \mathcal{T}$.

2.2 Estimation

The estimation is based on local linear quantile regressions, in which the conditional quantile functions on the two sides of the cutoff are approximated by:

$$\begin{aligned} Q(\tau|x,z) &\approx \alpha_0^-(\tau) + \alpha_1^-(\tau)(x - x_0) + z'\beta^-(\tau) + (x - x_0)z'\gamma^-(\tau) \text{ (for } x \text{ below the cutoff)}, \\ Q(\tau|x,z) &\approx \alpha_0^+(\tau) + \alpha_1^+(\tau)(x - x_0) + z'\beta^+(\tau) + (x - x_0)z'\gamma^+(\tau) \text{ (for } x \text{ above the cutoff)}. \end{aligned}$$

The interactive terms $(x - x_0)z'\gamma^-(\tau)$ and $(x - x_0)z'\gamma^+(\tau)$ are important and they are discussed below in Remark 1. The estimation solves the following two minimization problems separately:

$$\begin{aligned} &\min_{\alpha_0^-, \alpha_1^-, \beta^-, \gamma^-} \sum_{i=1}^{n} \rho_\tau \left(y_i - \alpha_0^- - \alpha_1^-(x_i - x_0) - z_i'\beta^- - (x_i - x_0)z_i'\gamma^- \right) (1 - d_i) K\left((x_i - x_0)/b_{n,\tau}\right), \\ &\min_{\alpha_0^+, \alpha_1^+, \beta^+, \gamma^+} \sum_{i=1}^{n} \rho_\tau \left(y_i - \alpha_0^+ - \alpha_1^+(x_i - x_0) - z_i'\beta^+ - (x_i - x_0)z_i'\gamma^+ \right) d_i K\left((x_i - x_0)/b_{n,\tau}\right), \end{aligned}$$

where n is the sample size, x_i is the running variable value for individual i, $d_i = 1(x_i \geq x_0)$ is the treatment indicator, z_i is a set of covariates, ρ_τ is the check function: $\rho_\tau(u) = u(\tau - 1\{u < 0\})$, $K(\cdot)$ is a kernel function, and $b_{n,\tau}$ is a quantile-dependent bandwidth discussed later. See Koenker (2005) for a comprehensive treatment of quantile regressions and Yu and Jones (1998) for local linear quantile regressions.

In the implementation, we solve the above two optimization problems for an equidistant grid of quantiles over $\mathcal{T}$ and then apply linear interpolation between adjacent quantiles to obtain continuous functions over quantiles. This gives us the estimated conditional quantile curves on the two sides of the cutoff: $\hat{Q}(\tau|x_0^-, z) = \hat{\alpha}_0^-(x, \tau) + z'\hat{\beta}^-(\tau)$, and $\hat{Q}(\tau|x_0^+, z) = \hat{\alpha}_0^+(x, \tau) + z'\hat{\beta}^+(\tau)$. The QTE estimate, prior to any bias correction, is given by

$$\hat{\delta}(\tau|z) = \hat{Q}(\tau|x_0^+, z) - \hat{Q}(\tau|x_0^-, z) \text{ for any } \tau \in \mathcal{T}.$$

This QTE estimate is affected by a bias term that depends on the second-order derivative of the conditional quantile function; its expression is given in Corollary 2 of Qu et al. (2024). The main effect of this bias is to distort the coverage level of the confidence band and the rejection frequency of the hypothesis tests under the null hypothesis. This motivates the usage of bias-corrected estimates at the cost of estimation efficiency. To estimate the bias, we first run two local quadratic regressions for the two sides of the cutoff for each τ. To that end, we solve the same minimization problem as the local linear regression case, except the local linear approximation is replaced by quadratic regression with the same bandwidth $b_{n,\tau}$. Then, the bias-corrected estimator is computed as (x can be either x_0^+ or x_0^-): $\hat{Q}(\tau|x, z) - \hat{B}_v(x, z, \tau)b_{n,\tau}^2$. The bias correction affects the distribution of the QTE estimator, and our methods incorporate an extra term into the distribution to account for this additional estimation uncertainty motivated by Calonico et al. (2014); see the discussions in Subsection 2.4.

Remark 1 *We now discuss how to interpret the estimates to ease the application. If z_i is a dummy variable, e.g., equal to one for females, then the QTEs for men and women are given by $\alpha_0^+(\tau) - \alpha_0^-(\tau)$ and $\alpha_0^+(\tau) - \alpha_0^-(\tau) + \beta^+(\tau) - \beta^-(\tau)$. If z_i is a continuous variable, then the QTE at $x = x_0$ for*

$z = z_0$ is given by $\alpha_0^+(\tau) - \alpha_0^-(\tau) + z_0'(\beta^+(\tau) - \beta^-(\tau))$. The interactive term $(x_i - x_0)z_i'$ makes $\partial Q(\tau|x,z)/\partial x$ vary with z. If z_i is a binary variable, then this slope is equal to α_1^+ and $\alpha_1^+ + \gamma^+$ for $z_i = 0$ and $z_i = 1$, respectively. It is essential to allow the coefficients of z_i to change at the cutoff, otherwise, the QTE estimate will be biased if the treatment effects are heterogeneous across z values.

Remark 2 *Including covariates does not appear to improve estimation efficiency in the quantile RD setting, because their role differs from the mean RD case. To see this, consider two scenarios. In both cases, suppose the running variable is x, the cutoff is at x_0, and the covariate z is binary ($z = 0, 1$), representing two groups. In the first scenario, the treatment effect is heterogeneous in z. If the model does not contain any covariates (i.e., z is not included), the RD estimator identifies the unconditional quantile treatment effect over the groups. Once z is included, the estimator recovers quantile treatment effects separately for groups 0 and 1. That is, when heterogeneity is present, including covariates leads to different estimands. The resulting estimates are not directly comparable, and the issue is therefore not efficiency. This contrasts with the RD-in-mean setting, where including covariates does not change the estimand: the intercept still identifies the average treatment effect, as shown in Calonico et al. (2019). In the second scenario, the treatment effect is homogeneous across z. Then, the true coefficient on the $z = 1$ indicator is equal to zero, and including z may actually reduce efficiency, as it increases the number of estimated parameters without reducing residual variation. For additional discussion and details, see Section 3.1 in Qu et al. (2024).*

2.3 Bandwidth selection

The package offers two methods to choose bandwidth parameters: cross-validation and minimizing the MSE. In both cases, the bandwidth at the median $b_{n,0.5}$ is determined first. This value is then used to determine bandwidths at other quantiles, using the formula of Yu and Jones (1998):

$$(b_{n,\tau}/b_{n,0.5})^{4+d} = 2\tau\,(1-\tau)\,/\,[\pi\phi(\Phi^{-1}(\tau))^2] \;\text{ for } \tau \in \mathcal{T},$$

where ϕ and Φ are the standard normal density and cumulative distribution functions.

Cross validation bandwidth: For a given candidate bandwidth, estimate the conditional median at (x_i, z_i) by a local linear or quadratic regression, treating x as an interior or a boundary point, leaving out (y_i, x_i, z_i). The goodness of fit is measured by the difference between y_i and the estimated conditional median. Repeat the estimation and compute the mean absolute deviation over 50% of the observations closest to x. The cross-validation bandwidth minimizes this mean absolute deviation.

MSE-optimal bandwidth: First obtain a pilot bandwidth for the median using leave-one-out cross validation. Then construct the MSE-optimal bandwidth for the median by applying this pilot bandwidth to calculate the necessary quantities in the bandwidth formula from Corollary 3 of Qu et al. (2024).

Providing two selection rules (the cross-validation bandwidth selection rule and the MSE-optimal rule) allows users to assess the sensitivity of their results to different choices. However, we note that, although the cross-validation bandwidth is intuitive, its theoretical properties in the current setting have not been formally studied. The package also allows users to directly specify bandwidth values without using these two methods, providing an additional channel for robustness analysis.

2.4 Uniform confidence band with/without robust bias correction

The confidence band we compute relies on the following asymptotic approximation in Corollary 2 of Qu et al. (2024):

$$(nb_{n,\tau})^{1/2}(\hat{Q}(\tau|x,z) - Q(\tau|x,z) - b_{n,\tau}^2 B_v(x,z,\tau)) = D_{1,v}(x,z,\tau) + o_p(1),$$

where x can be either x_0^+ or x_0^-, $B_v(x,z,\tau)$ is a bias term, and $D_{1,v}(x,z,\tau)$ converges to a Gaussian process over τ with mean zero with a pivotal distribution conditioning on the data, making it readily simulatable.

Using this approximation, the uniform confidence band without bias correction (e.g., ignoring the bias) is computed as follows: Define $\sigma_{n,\tau}$ to be an estimate of $(nb_{n,\tau}^d)^{-1/2}[\mathrm{E}D_{1,v}(x,z,\tau)^2]^{1/2}$, obtained via simulations. Compute the band as $[\hat{Q}(\tau|x,z) - \sigma_{n,\tau}C_p,\ \hat{Q}(\tau|x,z) + \sigma_{n,\tau}C_p]$, where C_p is the p-th percentile of $\sup_{\tau\in\mathcal{T}} |D_{1,v}(x,z,\tau) / \sqrt{\mathrm{E}D_{1,v}(x,z,\tau)^2}|$. This band is wider at quantiles with sparse data.

To obtain a confidence band with robust bias correction, we implement the following steps: First, run a local quadratic regression for each quantile to estimate the bias $B_v(x,z,\tau)$, denoted as $\hat{B}_v(x,z,\tau)$, and compute the bias corrected estimator $(nb_{n,\tau}^d)^{1/2}(\hat{Q}(\tau|x,z) - \hat{B}_v(x,z,\tau)b_{n,\tau}^2)$. This estimator admits the following approximation by Lemma 2 in Qu et al. (2024): $(nb_{n,\tau}^d)^{1/2}(\hat{Q}(\tau|x,z) - \hat{B}_v(x,z,\tau)b_{n,\tau}^2 - Q(\tau|x,z)) = D_{1,v}(x,z,\tau) - D_{2,v}(x,z,\tau) + o_p(1)$ over $\mathcal{T}$, where $D_{1,v}(x,z,\tau)$ is as stated above, and the new term $D_{2,v}(x,z,\tau)$, is due to bias estimation. The terms $D_{1,v}(x,z,\tau)$ and $D_{2,v}(x,z,\tau)$ capture the estimation uncertainty of $\hat{Q}(\tau|x,z)$ and $\hat{B}_v(x,z,\tau)$, respectively. The resulting uniform confidence band is $[\hat{Q}(\tau|x,z) - \hat{B}_v(x,z,\tau) - \sigma_{n,\tau}C_p,\ \hat{Q}(\tau|x,z) - \hat{B}_v(x,z,\tau) + \sigma_{n,\tau}C_p]$, where $\sigma_{n,\tau}$ and C_p are as before, with $D_{1,v}(x,z,\tau)$ replaced by $D_{1,v}(x,z,\tau) - D_{2,v}(x,z,\tau)$. This band is centered at the bias corrected estimate and is wider than the band without bias correction due to the presence of $D_{2,v}(x,z,\tau)$. More details on computing this confidence band can be found in PROC-A on p.528 of Qu et al. (2024).

2.5 Hypothesis testing

To compute the tests, as before, let $\hat{\delta}(\tau|z) = \hat{Q}(\tau|x_0^+,z) - \hat{Q}(\tau|x_0^-,z)$. Let $\hat{w}(\tau) \geq 0$ be a user-chosen weight function, satisfying $\hat{w}(\tau) \xrightarrow{p} w(\tau)$, a smooth function over $\mathcal{T}$. Define $W(\tau) = (nb_{n,\tau}^d)^{1/2}\hat{w}(\tau)(\hat{\delta}(\tau|z) - b_{n,\tau}^2(\hat{B}_v(x_0^+,z,\tau) - \hat{B}_v(x_0^-,z,\tau)))$, where $\hat{B}_v$ represents the bias estimate. The hypotheses of treatment significance, homogeneity, and unambiguity are tested using the following statistics, respectively:

$$
\begin{aligned}
WS(\mathcal{T}) &= \sup_{\tau\in\mathcal{T}} |W(\tau)|, \\
WH(\mathcal{T}) &= \sup_{\tau\in\mathcal{T}} \left| W(\tau) - \frac{\sqrt{nb_{n,\tau}^d}\hat{w}(\tau)}{\int_{s\in\mathcal{T}} \sqrt{nb_{n,s}^d}\hat{w}(s)ds} \int_{\tau\in\mathcal{T}} W(\tau)d\tau \right|, \\
WA(\mathcal{T}) &= \sup_{\tau\in\mathcal{T}} |1(W(\tau) \leq 0) W(\tau)|.
\end{aligned}
$$

In the case of non-positive effects under the null hypothesis, replace $1(W(\tau) \leq 0)$ by $1(W(\tau) \geq 0)$. The tests have built-in bias corrections. No restrictions on biases are imposed across quantiles. To implement tests without bias correction, simply omit the term $(\hat{B}_v(x_0^+,z,\tau) - \hat{B}_v(x_0^-,z,\tau))$ when computing the test, and the critical values are adjusted automatically.

Sometimes it is desirable to set $\hat{w}(\tau)$ such that the standard deviation of $(nb_{n,\tau}^d)^{1/2}(\hat{\delta}(\tau|z) - b_{n,\tau}^2(\hat{B}_v(x_0^+,z,\tau) - \hat{B}_v(x_0^-,z,\tau)))$ is equalized across quantiles under the null hypothesis. Or, one might assign equal weight to all quantiles. The package provides both options. The asymptotic distributions of the tests, under a general $\hat{w}(\tau)$, are given in Corollary 5 of Qu et al. (2024).

2.6 Local QTE in RCT

As highlighted in the introduction, the functions in **QTE.RD** are flexibly designed to accommodate more than the RD design. For example, they can be used to analyze data from a randomized controlled trial. In this case, the two sides of the cutoff are replaced by observations from the control and treatment groups, respectively. The nonparametric component of the model x will be a variable that is highly predictive of the outcome of the

experiment. The linear component of the model includes other covariates z, which explores the heterogeneity in the treatment effect.

Specifically, let $Q_1(\tau|x,z)$ be the conditional quantile function of the treatment ($d = 1$) group and $Q_0(\tau|x,z)$ be that of the control ($d = 0$) group, then the QTE at (x,z) is defined as $Q_1(\tau|x,z) - Q_0(\tau|x,z)$. If $x = x_0$ for some x_0, the estimate gives the local effect near a chosen covariate value x_0, placing no restriction on the effects away from x_0. The main difference from the RD case is that here x_0 typically represents an interior point instead of a boundary point for the purpose of estimation and inference. The next section will focus on the RD setting only, while the empirical application section shows how to use the functions in the package to analyze data from the RCT as well as from the RDD.

3 R functions

This section explains the main R functions in the package.

`QTE and Uniform confidence band`

The function `rd.qte()` provides the QTE and $100 \cdot (1-\alpha)\%$ uniform confidence band. Save data in `y`, `x`, `d`, specify appropriate values for `x0`, `z0`, `tau`, and run

```
rd.qte(y,x,d,x0,z0,tau,bdw=8,bias=1)
```

This line of code estimates the conditional QTE with bias correction. When no covariates are included, `x` is simply a vector of the running variable and `z0` can be left unspecified. When covariates are included, `x` should be a matrix with the running variable in the first column and the covariates in the remaining columns. In this case, `z0`, which specifies the covariate subgroups to be evaluated, must be explicitly provided, as illustrated in Remark 1. The option `bias=1` means that the QTE estimate is bias corrected. When `bias=0`, the above command estimates the QTE without bias correction. Additional arguments have the following meanings. The quantile indexes to estimate, $\mathcal{T}$, are denoted by `tau`. For example, when $\mathcal{T} = [0.1, 0.9]$, one may set '`tau = seq(0.1,0.9,by=0.05)`'. It will generate an evenly spaced grid with increment 0.05. If `bdw` is set to a scalar, then it is interpreted as the bandwidth for the median, and the bandwidth values for other quantiles are determined within the code using Yu and Jones's (1998) formula. If `bdw` is a vector with the same dimension as `tau`, then the program will use these values for the respective quantiles accordingly.

If a user saves outputs of `rd.qte()` in an object `A`, the QTE estimate is saved in `A$qte`. `A` also has a few extra outcomes. `A$qm.est` is $\hat{Q}(\tau|x_0^-,z)$ and `A$qp.est` is $\hat{Q}(\tau|x_0^+,z)$. To obtain a uniform band, one can use `summary.qte()`.[1] This can be done by

```
summary(A,alpha=0.1)
```

Because '`bias=1`' when running `rd.qte()`, the uniform band that will be produced is the robust confidence band. If '`bias=0`', the uniform band would not incorporate bias adjustments. Because '`alpha=0.1`', one will get a 90% uniform band.

If a user saves outputs of the summary function in an object `A2`, the uniform confidence band will be saved in `A2$uband`. If '`bias=1`', `A$qte` and `A2$uband` are the bias-corrected QTE and uniform bands, and if '`bias=0`', they are not bias corrected. In addition, the uniform confidence bands for $\hat{Q}(\tau|x_0^-,z)$ and $\hat{Q}(\tau|x_0^+,z)$ are saved in `A2$uband.m` and `A2$uband.p`, respectively. These conditional quantile functions will be bias corrected if '`bias=1`'. For all results, the values are ordered as in `tau`. For example, if '`tau= seq(0.1,0.9,by=0.05)`', then the first value is for the 10-th percentile, and so forth.

`Testing Hypotheses on QTE`

[1] This function is a S3 method for class `qte`.

The function `rdq.test()` tests the hypotheses on QTE. To test the treatment significance hypothesis, run

```
rdq.test(y,x,d,x0,z0,tau,bdw,bias,alpha=0.1,type=1,std.opt)
```

The option `alpha` sets the desired confidence level $1-\alpha$. When '`alpha=0.1`', one will get a critical value at the 10% level. When '`alpha=c(0.1,0.05)`', critical values at the 10% and 5% levels will be reported. The bandwidth value `bdw` can be either a scalar (setting the bandwidth at the median) or a vector with the same length as `tau`. If '`bias=1`', the test statistic is bias corrected and critical values are robust to the bias correction. When '`std.opt=1`', the test statistic is standardized by the pointwise standard deviations of the limiting process. As a result, the quantiles that are estimated imprecisely receive less weight in the construction. When '`std.opt=0`', the tests are not standardized, i.e., setting $\hat{w}(\tau) = 1$, as explained in Section 2.5. The default is '`std.opt=1`'.

To test the treatment homogeneity hypothesis, just change the `type` option to '`type=2`'. For the unambiguity hypothesis with the effects unambiguously positive under the null hypothesis[2], run

```
rdq.test(y,x,d,x0,z0,tau,bdw,alpha=0.1,type=3)
```

Conversely, if the effects are unambiguously negative under the null hypothesis, set '`type=4`'.[3] One can set multiple values for `type`. For example, when '`type=c(1,2,3,4)`', all four hypotheses (significance, homogeneity, positive and negative unambiguity) will be tested.

If a user saves outputs of `rdq.test()` in an object `B`, test statistics, critical values, and p-values are saved in `B$test.stat`, `B$cr.value`, and `B$p.value`, respectively.

Bandwidth selection

To obtain a bandwidth (at the median), run

```
rdq.bandwidth(y,x,d,x0,z0,cv=1,val=5:20)
```

The function `rdq.bandwidth()` can provide two types of bandwidth: the cross-validation (CV) bandwidth and the (MSE) optimal bandwidth. When '`cv=1`', the function produces both. When '`cv=0`', the MSE optimal bandwidth is obtained. The CV bandwidth is global with respect to the model. Even when QTEs are conditional on covariates, a single CV bandwidth will be obtained. In contrast, the MSE optimal bandwidth is local to z_0, meaning that the optimal bandwidth values will be different across covariate subgroups.

The CV bandwidth requires a series of candidate values. When '`val=5:20`' as in the example above, the CV procedures tries each of $\{5, 6, \ldots, 20\}$ to select the optimal CV bandwidth value.

The optimal bandwidth requires a pilot bandwidth in order to estimate nuisance parameters in the asymptotic MSE expression. When '`cv=1`', the CV bandwidth is used for the pilot bandwidth at $\tau = 0.5$. When '`cv=0`', the value provided by the argument `hp` will be used for the pilot bandwidth. See the command below for this case. If '`cv=1`', `hp` can be ignored.

```
rdq.bandwidth(y,x,d,x0,z0,cv=0,hp=10)
```

`rdq.bandwidth()` has some additional arguments as shown below.

```
rdq.bandwidth(y,x,d,x0,z0,cv=1,val=5:10,pm.each=1,bdy=1,p.order=1,xl=0.5)
```

[2]This hypothesis is sometimes referred to as the positive dominance hypothesis.
[3]This type of unambiguity hypothesis is referred to as the negative dominance hypothesis.

In the case the additional arguments are not specified, reasonable default values will be used. These additional arguments have the following meanings.

The option `pm.each` concerns the CV bandwidth. When '`pm.each=1`', it calculates the CV bandwidth on each side of the cutoff `x0`. When '`pm.each=0`', it will treat `x0` as an interior point (assume that there is no jump at x_0) and obtains a single CV bandwidth. The default is '`pm.each=0`'.

The option `p.order` determines how the CV bandwidth is calculated. When '`p.order=1`', a local linear regression is used, when '`p.order=2`', a local quadratic regression is used. The default is the local linear regression.

When the option '`bdy=1`', the CV procedure treats each evaluation point as a boundary value, as suggested in Imbens and Lemieux (2008). This is the default option. When '`bdy=0`', the CV procedure treats each evaluation point as an interior value. If '`xl=0.5`', then the CV bandwidth uses the 50% of observations closest to x_0. The default value is 0.5. The MSE-optimal bandwidth uses the boundary value formula. Note that it is valid for an interior point as well.

If a user saves outputs of `rdq.bandwidth()` in an object `C`, and if '`cv=1`', `C$cv` is the CV bandwidth. For the MSE optimal bandwidth, values for each side of the cutoff are calculated separately. `C$opt.m` (and `C$opt.p`) is the optimal bandwidth from the left (right) side of the cutoff. All the reported bandwidth values are for the median where $\tau = 0.5$.

`Plot QTE`

The function `plot.qte()` makes QTE plot with uniform confidence bands. It has the syntax[4]

```
plot(A2)
```

where `A2` is an object produced by `summary.qte()` fitting. The required inputs are the quantile index (saved in `A2$tau`), the QTE estimate (`A2$qte`), and the uniform band (`A2$uband`). The function has an option `ptype`. Set `ptype=1` to obtain QTE plots and `ptype=2` for conditional quantile plots. The default value is 1.

Section 4 offers step-by-step guides for these functions using an empirical example, which features RCT as well as RDD.

4 Impact of tracking

As explained in the introduction, in this randomized experiment conducted in Kenya, schools were randomly assigned to tracking and non-tracking schools. This variation from the RCT allows one to test whether tracking is beneficial to all students, including low achieving students. Within tracking schools, students above the median of the baseline test were assigned to the upper section. This variation from the RDD allows one to ask whether a student at the median would be better off if she is assigned to the upper section. We will consider both types of variations in this empirical illustration.

4.1 Data and Variables

First we read the data set of Duflo et al. (2011)[5] and define some key variables.

```
data("ddk_2011")
trk <- ddk_2011$tracking
con <- ddk_2011$etpteacher
hgh <- ddk_2011$highstream
yy  <- ddk_2011$ts_std
xx  <- ddk_2011$percentile
```

[4]This function is a S3 method for class `qte`.

[5]The dataset is included in the package. For additional details, please refer to the package manual.

There are three indicator variables; `trk` takes 1 for tracking schools, `con` takes 1 for students assigned to a contract teacher, `hgh` takes 1 for students assigned to high-achieving sections (if in tracking schools). The variable `yy` is the endline test scores normalized by the mean and standard deviation of non-tracking schools and `xx` is student's percentile rank from the baseline test. Because the outcome variable is normalized, the unit of the effect is a standard deviation of the endline test score (of non-tracking schools).

4.2 RDD

This section focuses on tracking schools and presents results from the RD design. We examine students near the median of the baseline test, and compare marginal students who just made the upper section to those who narrowly missed it. The dependent variable and the running variable (`yc` and `xc` below) include students in tracking schools only. The cutoff point is the median of the baseline percentile distribution ($x_0 = 50$), and the treatment indicator (`dc` below) takes 1 if students are in high achieving sections.

```
yc <- yy[trk==1]
xc <- xx[trk==1]
dc <- hgh[trk==1]
x0 <- 50
tlevel <- 1:9/10
hh <- 20
```

The last two lines set the values of two parameters; `tlevel` defines the quantile range $\mathcal{T} = [0.1, 0.9]$ and `hh` is the bandwidth at the median. More details on bandwidth selection will be discussed later.

QTE from RDD without covariates

In `rd.qte()`, when `x` includes the running variable only and `z0` is unspecified, one can estimate quantile effects at `x0` without covariate.

```
A <- rd.qte(y=yc,x=xc,d=dc,x0,z0=NULL,tau=tlevel,bdw=hh,bias=1)
A2 <- summary(A,alpha=0.1)
A2
```

```
                                QTE
-------------------------------------------------------------------------
            Bias cor.    Pointwise         Uniform
   Tau         Est.     Robust S.E.    90% Conf. Band
    0.1      -0.104       0.137      -0.427       0.218
    0.2      -0.001       0.139      -0.327       0.324
    0.3      -0.068       0.146      -0.410       0.274
    0.4      -0.074       0.148      -0.423       0.274
    0.5      -0.157       0.173      -0.564       0.250
    0.6      -0.069       0.211      -0.565       0.426
    0.7      -0.020       0.262      -0.636       0.597
    0.8      -0.023       0.309      -0.749       0.702
    0.9      -0.003       0.252      -0.595       0.590
```

The outcome table shows some essential elements of the analysis including point estimates, standard errors, and uniform confidence bands. Because the bias option is activated, '`bias=1`', the table reports the bias corrected point estimate and the robust standard error and robust uniform band. If '`bias=0`', one would obtain QTE estimates and uniform bands without the bias correction. Because '`alpha=0.1`', a 90% uniform confidence band is reported. If '`alpha=0.05`', one would get a 95% uniform confidence band.

The estimated quantile effects are small in magnitude (the maximum effect is -0.157 standard deviation when $\tau = 0.5$) and the uniform confidence band includes zero throughout the quantile range. This confirms a finding in Duflo et al. (2011) who concluded that

"the median student in tracking schools scores similarly whether assigned to the upper or lower section." The QTE estimate provides even stronger evidence that not only on average but also on the entire endline score distribution, students near the median of the initial test scores fare similarly regardless of whether they were assigned to the upper or lower ability section.

To examine the shape of the effect graphically, `plot.qte()` function can be used to produce QTE plots along with uniform confidence bands as in Figure 1.

```
y.text <- "test scores"
m.text <- "Effects of Assignment to Lower vs. Upper sections"
plot(A2,ytext=y.text,mtext=m.text)
```

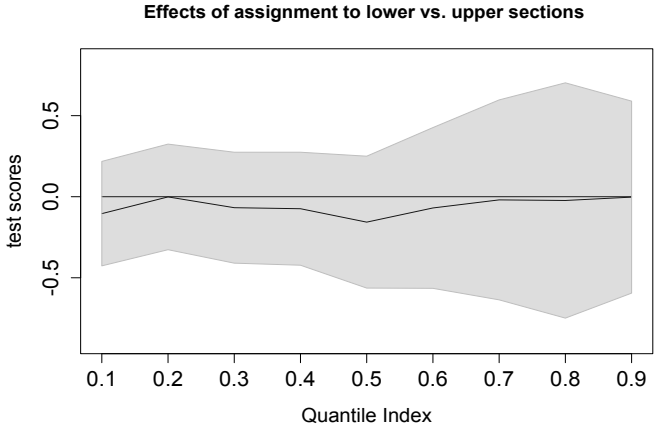


Figure 1: QTE Estimates from RDD

It is of interest to examine the conditional quantile functions from two sides of the cutoff. The function `plot.qte()` can make such plots with the option 'ptype=2'. The inputs for conditional quantile plots include estimates for two conditional quantile functions, `qp` and `qm`, and their uniform bands, `bandp` and `bandm`. These outputs are produced by `summary.qte()` and already saved in `A2` as the next example shows. In Figure 2, we put two figures on top of each other. To produce separate figures, simply apply the command twice, once for each side of the cutoff.

```
y.text <- "test scores"
m.text <- "Conditional quantile functions"
sub.text <- c("Upper section","Lower section")
plot(A2,ptype=2,ytext=y.text,mtext=m.text,subtext=sub.text)
```

To test the (lack of) effect, one can use the `rdq.test()` function. When 'alpha=c(0.1,0.05)', it provides critical values at the 10% and 5% levels. The `type` option determines the type of tests to be conducted. The lines below set 'type=c(1,2,3,4)', leading to tests for all four hypotheses including significance, homogeneity, and positive and negative dominance.

```
B <- rdq.test(y=yc,x=xc,d=dc,x0,z0=NULL,tau=tlevel,bdw=hh,bias=1,alpha=c(0.1,0.05),
+             type=c(1,2,3,4))

                    Testing hypotheses on quantile process
--------------------------------------------------------------------------------
NULL Hypthoesis                              test stat.  critical value   p value
                                                           10%       5%
================================================================================
Significance: QTE(tau|x,z)=0 for all taus      0.86     2.36     2.64     0.94
Homogeneity: QTE(tau|x,z) is constant          0.52     1.90     2.13     0.98
Dominance: QTE(tau|x,z)>=0 for all taus        0.86     2.10     2.41     0.57
Dominance: QTE(tau|x,z)<=0 for all taus        0.00     2.06     2.29     1.00
```

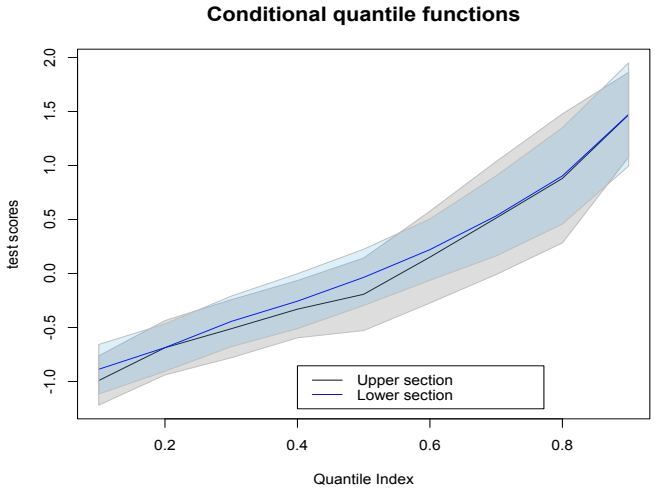


Figure 2: Conditional Quantile Functions

The outcome table displays the null hypotheses to be tested, test statistics, critical values, and p-values. All four tests indicate that QTEs are likely to be zero over the entire quantile range.

The empirical evidence from the RDD indicates that there is no difference in endline achievement between marginal students regardless of whether they were assigned to the upper or lower section. Because students in the upper section had much higher achieving peers, this implies that there may be a factor that offsets the positive peer effect. One possibility is that tracking may allow teachers to adjust their instruction to students' needs. Exploring this potential channel, Duflo et al. (2011) documented evidence that teachers had incentives to focus on the students at the top of the distribution. If this is the case, the median students from the bottom section may get benefits from the instruction that better matches their need.

The bandwidth (at the median) can be estimated as follows.

```
C <- rdq.bandwidth(y=yc,x=xc,d=dc,x0,z0=NULL,cv=1,val=(5:20))
C

                    Selected Bandwidths
------------------------------------------------------------
Method                                        Values
============================================================
Cross Validation                                20
MSE Optimal                                    16.3         16.3
```

Because the cross-validation option is on, 'cv=1', the table reports both CV and MSE optimal bandwidths. The candidate values are $\{5, 6, \ldots, 20\}$ for cross-validation, as given by 'val=(5:20)'. We have used the bandwidth 20 because it was selected by the cross validation method. If the sample is very large, computing the CV bandwidth may take a long time. In such a case, set 'cv=0' and use the MSE optimal bandwidth at least for the initial stage of data exploration.

QTE from RDD with covariates

To see heterogeneity in the effect of tracking, one can include additional covariates. This section compares effects of tracking for boys and girls. The covariate zc is a female dummy and the evaluation point z_0 is set by 'z.eval = c(0,1)'. The order of display in the outcome table is the same as the order of the group in z_0.

```
zc <- ddk_2011$girl[trk==1]
z.eval <- c(0,1)
A <- rd.qte(y=yc,x=cbind(xc,zc),d=dc,x0,z0=z.eval,tau=tlevel,bdw=hh,bias=1)
A2 <- summary(A,alpha=0.1)
```

```
                                QTE
-------------------------------------------------------------------------
             Bias cor.     Pointwise          Uniform
    Tau         Est.     Robust S.E.    90% Conf. Band
-------------------------------------------------------------------------
 Group-1   (boys)
     0.1       0.295       0.187      -0.154       0.743
     0.2       0.090       0.224      -0.445       0.625
     0.3       0.063       0.207      -0.432       0.559
     0.4      -0.026       0.232      -0.581       0.528
     0.5       0.031       0.275      -0.628       0.689
     0.6       0.353       0.333      -0.443       1.150
     0.7       0.597       0.369      -0.286       1.481
     0.8       0.160       0.466      -0.955       1.275
     0.9       0.159       0.416      -0.835       1.154
-------------------------------------------------------------------------
 Group-2   (girls)
     0.1      -0.406       0.141      -0.737      -0.074
     0.2      -0.161       0.195      -0.620       0.297
     0.3      -0.100       0.224      -0.626       0.426
     0.4      -0.233       0.241      -0.798       0.332
     0.5      -0.475       0.270      -1.108       0.158
     0.6      -0.291       0.291      -0.976       0.393
     0.7      -0.158       0.363      -1.010       0.695
     0.8      -0.236       0.433      -1.253       0.781
     0.9       0.000       0.322      -0.756       0.756
```

Because '`z.eval <-c(0,1)`' and $z_0 = 0$ means boys, the outcome table shows results for boys first (shown as Group-1) and girls later (Group-2). For boys, the quantile effects of being in the upper ability section is positive but insignificant. For girls, the effects are mostly negative and insignificant. But at the bottom of the outcome distribution, when $\tau = 0.1$, the negative effect turns to be significant. To see the group-wise difference graphically, one can draw QTE plots as follows.

```
y.text <- "test scores"
m.text <- c("Boys","Girls")
plot(A2,ytext=y.text,mtext=m.text)
```

The plot clearly shows that tracking has a positive but insignificant effect for marginal male students, but the effect is negative for marginal female students and significantly so at the left tail. To explore further, it will be useful to draw plots for the conditional quantile functions separately for each group.

```
y.text <- "test scores"
m.text <- c("Boys","Girls")
sub.text <- c("Upper section","Lower section")
plot(A2,ptype=2,ytext=y.text,mtext=m.text,subtext=sub.text)
```

The plot is omitted to save space, but it shows that for girls, the conditional quantile function of endline test scores for the upper section is consistently below that of the lower section, and the difference is largest at the left tail.

Tests for hypotheses for each group can be done as well.

```
B <- rdq.test(y=yc,x=cbind(xc,zc),d=dc,x0,z0=z.eval,tau=tlevel,bdw=hh,bias=1,
+               alpha=c(0.1,0.05),type=c(1,2,3,4))
```

Figure 3: QTE Estimates from RDD by Student Gender

The results indicate that it is not possible to reject hypotheses that the QTE is consistently zero for boys, but there is evidence that the effects can be negative for girls. For female students the null hypothesis of no effect (significance) and positive uniform effect (dominance) are rejected at the 5% confidence level. The bandwidth can be selected as well.

```
C <- rdq.bandwidth(y=yc,x=cbind(xc,zc),d=dc,x0,z0=z.eval,cv=1,val=(5:20))
```

```
                    Selected Bandwidths
-----------------------------------------------------------
Method                                        Values
===========================================================
Cross Validation                                19
MSE Optimal,Group-1                            14.3         14.2
MSE Optimal,Group-2                            17.9         20.0
```

When users would like to see the effect of the bias correction on point estimates and uniform bands, it will be convenient to use the function `rdq.band()`. Its options are the same as those in `rd.qte()`. The difference is that it implements estimation with and without bias correction and presents results side by side.

```
D <- rdq.band(y=yc,x=cbind(xc,zc),d=dc,x0,z0=z.eval,tau=tlevel,bdw=hh,alpha=0.1)
```

```
                        QTE and Uniform Bands
-------------------------------------------------------------------------
                    Bias cor.          90% Uniform Conf. Band
   Tau        Est.      Est.       Non-robust            Robust
-------------------------------------------------------------------------
 Group-1  (boys)
     0.1     0.118     0.295    -0.194     0.430    -0.123     0.712
     0.2     0.014     0.090    -0.366     0.394    -0.415     0.596
     0.3     0.022     0.063    -0.326     0.370    -0.415     0.542
     0.4    -0.023    -0.026    -0.425     0.379    -0.595     0.542
     0.5     0.044     0.031    -0.433     0.522    -0.645     0.706
     0.6     0.093     0.353    -0.484     0.670    -0.469     1.176
     0.7     0.194     0.597    -0.451     0.839    -0.288     1.482
     0.8     0.096     0.160    -0.684     0.876    -0.923     1.243
     0.9     0.267     0.159    -0.431     0.965    -0.800     1.118
-------------------------------------------------------------------------
 Group-2  (girls)
     0.1    -0.204    -0.406    -0.464     0.056    -0.736    -0.075
     0.2    -0.111    -0.161    -0.460     0.238    -0.619     0.296
```

```
0.3   -0.128   -0.100   -0.542    0.286   -0.639    0.439
0.4   -0.186   -0.233   -0.652    0.281   -0.823    0.358
0.5   -0.335   -0.475   -0.839    0.170   -1.117    0.167
0.6   -0.136   -0.291   -0.685    0.413   -0.988    0.406
0.7   -0.142   -0.158   -0.814    0.530   -1.029    0.714
0.8   -0.148   -0.236   -0.938    0.642   -1.275    0.803
0.9    0.085    0.000   -0.522    0.692   -0.783    0.783
```

Without bias correction, the effect for girls at the 10th percentile is no longer statistically significant, as the estimate is smaller. Otherwise, the conclusion does not change.

4.3 RCT

The schools in the sample were randomly assigned to tracking and non-tracking schools. Using this random variation, one can compare students between tracking and non-tracking schools to examine the impact of tracking on the entire student population. The tracking variable `trk` is the treatment assignment d for the RCT. We use the baseline test score as the nonparametric component x. Because it is highly predictive of the endline test score, it is natural to examine the effects of tracking separately for groups at various points of the initial performance distribution. To examine the heterogeneity in effects, we use student and teacher characteristics as covariates z. We continue to use '`hh=20`' for the bandwidth at the median. Users can change the bandwidth value and examine the robustness of results.

QTE from RCT without covariates

```
dr <- trk
A <- rd.qte(y=yy,x=xx,d=dr,x0=50,z0=NULL,tau=tlevel,bdw=hh,bias=1)
A2 <- summary(A,alpha=0.1)
```

```
                                QTE
------------------------------------------------------------------------
            Bias cor.    Pointwise         Uniform
   Tau         Est.     Robust S.E.    90% Conf. Band
    0.1       0.234       0.051       0.115       0.354
    0.2       0.227       0.063       0.079       0.374
    0.3       0.293       0.064       0.143       0.443
    0.4       0.278       0.068       0.119       0.437
    0.5       0.304       0.075       0.128       0.480
    0.6       0.308       0.086       0.106       0.509
    0.7       0.308       0.106       0.060       0.556
    0.8       0.351       0.135       0.034       0.668
    0.9       0.280       0.139      -0.044       0.605
```

To estimate the QTE without a covariate, set '`z0=NULL`'. By letting '`x0=50`', we focus on the median students from the initial achievement distribution and estimate the effect of assigning them to tracking or non-tracking schools. For this group, test scores were between 0.227 ($\tau = 0.2$) and 0.351 ($\tau = 0.8$) standard deviations higher in tracking schools than in non-tracking schools when measured by quantile effects. The size of the effect is notably higher than the average unconditional effect, 0.14 standard deviations higher in tracking schools, as reported in Duflo et al. (2011). This difference suggests that the effect of tracking may be quite different for high and low-achieving students.

To check this, one can change the value of x_0. Set '`x0 = 20`', then one can study the effect of tracking for the initially low achieving students.

```
A <- rd.qte(y=yy,x=xx,d=dr,x0=20,z0=NULL,tau=tlevel,bdw=hh,bias=1)
```

The quantile effects for this group of low achieving students from the baseline test are indeed much smaller. Test scores were up to 0.179 standard deviation higher in tracking schools than in non-tracking schools. Next, set '`x0 = 80`' and examine the initially high achieving students.

```
A <- rd.qte(y=yy,x=xx,d=dr,x0=80,z0=NULL,tau=tlevel,bdw=hh,bias=1)
```

The estimated effects are larger than those for $x_0 = 20$ but still smaller than those for $x_0 = 50$. The results so far suggest that the impact of tracking from the RCT is the strongest around the median of the initial achievement distribution. This finding is in harmony with Figure 3 in Duflo et al. (2011).

By student's gender & teacher's type

There are two types of teachers: regular teachers who are civil-servants and contract teachers who are hired on short-term contracts by local school committees. The contract teachers have much stronger incentives to teach well because a good record of performance may lead them to a regular teaching job. To examine heterogeneous effects across groups defined by student's gender and teacher's type, define the resulting four groups as follows. The covariates z include indicators for student's gender and for contract teacher, and $z_0 = ((0,0),(1,0),(0,1),(1,1))'$. This specification means that Group-1 consists of boys taught by regular teachers, while the remaining three groups are defined accordingly

```
zw <- cbind(ddk_2011$girl,con)
z.eval <- cbind(rep(c(0,1),2),rep(c(0,1),each=2))
A <- rd.qte(y=yy,x=cbind(xx,zw),d=dr,x0=50,z0=z.eval,tau=tlevel,bdw=hh,bias=1)
A2 <- summary(A,alpha=0.1)
```

An outcome table with four groups is not easy to read. It will be easier to examine results visually by QTE plots as in Figure 4.

```
y.text <- "test scores"
m.text <- c("boys & regular teachers","girls & regular teachers",
+              "boys & contract teachers","girls & contract teachers")
plot(A2,ytext=y.text,mtext=m.text)
```

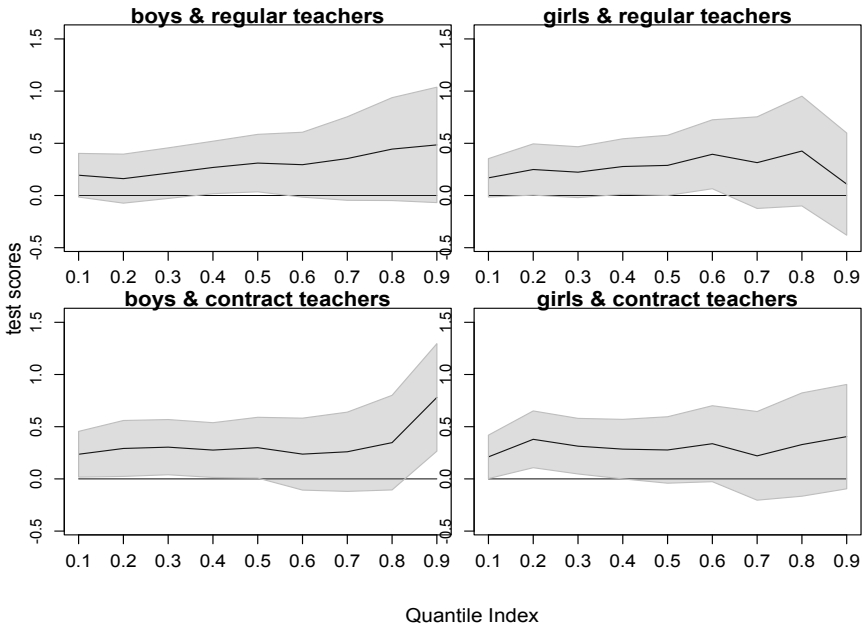


Figure 4: QTE Estimates from RCT by Student Gender and Teacher Type

The biggest effect can be found for boys taught by contract teachers.

By age of students

Students in the sample differ greatly in age. The average student at the endline test is 9.3 years old. But the age of the middle ninety percent of students ranges from 7 to 12 years at the time of the test. Below we compare the effects across four age groups.

```
zw <- ddk_2011$agetest
z.eval <- c(7,9,10,11)
A <- rd.qte(y=yy,x=cbind(xx,zw),d=dr,x0=50,z0=z.eval,tau=tlevel,bdw=hh,bias=1)
A2 <- summary(A,alpha=0.1)
```

The effects by age groups are displayed in Figure 5. The bigger effects can be found for younger students. The maximum quantile effect of tracking at age seven is 0.602 standard deviation, while at age twelve it is 0.285 standard deviation.

```
y.text <- "test scores"
m.text <- c("age 7","age 9","age 10","age 12")
plot(A2,ytext=y.text,mtext=m.text)
```

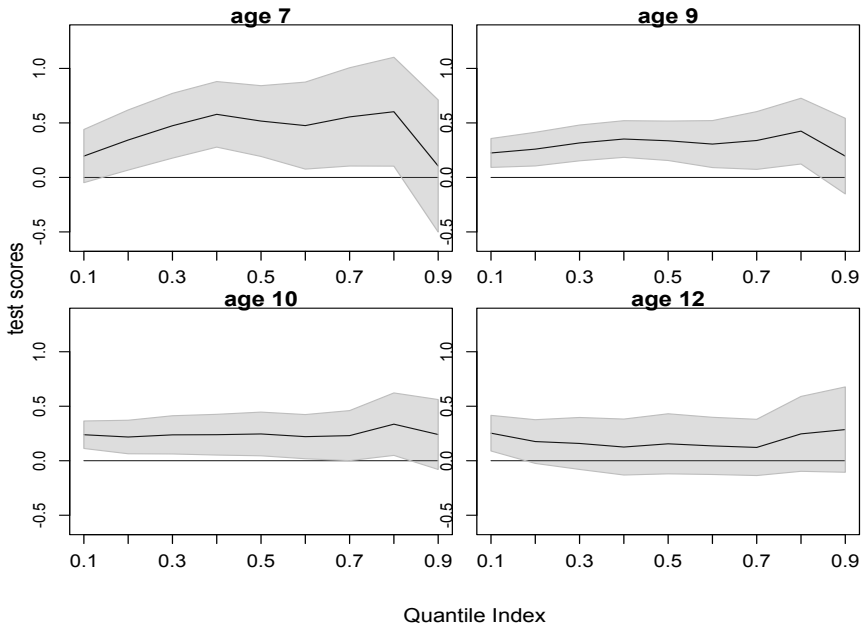


Figure 5: QTE Estimates from RCT, by Student Age

Testing hypotheses for each age group can also be conducted as follows.

```
B <- rdq.test(y=yy,x=cbind(xx,zw),d=dr,x0=50,z0=z.eval,tau=tlevel,bdw=hh,
+                bias=1,alpha=c(0.1,0.05),type=c(1,2,3,4))
```

Test results indicate that QTEs are significant and uniformly positive for all four age groups.

5 Conclusion

QTE.RD is a comprehensive R package designed for analyzing quantile treatment effects under sharp RD designs. The package enables researchers to test, estimate, and conduct uniform inference on quantile treatment effects (QTEs), incorporating covariates, implementing robust bias correction, selecting the bandwidth, and plotting the estimation results, all in the same place. To our knowledge, this is the first comprehensive R package for estimating quantile effects under RD designs.

The package can be expanded in two directions to encompass a greater range of empirical applications. The first is to accommodate time-series RD designs (Hausman and Rapson,

2018). Developing valid inference results for time-series data would be the first step in achieving this goal. The second is to allow for more than a few covariates in the model, which might require incorporating penalization or some covariate selection methods to guide model specification. We intend to pursue these directions and expand the capacity of this package accordingly.

Acknowledgments

Jungmo Yoon acknowledges financial support from the National Research Foundation of Korea (NRF-2020S1A5A8042547).

References

S. Calonico, M. D. Cattaneo, and R. Titiunik. Robust nonparametric confidence intervals for regression-discontinuity designs. *Econometrica*, 82(6):2295–2326, 2014. URL https://doi.org/10.3982/ECTA11757. [p73]

S. Calonico, M. D. Cattaneo, M. H. Farrell, and R. Titiunik. Regression discontinuity designs using covariates. *The Review of Economics and Statistics*, 101(3):442–451, 2019. URL https://doi.org/10.1162/rest_a_00760. [p74]

H. D. Chiang, Y.-C. Hsu, and Y. Sasaki. Robust uniform inference for quantile treatment effects in regression discontinuity designs. *Journal of Econometrics*, 211(2):589 – 618, 2019. URL https://doi.org/10.1016/j.jeconom.2019.03.006. [p71]

K. Doksum. Empirical probability plots and statistical inference for nonlinear models in the two-sample case. *The Annals of Statistics*, 2(2):267–277, 1974. ISSN 00905364. URL https://doi.org/10.1214/aos/1176342662. [p69]

E. Duflo, P. Dupas, and M. Kremer. Peer effects, teacher incentives, and the impact of tracking: Evidence from a randomized evaluation in Kenya. *American Economic Review*, 101(5):1739–74, August 2011. URL https://doi.org/10.1257/aer.101.5.1739. [p71, 72, 78, 79, 81, 84, 85]

B. R. Frandsen, M. Frölich, and B. Melly. Quantile treatment effects in the regression discontinuity design. *Journal of Econometrics*, 168(2):382–395, 2012. ISSN 0304-4076. URL http://dx.doi.org/10.1016/j.jeconom.2012.02.004. [p71]

C. Hausman and D. S. Rapson. Regression discontinuity in time: Considerations for empirical applications. *Annual Review of Resource Economics*, 10(1):533–552, 2018. doi: 10.1146/annurev-resource-121517-033306. URL https://doi.org/10.1146/annurev-resource-121517-033306. [p86]

G. W. Imbens and T. Lemieux. Regression discontinuity designs: A guide to practice. *Journal of Econometrics*, 142(2):615–635, 2008. ISSN 0304-4076. URL http://dx.doi.org/10.1016/j.jeconom.2007.05.001. [p78]

R. Koenker. *Quantile Regression*. Cambridge University Press, 2005. URL https://doi.org/10.1017/CBO9780511754098. [p73]

E. L. Lehmann. *Nonparametrics: statistical methods based on ranks*. Holden-Day, San Francisco., 1975. URL https://doi.org/10.2307/2344536. [p69]

Z. Qu and J. Yoon. Uniform inference on quantile effects under sharp regression discontinuity designs. *Journal of Business and Economic Statistics*, 37(4):625–647, 2019. URL https://doi.org/10.1080/07350015.2017.1407323. [p69]

Z. Qu, J. Yoon, and P. Perron. Inference on conditional quantile processes in partially linear models with applications to the impact of unemployment benefits. *The Review of Economics and Statistics*, pages 521–541, 03 2024. URL https://doi.org/10.1162/rest_a_01168. [p69, 70, 72, 73, 74, 75]

D. L. Thistlethwaite and D. T. Campbell. Regression-discontinuity analysis: An alternative to the ex post facto experiment. *Journal of Educational psychology*, 51(6):309–317, 1960. URL https://doi.org/10.1037/h0044319. [p69]

K. Yu and M. C. Jones. Local linear quantile regression. *Journal of the American Statistical Association*, 93(441):228–237, 1998. ISSN 01621459. URL https://doi.org/10.2307/2669619. [p73, 74]

Zhongjun Qu
Department of Economics
Boston University
USA
qu@bu.edu

Jungmo Yoon
College of Economics and Finance
Hanyang University
Korea, South
ORCID: 0000-0002-6686-4739
jmyoon@hanyang.ac.kr

BoundaryStats: An R Package to Calculate Boundary Overlap Statistics

by Amy Luo

Abstract Ecologists and epidemiologists frequently rely on spatially distributed data. Studies in these fields may concern geographic boundaries, as environmental variation can determine the spatial distribution of organismal traits or diseases. In such cases, environmental boundaries produce coincident geographic boundaries in, for example, disease prevalence. Boundary analysis can be used to investigate the co-occurrence of organismal trait or disease boundaries and underlying environmental boundaries. Within boundary analysis, boundary overlap statistics test for the presence of significant geographic boundaries and spatial associations between the boundaries of two variables. There is one pre-existing implementation of boundary overlap statistics, though it is only on Windows and ESRI ArcView, limiting the availability of boundary overlap statistics to researchers. I have created BoundaryStats—an R package available on CRAN—that implements boundary overlap statistics. BoundaryStats is the first open-source, cross-platform implementation of these statistical methods, making the statistics more widely accessible to researchers.

1 Introduction

Geographic boundaries are an intrinsic feature of spatial ecology and epidemiology, as the relationships between an underlying environmental variable and organismal traits or disease prevalence often produce coincident geographic boundaries. Boundaries are areas in which spatially distributed variables (e.g., bird plumage coloration, disease prevalence, annual rainfall) rapidly change over a narrow geographic space. They can also represent edges or discontinuities (e.g., neighborhood edges, ecotype boundaries). Boundary zones themselves may be of interest; for example, the temporal boundary dynamics between ecotypes can provide insight into the factors that produce mosaic landscapes (Bowman et al., 2023).

Boundary analysis involves the analysis of spatial boundaries to answer questions about values within a bounded area, patterns of change across a landscape, and associations between the spatial patterns of multiple variables (Jacquez, 2010). Within boundary analysis, boundary overlap statistics can be used to test the association between the boundaries of two spatially distributed variables (Jacquez et al., 2000). These statistics fall within two categories: boundary statistics (i.e., tests for the presence of cohesive boundaries) and boundary overlap statistics (i.e., tests for spatial association between boundaries). BoundaryStats runs two boundary statistics and three boundary overlap statistics, as initially described in Jacquez (1995) and Fortin et al. (1996). The boundary statistics are (1) the length of the longest boundary and (2) the number of cohesive boundaries on the landscape (Fortin et al., 1996). The boundary overlap statistics are (1) the amount of direct overlap between boundaries in variables A and B, (2) the mean minimum distance between boundaries in A and B, and (3) the mean minimum distance from boundaries in A to boundaries in B (Fortin et al., 1996; Jacquez, 1995).

While other spatial statistics account for complications like spatial autocorrelation and environmental heterogeneity (Wagner and Fortin, 2005), boundary overlap statistics can uniquely leverage geographic discontinuities to answer spatial questions. By identifying significant cohesive boundaries, researchers can delineate relevant geographic sampling units (e.g., populations as conservation units for a species or human communities with increased disease risk) (Jacquez, 2010). Associations between the spatial boundaries of two variables can be useful in assessing the extent to which an underlying landscape variable drives the spatial distribution of a dependent variable. For example, ecologists are often interested in whether landscape-level ecological boundaries limit gene flow, thereby producing population structure; if the putative ecological boundary is limiting gene flow, one

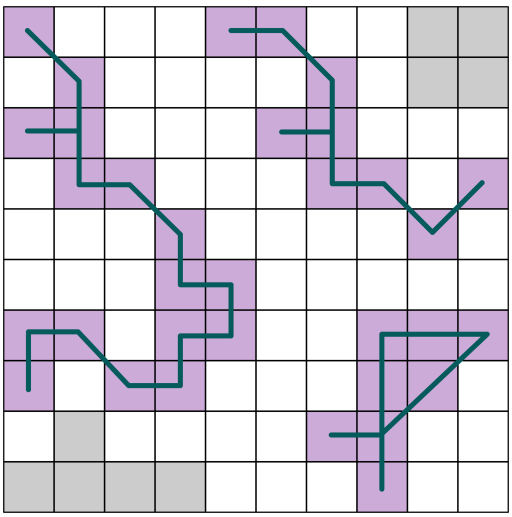


Figure 1: Example boundary subgraphs. Gray cells are missing values, white cells are non-boundary cells, and purple cells are boundary elements. Subgraphs are each represented using a line connecting all the boundary element cells that comprise them.

would expect concordant geographic boundaries in the ecological variable and population structure (Tarroso et al., 2014; Wagner and Fortin, 2013). The presence of boundaries can similarly limit the distribution of taxonomically similar species (Polakowska et al., 2012). In an epidemiological context, this may look like neighborhood effects on public health outcomes, including COVID-19 infection risk (Hong et al., 2021; van Ham et al., 2012) or spatial relationships between high pollutant density and increased disease risk (Adimalla et al., 2020; Waller et al., 1992).

Currently, there is at least one tool that has implemented boundary overlap statistics: GEM, which was released as an extension of ESRI ArcView and a standalone Windows package. GEM is not available as a cross-platform, free, and open-source software, thereby limiting its accessibility to researchers. **BoundaryStats** implements boundary and boundary overlap statistics in R. It is available to download on CRAN, making the tools more accessible for researchers, especially in epidemiology and spatial ecology.

2 Boundary definitions

In this framework, we classify raster cells into a pseudobinary: boundary elements (1), non-boundary cells (0), or missing data (NA). For categorical variables, the algorithm for identifying boundary elements is simple: if any of a cell's neighbors—based on the queen criterion (i.e., eight neighboring cells, including diagonal cells)—belongs to a different category, the cell is classified as a boundary element. For quantitative variables, boundaries exist where two different but internally homogeneous areas neighbor one another (e.g., an area with very low values meets an area with very high values), but these boundaries are generally fuzzy; they represent steep transitions that can still occur over the width of multiple cells. Therefore, we define boundary elements as the cells with the highest boundary intensity values, with the threshold set by the user. Boundary intensity values can be calculated through several different methods, including the magnitude of change across cells or the probability that cells belong to each neighboring spatial group (see details below). Boundaries are defined here as subgraphs of boundary elements, or contiguous cells that are all marked as boundary elements (Figure 1).

Boundary intensities for variables with landscape-level patterns can be calculated in a number of ways, including: lattice- and triangulation-wombling (Fortin et al., 1996; Jacquez, 1995; St-Louis et al., 2004; Strydom and Poisot, 2023), fuzzy set modeling (Jacquez, 1995), Monmonier's algorithm (Manni et al., 2004), spatial Bayesian clustering (Caye et al., 2016;

Safner et al., 2011), agglomeration of inner lines (Wei and Larsen, 2019), and removal of outer lines (Wei and Larsen, 2019). For quantitative variables, BoundaryStats will accept raster objects with the spatial variable directly or boundary intensity values calculated from these or other methods. If given boundary intensity values, boundary elements will be classified directly using the top percent of values. The default proportion of values is 0.2, though this threshold can be changed by the user. When given the variable directly, BoundaryStats will use the Sobel-Feldman operator to calculate the boundary intensity. In accepting either the variables or boundary intensities, there is flexibility for users to define boundaries using relevant metrics for their data.

The Sobel-Feldman operator is commonly used for edge detection in computer vision applications. It approximates the magnitude of the partial derivative (i.e., rate of change) across each cell using the following kernels:

$$G_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} * A \tag{1}$$

$$G_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} * A \tag{2}$$

where A is the input raster cell and its queen neighbors, and G_x and G_y are the rates of variable change in the horizontal or vertical directions, respectively. The boundary intensity value is the overall rate of change:

$$G = \sqrt{G_x^2 + G_y^2} \tag{3}$$

3 Statistics

BoundaryStats runs two boundary statistics and three boundary overlap statistics, as initially described in Jacquez (1995) and Fortin et al. (1996). Below, I describe these five statistics.

3.1 Number of subgraphs

The first boundary statistic is the number of subgraphs, which describes the number of boundaries on the landscape for a variable. In a raster of boundary elements, it is the number of unique subgraphs, or sets of contiguous boundary element cells (three subgraphs each in Figure 2, Panels A and B).

3.2 Longest subgraph

The other boundary statistic included here is the length of the longest subgraph, or boundary. The function calculates the longest length across each subgraph, then converts the length to distance based on the cell resolution and the projection of the raster. The length of the longest subgraph is then retained (Figure 2, Panel C).

3.3 Direct overlap

The direct overlap statistic is a count of the number of overlapping boundary elements of two variables, when the two raster objects are overlaid (Figure 2, Panel D).

3.4 Mean minimum distance between boundaries

This statistic describes the spatial proximity between boundaries of variables x and y, as defined by the mean distance to the nearest boundary element of the other variable. Spatial

relationships between boundaries may not result in direct overlap, so this statistic accounts for potential correlations in non-overlapping boundaries. For each boundary element in x, the function calculates the distance to the nearest boundary element in y, then repeats the inverse for each boundary element in y (Figure 2, Panels E and F). It then takes the mean of these minimum distances across all boundary elements in both raster objects:

$$O_{xy} = \frac{\sum_{i=1}^{N_x} \min(d_i) + \sum_{j=1}^{N_y} \min(d_j)}{N_x + N_y} \tag{4}$$

where i and j are boundary elements for variables x and y, respectively; $\min(d_i)$ is the minimum distance between boundary element i to a boundary element for y; $\min(d_j)$ is the minimum distance between boundary element j to a boundary element in x; and N_x and N_y are the number of boundary elements for x and y, respectively.

3.5 Mean minimum distance from boundary x to boundary y

This statistic describes the mean distance from boundary elements in x to the nearest boundary element in y. It is an indicator for whether the boundaries in x depend on y. The reciprocal nature of the previous statistic implies some reciprocity of effect, as opposed to the unidirectionality implicit here. For each boundary element in the raster for x, the function calculates the distance to the nearest boundary element of y, then takes the mean across all boundary elements in x (Figure 2, Panel E):

$$O_x = \frac{\sum_{i=1}^{N_x} \min(d_i)}{N_x} \tag{5}$$

4 Neutral models

In addition to calculating each statistic, BoundaryStats uses iterations of a neutral landscape model to determine whether the boundaries in the input landscape differ from a random landscape. Users select a neutral landscape model and number of iterations of that model to produce a null distribution of each statistic, based on the selected model and the structure of the input landscape. BoundaryStats implements three neutral landscape models: stochastic landscapes, Gaussian random fields, and modified random clusters. All three neutral landscape models draw some parameters from the original raster and simulate landscapes with similar parameters. Cells with missing values (i.e., NA values) will be ignored in all models.

The simplest neutral landscape model is complete stochasticity. While this model is not realistic—a complete lack of spatial autocorrelation is unlikely and may inflate the statistical significance of the observed data—we include it here as a complete null for users who are interested in a lack of spatial autocorrelation. This method takes all the cell values from the input raster and assigns each value to a random cell. Each cell in the simulated raster is assigned a value from the original dataset, with no replacement of values. The simulated raster has the exact same values as the original raster, but values are randomly placed with no spatial autocorrelation.

The next neutral landscape model simulates a Gaussian random field with the same degree of spatial autocorrelation as the input raster. It is suited for continuous or discrete quantitative variables. This method calculates the range of autocorrelation in the original raster by fitting a variogram using functions from **gstat** (Pebesma, 2004). The function then simulates a Gaussian random field with the same range of spatial autocorrelation, extent, and resolution as the input raster using methods from the **fields** package.

The modified random cluster model is an implementation of the method described by Saura and Martínez-Millan for simulating neutral landscapes for categorical variables (Saura

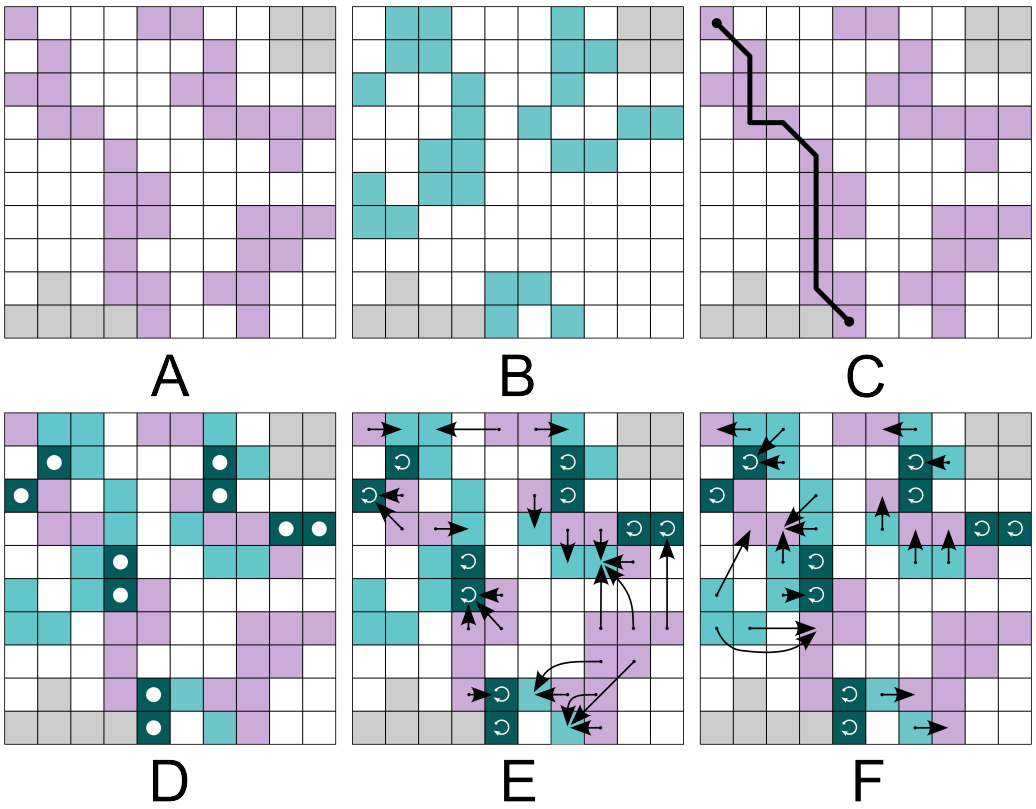


Figure 2: Example boundaries and statistics. (A) and (B) are boundary elements for hypothetical variables A and B. White cells are non-boundaries, gray are missing values, and purple or teal are boundary elements. (C) Length of the longest subgraph. (D) Produced by overlaying cells in A and B. Dark blue cells, highlighted by white dots, are where the boundary elements overlap. (E) For every boundary element for variable A, the nearest boundary element for variable B. Circular arrows indicate distance to self. (F) For every boundary element for B, the nearest boundary element for A.

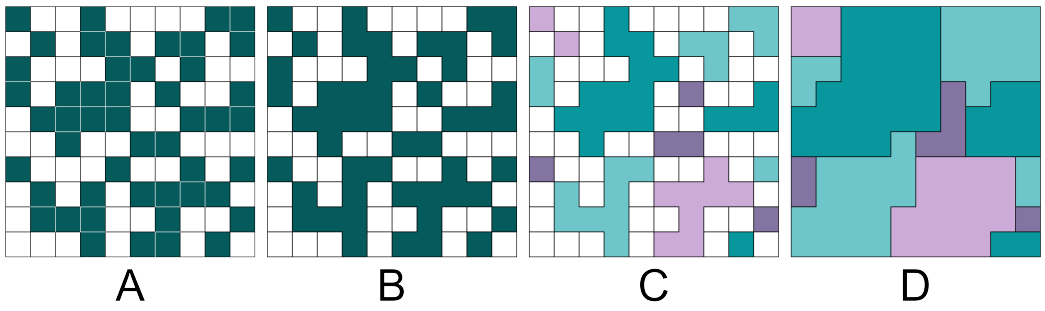


Figure 3: Modified random cluster procedure, adapted from Saura and Martínez-Millan 2000. (A) Percolated raster with p = 0.5. (B) Marked cells merged into clusters. (C) Clusters assigned a category. (D) Unmarked cells filled based on neighbors.

and Martínez-Millan, 2000). The first step is a percolated raster (Figure 3, Panel A). Each cell is assigned a value $0 \leq x \leq 1$ from a uniform distribution, and cells with values above a threshold probability p are marked. p is defined by the user, and higher values of p result in larger cluster sizes in the final simulated raster. Next, contiguous sets of marked cells are grouped into clusters, using the rook criterion (i.e., neighbors are the four edge-touching neighbors) (Figure 3, Panel B). Clusters are then assigned a category (Figure 3, Panel C). Categories from the input raster are chosen one at a time, and random clusters are assigned to that category. When the proportion of that category in the simulated raster reaches the proportion in the original raster, clusters are then assigned to the next category, until all the clusters are assigned. In the last step, the unmarked cells are categorized based on the most frequent category among their neighbors using the queen criterion (Figure 3, Panel D). If there is a tie between two categories, one of the tied categories is picked at random. If all neighbors are unassigned, a random category is picked; probabilities for each category are based on their proportions in the input raster.

5 Implementation and example

Data in this example are from Luo et al. (2024), in which the authors hypothesized that song divergence is facilitating genetic divergence in white-crowned sparrows through speciation by sexual selection. The data below are song boundaries and genetic admixture interpolations from the study. White-crowned sparrows sing different songs that vary across the landscape, and song boundaries are the spatial transitions between two song groups. The boundary intensities between song groups were estimated using GeoOrigins (Hulme-Beaman et al., 2020), based on the acoustic dissimilarity and spatial relationship between recorded songs. Genetic admixture is the estimated proportion of an individual's genetic material from different populations. In this case, there are two populations (north and south), so the admixture coefficients are the estimated proportions of genetic material from the northern population. The admixture coefficients of individual birds were estimated in fastSTRUCTURE (Raj et al., 2014), and the values were interpolated using local kriging in **gstat**.

5.1 Read in data

Read in raster data to **terra** (Hijmans, 2023) `SpatRaster` objects (Figure 4). The two objects need the same projection, extent, and resolution.

```
library(terra)
library(magrittr)

songs <- rast('data/2010_2022_song_boundaries.asc')
genetic <- rast('data/genetic_interpolation.asc') %>%
```

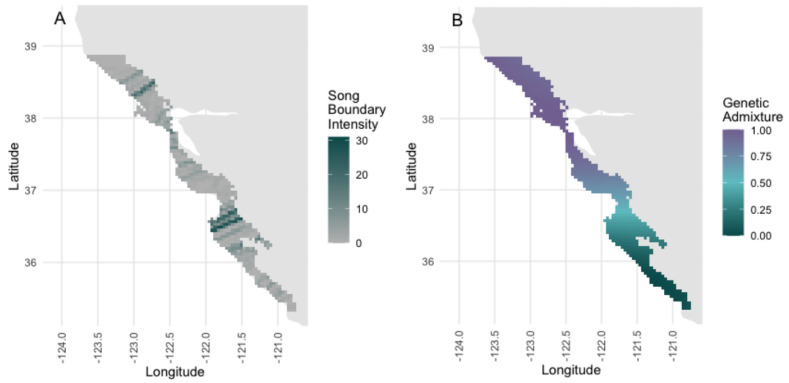


Figure 4: Maps of (A) song boundary intensity and (B) genetic admixture between two populations.

```
  resample(., songs)
songs <- crop(songs, genetic) %>%
  mask(., genetic)
```

5.2 Calculate spatial boundaries for variables

The first step of the analysis is to define which cells are boundary elements (i.e., part of a boundary) using the `define_boundary` function. By default, the function takes quantitative variables (`cat = FALSE`). For quantitative variables, boundary intensity can be calculated however the user chooses; if the input raster already contains boundary intensities, the argument `calculate_intensity` should be set to `FALSE` (default). Users can also set calculate_intensity to `TRUE` to use the Sobel-Feldman operator to calculate the boundary intensities.

The song raster already contains boundary intensity values, from which boundary elements can be directly determined. But the values in the genetic raster are the trait data, so boundary intensity needs to be calculated from the genetic admixture coefficient values (`calculate_intensity = TRUE`).

```
library(BoundaryStats)

song_boundaries <- define_boundary(songs)
genetic_boundaries <- define_boundary(genetic, calculate_intensity = TRUE)
```

5.3 Plot boundary overlap

This optional step is to visualize where the boundaries of the two variables are overlapping using `plot_boundary` (Figure 5). The function is a wrapper function for `ggplot` from **ggplot2** (Wickham, 2016), and the colors and trait names can optionally be customized. If `output_raster` is `TRUE` (default is `FALSE`), then the function will return a `SpatRaster` object with one layer that includes boundary elements for each trait and where the boundary elements overlap.

```
plot_boundary(genetic_boundaries, song_boundaries, trait_names = c("Genetic", "Song"))
```

5.4 Create null distributions for statistics

For both boundary statistics, use the function `boundary_null_distrib`. For the three overlap statistics, use the function `overlap_null_distrib`. Both functions simulate random

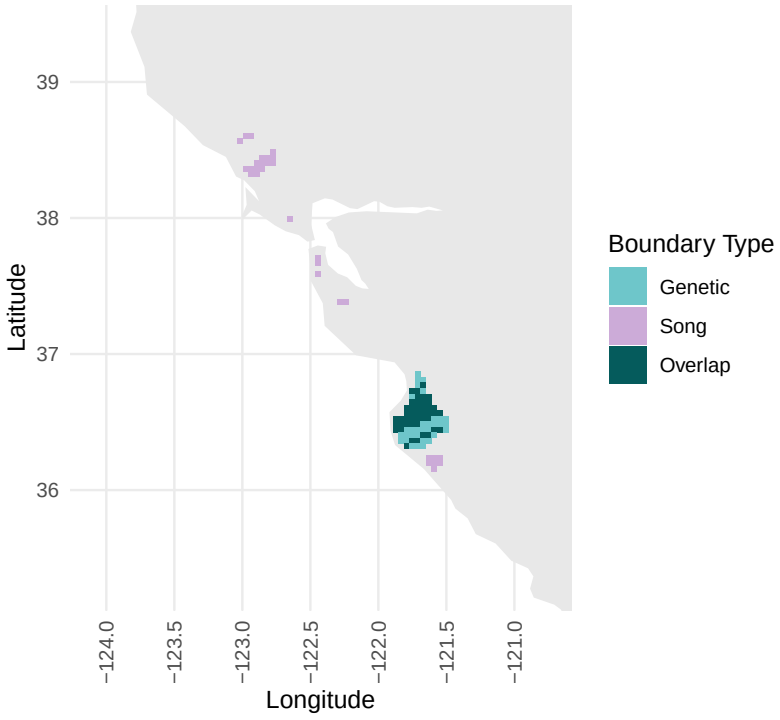


Figure 5: Output of the plot_boundary function.

iterations of a raster based on the specified neutral landscape model and input data. Statistics are calculated for each iteration, and custom null probability distributions are calculated based on the iterations. The resulting objects will be used for the statistical tests described in the section below.

Both functions take the `SpatRaster` object(s), a neutral landscape model, and the number of iterations. Further arguments may be required, depending on these arguments. For `overlap_null_distrib`, separate models can be specified for the two variables, and the variable in the first argument is assumed to depend on the variable in the second argument. The argument `rand_both` specifies whether the function should simulate random landscapes for the second `SpatRaster` object; since some hypotheses assume x depends on a specific underlying distribution of boundaries in y, users can choose to keep boundaries for y static for each iteration. For this example, the genetic boundary is hypothesized to depend on song boundaries. Therefore, the `SpatRaster` object containing the genetic admixture interpolation is the first argument, and I keep the song boundaries static (`rand_both = FALSE`).

```
song_boundary_null <- boundary_null_distrib(songs, calculate_intensity = FALSE, cat = FALSE,
    n_iterations = 100, threshold = 0.2, model = "gaussian")
genetic_boundary_null <- boundary_null_distrib(genetic, calculate_intensity = TRUE,
    cat = FALSE, n_iterations = 100, threshold = 0.2, model = "gaussian")

boundary_overlap_null <- overlap_null_distrib(genetic, songs, rand_both = FALSE,
   n_iterations = 100, x_calculate_intensity = TRUE, threshold = 0.2, x_model = "gaussian")
```

5.5 Run statistical tests

The two functions for boundary statistics require only the raster with boundary elements and the matching null distribution object, produced by `boundary_null_distrib`.

```
n_boundaries(song_boundaries, song_boundary_null)

#> n_boundary    p-value
```

```
#>         13          0

longest_boundary(song_boundaries, song_boundary_null)

#> longest_boundary          p-value
#>         45260.75             0.06

n_boundaries(genetic_boundaries, genetic_boundary_null)

#> n_boundary    p-value
#>          1          0

longest_boundary(genetic_boundaries, genetic_boundary_null)

#> longest_boundary          p-value
#>         61454.64             0.00
```

The functions for boundary overlap statistics also take the boundary element rasters and null distribution as arguments. In this case, it requires two boundary element `SpatRaster` objects, one for each variable. The order of the variables should match the order used in `overlap_null_distrib`. I am interested in whether the genetic boundary depends on song boundaries (i.e., a unidirectional trend), so I am using the `n_overlap_boundaries` (direct overlap) and `average_min_x_to_y` (O_x) tests but not the `average_min_distance` (O_{xy}) test. The genetic boundary raster is the first argument, and the song boundary raster is the second argument.

```
n_overlap_boundaries(genetic_boundaries, song_boundaries, boundary_overlap_null)

#> n_overlapping       p-value
#>            44             0

average_min_x_to_y(genetic_boundaries, song_boundaries, boundary_overlap_null)

#> avg_min_x_to_y        p-value
#>       1911.751          0.000
```

5.6 Interpretation of example data output

When analyzing the data from Luo et al. (2024), the boundary statistics for the genetic data were significant, suggesting the presence of one cohesive genetic boundary. The boundary statistics for the song data were less clear, as the p-values were at or around 0.05. If the analysis was repeated with more iterations of the neutral landscape model, it may suggest multiple cohesive song boundaries. Results from the boundary overlap statistics show significant direct overlap between the genetic and song boundaries and spatial proximity from the genetic boundary to song boundaries, suggesting a spatial correlation between boundaries of the two variables. While boundary overlap statistics can only demonstrate a correlation between boundaries, the results are generally consistent with the hypothesis that song boundaries are facilitating a coincident genetic boundary.

6 Summary

BoundaryStats implements five boundary overlap statistics. Boundary analyses like the boundary overlap statistics implemented here can be used across many contexts that make use of spatially distributed data. For example, spatial ecologists and epidemiologists can use boundary overlap statistics to assess whether environmental variables are influencing

the distribution of organismal traits or disease occurrences. Environmental influences can, in some cases, be detected through the co-occurrence and coincidence of geographic boundaries; environmental boundaries may produce boundaries in the variables of interest. As such, this new open-source, cross-platform implementation will make boundary statistical methods more widely accessible to researchers.

References

N. Adimalla, J. Chen, and H. Qian. Spatial characteristics of heavy metal contamination and potential human health risk assessment of urban soils: A case study from an urban region of South India. *Ecotoxicology and Environmental Safety*, 194(110406), 05 2020. doi: 10.1016/j.ecoenv.2020.110406. URL https://linkinghub.elsevier.com/retrieve/pii/S0147651320302451. [p90]

D. M. J. S. Bowman, S. Ondei, A. Lucieer, S. Foyster, and L. D. Prior. Forest-sedgeland boundaries are historically stable and resilient to wildfire at Blakes Opening in the Tasmanian Wilderness World Heritage Area, Australia. *Landscape Ecology*, 38:205–222, 2023. doi: 10.1007/s10980-022-01558-x. [p89]

K. Caye, T. M. Deist, H. Martins, O. Michel, and O. François. TESS3: Fast inference of spatial population structure and genome scans for selection. *Molecular Ecology Resources*, 16(2): 540–548, 2016. doi: 10.1111/1755-0998.12471. [p90]

M. Fortin, P. Drapeau, and G. M. Jacquez. Quantification of the spatial co-occurrences of ecological boundaries. *Oikos*, 77(1):51–60, 10 1996. doi: 10.2307/3545584. URL https://www.jstor.org/stable/3545584?origin=crossref. [p89, 90, 91]

R. J. Hijmans. *terra: Spatial data analysis*. 2023. URL https://CRAN.R-project.org/package=terra. [p94]

B. Hong, B. J. Bonczak, A. Gupta, L. E. Thorpe, and C. E. Kontokosta. Exposure density and neighborhood disparities in COVID-19 infection risk. *Proceedings of the National Academy of Sciences*, 118(13):e2021258118, 2021. doi: 10.1073/pnas.2021258118. [p90]

A. Hulme-Beaman, A. Rudzinski, J. E. J. Cooper, R. F. Lachlan, K. Dobney, and M. G. Thomas. geoorigins: A new method and R package for trait mapping and geographic provenancing of specimens without categorical constraints. *Methods in Ecology and Evolution*, 11(10): 1247–1257, 2020. doi: 10.1111/2041-210X.13444. [p94]

G. M. Jacquez. The map comparison problem: Tests for the overlap of geographic boundaries. *Statistics in Medicine*, 14(21-22):2343–2361, 1995. doi: 10.1002/sim.4780142107. [p89, 90, 91]

G. M. Jacquez. Geographic boundary analysis in spatial and spatio-temporal epidemiology: Perspective and prospects. *Spatial and Spatio-temporal Epidemiology*, 1(4):207–218, 2010. doi: 10.1016/j.sste.2010.09.003. [p89]

G. M. Jacquez, S. Maruca, and M. Fortin. From fields to objects: A review of geographic boundary analysis. *Journal of Geographical Systems*, 2:221–241, 2000. doi: 10.1007/PL00011456. [p89]

A. R. Luo, S. Lipshutz, J. Phillips, R. T. Brumfield, and E. P. Derryberry. Song and genetic divergence within a subspecies of white-crowned sparrow (*zonotrichia leucophrys nuttalli*). *PLoS ONE*, 19(5):e0304348, 05 2024. doi: 10.1371/journal.pone.0304348. [p94, 97]

F. Manni, E. Guérard, and E. Heyer. Geographic patterns of (genetic, morphological, linguistic) variation: How barriers can be detected with Monmonier'salgorithm. *Human Biology*, 76(2):173–190, 2004. doi: 10.1353/hub.2004.0034. [p90]

E. J. Pebesma. Multivariable geostatistics in S: The gstat package. *Computers & Geosciences*, 30(7):683–691, 2004. doi: 10.1016/j.cageo.2004.03.012. [p92]

A. E. Polakowska, M. Fortin, and A. Couturier. Quantifying the spatial relationship between bird species' distributions and landscape feature boundaries in southern Ontario, Canada. *Landscape Ecology*, 27(10):1481–1493, 12 2012. doi: 10.1007/s10980-012-9804-6. [p90]

A. Raj, M. Stephens, and J. K. Pritchard. fastSTRUCTURE: Variational inference of population structure in large SNP data sets. *Genetics*, 197(2):573–589, 06 2014. doi: 10.1534/genetics.114.164350. URL https://academic.oup.com/genetics/article/197/2/573/6074271. [p94]

T. Safner, M. P. Miller, B. H. McRae, M. Fortin, and S. Manel. Comparison of Bayesian clustering and edge detection methods for inferring boundaries in landscape genetics. *International Journal of Molecular Sciences*, 12(2):865–889, 01 2011. doi: 10.3390/ijms12020865. URL http://www.mdpi.com/1422-0067/12/2/865. [p91]

S. Saura and J. Martínez-Millan. Landscape patterns simulation with a modified random clusters method. *Landscape Ecology*, 15:661–678, 2000. doi: 10.1023/A:1008107902848. [p92]

V. St-Louis, M. Fortin, and A. Desrochers. Spatial association between forest heterogeneity and breeding territory boundaries of two forest songbirds. *Landscape Ecology*, 19(6):591–601, 08 2004. doi: 10.1023/B:LAND.0000042849.63040.a9. URL http://link.springer.com/10.1023/B:LAND.0000042849.63040.a9. [p90]

T. Strydom and T. Poisot. Spatialboundaries.jl: Edge detection using spatial wombling. *Ecography*, 2023(5):e06609, 05 2023. doi: 10.1111/ecog.06609. URL https://onlinelibrary.wiley.com/doi/10.1111/ecog.06609. [p90]

P. Tarroso, R. J. Pereira, F. Martínez-Freiría, R. Godinho, and J. Brito. Hybridization at an ecotone: ecological and genetic barriers between three Iberian vipers. *Molecular Ecology*, 23(5):1108–1123, 03 2014. doi: 10.1111/mec.12671. URL https://onlinelibrary.wiley.com/doi/10.1111/mec.12671. [p90]

M. van Ham, D. Manley, N. Bailey, L. Simpson, and D. Maclennan, editors. *Neighbourhood effects research: New perspectives*. Springer Netherlands, 2012. doi: 10.1007/978-94-007-2309-2. URL http://dx.doi.org/10.1007/978-94-007-2309-2. [p90]

H. H. Wagner and M. Fortin. Spatial analysis of landscapes: Concepts and statistics. *Ecology*, 86(8):1975–1987, 2005. doi: 10.1890/04-0914. [p89]

H. H. Wagner and M. Fortin. A conceptual framework for the spatial analysis of landscape genetic data. *Conservation Genetics*, 14(2):253–261, 2013. doi: 10.1007/s10592-012-0391-5. URL http://link.springer.com/10.1007/s10592-012-0391-5. [p90]

L. A. Waller, B. W. Turnbull, L. C. Clark, and P. Nasca. Chronic disease surveillance and testing of clustering of disease and exposure: Application to leukemia incidence and TCE-contaminated dumpsites in upstate New York. *Environmetrics*, 3(3):281–300, 1992. doi: 10.1002/env.3170030303. [p90]

X. Wei and C. P. S. Larsen. Methods to detect edge effected reductions in fire frequency in simulated forest landscapes. *ISPRS International Journal of Geo-Information*, 8(6):277, 2019. doi: 10.3390/ijgi8060277. URL https://www.mdpi.com/2220-9964/8/6/277. [p91]

H. Wickham. *ggplot2: Elegant graphics for data analysis*. 2016. URL https://ggplot2.tidyverse.org. [p95]

Amy Luo
University of Tennessee, Knoxville
Department of Ecology and Evolutionary Biology
Knoxville, Tennessee
ORCiD: 0000-0002-9197-4673
amy.luo.15@gmail.com

R package GofCens: Goodness-of-Fit Methods for Right-Censored Data

by Mireia Besalú, Klaus Langohr, Matilde Fransciso, Arnau Garcia-Fernández, and Guadalupe Gómez Melis

Abstract This paper presents the R package GofCens, which offers graphical tools and implements goodness-of-fit tests for right-censored data. The first part provides a thorough review of current methodologies for assessing goodness of fit in the presence of right-censored data. The subsequent sections present the main functions of GofCens and are illustrated by means of a right-censored sample from a log-normal distribution and a data set on NBA players' mortality, which is included in the package.

1 Introduction

Goodness-of-fit techniques are important to test the validity of parametric models and to ensure that the modeling assumptions hold true. Historically, goodness-of-fit tests have been developed for complete data, that is, when all the individual sample measurements have been observed. The Kolmogorov-Smirnov, Cramér-von Mises, and Anderson-Darling statistics are among the most commonly used goodness-of-fit tests. They are based on a measure of the discrepancy between the empirical and theoretical distribution functions.

In order to account for censored data, these statistics can be modified replacing the empirical distribution function by the product-limit estimator of the distribution function. Modifications of Kolmogorov-Smirnov statistics for censored or truncated data go back to Fleming et al. (1980) and Koziol and Byar (1975). Pioneer extensions of Cramér-von Mises and Anderson-Darling goodness-of-fit statistics to account for random right-censored data are found in Pettitt and Stephens (1976) and Koziol and Green (1976). Concerning chi-squared tests, Mihalko and Moore (1980) propose an extension to censored data, specifically for type II right-censored data.

When the theoretical distribution function is completely specified and the data are uncensored, the above tests are all distribution-free, with known distributions. However, this property no longer holds when data are censored or when the theoretical distribution function involves unknown parameters. Moreover, there is no general theory of asymptotic optimality in such cases (Lehmann and Romano, 2005). In fact, any test can achieve high asymptotic power or perform uniformly well against local or contiguous alternatives, especially when the family of possible alternatives is extensive (Janssen, 2000). Given these limitations, graphical techniques have become a standard and straightforward way to assess distributional assumptions. They allow for a more intuitive examination of model fit. In addition, graphical methods complement formal goodness-of-fit tests. They often provide insights that the tests alone may not reveal.

The most well-known plots for assessing goodness-of-fit are probability plots. These include the Probability–Probability (P–P) plot, which compares the theoretical and estimated cumulative distribution functions, and the Quantile–Quantile (Q–Q) plot, which compares theoretical and estimated quantiles. There are also two alternatives to these plots: the Stabilised Probability plot (Michael, 1983), which transforms the axes of the P-P plot to approximately get the same variance in each plotted point, and the Empirically Rescaled plot (Waller and Turnbull, 1992), which is very useful when there is a high percentage of random right-censored data.

Goodness-of-fit methods for complete data and right-censored observations are widely available. However, many of these methods are not implemented in `R`, and those that are tend to be scattered across different packages. Among the publicly available computational tools in `R`, the **fitdistrplus** package (Delignette-Muller and Dutang, 2015; Delignette-Muller et al., 2025) provides the function `fitdistcens()`, which returns the result of the fit of

any parametric distribution to a possibly right-censored data set. The function `cdfcomp()` of the same package can be used to graphically compare multiple fitted distributions with uncensored data. The `probplot()` function from the **e1071** package (Meyer et al., 2024) generates probability plots for specified theoretical distribution. Additionally, the `distChooseCensored()` function from the **EnvStats** package (Millard, 2013; Millard and Kowarik, 2025) performs Shapiro-Wilk and Shapiro-Francia goodness-of-fit tests for normal, log-normal, and Gamma distributions, handling both complete and singly censored data (Royston, 1993).

In this work, we present the `R` package **GofCens** (Langohr et al., 2025), which provides various analytical methods and graphical tools to assess the goodness of fit of parametric models for non-negative, right-censored lifetime data. These tools can also be applied to complete lifetime data. Right censoring is restricted to type I or random censoring, and non-informative censoring is assumed for the failure time. In what follows, we review the analytical and graphical goodness-of-fit techniques for complete and right-censored data implemented in **GofCens**. Thereafter, the main functions of the **GofCens** package are presented and applied to a real data set, respectively. We conclude the paper with some final remarks on the current state of the package and possible enhancements.

2 Methods

Let T denote the time to an event of interest, with distribution function F. The **GofCens** package provides goodness-of-fit methods to assess whether a univariate sample from T, either right-censored or complete, comes from a specified distribution family $F_0(t;\theta)$, such as the Weibull, where θ is a vector of unknown parameters. Formally, the null hypothesis in a goodness-of-fit test is given by $H_0\colon F(t) = F_0(t;\theta)$. Specifically, the **GofCens** package provides implementations of well-known tests such as the Kolmogorov-Smirnov, Cramér-von Mises, and Anderson-Darling tests based on the empirical distribution function for complete data and their extensions for right-censored data. Additionally, **GofCens** includes a chi-squared-type test based on the squared differences between observed and expected counts using random cells, with an extension tailored for right-censored data. Recognizing that these tests may not always yield high power, **GofCens** complements them with a series of graphical tools to aid in selecting the most suitable parametric model.

Goodness-of-fit tests based on the empirical distribution function are typically developed under a fully specified null hypothesis $H_0\colon F(t) = F_0(t;\theta^*)$, such as a Weibull distribution with known parameters (e.g., $\alpha = 2$ and $\beta = 1$). They are less often formulated for the more general case $H_0 : F(t) = F_0(t;\theta)$, where θ is unknown. This distinction has important implications for how such tests are applied in practice. This situation is common when fitting empirical data to parametric distributions with unknown parameters. To address it, we replace the unknown parameter θ with its maximum likelihood estimator $\hat{\theta}$. Following the recommendations in Capasso et al. (2009), the implementation of the four tests —Kolmogorov-Smirnov, Cramér-von Mises, Anderson-Darling, and chi-squared tests— uses bootstrap techniques that involve the maximum-likelihood re-estimation of the unknown parameters θ in each simulated sample as we describe in Subsection Bootstrap methods.

In what follows, we briefly outline the theory underlying these methods for testing $H_0\colon F(t) = F_0(t;\theta^*)$ assuming that the data consist of a sample of n individuals, each subject to random right censoring. We denote by $T_1,\ldots,T_n$ independent random variables with a common distribution $F(\cdot)$ and by $C_1,\ldots,C_n$ independent censoring random variables with a common distribution $H(\cdot)$, independent of $T_1,\ldots,T_n$. The observed variables are defined as $Y_i = \min(T_i, C_i)$ and $\delta_i = \mathbf{1}\{T_i \leq C_i\}$, $i = 1,\ldots,n$.

Tests based on empirical distribution functions

In order to test $H_0 : F(t) = F_0(t;\theta^*)$ for all $t \geq 0$, we can use statistics based on a distance between $F_0(t;\theta^*)$ and $\widehat{F}_n(t)$, the empirical distribution function for complete data, or either the Kaplan-Meier or the Nelson-Aalen estimator, if right-censored data are present. In both cases, large values of the distance indicate evidence against the hypothesized model.

Kolmogorov-Smirnov test

The Kolmogorov-Smirnov goodness-of-fit test is the most used analytical method to test goodness of fit when one is dealing with uncensored data. Fleming et al. (1980) published a modification of the Kolmogorov-Smirnov test in an attempt to obtain increased power when applied to uncensored data. They also generalized this test for arbitrarily right-censored data.

For complete data, the Kolmogorov-Smirnov statistic is defined as $D_n = \sup_t |\widehat{F}_n(t) - F_0(t;\theta^*)|$, where $\widehat{F}_n(t)$ is the empirical distribution function. To account for right-censored data, the Kolmogorov-Smirnov statistic is modified as

$$\widehat{D}_n = \sup_{0\leq t\leq t_m} |\widehat{F}_n(t) - F_0(t;\theta^*)| = \sup_{0\leq t\leq t_m} \left| \int_0^t \frac{\widehat{S}_n(t) S_0(s;\theta^*)}{\widehat{S}_n(s)} d\left[\widehat{\Lambda}_n(s) - \Lambda_0(s;\theta^*)\right] \right|. \quad (1)$$

Here, $\widehat{F}_n$, is replaced by $\widehat{F}_n = 1 - \widehat{S}_n = 1 - e^{-\widehat{\Lambda}_n}$, being $\widehat{\Lambda}_n$ the Nelson-Aalen estimator of the cumulative hazard function. In this case, S_0 and Λ_0 denote the survival and the cumulative hazard function of the hypothesized distribution F_0, respectively, and t_m is the largest observed time in the sample.

The standardized modified Kolmogorov-Smirnov statistic for censored data converges in distribution to $\sup_{0\leq t\leq F_0(t_m;\theta^*)} |B(F_0(t;\theta^*))|$, where $B(t)$ is a Brownian bridge. According to the results in Koziol and Byar (1975), the asymptotic distribution of $\widehat{D}_n$ is given by

$$\lim_{n\to\infty} P(\sqrt{n}\widehat{D}_n \leq k) = \sum_{j=-\infty}^{\infty} (-1)^j e^{-2j^2k^2} P\left(\left|Z - 2jk\sqrt{\frac{1-T}{T}}\right| < k\sqrt{\frac{1}{T-T^2}}\right), \quad (2)$$

where Z is a standard normal random variable and $T = F_0(t_m;\theta^*)$. An approximation for the p value is proposed by Fleming et al. (1980), which is considered acceptable when the p value is less than 0.8 and excellent when it is less than 0.2.

Cramér-von Mises-type test

For uncensored data, the Cramér-von Mises statistic is given by

$$M_n = n \int_{-\infty}^{+\infty} \left(\widehat{F}_n(t) - F_0(t;\theta^*)\right)^2 dF_0(t).$$

For right-censored data, let $Y_1, \ldots, Y_{n_r}$ represent the n_r observed failure times, that is, $\sum_{i=1}^{n} \delta_i = n_r$. If we transform the order failure times $Y_{(1)}, \ldots, Y_{(n_r)}$ into Uniform$(0,1)$ random variables $u_{(1)}, \ldots, u_{(n_r)}$, defined as $u_{(i)} = F_0(Y_{(i)};\theta^*)$, and adopt the convention that $u_{(0)} = 0$ and $u_{(n_r+1)} = 1$, we can express the modified Cramér-von Mises statistic as

$$\widehat{M}_n = n_r \sum_{j=1}^{n_r+1} \widehat{F}_n(u_{(j-1)})(u_{(j)} - u_{(j-1)}) \left(\widehat{F}_n(u_{(j-1)}) - (u_{(j)} + u_{(j-1)})\right) + \frac{n_r}{3}, \quad (3)$$

which can be easily computed. However, although the asymptotic distribution of $\widehat{M}_n$ has been studied by Pettitt and Stephens (1976) and Koziol and Green (1976), its practical implementation is not straightforward.

Anderson-Darling test

The Anderson-Darling test statistic is a modification of M_n where the integrand is weighted by the inverse of $F_0(t)(1 - F_0(t))$:

$$A_n = n \int_{-\infty}^{+\infty} (\widehat{F}_n(t) - F_0(t;\theta^*))^2 \frac{dF_0(t;\theta^*)}{F_0(t;\theta^*)(1 - F_0(t;\theta^*)}.$$

For right-censored data, using the same notation as in the Cramér-von Mises statistic, the Anderson-Darling statistic is computed as

$$\begin{aligned} \widehat{A}_n = -n_r + n_r \sum_{j=1}^{n_r} (\widehat{F}_n(u_{(j-1)}) - 1)^2 \left[\log|1 - u_{(j-1)}| - \log|1 - u_{(j)}|\right] \\ + n_r \sum_{j=1}^{n_r-1} \widehat{F}_n^2(u_{(j)}) \left[\log|u_{(j+1)}| - \log|u_{(j)}|\right] - n_r \log|u_{(n)}|. \end{aligned} \tag{4}$$

As with the Cramér-von Mises statistic, the asymptotic distribution of $\widehat{A}_n$ has been studied by Pettitt and Stephens (1976) but its practical implementation is not straightforward.

Chi-squared type tests

Kim (1993) introduced a general class of χ^2 goodness-of-fit statistics for randomly right-censored data, extending Pearson's statistic where the cells are replaced by random cells.

Random cells are built as intervals in $\mathbb{R}$ whose boundaries depend on some sample summaries. These boundaries are defined by a vector φ of r statistics, yielding intervals $I_j(\varphi) = (a_{j-1}(\varphi), a_j(\varphi)]$, where $j = 1, \ldots, M$, and $-\infty = a_0(\varphi) < a_1(\varphi) < \cdots < a_M(\varphi) = \infty$. For instance, we might take the quantiles $1/M$, $2/M, \ldots, 1$ as random cells boundaries $a_1(\varphi), \ldots, a_M(\varphi)$.

The observed frequency of cell j, $N_{nj}(\varphi) = n \int_{I_j(\varphi)} d\widehat{F}_n$, is calculated using either the empirical distribution function or 1 minus the Kaplan-Meier estimator of the survival function if there are right-censored data. The expected probability of cell j under the null hypothesis of a fully specified distribution, $F_0(t;\theta^*)$, is denoted by $p_j(\theta^*, \varphi_n)$.

The class of χ^2 statistics is defined as

$$T_n = V_n^\top(\theta^*, \varphi_n) \cdot K_n(\theta^*, \varphi_n) \cdot V_n(\theta^*, \varphi_n), \tag{5}$$

where $K_n(\theta^*, \varphi)$ is a symmetric, non-negative definite matrix. $K_n(\theta^*, \varphi)$ is, in most cases, the inverse of a consistent estimator of the asymptotic variance-covariance matrix of the random vector $V_n(\theta^*, \varphi_n)$. The matrix can be computed following formulas (3.1) in Kim (1993). The vector $V_n(\theta^*, \varphi) = (\nu_{nj})_{j=1,\ldots,M}$, contains differences between observed and expected frequencies. Each component is defined as

$$\nu_{nj}(\theta^*, \varphi_n) = n^{-\frac{1}{2}} N_{nj}(\varphi_n) - n^{\frac{1}{2}} \cdot p_j(\theta^*, \varphi_n).$$

The limiting distribution of T_n is a linear combination of independent noncentral χ^2-distributed variables which is complex to implement.

Computation of test statistic's distribution

The limiting distributions of the statistics in (3), (4), and (5) under the null hypothesis $H_0 : F(t) = F_0(t;\theta)$, when the parameter θ is unknown and replaced by its maximum likelihood estimate $\hat{\theta}$, are too complex to implement directly. Therefore, the corresponding p values are computed using bootstrap methods (see Subsubsection Bootstrap methods for right-censored data). For consistency, the p values of the Kolmogorov-Smirnov test are also

obtained using bootstrap.

The function `KScens()` of the **GofCens** package calculates the value of the test statistic of the Kolmogorov-Smirnov test, and, by default, approximates the p value using bootstrap methods. Alternatively, the function also allows p value computation based on the asymptotic distribution given in (2). The function `CvMcens()` in the **GofCens** package calculates the observed Cramér-von Mises statistic adapted for right-censored data, and provides an approximation of the corresponding p value using bootstrap methods. The function `ADcens()` in the **GofCens** package calculates the observed Anderson-Darling statistic for right-censored data and provides an approximation of the corresponding p value using bootstrap methods. The `chisqcens()` function in the **GofCens** package uses bootstrap techniques to compute the p values.

Bootstrap methods for right-censored data

Since the asymptotic distributions of the described goodness-of-fit statistics are unknown when the null hypothesis is not fully specified, bootstrap techniques offer a practical alternative for computing p values. The bootstrap method is as well a practical alternative if H_0 is fully specified, but the asymptotic distribution depends on a complex expression. This section describes the application of bootstrapping to derive p values for tests presented in the previous subsection. We use the notation G_n to refer to any of the following test statistics: the Kolmogorov–Smirnov statistic defined in (1), the Cramér–von Mises statistic in (3), the Anderson–Darling statistic in (4), and the chi-squared statistic in (5).

To test the null hypothesis $H_0 : F(t) = F_0(t;\theta)$ using complete and right-censored data, we follow the guidelines of Capasso et al. (2009). Accordingly, the observed data is utilized to estimate the parameter θ, denoted as $\hat{\theta}_n$, using maximum likelihood estimation. Subsequently, we generate B independent bootstrap samples of the same size (n) as the original data set as follows. Specifically, for each iteration b, $b = 1, \ldots, B$, we compute the bootstrap statistic $(\widehat{G}_n^{\hat{\theta}_n})_b$ carrying out the following steps:

1. Generation of survival times $T_1^b, \ldots, T_n^b$ from the fitted distribution $F_0(t;\hat{\theta}_n)$.
2. Generation of censoring times $C_1^b, \ldots, C_n^b$ from the non-parametric estimation of H obtained with the Kaplan-Meier estimator.
3. Generation of observed survival times $Y_i^b = \min(T_i^b, C_i^b)$, and event indicators $\delta_i^b = \mathbf{1}\{T_i^b \leq C_i^b\}, i = 1, \ldots n$.
4. Maximum likelihood estimation of the parameter, $\hat{\theta}_n^b$, given $(Y_i^b, \delta_i^b), i = 1, \ldots n$.
5. Computation of the bootstrap statistic, $(\widehat{G}_n^{\hat{\theta}_n})_b$.

By repeating this process for many bootstrap samples —the default value of the functions `KScens()`, `CvMcens()`, `ADcens()`, and `chisqcens()` is $B = 999$—, the sequence of bootstrap statistics, $(\widehat{G}_n^{\hat{\theta}_n})_b$, $b = 1, \cdots, B$, represents the empirical distribution of the statistic G_n under the null hypothesis.

The p value is calculated by comparing the observed test statistic $\widehat{G}_n$ to the empirical distribution of the bootstrap statistic. Specifically, the p value is the proportion of bootstrap statistic values $(\widehat{G}_n^{\hat{\theta}_n})_b$ greater than or equal to the observed statistic $\widehat{G}_n$:

$$p = \frac{1}{B+1}\Big(\sum_{i=1}^{B} \mathbf{1}\{\widehat{G}_n \leq (\widehat{G}_n^{\hat{\theta}_n})_b\} + 1\Big),$$

where adding 1 in both the numerator and the denominator avoids a zero p value.

Note that steps 2 and 3, which involve generating the right-censored sample, are crucial. This is because the distribution of G_n depends not only on F_0 but also on H and the proportion of censored data. The functions `KScens()`, `CvMcens()`, `ADcens()`, and `chisqcens()` are

designed so that, if the original data contains no right-censored observations, the censoring times in step 2 are automatically set to ∞.

Moreover, as explained and illustrated in Section 'Goodness-of-fit tests' below, these functions allow testing the null hypothesis of a fully specified choice of the parameters, $H_0 : F(t) = F_0(t; \theta^*)$. In this case, the survival times in step 1 are generated from $F_0(t; \theta^*)$.

Goodness-of-fit plots

The **GofCens** package provides six different plots for evaluating goodness of fit, applicable to both right-censored and complete data. In most cases, a roughly straight line formed by the points suggests good agreement with the hypothesized theoretical distribution.

Probability-probability and Stabilized Probability plots

The probability-probability plot (P-P plot) maps $\widehat{F}_0(t)$ against $\widehat{F}_n(t)$, where $\widehat{F}_0(t)$ corresponds to either $F_0(t; \theta)$ with the unknown parameters replaced by their maximum likelihood estimates or to $F_0(t; \theta^*)$. $\widehat{F}_n$ represents either the empirical distribution function or 1 minus the Kaplan-Meier estimator of the survival function, if right-censored data are present.

In order to enhance the interpretability of the P-P plot and because some of the plotted points have a larger variability than others, Michael (1983) proposed the Stabilized Probability plot (SP plot). In the same way as the arcsin transformation can be used to stabilize the variance of a uniform order statistic, this function can be applied to stabilize the variance of $\widehat{F}_0(t)$. The variances of the resulting SP plotted points are all approximately equal.

Quantile-quantile plot

The quantile-quantile plot (Q-Q plot) maps the theoretical quantiles against the estimated quantiles, that is, $\widehat{F}_0^{-1}(\widehat{F}_n(t))$ against t. An empirical rescaling of the axes can help to overcome the problem that the plotted points might not be evenly spread, in particular in the presence of right-censored data. The Empirically Rescaled plot (ER plot) plots $\widehat{F}_u(\widehat{F}_0^{-1}(\widehat{F}_n(t)))$ against $\widehat{F}_u(t)$, where $\widehat{F}_u(t)$ is the empirical cumulative distribution function of the points corresponding to the uncensored observations (Waller and Turnbull, 1992).

Table 1 provides an overview of the axes of the probability and quantile-quantile plots.

Plot	Abscissa	Ordinate
P-P plot	$\widehat{F}_n(t)$	$\widehat{F}_0(t)$
Q-Q plot	t	$\widehat{F}_0^{-1}(\widehat{F}_n(t))$
SP plot	$\frac{\pi}{2} \arcsin\left(\widehat{F}_n(t)^{\frac{1}{2}}\right)$	$\frac{\pi}{2} \arcsin\left(\widehat{F}_0(t)^{\frac{1}{2}}\right)$
ER plot	$\widehat{F}_u(t)$	$\widehat{F}_u(\widehat{F}_0^{-1}(\widehat{F}_n(t)))$

Table 1: Probability and quantile-quantile plots to assess goodness of fit.

Cumulative hazard and Kaplan-Meier plots

Another set of plots consists of cumulative hazard plots, derived from transforming the cumulative hazard function Λ in such a way that it becomes linear in t or in $\log(t)$. The Nelson-Aalen estimator $\widehat{\Lambda}$ of Λ is computed from the data, and the distribution-specific transformation $A(\widehat{\Lambda}(t))$ is then plotted against either t or its natural logarithm. The transformations for all the distributions considered in the **GofCens** package are detailed in Table 2.

In addition, the **GofCens** package includes a function to graphically compare the Kaplan-Meier estimate of the survival function $(1 - \widehat{F}_n(t))$ with the parametric estimations of the

Distribution	Survival function	Cum. hazard function	Plot
Exponential [Exp(β)]	$e^{-\frac{t}{\beta}}$	$\frac{t}{\beta}$	$\widehat{\Lambda}(t)$ vs t
Weibull [Wei(α, β)]	$e^{-(\frac{t}{\beta})^{\alpha}}$	$(\frac{t}{\beta})^{\alpha}$	$\log \widehat{\Lambda}(t)$ vs $\log t$
Gumbel [Gum(μ, β)]	$1 - e^{-e^{-\frac{t-\mu}{\beta}}}$	$-\log\left(1 - e^{-e^{-\frac{t-\mu}{\beta}}}\right)$	$-\log\left(-\log(1 - e^{-\widehat{\Lambda}(t)})\right)$ vs t
Log-Logistic [LLogis(α, β)]	$\frac{1}{1+\left(\frac{t}{\beta}\right)^{\alpha}}$	$\log\left(1+\left(\frac{t}{\beta}\right)^{\alpha}\right)$	$\log\left(e^{\widehat{\Lambda}(t)} - 1\right)$ vs $\log t$
Logistic [Logis(μ, β)]	$\frac{e^{-\frac{t-\mu}{\beta}}}{1+e^{-\frac{t-\mu}{\beta}}}$	$\log\left(1+e^{-\frac{t-\mu}{\beta}}\right)$	$\log\left(e^{\widehat{\Lambda}(t)} - 1\right)$ vs t
Log-Normal [LN(μ, β)]	$\int_{\frac{\log t-\mu}{\beta}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$	$-\log\left(1 - \Phi\left(\frac{\log t-\mu}{\beta}\right)\right)$	$\Phi^{-1}\left(1 - e^{-\widehat{\Lambda}(t)}\right)$ vs $\log t$
Normal [N(μ, β)]	$\int_{t}^{\infty} \frac{1}{\beta\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\beta^2}} dx$	$-\log\left(1 - \Phi\left(\frac{t-\mu}{\beta}\right)\right)$	$\Phi^{-1}\left(1 - e^{-\widehat{\Lambda}(t)}\right)$ vs t
4-Parameter Beta [Beta(α, γ, a, b)]	$1 - \frac{B_{(\alpha,\gamma,a,b)}(t)}{B(\alpha,\gamma)}$	$-\log\left(1 - \frac{B_{(\alpha,\gamma,a,b)}(t)}{B(\alpha,\gamma)}\right)$	$B^{-1}_{(\alpha,\gamma,a,b)}\left(B(\alpha,\gamma)\left(1 - e^{-\widehat{\Lambda}(t)}\right)\right)$ vs t

Table 2: List of the distributions that are implemented in the **GofCens** package. The last column shows the specific transformations of time and the estimated cumulative hazard function of the cumulative hazard plots, which are implemented by the function `cumhazPlot()`. The shape parameters are denoted by $\alpha > 0$ and $\gamma > 0$, the location parameter by $\mu \in R$, and the scale parameter by $\beta > 0$. The time t will be always positive. $B(\alpha, \gamma)$ denotes the Beta function and $B_{(\alpha,\gamma,a,b)}(t) = \int_0^{\frac{t-a}{b-a}} u^{\alpha-1}(1-u)^{\gamma-1}du$.

survival function from the parametric models under study ($1 - \widehat{F}_0(t)$).

3 The GofCens package

Installation and dependencies

The **GofCens** package can easily be installed from any CRAN repository by running the `R` command

```
R> install.packages("GofCens")
```

As shown on the corresponding web page of the CRAN (https://cran.r-project.org/web/packages/GofCens/index.html), **GofCens** depends on the packages **survival** (Therneau and Grambsch, 2000) and **actuar** (Dutang et al., 2008; Goulet et al., 2025). The former is needed to estimate the survival and cumulative hazard functions, whereas the latter implements several distributions that are not available in standard `R` packages. Additionally, **GofCens** imports functions from the following packages: the `fitdist()` and `fitdistcens()` functions of the **fitdistrplus** package (Delignette-Muller and Dutang, 2015; Delignette-Muller et al., 2025) provide the estimation of the distributions' parameters; the packages **ggplot2** (Wickham, 2016; Wickham et al., 2025), **gridExtra** (Auguie, 2017), and the base package **grid** are needed for several features of the graphical tools; and the bootstrap methods to compute the p values of the Kolmogorov-Smirnov, Cramér-von Mises, Anderson-Darling,

and chi-squared type tests are implemented with the function `boot()` of the **boot** package (Canty and Ripley, 2024). Moreover, the `is()` function from the base package **methods** is used internally to verify object classes during function execution.

Package overview

The **GofCens** package provides both graphical tools and statistical test functions to assess goodness of fit for the distributions shown in Table 2.

The graphical functions, which facilitate visual evaluation of the model adequacy, include:

- Function `probPlot()` draws the probability and quantile-quantile plots of Table 1.
- Function `cumhazPlot()` draws the cumulative hazard plots of Table 2.
- Function `kmPlot()` draws the Kaplan-Meier estimates of the survival function and overlays parametric fits of the survival function.

The statistical test functions, which implement goodness-of-fit tests, include:

- Function `KScens()` implements the Kolmogorov-Smirnov test adapted to right-censored data.
- Function `CvMcens()` implements the Cramér-von Mises test adapted to right-censored data using bootstrap techniques.
- Function `ADcens()` implements the Anderson-Darling test adapted to right-censored data using bootstrap techniques.
- Function `gofcens()` computes the test statistics of the Kolmogorov-Smirnov, Cramér-von Mises, and Anderson-Darling tests adapted to right-censored data and returns the corresponding p values.
- Function `chisqcens()` implements the chi-squared type test based on random cells and using bootstrap techniques.

These functions share several features: although they are primarily designed for right-censored data, they can also be applied to complete datasets. They provide parameter estimates for the distributions under study and support two input formats: users can either provide vectors with the observed survival times and the corresponding event indicator, or use the common formula interface with a `Surv` object. Earlier versions of the functions were implemented in `R` by Febrer Galvany (2015), Besalú Mayol (2016), and García Carrasco (2017).

Specific features of the statistical test functions include the following: On the one hand, these functions can be used to assess the goodness of fit for any distribution, provided that the corresponding `p`*name*, `d`*name*, and `r`*name* functions are available. On the other hand, they return test-specific objects for which `print`, `summary`, and `print.summary` methods are implemented.

In addition, the **GofCens** package comes with a data frame called `nba`, which contains the survival times of 3962 former players of the National Basketball Association (NBA) until 2019. These data were analyzed previously by Martínez et al. (2022).

A simulated example

Following, we will present the main features of the eight functions and illustrate their use with simulated survival times. For this purpose, we generate 300 survival times from a log-normal distribution with location parameter $\mu = 2$ and scale parameter $\beta = 1$,

i.e., $T \sim LN(2, 1)$, and 300 censoring times from an exponential distribution with scale parameter $\beta = 20$, i.e, $C \sim Exp(20)$:

```
R> set.seed(123)
R> survt <- round(rlnorm(300, 2, 1), 2)
R> censt <- round(rexp(300, 1 / 20), 2)
```

The observed right-censored survival times, $Y = \min(T, C)$, and the corresponding event indicators, $\delta = \mathbf{1}\{T \leq C\}$, are computed as follows:

```
R> times <- pmin(survt, censt)
R> delta <- as.numeric(survt <= censt)
```

In total, 106 (35.3%) survival times of the generated sample are right-censored:

```
R> table(delta)
  0   1
106 194
```

Graphical tools: Functions `probPlot()`**,** `cumhazPlot()`**, and** `kmPlot()`

Function `probPlot()`

The `probPlot()` function allows users to assess graphically how well any of the distributions in Table 2 fit to a sample of right-censored survival or complete times. For this purpose, the function implements the four probability and quantile-quantile plots in Table 1. By default, the function draws all four plots, but provides users with the option to choose only a subset of these. If the user does not provide specific distribution parameters, the maximum likelihood estimates are used to draw the plots. One graphical feature of the function is the option for users to choose between the base package **graphics**, which is the default, and the **ggplot2** package for drawing the plots. In the former case, `probPlot()` internally calls the `par()` function, enabling users to set graphical parameters within the function call.

For the illustration of the `probPlot()` function, we assess how well the log-normal distribution (left panel of Figure 1) and the Weibull distribution (right panel of Figure 1) fit to the sample data. The four plots in the left panel show fairly straight lines, thus confirming that the underlying distribution is the log-normal distribution, whereas the plots in the right panel clearly show that the Weibull distribution would not be an adequate distribution to model the data.

The `R` code used to draw the plots in the left and right panels of Figure 1 is the following:

```
R> library("GofCens")
R> probPlot(times, delta, distr = "lognormal", cex.lab = 1.3)
R> probPlot(times, delta, distr = "weibull", ggp = TRUE)
```

To illustrate additional functionalities of this function, we apply it again to the data set to assess the suitability of fitting a log-normal distribution with parameters $\mu = 2$ and $\beta = 1$. For this purpose, we employ the `params0` argument, whose default value is `NULL`, to specify the parameter values. Additionally, the main argument is a `Surv` object and we utilize the `m` argument to display the four probability plots in a single row, as shown in Figure 2. From these plots, it becomes evident that this log-normal distribution does not fit well to the data. Furthermore, we set `prnt = TRUE` so that the function provides both the parameter values used in the probability plots and the maximum likelihood estimates of the parameters, together with the values of the Akaike (AIC) and the Bayesian (BIC) information criteria.

```
R> probPlot(Surv(times, delta) ~ 1, distr = "lognormal", m = matrix(1:4,
+    nrow = 1), params0 = list(location = 2, scale = 1.5), ggp = TRUE, prnt = TRUE)
```

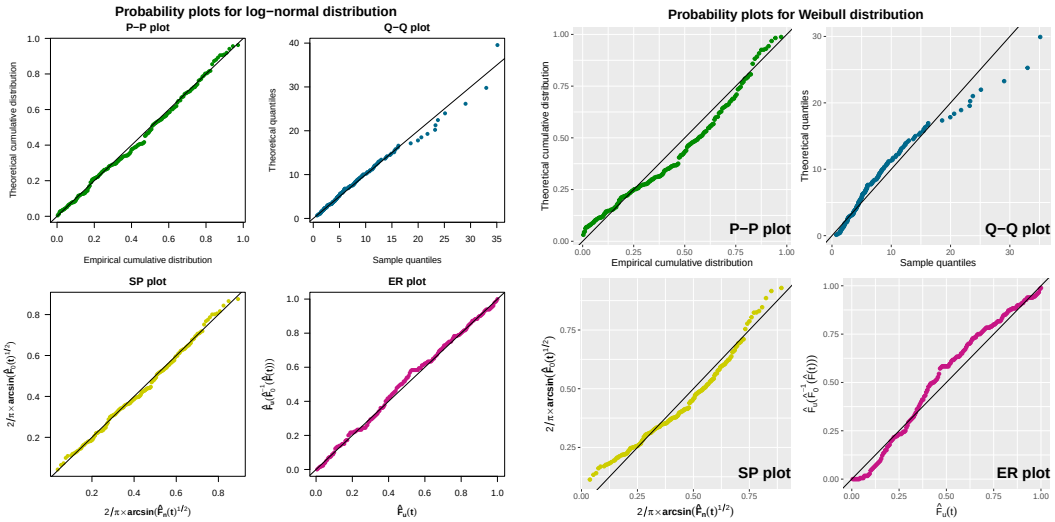


Figure 1: Probability plots applied to a right-censored sample from a log-normal distribution assuming log-normal (left panel) and Weibull distributions (right panel). Both figures are drawn with the `probPlot()` function.

```
Distribution: log-normal

Parameters used in probability plots:
Location: 2
   Scale: 1.5

Parameter estimates:
Location (se): 1.988 (0.058)
   Scale (se): 0.883 (0.045)

   AIC: 1266.899
   BIC: 1274.307
```

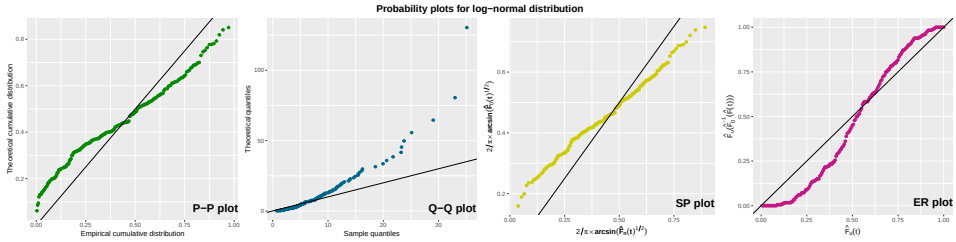


Figure 2: Probability plots applied to a right-censored sample from a log-normal distribution assuming a log-normal distribution with specific parameters ($\mu = 2, \beta = 1.5$). The plots are drawn with the `probPlot()` function.

Function `cumhazPlot()`

The `cumhazPlot()` function can be used to generate cumulative hazard plots for any of the distributions listed in Table 2. As previously noted, the cumulative hazard plot is based on a transformation of the cumulative hazard function to achieve linearity in either t or $\log(t)$. Consequently, given a set of survival times, this function serves as a useful tool to assess which parametric model best fits the data.

By default, the function generates cumulative hazard plots for the Weibull, Gumbel, log-logistic, logistic, log-normal, and normal distributions. Additionally, users can opt to

choose the exponential and beta distributions or select any subset of these distributions. Similar to the `probPlot()` function, users can choose between the base **graphics** package (default) or **ggplot2** to draw the plots.

Setting the argument `prnt` to `TRUE` (default is `FALSE`), the function also returns the maximum likelihood estimates for the specified distribution, along with the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) values.

An illustration with the previously generated right-censored sample from the log-normal distribution is shown in Figure 3. As expected, the points of the cumulative hazard plot for the log-normal distribution show a fairly straight line, whereas most of the other distributions under study can clearly be discarded. This conclusion is further supported by comparing the AIC and BIC values across all distributions, with the log-normal distribution showing the lowest values.

The corresponding code is the following, where argument `degs = 2` is used to return the parameter estimates, which are shown below, with two decimal digits (default is `degs = 3`).

```
R> cumhazPlot(times, delta, font.lab = 4, cex.lab = 1.3, degs = 2, prnt = TRUE)

Parameter estimates

weibull
   Shape (se): 1.27 (0.07)
   Scale (se): 10.98 (0.62)
   AIC: 1304.05
   BIC: 1311.45

loglogistic
   Shape (se): 1.94 (0.11)
   Scale (se): 7.18 (0.42)
   AIC: 1272.48
   BIC: 1279.89

lognormal
Location (se): 1.99 (0.06)
   Scale (se): 0.88 (0.05)
   AIC: 1266.9
   BIC: 1274.31

gumbel
Location (se): 6.38 (0.33)
   Scale (se): 4.95 (0.3)
   AIC: 1342.49
   BIC: 1349.9

logistic
Location (se): 8.51 (0.43)
   Scale (se): 3.84 (0.23)
   AIC: 1430.25
   BIC: 1437.66

normal
Location (se): 9.89 (0.49)
   Scale (se): 7.54 (0.38)
   AIC: 1456.22
   BIC: 1463.63
```

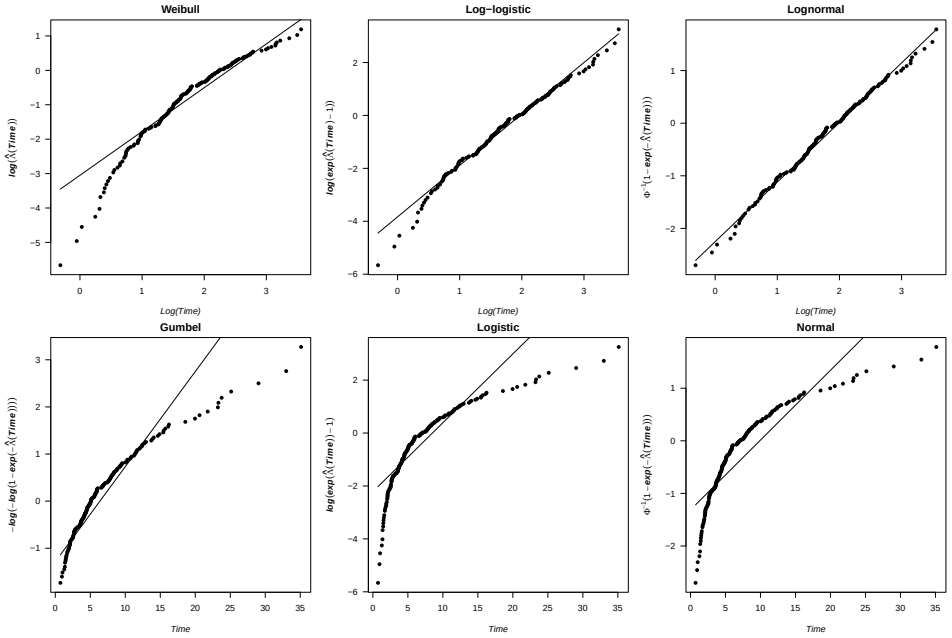


Figure 3: Cumulative hazard plots for six different distribution applied to a right-censored sample from a log-normal distribution. The plots are drawn with the `cumhazPlot()` function.

If users want to assess the goodness of fit of the exponential and the beta distribution and compare these with the fit of the log-normal distribution, they must specifically select these distributions, as shown in the following example. Notice that users need to provide the limits for the beta distribution, as the default domain, $(0, 1)$, does not cover the range of the sample's survival times. In this example, plots are drawn using the **ggplot2** package (`ggp = TRUE`), and parameter estimates are not displayed on the screen. The resulting plots are shown in Figure 4, where we observe that neither the exponential nor the beta distribution fits the data well. The corresponding `R` code is the following.

```
R> cumhazPlot(times, delta, distr = c("exponential",  "beta", "lognormal"),
+    betaLimits = c(0, 100), ggp = TRUE)
```

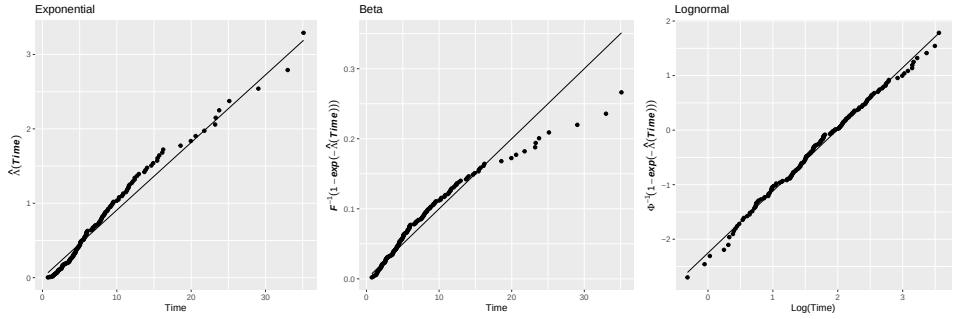


Figure 4: Cumulative hazard plots for the exponential, beta, and log-normal distribution applied to a right-censored sample from a log-normal distribution. The domain of the beta distribution is $(0, 100)$.

Function `kmPlot()`

The `kmPlot()` function allows users to graphically compare the nonparametric Kaplan-Meier estimate of the survival function for a right-censored or complete sample of survival times with parametric estimates based on different models. For this purpose, each parametric estimate is individually overlaid on the Kaplan-Meier estimator of $S(t)$. Likewise the `cumhazPlot()` function, this is done by default for the Weibull, Gumbel, log-logistic, logistic,

log-normal, and normal distributions, with the option to also include the exponential and beta distributions. The plots are generated using the **graphics** package by default, but setting `ggp = TRUE` enables the use of the **ggplot2** package. With both options, the pointwise 95% confidence intervals are plotted.

In the example shown in Figure 5, the estimated survival functions for the six default distributions are overlaid on the nonparametric estimate of $S(t)$. As expected, the log-normal distribution exhibits the best fit to the data. However, based on these plots, the log-logistic distribution might also be considered as a parametric model that fits the sample data well.

The R code used to draw the Kaplan-Meier plots in Figure 5 is the following:

```
R> kmPlot(times, delta, ggp = TRUE)
```

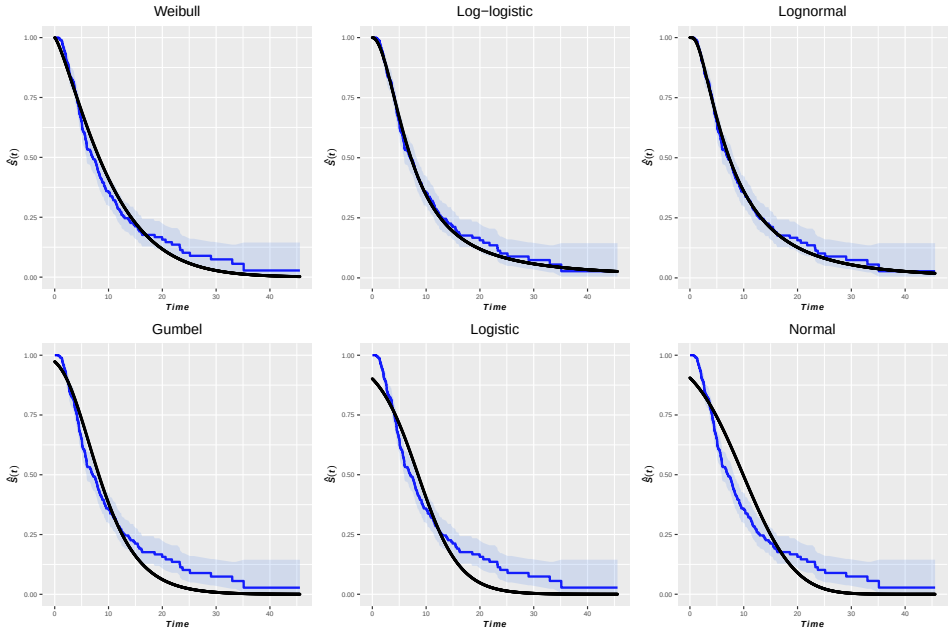


Figure 5: Kaplan-Meier plots for six different distribution applied to a right-censored sample from a log-normal distribution. The shaded area represent the pointwise 95% confidence intervals of the survival function. The plots are drawn with the `kmPlot()` function.

Goodness-of-fit test functions

In this section, we briefly describe and illustrate the use of the five functions in the **GofCens** package that implement the tests presented in the 'Methods' section.

The basic `print` methods of all functions return the test statistic and the p-value, whereas the `summary` methods do also provide the maximum likelihood estimates for the parameters of the hypothesized distribution $F_0(t;\theta)$ as well as the values of both the AIC and BIC. The functions have also in common that they offer the option to test the null hypothesis for specific parameter values of F_0, i.e., $F_0(t;\theta^*)$. Each function returns an object of its own class, enabling the use of object-specific `print` and `summary` methods.

Function `KScens()`

The `KScens()` function enables users to assess the goodness of fit using the Kolmogorov-Smirnov test adapted to right-censored data. It supports the eight distributions in Table 2 as well as any other distribution for which the corresponding p*name*, d*name*, and r*name* functions are implemented. The function computes the test statistic in (1) and, by default, estimates the p value using bootstrap methods.

As illustrated below with the data from the generated right-censored sample, the function returns, by default, a list with two elements:

- The null distribution.
- The test result, which includes the Kolmogorov-Smirnov test statistic (`A`) and the p value.

Additionally, the `summary` method provides two more elements:

- The maximum likelihood estimates of the parameters of the distribution under study.
- The values of the AIC and BIC.

In the illustration, we run the `KScens()` function to assess the goodness of fit of the log-normal and the Weibull distributions, first using bootstrapping to approximate the p value, and second calculating the p value following the results in Fleming et al. (1980). Additionally, in the second example, we apply the function `summary` method, which returns the parameter estimates, their standard errors and both the AIC and BIC. As shown, the p value obtained for the log-normal distribution is, as expected, larger (`0.115`) than the p value (`0.042`) for the Weibull distribution.

```
R> set.seed(123)
R> KScens(times, delta, distr = "lognormal")

Null hypothesis: the data follows a lognormal distribution

KS Test results:
      A p-value
  0.781   0.115

R> set.seed(123)
R> summary(KScens(times, delta, distr = "weibull", boot = FALSE))

Distribution: weibull

KS Test results:
      A p-value
  1.391   0.042

Parameter estimates (se):
shape             scale
1.273 (0.067)     10.98 (0.621)

AIC: 1304.046
BIC: 1311.453
```

Functions `CvMcens()` and `ADcens()`

The `CvMcens()` function calculates the test statistic of the Cramér-von Mises test in (3), while the `ADcens()` function calculates the test statistic of the Anderson-Darling test in (4), and both employ bootstrap techniques to estimate the corresponding p values. These functions share the same arguments, allowing users to specify the number of bootstrap samples, among other options. The default number of bootstrap samples for both functions is 999, which may result in lengthy computation times for large sample sizes. In such cases, users might consider reducing the number of bootstrap samples. For example, in the illustrations of the `CvMcens()` function below, with $n = 300$ and 999 bootstrap samples, the computation

times are approximately 15 seconds using R version 4.5.0 on Windows 10 x64 (build 19045) with an Intel(R) Core(TM) i5-8500 processor (3.00 GHz).

The null hypotheses in the illustrations remain the same as before, as do the conclusions: we would conclude that the data may come from a log-normal distribution, but not from a Weibull distribution.

```
R> set.seed(123)
R> summary(CvMcens(times, delta, distr = "lognormal"))

Distribution: lognormal

CvM Test results:
    CvM p-value
  0.054   0.450

Parameter estimates (se):
location          scale
1.988 (0.058)     0.883 (0.045)

AIC: 1266.899
BIC: 1274.307

R> set.seed(123)
R> summary(CvMcens(times, delta, distr = "weibull"))

Distribution: weibull

CvM Test results:
    CvM p-value
  0.376   0.001

Parameter estimates (se):
shape             scale
1.273 (0.067)     10.98 (0.621)

AIC: 1304.046
BIC: 1311.453
```

Applying the function `ADcens()` in the same way as `CvMcens()`, the p values are similar: $p = 0.52$ in the case of the log-normal distribution and $p = 0.001$ with the Weibull distribution.

Note that due to the use of resampling methods, with both functions it is advisable to previously set the seed of R's random number generation in order to guarantee reproducibility of the test results.

As mentioned before, users can also test the null hypothesis that the data come from a specific choice of the distribution under study, i.e., $H_0\colon F(t) = F_0(t;\theta^*)$. While this may be less relevant in practical applications, it can be useful for examining the properties of these tests. To perform such a test, users need to specify particular values for the distribution's parameters. For example, below we test the null hypothesis that the data follow a log-normal distribution with location and scale parameters $\mu = 2$ and $\beta = 1.5$. In this case, as shown, the Anderson-Darling test clearly rejects the null hypothesis (`p = 0.001`). Note that the output now includes both the specified parameter values for the null hypothesis and the maximum likelihood estimates. In addition, we use the `outp` argument of the `summary.ADcens()` function to change the format of the output.

```
R> set.seed(123)
R> summary(ADcens(times, delta, distr = "lognormal",
```

```
+    params0 = list(location = 2, scale = 1.5)), outp = "table")

Distribution: lognormal

Null hypothesis:
--------- | ---------
Parameter | Value
--------- | ---------
location  | 2
scale     | 1.5
--------- | ---------

AD Test results:
------- | -------
Metric  | Value
------- | -------
AD      | 9.536
p-value | 0.001
------- | -------

Parameter estimates:
--------- | --------- | ---------
Parameter | Value     | s.e.
--------- | --------- | ---------
location  | 1.988     | 0.058
scale     | 0.883     | 0.045
--------- | --------- | ---------

AIC: 1266.899
BIC: 1274.307
```

Function `gofcens()`

Rather than performing each of the three tests individually, users can run the Kolmogorov-Smirnov, Cramér-von Mises, and Anderson-Darling tests simultaneously with the `gofcens()` function, which calls each corresponding test function. This approach is especially useful for comparing the tests in a specific context or in a larger study, such as comparing their statistical power. However, a potential drawback is longer computation times, as each test function employs bootstrap methods to estimate the p values. In the following examples, using the same data and null hypotheses as before, the computation time is approximately 20 seconds on `R` version 4.5.0, running on Windows 10 x64 (build 19045) with an Intel(R) Core(TM) i5-8500 processor (3.00 GHz). Note that in the second example, we modify options in the `summary.gofcens()` function: the number of bootstrap samples is displayed, while the AIC and BIC values are omitted.

```
R> set.seed(123)
R> gofcens(times, delta, distr = "lognormal")

Null hypothesis: the data follows a lognormal distribution

Test statistics
   KS   CvM    AD
0.781 0.054 0.500

p-values
   KS   CvM    AD
```

```
0.115 0.450 0.520

R> set.seed(123)
R> summary(gofcens(times, delta, distr = "weibull"), print.AIC = FALSE,
+    print.BIC = FALSE, print.infoBoot = TRUE)

Distribution: weibull

Test statistics
   KS   CvM    AD
1.391 0.376 2.828

p-values
   KS   CvM    AD
0.001 0.001 0.001

Parameter estimates (se):
shape             scale
1.273 (0.067)     10.98 (0.621)

Number of bootstrap samples: 999
```

Function `chisqcens()`

The `chisqcens()` function implements the chi-squared type test of Kim (1993) explained in Section 'Chi-squared type tests'. Likewise the `KScens()`, `CvMcens()`, and `ADcens()` functions, the computation of the p value is achieved with bootstrap techniques using `BS = 999` as default value for the number of bootstrap samples. The function's default output includes the hypothesized distribution, the value of the test statistic, and the corresponding p value. The extended output provided by the `summary.chisqcens()` function also includes the parameter estimates, the the AIC and BIC values as well as two values related to the number of random cells: the number initially chosen by the user (`Original`) and the final number used in the analysis (`Final`). The latter may be smaller due to the presence of right-censored data. For example, in the illustrations below using right-censored data from a log-normal distribution, we set `M = 8` random cells, but this number is reduced to `M = 7` due to censoring.

```
R> set.seed(123)
R> chisqcens(times, delta, M = 8, distr = "lognormal")

Null hypothesis: the data follows a lognormal distribution

Chi-squared Test results:
Statistic   p-value
    3.123     0.154


R> set.seed(123)
R> summary(chisqcens(times, delta, M = 8, distr = "weibull"))

Distribution: weibull

Chi-squared Test results:
Statistic   p-value
    8.599     0.002

Parameter estimates (se):
```

```
shape             scale
1.273 (0.067)     10.98 (0.621)

Cell numbers:
Original    Final
       8        7

AIC: 1304.046
BIC: 1311.453
```

Again, based on the outputs, we would choose the log-normal distribution instead of the Weibull distribution, because its value of the test statistic is clearly smaller (3.123 vs 8.599) and the p value is far larger ($p = 0.154$ vs $p = 0.002$).

The following illustration highlights two features of the goodness-of-fit test functions in the **GofCens** package. First, all functions return structured objects that can be stored as standard objects for further use in R. Second, users can test the goodness-of-fit of distributions different from the eight distributions in Table 2. In this example, we apply the `chisqcens()` function using the gamma distribution. As shown, the parameter estimates are reported with the default names `theta1` and `theta2`, and the result is an object of class `chisqcens`, further processed with the `summary` method to display detailed results.

```
R> set.seed(123)
R> result <- summary(chisqcens(times, delta, M = 8, distr = "gamma"))
R> result

Distribution: gamma

Chi-squared Test results:
Statistic   p-value
    6.176     0.006

Parameter estimates (se):
theta1            theta2
1.65 (0.144)     0.165 (0.019)

Cell numbers:
Original    Final
       8        7

AIC: 1293.124
BIC: 1300.532

R> class(result)

[1] "summary.chisqcens" "chisqcens"
```

4 Real data example: Survival times of retired NBA players

In this section, we apply several functions of the **GofCens** package to determine the parametric model that best fits the survival times of former NBA players.

The data frame `nba` comes with the GofCens package and contains the survival times (variable `survtime`) of all 3962 former players of the National Basketball Association (NBA) from its inception until July 2019. These data have been published and analyzed by Martínez et al. (2022), where survival times are measured as the elapsed time (in years) from the end of the NBA career until either death (`cens == 1`) or July 31, 2019 (`cens == 1`). By this

date, 864 (21.8%) of the former players had died, with uncensored post-NBA survival times ranging from a few days until nearly 70 years. The estimated median survival time is 54.1 years as shown below in the output of the function npsurv of the **rms** package (Harrell Jr, 2025) and Figure 6, which shows the Kaplan-Meier estimate of the survival function.

```
R> data("nba")
R> library("rms")
R> npsurv(Surv(survtime, cens) ~ 1, nba)

Call: npsurv(formula = Surv(survtime, cens) ~ 1, data = nba)
        n events median 0.95LCL 0.95UCL
[1,] 3962    864   54.1    53.1    55.1
```

The R code to draw the plot in Figure 6 is the following:

```
R> par(las = 1, cex.lab = 1.3, cex.axis = 1.2, font.lab = 4, font.axis = 2,
+    mar = c(5, 5, 2, 2), yaxs = "i", xaxs = "i")
R> survplot(npsurv(Surv(survtime, cens) ~ 1, nba), lwd = 3,
+    xlab = "Years after NBA career", time.inc = 5, col.fill = grey(0.6),
+    ylab = "Estimated survival probability", xlim = c(0, 75))
R> abline(h = 0.5, lwd = 2, lty = 2)
```

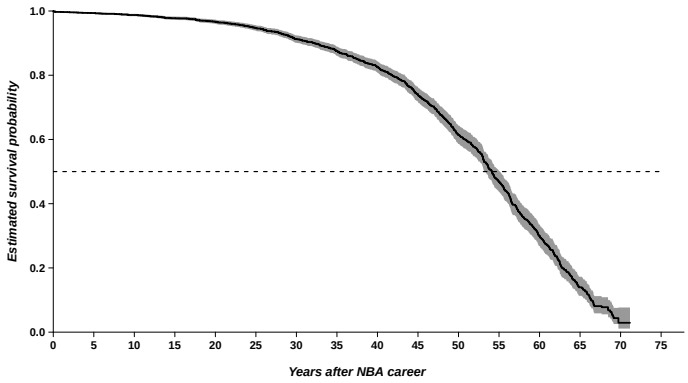


Figure 6: Survival function of times from the end of the NBA career until death among retired players. The shaded area represents the confidence bands of the survival function.

In order to model parametrically the survival times and estimate the median, we need to know which distribution is the most appropriate distribution. For this purpose, we take advantage of the cumhazPlot() function, which provides the six cumulative hazard plots shown in Figure 7:

```
R> cumhazPlot(Surv(survtime, cens) ~ 1, nba, font.lab = 4, cex.lab = 1.3,
+    lwd = 3, colour = "blue")
```

According to the plots in Figure 7, the logistic distribution fits reasonably well to the data, even though the corresponding plot does not show a completely straight line of the points. In addition, we could also consider the normal distribution for parametric analyses of the survival times. To choose one of either distributions, we run the following code to draw the probability and quantile-quantile plots that are shown in the left and right panels of Figure 8 with the probPlot() function.

```
R> probPlot(Surv(survtime, cens) ~ 1, nba, distr = "logistic", ggp = TRUE)
R> probPlot(Surv(survtime, cens) ~ 1, nba, distr = "normal", ggp = TRUE)
```

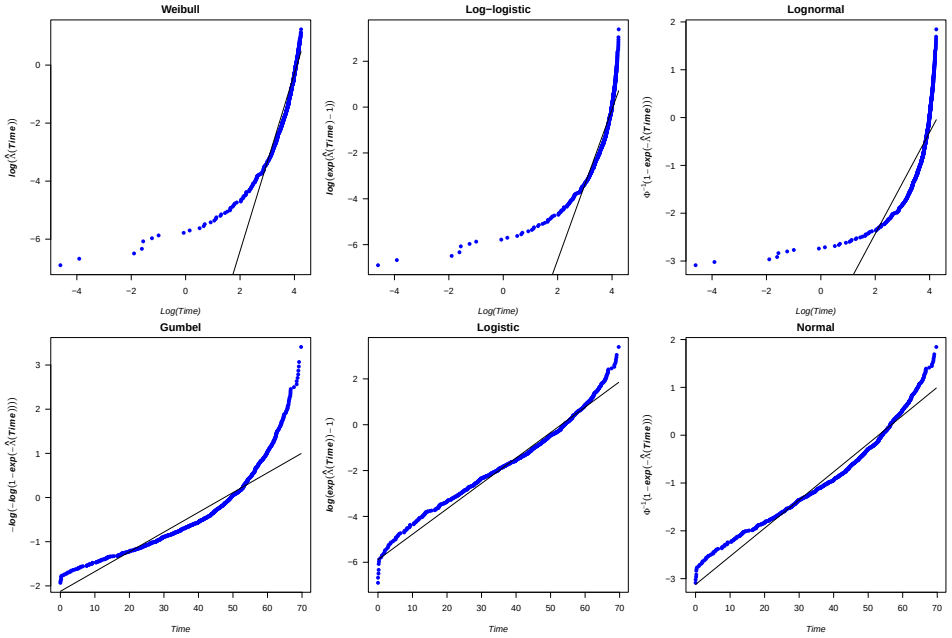


Figure 7: Cumulative hazard plots for six different distributions applied to the survival times of former NBA players. The plots are generated with the `cumhazPlot()` function.

According to the plots in Figure 8, the logistic distribution appears to be a slightly better choice than the normal distribution, as the points in all four plots on the right panel show some curvature.

Note that in the previous function calls, we used the `formula` versions of both functions, which allows for the inclusion of the data frame `nba` in the argument list, which simplifies their use.

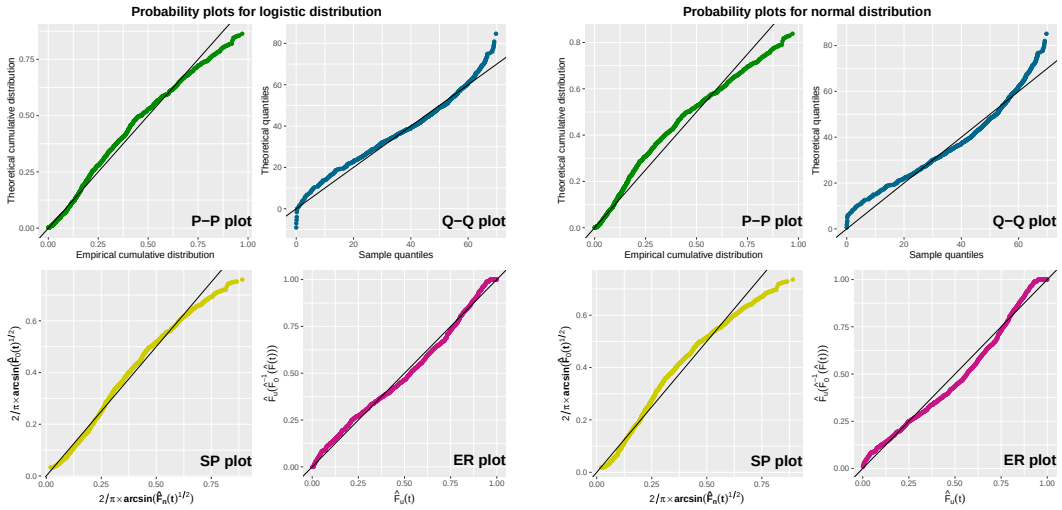


Figure 8: Logistic and normal probability plots for the survival times of former NBA players. The plots are drawn with the `probPlot()` function.

Finally, we apply the `gofcens()` function to both distributions in order to complement the plots with the results of the Kolmogorov-Smirnov, Cramér-von Mises, and Anderson-Darling tests.

```
R> set.seed(123)
R> summary(gofcens(Surv(survtime, cens) ~ 1, nba, distr = "logistic"))

Distribution: logistic
```

```
Test statistics
    KS    CvM     AD
 2.099  1.277 13.849

p-values
   KS   CvM    AD
0.001 0.009 0.002

Parameter estimates (se):
location          scale
53.077 (0.457)     9.013 (0.218)

AIC: 8460.297
BIC: 8472.866

R> set.seed(123)
R> summary(gofcens(Surv(survtime, cens) ~ 1, nba, distr = "normal"))

Distribution: normal

Test statistics
    KS    CvM     AD
 2.736  2.666 22.957

p-values
   KS   CvM    AD
0.001 0.002 0.001

Parameter estimates (se):
location          scale
53.004 (0.518)     16.977 (0.368)

AIC: 8512.579
BIC: 8525.148
```

Following the test results, we would reject the null hypothesis in both cases: the Kolmogorov-Smirnov, Cramér-von Mises, and Anderson-Darling tests all yield small p-values, indicating that neither the logistic nor the normal distribution are adequate models for the survival times of former NBA players. This result is not unexpected given the large sample size, which increases the power of the tests and makes even minor deviations from the model detectable. Despite the rejection, the diagnostic plots in Figures 7 and 8 suggest that the logistic distribution provides a somewhat better fit to the data than the normal distribution. Therefore, it seems reasonable to use the logistic model as a practical approximation. Based on the maximum likelihood estimates, the fitted parameters are $\hat{\mu} = 53.08$ and $\hat{\beta} = 9.01$, yielding an estimated median survival time of approximately 53.1 years, which is close to the non-parametric estimate of 54.1 years.

5 Conclusion

In this paper we have presented the R package **GofCens**, whose functions offer both analytical methods and graphical tools to assess the goodness of fit for right-censored lifetime data. While the graphical functions can be used to assess the goodness of fit for the eight distributions listed in Table 2, which represent the most common ones in survival analysis, the goodness-of-fit test functions are more flexible and can be applied to any distribution

available in R. **GofCens** incorporates new functions for goodness-of-fit analyses while unifying functions that are otherwise scattered across various R packages. As shown with several examples, this package is a user-friendly tool that enables users to determine the most suitable parametric distribution for analyzing their data.

For the analysis of a given data set in practice, it is important to highlight the usefulness of including probability or cumulative hazards plots alongside the goodness-of-fit tests provided in **GofCens**. While statistical tests offer summary statistics, such plots offer a comprehensive view of the data, thus revealing peculiarities that might escape detection through summary statistics alone. Moreover, the importance of selecting a specific distribution depends on the context of the analysis. For example, precise distributional assumptions may be less critical when evaluating test statistics, but become essential when estimating quantiles or making extrapolations. The selection of any of the four tests presented herein is, to some extent, subjective, influenced by personal preference and convenience, as none of them emerges as optimal for all censoring situations, parametric families, or sample sizes.

In recent years, additional goodness-of-fit techniques for right-censored data have been developed, including transformation-based procedures (Goldmann et al., 2015) and modifications of classical tests using pseudo-complete samples or randomization strategies (Balakrishnan et al., 2015). While these methods can offer improved performance in specific settings, they often involve specialized implementation or focus on narrower applications. In contrast, **GofCens** implements established, flexible approaches with broad applicability and interpretability, supporting classical tests enhanced by resampling and graphical diagnostics.

Feasible extensions of this package include expanding the range of distributions supported by the graphical functions. Since, for example, the `cumhazPlot()` function requires distribution-specific transformations of the cumulative hazard function, such an extension is not straightforward. Additional developments could involve adapting its functions to handle left-truncated and interval-censored data. Furthermore, exploring residual-based methods to assess the parametric assumptions an accelerated failure time model may be a promising direction for further development.

Acknowledgements

M. Besalú's, G. Gómez Melis' and K. Langohr's research was funded by MICIU/AEI/10.13039/2501100011033 and FEDER, UE [PID2023-148033OB-C21] and by Agència de Gestió d'Ajuts Universitaris i de Recerca [2021 SGR 01421] (GRBIO) from Generalitat de Catalunya (Spain).

We thank the anonymous referees for their valuable and significant feedback.

References

B. Auguie. *gridExtra: Miscellaneous Functions for "Grid" Graphics*, 2017. URL https://CRAN.R-project.org/package=gridExtra. R package version 2.3. [p106]

N. Balakrishnan, E. Chimitova, and M. Vedernikova. An empirical analysis of some non-parametric goodness-of-fit tests for censored data. *Communications in Statistics - Simulation and Computation*, 44(4):1101–1115, 2015. [p121]

M. Besalú Mayol. Testing goodness-of-fit of parametric survival models for right-censored data. Master's thesis, School of Mathematics and Statistics, Universitat Politècnica de Catalunya - Barcelona-Tech, 2016. URL https://upcommons.upc.edu/handle/2117/88820. [p107]

A. Canty and B. Ripley. *boot: Bootstrap R (S-Plus) Functions*, 2024. URL https://CRAN.R-project.org/package=boot. R package version 1.3-31. [p107]

M. Capasso, L. Alessi, M. Barigozzi, and G. Fagiolo. On approximating the distributions of goodness-of-fit test statistics based on the empirical distribution function: The case of unknown parameters. *Advances in Complex Systems*, 12(2):157–167, 2009. [p101, 104]

M. Delignette-Muller and C. Dutang. fitdistrplus: An R package for fitting distributions. *Journal of Statistical Software*, 64(4):1–34, 2015. [p100, 106]

M.-L. Delignette-Muller, C. Dutang, R. Pouillot, J.-B. Denis, and A. Siberchicot. *fitdistrplus: Help to Fit of a Parametric Distribution to Non-Censored or Censored Data*, 2025. URL https://CRAN.R-project.org/package=fitdistrplus. R package version 1.2-2. [p100, 106]

C. Dutang, V. Goulet, and M. Pigeon. actuar: An R package for actuarial science. *Journal of Statistical Software*, 25(7):38, 2008. URL https://www.jstatsoft.org/v25/i07. [p106]

A. Febrer Galvany. Analytical and graphical goodness of fit methods for parametric survival models with right-censored data. Master's thesis, School of Mathematics and Statistics, Universitat Politècnica de Catalunya - Barcelona-Tech, 2015. URL https://upcommons.upc.edu/handle/2099.1/26129. [p107]

T. R. Fleming, J. R. O'Fallon, P. C. O'Brien, and D. P. Harrington. Modified Kolmogorov-Smirnov test procedures with application to arbitrarily right-censored data. *Biometrics*, 36 (4):607–625, 1980. [p100, 102, 113]

D. García Carrasco. Goodness-of-fit R package for right-censored data. Master's thesis, School of Mathematics and Statistics, Universitat Politècnica de Catalunya - Barcelona-Tech, 2017. URL https://upcommons.upc.edu/handle/2117/106177. [p107]

C. Goldmann, B. Klar, and S. Meintanis. Data transformations and goodness-of-fit tests for type-II right censored samples. *Metrika*, 78:59–83, 2015. [p121]

V. Goulet, C. Dutang, M. Pigeon, et al. *actuar: Actuarial Functions and Heavy Tailed Distributions*, 2025. URL https://CRAN.R-project.org/package=actuar. R package version 3.3-5. [p106]

F. E. Harrell Jr. *rms: Regression Modeling Strategies*, 2025. URL https://CRAN.R-project.org/package=rms. R package version 8.0-0. [p118]

A. Janssen. Global power functions of goodness-of-fit tests. *Annals of Statistics*, 28:239–253, 2000. [p100]

J. H. Kim. Chi-square goodness-of-fit tests for randomly censored data. *The Annals of Statistics*, 21(3):1621–1639, 1993. [p103, 116]

J. A. Koziol and D. P. Byar. Percentage points of the asymptotic distributions of one and two sample K-S statistics for truncated or censored data. *Technometrics*, 17(4):507–510, 1975. [p100, 102]

J. A. Koziol and S. B. Green. A Cramér-von Mises statistic for randomly censored data. *Biometrika*, 63(3):465–474, 1976. [p100, 102]

K. Langohr, M. Besalú, M. Francisco, A. Garcia, and G. Gómez. *GofCens: Goodness-of-Fit Methods for Right-Censored Data*, 2025. URL https://CRAN.R-project.org/package=GofCens. R package version 1.5. [p101]

E. L. Lehmann and J. P. Romano. *Testing Statistical Hypotheses*. Springer Texts in Statistics. Springer-Verlag, New York, 3rd edition, 2005. [p100]

J. A. Martínez, K. Langohr, J. Felipo, L. Consuegra, and M. Casals. Data set on mortality of National Basketball Association (NBA) players. *Data in Brief*, 45:108615, 2022. [p107, 117]

D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2024. URL https://CRAN.R-project.org/package=e1071. R package version 1.7-16. [p101]

J. R. Michael. The stabilized probability plot. *Biometrika*, 70(1):11–17, 1983. [p100, 105]

D. Mihalko and D. Moore. Chi-square tests of fit for type II censored data. *Annals of Statistics*, 8(3):625–644, 1980. [p100]

S. P. Millard. *EnvStats: An* R *Package for Environmental Statistics*. Springer-Verlag, New York, 2013. ISBN 978-1-4614-8455-4. [p101]

S. P. Millard and A. Kowarik. *EnvStats: Package for Environmental Statistics, Including US EPA Guidance*, 2025. URL https://CRAN.R-project.org/package=EnvStats. R package version 3.1.0. [p101]

A. N. Pettitt and M. A. Stephens. Modified Cramér-von Mises statistics for censored data. *Biometrika*, 63(2):291–298, 1976. [p100, 102, 103]

P. Royston. A toolkit for testing for non-normality in complete and censored samples. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 42(1):37–43, 1993. [p101]

T. M. Therneau and P. M. Grambsch. *Modeling Survival Data: Extending the Cox Model*. Springer-Verlag, New York, 2000. ISBN 0-387-98784-3. [p106]

L. Waller and B. Turnbull. Probability plotting with censored data. *American Statistician*, 46 (1):5–12, 1992. [p100, 105]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p106]

H. Wickham, W. Chang, L. Henry, et al. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2025. URL https://CRAN.R-project.org/package=ggplot2. R package version 3.5.2. [p106]

Mireia Besalú
Department of Statistics and Operations Research, Biostatistics and Bioinformatics Group and Institute for Research and Innovation in Health (IRIS)
Universitat Politècnica de Catalunya· BarcelonaTech (UPC)
Jordi Girona 1-3, 08034 Barcelona
Spain
https://orcid.org/0000-0003-0473-2404
mireia.besalu@upc.edu

Klaus Langohr
Department of Statistics and Operations Research, Biostatistics and Bioinformatics Group and Institute for Research and Innovation in Health (IRIS)
Universitat Politècnica de Catalunya· BarcelonaTech (UPC)
Jordi Girona 1-3, 08034 Barcelona
Spain
https://orcid.org/0000-0001-7075-9192
klaus.langohr@upc.edu

Matilde Francisco
Department of Statistics and Operations Research and Biostatistics and Bioinformatics Group,
Universitat Politècnica de Catalunya· BarcelonaTech (UPC)
Jordi Girona 1-3, 08034 Barcelona
Spain
https://orcid.org/0009-0009-1982-0547
matilde.martins.da.palma@upc.edu

Arnau Garcia-Fernández
Department of Statistics and Operations Research and Biostatistics and Bioinformatics Group

Universitat Politècnica de Catalunya· BarcelonaTech (UPC)
Jordi Girona 1-3, 08034 Barcelona
Spain
https://orcid.org/0009-0009-7370-6980
arnau.garcia.fernandez@upc.edu

Guadalupe Gómez Melis
Department of Statistics and Operations Research, Biostatistics and Bioinformatics Group and Institute for Research and Innovation in Health (IRIS)
Universitat Politècnica de Catalunya· BarcelonaTech (UPC)
Jordi Girona 1-3, 08034 Barcelona
Spain
https://orcid.org/0000-0003-4252-4884
lupe.gomez@upc.edu

moonboot: An R Package Implementing m-out-of-n Bootstrap Methods

by Christoph Dalitz and Felix Lögler

Abstract The m-out-of-n bootstrap is a possible workaround to compute confidence intervals for bootstrap inconsistent estimators, because it works under weaker conditions than the n-out-of-n bootstrap. It has the disadvantage, however, that it requires knowledge of an appropriate scaling factor tau(n) and that the coverage probability for finite n depends on the choice of m. This article presents an R package **moonboot** which implements the computation of m-out-of-n bootstrap confidence intervals and provides functions for estimating the parameters tau(n) and m. By means of Monte Carlo simulations, we evaluate the different methods and compare them for different estimators.

1 Introduction

The *bootstrap* is a resampling method introduced by Efron (1979), which repeatedly simulates samples of the same size n as the observed data by n-fold drawing with replacement. Let us call this method the *n-out-of-n bootstrap*, in order to distinguish it from different sampling schemes. For each of these samples, the estimator T of interest is computed, thereby simulating a sample $T_1^*, \ldots, T_R^*$ for the distribution of T, where R is the number of bootstrap repetitions. For sufficiently smooth estimators T, this bootstrap distribution asymptotically approaches the true distribution of T (Giné and Zinn, 1990; Shao and Tu, 1995), and it can thus be used to construct confidence intervals for T. The coverage probability of confidence intervals based on flipping the quantiles of the bootstrapped distribution $T_1^*, \ldots, T_R^*$ at the point estimate ("basic bootstrap") can be shown to be first order accurate, i.e. up to $O(n^{-1/2})$. Under certain conditions, it is even possible to make the confidence intervals second order accurate, i.e. up to $O(n^{-1})$, by studentized sampling ("bootstrap-t") or accelerated bias correction ("BC_a bootstrap") (Hall, 1988; DiCiccio and Efron, 1996; Davison and Hinkley, 1997). Compared to other methods for estimating confidence intervals, the bootstrap is more versatile because it neither relies on a likelihood function, nor does it require asymptotic normality. The bootstrap has thus become a standard technique for estimating confidence intervals, and it is even included in vanilla R with the package **boot** (Canty and Ripley, 2021), which implements five different methods in `boot.ci()`.

There are, however, *bootstrap inconsistent* estimators, i.e., estimators for which the n-out-of-n bootstrap fails to yield confidence intervals with an asymptotically correct coverage probability. Examples include extreme order statistics (Bickel et al., 1997), the Grenander estimator of a monotonous density (Sen et al., 2010), Chernoff's estimator of the mode and Tukey's shorth (Léger and MacGibbon, 2006), or Chatterjee's rank correlation index (Lin and Han, 2024; Dalitz et al., 2024). See the discussion by Lin and Han (2024) for a literature review and further examples. A possible workaround in these cases is the *m-out-of-n bootstrap*, which samples only $m < n$ observations. This can be done with or without replacement. Politis and Romano (1994), who introduced this method, only considered sampling without replacement, which they called *subsampling*. They have shown that the m-out-of-n bootstrap without replacement works under much weaker conditions than the n-out-of-n bootstrap, provided m is chosen such that, for $n \to \infty$, $m \to \infty$ and $m/n \to 0$. The only condition is that the estimator, suitably scaled by some factor τ_n, possesses a limit distribution, whereas no smoothness of the estimator or uniformity of the convergence are required. If the sampling is instead done with replacement, Bickel et al. (1997) have shown that a similar result requires additional restrictions on the estimator. Sampling without replacement thus works under weaker conditions than sampling with replacement, and, in the present article, we therefore only consider sampling without replacement. Our package

moonboot[1] supports sampling both with and without replacement by means of an argument `replace`, which is set to `FALSE` by default.

The wider applicability of the m-out-of-n bootstrap comes at a price, though. One drawback is that the scaling factor τ_n must be known. Bertail et al. (1999) explained that it must be chosen such that $\tau_n^2 \operatorname{Var}(T)$ converges to some constant, and they also suggested a method to estimate the scaling factor with another bootstrap. For root-n consistent estimators, it is $\tau_n = \sqrt{n}$, but not all estimators are root-n consistent, especially if they are bootstrap inconsistent, and even if root-n consistency is conjectured it can be difficult to prove (Lin and Han, 2024). Moreover, the m-out-of-n bootstrap can have less than first order accuracy: Politis and Romano (1994) gave a theoretical example with accuracy $O(n^{-1/3})$ for an optimal choice of m, and even worse for other choices of m. That the accuracy depends on the choice of m was demonstrated, too, in a simulation study by Kleiner et al. (2014). And for the sample quantiles and sampling with replacement, Arcones (2003) has shown that the choice $m \propto n^{2/3}$ is optimal. This raises the problem of how to choose m, and several heuristics have been suggested in the literature (Politis et al., 1999; Götze and Račkauskas, 2001; Bickel and Sakov, 2008; Chung and Lee, 2001; Lee and Yang, 2020).

Due to these shortcomings, the usual n-out-of-n bootstrap is generally preferable for bootstrap consistent estimators. However, there *are* bootstrap inconsistent estimators and there is thus need for an R package that facilitates the application of the m-out-of-n bootstrap by providing the required functions. To this end, we present a new package **moonboot**, evaluate the algorithms implemented therein by means of Monte Carlo simulations, and give recommendations and use cases for their application. The name of the package was inspired by an article by Götze and Račkauskas (2001), who abbreviated the m-out-of-n bootstrap as "moon bootstrap".

2 The m-out-of-n bootstrap

Let $X_1, \ldots, X_n$ be i.i.d. random variables, θ be some parameter of their underlying distribution, and $T_n = T_n(X_1, \ldots, X_n)$ be an estimator for θ. Let us additionally assume[2] that $E(T_n^2) < \infty$. The m-out-of-n bootstrap requires that, for some scaling factor τ_n which behaves for $n \to \infty$ as

$$\frac{\tau_{m(n)}}{\tau_n} \to 0 \quad \text{for} \quad m(n) \to \infty \quad \text{with} \quad \frac{m(n)}{n} \to 0, \tag{1}$$

the estimator converges in distribution to some limit law when centered around the true parameter θ and scaled with τ_n:

$$S_n = \tau_n(T_n - \theta) \overset{n\to\infty}{\longrightarrow} S \text{ in distribution} \tag{2}$$

Note that condition (1) implies that $\tau_n \to \infty$, and thus the convergence (2) requires that T_n converges to θ in probability (in other words: T_n must be consistent), because it follows from (2) that

$$P(|T_n - \theta| > \varepsilon) = F_n(-\tau_n\varepsilon) + (1 - F_n(\tau_n\varepsilon)) \overset{n\to\infty}{\longrightarrow} 0$$

where F_n is the cumulative distribution function of S_n. Moreover, if additionally the second moment of S_n converges to some constant, the scaling factor τ_n is related to the rate of decrease of the variance of T_n:

$$V = \lim_{n\to\infty} \operatorname{Var}(S_n) = \lim_{n\to\infty} \operatorname{Var}(\tau_n T_n) \quad \Rightarrow \quad \operatorname{Var}(T_n) \sim \frac{V}{\tau_n^2} \quad \text{for} \quad n \to \infty \tag{3}$$

This relationship allows one to determine τ_n theoretically by means of an analytical calculation, or to estimate it by means of Monte Carlo simulations.

[1] This article used version 2.0.x, available, e.g., from https://github.com/cdalitz/moonboot.

[2] Politis and Romano (1994) did not make this assumption, but it is necessary for the estimation of the scaling factor τ_n with the method by Bertail et al. (1999).

Politis and Romano (1994) have shown that, under the assumptions (2) and (1), a confidence interval with asymptotic coverage probability $(1-\alpha)$ can be constructed as

$$\left[T_n - \frac{q(1-\alpha/2)}{\tau_n},\ T_n - \frac{q(\alpha/2)}{\tau_n}\right] \tag{4}$$

where $q(1-\alpha/2)$ and $q(\alpha/2)$ are the quantiles of the scaled bootstrap distribution $\tau_m(T_m^* - T_n)$, where T_m^* denotes the bootstrap samples obtained by m-fold drawing without replacement. In **moonboot**, this interval is implemented by `mboot.ci(...,method="politis")`.

If the variance of S_n converges to some value σ^2 and the limiting distribution of S is the *normal distribution* with standard deviation σ, then the m-out-of-n bootstrap can alternatively be used to estimate the variance σ^2 and compute the standard confidence interval

$$T_n \pm z_{1-\alpha/2}\,\hat{\sigma}_m^* \quad \text{with} \quad \hat{\sigma}_m^* = \frac{\tau_m}{\tau_n}\sqrt{\mathrm{Var}(T_m^*)} \tag{5}$$

In **moonboot**, this is implemented by `mboot.ci(...,method="norm")`. Although this interval is only reasonable for asymptotically normal estimators, its coverage probability can have a faster convergence to the nominal value than the interval (4) provided asymptotic normality indeed holds (Dalitz et al., 2024).

Sherman and Carlstein (2004) observed that the scaling with τ_m/τ_n in (4) can be omitted if the interval is centered around one of the T_m instead of T_n. They suggested using the first m samples for computing T_m, but this is an arbitrary choice and thus even the location of the confidence interval is not uniquely determined by the data. Moreover, the lack of scaling increases the interval length. The greater length is compensated by the higher volatility of the location so that the coverage probability still is close to the nominal value. This does not hold, however, for drawing without replacement as m approaches n, and a compromise must be made between small interval length and approximate coverage probability. Sherman and Carlstein (2004) suggested a heuristic method for choosing m on the basis of a double bootstrap, and they gave the rule of thumb to choose m between $n^{1/2}$ and $n^{2/3}$. However, due to the greater interval length, this method should only be used as a last resort if the convergence rate τ_n is unknown and cannot be estimated from the data. In **moonboot**, this method is implemented by `mboot.ci(...,method="sherman")` and the method for estimating m by `estimate.m.sherman()`.

2.1 Estimation of τ_n

According to Eq. (3), the formula for the scaling factor τ_n can be determined by an analytic investigation of $\mathrm{Var}(T_n)$, which can not only be difficult for some estimators, but it requires an ad hoc study of the specific estimator T_n under consideration. This thwarts the application of the m-out-of-n bootstrap out-of-the-box.

Fortunately, the relationship (3) makes it possible to estimate τ_n by another bootstrap (Bertail et al., 1999). If the variance $V_m = \mathrm{Var}(T_m^*)$ is estimated by sampling with different subsampling sizes m and τ_n is assumed to be of the form $\tau_n = n^\beta$, then the asymptotic relationship (3) becomes

$$\log V_m \approx -2\beta \log m + \log V \tag{6}$$

and β can be estimated with a least-squares fit. For the choice of the test values m, Bertail et al. (1999) suggested $m_i = n^{\gamma_i}$, but wrote that "the difficult problem of choosing the γ_i's requires more work on a case-by-case basis"[3]. This casts doubt on the whole method, because removing the necessity of a detailed asymptotic analysis of the estimator is the whole point of estimating τ_n from the data. Indeed, we have not found a sequence of values for the γ_i that worked equally well in all of our examples. We thus leave the choice of the γ-sequence as an option to the user and use the default of five values between 0.2 and 0.7, which worked reasonably well for some of our estimators.

[3]Note that Bertail et al. (1999) wrote n^{β_i} for the test values m_i, but we have renamed them to n^{γ_i} to avoid confusion with the power β that is to be estimated.

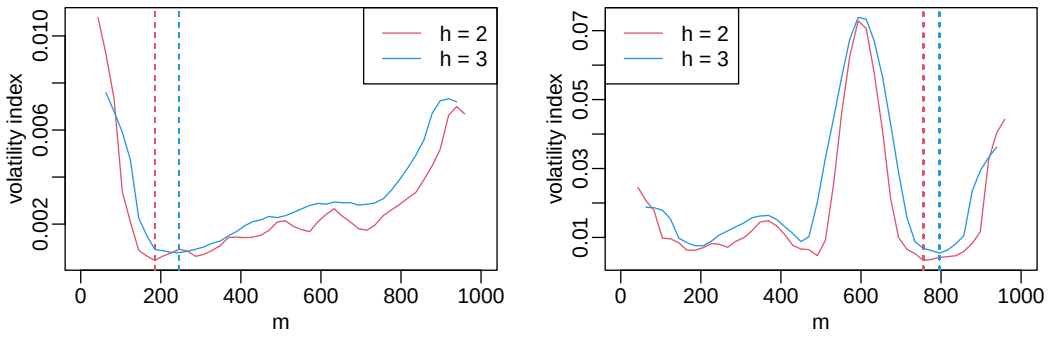


Figure 1: Volatility index after Politis et al. (1999) as a function of m for Chatterjee's correlation (*xicor*) and Tukey's shorth for $n = 1000$.

Bertail et al. (1999) suggested the alternative of using quantiles instead of the variance, but as the method involves taking logarithms, this makes no sense when the estimator tends to be negative or close to zero. As a workaround, (Politis et al., 1999) suggested replacing V_m in Eq. (6) with the average of quantile ranges

$$Q_m = \frac{1}{J}\sum_{j=1}^{J}\left(\text{quantile}(T_m^*, \alpha_j^{high}) - \text{quantile}(T_m^*, \alpha_j^{low})\right) \tag{7}$$

We have implemented this, too, with the choice $\alpha_j^{high} = 0.75 + j \cdot 0.05$ and $\alpha_j^{low} = 0.25 - j \cdot 0.05$ for $j = 0, \ldots, 4$, but the results in section 2.4.1 show that this makes almost no difference in comparison to using the variance.

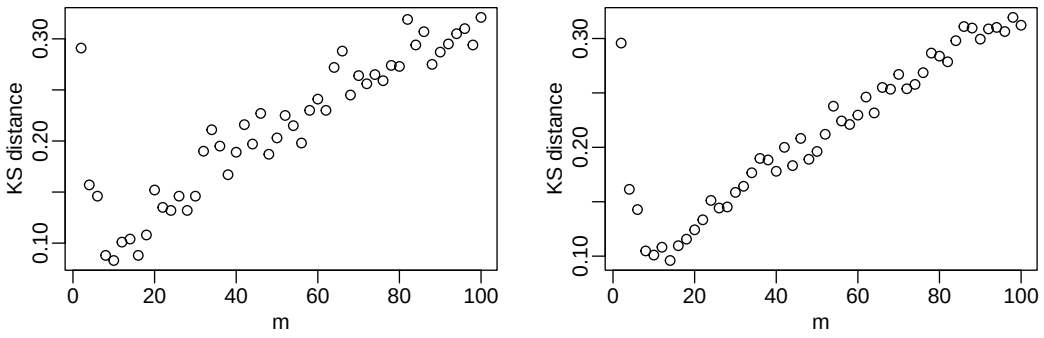
This method is implemented in **moonboot** by `estimate.tau()`, which is automatically called by `mboot.ci()` when no rate for the parameter `tau` is provided.

2.2 Choice of m

According to Politis and Romano (1994), the m-out-of-n bootstrap asymptotically works for any choice of m satisfying $m \to \infty$ and $m/n \to 0$ as $n \to \infty$. This, however, leaves a wide range of choices, e.g. $m = cn^\beta$ for any $\beta \in (0, 1)$, and the simulations in section 2.4 show that the convergence rate of the coverage probability depends on the choice of m. It might thus be desirable to choose m in a data-dependent way, for which different methods have been suggested in the literature (Politis et al., 1999; Chung and Lee, 2001; Götze and Račkauskas, 2001; Bickel and Sakov, 2008; Lee and Yang, 2020).

Politis et al. (1999, ch. 9.3.2) suggested a *minimum volatility method* for estimating m, which is based on the idea that there should be some range for m where its choice has little effect on the estimated confidence interval endpoints. The optimal m is that with the lowest "volatility" of the confidence interval, which is defined as the running standard deviation of the interval endpoints around each specific choice for m. Before computing the standard deviation, the interval endpoints should be smoothed out, too, by averaging over the endpoints computed for the neighboring choices for m.

The algorithm has two parameters, the window width h_{ci} for smoothing the interval end points and the window width h_σ for computing the running standard deviation. Politis et al. (1999) recommended $h_{ci} = h_\sigma = 2$ or 3, which corresponds to a range of 5 or 7 values. To avoid finding a minimum that is too close to n, as happens in Fig. 1b, we have restricted the search range to values $m < n/2$. For efficiency reasons, Politis et al. (1999) recommended not to try every m, but only a grid of equidistant values. Our implementation in the function `estimate.m.volatility()` in the package **moonboot** therefore only tries out a maximum of 50 values for m.



1. single sample 2. averaged over multiple samples

Figure 2: Kolmogorov distance between the CDFs of S^*_m and $S^*_{m/2}$ for the estimator of the maximum of a uniform distribution with $n = 200$. The left results are for a single sampling per m, and the right results for averaging over fivefold sampling.

The methods by Götze and Račkauskas (2001) and Bickel and Sakov (2008) are both based on minimizing the distance between the distributions of S^*_m and $S^*_{m'}$ for different choices m and m' of the subsampling size. As a distance measure, both articles utilized the Kolmogorov distance, i.e., the maximum distance between the cumulative distribution functions. The methods differ in the values that are tried out for m and m': Götze and Račkauskas (2001) suggested searching the minimum among all values for m and $m/2$, whereas Bickel and Sakov (2008) searched the minimum distance between m_j and m_{j+1} for $m_j = \lceil q^j n \rceil$, with the recommendation to set $q = 0.75$. These methods are provided by **moonboot** as `estimate.m(...,method="goetze")` and `estimate.m(...,method="bickel")`, respectively.

Trying out all combinations of m and $m/2$, as suggested by Götze and Račkauskas (2001), has a runtime of order $O(Rn^2)$, and it would thus be preferable to use a more efficient minimum search like, e.g., a Golden Section Search (Press et al., 1992). Such algorithms, however, are devised for deterministic functions, not for random functions. As can be seen in Fig. 2, the Kolmogorov distance $d(m, m/2)$ between two randomly sampled distributions scatters considerably and this is only somewhat remedied by repeated sampling. This means that even the usual initial bracketing process by going downhill with increasing step size is unreliable when sampling from specific data, even if, on average, the function actually has a non-boundary minimum[4]. We therefore stuck to the original suggestion by Götze and Račkauskas (2001) and have implemented an exhaustive search over all even values for m.

Two other methods have been suggested in the literature, which we have not included in **moonboot**. Chung and Lee (2001) based their method on the assumptions of root-n consistency and asymptotic normality of the estimator. Although the assumption of root-n consistency might be overcome by replacing factors $n^{1/2}$ with τ_n in the formulas, the normality assumption is essential for that method because it is based on an Edgeworth expansion. In our tests with non-normal estimators, the coverage probabilities were very low and did not even converge to the nominal value for large n. Lee and Yang (2020) suggested using a double bootstrap for estimating the coverage probability for different values of m and to choose the m where this estimate is close to the nominal value. In our experiments, however, the estimated coverage probability based on an in-sample bootstrap was either way too low or almost one and thus rarely close to the nominal value. For *xicor*, this behavior was not even monotone in m, which made the method difficult to apply. Moreover, it has a runtime complexity of $O(R^2n^2)$, which made Monte Carlo simulations for its evaluation infeasible.

[4]Fig. 2 in the article by Götze and Račkauskas (2001) shows that even this is not guaranteed, and the method can thus fail.

2.3 Functions provided by *moonboot*

Like in the **boot** package, the computation of the confidence interval is split into two steps: the sampling procedure for simulating the bootstrap distribution T_m^*, and the computation of the confidence interval therefrom. Using the same interface makes it easy for a user to switch between **boot** and **moonboot**. Moreover, this interface is very flexible because it allows for statistics defined on complicated data structures for which only the user knows how to do the sub-indexing. The interface consists of the following two functions:

`mboot(data, statistic, m, R=1000, replace=FALSE, ...)`
: Simulates the bootstrap distribution of the given estimator *statistic*, which must have been defined as a function with two arguments: `statistic(data,indices)`. For multidimensional data, each row in *data* is assumed to be one data point. Additional arguments are passed to *statistic*.

`mboot.ci(boot, conf=0.95, tau=NULL, type=c("all", "basic", "norm"))`
: Estimates the confidence interval with Eqs. (4) ("basic") or (5) ("norm"). *tau* must be a function that computes τ_n from its argument n. If it is not provided, it is estimated with `estimate.tau()` with the default settings of this function. According to the results in Section 2.4.1, this is not recommended, though, and it is preferable to provide the correct value for *tau*.

Calling these functions requires knowledge of the scaling factor τ_n and a choice for the subsample size m. For the cases that τ_n is not known or that m shall be chosen in a data-based way, two other functions are provided:

```
estimate.tau(data, statistic, R=1000, replace=FALSE, min.m=3,
             gamma=seq(0.2, 0.7, length.out=5), method="variance", ...)
```
: Estimates the scaling factor with the method by Bertail et al. (1999). The values for m are tried out as $m_i = n^{\gamma_i}$. *min.m* is the minimum sample size for which the statistic makes sense and can be computed. The estimation is based on the scaling behavior of the variance (`method="variance"`) or of the quantile ranges (`method="quantile"`).

```
estimate.m(data, statistic, tau, R=1000, replace=FALSE, min.m=3,
           method="bickel", params=NULL, ...)
```
: Estimates m with the method by Bickel and Sakov (2008) (`method="bickel"`), the method by Götze and Račkauskas (2001) (`method="goetze"`), or with the volatility index (`method="politis"`). *params* is a list that can be used to pass additional parameters for the underlying method, e.g. `params=list(q=0.75)` for the method by Bickel and Sakov (2008). *min.m* is the minimum sample size for which the statistic makes sense and can be computed.

2.4 Typical usage

Let us assume that you want to estimate a confidence interval for an estimator `my.stat` on the basis of some data `x`. Then you must first define a wrapper function that computes `my.stat` only for the data selected according to the provided indices:

```
boot.stat <- function(dat, indices) {
  my.stat(dat[indices])
}
```

Without any knowledge about the asymptotic properties of `my.stat`, you can estimate a 95% confidence interval with:

estimator $\hat{\theta}$	*data model*	*true* θ	τ_n
`max(x)`	$x_i \sim \text{unif}(0,1)$	1	n
`moonboot::shorth(x)`	$x_i \sim \text{norm}(0,1)$	0	$n^{1/3}$
`XICOR::xicor(x,y)`	$x_i \sim \text{unif}(-1,1), y_i \sim x_i + \mathcal{N}(0,0.5)$	0.3818147	$\sqrt{n}$
`mean(x)`	$x_i \sim \text{power}(2,0,1)$	3/4	$\sqrt{n}$

Table 1: Tested estimators and data generation processes. $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ are the simulated data sets.

```
boot.out <- mboot(x, boot.stat, m=sqrt(NROW(x)))
ci <- mboot.ci(boot.out, type="basic")
print(ci)
```

Beware that this automatically estimates the asymptotic convergence rate $1/\tau_n^2$ of the variance of `my.stat` with the method by Bertail et al. (1999). It is thus better to provide this rate in the parameter `tau`, if you happen to know it. For a root-n consistent estimator, e.g., it is $\tau_n = \sqrt{n}$ and you can call `mboot.ci()` as follows[5]:

```
ci <- mboot.ci(boot.out, tau=function(n) { n^0.5 }, type="basic")
```

3 Examples

In order to evaluate the implemented methods, we applied them to four different estimators, three of which were bootstrap inconsistent. A summary can be found in Tbl. 1, and the detailed description follows. $x_1, \ldots, x_n$ denote the i.i.d. sample values drawn from the given distributions.

Maximum of a uniform distribution (max). The maximum likelihood estimator for the upper bound of a uniform distribution between zero and θ is $\hat{\theta} = \max\{x_1, \ldots, x_n\}$. This estimator was already given by Bickel et al. (1997) as an example of a bootstrap inconsistent estimator. It has the nice property that its probability density $g(t)$ can be readily computed as

$$g(t) = \begin{cases} n t^{n-1} \theta^{-n} & \text{for } 0 \le t \le \theta \\ 0 & \text{else} \end{cases} \tag{8}$$

This estimator is neither root-n consistent, nor asymptotically normal. The scaling factor τ_n can be computed from Eq. (8) as

$$\text{Var}(\hat{\theta}) = \theta^2 \frac{n}{(n+2)(n+1)^2} \overset{n\to\infty}{\sim} \frac{\theta^2}{n^2} \quad \Rightarrow \quad \tau_n = n \tag{9}$$

Tukey's Shorth (shorth). This is the mean of the data points in the shortest interval that contains half of the data. For symmetric distributions with a strongly unimodal density, Andrews et al. (1972, p. 50ff) has shown that this estimator is cube root consistent, i.e., $\tau_n = n^{1/3}$. Bootstrap inconsistency of the shorth was mentioned by Léger and MacGibbon (2006). We have implemented this estimator in **moonboot** as `shorth()` and applied it to normally distributed data.

Chatterjee's rank correlation (xicor). This coefficient ξ_n was introduced by Chatterjee (2021) as an estimator for an index $\xi(X, Y)$ that is zero, if the two random variables X and Y are independent, and one, if Y is a measurable function of X. For continuous X and Y, this is an interesting example of a root-n consistent estimator (Lin and Han, 2022) that is bootstrap inconsistent (Lin and Han, 2024; Dalitz et al., 2024). We have used the implementation

[5] It is more efficient to just set `tau=sqrt`, of course, but the example uses a more complicated way for the sake of clarity.

method	*estimator*	*true* β	(γ_1, γ_5)	n	*estimated* β	n	*estimated* β
variance	mean	0.5	$(\mathbf{0.2}, \mathbf{0.5})$	100	0.4723	500	0.4870
			$(0.4, 0.8)$		0.5964		0.5480
	max	1.0	$(0.2, 0.5)$		0.6789		0.7730
			$(\mathbf{0.4}, \mathbf{0.8})$		0.9595		1.0043
	xicor	0.5	$(0.2, 0.5)$		0.1821		0.2945
			$(\mathbf{0.4}, \mathbf{0.8})$		0.4914		0.5052
quantile range	mean	0.5	$(\mathbf{0.2}, \mathbf{0.5})$	100	0.5394	500	0.5164
			$(0.4, 0.8)$		0.6167		0.5600
	max	1.0	$(0.2, 0.5)$		0.8355		0.8722
			$(\mathbf{0.4}, \mathbf{0.8})$		1.1068		1.0872
	xicor	0.5	$(0.2, 0.5)$		0.0332		0.2128
			$(\mathbf{0.4}, \mathbf{0.8})$		0.4880		0.5171

Table 2: Dependency of the estimation of $\tau_n = n^\beta$ on the chosen endpoints for the sequence $m_i = n^{\gamma_i}$ and two different values for n. The better choice is marked in bold for each estimator. The estimated values for the power β in τ_n have been averaged over 100 estimations.

`xicor()` from the package **XICOR** (Chatterjee and Holmes, 2023) to compute ξ_n, and the function given by Dalitz et al. (2024) to compute $\xi(X, Y)$ for the model $X \sim Y + \varepsilon$, where ε is normally distributed with zero mean and $\sigma = 0.5$. Note that we have used the original definition of ξ_n by Chatterjee (2021), not the bias reduced form suggested by Dalitz et al. (2024).

Mean of an unsymmetric distribution (mean). The mean is a very well-behaved statistic: it is asymptotically normal, root-n consistent and bootstrap consistent. It can thus serve as a simple test case of how well the m-out-of-n bootstrap performs for bootstrap consistent estimators. We have simulated data according to the density $f(x) = 3x^2$ for $x \in [0, 1]$, which was already used in a study by Dalitz (2017), and for which we have implemented the random number generator `rpower()` in **moonboot**.

4 Results

For all four estimators, we have evaluated the methods for estimating τ_n and the different choices for m in the m-out-of-n bootstrap without replacement by means of Monte Carlo simulations. We have computed confidence intervals with a nominal coverage probability P_{cov} of 0.95 and estimated the actual coverage probability by repeating the computation $N = 10^4$ times. This means that the accuracy (i.e. the width of a 95% confidence interval) of the estimated P_{cov} is about ± 0.005. Using a larger N would have made the simulation for the method by Goetze intractable. All simulations were done with $R = 1000$ bootstrap repetitions.

4.1 Estimation of τ_n

Our simulations confirmed that the choice for the extreme values of the trial values $m_i = n^{\gamma_i}$ for $i = 1, \ldots, 5$ indeed influence the accuracy of the estimation of τ_n. For the *mean* estimator, e.g., we achieved the best results for $\gamma_1 = 0.2$ and $\gamma_5 = 0.5$, whereas this choice was poor for *xicor* and *max* (see Tbl. 2). The results for estimation by means of the variance or quantile ranges are similar. We thus conclude that it does not matter which method is used. The results for *xicor*, however, are a warning that the estimation of τ_n with the method by Bertail et al. (1999) can be grossly inaccurate, unless the range of the γ_i has been luckily guessed.

4.2 Choice of m

To see how the coverage probability depends on the choice of m, we have first used the fixed formula $m = n^\beta$ with different choices for β. The results in Fig. 3 show that the coverage probability indeed depends on the choice of m. Moreover, the optimal choice for β depends on the estimator: $\beta = 1/3$ was, e.g., a decent choice for *shorth*, but a poor choice for *xicor*.

The results for the data based methods for choosing m are summarized in Fig. 4. The method by Politis et al. (1999) cannot be recommended, because the resulting coverage probability never approached the nominal value in our simulations. For the *shorth*, it was only about 0.7 even for high values of n. For the other estimators, it was somewhat greater, but nevertheless too small. This behavior can be understood from the curves showing $m(n)$ in Fig. 4: The method by Politis et al. (1999) always chooses an m proportional to n, which means that the condition $\lim_{n\to\infty} m/n = 0$ is violated.

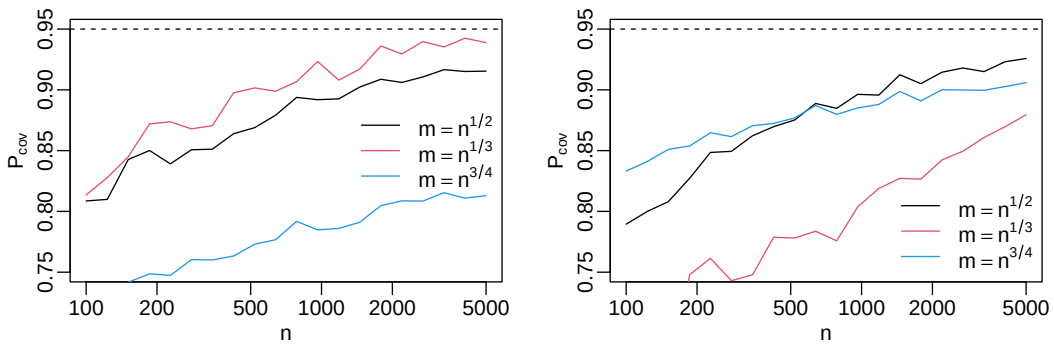
The methods according to Götze and Račkauskas (2001) and Bickel and Sakov (2008), on the other hand, both showed for all estimators a similar coverage probability which approached the nominal value with increasing n. The method by Bickel and Sakov (2008) was slightly better for three of the four tested estimators. It should be noted, though, that there was considerable fluctuation in the estimated values for m, as can be concluded from the wide error bars in Fig. 4. As the method by Götze and Račkauskas (2001) has a much higher runtime of order $O(Rn^2)$, the method by Bickel and Sakov (2008), which has a runtime of order $O(Rn)$, is preferable from a runtime perspective, too.

Out of curiosity, we have also compared the m-out-of-n bootstrap without replacement with the ordinary n-out-of-n bootstrap for the estimator *mean*, which is bootstrap consistent and thus allows for a comparison. For bootstrap consistent estimators with additional smoothness properties[6], it was generally proven by Bickel et al. (1997) that estimates based on the m-out-of-n bootstrap with or without replacement are less efficient than those based on the n-out-of-n bootstrap. The mean is an example of such an estimator. As can be seen in Fig. 5, the m-out-of-n bootstrap is competitive with the basic bootstrap only for large $n \gtrsim 2000$ and a luckily chosen m. For a data based choice of m with the method by Bickel and Sakov (2008), the coverage probability is smaller, although the difference in the interval length is small. This again demonstrates that, for bootstrap consistent estimators, the usual n-out-of-n bootstrap is preferable, especially because the best choice for m is not known, in general.

5 Violation of the assumptions

The assumptions (1) & (2), under which the m-out-of-n bootstrap works, are quite weak, but there are nevertheless some estimators that violate these conditions. For example,

[6]Most notably that it is twice Fréchet differentiable.



a. shorth b. xicor

Figure 3: Coverage probability P_{cov} of the m-out-of-n bootstrap without replacement for the choices $m = n^\beta$ with $\beta \in \{\frac{1}{2}, \frac{1}{3}, \frac{3}{4}\}$. Note that the n-axis uses a logarithmic scale.

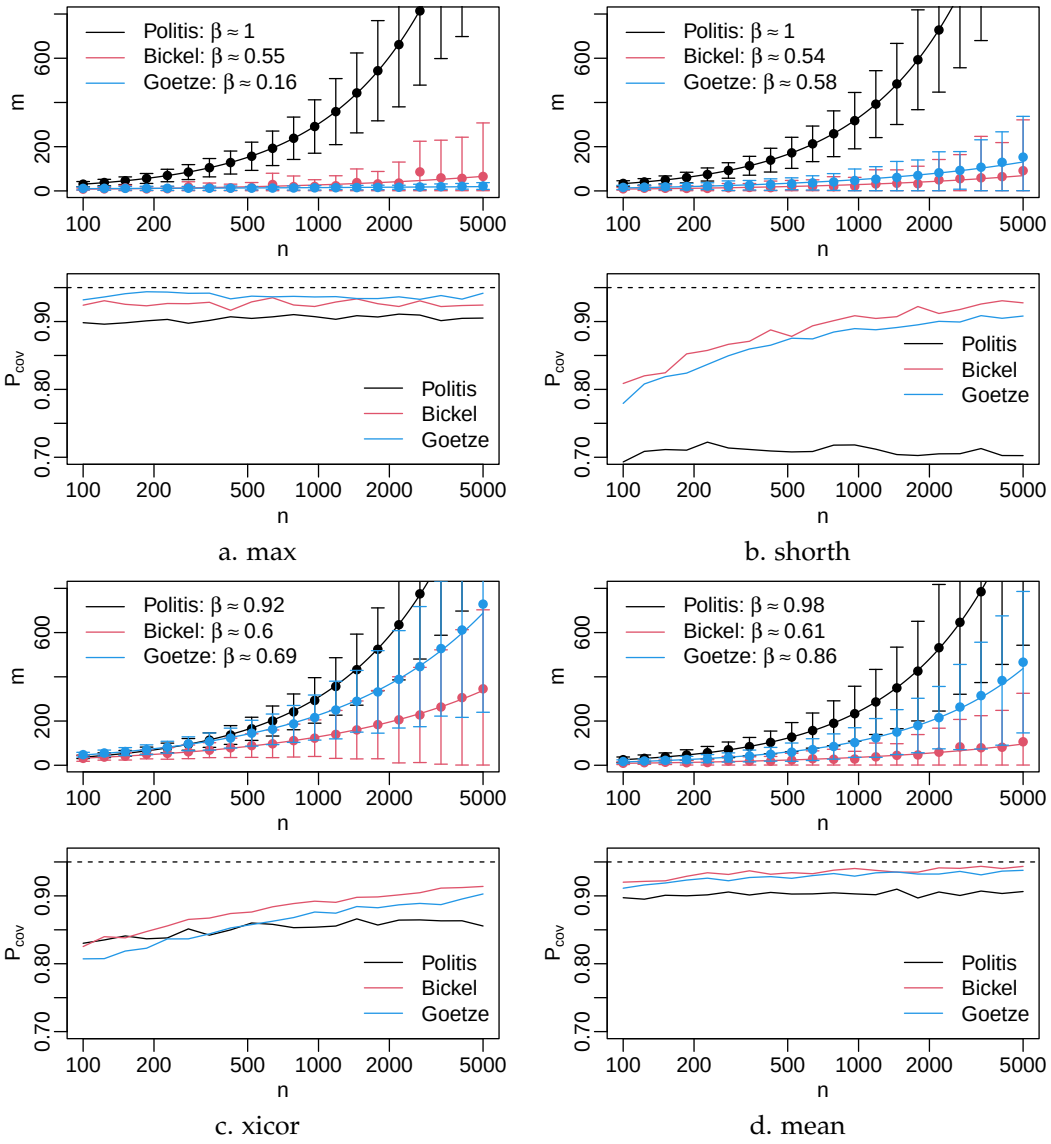


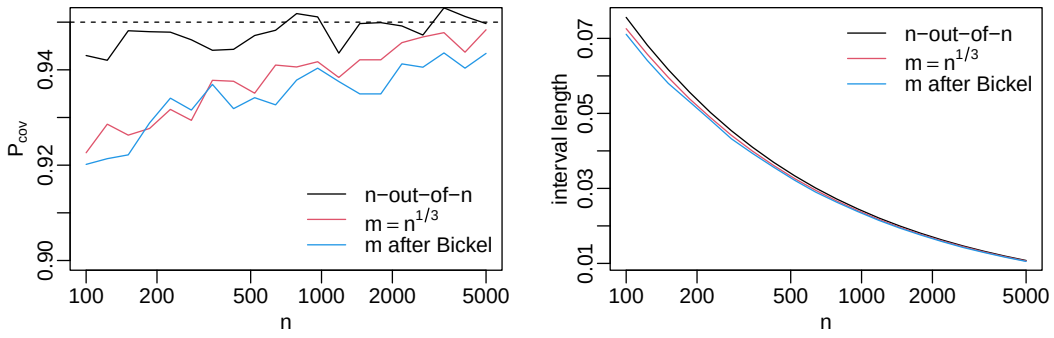
Figure 4: Comparison of the three data based methods for choosing m in the m-out-of-n bootstrap without replacement. The upper plots show the mean value $m(n)$ together with error bars $\pm \mathrm{sd}(m)$ and a least squares fit $\log(m) = \log(cn^{\beta})$. The bottom plots show the resulting coverage probabilities. Note that the n-axis uses a logarithmic scale.

inconsistent estimators, i.e., estimators that do not converge in probability to the true parameter value, violate the assumptions because consistency is a necessary condition for the assumptions to hold, as shown in section 2.2. Using **moonboot** in such a case is an error on the side of the user, and if the user has determined the scaling factor τ_n by an analysis of the estimator, he usually will be aware of this. It might be, however, that a user relies on the fully automated estimation of τ_n provided by **moonboot** and thus is not aware of a violation of the assumptions.

We therefore have tested our package with two unbiased, yet inconsistent estimators, too. Please note that these estimators are bizarre examples and are not meant to be used in practice. The first estimator is only the very first observation as an estimator for the mean μ of the unsymmetric distribution `moonboot::dpower(..,2,0,1)`:

$$\hat{\mu}_1 = X_1 \tag{10}$$

The distribution of this estimator is the same for all sample sizes, and the scaling factor is thus $\tau_n \propto 1$, which fulfills condition (2), but violates condition (1). The second estimator



a. coverage probability b. average interval length

Figure 5: Comparison of the m-out-of-n bootstrap without replacement with the ordinary n-out-of-n (basic) bootstrap for the bootstrap consistent estimator *mean*. Note that the n-axis uses a logarithmic scale.

estimates the parameter λ of a Poisson distribution. For a Poisson distribution, both mean and variance are λ, which allows for the construction of an unbiased estimator for λ from the unbiased estimators $\overline{X}$ for the mean and S^2 for the variance as follows:

$$\hat{\lambda}_n = n\overline{X} - (n-1)S^2 = \sum_{i=1}^{n} X_i - \sum_{i=1}^{n} \left(X_i - \overline{X}\right)^2 \tag{11}$$

As both $\overline{X}$ and S^2 are root-n consistent estimators, the scaling factor for condition (2) to hold is $\tau_n \propto n^{-1/2}$, which again is in violation of condition (1).

As the assumptions (1) & (2) are only sufficient, but not necessary conditions for the m-out-of-n bootstrap, we first estimated the coverage probability of the "basic" m-out-of-n bootstrap interval when the correct scaling factor τ_n is provided. For $\hat{\mu}_1$, the coverage probability obviously neither depends on n nor on the subsampling size m, and it turned out to be about 0.52 for a nominal 0.95 interval. For $\hat{\lambda}_n$, the coverage probability decreased with increasing n and was only about 0.17 for $n = 5000$ and $m = \sqrt{n}$. These results show that the m-out-of-n bootstrap indeed does not work for these inconsistent estimators.

We then estimated the scaling factors with `estimate.tau()` for $n = 100$ and $n = 500$, and, for all parameter settings, the mean estimates for β in $\tau_n = n^\beta$ were less than 0.002 for $\hat{\mu}_1$ and less than -0.40 for $\hat{\lambda}_n$. This means that, for these estimators, the violation of the assumptions can be automatically detected due to suspiciously small estimated values for β. In order to make the user aware of this issue, `mboot.ci()` gives a warning if τ_n decreases or if its increase is suspiciously small, i.e., if it increases at a slower rate than $n^{0.01}$.

6 Conclusions

The **moonboot** package provides ready-to-run implementations of the m-out-of-n bootstrap and methods for estimating its parameters. Our simulations have shown that the quality of the data based method for estimating the scaling factor τ_n is highly sensitive to a parameter for which no universal recommendations can be made. It is thus recommended to estimate τ_n by a different method, e.g., by model based Monte Carlo simulations of the variance of the estimator and estimating it by means of Eq. (3). For a data based choice of m, the method by Bickel and Sakov (2008) as implemented in `estimate.m(...,method="bickel")` can be recommended, according to our simulations.

Acknowledgements

We thank the anonymous reviewers for their valuable comments and for the suggestion to investigate the case when the assumptions are violated.

References

D. F. Andrews, P. J. Bickel, F. R. Hampel, P. J. Huber, W. H. Rogers, and J. W. Tukey. *Robust Estimators of Location*. Princeton University Press, Princeton, USA, 1972. [p131]

M. A. Arcones. On the asymptotic accuracy of the bootstrap under arbitrary resampling size. *Annals of the Institute of Statistical Mathematics*, 55:563–583, 2003. [p126]

P. Bertail, D. N. Politis, and J. P. Romano. On subsampling estimators with unknown rate of convergence. *Journal of the American Statistical Association*, 94(446):569–579, 1999. [p126, 127, 130, 131, 132]

P. J. Bickel and A. Sakov. On the choice of m in the m out of n bootstrap and confidence bounds for extrema. *Statistica Sinica*, 18(3):967–985, 2008. [p126, 128, 129, 130, 133, 135]

P. J. Bickel, F. Götze, and W. R. van Zwet. Resampling fewer than n observations: gains, losses, and remedies for losses. *Statistica Sinica*, 7(1):1–31, 1997. [p125, 131, 133]

A. Canty and B. D. Ripley. *boot: Bootstrap R (S-Plus) Functions*, 2021. R package version 1.3-28. [p125]

S. Chatterjee. A new coefficient of correlation. *Journal of the American Statistical Association*, 116(536):2009–2022, 2021. doi: 10.1080/01621459.2020.1758115. [p131, 132]

S. Chatterjee and S. Holmes. *XICOR: Robust and generalized correlation coefficients*, 2023. R package. https://CRAN.R-project.org/package=XICOR. [p132]

K.-H. Chung and S. M. Lee. Optimal bootstrap sample size in construction of percentile confidence bounds. *Scandinavian Journal of Statistics*, 28(1):225–239, 2001. [p126, 128, 129]

C. Dalitz. Construction of confidence intervals. Technical Report 2017-01, Hochschule Niederrhein, Fachbereich Elektrotechnik und Informatik, 2017. [p132]

C. Dalitz, J. Arning, and S. Goebbels. A simple bias reduction for Chatterjee's correlation. *Journal of Statistical Theory and Practice*, 18:51, 2024. doi: 10.1007/s42519-024-00399-y. [p125, 127, 131, 132]

A. C. Davison and D. V. Hinkley. *Bootstrap Methods and their Application*. Cambridge University Press, 1997. [p125]

T. J. DiCiccio and B. Efron. Bootstrap confidence intervals. *Statistical Science*, 11(3):189–228, 1996. [p125]

B. Efron. Bootstrap methods: another look at the jackknife. *Annals of Statistics*, 7(1):1–26, 1979. [p125]

E. Giné and J. Zinn. Bootstrapping general empirical measures. *The Annals of Probability*, pages 851–869, 1990. [p125]

F. Götze and A. Račkauskas. Adaptive choice of bootstrap sample sizes. *Lecture Notes-Monograph Series*, 36 (State of the Art in Probability and Statistics):286–309, 2001. [p126, 128, 129, 130, 133]

P. Hall. Theoretical comparison of bootstrap confidence intervals. *The Annals of Statistics*, 16 (3):927–953, 1988. [p125]

A. Kleiner, A. Talwalkar, P. Sarkar, and M. I. Jordan. A scalable bootstrap for massive data. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 76(4):795–816, 2014. [p126]

S. M. Lee and P. Yang. Bootstrap confidence regions based on M-estimators under nonstandard conditions. *The Annals of Statistics*, 48(1):274–299, 2020. [p126, 128, 129]

C. Léger and B. MacGibbon. On the bootstrap in cube root asymptotics. *Canadian Journal of Statistics*, 34(1):29–44, 2006. [p125, 131]

Z. Lin and F. Han. Limit theorems of Chatterjee's rank correlation, 2022. Preprint. https://arxiv.org/abs/2204.08031. [p131]

Z. Lin and F. Han. On the failure of the bootstrap for Chatterjee's rank correlation. *Biometrika*, page asae004, 2024. doi: 10.1093/biomet/asae004. [p125, 126, 131]

D. N. Politis and J. P. Romano. Large sample confidence regions based on subsamples under minimal assumptions. *The Annals of Statistics*, 22(4):2031–2050, 1994. doi: 10.1214/aos/1176325770. [p125, 126, 128]

D. N. Politis, J. P. Romano, and M. Wolf. *Subsampling*. Springer, New York, 1999. [p126, 128, 133]

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, USA, 2nd edition, 1992. [p129]

B. Sen, M. Banerjee, and M. Woodroofe. Inconsistency of bootstrap: The Grenander estimator. *The Annals of Statistics*, 38(4):1953–1977, 2010. [p125]

J. Shao and D. Tu. *The jackknife and bootstrap*. Springer, 1995. [p125]

M. Sherman and E. Carlstein. Confidence intervals based on estimators with unknown rates of convergence. *Computational Statistics & Data Analysis*, 46(1):123–139, 2004. [p127]

Christoph Dalitz
Niederrhein University of Applied Sciences
Institute for Pattern Recognition
Reinarzstr. 49
Germany
(ORCiD 0000-0002-7004-5584)
christoph.dalitz@hs-niederrhein.de

Felix Lögler
Niederrhein University of Applied Sciences
Institute for Pattern Recognition
Reinarzstr. 49
Germany

Rgof and R2sample: Testing and Benchmarking for the Univariate Goodness-of-Fit and Two-Sample Problems

by Wolfgang Rolke

Abstract In a goodness-of-fit problem one attempts to see whether a data set might have been generated by some theoretical probability distribution, possibly with unknown parameters. In a (non-parametric) two-sample problem one wants to check whether two data sets could have come from the same unspecified distribution. Both are among the oldest and best studied problems in Statistics. While they seem quite different, there are a number of similarities between these problems, not the least of which is that many methods exist that have versions for both. The two packages discussed in this article bring together a large number of methods and many different scenarios, most of which do not yet have existing implementations. They also include routines that allow a developer of a new method to quickly compare its performance (aka power) to those included in the package for a large number of cases.

1 Introduction

Both the goodness-of-fit and the nonparametric two-sample problem have histories going back a century, with many contributions by some of the most eminent statisticians. In the goodness-of-fit problem we have a sample $(x_1, .., x_n)$ drawn from a random variable X. We also have a probability distribution F, possibly with unknown parameters, and we wish to test $H_0 : X \sim F$. In the two-sample problem we also have a second sample $(y_1, .., y_m)$ drawn from some distribution G, and here we want to test $H_0 : F = G$, that is we want to test whether the two data sets were generated by the same (unspecified) distribution.

The literature on both of these problems is vast and steadily growing. Detailed discussions can be found in D'Agostini and Stephens (1986), Thas (2010), Raynor et al. (2012). For an introduction to Statistics and hypothesis testing in general see Casella and Berger (2002) or Bickel and Doksum (2015).

Some tests such as the Kolmogorov-Smirnov test are already implemented for both problems in base **R** R Core Team (2021). Many others can be run through various existing packages, for example the Anderson-Darling goodness-of-fit test is available in the **R** package *ADGofTest* Bellosta (2011). There are a number of packages with tests that focus on a specific distribution, for example the *nortest* Gross and Ligges (2015) package has five tests for composite normality. There are also packages that allow the user to run several tests, for example the *twosamples* Dowd (2022), *dgof* Taylor and Emerson (2009), *EnvStats* Millard and Kowarik (2017) and *goftest* Faraway et al. (2007) packages.

However, there are no packages that bring together as many tests as *R2sample* and *Rgof*. Moreover, some methods do not currently have any *R* implementations, for example Zhang's test, Lehmann-Rosenblatt, the Wasserstein p1 test and almost all tests for discrete data.

Both packages have the following features:

- many methods are implemented for both continuous and discrete data.
- the methods are implemented using both *Rcpp* Eddelbuettel et al. (2024) and parallel programming.
- the packages include routines to run several test and then find a corrected p value for the combination of tests.

- some of the methods allow for data with weights.
- the routines allow for a random sample size, assumed to come from a Poisson distribution.
- in the two-sample problem some methods make use of large-sample formulas, therefore allowing for very large data sets.
- the routines can also use any other user-defined tests.
- the packages include routines to easily carry out power studies and draw power graphs.
- the packages include routines to easily compare the power of a new test to those included in the packages.
- in the two-sample package the user can provide a routine that generates new data from a model. This can be used as an alternative to the permutation method to find p values.

There are several reasons for including tests for discrete data. In the context of a computer program this means a finite (and usually fairly small) number of different values which then repeat many times.

- Tests for discrete data such as from Binomial or Poisson distributions are of interest in their own right.
- There are currently almost no implementations of either goodness-of-fit or two-sample methods for discrete data in *R*.
- It also makes it possible to apply the tests to very large continuous data sets via discretization. While a goodness-of-fit test for a continuous data set with (say) 100,000 observations can be done in a matter of a few minutes, for larger data sets the calculations will be quite time consuming. Data sets with many millions of observations are not uncommon today. Binning the data and then running the corresponding discrete tests however is quite fast.
- There are also situations where the underlying distribution is continuous but the data is collected in binned form. This is for example often the case for data from high energy physics experiments and from astronomy because of finite detector resolution. In some fields this is referred to as histogram data. For the purpose of either the goodness-of-fit or two-sample problems standard discrete data and histogram data can be treated the same, with the midpoints of the bins used as observations where such are needed.

For the two-sample problem p values are found via the permutation method. If the data sets are large for some of the tests the p values can be found via large sample approximations. In the goodness-of-fit case p values are always found via simulation. While large sample approximations are know for some methods such as Kolmogorov-Smirnov and Anderson-Darling, there are no known large sample theories for most of the other tests. Moreover, in the more common situation where the distribution under the null hypothesis depends on parameters, which have to be estimated from the data, even those tests no longer have known large sample theories and one is forced to use simulation to find p values.

The packages *Rgof* Rolke (2023b) and *R2sample* Rolke (2023a) are available from CRAN.

2 Goodness-of-fit / two-sample hybrid problem

As was mentioned in the abstract, while the goodness-of-fit problem and the two-sample problem are quite different, they also share certain features such as methods that exist for both. On a deeper level they are both hypothesis tests in the Fisherian sense, in that they are

tests without an alternative hypothesis. These tests are usually done for confirmation, that is in the goodness-of-fit case the researcher wants to make sure that his assumed probability model is reasonably good, without any consideration of how it might be wrong.

There is yet another connection between these types of problems. Sometimes one wants to carry out a goodness-of-fit test. However, the model under the null hypothesis is quite complex with a large number of nuisance parameters Therefore calculating values from the distribution function requires integration in high dimensions and is at present not feasible. It is however possible to sample from the distribution. So the problem now changes from a goodness-of-fit to a two-sample problem.

If the null hypothesis in the goodness-of-fit problem does not fully specify the distribution but just its functional form one can then estimated the parameters from the data. However, in this situation the permutation method for estimating the p value fails, it is extremely conservative. Instead the user can provide a routine to generate new data, essentially using a parametric bootstrap approach.

3 The types of problems

The problems that can be analyzed with these packages are as follows:

- **Goodness-of-Fit Problem - Continuous Data**: We have a sample x of size of n drawn from some random variable X. F is a continuous probability distribution, which may depend on unknown parameters. We want to test $X \sim F$.
- **Goodness-of-Fit Problem - Discrete Data**: We have a set of values *vals* and a vector of counts x. F is a discrete probability distribution, which may depend on unknown parameters. We want to test $X \sim F$.
- **Two-sample Problem - Continuous Data**: We have a sample x of size of n, drawn from some unknown continuous probability distribution F, and a sample y of size m, drawn from some unknown continuous probability distribution G. We want to test $F = G$.
- **Two-sample Problem - Discrete Data**: We have a set of values *vals* and vectors of counts x and y, drawn from some unknown discrete probability distributions F and G. We want to test $F = G$.

In all of the above problems, the sample size can either be fixed or follow a Poisson distribution with a known rate. In all cases the data can be weighted. In all cases the user can provide his/her own testing method.

4 The methods

In the following we list the methods included in the packages. Most are well known and have been in use for a long time. For their details see the references. They are:

Method	*Rgof*		*R2sample*	
	Continuous	Discrete	Continuous	Discrete
Chi-Square Tests	Yes	Yes	Yes	Yes
Kolomorov-Smirnov	Yes	Yes	Yes	Yes
Kuiper	Yes	Yes	Yes	Yes
Cramer-von Mises	Yes	Yes	Yes	Yes
Anderson-Darling	Yes	Yes	Yes	Yes
Zhang's tests	Yes	No	Yes	No
Wasserstein	Yes	Yes	Yes	Yes
Watson's test	Yes	Yes	No	No
Lehmann-Rosenblatt	No	No	Yes	Yes

There are of course many other tests that could have also been implemented in the routines. All the tests included share the following features. They are true omnibus tests, that is not designed with any specific alternative in mind. For this reason we did not include the class of Neyman's smooth tests, for example. Moreover they are all tests that do not depend on some tuning parameters. The exception here are the chi-square tests, which depend on the choice of the number and shape of the bins. The chi-square tests are included because they are so well known and widely used, even so their power often leaves much to be desired.

We denote the cumulative distribution function (cdf) by F, its empirical distribution function (edf) by $\widehat{F}$. In the case of the two-sample problem we also have the edf of the second data set $\widehat{G}$ and the edf of the combined data set $\widehat{H}$.

1. **Chi-Square Tests**

In the case of continuous data the routines include eight chi-square tests, with either equal size (ES) or equal probability (EP) bins, either a large (*nbins[1]*=50) or a small (*nbins[2]*=10) number of bins and with either the Pearson (P) or the log-likelihood (L) formula. Here and in what follows *nbins* and similar items are arguments to the routines that the user can change. So the combination of a large number of equal size bins and Pearson's chi-square formula is denoted by *ES-l-P*, etc.

In the case of discrete data the type and the number of classes is already given, and for a second test these are combined for a total of *nbins[2]*=10. Again both chi-square formulas are used. So here the case of a large number of bins and Pearson's formula is denoted by *l-P*.

In all cases neighboring bins with low counts are joined until all bins have a count of at least *minexpcount*=5. In all cases the p values are found using the usual chi-square approximation.

If parameters have to be estimated, this is done via the user-provided routine *phat*. As long as the method of estimation used is consistent and efficient and the expected counts are large enough the chi-square statistic will have a chi-square distribution, as shown by Fisher (1922) and Fisher (1924).

Alternatively we can use the argument *ChiUsePhat=FALSE*. In that case the value provided by *phat* is used as a starting point but the parameters are estimated via the method of minimum chi-square. This method has the desirable feature that if the null hypothesis is rejected for this set of values, it will always be rejected for any other as well. For a discussion of this estimation method see Berkson (1980).

2. **Kolmogorov-Smirnov (KS)**

This test is based on the largest absolute distance between F and $\widehat{F}$ in the goodness-of-fit problem and between $\widehat{F}$ and $\widehat{G}$ in the two-sample problem. The tests were first proposed in Kolmogorov (1933), Smirnov (1939) and are among the most widely used tests today. There is a known large sample distribution of the test statistic in the two-sample problem, which

is used either if both sample sizes exceed 10000 or if the argument *UseLargeSample=TRUE* is set. In the goodness-of-fit case the large sample theory is known only in the case of a fully specified distribution under the null hypothesis. Because this is rarely of interest the large sample approximation is not used.

3. **Kuiper (K)**

This test is closely related to Kolmogorov-Smirnov, but it uses the sum of the largest positive and negative differences as a test statistic. It was first proposed in Kuiper (1960).

4. **Cramer-vonMises (CvM)**

This test is based on the integrated squared differences:

- Goodness-of-Fit: $\int_{-\infty}^{\infty} \left(F(x) - \widehat{F}(x)\right)^2 dF(x)$
- Two-Sample: $\int_{-\infty}^{\infty} \left(\widehat{F}(x) - \widehat{G}(x)\right)^2 d\widehat{H}(x)$

The goodness-of-fit version is discussed in Cramer (1928) and von Mises (1928). The two-sample version was proposed in Anderson (1962).

5. **Anderson-Darling (AD)**

This test is similar to the Cramer-vonMises test but with an integrand that emphasizes the tails:

- Goodness-of-Fit: $\int_{-\infty}^{\infty} \frac{\left(F(x)-\widehat{F}(x)\right)^2}{F(x)(1-F(x)} dF(x)$
- Two-Sample: $\int_{-\infty}^{\infty} \frac{\left(\widehat{F}(x)-\widehat{G}(x)\right)^2}{\widehat{H}(x)(1-\widehat{H}(x)} d\widehat{H}(x)$

It was first proposed in Anderson and Darling (1952). The two-sample version is discussed in Pettitt (1976).

6. **Zhang's tests (ZA, ZK and ZC)**

These tests were proposed in Zhang (2002) and Zhang (2006). They are variations of test statistics based on the likelihood ratio and different weight functions. Note that these tests do not work for discrete data, that is, they never achieve the correct type I error rate. They are therefore not run for discrete data.

7. **Wasserstein p=1 (Wassp1)**

A test using the Wasserstein p=1 metric. It is based on a comparison of quantiles. In the goodness-of-fit case these are the quantiles of the data set and the quantiles of the cdf, and in the two-sample problem they are the quantiles of the individual data sets and the quantiles of the combined data set. If $n = m$ the test statistic in the continuous case takes a very simple form: $\frac{1}{n}\sum_{i=1}^{n} |x_i - y_i|$. In the goodness-of-fit problem for continuous data the user has to supply a function that calculates the inverse of the cdf under the null hypothesis. For a discussion of the Wasserstein distance see Vaserstein (1969).

There are also a number of tests which are only implemented for either the goodness-of-fit or the two-sample problem:

8. **Watson's Test (W), Goodness-of-Fit Problem**

This test is closely related to the Cramer-vonMises test. It adjust that tests statistic via a squared difference of the mean of $\widehat{F}(x_i)$ and 0.5. It was proposed in Watson (1961).

9. **Lehmann-Rosenblatt (LR), Two-sample Problem**

Let r_i and s_i be the ranks of x and y in the combined sample, then the test statistic is given by

$$\frac{1}{nm(n+m)}\left[n\sum_{i=1}^{n}(r_i-1)^2+m\sum_{i=1}^{m}(s_i-1)^2\right]$$

For details see Lehmann (1951) and Rosenblatt (1952).

5 Simultaneous inference

As no single test can be relied upon to consistently have good power, it is reasonable to employ several of them. We would then reject the null hypothesis if any of the tests does so, that is, if the smallest p value is less than the desired type I error probability α.

This procedure clearly suffers from the problem of simultaneous inference, and the true type I error probability will be much larger than α. It is however possible to adjust the p value so it does achieve the nominal type I error. A sketch of the algorithm is as follows:

- generate a new data set under the null hypothesis, run the desired tests and record the smallest p value.
- repeat B(=1000) times.
- use the empirical distribution function $\hat{F}_p$ of the B smallest p values to estimate their distribution function.
- apply $\hat{F}_p$ to the smallest p value of the data set. This is essentially the probability integral transform.

Here is an example: say the null hypothesis specifies a uniform $[0,1]$ distribution and a sample size of 250. Next we find the smallest p value in each run for two selections of four methods. One includes the methods by Wilson, Anderson-Darling, Zhang's ZC and a chi square test with a small number of bins and using Pearson's formula. This selection has good power against a large number of alternatives. As a second selection we use the methods by Kolmogorov-Smirnov, Kuiper, Anderson-Darling and Cramer-vonMises. For the case where the null hypothesis specifies a Uniform $[0,1]$ distribution these tests turn out to be highly correlated.

Next we find the empirical distribution function for the two sets of p values and draw their graphs. We also add the curve for the cases of four identical tests and the case of four independent tests, which of course is the Bonferroni correction. These are shown in figure 1.

As one would expect, the two curves for the p values fall between the extreme cases of total dependence and independence. Moreover, the curve of our good selection is closer to the curve of independence than the selection of correlated methods.

Finally we simply have to apply this function to the smallest p value found for the actual data.

Rgof::gof_test_adjusted_pvalues and *R2sample::twosample_test_adjusted_pvalues* find these adjusted p values. Their arguments are the same as those of *Rgof::gof_test* and *R2sample::twosample_test*, see the section Usage 2.7. For an example that uses this adjustment method in the context of simultaneous confidence bands see Aldor-Noima et al. (2013).

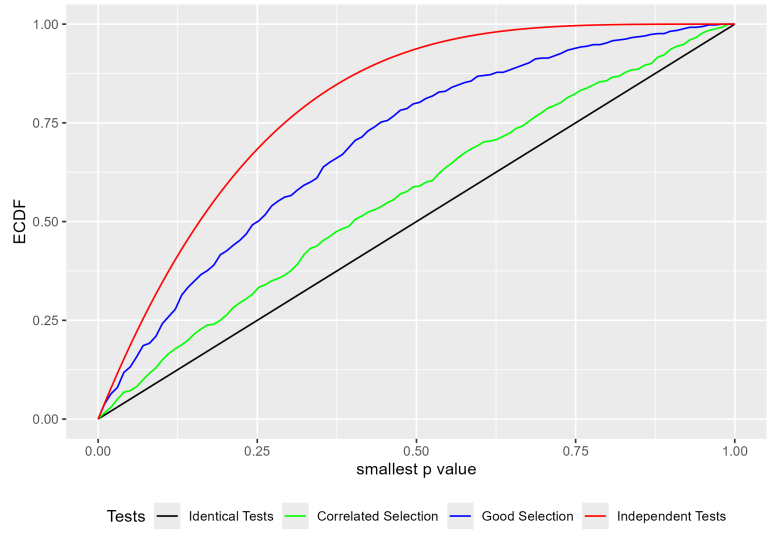


Figure 1: Distribution functions of the smallest p value for four dependence cases.

6 Special circumstances

6.1 Random sample size

In some cases the sample size is not determined at the beginning of the experiment but is a consequence of other factors. As an example, in high energy physics experiments the sample size is determined by the luminosity (aka energy) at which the accelerator is run, and by how long. In such a situation the distributions of the test statistics are different from the fixed sample size case, and there are no known null distributions. In the case of the chi-square tests, for example, the bin counts are now independent according to the theory of marked Poisson processes, and therefore the degrees of freedom need to be adjusted accordingly. Often though it is reasonable to assume that the sample size has a Poisson distribution. If so the routines in *Rgof* and *R2sample* have an argument *rate* to indicate a random sample size from a Poisson distribution with rate λ.

6.2 Weighted samples

Another variation is as follows. Say we have a continuous random variable X and a weight function w. There also exists a random variable Y such that $E[f(Y)] = E[f(X)w(x)]$ for (almost) any function f. In other words, these are weights as one encounters in importance sampling. Say we wish to test whether the distribution of Y is F but using observations from X with their weights. This is done very easily with the routines by supplying the weights as arguments. These weights can then be used to find for example the empirical distribution function, and with it run tests such as Kolmogorov-Smirnov or Anderson-Darling.

One field were this situation is common is high energy physics. There we have the Standard Model, the current best model for explaining the physics of collision experiments in particle accelerators such as the Large Hadron Collider at CERN. Say we wish to test some specific part of this theory, that is we want to do a goodness-of-fit test. However, the Standard Model depends on dozens of parameters. The calculations of the probabilities needed for a goodness-of-fit test are at present not feasible. Among other issues they would require integration in very high dimensions. However, it is possible to generate a Monte Carlo sample from the Standard Model, so instead we can run a two-sample test, comparing the data to the Monte Carlo sample. There is however another problem. The Monte Carlo sampling of the Standard Model is very computationally expensive. There exist a number of such samples, each for a specific set of the parameters of the Standard Model. Now if the test we wish to do requires a sample with a slightly different set of parameters we can use

an existing sample and importance sampling. The routines in the packages discussed here make this very easy.

7 Usage

7.1 Goodness-of-fit problem - testing

The routine to carry out hypothesis tests is *Rgof::gof_test*. It's arguments are

- x: a data set, either the continuous outcomes or the counts in the discrete case.
- vals=NA: all possible values of the discrete random variable, or NA if data is continuous.
- pnull: a function to calculate values for the cdf.
- rnull: a function to generate new data under the null hypothesis.
- w=function(x) -99: a weight function if weights are present, or -99 if not.
- phat=function(x) -99: a function to estimate parameters, or -99 if null hypothesis is simple and no parameters are estimated.
- TS: routine to calculate test statistics other than those included.
- TSextra: a list passed to TS.
- nbins=c(50, 10): number of bins to use in chi-square tests.
- rate=0: rate of Poisson if sample size is random, 0 if sample size is fixed.
- Range=c(-Inf, Inf): range of continuous random variable.
- B=5000: number of simulation runs.
- minexpcount=5: required minimal expected counts for chi-square tests.
- maxProcessors=1: number of cores to use for parallel processing, 1 means no parallel programming.
- doMethod="all": vector with names of methods, if not all are to be included.

The format of the routines pnull, rnull and w has to be as follows. In the continuous case we will use as an example the normal distribution and in the discrete case the Binomial distribution with 10 tries.

- Continuous data, no parameter estimation: a function of one variable, the data. For example, pnull=function(x) pnorm(x)
- Continuous data, with parameter estimation: a function of two variables, the data and a vector of parameter estimates. For example, pnull=function(x,p) pnorm(x,p[1],p[2])
- Discrete data, no parameter estimation: a function without arguments. For example, pnull=function() pbinom(0:10, 10, 0.5)
- Discrete data, with parameter estimation: a function of one variable, a vector of parameter estimates. For example, pnull=function(p) pbinom(0:10, 10, p)

Continuous data

As an example we generate $N = 1000$ observations from a standard normal distribution. Then we test to see whether the data comes from a normal distribution with the mean estimated from the data, so in this case the null hypothesis is true:

```
pnull = function(x, mu=0) pnorm(x, mu) # cdf under null hypothesis
rnull = function(mu=0) rnorm(1000, mu) # generate data under null hypothesis
phat = function(x) mean(x) # estimate parameter
x = rnull() # data from distribution under the null hypothesis
Rgof::gof_test(x, NA, pnull, rnull, phat=phat)
```

```
#> $statistics
#>      KS       K       AD      CvM       W        ZA      ZK      ZC
#>   0.0208   0.0407   0.6500   0.0926   0.0911   3.2940  1.4780  12.4900
#>    ES-l-P   ES-s-P   EP-l-P   EP-s-P   ES-l-L   ES-s-L  EP-l-L   EP-s-L
#>   37.4800   9.6060   51.7300  11.9200  38.5700  9.4030  50.2600  12.4700
#>
#> $p.values
#>     KS      K     AD     CvM      W      ZA      ZK      ZC
#> 0.4712 0.4712   0.2328 0.2546  0.2354  0.6292  0.7692  0.5828
#> ES-l-P  ES-s-P  EP-l-P  EP-s-P  ES-l-L  ES-s-L  EP-l-L EP-s-L
#> 0.4011  0.2120  0.3302  0.1550  0.3542  0.2250  0.3840  0.1315
```

If we wish to find an adjusted p value for a combination of tests we can run

```
Rgof::gof_test_adjusted_pvalue(x, NA, pnull, rnull, phat=phat)
```

```
#> p values of individual tests:
#> W :  0.128
#> ZC :  0.232
#> AD :  0.08
#> ES-s-P :  0.212
#> adjusted p value of combined tests: 0.2267
```

Next we generate a data set from a t distribution with 5 degrees of freedom, so now the null hypothesis is false. Here and in the examples that follow we only show the p values of the tests:

```
y = rt(1000, 5) # data where null hypothesis is false
Rgof::gof_test(y, NA, pnull, rnull, phat=phat)[["p.values"]]
```

```
#>     KS       K      AD      CvM      W       ZA     ZK     ZC
#>   0.0106  0.0106  0.0000  0.0048   0.0034  0.0000  0.0000 0.0000
#>   ES-l-P  ES-s-P  EP-l-P  EP-s-P  ES-l-L  ES-s-L  EP-l-L  EP-s-L
#>   0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0408
```

If the quantile function (aka inverse) of the cdf is known it can be included. It is then used in some of the chi-square tests and the Wasserstein test. It can be passed to the routine that finds the test statistic(s) via the list TSextra:

```
TSextra = list(qnull=function(x, mu) qnorm(x, mu))
Rgof::gof_test(x, NA, pnull, rnull, phat=phat, TSextra=TSextra)[["p.values"]]
```

```
#>     KS      K      AD      CvM      W      ZA      ZK      ZC    Wassp1
#>  0.4710  0.4710  0.2288  0.2444  0.2238  0.6198  0.7664  0.5762  0.1760
#>  ES-l-P  ES-s-P  EP-l-P  EP-s-P  ES-l-L  ES-s-L  EP-l-L  EP-s-L
#>  0.4011  0.2120  0.5634  0.4413  0.3542  0.2250  0.5182  0.4210
```

A user can also use this routine to run their own test. For example, let's say we wish to include the Neyman's smooth test from the *DDST* (P Niecek 2016) package:

```
NeymanSmoothTest = function(x, pnull, param) {
  ts=as.numeric(unlist(ddst::ddst.norm.test(x))[1])
  names(ts) = "DDST"
  ts
}
Rgof::gof_test(x, NA, pnull, rnull, phat=phat, TS=NeymanSmoothTest)[["p.values"]]

#>   DDST
#> 0.5482

Rgof::gof_test(y, NA, pnull, rnull, phat=phat, TS=NeymanSmoothTest)[["p.values"]]

#> DDST
#>    0
```

The routine has to have the following form:

```
newTS(x, pnull, param, TSextra)
```

x is the data set and pnull the distribution function under the null hypothesis, as described above. param is the estimated parameters in the case of a composite null hypothesis and is ignored in the case without parameter estimation. The argument TSextra, a list of items also needed for calculating the test statistic, is optional.

Next we assume that the sample size was random and drawn from a Poisson distribution with rate 950. One of the consequences of this is that now the degrees of freedom of the chi-square tests is the number of bins - number of estimated parameters rather than number of bins - 1 - number of estimated parameters.

```
Rgof::gof_test(x, NA, pnull, rnull, phat=phat, TSextra=TSextra, rate=950)[["p.values"]]

#>     KS      K      AD     CvM      W       ZA      ZK      ZC    Wassp1
#>  0.4736  0.4736  0.2406  0.2580  0.2372  0.6206  0.7642  0.5782  0.1904
#>  ES-l-P ES-s-P EP-l-P EP-s-P ES-l-L ES-s-L EP-l-L EP-s-L
#>  0.2602 0.1210 0.4009 0.2779 0.2936 0.1517 0.4539 0.2957
```

As an example for the use of importance sampling weights we generate the data from a mixture of two normal random variables but as above test for a simple normal distribution with unknown mean.

```
rnull = function(mu=0) c(rnorm(500, -1), rnorm(500, 1))
x = rnull()
w=function(x, mu=0) dnorm(x, mu)/(dnorm(x, -1)/2+dnorm(x, 1)/2)
Rgof::gof_test(x, NA, pnull, rnull, w=w, phat=phat)[["p.values"]]

#>     KS      K     CvM      AD
#>  0.7990  0.7718  0.6330  0.4516

Rgof::gof_test(y, NA, pnull, rnull, w=w, phat=phat)[["p.values"]]

#>     KS      K    CvM     AD
#>     0       0     0      0
```

Discrete data

Here we will consider the following example. The null hypothesis specifies a binomial distribution with $n = 100$ trials and a success probability p, estimated from the data. As an example where the null hypothesis is false we generate data that is a mixture of a binomial distribution with $p = 0.5$ and a discrete uniform distribution on the integers from 30 to 70.

```
set.seed(1234)
vals = 0:100 # all possible values
pnull = function(p=0.5) pbinom(0:100, 100, p)
rnull = function(p=0.5) table(c(0:100,rbinom(1000, 100, p)))-1
phat = function(x) mean(0:100*x)/1000
x = rnull()
Rgof::gof_test(x, vals, pnull, rnull, phat=phat)$p.values

#>      KS      K      AD     CvM      W     Wassp1    l-P    s-P    l-L     s-L
#>  0.2320  0.2752  0.4858  0.3132  0.2460  0.6257  0.1599  0.5592  0.1456 0.1456

y = table(c(0:100, rbinom(900, 100, 0.5), sample(30:70, size=100, replace=TRUE)))-1
Rgof::gof_test(y, vals, pnull, rnull, phat=phat)$p.values

#>     KS      K       AD     CvM      W       l-P    s-P      l-L    s-L
#>  0.3846  0.3290  0.0000  0.0174  0.0114  0.0000  0.0000  0.0000  0.0000
```

Here we have an example where most tests correctly reject the null hypothesis but some do not.

Note that the routine *rnull* has to insure that all values of *vals* are present, even if many have counts of zero.

Again the user can provide his/her own test statistic. The routine has to be as follows:

```
newTS(x, pnull, param, vals, TSextra)
```

Here x is the counts and pnull the distribution function under the null hypothesis as described above. param is the estimated parameters in the case of a composite null hypothesis and is ignored in the case without parameter estimation. vals is the set of values where $P(X = vals) > 0$. The argument TSextra, a list of items also needed for calculating the test statistic, is optional.

7.2 Goodness-of-fit problem - power estimation

To estimate the power of the various tests we can use the function *gof_power*. It's arguments are the same as *gof_test*, as well as

- ralt: a function that generates data under the alternative hypothesis.
- param_alt: a vector of values to be passed to ralt.
- With.p.value=FALSE: set to TRUE if the new user supplied routine calculates p values.
- alpha=0.05: type I error probability to be used for test.
- B=1000: number simulation runs.

As an example say we wish to estimate the power of the tests when the null hypothesis specifies a normal distribution, but the data comes from a t distribution. We have 500 observations, and both the mean and standard deviation are estimated from the data. The package also includes the routine *plot_power*, which has as its argument the output from the *gof_power* command and draws the power curve. It is shown in figure 2.

```
pnull = function(x, p=c(0,1)) pnorm(x, p[1], p[2]) # cdf under null hypothesis
rnull = function(p=c(0,1)) rnorm(500, p[1], p[2]) # generate data under null hypothesis
phat = function(x) c(mean(x), sd(x)) # estimate parameters
TSextra = list(qnull = function(x, p=c(0,1)) qnorm(x, p[1], p[2])) # quantile function
ralt = function(df) rt(500, df) # generate data under alternative
tmp=Rgof::gof_power(pnull, NA, rnull, ralt, param_alt=4*1:10, phat=phat, TSextra = TSextra)
Rgof::plot_power(tmp, "df", "Standard Normal vs t Distributions")
```

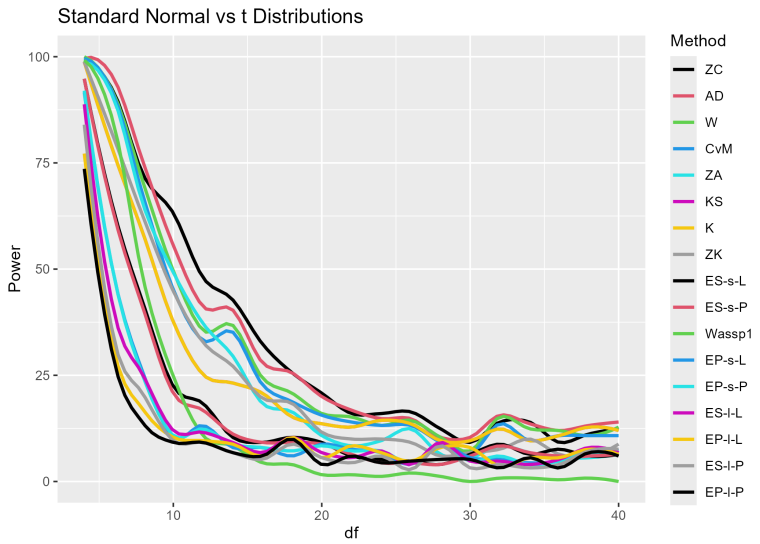


Figure 2: Power graph for goodness-of-fit-tests of normal vs t distributions, with mean and standard deviation estimated from the data.

plot_power has the arguments

- pwr: a matrix, usually the output of the *gof_power* command.
- xname: name of parameter of ralt.
- title=" ": title of graph
- Smooth=TRUE: should curves be smoothed?
- span=0.25: parameter for smoothing routine.

Power estimation for discrete data works the same. As an example consider the following. One data set comes from a Poisson distribution, restricted to the set of integers from 70 to 140. The second data set is a 50-50 mixture of Poisson random variables with a rate of 100 and a rate of $100 + \lambda$. The power graph is in figure 3.

```
vals=70:140
pnull=function(lambda) (ppois(70:140, lambda)-ppois(69,lambda))/(ppois(140, lambda)-ppois(69,lambda
rnull=function(lambda) {
  vals=70:140
  x=rpois(1000, lambda)
  x[x<70]=70
  x[x>140]=140
  x=table(c(70:140, x))-1
}
phat=function(x) sum(70:140*x)/1000
ralt=function(lambda) {
  vals=70:140
```

```
  x=c(rpois(500, 100), rpois(500, 100+lambda))
  x[x<70]=70
  x[x>140]=140
  x=table(c(70:140, x))-1
}
tmp=Rgof::gof_power(pnull, vals, rnull, ralt, param_alt=2*0:8, phat=phat)
Rgof::plot_power(tmp, "lambda", "Poisson vs Mixture of Poisson Distributions")
```

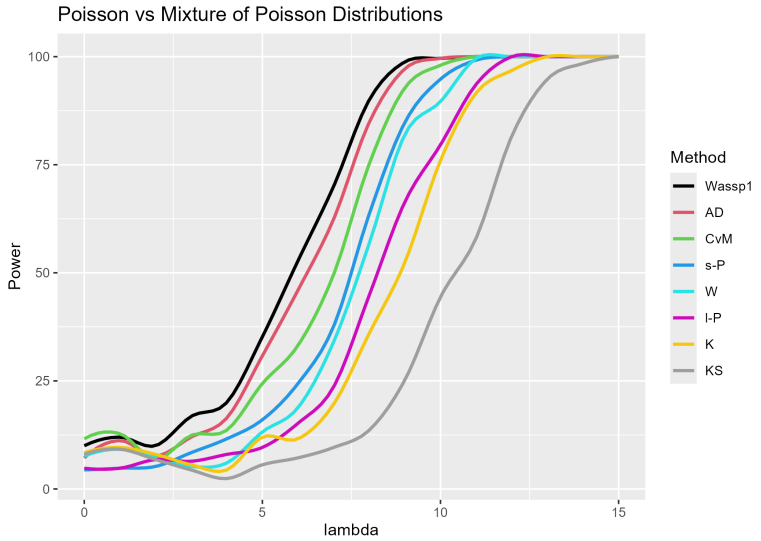


Figure 3: Power graph for goodness-of-fit-tests of Poisson vs a mixture of Poisson distributions, with the rate estimated from the data.

7.3 Two-sample problem - testing

p values are generally found using the permutation method. The idea of a permutation test is simple. Say we have data sets $x_1, .., x_n$ and $y_1, .., y_m$. They are combined into one large data set $x_1, .., x_n, y_1, .., y_m$, permuted and split again in sets of size n and m. Under the null hypothesis these new data sets come from the same distribution as the actual data. Therefore calculating the tests statistics for them and repeating many times one can build up the distributions of the test statistics and find p values from them.

In the case of continuous data the routines also allow for the use of large sample formulas for some of the tests. In the discrete case none of the methods (outside of the chi-square tests) has a known large sample theory.

In the discrete case the permutation method is somewhat more complicated. Say we have data sets with values $v_1, .., v_k$ and counts $x_1, .., x_k, y_1, .., y_k$. One can then simply expand these to yield a large data set with $x_1 + y_1$ v_1's, $x_2 + y_2$ v_2's and so on. Then this vector is permuted and split as described above. The drawback of this sampling method is that its calculation speed increases with the sample size and would be impossible for data sets with very large counts.

Alternatively *R2sample* provides the following option. One can show that the distribution of the permuted data sets is as follows: let $n = \sum x_i$ and $m = \sum y_i$, then

$$P(\boldsymbol{X} = \boldsymbol{a}|\boldsymbol{x}, \boldsymbol{y}) = \left[\prod_{j=1}^{k} \binom{x_j + y_j}{a_j}\right] / \binom{n+m}{n}$$

for any $\boldsymbol{a}$ such that $0 \leq a_i \leq x_i; i = 1, .., k$ and $\sum a_i = n$.

It is possible to sample from this distribution as follows: Let N and M be the sample sizes of the two data sets, respectively. Let $p = N/(N+M)$ be the proportion of events

in the first data set. Say that in a data set we have x_1 observations of the smallest discrete value v_1 in the first data set and y_1 in the second. We can then generate a random sample by drawing an observation from a Binomial with parameters $x_1 + y_1$ and p. We repeat this for all values of the discrete random variable. We also need to insure, though, that the total number of observations in the first simulated data set is again N and in the second data set M. If this is not so, we randomly choose two values, giving a higher priority for those with high counts, and flipping one observation between the two data sets. This is repeated until the simulated data set has N events total.

The routine is *twosample_test*. It's arguments are

- x, y: the two data sets, either the observations in the continuous case or the counts in the discrete case. x can also be a list with elements x and y, and then y is ignored.
- vals=NA: the possible values of the discrete random variables, or NA for continuous data.
- TS: routine to calculate test statistics other than those included
- TSextra: a list passed to TS
- wx = rep(1, length(x)): weights for x data
- wy = rep(1, length(y)): weights for y data
- B=5000: number of simulation runs
- nbins=c(50,10): number of bins to use in chi -quare tests
- minexpcount=5: required minimal expected counts for chi-square tests
- maxProcessors: number of cores to use for parallel processing, default is 1 less than are detected on computer
- UseLargeSample: should large sample approximations be used instead of permutation?
- samplingmethod="Binomial": sampling method for discrete data.
- rnull: a function that generates data from a model, possibly with parameter estimation. This is needed in the goodness-of-fit/two-sample hybrid problem.
- doMethod="all": vector with names of methods, if not all are to be included.

The arguments match those of *gof_test*, where this makes sense.

Continuous data

The x and $y1$ data sets come from a standard normal distribution, and $y2$ from a normal distribution with mean 1.

```
x = rnorm(100)
y1 = rnorm(150)
y2 = rnorm(150, 1)
R2sample::twosample_test(x, y1)[["p.values"]]
```

```
#>      KS   Kuiper    CvM     AD       LR      ZA     ZK      ZC
#>   0.9514  0.8730  0.9242  0.9126  0.8992  0.7480  0.4502  0.7888
#>   Wassp1 ES large ES small EP large EP small
#>   0.8580   0.6309   0.2454   0.4334   0.9496
```

```
R2sample::twosample_test(x, y2)[["p.values"]]
```

```
#>       KS   Kuiper      CvM       AD       LR       ZA       ZK       ZC
#>        0        0        0        0        0        0        0        0
#>   Wassp1 ES large ES small EP large EP small
#>        0        0        0        0        0
```

Again, the user can provide their own test statistic:

```
DiffStandardizedMeans = function(x, y) {
   TS = abs(mean(x)/sd(x)-mean(y)/sd(y))
   names(TS) = "DSM"
   TS
}
R2sample::twosample_test(x, y1, TS=DiffStandardizedMeans)[["p.values"]]

#>    DSM
#>   0.585

R2sample::twosample_test(x, y2, TS=DiffStandardizedMeans)[["p.values"]]

#> DSM
#>   0
```

The user supplied routine has to be a function of the two data sets x and y and optionally a list TSextra.

As an example for weighted data, let's say the x data set actually came from a t distribution with 5 degrees of freedom:

```
x = rt(100, 5)
wx = dnorm(x)/dt(x, 5)
R2sample::twosample_test(x, y1, wx=wx)[["p.values"]]

#>       KS   Kuiper    CvM     AD    ES large  ES small  EP large  EP small
#>    0.2400  0.2354  0.2522  0.2560   0.7627    0.8632    0.3814    0.8636

R2sample::twosample_test(x, y2, wx=wx)[["p.values"]]

#>       KS   Kuiper      CvM       AD ES large ES small EP large EP small
#>    0.0004   0.0004   0.0000   0.0000   0.0001   0.0000   0.0270   0.0000
```

If the data sets are very large using permutation to derive the null distribution of the test statistics can be very slow. In this case one can use the argument *UseLargeSample=TRUE*. This will be done automatically if both sample sizes are at least 10000.

```
x = rnorm(1e5)
y1 = rnorm(1e5)
y2 = rnorm(1e5, 0.02)
R2sample::twosample_test(x, y1)[["p.values"]]

#>      KS   Kuiper    CvM      AD
#>   0.098  0.1214   0.1602   0.1330
#>   ES large  ES small   EP large  EP small
#>    0.2683    0.3347     0.0974    0.1988

R2sample::twosample_test(x, y2)[["p.values"]]

#>     KS    Kuiper      CvM     AD
#>  0.0112  0.0542    0.0046  0.0013
#>   ES large  ES small  EP large  EP small
#>    0.0935     0.0074   0.6170    0.0717
```

Discrete data

As an example for the case of discrete data we will use two data sets from geometric random variables with slightly different rates. x and y have to have the same length as vals and x+y has to be positive for all values in vals.

```
x = table(rgeom(1000, 0.7))
y = table(rgeom(1000, 0.8))
vals = unique(c(names(x), names(y))) # all values from either x or y
x1 = rep(0, length(vals))
names(x1)=vals
y1 = x1
x1[names(x)]=x
y1[names(y)]=y
vals = as.numeric(vals)
R2sample::twosample_test(x1, y1, vals)[["p.values"]]

#>     KS Kuiper    CvM     AD     LR     ZA Wassp1  large  small
#>      0      0      0      0      0      0      0      0      0
```

Again the user can supply their own test. The routine has to be a function of the two data sets x and y and a vector *vals* of possible values of the discrete random variable. A list TSextra is optional.

7.4 Two-sample problem - power estimation

The package *R2sample* includes the routine *twosample_power*. The arguments TS, TSextra, B, nbins, minexpcount, UseLargeSample and maxProcessor are the same as in *twosample_test*. In addition we have

- f: a function that generates a list with two vectors called x and y and (in the case of discrete data) a vector vals. The function can have zero, one or two arguments.
- ... arguments passed to f.
- With.p.value=FALSE: set to TRUE is user supplied routine calculates p values.
- alpha=0.05: type I error probability for the tests.

As an example for continuous data we again consider the case of normal vs t distributions. One data set with 10001 observations comes from a standard normal distribution, the other with 10002 observations from a t distribution with df degrees of freedom. Because of the large sample sizes the large sample approximations are used.

```
f=function(df) {
  x=rnorm(10001)
  y=rt(10002, df)
  list(x=x, y=y)
}
tmp=R2sample::twosample_power(f, df=seq(5, 100, 5))
R2sample::plot_power(tmp, "df", "Data from Standard Normal vs t Distributions")
```

For discrete data we again consider the case of a Poisson random variable vs a 50-50 mixture of Poisson random variables:

```
f=function(a) {
  vals=70:140
```

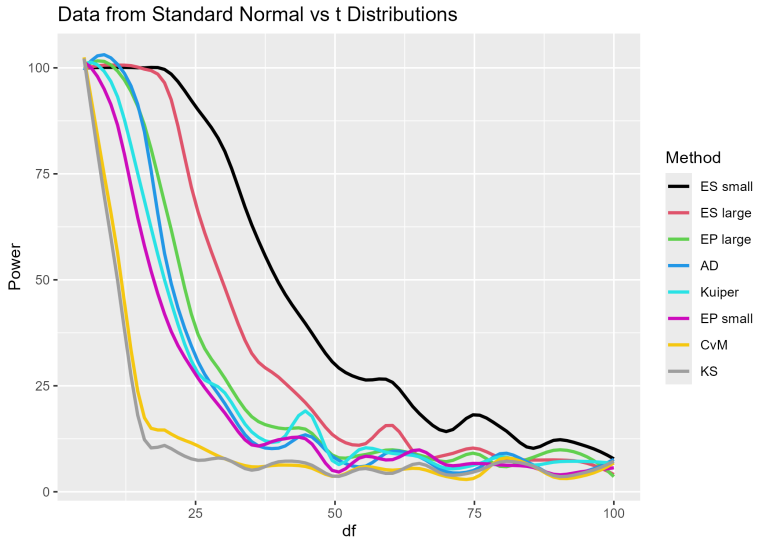


Figure 4: Power graph for two-sample tests where one data set comes from a standard normal distribution and the other from a t distribution.

```
  x=rpois(1000, 100)
  x[x<70]=70
  x[x>140]=140
  x=table(c(70:140, x))-1
  y=c(rpois(500, 100), rpois(500, 100+a))
  y[y<70]=70
  y[y>140]=140
  y=table(c(70:140, y))-1
  I=seq_along(vals)[x+y>0]
  list(x=x[I], y=y[I], vals=vals[I])
}
tmp=R2sample::twosample_power(f, a=seq(0,5,0.25))
R2sample::plot_power(tmp, "lambda", "Data from Poisson vs Mixture of Poisson Distributions")
```

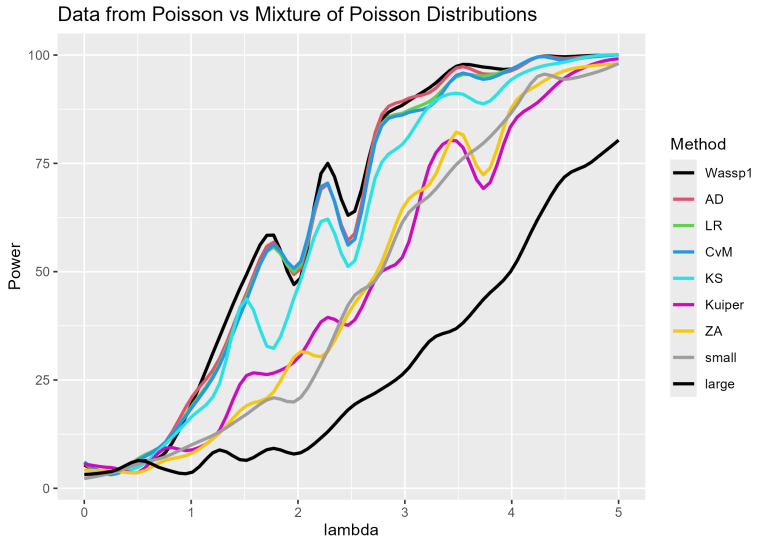


Figure 5: Power graph for two-sample tests where one data set comes from Poisson distribution and the other from a mixture of two Poisson distributions.

8 Benchmarking

Say a researcher has developed a new method for the univariate goodness-of-fit problem and wants to see how it stacks up in comparison to the standard methods such as the chi-square tests or the Kolmogorov-Smirnov test. Both *Rgof* and *R2sample* packages include the routine *run_studies*, which makes this very easy.

As a specific example say we wish to see whether the Kolmogorov-Smirnov test or the Anderson-Darling test has better power when the null hypothesis specifies a normal distribution with mean and standard deviation unspecified, but in reality the data comes from a t distribution. Say we run the *Rgof::gof_power* command for a sample size of 500, a t distribution with 8 degrees of freedom and a true type I error of 5%, and we find that the Kolmogorov-Smirnov test has a power of 43% whereas the Anderson-Darling test has a power of 70%. It is then true that the Anderson-Darling test will also have a higher power for any other combination of sample size, degrees of freedom and true type I error. In order to assess the ranking of the methods it therefore suffices to run each case study with just one combination of n, α and the parameter under the alternative.

8.1 Goodness-of-fit problem

Here the user needs to create a function that finds either the test statistic or, if possible, the p value of the new test. Its arguments should be as described previously. Say the routine is called myTS and is designed for continuous data and calculates the test statistic. Then the user can run

```
run_studies(myTS)
```

This will run 20 different case studies and provide some information on the relative power of the new method when compared to those included in *Rgof*. For a list of the case studies see the appendix.

If the routine actually calculates a p value, run instead

```
run_studies(myTS, With.p.value=TRUE)
```

This will of course be much faster as it does not require simulation.

The arguments of *run_studies* are

- TS: the name of the new test routine
- study: the name of the study to run, or all studies if missing
- TSextra: a list of additional info passed to TS, if such are needed
- With.p.value=FALSE: TRUE if routine finds p values
- BasicComparison=TRUE: if TRUE the values for sample size, type I error etc from included studies are used.
- nsample = 500: desired sample size
- alpha = 0.05: desired type I error probability
- param_alt: (list of) parameters for alternative distributions
- maxProcessor: number of cores to use for parallel programming, number of cores - 1 if missing
- B = 1000: number of simulation runs

As an example we will use the R built-in ks.test to do the Kolmogorov-Smirnov test:

```
myTS=function(x, pnull, param) {
   if(length(formals(pnull))==1) # case studies without parameter estimation
      mypnull=function(x) pnull(x)
  else mypnull=function(x) pnull(x, param) # case studies with parameter estimation
   z=ks.test(x, mypnull)[["p.values"]]
   names(z)="RKS"
   z
}
pwrs=run.studies(myTS, With.p.value = TRUE)

#> Average number of studies a method is close to the best::
#> EP-l-P ES-l-P EP-l-L  RKS   ES-l-L EP-s-P EP-s-L   KS      K  ES-s-P
#>  5.450  5.9   6.200  6.900  6.950  8.875  9.175  9.400  9.400 10.175
#>   W      CvM  ES-s-L  ZA      ZK   Wassp1   ZC     AD
#> 10.525 10.550 10.850 11.625 11.850 12.050 12.300 13.050
```

Note that the performance of RKS is much lower than that of the Kolmogorov-Smirnov test included in the *Rgof* package. This is due to the fact the *R* routine ks.test does not actually allow for parameter estimation.

As another example say the user routine calculates the test statistic and the user wants to find the power of the methods in the case where the null hypothesis specifies a normal distribution with mean and standard deviation estimated from the data, but the true distribution is a t distributions with df degrees of freedom. He also wants a sample size of 1000 and a true type I error of 0.1 Then he can run

```
run_studies(myTS, "normal.t.est", nsample=1000, alpha=0.1, param_alt=5*1:10)
```

8.2 Two-sample problem

This works in the exact same way as in the goodness-of-fit problem.

8.3 Real data examples

R sunspots data

This data set has the monthly mean relative sunspot numbers from 1749 to 1983, collected at Swiss Federal Observatory, Zurich until 1960, then Tokyo Astronomical Observatory. It is part of the base *R* program. It is of course a time series, but we will treat it here as if it were independent data.

A histogram of the data suggests that an exponential model might be a good fit. However, all the tests in *Rgof* reject that null hypothesis:

```
library(ggplot2)
dta=data.frame(Sunspots=c(sunspots))
ggplot(dta, aes(x=Sunspots)) +
  geom_histogram(color="black", fill="white") +
  ggtitle("Sunspots Data")


pnull = function(x, p=1) pexp(x, p)
TSextra=list(qnull= function(x, p=1) qexp(x, p))
rnull = function(p) rexp(N, p)
phat = function(x) 1/mean(x)
Rgof::gof_test(c(sunspots), NA, pnull, rnull, phat=phat, TSextra = TSextra, Range=c(0, Inf))
```

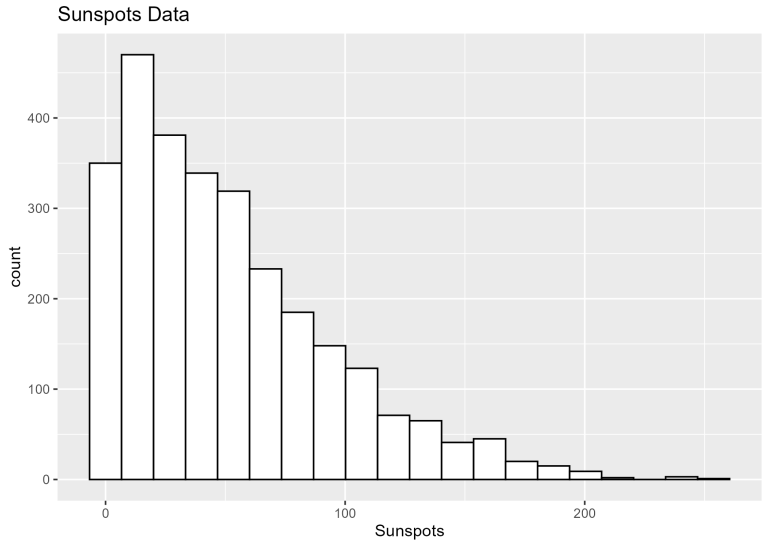


Figure 6: Histogram of sunspots data set.

```
#> $statistics
#>      KS     K      AD     CvM      W        ZA     ZK       ZC
#>   0.0631 0.1260  0.000  2.8873  1.8894   0.0000  0.0000  0.0000
#>   Wassp1   ES-l-P   ES-s-P   EP-l-P   EP-s-P   ES-l-L   ES-s-L   EP-l-L
#>   5.2559 154.7900 100.0300 252.8700  84.6450 170.2800 122.7200 255.0200
#>   EP-s-L
#>  84.1610
#>
#> $p.values
#>     KS      K     AD    CvM      W     ZA     ZK     ZC Wassp1 ES-l-P ES-s-P
#>      0      0      0      0      0      0      0      0      0      0      0
#> EP-l-P EP-s-P ES-l-L ES-s-L EP-l-L EP-s-L
#>      0      0      0      0      0      0
```

Death by horsekicks data

This is the famous data set first discussed in Bortkewitsch (1898) and analyzed in many statistics text books and articles, see for example Preece et al. (1988). It is the number of soldiers in the Prussian army killed by being kicked by a horse between 1875 to 1894. The data is

Number of Deaths	Frequencies
0	109
1	65
2	22
3	3
4	1
5+	0

We will run the tests to see whether a Poisson model might fit this data set:

```
vals=0:5
x=c(109, 65, 22, 3, 1, 0)
pnull = function(lambda=1) c(ppois(0:4,lambda), 1)
rnull = function(lambda=1) {
  x =  rpois(200, lambda)
```

```
  x[x>5]=5
  table(c(0:5, x))-1
}
phat = function(x) sum(0:5*x)/200
Rgof::gof_test(x, vals, pnull, rnull, phat=phat)

#> $statistics
#>     KS      K     AD    CvM      W Wassp1    l-P    s-P    l-L    s-L
#> 0.5434 0.5481 0.0323 0.0026 4.5350 0.0124 0.0628 0.0628 0.0625 0.0625
#>
#> $p.values
#>     KS      K     AD    CvM      W Wassp1    l-P    s-P    l-L    s-L
#> 0.4830 0.6372 0.9434 0.9412 0.5288 0.9386 0.8021 0.8021 0.8026 0.8026
```

All methods fail to reject the null hypothesis, so the Poisson distribution is a good model for the horsekick data.

9 Conclusion

The R packages *Rgof* and *R2sample* bring together a large number of methods for the univariate goodness-of-fit problem and the univariate two-sample problem. The routines make it easy to run these tests simultaneously. They are implemented for both continuous and discrete data and can handle a number of different situations such as random sample sizes and weighted data. The packages also include routines for power estimation as well as routines for benchmarking new tests.

10 Appendix

10.1 Case Studies for Goodness-of-fit Problem

Without parameter estimation

1. uniform.linear — U[0,1] vs a linear model on [0,1] with slope s.
2. uniform.quadratic — U[0,1] vs a quadratic model with vertex at 0.5 and some curvature a.
3. uniform.bump — U[0,1] vs U[0,1]+N(0.5,0.05).
4. uniform.sine — U[0,1] vs U[0,1]+Sine wave
5. beta22.betaaa — Beta(2,2) vs Beta(a,a)
6. beta22.beta2a — Beta(2,2) vs Beta(2,a)
7. normal.shift — N(0,1) vs N(μ,1)
8. normal.stretch — N(0,1) vs N(0, σ)
9. normal.t — N(0,1) vs t(df)
10. normal.outlier1 — N(0,1) vs N(0,1)+U[2,3]
11. normal.outlier2 — N(0,1) vs N(0,1)+U[-3,-2]+U[2,3]
12. exponential.gamma — Exp(1) vs Gamma(1,b)
13. exponential.weibull — Exp(1) vs Weibull(1,b)
14. exponential.bump — Exp(1) vs Exp(1)+N(0.5,0.05)

15. trunc.exponential.linear Exp(1) vs Linear, on [0,1]

With parameter estimation

16. normal.t.est N(μ,σ) vs t(df)
17. exponential.weibull.est Exp(λ) vs Weibull(1,b)
18. trunc.exponential.linear.est Exp(λ) vs Linear, on [0,1]
19. exponential.gamma.est Exp(λ) vs Gamma(1,b)
20. normal.cauchy.est N(μ,σ) vs Cauchy (Breit-Wigner)

10.2 Case Studies for two-sample problem

The first 14 case studies are the same as those of the goodness-of-fit problem. The others are

15. gamma.normal Gamma(μ) vs N($\bar{x}$, sd(x)), here the mean of the normal distribution are the sample mean and sample standard deviation of the x data set.
16. normal.normalmixture N(0,1) vs N($-\mu$,1)+N(μ,1)
17. uniform.uniformmixture U[0,1] vs. αU[0,1/2]+(1-α)U[1/2,1]
18. uniform.betamixture U[0,1] vs. αU[0,1/2]+(1-α)Beta(2,2)
19. chisquare.noncentral $\chi^2(5)$ vs. $\chi^2(5,\tau)$
20. uniform.triangular U[0,1] vs. triangular

References

S. Aldor-Noima, L. D. Brown, A. Buja, R. A. Stine, and W. Rolke. The power to see: A new graphical test of normality. *The American Statistician*, 67, 2013. [p143]

T. W. Anderson. On the distribution of the two-sample Cramer-von Mises criterion. *Annals of Mathematical Statistics*, 33(3):1148–1159, 1962. [p142]

T. W. Anderson and D. A. Darling. Asymptotic theory of certain goodness-of-fit criteria based on stochastic processes. *Annals of Mathematical Statistics*, 23:193–212, 1952. [p142]

C. J. G. Bellosta. *ADGofTest: Anderson-Darling GoF test*, 2011. URL https://CRAN.R-project.org/package=ADGofTest. R package version 0.3. [p138]

J. Berkson. Minimum chi-square, not maximum likelihood. *Ann. Math. Stat*, 8(3):457–487, 1980. [p141]

P. J. Bickel and K. A. Doksum. *Mathematical Statistics Vol 1 and 2*. CRC Press, 2015. [p138]

L. Bortkewitsch. *Das Gesetz der kleinen Zahlen*. Teubner, 1898. [p157]

G. Casella and R. Berger. *Statistical Inference*. Duxbury Advanced Series in Statistics and Decision Sciences. Thomson Learning, 2002. [p138]

H. Cramer. On the composition of elementary errors. *Scandinavian Actuarial Journal*, 1:13–74, 1928. [p142]

R. B. D'Agostini and M. A. Stephens. *Goodness-of-Fit Techniques*. Statistics: Textbooks and Monographs. Marcel Dekker, 1986. [p138]

C. Dowd. *twosamples: Fast Permutation Based Two Sample Tests*. None, 2022. URL https://CRAN.R-project.org/package=twosamples. R package version 2.0.0. [p138]

D. Eddelbuettel, R. Francois, J. Allaire, K. Ushey, Q. Kou, N. Russell, I. Ucar, D. Bates, and J. Chambers. *Rcpp: Seamless R and C++ Integration*, 2024. URL https://CRAN.R-project.org/package=Rcpp. R package version 1.0.12. [p138]

J. Faraway, G. Marsalia, J. Marsalia, and A. Baddeley. *goftest: Classical Goodness-of-Fit Tests for Univariate Distributions*. None, 2007. URL https://CRAN.R-project.org/package=goftest. R package version 1.2.3. [p138]

R. A. Fisher. On the interpretation of chi square from contingency tables, and the calculation of P.J.R. *Journal of the Royal Statistcal Society*, 85:87–94, 1922. [p141]

R. A. Fisher. The conditions under which chi square measures the discrepancy between observation and hypothesis. *Journal of the Royal Statistcal Society*, 87:442–450, 1924. [p141]

J. Gross and U. Ligges. *nortest: Tests for Normality*, 2015. URL https://CRAN.R-project.org/package=nortest. R package version 1.0-4. [p138]

A. Kolmogorov. Sulla determinazione empirica di una legge di distribuzione. *G. Ist. Ital. Attuari.*, 4:83–91, 1933. [p141]

N. H. Kuiper. Tests concerning random points on a circle. *Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen*, 63:38–47, 1960. [p142]

E. Lehmann. Consistency and unbiasedness of certain nonparametric tests. *Ann. MAth. Statist.*, 22(1):165–179, 1951. [p143]

S. P. Millard and A. K. Kowarik. *EnvStats: Package for Environmental Statistics, Including US EPA Guidance*. None, 2017. URL https://CRAN.R-project.org/package=EnvStats. R package version 3.5.0. [p138]

A. Pettitt. A two-sample anderson-darling rank statistic. *Biometrika*, 63 No.1:161–168, 1976. [p142]

D. A. Preece, G. J. S. Ross, and P. J. Kirby. Bortkewitsch's horse-kicks and the generalised linear model. *Journal of the Royal Statistical Society*, 37, 1988. [p157]

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/. [p138]

J. C. Raynor, O. Thas, and D. J. Best. *Smooth Tests of Goodness of Fit*. Wiley Sons, 2012. [p138]

W. Rolke. *R2sample: 1d Two-Sample Tests*. None, 2023a. URL https://CRAN.R-project.org/package=R2sample. R package version 1.0.1. [p139]

W. Rolke. *Rgof: 1d Goodness of Fit Tests*. None, 2023b. URL https://CRAN.R-project.org/package=Rgof. R package version 1.0.1. [p139]

M. Rosenblatt. Limit theorems associated with variants of the Von Mises statistic. *Ann. Math. Statist.*, 23:617–623, 1952. [p143]

N. Smirnov. Estimate of deviation between empirical distribution functions in two independent samples. *Bull. Moscow Univ.*, 2:3–16, 1939. [p141]

B. A. Taylor and J. W. Emerson. *dgof: Discrete Goodness-of-Fit Tests*. None, 2009. URL https://CRAN.R-project.org/package=dgof. R package version 1.5.1. [p138]

O. Thas. *Continuous Distributions*. Springer Series in Statistics. Springer, 2010. [p138]

L. N. Vaserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 5(3):64–72, 1969. [p142]

R. E. von Mises. *Wahrscheinlichkeit, Statistik und Wahrheit*. Springer, 1928. [p142]

G. S. Watson. Goodness-of-fit tests on a circle. *Biometrica*, 48:109–114, 1961. [p143]

J. Zhang. Powerful goodness-of-fit tests based on likelihood ratio. *Journal of the RSS (Series B)*, 64:281–294, 2002. [p142]

J. Zhang. Powerful two-sample tests based on the likelihood ratio. *Techometrics*, 48:95–103, 2006. [p142]

Wolfgang Rolke
University of Puerto Rico - Mayaguez
Department of Mathematical Sciences
Mayaguez, Puerto Rico, USA, 00681
ORCiD: 0000-0002-3514-726X
wolfgang.rolke@upr.edu

Rendering LaTeX in R

by Paul Murrell

Abstract The xdvir package provides functions for rendering LaTeX fragments as labels, annotations, and data symbols in R plots. There are convenient high-level functions for rendering LaTeX fragments, including labels on ggplot2 plots, plus lower-level functions for more fine control over the separate authoring, typesetting, and rendering steps. There is support for making use of LaTeX packages, including TikZ graphics. The rendered LaTeX output is fully integrated with R graphics output in the sense that LaTeX output can be positioned and sized relative to R graphics output and vice versa.

1 Introduction

Text labels, titles, and annotations are essential components of any data visualization. Viewers focus a lot of their attention on text (Borkin et al., 2016), text is the most effective way to communicate some types of information (Hearst, 2023), and the message obtained from a data visualization can be heavily influenced by the text on a plot (Kong et al., 2018).

R provides relatively flexible tools for adding text labels to plots. For example, in the `graphics` package, we can specify an overall plot title and axis titles via the `main`, `xlab`, and `ylab` arguments to the `plot()` function and we can add text at arbitrary locations on the plot with the `text()` and `mtext()` functions.

Unfortunately, these core tools for drawing text are quite limited in terms of the formatting of the text. For example, there is no facility for emphasizing an individual word using a **bold** or *italic* face within a text label.

The **ggtext** (Wilke and Wiernik, 2022a) and **gridtext** (Wilke and Wiernik, 2022b) packages greatly improved the situation by allowing text labels to include a small subset of markdown and HTML (plus CSS). This allowed, for example, changes in font face and color within text labels.

More recently, the **marquee** package (Pedersen and Mitáš, 2025) improved the situation a great deal further by providing full support for markdown within text labels. This made it possible to lay out more complex arrangements of text and even graphical content within text labels.

However, despite these advances, there are still some text formatting tasks that remain out of reach. For example, Figure 1 shows a plot with a text annotation in the top-right corner that contains a combination of features that cannot be produced using the available text-drawing tools.

The annotation in Figure 1 may not appear to be particularly special nor particularly complicated at first glance, but it harbors several important details:

- The text is a mixture of plain text and mathematical expressions (like $\bar{z}_i$). Furthermore, the mathematical expressions use a different font (Latin Modern) than the plain text (TeX Gyre Adventor) and the mixture is broken across multiple lines.

  The R graphics system can draw mathematical expressions (Murrell and Ihaka, 2000) and that includes a mixture of plain text and mathematical expressions. Furthermore, the R graphics system uses a separate symbol font for mathematical expressions compared to plain text. However, further changes in font within the plain text are not possible and line breaks are not supported. There is also the problem that the typesetting of mathematical expressions in R graphics is not of a very high quality.

- The text is not all the same color; the final two words (but not the full stop) are red. Furthermore, the final two words are **bold**; they have a different font face compared to the rest of the text.

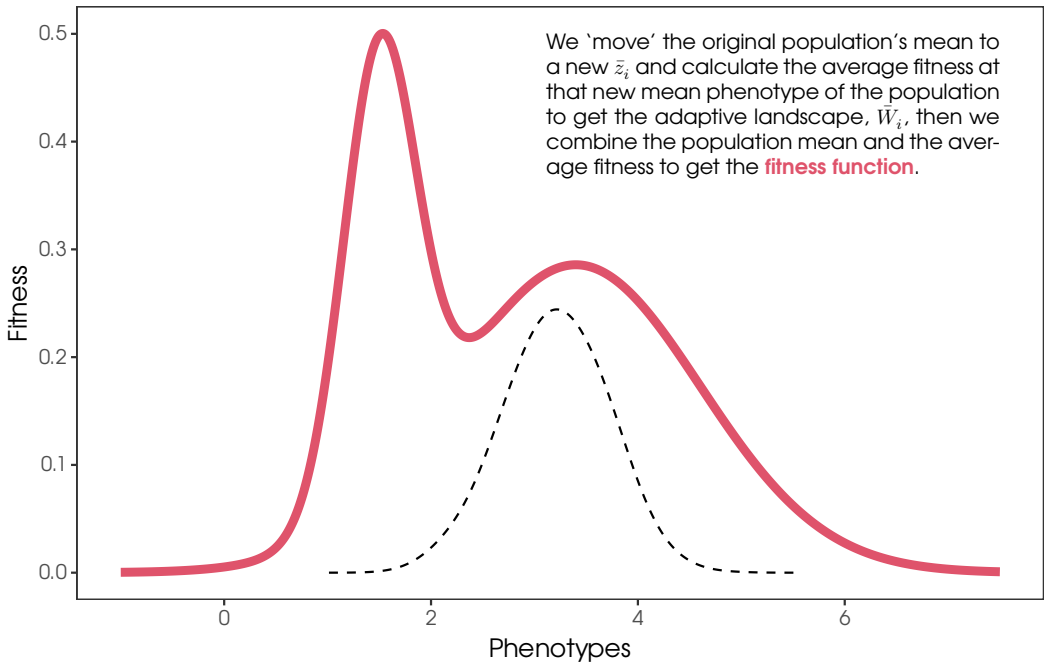


Figure 1: A plot with a text annotation in the top-right corner that contains several typesetting challenges: in-line mathematical expressions like $\bar{z}_i$; changes in color so that the last two words match the colour of the thicker line in the plot; and automated line-breaks with full justification and hyphenation.

The R graphics system can only draw a character value with a single color and a single font face. The **gridtext** and **ggtext** packages make it possible to change color within a character value, but they do not allow a mixture of character values and R mathematical expressions. Furthermore, they do not provide support for MathML `<math>` tags.

- The text is broken over multiple lines. Furthermore, the text is fully justified (not ragged-left or ragged-right justified) and one word has been split across lines and hyphenated. Although it is not obvious from the plot itself, the line breaks were also automatically generated to fit the text into a fixed width.

  The R graphics system can draw a character value across multiple lines, but only if explicit newlines are embedded in the character value (i.e., the line breaks are manual). The base function `strwrap()` can be used to break lines, but it is only designed for monospaced terminal output and ragged-right justification. The **gridtext** and **ggtext** packages can calculate simple automated line breaks, but they will not break a word across lines (or hyphenate) and they cannot fully justify the resulting text. The **marquee** package can automate line breaks and fully justify text, but it cannot hyphenate nor can it produce mathematical expressions.

The features outlined above are all examples of *typesetting*; determining an arrangement of individual characters and symbols (glyphs) that could be as simple as placing one character after another (from left to right), but could also be as complex as arranging mathematical symbols, splitting text into multiple columns, or writing text vertically from top to bottom.

From R 4.3.0, it has been possible to draw text from a set of typeset glyphs using the functions `grDevices::glyphInfo()` and `grid::grid.glyph()` (Murrell et al., 2023). This facility offers the promise of being able to render arbitrary typeset text in R. However, it presupposes that we are able to generate a set of typeset glyphs.

The **marquee** package provides an example of a package that can generate typeset glyphs. It is capable of converting markdown input into a set of glyphs and their positions, which are then rendered in R.

This article describes the `xdvir` package, which is another example of a package that can generate typeset glyphs. In this case, the input is LaTeX, a TeX engine processes the LaTeX source to create DVI output, which is essentially a set of glyphs and their positions, and then `xdvir` reads the DVI output and renders the result in R.[1] The benefit of the `xdvir` package is that it provides access to the typesetting capabilities of LaTeX, which includes hyphenation, fully justified text, mixtures of plain text and mathematical expressions—all of the features demonstrated in Figure 1—and much more.

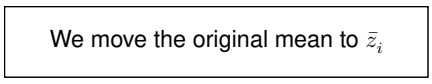
The next section describes the basic usage of the `xdvir` package. This is followed by a section that breaks down the design of the `xdvir` package to show the steps that are required to render LaTeX output in R. Subsequent sections explore several of the complexities that can arise with rendering LaTeX text in R graphics and some of the solutions that the `xdvir` package provides. The article ends with several extended examples of rendering LaTeX text in R.

2 LaTeX text labels in R

The simplest way to draw LaTeX text with the `xdvir` package is to call the `grid.latex()` function. The first argument to this function is a character value, which is interpreted as a fragment of LaTeX code. For example, the following code draws a text label that contains a subset of the larger annotation from Figure 1. We use just a subset here in order to keep the code readable.

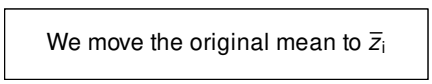
Because LaTeX code tends to contain a large number of backslashes, the code below uses the `r"(...)"` syntax for raw character constants, so that we do not have to escape each backslash with a double backslash. The resulting image is shown below the code. Although it is not immediately obvious from that image, the text, or rather the glyphs, in the image are rendered by R.

```
library(xdvir)

simpleTeX <- r"(We move the original mean to $\bar z_i$)"

grid.latex(simpleTeX)
```



It is possible to produce something similar to this result using the *plotmath* feature in R, as shown in the following code (and the image below the code). However, this demonstrates that one advantage of using `xdvir`, even for a simple piece of text like this, is the superior quality of the LaTeX fonts and typesetting for mathematical expressions.

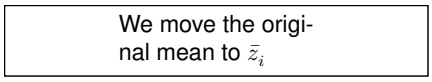
```
plotmath <- expression("We move the original mean to "*bar(italic(z))[i])

grid.text(plotmath)
```



[1] This process mirrors one way of working with LaTeX documents: LaTeX source (a `.tex` file) is processed by a TeX engine to produce DVI output (a `.dvi` file) and then the DVI output is processed to render PDF output (a `.pdf` file). The final processing step is performed by a *DVI driver*, for example `dvipdf` for PDF output or `dvips` for PostScript output. Different TeX engines produce different DVI output. For example, LuaTeX produces a `.dvi` file, but XeTeX produces a slightly different DVI output in the form of a `.xdv` file. Consequently, there are different DVI drivers for different DVI output, for example `xdvipdfmx` for processing a `.xdv` file to a `.pdf` file. The `xdvir` package is essentially a DVI driver that works for both `.dvi` files and `.xdv` files and produces R output, hence the name `xdvir`.

Another immediate benefit of xdvir is that we can automatically fit the text within a specified width. For example, the following code draws the LaTeX fragment tex again, but this time forces it to fit within a column that is half the width of the image.

```
grid.latex(simpleTeX, width=.5)
```



As the function name grid.latex() suggests, that function produces low-level drawing in the grid package graphics system. The text is just drawn relative to the current grid viewport, wherever that may be on the page. While this is extremely flexible, it is more likely that we want to combine and coordinate the text with a high-level plot of some sort, like the annotation in Figure 1. There are various ways that low-level grid drawing can be combined with a high-level plot, but we will leave those demonstrations to later sections.

Instead, for now, we will demonstrate a more common scenario: drawing LaTeX text labels on a **ggplot2** plot (Wickham, 2016). For this purpose, the xdvir package provides the element_latex() function. This allows us to specify a LaTeX fragment as a plot label and then we can indicate the special nature of the label via the ggplot2::theme() function.

For example, the following code uses the same LaTeX fragment from the example above as the title of a **ggplot2** plot. The resulting plot is shown in Figure 2. One detail about this result is that the text in this title is larger than the text drawn by the call to grid.latex() above, even though exactly the same TeX fragment is being drawn. A closer inspection reveals that the font is also different. These differences reflect the fact that grid.latex() and element_latex() respect the graphical parameter settings—font families and font sizes—that are in effect when the LaTeX fragment is drawn. In Figure 2 that means respecting the theme settings of the **ggplot2** plot. The ggIntro object in the code below contains a description of the main **ggplot2** plot from Figure 1. The code for generating ggIntro is not shown in order to keep the code below readable, but it is available in the supplementary materials for this article.

```
library(ggplot2)

ggIntro +
    labs(title=simpleTeX) +
    theme(plot.title=element_latex())
```

The xdvir package also provides a geom_latex() function for drawing text labels, similar to the standard ggplot2::geom_text() function. The values specified for the label aesthetic for geom_latex() are treated as fragments of LaTeX code. For example, Figure 3 shows a plot with a set of red points and a set of red labels, one for each point. The points are drawn using the standard ggplot::geom_point() function, but the labels are drawn using geom_latex() from the xdvir package. The red labels for the red points in Figure 3 are small LaTeX fragments that each describe a simple LaTeX mathematical expression. The data set used for the red points and labels is stored in a data frame called means and the LaTeX fragments are in a column called label, as shown below.

```
means$label
```

```
#> [1] "$\\bar x_1$" "$\\bar x_2$" "$\\bar x_3$" "$\\bar x_4$" "$\\bar x_5$"
```

The following code draws the plot in Figure 3. A call to ggplot2::geom_point() draws the red points and a call to geom_latex() draws the red labels. The ggGeom object in the code below describes the main plot, which consists of gray dots, horizontal and vertical lines, and y-axis labels. The code for generating ggGeom is not shown in order to keep the code below readable, but it is available in the supplementary materials for this article.

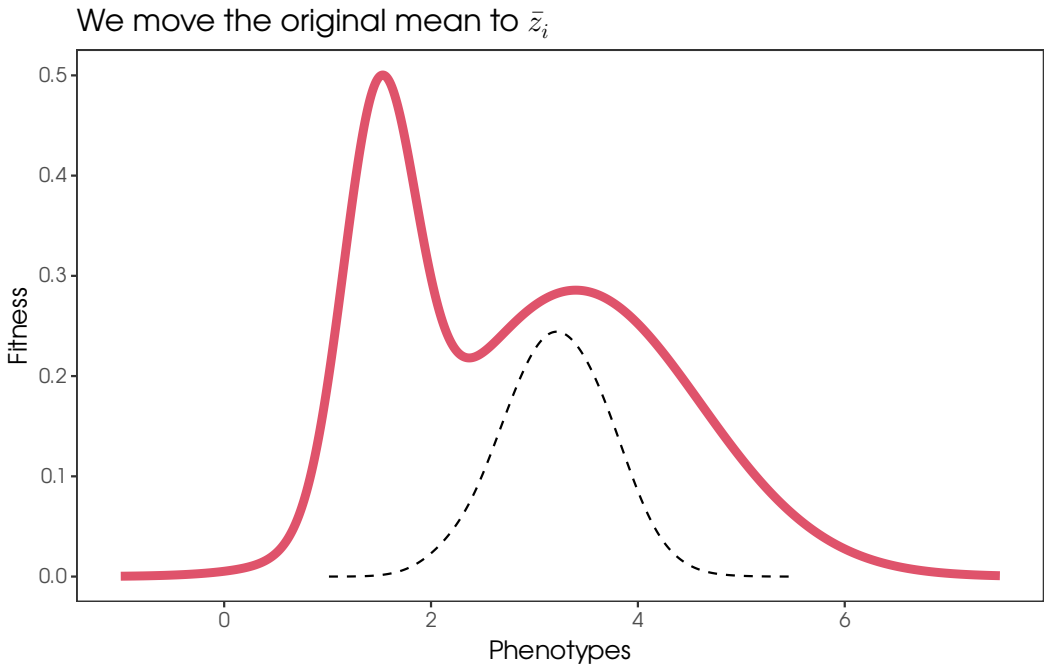


Figure 2: The `ggplot2` plot from Figure 1, without the text annotation, but with a title that was specified using a LaTeX fragment and the function `element_latex()`.

```
ggGeom +
    geom_point(aes(x, sample), data=means, colour=2, size=4) +
    geom_latex(aes(x, sample, label=label), data=means,
               size=6, vjust=-.4, colour=2)
```

3 Under the hood

The previous section showed that simple usage of the `xdvir` package only requires specifying a LaTeX fragment as the text to draw. For example, several examples used the LaTeX fragment shown below.

```
simpleTeX
```

```
#> [1] "We move the original mean to $\\bar z_i$"
```

The `grid.latex()` function has three tasks to perform in order to draw that LaTeX fragment in R:

Authoring: The LaTeX fragment has to be turned into a complete LaTeX document.

The `author()` function in the `xdvir` package allows us to perform this step separately. For example, the following code takes the LaTeX fragment `simpleTeX` and produces a complete LaTeX document, `simpleDoc`, that is ready to typeset.

```
simpleDoc <- author(simpleTeX)
```

```
simpleDoc
```

```
#> %% R package xdvir_0.1.3; engine name: XeTeX; engine version: XeTeX 3.141592653-2.6-
#> \documentclass[varwidth]{standalone}
#> \usepackage{unicode-math}
```

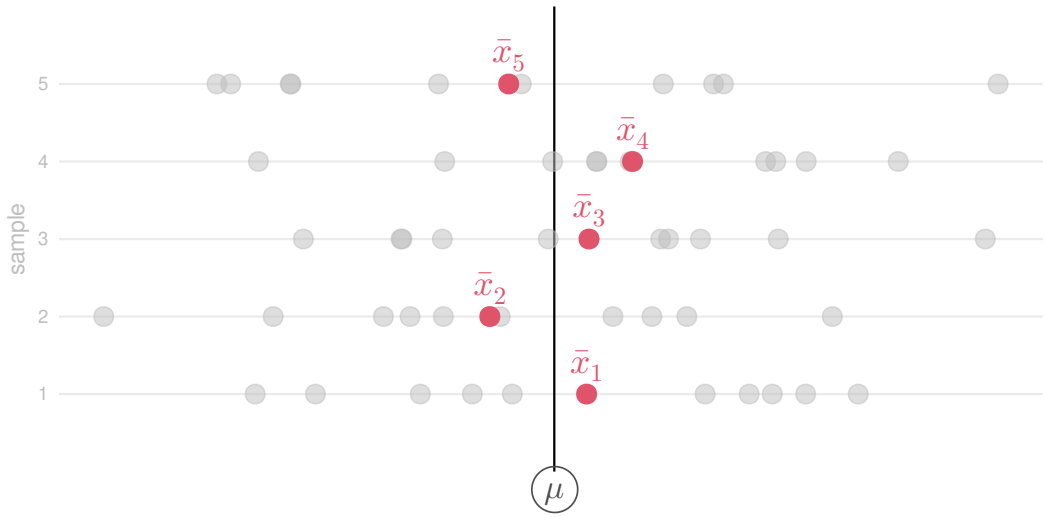


Figure 3: A `ggplot2` plot with text labels specified as LaTeX fragments and drawn using the `geom_latex()` function.

```
#> \begin{document}
#> We move the original mean to $\bar z_i$
#> \end{document}
```

Typesetting: The LaTeX document has to be typeset to produce a set of glyphs and their positions.

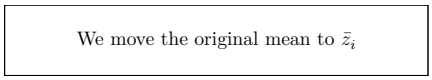
The `typeset()` function in the `xdvir` package allows us to perform this step separately. For example, the following code takes the LaTeX document `simpleDoc` and produces a `"DVI"` object, `simpleDVI`, that contains instructions specifying the fonts to use (lines that contain `x_fnt_def` and `fnt_num` in the output below), the glyphs to use from those fonts (lines that contain `x_glyph` in the output below), and where to draw those glyphs (lines that contain `down` and `right` and `x_glyph`). The output shown below has been trimmed to save space and to make it easier to read.

```
simpleDVI <- typeset(simpleDoc)
simpleDVI

#> pre           version=7, num=25400000, den=473628672, mag=1000,
#>           comment=R package xdvir_0.1.3; engine name: XeTeX; engine version: XeTe
#> bop           counters=1 0 0 0 0 0 0 0 0 0, p=-1
#> xxx1          k=47
#>               x=pdf:pagesize width 143.26802pt height 9.48027pt
#> down3         a=-4114988
#>
#> ...
#>
#> push
#> x_fnt_def     fontnum=40, ptsize=655360
#>           fontname=/home/mitchell/.TinyTeX/texmf-dist/fonts/opentype/public/lm/lm
#> fnt_num_40
#> x_glyph       id=113, x=0, y=0
#> x_glyph       id=50, x=619315, y=0
#> w3            b=218235
#> x_glyph       id=75, x=0, y=0
#>
#> ...
```

Rendering: The result of the typesetting step has to be drawn in R.

The `render()` function in the `xdvir` package allows us to perform this step separately. For example, the code below renders the typesetting information from the `simpleDVI` object in R. The resulting image is shown below the code.

```
render(simpleDVI)
```



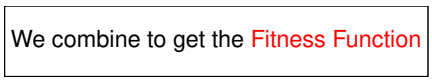
One detail about the output above is that the rendered text from this `render()` call is smaller and in a different font compared to the example from the previous section, which was produced by a `grid.latex()` call. This reflects the fact that `grid.latex()`, in the authoring step, respects the font family and font size that are in effect when the text is rendered. By contrast, the `render()` call is drawing typeset information from a LaTeX document that just makes use of the default LaTeX font, Computer Modern (or to be more precise, a modernized version called Latin Modern) at 10pt.

4 LaTeX packages

The code examples so far have dealt with relatively simple fragments of LaTeX code that consist of just text plus some simple mathematical expressions. While this is already useful, it barely scratches the surface of what is possible with LaTeX code.

Many additional effects can be obtained with LaTeX code by loading LaTeX packages. As a simple example, changing the color of text requires loading the LaTeX package `xcolor`. These LaTeX packages can be loaded using the `packages` argument of the `grid.latex()` function (or the `element_latex()` function or the `geom_latex()` function). For example, the following code draws text with the last two words in red.

```
colourTeX <- r"(We combine to get the \textcolor{red}{Fitness Function})"

grid.latex(colourTeX, packages="xcolor")
```



The argument `packages="xcolor"` is used in the authoring step to load the package in the LaTeX document preamble. This is demonstrated below with an explicit call to the `author()` function. We can see that `\usepackage{xcolor}` has been added to the LaTeX document.

```
colourDoc <- author(colourTeX, packages="xcolor")

colourDoc
```

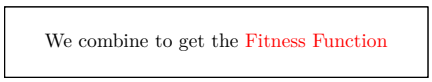
```
#> %% R package xdvir_0.1.3; engine name: XeTeX; engine version: XeTeX 3.141592653-2.6-
#> \documentclass[varwidth]{standalone}
#> \usepackage{unicode-math}
#> \usepackage{xcolor}
#> \begin{document}
#> We combine to get the \textcolor{red}{Fitness Function}
#> \end{document}
```

This in turn affects the typesetting step: without the `xcolor` package, the LaTeX command `\textcolor` would not be recognized; with the `xcolor` package, the `\textcolor` command produces instructions to change color in the `"DVI"` output. This is demonstrated below with an explicit call to the `typeset()` function. An example of the color-change instructions is the line containing `color push` in the output below the code.

```
colourDVI <- typeset(colourDoc)
colourDVI

#> pre          version=7, num=25400000, den=473628672, mag=1000,
#>          comment=R package xdvir_0.1.3; engine name: XeTeX; engine version: XeTe
#> bop          counters=1 0 0 0 0 0 0 0 0 0, p=-1
#>
#> ...
#>
#> x_fnt_def    fontnum=40, ptsize=655360
#>          fontname=/home/mitchell/.TinyTeX/texmf-dist/fonts/opentype/public/lm/lm
#> fnt_num_40
#> x_glyph      id=113, x=0, y=0
#> x_glyph      id=50, x=619315, y=0
#>
#> ...
#>
#> xxx1         k=20
#>              x=color push rgb 1 0 0
#> x_glyph      id=54, x=0, y=0
#> x_glyph      id=66, x=427950, y=0
#>
#> ...
```

The argument `packages="xcolor"` is also used in the rendering step because, without it, the rendering would not take any notice of the instructions to change color. This is demonstrated below with an explicit call to the `render()` function. The resulting image differs from the previous one because it uses the default LaTeX font, but we can see the same change in color for the last two words.

```
render(colourDVI, packages="xcolor")
```

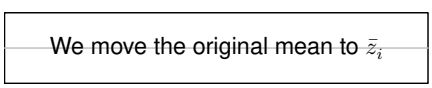


There are several LaTeX packages with predefined support in the `xdvir` package, including `xcolor` for changes in color and `fontspec` for changes in font. Support can be added for other LaTeX packages with the `LaTeXpackage()` function. We will see other predefined packages and an example of defining a new LaTeX package in later sections.

5 Justifying text

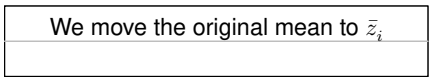
By default, the LaTeX text drawn by `grid.latex()` is centered upon a specified location. For example, the following code draws the `simpleTeX` fragment vertically centered at a location half-way up the image (as indicated by the gray line).

```
grid.latex(simpleTeX, y=.5)
```



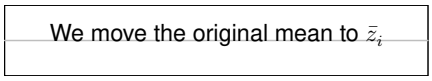
We can specify a different justification using the `vjust` argument. For example, the following code draws the same `simpleTeX` fragment at the same location, but with a bottom-justification. Notice that the bottom of the text is based on the bounding box of the text, so the bottom of the text is the bottom of the subscript "i".

```
grid.latex(simpleTeX, y=.5, vjust="bottom")
```



In some situations it will be much more useful to justify text relative to the text baseline, as shown by the following code.

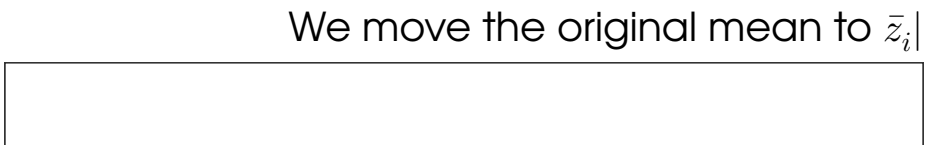
```
grid.latex(simpleTeX, y=.5, vjust="baseline")
```



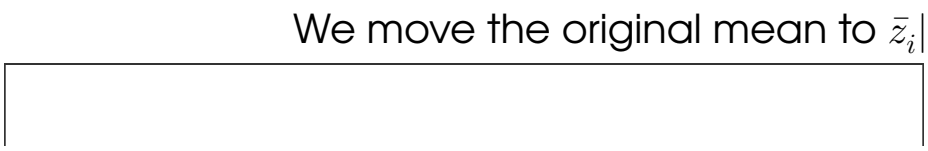
The `xdvir` package has a very simple algorithm for determining the text baseline, but there is also predefined support for the LaTeX package `preview`, which produces a more reliable baseline. That baseline can be accessed, assuming the `preview` package is loaded, by specifying `vjust="preview-baseline"`.

There is also an `hjust` argument for horizontal justification. This accepts the standard values, `"left"`, `"centre"`, and `"right"`, but also accepts `"bbleft"`, `"bbcentre"`, and `"bbright"`. The latter three are based on a bounding box around the actual ink that is drawn, which does not include space before or after glyphs (left-side bearing and right-side bearing). The following code provides a demonstration of the difference by drawing the simple LaTeX fragment from previous examples as the title of a ggplot2 plot. We add a (mathematical) vertical bar to the end of the LaTeX fragment and draw the title larger than normal and justify the text against the right side of the plot region, using `"right"` justification first and then using `"bbright"` justification. The output below the code just shows the very top of the plot in order to save space.

```
rightBearingTeX <- paste0(simpleTeX, "$|$")

ggIntro +
    labs(title=rightBearingTeX) +
    theme(plot.title=element_latex(size=20, hjust="right"))
```




```
ggIntro +
    labs(title=rightBearingTeX) +
    theme(plot.title=element_latex(size=20, hjust="bbright"))
```



The difference between the two plots is that the second vertical bar is precisely aligned with the right edge of the plot region whereas the first vertical bar is slightly to the left of the right edge of the plot region (because of the right-side bearing of the vertical bar glyph). This is a very small detail, but it is something that can be visually jarring if we are trying to align components of a plot in order to produce a clean design. This fine level of control is exactly the sort of precision that we are seeking by working with LaTeX typesetting.

6 Integrating text

Justifying LaTeX text is a simple example of a larger problem: *integrating* LaTeX text. For example, the text annotation in Figure 1 is integrated with the plot in the sense that it is

positioned relative to the plot region. In fact, closer inspection reveals that the text annotation is carefully top-justified with the maximum y-value of the red line and right-justified with the maximum x-value of the red line.

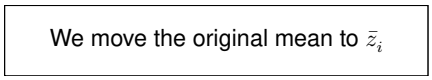
Put in terms of *integration* rather than justification, the text annotation in Figure 1 is integrated with the plot because the LaTeX text is drawn at a location that is coordinated with the location of other R graphics drawing in the plot.

Another example of integration, that reverses the roles, is coordinating other R graphics drawing with the location of LaTeX text. The following code provides a simple example. The LaTeX fragment is the simple one from previous examples with two additions: there are `\zsavepos` commands to mark specific locations within the text and associate them with labels (`"a"` and `"b"`); and there are `\Rzmark` commands to export those locations for R to see.

```
zrefTeX <- r"(We move the original\zsavepos{a} mean to \zsavepos{b}$\bar z_i$
\Rzmark{a}\Rzmark{b})"
```

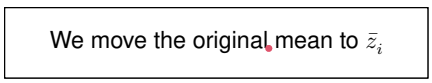
If we render this LaTeX fragment, we just get the familiar output. The commands that we added to the LaTeX fragment are based on the LaTeX package `zref`, so we must load that package.

```
grid.latex(zrefTeX, packages="zref")
```



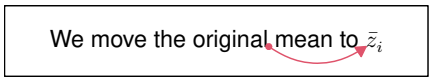
However, we can now access the special locations in the LaTeX output using the `getMark()` function from the `xdvir` package. For example, the following code accesses location `"a"`, which is just after the word "original", and draws a small red dot at that location.

```
a <- getMark("a")
grid.circle(a$devx, a$devy, r=unit(.5, "mm"), gp=gpar(col=2, fill=2))
```



The following code accesses location `"b"`, which is just before the letter "z", and draws a curved arrow from `"a"` to `"b"`.

```
b <- getMark("b")
grid.xspline(unit.c(a$devx, .5*(a$devx + b$devx), b$devx),
             unit.c(a$devy, a$devy - unit(3, "mm"), a$devy),
             shape=-1, gp=gpar(col=2, fill=2),
             arrow=arrow(length=unit(2, "mm"), type="closed"))
```



The exported locations also produce "anchors" that we can use to justify LaTeX text. For example, the following code draws the simple LaTeX fragment with position `"a"` at the center of the image (which is indicated by gray lines).

```
grid.latex(zrefTeX, packages="zref", hjust="a", vjust="a")
```

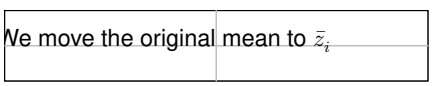


Figure 4 provides a more realistic demonstration. This figure shows the plot from Figure 1 with a line added to visually connect the thick red line with the red part of the LaTeX annotation. The code for this plot is not shown for reasons of space, but it makes use of the same basic idea as the code above by saving locations within the LaTeX output and then accessing them with the `getMark()` function. The full code is available in the supplementary materials for this article.

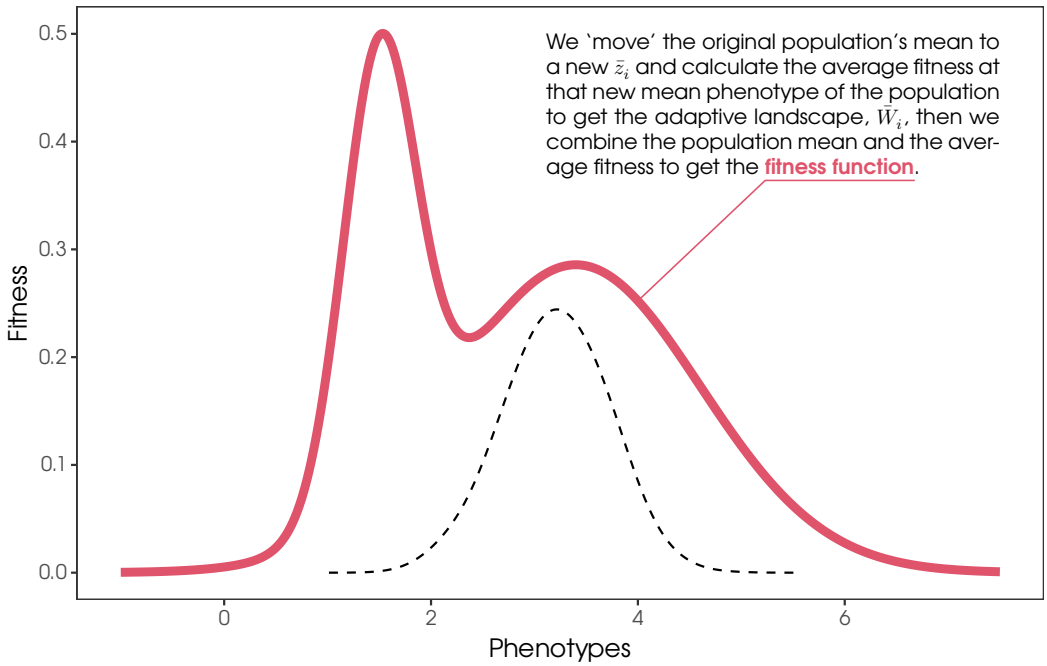


Figure 4: The `ggplot2` plot from Figure 1, including the LaTeX annotation, with a line added relative to marked locations within the LaTeX annotation (and relative to the thick red line).

7 LaTeX graphics

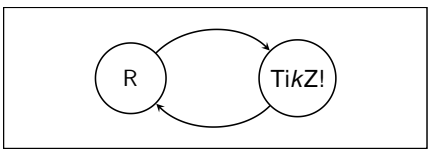
The examples so far have demonstrated using LaTeX code to describe text labels, combined with using R to draw general graphics—lines and circles and so on. It is also possible to use LaTeX to draw general graphics. In particular, the LaTeX package Ti*k*Z provides very powerful and flexible graphics facilities. The `xdvir` package provides support for the LaTeX package Ti*k*Z, so we are able to render Ti*k*Z graphics in R.

For example, the following LaTeX code describes a Ti*k*Z picture consisting of two text labels enclosed within circles, with arrows connecting the circles.

```
tikzTeX <- r"(%
\path (0, 0) node[circle,minimum size=.5in,draw,thick] (x) {\sffamily{R}}
      (3, 0) node[circle,minimum size=.5in,draw,thick] (y) {Ti\textit{k}Z!};
\draw[-{stealth},thick] (x) .. controls (1, 1) and (2, 1).. (y);
\draw[-{stealth},thick] (y) .. controls (2, -1) and (1, -1) .. (x);)"
```

The following code draws this Ti*k*Z picture in R. The argument `packages="tikzPicture"` is necessary to ensure that the Ti*k*Z package is loaded in the authoring step, that Ti*k*Z output is produced in the typesetting step, and that R takes notice of the Ti*k*Z output in the rendering step.

```
grid.latex(tikzTeX, packages="tikzPicture")
```

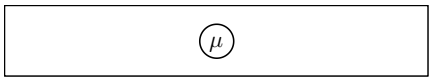


The label on the x-axis of Figure 3 is another simple Ti*k*Z picture that uses Ti*k*Z commands to draw the Greek letter mu within a circle. This example is not completely trivial because it uses the LaTeX concept of "phantom" text to make the circle large enough to fit a capital "M" even though no such character is drawn. This is another example of the detailed typesetting capabilities that access to LaTeX provides.

```
muDot <- r"(%
\begin{tikzpicture}
\node[draw,circle,thick,inner sep=0.5mm]{\vphantom{M}$\mu$};
\end{tikzpicture})"
```

The LaTeX code this time includes an explicit `\begin{tikzpicture}` and `\end{tikzpicture}`. Those commands were implicitly added in the previous example because we specified `packages="tikzPicture"`. This time, we have explicitly provided the commands, so we just specify `packages="tikz"`.

```
grid.latex(muDot, packages="tikz")
```



We will see a more complex example of Ti*k*Z output in a later section. Figure 5 is also a Ti*k*Z picture that has been rendered in R.

8 Programmatic generation of LaTeX

One obstacle to adopting the `xdvir` package is that it assumes that the user knows how to create a LaTeX fragment (i.e., write LaTeX code). While the `xdvir` package provides some assistance so that the user is only required to write a LaTeX fragment rather than a complete LaTeX document, LaTeX fragments for text labels tend to be more complex than plain text labels, thanks to the additional markup that is required.

However, LaTeX code is still just text. This means that all of the text-generating tools in R are available to help with authoring LaTeX fragments. For example, the labels used to render text data symbols in Figure 3 could be generated via a simple call to the `paste0()` function, as shown below.

```
paste0("$\\bar x_", 1:5, "$")
```

```
#> [1] "$\\bar x_1$" "$\\bar x_2$" "$\\bar x_3$" "$\\bar x_4$" "$\\bar x_5$"
```

There are also packages that can generate larger fragments of LaTeX code. For example, there are packages like **xtable** (Dahl et al., 2019) and **latexpdf** (Bergsma, 2023) for generating LaTeX tables and the **rmarkdown** package (Xie et al., 2018) can generate LaTeX documents from Markdown input. The Literate Programming section of the Reproducible Research CRAN Task View provides a more comprehensive list of relevant R packages. The **texPreview** package (Sidi and Polhamus, 2024) may also be helpful for previewing the output of LaTeX code within an R session.

Some of these tools can be particularly useful for generating larger chunks of LaTeX code, although the LaTeX code that is produced may consist of entire documents rather than just LaTeX fragments. The next section describes how we can cope with that situation.

9 Customization and debugging

Most of the examples in this article take a fragment of LaTeX code and pass it to the `grid.latex()` function, which performs an authoring step, a typesetting step, and a rendering step. We saw in a previous section that there are functions `author()`, `typeset()`, and `render()` that allow us to perform these steps separately (see Figure 5). This provides more control over the individual steps and allows us to inspect the results of the individual steps, which can be useful for debugging. In this section, we explore further options for controlling the authoring, typesetting, and rendering steps.

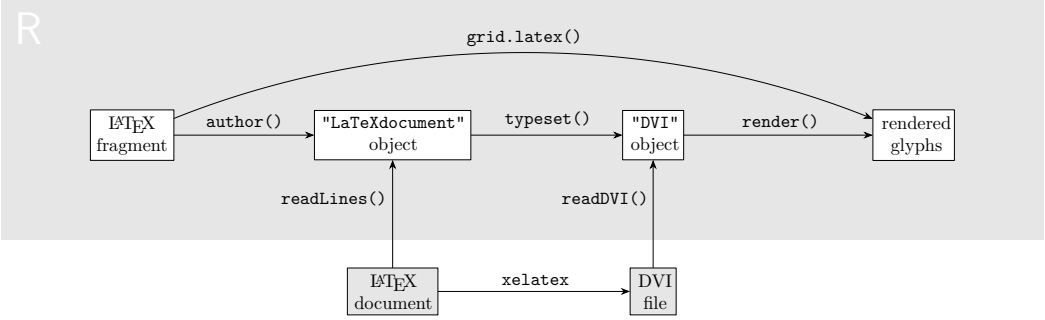


Figure 5: The design of the 'xdvir' package.

The `author()` function transforms a LaTeX fragment into a complete LaTeX document. Although there are arguments to the `author()` function that allow some control over that transformation, e.g., the `packages` argument, it does not allow full control over the composition of the LaTeX document. Fortunately, a LaTeX document within R is essentially just a character vector, so another way to author a LaTeX document is to create an external text file and read that into R. This allows complete control over the content of the LaTeX document. Another possibility is that we want to use a LaTeX document that we did not create, for example, if we write Markdown code and convert it to LaTeX code.

The `typeset()` function transforms a LaTeX document into a `"DVI"` object that contains a set of typeset glyphs. There is limited control over this process as well, with only the `engine` argument allowing us to select between `"xetex"` or `"luatex"`. Again, one way to obtain greater control is to perform this step outside of R by running a TeX engine, e.g., `xelatex`, on an external text file to produce a DVI file. The `xdvir` package provides the `readDVI()` function to read external DVI files into R and these can then be passed to the `render()` function for drawing.

One important caveat is that both a `"LaTeXdocument"` object that is produced by the `author()` function and a `"DVI"` object that is produced by the `typeset()` function contain information about how they were created, for example, the TeX `engine` that was specified and the LaTeX packages that were loaded. The `typeset()` function checks this information and warns if we ask to typeset a `"LaTeXdocument"` that was produced for a different TeX engine. Similarly, the `render()` function, which also has an `engine` argument, checks and warns if we ask to render a `"DVI"` object that was produced using a different TeX engine.

External LaTeX documents and DVI files do not (explicitly) contain this information so it is up to the user to ensure that the TeX engine, and any LaTeX packages, are consistent with the arguments provided to the functions `typeset()` and `render()`. In some situations, even with the appropriate level of care, it will be impossible to avoid warnings.

10 Example 1

This section demonstrates a more complete example of rendering LaTeX text within a plot. The plot, shown in Figure 6, provides a clear example of the more advanced typesetting capabilities of LaTeX; the text annotation in the top-left corner of the plot is not only typeset into two columns, but both columns are fully justified and feature several examples of hyphenation.

This example also demonstrates one way to integrate a `grid.latex()` call with a plot that was drawn using functions from the `graphics` package. We will also see a simple demonstration of the `LaTeXpackage()` function to allow use of a LaTeX package that has no predefined support in `xdvir`.

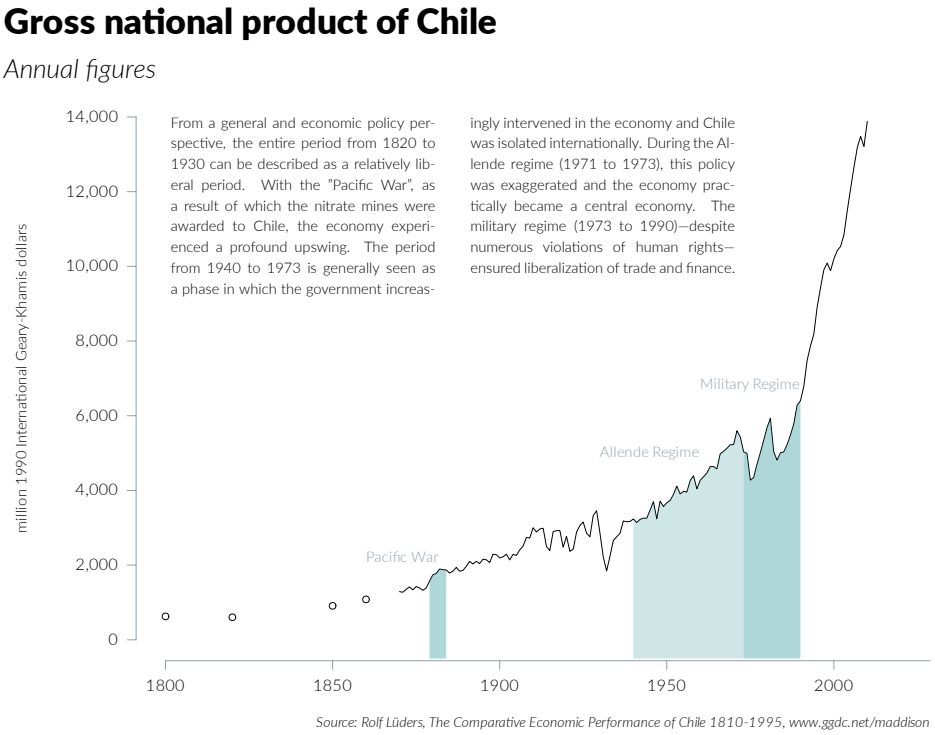


Figure 6: A plot with a two-column text annotation. This plot is an adaptation of Figure 4.1 from Thomas Rahlf's book "Data Visualisation with R" (Rahlf, 2017).

The details of the code that produces the main plot—everything except the two columns of text in the top-left corner—are not relevant to this article so we perform this drawing just with a call to a `rahlfPlot()` function that is defined in the supplementary material for the article. The result is shown in Figure 7.

```
rahlfPlot()
```

Because the main plot is drawn using functions from the `graphics` package, in order to integrate the output from `grid.latex()` with the plot, we need to convert the plot to an equivalent drawing that uses functions from the `grid` package. This can be achieved with the `grid.echo()` function from the **gridGraphics** package (Murrell and Wen, 2020), as shown below.

```
library(gridGraphics)
grid.echo()
```

We want to integrate the LaTeX text with the main plot. In particular, we want the top of the text to be aligned with the value 14,000 on the y-scale of the plot. There is also a 1cm gap between the left of the text and the y-axis line. In order to achieve this, we can navigate to the `grid` viewport that corresponds to the main plot region, which also has scales that match the plot scales. The naming scheme for the `grid` viewports that `grid.echo()` generates is described in Murrell (2015).

```
downViewport("graphics-window-1-1")
```

We are now ready to render the LaTeX text within the plot. The LaTeX code for this example is shown below. This is a larger LaTeX fragment than we have previously seen, but more importantly it contains a larger number of LaTeX commands to control the typesetting of the text. For example, we control the font family with a `\setmainfont` command, we control

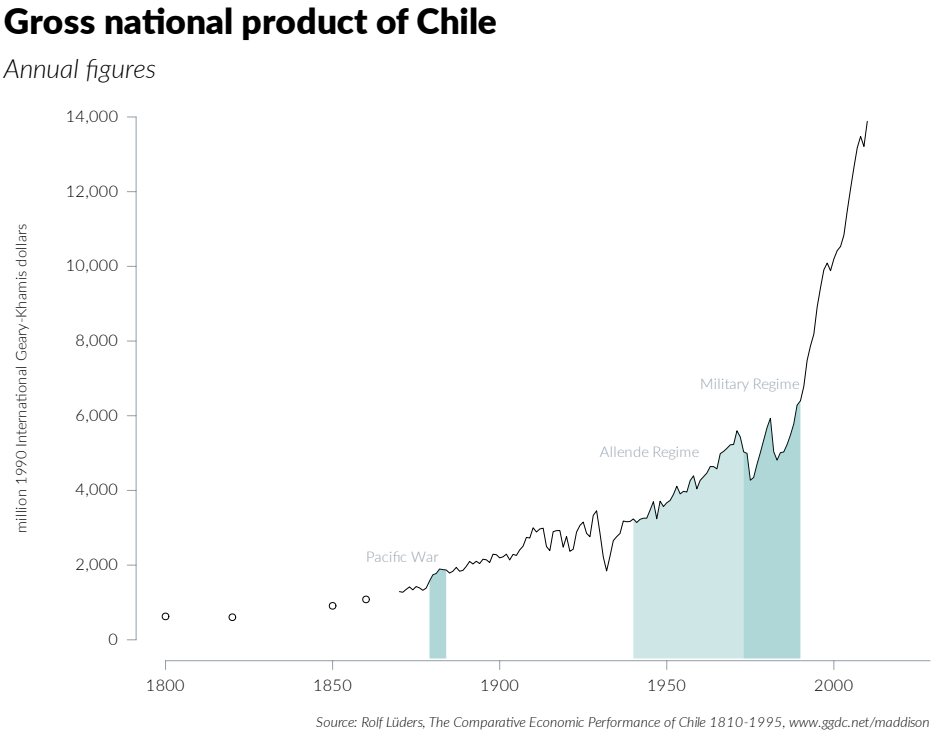


Figure 7: The main plot from Figure 6 without the two columns of text annotation. This plot is drawn using functions from the `graphics` package.

font size and vertical line spacing with a `\fontsize` command, we control the overall width of the text using a `minipage` environment, we set the number of columns using a `multicol` environment, and we control the horizontal spacing between columns with a `\setlength` command.

```
#> \setmainfont{Lato-Light}
#> \fontsize{12pt}{17pt}\selectfont
#> \setlength{\columnsep}{1cm}
#> \begin{minipage}[t]{16.25cm}
#> \begin{multicols}{2}
#> From a general and economic policy perspective, the entire period from
#> 1820 to 1930 can be described as a relatively liberal period. With the
#> "Pacific War", as a result of which the nitrate mines were awarded to
#> Chile, the economy experienced a profound upswing. The period from
#> 1940 to 1973 is generally seen as a phase in which the government
#> increasingly intervened in the economy and Chile was isolated
#> internationally. During the Allende regime (1971 to 1973), this policy
#> was exaggerated and the economy practically became a central
#> economy. The military regime (1973 to 1990)---despite numerous
#> violations of human rights---ensured liberalization of trade and
#> finance.
#> \end{multicols}
#> \end{minipage}
```

The `\setmainfont` and `\fontsize` commands in the LaTeX code require the LaTeX package `fontspec` to be loaded, but this is not a problem because there is predefined support for `fontspec` in the `xdvir` package. However, the `multicol` environment in the LaTeX code requires the LaTeX package `multicol` and there is no predefined support for that in `xdvir`. The following code uses the `LaTeXpackage()` function to provide support for the LaTeX

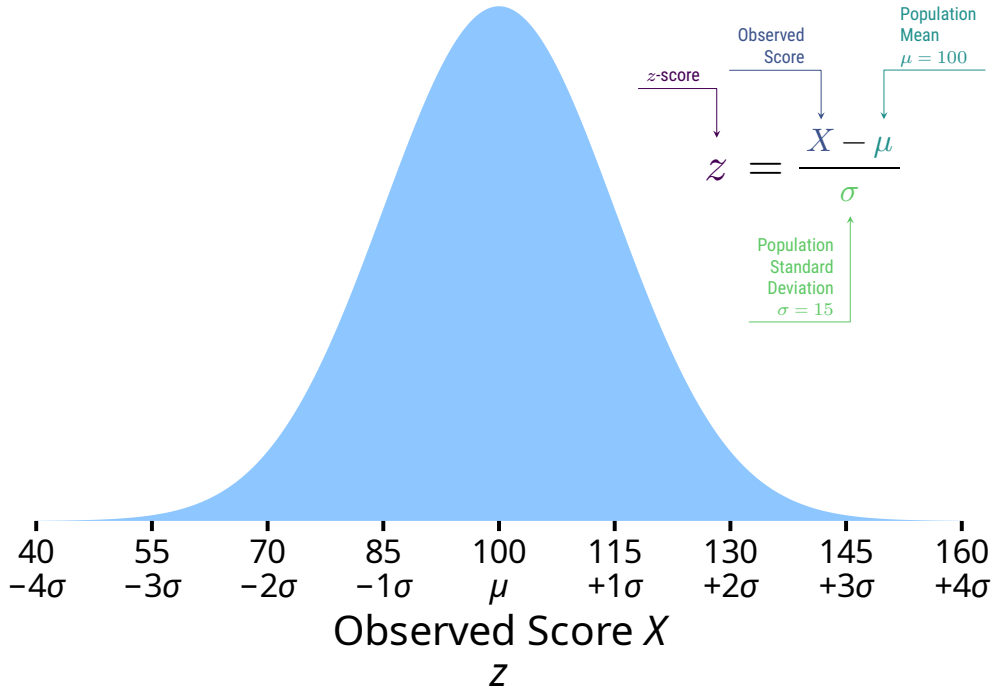


Figure 8: A plot with annotated mathematical expression. This plot is an adaptation of the plot in Schneider.

package `multicol`. In a simple case like this, all we have to do is provide a name for the package (`"multicol"`) and use the `preamble` argument to provide the LaTeX code that should be added in the authoring step to load the LaTeX package. We also call the `registerPackage()` function so that we can refer to this LaTeX package just by its name.

```
multicol <- LaTeXpackage("multicol",
                         preamble="\\usepackage{multicol}")
registerPackage(multicol)
```

Finally, we call `grid.latex()` to add the LaTeX text to the plot. The object `rahlfTeX` contains the LaTeX code, we specify the LaTeX packages that have to be loaded, including the `"multicol"` package that we just registered, and we position the text 1cm in from the left of the the plot viewport and at 14,000 on the y-axis. The final result is shown in Figure 6.

```
grid.latex(rahlfTeX,
           packages=c("fontspec", "multicol"),
           x=unit(1, "cm"), y=unit(14000, "native"),
           hjust="left", vjust="top")
```

11 Example 2

This section looks at another more complete example of a plot with a LaTeX annotation (Figure 8). This example demonstrates the sophisticated effects that are possible by combining TikZ graphics with LaTeX typesetting, in this case to produce an annotated mathematical expression. This example also demonstrates a way to integrate lower-level `grid.latex()` output with a **ggplot2** plot (rather than using `element_latex()` or `geom_latex()`).

The main plot in this example is a **ggplot2** plot. The details of the code that generates the main plot are not particularly relevant to this article, so the main plot is described in the object `ggSchneider`, which is defined in the supplementary materials for the article. One

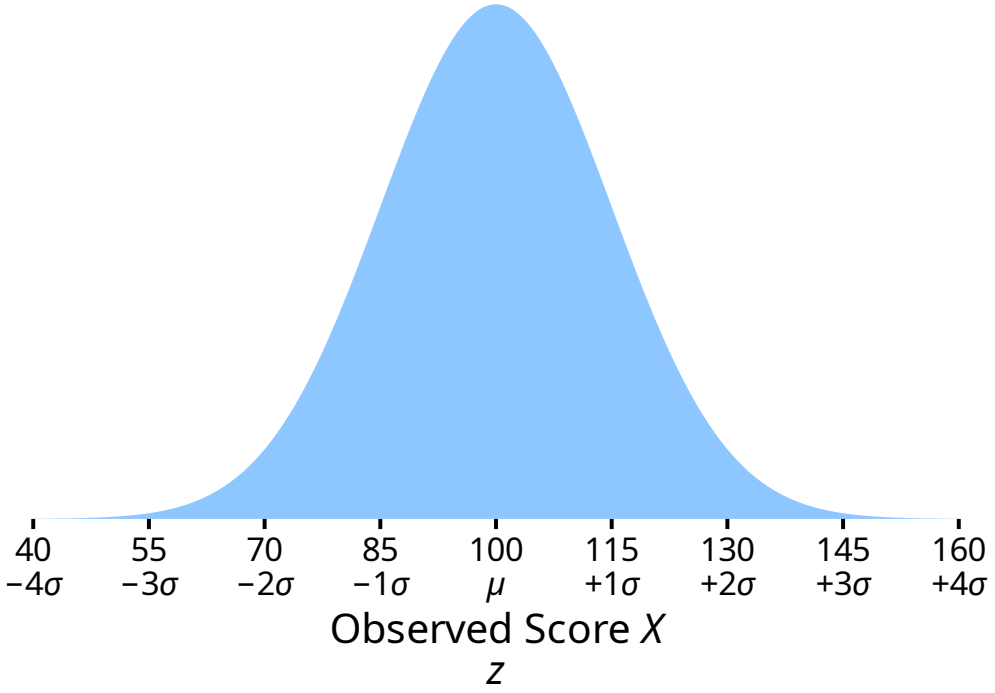


Figure 9: The main plot from Figure 8 without the annotated mathematical expression. This plot is produced using the packages **ggplot2** and **ggtext**.

point worth noting is that the labeling on the x-axis, which combines italic Greek letters with upright digits and signs, is produced using the **ggtext** package. In other words, this example combines two levels of text annotation: labels on the x-axis that are relatively simple, but still beyond the capabilities of core R text drawing; and much more sophisticated text annotations that require access to a complex system like LaTeX. The main plot produced by `ggSchneider` is shown in Figure 9.

```
ggSchneider
```

The start of the LaTeX code for the annotated expression is shown below (the full code is included in the supplementary materials for this article). The LaTeX code is arranged in three blocks: the first block of code defines some colors; the second block describes the main mathematical expression, but includes some `\eqnmark` commands to save locations within the expression; and the third block shows one of the additional mathematical expression annotations, which refers to one of the saved locations within the main mathematical expression, in this case the "z", and positions a label relative to that location, in this case the label "z-score", which is positioned above and to the left of the "z".

```
#> \definecolor{myviolet}{HTML}{440154}
#> \definecolor{myblue}{HTML}{3B528B}
#> \definecolor{myindigo}{HTML}{21908C}
#> \definecolor{mygreen}{HTML}{5DC863}
#>
#> \huge$
#> \eqnmark[myviolet]{z}{z} =
#> \frac{
#>     \eqnmark[myblue]{x}{X}-
#>     \eqnmark[myindigo]{mu}{\mu}}{
#>     \eqnmark[mygreen]{sigma}{\sigma}}
#> $
```

```
#>
#> \annotate[
#>     yshift=1em,
#>     myviolet,
#>     align=right]
#>     {above, left}
#>     {z}
#>     {$z$-score}
#>
```

There are several LaTeX packages required by this LaTeX code, in particular the `\eqnmark` and `\annotate` commands require the LaTeX package `annotate-equations`. As in the previous example, we can add support for this package using the `LaTeXpackage()` and `registerPackage()` functions. One difference this time is that the `annotate-equations` package is being loaded from a local `TeX` directory. The previous example relied on the LaTeX package being available as part of the user's (or the system-wide) TeX installation.

```
annotateEquations <-
    LaTeXpackage(name="annotate",
                 preamble="\\usepackage{TeX/annotate-equations}")
registerPackage(annotateEquations)
```

The LaTeX package `annotate-equations` is built on Ti*k*Z graphics. We do not need to load the LaTeX package `tikz` because `annotate-equations` will do that automatically. However, `xdvir` by default makes use of the bounding box information from Ti*k*Z graphics and, for images with saved locations like this, that bounding box is unreliable. The predefined support for the LaTeX package `tikz` in the `xdvir` package includes a `tikzPackage()` function that allows us to load TikZ, but ignore its bounding boxes, as shown in the following code.

```
tikzNoBBox <-
    tikzPackage(name="tikzNoBBox", bbox=FALSE)
registerPackage(tikzNoBBox)
```

Finally, we will use the LaTeX package `roboto` to access specific variations of the Roboto font for the text labels in the annotated mathematical expressions.

```
roboto <-
    LaTeXpackage(name="roboto",
                 preamble="\\usepackage[sfdefault,condensed]{roboto}")
registerPackage(roboto)
```

Rendering the annotated mathematical expression on the plot requires integrating the LaTeX output with the **ggplot2** plot. In particular, we want to align the top of the LaTeX output with the top of the density curve and we want to align the right side of the LaTeX output with the right edge of the label "160" on the x-axis.

We saw in an earlier section how to use `element_latex()` to draw LaTeX text in labels such as the plot title on a **ggplot2** plot and how to use `geom_latex()` to draw LaTeX text as data symbols. In this example, we are adding a single LaTeX annotation at a specific position within a **ggplot2** plot, so we use the **gggrid** package (Murrell, 2022). This package provides the `grid_panel()` function, which we can add to a **ggplot2** plot, much like the standard `ggplot2::geom_point()` function, to add `grid` drawing to a `ggplot2` plot. The first argument to `grid_panel()` is a function that must generate a `grid` grob for **ggplot2** to draw, based on the data values that **ggplot2** passes to it. In this case, we define a function called `annotation()`, which calls the `xdvir` function `latexGrob()`. The `latexGrob()` function is similar to `grid.latex()` except that it creates a description of something to draw rather than immediately drawing it. We pass to `latexGrob()` the LaTeX code to draw the annotated mathematical expression (`schneiderTeX`), a set of `packages` to load, and arguments that position the output relative to the plot. The final result is shown in Figure 8.

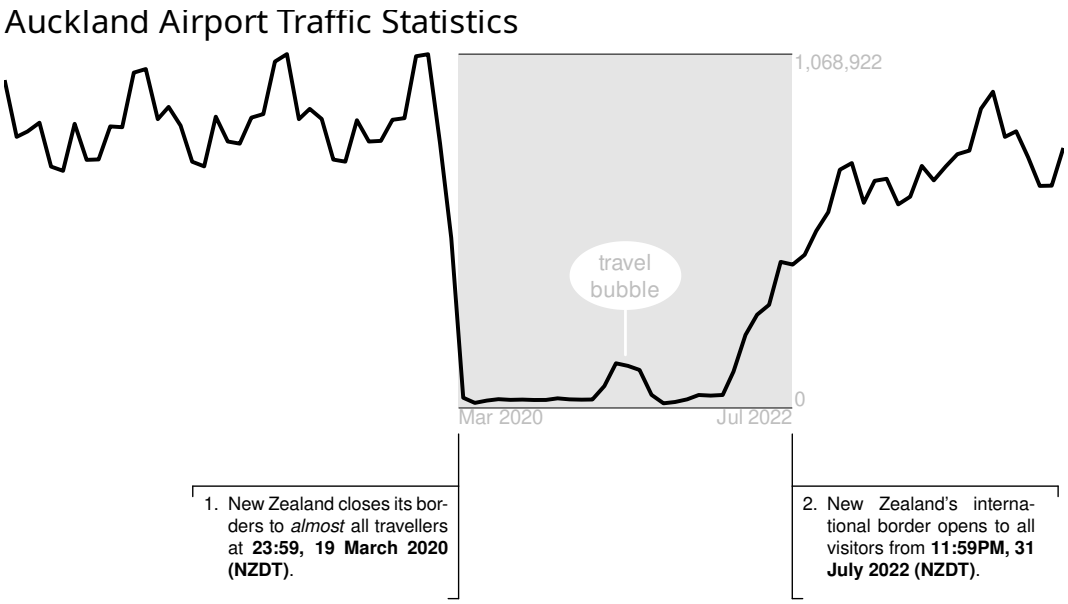


Figure 10: A plot with numbered list items as annotations below the plot. This plot is an adaptation of Figure 10 from Murrell (in press).

```
library(gggrid)

annotation <- function(data, coords) {
    latexGrob(schneiderTeX,
              packages=c("tikzNoBBox", "annotate", "roboto", "xcolor"),
              x=unit(coords$x, "npc") + 0.5*stringWidth("160"),
              y=coords$y, hjust=1, vjust=1)
}

ggSchneider +
    grid_panel(annotation,
               aes(x=x, y=y),
               data=data.frame(x=160, y=dnorm(100, mean=100, sd=15)))
```

12 Example 3

This section provides another demonstration of the range of possibilities that is provided by LaTeX typesetting. This time we add annotations that are formatted as numbered list items below a plot (Figure 10).

The main plot is a **ggplot2** plot with a number of relatively simple annotations already added. The details of the code are not particularly relevant to this article, so the main plot is described in the object `ggANZJS`, which is defined in the supplementary materials for the article. One point worth noting is that the LaTeX annotations that we will be adding are required to fit within the lines that extend below the plot. In other words, we will be specifying a fixed width for the LaTeX output to fit into. The main plot produced by `ggANZJS` is shown in Figure 11.

```
ggANZJS
```

We will focus on drawing just the left-hand LaTeX annotation. The LaTeX code is shown below. This includes commands to control the font size and an `enumerate` environment that creates a numbered list item.

```
#> %
```

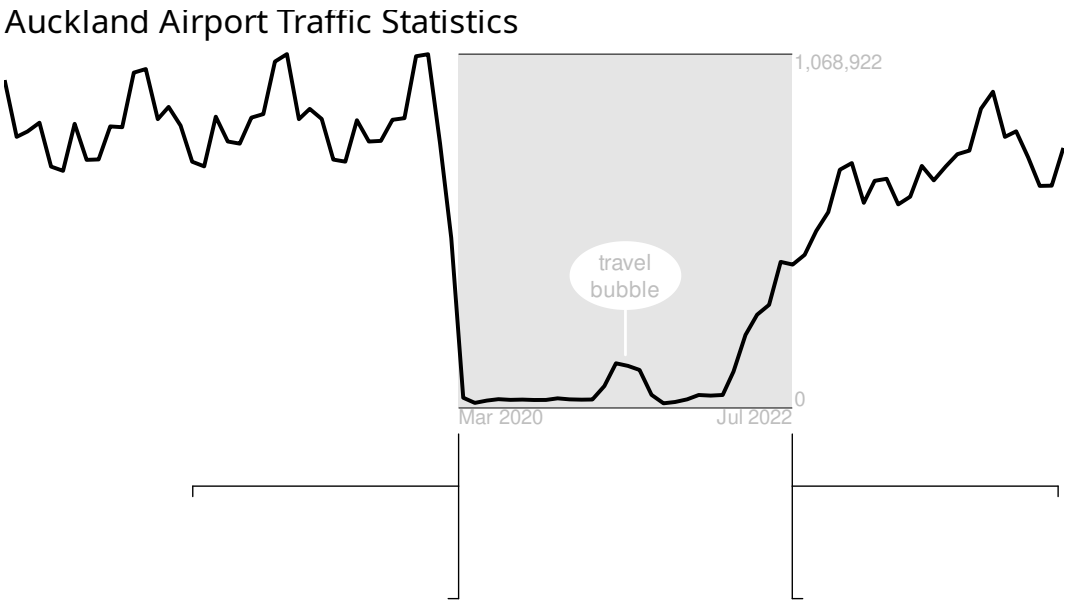


Figure 11: The main plot from Figure 10 without the numbered list items as annotations. This plot is produced using the **ggplot2** package.

```
#> \fontsize{10}{12}
#> \selectfont
#> \begin{enumerate}
#> \item New Zealand closes its borders to \textit{almost} all travellers at
#> \textbf{23:59, 19 March 2020 (NZDT)}.
#> \end{enumerate}
```

As with the previous example, we have a single annotation that we want to position quite carefully, so we define a function that generates a `grid` grob to use with the `grid_panel()` function from the **gggrid** package. The `labelLeft()` function calls `latexGrob()`, gives it the LaTeX code to draw (`closeTeX`), specifies the position for the LaTeX output, and specifies a `width` for the output to be typeset within.

```
labelLeft <- function(data, coords) {
    x1 <- coords$x[1]
    x2 <- coords$x[2]
    w <- unit(1 - x2, "npc") - unit(1, "mm")
    gap <- 15
    latex1 <- latexGrob(closeTeX,
                        x=unit(x1, "npc") - unit(2, "mm"),
                        y=unit(0, "npc") - unit(gap, "mm") - unit(2, "mm"),
                        hjust=1, vjust=1,
                        width=w)
}
```

The following code combines the left-hand label annotation, and a very similar right-hand label annotation, with the `ggANZJS` plot. The final result is shown in Figure 10.

```
ggANZJS +
    grid_panel(labelLeft,
               aes(x=borders),
               data=data.frame(borders=c(borderClosed, borderOpen))) +
    grid_panel(labelRight,
               aes(x=borders),
               data=data.frame(borders=c(borderClosed, borderOpen)))
```

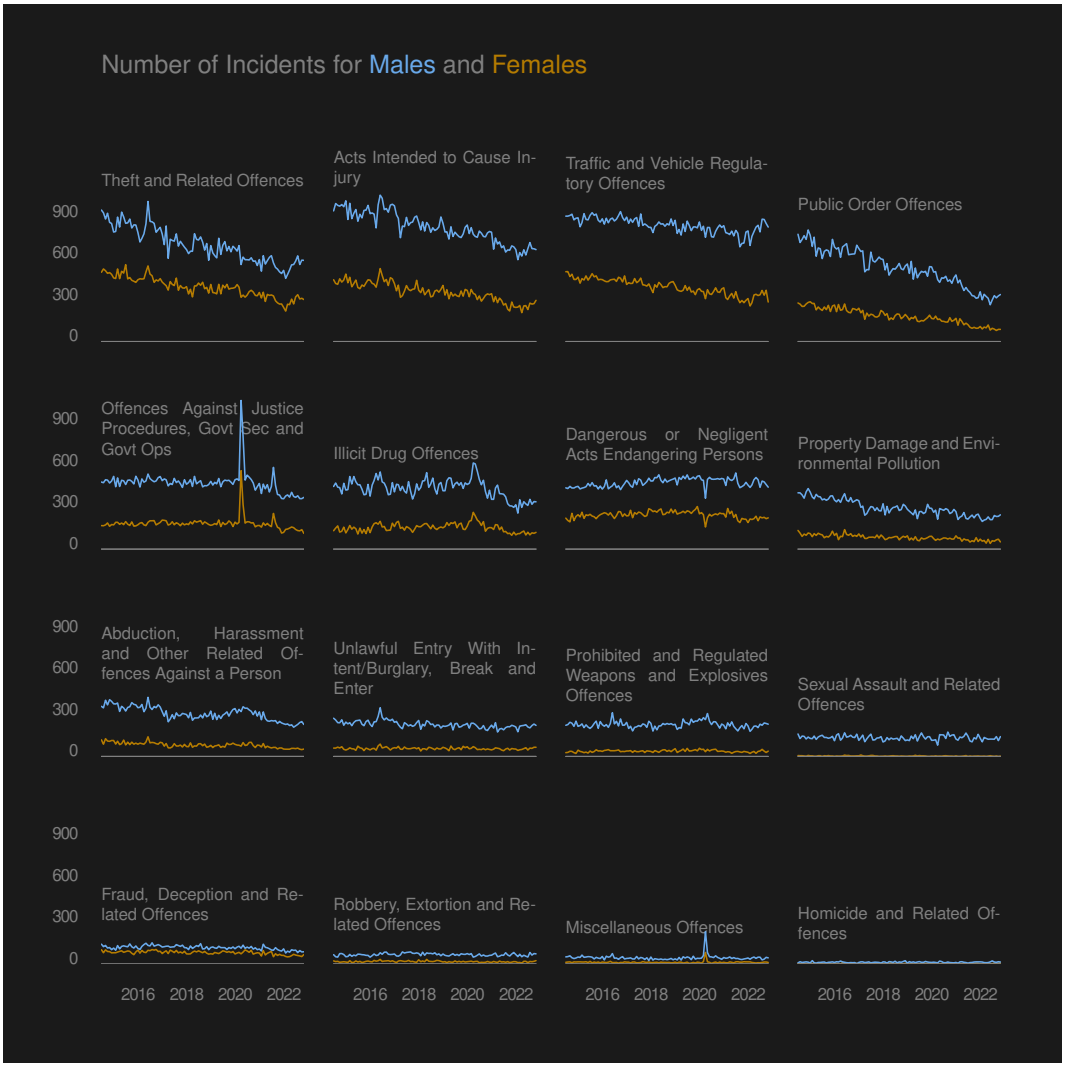


Figure 12: A **lattice** plot with LaTeX text used for the plot title and for annotations in each panel.

13 Example 4

This section provides an example of integrating `grid.latex()` output with a multi-panel **lattice** plot (Sarkar, 2008). The plot is shown in Figure 12.

The main plot is a **lattice** plot consisting of multiple panels, with separate lines for males and females. The details of the code for generating the main plot are not relevant to this article, so it is described in the object `latticeCrime`, which is defined in the supplementary material. The main plot produced by `latticeCrime` is shown in Figure 13.

```
latticeCrime
```

We can add drawing to each panel of a **lattice** plot by providing a *panel function*. The panel function is passed the relevant data for the panel, and the code within the panel function is run in the panel viewport, which means that the appropriate axis scales are available. This means that we can include a call to `grid.latex()` within a panel function in order to add LaTeX text to each panel. For example, the following code defines the panel function for Figure 12. This function calculates the appropriate label for the panel and encloses that within a LaTeX `minipage` environment that is the width of the panel. This means that the label is typeset to be fully-justified within the panel (unless it is a single line that is narrower than the panel). We use a `minipage` environment in the LaTeX fragment rather than just using the `width` argument to `grid.latex()` because `minipage` produces a more precise

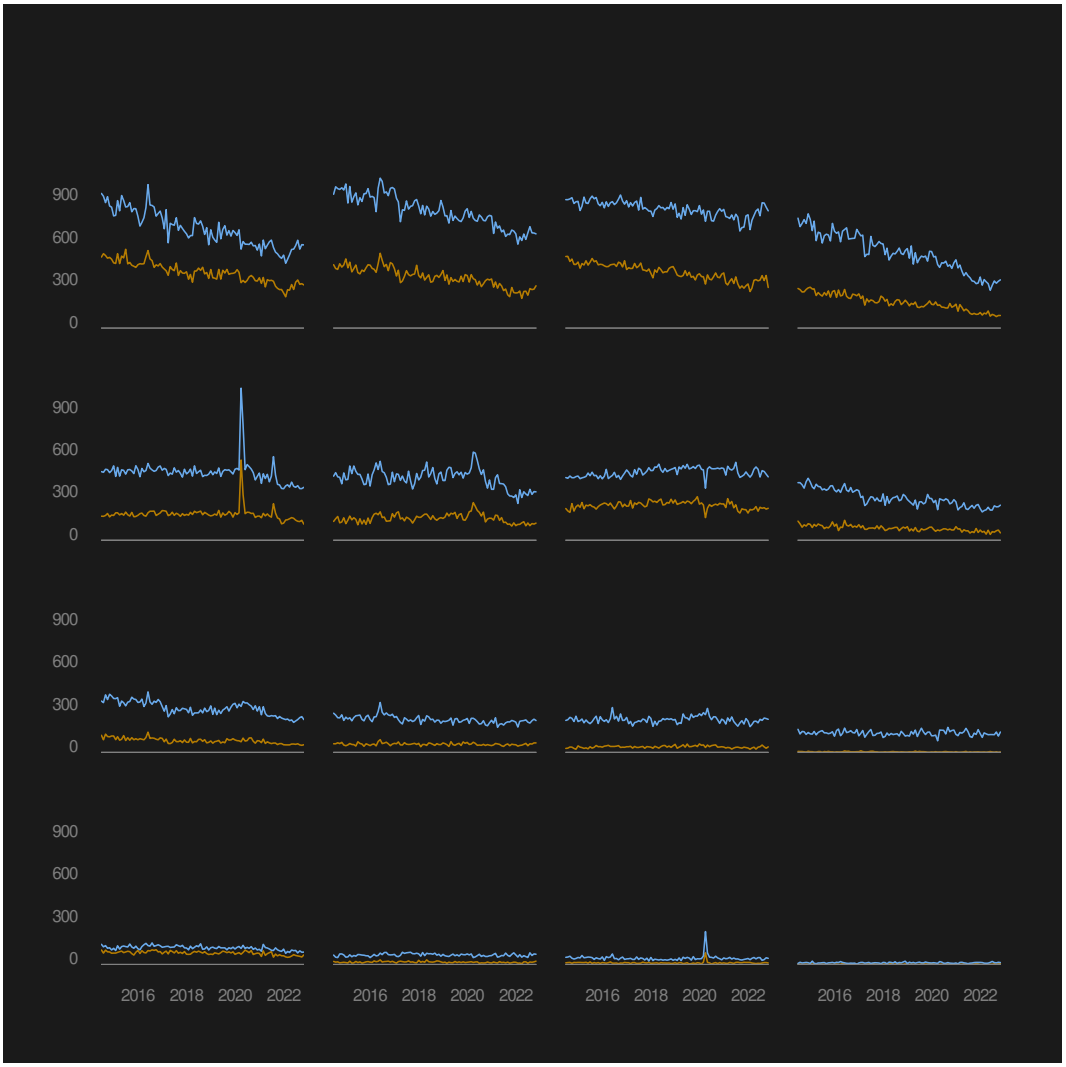


Figure 13: The main plot from Figure 12 without the title and annotations in each panel. This plot is produced using the **lattice** package.

width. The panel function then calls `grid.latex()` to draw that LaTeX fragment, placing the label slightly above the first data value for males. The call to the `mainPanel()` function draws the yellow and blue lines that are part of the main plot.

```
latexPanel <- function(x, y, subscripts, groups, ...) {
    type <- crime$Type[subscripts][1]
    labelY <- y[groups == "Male"][1]
    labelWidth <- convertWidth(unit(1, "npc"), "in", valueOnly=TRUE)
    panelTeX <- paste0("\\begin{minipage}{", labelWidth, "in}",
                       type,
                       "\\end{minipage}")
    grid.latex(panelTeX,
               x=0, hjust="left",
               y=unit(labelY, "native") + unit(4, "mm"), vjust="bottom",
               gp=gpar(col=lightGrey, fontsize=8))
    mainPanel(x, y, subscripts, groups, ...)
}
```

The title of a **lattice** plot can be specified as a `grid` grob. This means that we can call `latexGrob()` to generate a title for the plot in Figure 12. The LaTeX fragment below describes the label, first defining three colors, and then giving the title text, with the words "Male" and "Female" colored differently.

```
titleTeX <- r"(%
\definecolor{lightGrey}{RGB}{128,128,128}
\definecolor{lattice1}{RGB}{105,169,234}
\definecolor{lattice2}{RGB}{181,124,1}
\color{lightGrey}
Number of Incidents for \textcolor{lattice1}{Males} and
\textcolor{lattice2}{Females}
)"
```

The following code calls `latexGrob()` to define the title. We pass the LaTeX fragment `titleTeX`, we position the title to line up with the left edge of the first column of panels, and we load the LaTeX package `xcolor` so that the colors work.

```
latexTitle <- latexGrob(titleTeX, x=titleX, hjust="left",
                        packages="xcolor")
```

The following code creates the final plot by adding the panel function `latexPanel` and the title `latexTitle` to the main plot `latticeCrime`. The final result is shown in Figure 12.

```
update(latticeCrime,
       panel=latexPanel,
       main=latexTitle)
```

14 Discussion

The `xdvir` package provides convenient high-level functions for rendering LaTeX fragments as labels, annotations, or data symbols on R plots. The package also provides lower-level functions that allow more fine control over the authoring, typesetting, and rendering of LaTeX code in R.

The benefit of the `xdvir` package is access to the typesetting capabilities of LaTeX. This ranges from relatively simple features like changes in font family, font weight, and font style, and automatic line breaks, to intermediate features like full justification, hyphenation, and

high-quality mathematical expressions, and more advanced features like enumerated lists, multiple columns, and Ti*k*Z graphics.

One limitation of the `xdvir` package is that rendering LaTeX fragments is noticeably slower than rendering simple character values. This is mainly because the typesetting step requires running a TeX engine to produce a DVI file. The `xdvir` package performs some caching in order to minimize the problem, but the time cost can still be quite large. For example, Figure 12 requires running a TeX engine 17 times.

Another limitation of the `xdvir` package is that it requires a graphics device that can render typeset glyphs. This currently includes the `pdf()` and `quartz()` devices, plus all devices based on the Cairo graphics library (Packard et al., 2025), and graphics devices provided by the `ragg` package (Pedersen and Shemanarev, 2025).

A final major limitation of `xdvir` is that it only currently supports two TeX engines: XeTeX and recent LuaTeX. The function `TeXstatus()` can be used to report on whether these are available. An implicit limitation is that `xdvir` requires a TeX installation, though that is simplified through a dependency on the **tinytex** package (Xie, 2024).

Given these limitations, it is worth discussing alternative approaches. The first section of this article mentioned **gridtext**, **ggtext**, and **marquee**. These packages provide alternative ways to render non-trivial text labels, but do so through Markdown and/or HTML rather than LaTeX. Although they may not be able to produce as wide a range of results compared to LaTeX code, they will perform much faster and require fewer dependencies than `xdvir`. There are also a number of packages that perform specific text-placement tasks, for example **geomtextpath** (Cameron and van den Brand, 2025), which can arrange text along an arbitrary path, and **directlabels** (Hocking, 2025) and **ggforce** (Pedersen, 2025), which provide functions for cleverly positioning text annotations, though without typesetting facilities. The advantage of `xdvir` by comparison with these packages is that it is possible to produce more advanced typesetting results thanks to having access to LaTeX.

The **tikzDevice** package (Sharpsteen and Bracken, 2023) is an interesting alternative because, where `xdvir` integrates LaTeX text with R graphics, **tikzDevice** reverses the process and integrates R graphics with LaTeX. The **tikzDevice** package provides an R graphics device that converts R plots into Ti*k*Z pictures so that R plots can include labels with LaTeX fragments and R plots can be deeply integrated with LaTeX documents. There is also a **ggtikz** package (Thomas, 2024) that builds on **tikzDevice** to allow Ti*k*Z annotations on **ggplot2** plots. The main difference compared to `xdvir` is the destination: if we use `xdvir`, we end up with LaTeX output within an R plot; if we use **tikzDevice** or **ggtikz**, we end up with an R plot within LaTeX output. If the final destination is a LaTeX document, then **tikzDevice** or **ggtikz** may provide more convenience and greater control. However, if the final destination is more general, or unknown, then `xdvir` may be the more appropriate solution.

The **latex2exp** package (Meschiari, 2022) is another package that works in the opposite direction to `xdvir`. This package takes a LaTeX fragment and converts it to an R *plotmath* expression. This allows users familiar with LaTeX to access R's math-drawing facility whereas `xdvir` allows users to access LaTeX's math-drawing facility, which is far superior. The advantage of **latex2exp**, as with many of these alternative approaches, is that it does not have any system dependencies, whereas `xdvir` requires a TeX installation.

Another alternative approach to including LaTeX output in R plots is to import an image of the LaTeX output. This approach harks back to early solutions for including LaTeX mathematical expressions in web pages by generating PNG images from LaTeX fragments. However, more modern technologies, such as SVG, mean that this approach can yield a much higher-quality result, as demonstrated by Schneider. One simple advantage of the `xdvir` approach is the level of convenience that it provides by automating the authoring and typesetting steps. The `xdvir` package also provides more possibilities to integrate LaTeX output with other drawing in R through anchors and saved positions.

Some of the limitations of `xdvir` may also be overcome by further development. For example, it may be possible to extend support to more TeX engines and to more graphics devices. Providing support for more LaTeX packages is another area for future work.

15 Acknowledgments

The `xdvir` package depends on Yihui Xie's **tinytex** package for the typesetting step. This package makes it much simpler to make use of TEX engines, including performing multiple runs when necessary, and much easier to install LATEX packages (and TEX itself).

Claus O. Wilke's **ggtext** package and Thomas Lin Pedersen's **marquee** package provided excellent templates for the integration of improved text-drawing facilities with **ggplot2**.

The author owes a debt of gratitude to Marc-Olivier Beausoleil (Figure 1), Thomas Rahlf (Figure 6), and Joel Schneider (Figure 8) for sharing their work and for giving either implicit or explicit permission to base several of the examples in this article on their work.

Finally, thanks to the journal editor and an anonymous reviewer who provided many helpful criticisms and suggestions that helped to improve both the package and this article.

References

T. Bergsma. *latexpdf: Convert Tables to PDF or PNG*, 2023. URL https://CRAN.R-project.org/package=latexpdf. R package version 0.1.8. [p173]

M. A. Borkin, Z. Bylinskii, N. W. Kim, C. M. Bainbridge, C. S. Yeh, D. Borkin, H. Pfister, and A. Oliva. Beyond memorability: Visualization recognition and recall. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):519–528, 2016. doi: 10.1109/TVCG.2015.2467732. [p162]

A. Cameron and T. van den Brand. *geomtextpath: Curved Text in 'ggplot2'*, 2025. URL https://CRAN.R-project.org/package=geomtextpath. R package version 0.1.5. [p185]

D. B. Dahl, D. Scott, C. Roosen, A. Magnusson, and J. Swinton. *xtable: Export Tables to LaTeX or HTML*, 2019. URL https://CRAN.R-project.org/package=xtable. R package version 1.8-4. [p173]

M. A. Hearst. Show it or tell it? text, visualization, and their combination. *Commun. ACM*, 66(10):68–75, Sept. 2023. ISSN 0001-0782. doi: 10.1145/3593580. URL https://doi.org/10.1145/3593580. [p162]

T. D. Hocking. *directlabels: Direct Labels for Multicolor Plots*, 2025. URL https://CRAN.R-project.org/package=directlabels. R package version 2025.6.24. [p185]

H.-K. Kong, Z. Liu, and K. Karahalios. Frames and slants in titles of visualizations on controversial topics. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–12, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356206. doi: 10.1145/3173574.3174012. URL https://doi.org/10.1145/3173574.3174012. [p162]

S. Meschiari. *latex2exp: Use LaTeX Expressions in Plots*, 2022. URL https://CRAN.R-project.org/package=latex2exp. R package version 0.9.6. [p185]

P. Murrell. The gridgraphics package. *The R Journal*, 7:151–162, 2015. ISSN 2073-4859. URL https://rjournal.github.io/. [p175]

P. Murrell. *gggrid: Draw with 'grid' in 'ggplot2'*, 2022. URL https://CRAN.R-project.org/package=gggrid. R package version 0.2-0. [p179]

P. Murrell. Text in R graphics. *Australian & New Zealand Journal of Statistics*, in press. doi: 10.1111/anzs.12438. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/anzs.12438. [p180]

P. Murrell and R. Ihaka. An approach to providing mathematical annotation in plots. *Journal of Computational and Graphical Statistics*, 9(3):582–599, 2000. doi: 10.1080/10618600.2000.10474900. URL https://www.tandfonline.com/doi/abs/10.1080/10618600.2000.10474900. [p162]

P. Murrell and Z. Wen. *gridGraphics: Redraw Base Graphics Using 'grid' Graphics*, 2020. URL https://CRAN.R-project.org/package=gridGraphics. R package version 0.5-1. [p175]

P. Murrell, T. L. Pedersen, and S. Urbanek. Rendering typeset glyphs in R graphics. Technical Report 2023-01, Department of Statistics, The University of Auckland, 2023. URL https://stattech.blogs.auckland.ac.nz/2023/05/06/2023-01-rendering-typeset-glyphs-in-r-graphics. version 1. [p163]

K. Packard, C. Worth, and B. Esfahbod. Cairo graphics library, 2025. URL https://www.cairographics.org/. Accessed: 2025-01-20. [p185]

T. L. Pedersen. *ggforce: Accelerating 'ggplot2'*, 2025. URL https://CRAN.R-project.org/package=ggforce. R package version 0.5.0. [p185]

T. L. Pedersen and M. Mitáš. *marquee: Markdown Parser and Renderer for R Graphics*, 2025. URL https://CRAN.R-project.org/package=marquee. R package version 1.0.0. [p162]

T. L. Pedersen and M. Shemanarev. *ragg: Graphic Devices Based on AGG*, 2025. URL https://CRAN.R-project.org/package=ragg. R package version 1.4.0. [p185]

T. Rahlf. *Data Visualisation with R: 100 Examples*. Springer Publishing Company, Incorporated, 1st edition, 2017. ISBN 3319497502. URL https://link.springer.com/book/10.1007/978-3-030-28444-2. [p175]

D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer, New York, 2008. ISBN 978-0-387-75968-5. URL http://lmdvr.r-forge.r-project.org. [p182]

W. J. Schneider. Annotated equations in ggplot2. URL https://wjschne.github.io/posts/2023-07-23-latex-equation-in-ggplot2/. [p177, 185]

C. Sharpsteen and C. Bracken. *tikzDevice: R Graphics Output in LaTeX Format*, 2023. URL https://CRAN.R-project.org/package=tikzDevice. R package version 0.12.6. [p185]

J. Sidi and D. Polhamus. *texPreview: Compile and Preview Snippets of 'LaTeX'*, 2024. URL https://CRAN.R-project.org/package=texPreview. R package version 2.1.0. [p173]

O. Thomas. *ggtikz: Post-Process 'ggplot2' Plots with 'TikZ' Code Using Plot Coordinates*, 2024. URL https://CRAN.R-project.org/package=ggtikz. R package version 0.1.3. [p185]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p165]

C. O. Wilke and B. M. Wiernik. *ggtext: Improved Text Rendering Support for 'ggplot2'*, 2022a. URL https://CRAN.R-project.org/package=ggtext. R package version 0.1.2. [p162]

C. O. Wilke and B. M. Wiernik. *gridtext: Improved Text Rendering Support for 'grid' Graphics*, 2022b. URL https://CRAN.R-project.org/package=gridtext. R package version 0.1.5. [p162]

Y. Xie. *tinytex: Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents*, 2024. URL https://github.com/rstudio/tinytex. R package version 0.54. [p185]

Y. Xie, J. Allaire, and G. Grolemund. *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida, 2018. ISBN 9781138359338. URL https://bookdown.org/yihui/rmarkdown. [p173]

Paul Murrell
The University of Auckland
Department of Statistics
Auckland, New Zealand
https://www.stat.auckland.ac.nz/~paul/
ORCiD: 0000-0002-3224-8858
paul@stat.auckland.ac.nz

lqmix: an R Package for Longitudinal Data Analysis via Linear Quantile Mixtures

by Marco Alfò, Maria Francesca Marino, Maria Giovanna Ranalli, Nicola Salvati

Abstract The analysis of longitudinal data gives the chance to observe how units' behavior changes over time, but it also poses a series of issues. These have been the focus of an extensive literature in the context of linear and generalized linear regression, moving also, in the last ten years or so, to the context of linear quantile regression for continuous responses. In this paper, we present `lqmix`, a novel `R` package that assists in estimating a class of linear quantile regression models for longitudinal data, in the presence of time-constant and/or time-varying, unit-specific, random coefficients, with unspecified distribution. Model parameters are estimated in a maximum likelihood framework via an extended EM algorithm, while the corresponding standard errors are derived via a block-bootstrap procedure. The analysis of a benchmark dataset is used to give details on the package functions.

1 Introduction

Quantile regression (Koenker and Bassett, 1978) has become quite a popular technique to model the effect of observed covariates on the conditional quantiles of a continuous response of interest. This represents a well-established practice for the analysis of data when the focus goes beyond the conditional mean. Comprehensive reviews on this topic can be found, among others, in Koenker and Hallock (2001), Yu et al. (2003), Koenker (2005), and Hao and Naiman (2007).

With longitudinal data, dependence between observations recorded from the same statistical unit needs to be taken into account to avoid bias and inefficiency in parameter estimates. Different modeling alternatives are available in the literature for handling such a dependence. For a general presentation, see e.g., Diggle et al. (2002) and Fitzmaurice et al. (2012), while for a focus on quantile regression see Marino and Farcomeni (2015). Here, we consider quantile regression models that include unit-specific random coefficients to describe such a dependence; this gives rise to a conditional model specification which allows one to draw unit-level inferential conclusions on the effect of covariates on the longitudinal outcome of interest. In this framework, a standard way of proceeding is based on specifying a parametric distribution for the random coefficients, as suggested, e.g., by Geraci and Bottai (2007, 2014). An alternative consists in leaving such a distribution unspecified and estimating it from the observed data, by using a finite mixture specification. This can arise from discrete, unit-specific, random coefficients with unspecified distribution that remains constant or evolves over time according to a hidden Markov chain, as proposed by Alfó et al. (2017) and Farcomeni (2012), respectively. A more flexible specification based on combining both time-constant (TC) and time-varying (TV), unit-specific, discrete, random coefficients may also be adopted, as proposed by Marino et al. (2018). When compared to fully parametric alternatives, this semi-parametric approach offers a number of specific advantages, as it helps (*i*) avoid unverifiable assumptions on the random coefficient distribution; (*ii*) account for extreme and/or asymmetric departures from the homogeneous model; (*iii*) avoid integral approximations and, thus, considerably reduce the computational effort for parameter estimation.

In this paper, we describe the `R` package (R Core Team, 2019) `lqmix` (Marino et al., 2025), available from the Comprehensive `R` Archive Network (CRAN) at `https://CRAN.R-project.org/package=lqmix`, which is intended to provide maximum likelihood (ML) estimates for TC and/or TV mixtures of linear quantile regression models for longitudinal data. An indirect estimation approach, based on an extended Expectation-Maximization algorithm (EM - Dempster et al., 1977) is employed.

The package lqmix shares features with some relevant R packages available for the analysis of longitudinal data on the CRAN repository. It is related to packages lme4 (Bates et al., 2015) and lqmm (Geraci, 2014) tailored, respectively, to the analysis of general clustered observations via mixed models for the conditional mean and the conditional quantiles of a response. Here, TC random coefficients following a parametric distribution are considered to model dependence between observations and a ML approach is employed to derive parameter estimates. lqmix is also strongly related to the LMest R package (Bartolucci et al., 2017). This allows modeling the mean of longitudinal observations via hidden Markov models (Zucchini and MacDonald, 2009; Bartolucci et al., 2013). In this case, TV random coefficients with unspecified distribution that evolves over time according to a Markov chain are considered to account for dependence between measures from the same unit and a ML approach is adopted to derive parameter estimates. Other R packages that can be fruitfully related to lqmix are rqpd (Koenker and Bache, 2014), pqrfe (Danilevicz et al., 2022), and npmlreg (Einbeck et al., 2018). The former allows for the estimation of linear quantile regression models for panel data by considering either a penalized fixed effect estimation (Koenker, 2004) or a correlated random-effect method (Abrevaya and Dahl, 2008). In both cases, parameters are estimated by minimizing an extended quantile loss function. Similarly, pqrfe allows for the estimation of quantile regression models for longitudinal data based on unit-specific fixed effects; three different estimation methods are implemented, each based on the minimization of a different loss function. Last, the npmlreg R package entails mixtures of (generalized) linear models for clustered observations by employing a ML approach for parameter estimation. Finally, it is worth mentioning the quantreg R package (Koenker, 2022) for estimation and inference in linear models for conditional quantiles. Comparing these alternative packages with lqmix, it is worth highlighting that the latter fills in a blank by providing modeling tools not available in these other packages and ensures greater flexibility thanks to the non-parametric nature of the random coefficient distribution, which allows to avoid untestable parametric assumptions. The second benefit of lqmix entails the modeling of quantiles rather than the mean. Quantile regression allows one to analyze the impact that predictors may have on different parts of the conditional response distribution, as well as to deal with outliers and/or heavy tails that make the Gaussian assumption typically used for continuous data unreliable.

The modeling of longitudinal data via quantile regression is also possible by considering packages and environments out of the R ecosystem. However, none of these alternatives allow for the inclusion of random coefficients or for the possibility of modeling them in a non-parametric and/or dynamic way. The Stata modules xtqreg (Machado and Santos Silva, 2021) and xtmdqr (Pons and Melly, 2022) allow for the estimation of linear quantile regression models for longitudinal data based on the use of fixed effects. These modules differ in the way model parameters are estimated: the former considers the method of moments introduced by Machado and Santos Silva (2019), while the latter builds up on the minimum distance approach described by Galvao and Wang (2015). The qregpd Stata module (Baker, 2016) is also worth mentioning, as it implements the quantile regression estimator developed in Graham et al. (2015) for panel data. Last, the qreg Stata module (Machado et al., 2021) allows for the estimation of linear quantile regressions under the assumption of independent observations. Similarly does the quantreg SAS procedure.

The paper is structured as follows. The different proposals available in the literature on finite mixtures of linear quantile regression models are described in Section 2.2, where the case of TC, TV, and the combination of TC and TV random coefficients are discussed. ML estimation is reviewed in Section 2.3 where details on the EM algorithm and the bootstrap procedure for standard error estimation are described. The proposed R package is presented in Section 2.4, where the analysis of a benchmark dataset is discussed. Further methods available for the main functions of the package and not presented in 2.4 are discussed in 2.5, while Section 2.6 describes the function to estimate linear quantile regression for independent data. Finally, Section 2.7 contains some concluding remarks.

2 Modeling alternatives

Let us consider the case in which a longitudinal study is planned to record the value of a continuous response variable Y and a set of covariates (or predictors or explanatory variables) on a sample of n statistical units, at T different measurement occasions. We assume a balanced study, with common (at least in unit-time) and equally spaced occasions, in discrete time. Let Y_{it} denote the value of the response for unit $i = 1, \ldots, n$, at occasion $t = 1, \ldots, T$, and y_{it} the corresponding realization. Further, let $\boldsymbol{Y}_i = (Y_{i1}, \ldots, Y_{iT})'$ and $\boldsymbol{y}_i = (y_{i1}, \ldots, y_{iT})'$ be the T-dimensional vector of responses for unit i and its realization, respectively. A frequent issue to address when dealing with longitudinal studies is that of missingness. This may entail both the response and the covariates. Here, we assume that the latter are fully observed, while an ignorable missingness (Rubin, 1976) may affect the outcome. That is, some units in the sample may present incomplete response sequences due to either monotone or non-monotone missing data patterns (Little and Rubin, 2002). In this sense, a varying number of measures T_i may be available for each unit, even though missingness is assumed to be independent from unobserved responses – M(C)AR assumption.

Random coefficient models represent a standard approach to analyze the effect of observed covariates on a response Y that is repeatedly observed over time. This also holds in the quantile regression framework, where the interest is in modeling the conditional quantiles of the response distribution as a function of fixed and random coefficients. To ensure flexibility and avoid unverifiable parametric assumptions on the random coefficient distribution, a specification based on finite mixtures represents a viable strategy to adopt. For this purpose, we developed the `lqmix` R package.

In this section, we describe the methodology underlying the proposed package, and present some alternative formulations of quantile regression models available in the literature to deal with longitudinal data. In detail, we consider models based on discrete, unit-specific, random coefficients with unspecified distribution to capture unit-specific sources of unobserved heterogeneity due to omitted covariates. According to the chosen specification, these random coefficients may remain constant and/or evolve over time, leading to a model based on TC, TV, or both TC and TV random coefficients, respectively. The following sections present each of these formulations in detail.

2.1 Linear quantile mixtures with TC random coefficients

For a given quantile level $q \in (0, 1)$, let $\boldsymbol{\beta}_q$ denote a quantile-dependent, p-dimensional, vector of parameters associated with the (design) vector $\boldsymbol{x}_{it} = (x_{it1}, \ldots, x_{itp})'$. Also, let $\boldsymbol{z}_{it} = (z_{it1}, \ldots, z_{itd})'$ denote a set of $d \geq 1$ covariates (not included in $\boldsymbol{x}_{it}$) associated with a vector of unit- and quantile-specific random coefficients $\boldsymbol{b}_{i,q} = (b_{i1,q}, \ldots, b_{id,q})'$. This latter accounts for unobserved heterogeneity that is not captured by the elements in $\boldsymbol{x}_{it}$ and may be used to describe dependence between repeated measurements from the same unit. In this sense, conditional on $\boldsymbol{b}_{i,q}$, the longitudinal responses $Y_{i1}, \ldots, Y_{iT_i}$ are assumed to be independent of each other (local independence assumption). Moreover, for a given quantile level $q \in (0, 1)$ and conditional on the vector $\boldsymbol{b}_{i,q}$, the response Y_{it} is assumed to follow an Asymmetric Laplace Distribution (ALD - e.g., Yu and Moyeed, 2001), with density

$$f_{y|b}(y_{it} \mid \boldsymbol{b}_{i,q}; q) = \left[\frac{q(1-q)}{\sigma_q}\right] \exp\left\{-\rho_q\left[\frac{y_{it} - \mu_{it,q}}{\sigma_q}\right]\right\}.$$

Here, $\rho_q(\cdot)$ denotes the quantile asymmetric loss function (Koenker and Bassett, 1978), while q, σ_q, and $\mu_{it,q}$ denote the skewness, the scale, and the location parameter of the distribution, respectively. The ALD is a working model that is used to recast estimation of parameters for the linear quantile regression model in a maximum likelihood framework. The location parameter of the ALD, $\mu_{it,q}$, is modeled as

$$\mu_{it,q} = \boldsymbol{x}'_{it}\boldsymbol{\beta}_q + \boldsymbol{z}'_{it}\boldsymbol{b}_{i,q}. \tag{1}$$

The modeling structure is completed by the mixing distribution $f_{b,q}(\boldsymbol{b}_{i,q}; \boldsymbol{\Sigma}_q)$, i.e., the distribution of the random coefficients $\boldsymbol{b}_{i,q}$, where $\boldsymbol{\Sigma}_q$ identifies a (possibly) quantile-dependent covariance matrix. Rather than specifying such a distribution parametrically as in, e.g., Geraci and Bottai (2014), Alfó et al. (2017) proposed to leave it unspecified and use a Non-Parametric Maximum Likelihood approach (NPML – Laird, 1978; Lindsay, 1983a,b) to estimate it directly from the observed data. This approach is known to lead to the estimation of a (quantile-specific) discrete mixing distribution defined over the set of locations $\{\boldsymbol{\zeta}_{1,q}, \ldots, \boldsymbol{\zeta}_{G_q,q}\}$, with mixture probabilities $\pi_{g,q} = \Pr(\boldsymbol{b}_{i,q} = \boldsymbol{\zeta}_{g,q})$, $i = 1, \ldots, n$, $g = 1, \ldots, G_q$, and $G_q \leq n$. Under this approach, for $\boldsymbol{b}_{i,q} = \boldsymbol{\zeta}_{g,q}$, the location parameter of the ALD in Equation (1) becomes

$$\mu_{itg,q} = \boldsymbol{x}'_{it}\boldsymbol{\beta}_q + \boldsymbol{z}'_{it}\boldsymbol{\zeta}_{g,q},$$

while the model likelihood is defined by the following expression:

$$L(\cdot \mid q) = \prod_{i=1}^{n} \sum_{g=1}^{G_q} \left[\prod_{t=1}^{T_i} f_{y|b}(y_{it} \mid \boldsymbol{b}_{i,q} = \boldsymbol{\zeta}_{g,q}; q) \right] \pi_{g,q}. \tag{2}$$

This equation clearly resembles the likelihood of a finite mixture of linear quantile regression models with TC, discrete, random coefficients.

2.2 Linear quantile mixtures with TV random coefficients

While the model specification introduced in the previous section accounts for unit-specific omitted covariates, it may fall short in handling time variations in such unobserved heterogeneity. To address this limitation Farcomeni (2012) introduced a linear quantile regression model with TV, discrete, random intercepts. Rather than allowing the distribution of these intercepts to vary freely, it is modeled via a discrete-time Markov chain that ensures parsimony and interpretability. While such a proposal entails unit-specific intercepts only, a broader specification with general TV, discrete, random coefficients can also be considered.

As before, let $\boldsymbol{x}_{it} = (x_{it1}, \ldots, x_{itp})'$ denote a vector of p covariates associated with the vector of parameters $\boldsymbol{\beta}_q = (\beta_{1,q}, \ldots, \beta_{p,q})'$. Further, let $\boldsymbol{w}_{it} = (w_{it1}, \ldots, w_{itl})'$ denote a set of $l \geq 1$ explanatory variables not included in $\boldsymbol{x}_{it}$, associated with a vector of unit-, time-, and quantile-specific random coefficients $\boldsymbol{\alpha}_{it,q} = (\alpha_{it1,q}, \ldots, \alpha_{itl,q})'$. These are assumed to evolve over time according to a homogeneous, first order, hidden Markov chain $\{S_{it,q}\}$ that depends on the specified quantile level q. In detail, $\{S_{it,q}\}$ is defined over the finite state space $\mathcal{S}_q = \{1, \ldots, m_q\}$, with initial and transition probabilities $\delta_{s_{i1,q},q}$ and $\gamma_{s_{it,q}|s_{it-1,q},q}$, defined as:

$$\delta_{s_{i1,q},q} = \Pr(S_{i1,q} = s_{i1,q})$$

and

$$\gamma_{s_{it,q}|s_{it-1,q},q} = \Pr(S_{it,q} = s_{it,q} \mid S_{it-1,q} = s_{it-1,q}).$$

As for the model based on TC random coefficients detailed above, local independence holds, together with the convenient assumption of conditional ALD for the responses. For a given quantile level $q \in (0,1)$, the joint conditional distribution of the observed T_i-dimensional response vector $\boldsymbol{y}_i = (y_{i1}, \ldots, y_{iT_i})'$ is given by

$$f_{y|s}(\boldsymbol{y}_i \mid \boldsymbol{s}_{i,q}; q) = f_{y|s}(y_{i1} \mid s_{i1,q}; q) \prod_{t=2}^{T_i} f_{y|s}(y_{it} \mid s_{it,q}; q).$$

Here, $\boldsymbol{s}_{i,q} = (s_{i1,q}, \ldots, s_{iT_i,q})'$ is the vector of states visited by the i-th unit and $f_{y|s}(y_{it} \mid s_{it,q}; q)$ denotes the Asymmetric Laplace density with skewness q, scale σ_q, and location parameter $\mu_{its_{it,q},q}$. This latter is modeled as

$$\mu_{its_{it,q},q} = \boldsymbol{x}'_{it}\boldsymbol{\beta}_q + \boldsymbol{w}'_{it}\boldsymbol{\alpha}_{s_{it,q},q}.$$

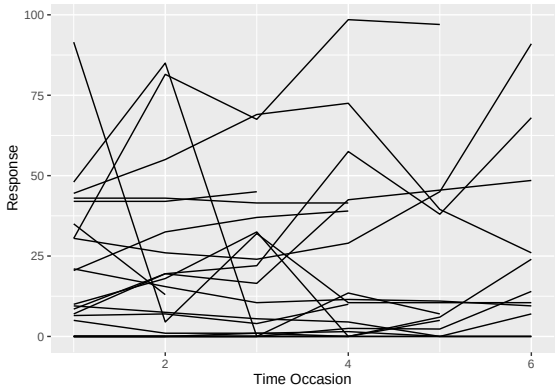


Figure 1: An example of longitudinal trajectories affected by both time-constant and time-varying unobserved heterogeneity

According to the assumptions above, the likelihood function is

$$L(\cdot \mid q) = \prod_{i=1}^{n} \sum_{s_{i1,q}=1}^{m_q} \cdots \sum_{s_{iT_i,q}=1}^{m_q} \left[\delta_{s_{i1,q},q} \prod_{t=2}^{T_i} \gamma_{s_{it,q}|s_{it-1,q},q} \right] \left[\prod_{t=1}^{T_i} f_{y|s}(y_{it} \mid s_{it,q}; q) \right].$$

When looking at the above expression, we may recognize it is a dynamic extension of Equation (2), as it represents the likelihood of a dynamic finite mixture of linear quantile regression models with TV, discrete, random coefficients. These coefficients are here assumed to vary as a function of the hidden states visited by the units over the observed time window.

2.3 Linear quantile mixtures with both TC and TV random coefficients

In some real data applications, both TC and TV sources of unit-specific unobserved heterogeneity may be present and influence the response distribution. We report in Figure 1 the longitudinal trajectories representing the evolution of a given continuous response measured over (at most) 6 time occasions for a sample of 25 statistical units. From this figure, it is clear that both TC and TV sources of unobserved heterogeneity affect the response. TC unobserved features may be responsible for differences between units in terms of baseline levels and systematic temporal trends; TV unobserved features may instead explain sudden temporal shocks characterizing individual trajectories. These latter may be rather difficult to capture via TC random coefficients or unit-specific random slopes associated with a time variable. In these situations, the linear quantile mixtures described above are no longer appropriate, as they may account for one source at a time only. To model empirical cases where both sources of between and within-unit variation are present, Marino et al. (2018) introduced a linear quantile regression model where TC and TV random coefficients may be jointly present in the linear predictor.

Let $\boldsymbol{x}_{it} = (x_{it1}, \ldots, x_{itp})'$ indicate a p-dimensional vector of covariates associated with the parameters $\boldsymbol{\beta}_q = (\beta_{1,q}, \ldots, \beta_{p,q})'$. Furthermore, let $\boldsymbol{z}_{it} = (z_{it1}, \ldots, z_{itd})'$ and $\boldsymbol{w}_{it} = (w_{it1}, \ldots, w_{itl})'$ denote two disjoint vectors of $d \geq 1$ and $l \geq 1$ explanatory variables (not included in $\boldsymbol{x}_{it}$), respectively. The former is associated with the vector of unit- and quantile-specific random coefficients $\boldsymbol{b}_{i,q} = (b_{i1,q}, \ldots, b_{id,q})'$ taking value in the set $\{\boldsymbol{\zeta}_{1,q}, \ldots, \boldsymbol{\zeta}_{G_q,q}\}$ with probability $\pi_{g,q} = \Pr(\boldsymbol{b}_{i,q} = \boldsymbol{\zeta}_{g,q}), g = 1, \ldots, G_q$. The latter is associated to the vector of unit-, time-, and quantile-specific random coefficients $\boldsymbol{\alpha}_{it,q} = (\alpha_{it1,q}, \ldots, \alpha_{itl,q})'$, which evolves over time according to a homogeneous, first order, quantile-dependent, hidden Markov chain $\{S_{it,q}\}$. As before, this is defined over the finite state space $\mathcal{S}_q = \{1, \ldots, m_q\}$ and is fully described by means of the initial probability vector $\boldsymbol{\delta}_q = (\delta_{1,q}, \ldots, \delta_{m_q,q})'$ and the transition probability matrix $\boldsymbol{\Gamma}_q$, with generic element $\gamma_{s_{it,q}|s_{it-1,q},q}$. The reader must note that, to ensure identifiability, the intercept term is included in either $\boldsymbol{z}_{it}$ or $\boldsymbol{w}_{it}$ (or in neither), but never in both. A similar principle applies to unit-specific, discrete, random slopes in the

model, ensuring that $\boldsymbol{z}_{it}$ and $\boldsymbol{w}_{it}$ have no common elements. Further, random coefficients $\boldsymbol{b}_{i,q}$ and $\boldsymbol{\alpha}_{it,q}$ are assumed to be independent.

For a given quantile level $q \in (0,1)$ and conditional on $\boldsymbol{b}_{i,q} = \boldsymbol{\zeta}_{g,q}$ and $\boldsymbol{\alpha}_{it,q} = \boldsymbol{\alpha}_{s_{it,q},q}$, longitudinal responses recorded from the same unit are assumed to be independent (local independence) and to follow an ALD with skewness, scale, and location parameter denoted by q, σ_q, and $\mu_{itgs_{it,q},q}$, respectively. This latter parameter is defined by the following regression model:

$$\mu_{itgs_{it,q},q} = \boldsymbol{x}'_{it}\boldsymbol{\beta}_q + \boldsymbol{z}'_{it}\boldsymbol{\zeta}_{g,q} + \boldsymbol{w}'_{it}\boldsymbol{\alpha}_{s_{it,q},q}.$$

Based on such assumptions, the likelihood function is

$$L(\cdot \mid q) = \prod_{i=1}^{n} \sum_{g=1}^{G_q} \sum_{s_{i1,q}=1}^{m_q} \cdots \sum_{s_{iT_i,q}=1}^{m_q} \left[\delta_{s_{i1,q},q} \prod_{t=2}^{T_i} \gamma_{s_{it,q}|s_{it-1,q},q} \right] \left[\prod_{t=1}^{T_i} f_{y|b,s}(y_{it} \mid \boldsymbol{b}_{i,q} = \boldsymbol{\zeta}_{g,q}, s_{it,q}; q) \right] \pi_{g,q},$$

where, as before, $f_{y|b,s}(y_{it} \mid \boldsymbol{b}_{i,q} = \boldsymbol{\zeta}_{g,q}, s_{it,q}; q)$ denotes the density of the ALD. In this framework, unobserved unit-specific features that remain constant over time are captured by the random coefficients $\boldsymbol{\zeta}_{g,q}$; sudden temporal shocks in the unit-specific profiles, due to TV sources of unobserved heterogeneity, are captured instead by the random coefficients $\boldsymbol{\alpha}_{s_{it,q},q}$.

To conclude this section, it is worth noticing that, when a single hidden state ($m_q = 1$) is considered and $\boldsymbol{b}_{i,q} = \boldsymbol{\zeta}_{g,q}$, the location parameter $\mu_{itgs_{it,q},q}$ simplifies to $\mu_{itg,q}$ and the model reduces to the linear quantile mixture with TC random coefficients only. Also, when a single mixture component is considered ($G_q = 1$), the location parameter $\mu_{itgs_{it,q},q}$ simplifies to $\mu_{its_{it,q},q}$ and the model reduces to the dynamic finite mixture of linear quantile regressions with TV random coefficients only. Last, when both G_q and m_q are equal to 1, the model reduces to a standard linear quantile regression model, without random coefficients. These properties make this latter specification more flexible and general than the alternatives described so far, at the cost of a higher computational complexity.

3 Model estimation and inference

In this section, we describe the algorithm for ML estimation of model parameters in the linear quantile mixture models detailed in the previous section. We focus on the specification based on both TC and TV random coefficients, as the simpler alternatives based on TC or TV random coefficients only arise as special cases. We provide details of an EM algorithm for parameter estimation in Section 2.3.1; the procedure for deriving standard errors and choosing the optimal number of mixture components and/or states of the hidden Markov chain is described in Section 2.3.2.

3.1 Maximum likelihood estimation

Let $\boldsymbol{\theta}_q$ denote the global set of free model parameters for a given quantile level $q \in (0,1)$. To derive an estimate for such a vector, we may rely on indirect maximization of the likelihood function via an extended version of the EM algorithm (Dempster et al., 1977). Let $u_{i,q}(g)$ and $v_{it,q}(h)$ be the indicator variables for $\boldsymbol{b}_{i,q} = \boldsymbol{\zeta}_{g,q}$ and $S_{it,q} = h$, respectively, and let $v_{it,q}(h,k) = v_{it-1,q}(h) \times v_{it,q}(k)$ denote the indicator variable for unit i moving from state h at occasion $t-1$ to state k at occasion t, with $g = 1, \ldots, G_q$, and $h, k = 1, \ldots, m_q$. For a given $q \in (0,1)$, the EM algorithm starts from the following complete-data log-likelihood

function:

$$\ell_c(\boldsymbol{\theta}_q) = \sum_{i=1}^{n} \left\{ \left[\sum_{g=1}^{G_q} u_{i,q}(g) \log \pi_{g,q} \right] + \left[\sum_{h=1}^{m_q} v_{i1,q}(h) \log \delta_{h,q} + \sum_{t=2}^{T_i} \sum_{h=1}^{m_q} \sum_{k=1}^{m_q} v_{it,q}(h,k) \log \gamma_{k|h,q} \right] + \left[\sum_{t=1}^{T_i} \sum_{g=1}^{G_q} \sum_{h=1}^{m_q} u_{i,q}(g) v_{it,q}(h) \log f_{y|b,s}(y_{it} \mid \boldsymbol{b}_{i,q} = \boldsymbol{\zeta}_{g,q}, S_{it,q} = h) \right] \right\}. \tag{3}$$

At the r-th iteration, the E-step of the EM algorithm requires the computation of the expected value of the complete-data log-likelihood in Equation (3), conditional on the observed data $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)'$ and the current parameter estimates $\hat{\boldsymbol{\theta}}_q^{(r-1)}$. That is, it requires the computation of

$$Q(\boldsymbol{\theta}_q \mid \hat{\boldsymbol{\theta}}_q^{(r-1)}) = \mathbb{E}\left[\ell_c(\boldsymbol{\theta}_q) \mid \boldsymbol{y}, \hat{\boldsymbol{\theta}}_q^{(r-1)} \right].$$

This means computing the posterior expectations of the indicator variables $u_{i,q}(g)$, $v_{it,q}(h)$, and $v_{it,q}(h,k)$. Regarding these two latter, a simplification is obtained by considering the forward and backward probabilities (Baum et al., 1970; Welch, 2003) typically used in the hidden Markov model framework; see Marino et al. (2018) for further details. In the M-step of the algorithm, parameter estimates $\hat{\boldsymbol{\theta}}_q$ are derived by maximizing $Q(\boldsymbol{\theta}_q \mid \hat{\boldsymbol{\theta}}_q^{(r-1)})$ with respect to $\boldsymbol{\theta}_q$.

The E- and the M-step are alternated until convergence, which is defined as the (relative) difference between two subsequent likelihood values being lower than a given threshold, $\varepsilon > 0$.

3.2 Standard errors and model selection

Following the standard procedure used in the quantile regression framework, standard errors for model parameter estimates are derived via a nonparametric bootstrap approach (see e.g., Buchinsky, 1995). In detail, we employ a block-bootstrap procedure, where a re-sampling of the statistical units is performed and the corresponding sequence of observed measurements is retained to preserve within-unit dependence (Lahiri, 1999).

Let $\hat{\boldsymbol{\theta}}_q^{(r)}$ denote the vector of model parameter estimates obtained in the r-th bootstrap sample, $r = 1, \ldots, R$. Estimates of the standard errors for the vector $\hat{\boldsymbol{\theta}}_q$ correspond to the diagonal elements of the matrix

$$\widehat{\mathbf{V}}(\hat{\boldsymbol{\theta}}_q) = \sqrt{\frac{1}{R} \sum_{r=1}^{R} \left(\hat{\boldsymbol{\theta}}_q^{(r)} - \hat{\boldsymbol{\theta}}_q \right) \left(\hat{\boldsymbol{\theta}}_q^{(r)} - \hat{\boldsymbol{\theta}}_q \right)'}.$$

A crucial point when dealing with finite mixtures is the choice of the number of components and/or hidden states. For a fixed quantile level q, a simple and frequently used solution is as follows. Parameter estimates are computed for varying combinations of the number of components and states, $[G_q, m_q]$; the model with the best fit, typically measured via log-likelihood or penalized likelihood criteria (such as AIC or BIC), is retained. However, given that these criteria may suffer from early stopping due to lack of progress rather than true convergence and the log-likelihood may present multiple local maxima, the EM algorithm is initialized from different starting points and the model corresponding to the highest likelihood value is retained before applying the penalization. In this regard, a deterministic start may be employed alongside a set of random start initializations. Specifically, the deterministic start is obtained by first estimating a linear regression model for the mean response using maximum likelihood, including all covariates – both those associated with fixed effects and those assumed to have a TC or TV random effect. The estimated fixed

regression coefficients serve as initial values for the corresponding parameters. Initial values for TC and/or TV random coefficients are obtained by adding an appropriate constant to the corresponding fixed effect estimates from the homogeneous model. Prior component probabilities and/or initial probabilities for the latent Markov chain are set uniformly; transition probabilities $\gamma_{k|h,q}, k, h = 1, \ldots, m_q$ are instead set to $(1 + w\mathbb{I}(k = h))/(m_q + w)$, where w is a proper tuning constant. Random starting points are generated by randomly perturbing the deterministic starting values.

While a multi-start strategy may mitigate the risk of local maxima, it is worth highlighting that the final solution may still lie on the boundary of the parameter space. This can lead to several issues such as (*i*) splitting components or latent states into multiple subgroups, (*ii*) convergence to (near-)identical parameter estimates across different components or hidden states; (*iii*) inflated standard errors; or (*iv*) instability in fixed-effect estimates. Therefore, it is strongly recommended to carefully examine the selected solution and prefer a simpler model (with a lower number of components and/or states) whenever there are indications of the above-mentioned issues. To conclude, note that the selection of the optimal number of components and/or states (G_q and/or m_q) should be guided by the need of capturing the unobserved sources of heterogeneity characterizing the data. In this sense, the primary goal is that of providing a sufficiently accurate approximation of the true, possibly continuous, distribution of the random coefficients in the model, rather than identifying homogeneous clusters of units, as typically done within the finite mixture framework.

4 The `lqmix` R package

In this section, we introduce the R package `lqmix`, developed to deal with linear quantile mixture models for longitudinal data. We illustrate the main functions for estimation and inference on the parameters of models described in the previous sections by considering the application to the labor pain benchmark dataset (Davis, 1991). Details on this dataset are given in the following.

4.1 Labor pain data

Firstly reported by Davis (1991) and since then analyzed by Jung (1996) and Geraci and Bottai (2007) among others, labor pain data come from a randomized clinical trial aiming at analyzing the effectiveness of a medication for relieving labor pain in women. A total of $n = 83$ women were randomized to a treatment/placebo group, and a response variable evaluating the self-reported amount of pain measured every 30 minutes on a 100-*mm* line was recorded. Here, 0 corresponds to absence of pain, while 100 corresponds to extreme pain. The number of available measurements per woman ranges from a minimum of 1 to a maximum of $T_i = 6, i = 1, \ldots, n$. That is, we are in the presence of an unbalanced longitudinal design, for which a MAR assumption seems to be reasonable. A total of $\sum_{i=1}^{83} T_i = 357$ measurements is available.

Together with the outcome of interest (`meas`), two covariates are available: `trt` denotes an indicator variable identifying the group each woman is assigned to (1 = treatment, 0 = placebo), while `time` identifies the measurement occasion the response was recorded at. Data are stored in the data frame `pain` included in the `lqmix` package, as shown below.

```
R> head(pain)

  id meas trt time
1  1  0.0   1    1
2  1  0.0   1    2
3  2  0.0   1    1
4  2  0.0   1    2
5  2  0.0   1    3
6  2  2.5   1    4
```

Data are severely skewed, and skewness changes magnitude and sign over time. In Figure 2, we report some selected diagnostics for a linear mixed model based on TC, Gaussian, random intercepts, using `trt`, `time`, and their interaction (`trt:time`) as covariates. The model is estimated by means of the `lmer` function implemented in the `lme4` R package.

```
R> outLME = lmer(meas ~ trt + time + trt:time + (1|id), data = pain)
R> par(mfrow = c(1,2))
R> qqnorm(residuals(outLME), main = "")
R> qqline(residuals(outLME))
R> qqnorm(unlist(ranef(outLME)$id), main = "")
R> qqline(unlist(ranef(outLME)$id))
```

In particular, Figure 2 reports the Normal probability plots for model residuals: the left panel shows unit- and time-specific residuals, while the right one shows the empirical Bayes estimates of unit-specific random intercepts. As it can be easily noticed, both plots indicate the presence of potentially influential observations in the data, as well as the violation of the Gaussian assumption for the random intercepts in the model. Therefore, using linear quantile mixtures seems to be a reasonable choice for the analysis of such data.

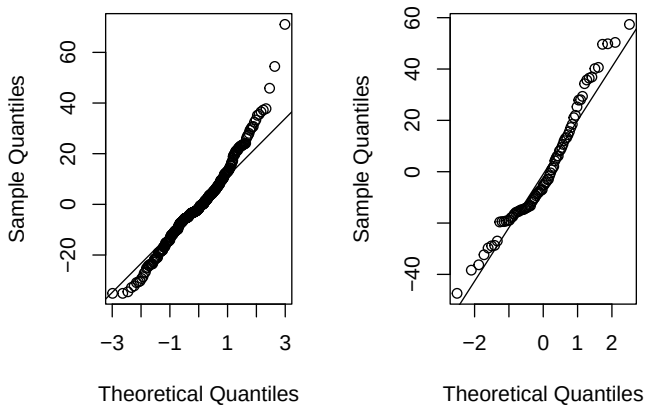


Figure 2: Labor pain data. Linear mixed effect model with Gaussian random intercepts. Normal probability plot for unit- and time-specific residuals (left plot) and for empirical Bayes estimates of unit-specific intercepts (right plot).

4.2 The `lqmix` function

The main function implemented in the `lqmix` R package is `lqmix`. It allows for the estimation of the parameters of a linear quantile mixture based on either TC, TV, or both types of discrete random coefficients. The input arguments can be displayed on the R console as follows:

```
R> args(lqmix)

function (formula, randomTC = NULL, randomTV = NULL, group, time, G = NULL, m = NULL, data,
  qtl = 0.5, eps = 10^-5, maxit = 1000, se = TRUE, R = 200, start = 0,
  parInit = list(betaf = NULL, betarTC =  NULL, betarTV = NULL, pg = NULL,
  delta = NULL, Gamma = NULL, scale = NULL),
  verbose = TRUE, seed = NULL, parallel = FALSE, ncores = 2)
```

The first mandatory argument, `formula`, denotes a two-side formula of the type `(resp` $\sim$ `Fexpr)`, where `resp` is the response variable and `Fexpr` is an expression determining the fixed-coefficient vector, x_{it}. On the other side, `randomTC` and `randomTV` are two one-side formulas of the type `(`$\sim$ `Rexpr1)` and `(`$\sim$ `Rexpr2)`, where `Rexpr1` and `Rexpr2` identify the columns of covariates associated with TC and TV random-coefficient vectors, z_{it} and w_{it}, respectively. Note that both these arguments are optional, so that the user may decide to estimate a linear quantile regression model with either TC or TV random coefficients, or with a combination of them. Further, variables reported in `Rexpr1` and `Rexpr2` must not overlap and, if any of them appears also in the fixed effect formula (`formula`), the corresponding parameter is estimated as a TC/TV random coefficient only. The arguments `group` and `time` are strings indicating the grouping and time variable, respectively. All such variables are taken from the data frame specified through the (mandatory) argument `data`. The arguments `G` and `m` are used to specify the number of mixture components and/or hidden states in the model, respectively, while `qtl` allows to specify the quantile level (by default, `qtl = 0.5`). The arguments `se`, `R`, and `start` allow to specify whether block-bootstrap standard errors should be computed (by default, `se = TRUE`), the number of bootstrap samples to be used for this purpose (by default, `R = 200`), and the initialization rule to consider. As regards this latter, three possible specifications are allowed: `start = 0` is used for a deterministic start of model parameters (the default option); `start = 1` is used for a random start of model parameters; `start = 2` is used to consider a starting rule based on given model parameters specified via the `parInit` list argument. The arguments `maxit`, `eps`, and `verbose` identify the maximum number of iterations for the EM algorithm (by default, `maxit = 1000`), the corresponding tolerance level (by default `eps` $= 10^{-5}$), and whether output should be printed (by default, `verbose = TRUE`), respectively. As for the `seed` argument, this is devoted to setting a seed for random number generation that is used for the random starting rule (`start=1`) described so far and for computing parameters' standard errors. The arguments `parallel` and `ncores` control parallel computing when standard errors are requested, with `ncores` defaulting to 2.

Estimating linear quantile mixtures with TC random coefficients

To explore features of the `lqmix` function, we start by considering a two-component linear quantile mixture ($G_q = 2$) for the median (`qtl` $= 0.5$) with a TC random intercept for the analysis of the pain data. This is estimated as follows:

```
R> outTC = lqmix(formula = meas ~ time + trt + trt:time, randomTC = ~1, time = "time",
  group = "id", G = 2, data = pain)

--------|-------|-------|--------|-------------|-------------|
  model |  qtl  |   G   |  iter  |      lk     |   (lk-lko)  |
--------|-------|-------|--------|-------------|-------------|
     TC |   0.5 |     2 |      0 |    -1682.31 |          NA |
     TC |   0.5 |     2 |      8 |    -1607.11 | 4.64719e-06 |
--------|-------|-------|--------|-------------|-------------|
Computing standard errors ...
  |=====================================================================| 100%
```

The running time for the above command is 5.212 seconds when run on an Apple M1 architecture (16GB, MacOS: Sequoia 15.6.1). This represents the architecture used for all the codes illustrated in the paper. In the following, we report the output of the above call to `lqmix`, which is an object of class `lqmix`, obtained by using the `print` method of the `S3` class.

```
R> outTC

Model: TC random coefficients with G=2 at qtl=0.5
```

```
**************************************************

---- Observed process ----

Fixed Coefficients:
    time      trt time.trt
 11.6669  -2.1659 -10.8335

Time-Constant Random Coefficients:
      (Intercept)
Comp1      1.3325
Comp2     43.3325

Residual scale parameter: 7.3289 - Residual standard deviation: 20.7294

---- Latent process ----

Mixture probabilities:
 Comp1  Comp2
0.6737 0.3263

Log-likelihood at convergence: -1607.11
Number of observations: 357 - Number of subjects: 83
```

Looking at the output, we may recognize two separate sections, reporting estimates for the observed and the latent process, respectively. As regards the observed process, the estimated fixed coefficients suggest how self-reported pain increases as time passes by, together with a positive effect of the treatment under investigation (negative sign for the `trt` variable), with benefits that increase with time. On the other side, the estimated random coefficients highlight the presence of two well-separated groups of women, reporting a low and a medium pain at the baseline, respectively. In the last part, the estimated scale parameter and the corresponding error standard deviation are shown. This latter corresponds to the standard deviation of an ALD. As regards the latent process, estimates highlight that 67.37% of women belong to the first mixture component (low-pain level); the remaining 32.63% belong to the second one (medium-pain level). All estimated parameters are stored in the object `outTC` as `betaf` (the fixed coefficients), `betarTC` (the TC random coefficients), `scale` (the scale parameter), `sigma.e` (the conditional standard deviation of responses), and `pg` (the component prior probabilities). Further, when leaving the statement `se = TRUE` (default value), the `outTC` object also contains information on the estimated standard errors of model parameters obtained via the bootstrap procedure detailed in Section "*Standard Errors and model selection*". These standard errors can be accessed by prefixing the corresponding parameter names with `se.`, as listed above.

The following information is also stored in the the `outTC` object: the log-likelihood value at convergence (`lk`), the number of model parameters (`npar`), the values of AIC and BIC (`aic` and `bic`), the quantile level (`qtl`), the number of mixture components (`G`), the total number of subjects and observations (`nsbjs` and `nobs`), the mixture components' posterior probabilities obtained at convergence of the EM algorithm (`postTC`), the bootstrap variance-covariance matrices of the regression coefficients (`vcov`), the type of missingness data are assumed to be affected by (`miss`), the estimated model (`model`), and the model call (`call`). Last, model matrices associated with fixed and TC random coefficients are stored in the `outTC` object as `mmf` and `mmrTC`, respectively.

Estimating linear quantile mixtures with TV random coefficients

The same `lqmix` function can be used to obtain parameter estimates for a quantile mixture with TV random coefficients. To analyze the labor pain data, we consider a linear quantile

mixture for the median (`qtl` $= 0.5$), with a TV random intercept defined over a two-state latent space ($\mathtt{m_q} = 2$).

```
R> outTV = lqmix(formula = meas ~ time + trt + trt:time, randomTV = ~1, time = "time",
  group = "id", m = 2, data = pain)

--------|-------|-------|--------|-------------|-------------|
  model |  qtl  |   m   |  iter  |      lk     |   (lk-lko)  |
--------|-------|-------|--------|-------------|-------------|
     TV |   0.5 |     2 |      0 |    -1688.99 |          NA |
     TV |   0.5 |     2 |     10 |    -1576.61 |    0.366345 |
     TV |   0.5 |     2 |     20 |    -1575.79 | 9.32048e-05 |
     TV |   0.5 |     2 |     29 |    -1575.79 | 9.43243e-06 |
--------|-------|-------|--------|-------------|-------------|
Computing standard errors ...
  |======================================================================| 100%
```

The running time for obtaining results is 11.285 seconds. The output is given below.

```
R> outTV

Model: TV random coefficients with m=2 at qtl=0.5
*****************************************************

---- Observed process ----

Fixed Coefficients:
    time      trt time.trt
  6.5012  -0.4923  -6.0013

Time-Varying Random Coefficients:
    (Intercept)
St1      0.4924
St2     60.9926

Residual scale parameter: 5.8337 - Residual standard deviation: 16.5003

---- Latent process ----

Initial probabilities:
   St1    St2
0.7667 0.2333

Transition probabilities:
         toSt1  toSt2
fromSt1 0.8883 0.1117
fromSt2 0.0271 0.9729

Log-likelihood at convergence: -1575.795
Number of observations: 357 - Number of subjects: 83
```

Results for the observed process allow us to derive similar conclusions to those detailed in the previous section, even though the magnitude of the effects is reduced. As regards the latent layer of the model, the `print` method shows the estimates for the initial and the transition probabilities of the hidden Markov chain. Based on such estimates, one may conclude that 76.67% of women start in the first hidden state which, based on the estimated time-varying random intercepts, is characterized by a low pain level. The remaining 23.33%

of women start the study with an intermediate baseline pain level. Looking at the estimated transition probabilities, we may notice that, regardless of the treatment effect, labor pain tends to increase as time passes by, with women being in the *low pain* state moving towards the *medium pain* state with probability equal to 0.1117. Estimated parameters are stored in the object `outTV` as `betaf` (the fixed coefficients), `betarTV` (the TV random coefficients), `scale` (the scale parameter), `sigma.e` (the conditional standard deviation of responses derived from an ALD with parameters `scale` and `qtl`), `delta` (the initial probability vector), and `Gamma` (the transition probability matrix). As detailed in the previous section, bootstrap standard errors are stored by using the prefix "`se.`" to the names of parameters of interest. Last, posterior probabilities for the latent states of the hidden Markov chain and the model matrix for the TV random coefficients are stored as `posTV` and `mmrTV`, respectively, in the `outTV` object.

Estimating linear quantile mixtures with TC and TV random coefficients

When specifying both the `randomTC` and the `randomTV` formulas, and therefore both `G` and `m`, the `lqmix` function allows for the estimation of linear quantile mixtures based on both TC and TV, discrete, random coefficients. To analyze labor pain data, we consider a linear quantile mixture for the median (`qtl = 0.5`), with a TV random intercept, based on $m_q = 2$ hidden states, and a TC random slope for the `time` variable, based on $G_q = 2$ components. To estimate such a model and consider a random start initialization (`start=1`), the following command can be run:

```
R> outTCTV = lqmix(formula = meas ~ trt + time + trt:time, randomTC = ~ time, randomTV = ~1,
  time = "time", group = "id", m = 2, G = 2, data = pain, se = FALSE, start = 1, seed = 10)

--------|-------|-------|-------|--------|-------------|-------------|
  model |  qtl  |   m   |   G   |  iter  |      lk     |   (lk-lko)  |
--------|-------|-------|-------|--------|-------------|-------------|
   TCTV |   0.5 |     2 |     2 |      0 |    -1723.86 |          NA |
   TCTV |   0.5 |     2 |     2 |     10 |    -1594.61 |     14.8135 |
   TCTV |   0.5 |     2 |     2 |     20 |    -1554.41 |     3.47302 |
   TCTV |   0.5 |     2 |     2 |     30 |    -1541.16 |    0.165982 |
   TCTV |   0.5 |     2 |     2 |     37 |    -1541.12 | 2.98304e-06 |
--------|-------|-------|-------|--------|-------------|-------------|
```

This requires a running time of 11.285 seconds. The output of the above call to `lqmix`, obtained through the `print` method of the `S3` class, is reported below.

```
R> outTCTV

Model: TC and TV random coefficients with m=2 and G=2 at qtl=0.5
****************************************************************

---- Observed process ----

Fixed Coefficients:
     trt trt.time
 -9.2859  -5.0428

Time-Constant Random Coefficients:
         time
Comp1  5.5428
Comp2 14.9178

Time-Varying Random Coefficients:
```

```
    (Intercept)
St1      8.7859
St2     69.7859

Residual scale parameter: 5.1259 - Residual standard deviation: 14.4983

---- Latent process ----

Mixture probabilities:
 Comp1  Comp2
0.7381 0.2619

Initial probabilities:
   St1    St2
0.8036 0.1964

Transition probabilities:
         toSt1  toSt2
fromSt1 0.9641 0.0359
fromSt2 0.0434 0.9566

Log-likelihood at convergence: -1541.12
Number of observations: 357 - Number of subjects: 83
```

As it is clear from the output, the linear quantile mixture with both TC and TV discrete random coefficients produces more detailed information when compared to those discussed so far. Also in this case, we may recognize in the output the two sections entailing the observed and the latent process, respectively. In the former, estimates for the fixed, TC, and TV random coefficients are reported. For the latent part, the output reports information on the mixture, the initial, and the transition probabilities.

By looking at the results, we conclude again that self-reported pain is lower for women under medication (negative sign for the `trt` variable) and that benefits increase with time. Looking at the estimated TC random coefficients and the corresponding prior probabilities, we may distinguish two well separated groups of women. They exhibit different trends in pain levels over time: 73.81% experience a mild increase in pain levels, whereas 26.19% show a steeper increase. On the other side, the estimated TV random coefficients identify two groups of women characterized by a low and a medium baseline labor pain level, respectively. At the beginning of the observation period, 80.36% of women belong to the first group, while the remaining 19.64% to the second. By looking at the estimated transition probability matrix, we conclude that the group composition remains largely unchanged over time once controlling for the other effects in the model ($\gamma_{hh}, 0.5 > 0.95, h = 1, 2$).

Estimated parameters are stored in the object `outTCTV`. Now both those related to the TC finite mixture as well as those related to the hidden Markov chain are referenced. In detail, `outTCTV` contains `betaf` (the fixed coefficients), `betarTC` (the TC random coefficients), `betarTV` (the TV random coefficients), `scale` (the scale parameter), `sigma.e` (the conditional standard deviation of responses derived from an ALD with parameters `scale` and `qtl`), `pg` (the component prior probabilities), `delta` (the initial probabilities), and `Gamma` (the transition probabilities). Standard errors (if computed), as well as additional information on the data, the estimated model, posterior probabilities, and model matrices are stored in the `outTCTV` object.

4.3 The `search_lqmix` function for model selection

As described in Section "*Standard errors and model selection*", the number of mixture components G_q and the number of hidden states m_q in the model are unknown quantities that need to be estimated. Moreover, a multi-start strategy is frequently needed to solve, at least

partially, the potential multimodality of the likelihood surface. Both issues can be addressed by means of the `search_lqmix()` function. Input arguments and corresponding default values can be displayed on the R console though the `args` function:

```
R> args(search_lqmix)

function (formula, randomTC = NULL, randomTV = NULL, group, time, Gv = NULL, mv = NULL, data,
  method = "bic", nran = 0, qtl = 0.5, eps = 10^-5, maxit = 1000, se = TRUE, R = 200,
   verbose = TRUE, seed = NULL, parallel = FALSE, ncores = 2)
```

Most of the arguments of this function correspond to those described so far. The remaining ones are specified as follows: `Gv` and `mv` denote vectors identifying the range for the number of mixture components, G_q, and the number of hidden states, m_q, to be considered, respectively, for a fixed quantile level $q \in (0,1)$. When both arguments are specified, the search of the optimal linear quantile mixture with TC and TV random coefficients is performed. When only one out the two arguments is specified, a linear quantile mixture based on either TC or TV random coefficients is estimated. A linear quantile regression with no random coefficients is also estimated either when both `mv` and/or `Gv` include the value 1, or when `Gv` includes 1 and `mv = NULL`, or when `mv` includes 1 and `Gv = NULL`. In this case, the function `lqr()` implemented in the `lqmix` R package and described in the following is employed.

The argument `method` is used for model selection purposes. One of three possible values is admitted: "`bic`" (by default), "`aic`", or "`lk`". The former two identify the optimal model as that providing the minimum value of the BIC or the AIC index, respectively. The latter, selects the optimal model as the one corresponding to the maximum log-likelihood value.

The argument `nran` specifies the number of random starts to be considered for each value in the range identified by `Gv` and `mv`. This number is strictly related to the type of model for which the optimal search is performed. In detail, following a similar strategy as that suggested by Bartolucci et al. (2017) for the `LMest` R package, when a linear quantile mixture based on TC random coefficients only is considered, the number of random initializations is set equal to $\texttt{nran} \times (\texttt{G}_\texttt{q} - 1)$; when only TV random coefficients are allowed, the number of random initializations is set equal to $\texttt{nran} \times (\texttt{m}_\texttt{q} - 1)$; last, when both TC and TV random coefficients are considered, the number of random initializations is set equal to $\texttt{nran} \times (\texttt{G}_\texttt{q} - 1) \times (\texttt{m}_\texttt{q} - 1)$. By default, $\texttt{nran} = 0$, so that no random initializations are considered. The `seed` argument is used to fix a seed and ensure reproducibility of results when the multi-start strategy based on random initializations is considered to estimate model parameters, as well as for deriving standard errors (when requested). Last, `parallel` and `ncores` are used for parallel computing of the standard errors.

We report below the results of the model selection strategy applied to the `pain` data, when focusing on a linear quantile mixture with a TV random intercept and a TC random slope associated to the variable `time`, for the quantile level $q = 0.5$. The search is done by looking for the optimal number of components (G_q) and hidden states (m_q), both in the set $\{1,2\}$, by setting `nran = 50`. The optimal model is selected according to the BIC index.

```
R>  sTCTV = search_lqmix(formula = meas ~ trt + time + trt:time, randomTC = ~time,
  randomTV = ~1, group = "id", time = "time", nran = 50, mv = 1:2, Gv = 1:2,
  data = pain, seed = 10)

Search the optimal linear quantile mixture model
************************************************
Random start: 0 ...
--------|-------|--------|-------------|
  model |  qtl  |  iter  |      lk     |
--------|-------|--------|-------------|
    HOM |   0.5 |      0 |    -1707.73 |
--------|-------|--------|-------------|
Random start: 0 ... 1  ... 2  ... 3  ... 4  ... 5  ... 6  ... 7  ... 8  ... 9  ... 10  ... 11  ...
```

```
--------|-------|-------|--------|-------------|-------------|
  model |  qtl  |   G   |  iter  |      lk     |   (lk-lko)  |
--------|-------|-------|--------|-------------|-------------|
     TC |   0.5 |     2 |      0 |    -1626.59 |          NA |
     TC |   0.5 |     2 |      1 |    -1626.59 | 6.65801e-07 |
--------|-------|-------|--------|-------------|-------------|
Random start: 0 ... 1  ... 2  ... 3  ... 4  ... 5  ... 6  ... 7  ... 8  ... 9  ... 10  ... 11  ...
--------|-------|-------|--------|-------------|-------------|
  model |  qtl  |   G   |  iter  |      lk     |   (lk-lko)  |
--------|-------|-------|--------|-------------|-------------|
     TV |   0.5 |     2 |      0 |    -1575.79 |          NA |
     TV |   0.5 |     2 |      4 |    -1575.79 | 8.79175e-06 |
--------|-------|-------|--------|-------------|-------------|
Random start: 0 ... 1  ... 2  ... 3  ... 4  ... 5  ... 6  ... 7  ... 8  ... 9  ... 10  ... 11  ...
--------|-------|-------|-------|--------|-------------|-------------|
  model |  qtl  |   m   |   G   |  iter  |      lk     |   (lk-lko)  |
--------|-------|-------|-------|--------|-------------|-------------|
   TCTV |   0.5 |     2 |     2 |      0 |    -1538.76 |          NA |
   TCTV |   0.5 |     2 |     2 |      3 |    -1538.76 | 8.36816e-06 |
--------|-------|-------|-------|--------|-------------|-------------|
Computing standard errors for the optimal model...
  |=========================================================================| 100%
```

The running time for the above command is 33.332 seconds and the output is stored in the `sTCTV` object. This can be shown by means of the `print` method of the class `S3` as follows:

```
R>  sTCTV

Opt model: TC and TV random coefficients with m=2 and G=2 at qtl=0.5
**********************************************************************

---- Observed process ----

Fixed Coefficients:
     trt trt.time
 -4.3237  -5.0294

Time-Constant Random Coefficients:
         time
Comp1  5.3294
Comp2 15.6627

Time-Varying Random Coefficients:
    (Intercept)
St1      4.0237
St2     49.1746

Residual scale parameter: 4.7858 - Residual standard deviation: 13.5363

---- Latent process ----

Mixture probabilities:
 Comp1  Comp2
0.6351 0.3649

Initial probabilities:
   St1    St2
```

```
0.7553 0.2447

Transition probabilities:
         toSt1  toSt2
fromSt1 0.9552 0.0448
fromSt2 0.1497 0.8503

Log-likelihood at convergence: -1538.76
Number of observations: 357 - Number of subjects: 83
```

As reported in the first output line, the optimal model for the median (`qtl = 0.50`) according to the BIC criterion (the default option) is based on $G_q = 2$ mixture components and $m_q = 2$ hidden states. Results we obtain are in line with those reported in the previous section. Differences are due to the model selection strategy that aims at identifying the global maximum of the likelihood function.

To provide further insights into the analysis of the `pain` data and look at the potential of the `lqmix R` package, we also search for the optimal model for a different quantile level. Specifically, we apply the `search_lqmix` function for `qtl = 0.75` to study the impact of observed covariates on higher pain levels. When looking at the results reported below, we may notice that again the optimal specification (according to the BIC criterion) is obtained for $G_q = 2$ and $m_q = 2$. Further, pain levels are lower when medication is taken, even though the estimated effect is lower than before. This result claims a lower beneficial effect of treatment for those women reporting higher pain levels. Comparing the estimated TC random coefficients for `qlt = 0.75` to those from the model for `qtl = 0.50`, we observe now a more pronounced increase of pain levels over time. The estimated TV random coefficients identify instead two groups of women declaring again a low and a medium baseline pain level, respectively. Baseline estimates, as expected, are now higher than those obtained for `qtl = 0.50`. The estimated transition probability matrix highlights a very high persistence in each of the two states; this means that baseline levels remain pretty constant over time.

```
R> sTCTV75 = search_lqmix(formula = meas ~ trt + time + trt:time, randomTC = ~time,
                  randomTV = ~1, nran = 50, group = "id", time = "time", mv = 1:2,
                      Gv = 1:2, data = pain, seed = 10, qtl = 0.75)

R> sTCTV75

Opt model: TC and TV random coefficients with m=2 and G=2 at qtl=0.75
*********************************************************************

---- Observed process ----

Fixed Coefficients:
     trt trt.time
 -1.0538  -7.6027

Time-Constant Random Coefficients:
         time
Comp1  8.5027
Comp2 17.9231

Time-Varying Random Coefficients:
    (Intercept)
St1      5.1538
St2     65.9891

Residual scale parameter: 4.2329 - Residual standard deviation: 17.8476
```

```
---- Latent process ----

Mixture probabilities:
 Comp1  Comp2
0.6965 0.3035

Initial probabilities:
   St1    St2
0.7393 0.2607

Transition probabilities:
         toSt1  toSt2
fromSt1 0.9761 0.0239
fromSt2 0.0526 0.9474

Log-likelihood at convergence: -1577.189
Number of observations: 357 - Number of subjects: 83
```

4.4 The `summary` method for `lqmix` and `search_lqmix` objects

The `summary` method, when applied to objects of class `lqmix` or `search_lqmix`, generates a summary object that allows for inference on model parameters. For `search_lqmix` objects, it returns the summary of the optimal model. In the following, we present the output of the `summary` method for the `sTCTV` object described so far.

```
R> summary(sTCTV)

Opt model: TC and TV random coefficients with m=2 and G=2 at qtl=0.5
*****************************************************************************

---- Observed process ----

Fixed Coefficients:
         Estimate St.Error z.value P(>|z|)
trt       -4.3237   2.6423 -1.6363  0.0997 .
trt.time  -5.0294   0.7494 -6.7108  <2e-16 ***

Time-Constant Random Coefficients:
           Estimate St.Error z.value   P(>|z|)
time_Comp1   5.3294   0.5207  10.236 < 2.2e-16 ***
time_Comp2  15.6627   0.8050  19.457 < 2.2e-16 ***

Time-Varying Random Coefficients:
                Estimate St.Error z.value P(>|z|)
(Intercept)_St1   4.0237   1.6985   2.369  0.0175 *
(Intercept)_St2  49.1746   4.8451  10.149  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual scale parameter: 4.7858 - Residual standard deviation: 13.5363

---- Latent process ----

Mixture probabilities:
      Estimate St.Error
```

```
Comp1   0.6351   0.0668
Comp2   0.3649   0.0668

Initial probabilities:
    Estimate St.Error
St1   0.7553   0.0507
St2   0.2447   0.0507

Transition probabilities:
             Estimate St.Error
fromSt1toSt1   0.9552   0.0181
fromSt1toSt2   0.0448   0.0181
fromSt2toSt1   0.1497   0.0546
fromSt2toSt2   0.8503   0.0546

Log-likelihood at convergence: -1538.76
Number of observations: 357 - Number of subjects: 83
```

As before, two different sections may be distinguished. In the former, estimates, standard errors, test statistics, and corresponding approximate p-values for assessing significance of model parameters for the observed process are reported. This information is completed with the estimates of the scale parameter and the conditional standard deviation of the error terms. In the latter section of the output, estimates and standard errors for the parameters characterizing the latent layer of the model are reported.

5 Other methods for `lqmix` and `search_lqmix` objects

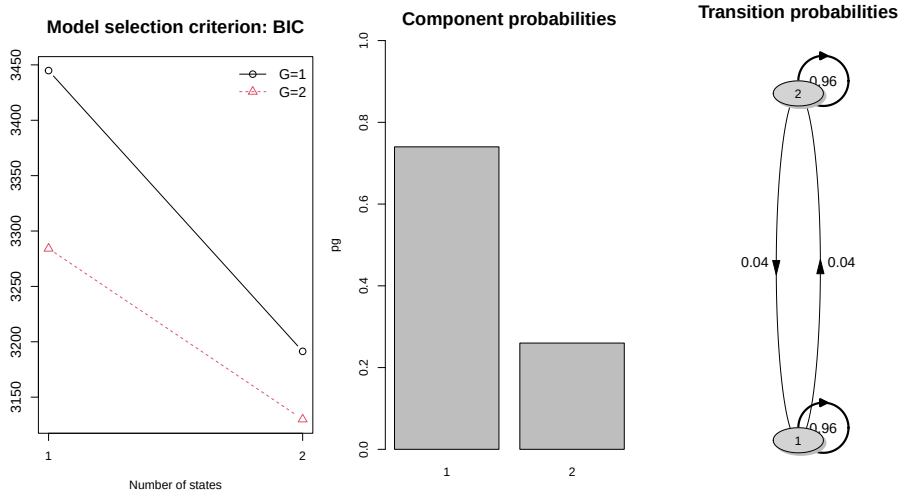
To provide users with a familiar and consistent interface, the `lqmix` R package includes basic methods typically available for regression model objects. Together with the `print` and the `summary` methods introduced in the previous sections, the `logLik` and `coef` methods are implemented. These return the maximum log-likelihood value obtained at convergence of the EM algorithm and the estimated fixed effect coefficients, respectively. Further, the methods `AIC` and `BIC` may be used for model selection purposes. When the above methods are applied to objects of class `search_lqmix`, these refer to the optimal specification identified via the `search_lqmix` function. The `plot` method returns graphical representations for the outputs of the `lqmix` and the `search_lqmix` functions. In detail, a graphical display of the mixture component probabilities and/or of the transitions across states of the hidden Markov chain is provided when applied to an `lqmix` object. A plot reporting the value of the chosen model selection criterion for varying number of components and/or hidden states (defined in `Gv` and `mv`, respectively) is also provided when applied to objects of class `search_lqmix`. We report in Figure 3 the results of such a method applied to the `sTCTV` object detailed in Section 2.4.3.

Further methods implemented in the package are `predict`, `residuals`, and `vcov`. The former two return the conditional predicted values and the conditional residuals, given the component membership and/or the state occupied at a given occasion by units in the sample; `vcov` returns instead the variance-covariance matrix of fixed model parameters obtained from the block-bootstrap procedure detailed above.

6 Linear quantile regression for independent data

The package `lqmix` is thought for dealing with longitudinal data. However, it also implements the function `lqr()` for estimating a homogeneous linear quantile regression model via a ML approach. This is obtained by exploiting the parallelism between the ALD and the asymmetric quantile loss function. Input arguments for the `lqr` function and corresponding default values are as follows:

Figure 3: Results from the `plot` method




```
R> args(lqr)

function (formula, data, qtl = 0.5, se = TRUE, R = 100, verbose = TRUE, seed = NULL,
  parallel = FALSE, ncores = 2)
```

These arguments are identical to those described above for the functions that are built to estimate parameters of linear quantile mixtures. As evident, also in this case the user may specify whether standard errors need to be computed; a bootstrap approach for independent observations is employed. By default, `R = 200` re-samples are considered. All methods described in the previous section and available for `lqmix` and `search_lqmix` objects are also available for objects of class `lqr`.

7 Conclusions

Quantile regression is a fundamental tool of analysis when (*i*) the response distribution is skewed, (*ii*) outlying values are present, (*iii*) interest entails both the center and the tails of the response distribution. When dealing with longitudinal data, dependence between observations from the same unit must be properly considered to avoid biased inference. Thus, linear quantile regression models with unit-specific random coefficients may be effectively employed.

In this paper, we present the `lqmix` R package to estimate finite mixtures of linear quantile regression models for continuous longitudinal data, possibly affected by random missingness. Strong and unverifiable parametric assumptions on the random coefficients are avoided by considering a mixture specification. Either TC or TV random coefficients, or a combination of both, can be fitted to deal with different sources of unobserved heterogeneity affecting the longitudinal process. That is, the package allows us to deal with unit-specific unobserved features (omitted covariates in the model) which may vary and/or stay constant over the observational period, while looking at the quantiles of the conditional response variable.

Three main functions are implemented in the package. A first one, `lqmix`, allows the estimation of the different model specifications obtained by considering TC or TV random coefficients only, or a combination of them. A second function, `search_lqmix`, allows for the searching of the optimal model specification, based on various model selection criteria. A third function, `lqr`, is devoted to the estimation of linear quantile regression models for cross-sectional data. For a concise understanding, we report in Table 1 a description of the main parameters required by such functions.

Table 1: Description of the main arguments for the functions `lqmix`, `search_lqmix`, and `lqr`.

Arguments	Description	`lqmix`	`search_lqmix`	`lqr`
`formula`	an object of class `formula`: a symbolic description of the model to be fitted of the form `resp` $\sim$ `Fexpr`	yes	yes	yes
`randomTC`	a one-sided formula of the form $\sim z_1 + z_2 + ... + z_r$	yes	yes	no
`randomTV`	a one-sided formula of the form $\sim w_1 + w_2 + ... + w_l$	yes	yes	no
`group`	a string indicating the grouping variable	yes	yes	no
`time`	a string indicating the time variable	yes	yes	no
`G`	number of mixture components associated to TC random coefficients	yes	no	no
`m`	number of states associated to the TV random coefficients	yes	no	no
`Gv`	vector of possible number of mixture components associated to TC random coefficients	no	yes	no
`mv`	vector of possible number of states associated to TV random coefficients	no	yes	no
`data`	a data frame containing the variables named in `formula`, `randomTC`, `randomTV`, `group`, and `time`	yes	yes	yes
`qtl`	quantile to be estimated	yes	yes	yes
`eps`	tolerance level for (relative) convergence of the EM algorithm	yes	yes	no
`maxit`	maximum number of iterations for the EM algorithm	yes	yes	no
`se`	standard error computation for the optimal model	yes	yes	yes
`R`	number of bootstrap samples for computing standard errors	yes	yes	yes
`nran`	number of repetitions of each random initialization	no	yes	no

When looking at the scalability of `lqmix`, the computational effort increases with the complexity of the model. Linear quantile mixtures of quantile regressions with TC random coefficients are computationally less demanding than those with TV random coefficients, which in turn are more efficient than models incorporating both TC and TV random terms. Additional factors influencing computational load include the number of repeated measures and, most significantly, the number of units in the sample. In our experience, `lqmix` easily handles datasets of small to moderate size (about 10 thousand units). For larger datasets, the computational effort increases, particularly in the estimation of standard errors. In this respect, parallel computing proves beneficial in managing computation times and maintaining performance.

Another aspect that is important to highlight entails the selection of the number of bootstrap samples to consider for the estimation of the standard errors, R. Practitioners are always recommended to set `R` to the value that ensures the stability of results, bearing in mind that the higher the model complexity the larger R is expected to be.

Further updates of the package will include the possibility to deal with multivariate responses in the spirit of Alfò et al. (2021). Here, outcome-specific random coefficients are considered to model the unobserved heterogeneity characterizing each response variable. The corresponding multivariate distribution is left unspecified and estimated directly from the data. This proposal can be extended to the TV setting, as well as to the mixed one (based on both TC and TV random coefficients) and implemented in the package. The inclusion of observed covariates on the latent layer of the model (as in mixtures of experts models) represents a further extension we aim at working on.

Acknowledgments

The work of Ranalli has been developed under the support of "Fondo Ricerca di Ateneo, Università degli Studi di Perugia, edition 2021, project: AIDMIX". The work of Marino has been supported by the "Department of Excellence" projects, 2018-2022 and 2023-2027, from Italian Ministry of Education, University and Research. The work of Salvati has been supported by the grant "PRIN 2025 PNRR Prot. P2025TB5JF: Quantification in the Context of Dataset Shift (QuaDaSh)" from Italian Ministry of Education, University and Research.

The work of Alfò has been supported by Sapienza university grant n. RG124191029C1395 "Latent variable models for complex health data".

References

J. Abrevaya and C. M. Dahl. The effects of birth inputs on birthweight: Evidence from quantile estimation on panel data. *Journal of Business and Economic Statistics*, 26:379–397, 2008. [p189]

M. Alfó, N. Salvati, and M. G. Ranalli. Finite mixtures of quantiles and M-quantile models. *Statistics and Computing*, 27:547–570, 2017. [p188, 191]

M. Alfò, M. F. Marino, M. G. Ranalli, N. Salvati, and N. Tzavidis. M-quantile regression for multivariate longitudinal data with an application to the Millennium Cohort Study. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 70:122–146, 2021. [p208]

M. Baker. QREGPD: Stata Module to Perform Quantile Regression for Panel Data. Statistical Software Components, Boston College Department of Economics, Mar. 2016. [p189]

F. Bartolucci, A. Farcomeni, and F. Pennoni. *Latent Markov Models for Longitudinal Data*. Chapman & Hall/CRC, 2013. [p189]

F. Bartolucci, S. Pandolfi, and F. Pennoni. `LMest`: An `R` package for latent Markov models for longitudinal categorical data. *Journal of Statistical Software*, 81:1–38, 2017. [p189, 202]

D. Bates, M. Mächler, B. Bolker, and S. Walker. Fitting linear mixed-effects models using `lme4`. *Journal of Statistical Software*, 67:1–48, 2015. [p189]

L. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41:164–171, 1970. [p194]

M. Buchinsky. Estimating the asymptotic covariance matrix for quantile regression models. A Monte Carlo study. *Journal of Econometrics*, 68:303–338, 1995. [p194]

I. M. Danilevicz, V. A. Reisen, and P. Bondon. `pqrfe`*: Penalized Quantile Regression with Fixed Effects*, 2022. URL `https://CRAN.R-project.org/package=pqrfe`. `R` package version 1.1. [p189]

C. S. Davis. Semi-parametric and non-parametric methods for the analysis of repeated measurements with applications to clinical trials. *Statistics in Medicine*, 10:1959–1980, 1991. [p195]

A. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B: Statistical Methodology*, 39:1–38, 1977. [p188, 193]

P. J. Diggle, P. J. Heagerty, K. Liang, and S. L. Zeger. *Analysis of Longitudinal Data*, volume 25 of *Oxford Statistical Science Series*. Oxford University Press, second edition, 2002. [p188]

J. Einbeck, R. Darnell, and J. Hinde. `npmlreg`*: Nonparametric Maximum Likelihood Estimation for Random Effect Models*, 2018. URL `https://CRAN.R-project.org/package=npmlreg`. `R` package version 0.46-5. [p189]

A. Farcomeni. Quantile regression for longitudinal data based on latent Markov subject-specific parameters. *Statistics and Computing*, 22, 2012. [p188, 191]

G. M. Fitzmaurice, N. M. Laird, and J. H. Ware. *Applied Longitudinal Analysis*. John Wiley & Sons, 2012. [p188]

A. F. Galvao and L. Wang. Efficient minimum distance estimator for quantile regression fixed effects panel data. *Journal of Multivariate Analysis*, 133:1–26, 2015. [p189]

M. Geraci. Linear quantile mixed models: the `lqmm` package for Laplace quantile regression. *Journal of Statistical Software*, 57:1–29, 2014. [p189]

M. Geraci and M. Bottai. Quantile regression for longitudinal data using the asymmetric Laplace distribution. *Biostatistics*, 8:140–54, 2007. [p188, 195]

M. Geraci and M. Bottai. Linear quantile mixed models. *Statistics and Computing*, 24:461–479, 2014. [p188, 191]

B. S. Graham, J. Hahn, A. Poirier, and J. L. Powell. Quantile regression with panel data. Technical report, National Bureau of Economic Research, 2015. [p189]

L. Hao and D. Naiman. *Quantile Regression*. Sage, 2007. [p188]

S.-H. Jung. Quasi-likelihood for median regression models. *Journal of the American Statistical Association*, 91:251–257, 1996. [p195]

R. Koenker. Quantile regression for longitudinal data. *Journal of Multivariate Analysis*, 91: 74–89, 2004. [p189]

R. Koenker. *Quantile Regression*. Cambridge University Press, 2005. [p188]

R. Koenker. `quantreg`*: Quantile Regression*, 2022. URL https://CRAN.R-project.org/package=quantreg. `R` package version 5.94. [p189]

R. Koenker and S. H. Bache. `rqpd`*: Regression Quantiles for Panel Data*, 2014. URL https://R-Forge.R-project.org/projects/rqpd/. `R` package version 0.6/r10. [p189]

R. Koenker and G. Bassett, Jr. Regression quantiles. *Econometrica*, 46:33–50, 1978. [p188, 190]

R. Koenker and K. Hallock. Quantile regression. *Journal of Economic Perspectives*, 15:143–156, 2001. [p188]

S. Lahiri. Theoretical comparisons of block bootstrap methods. *The Annals of Statistics*, 27: 386–404, 1999. [p194]

N. Laird. Nonparametric maximum likelihood estimation of a mixing distribution. *Journal of the American Statistical Association*, 73:805–811, 1978. [p191]

B. G. Lindsay. The geometry of mixture likelihoods: a general theory. *The Annals of Statistics*, 11:86–94, 1983a. [p191]

B. G. Lindsay. The geometry of mixture likelihoods, part II: the Exponential Family. *The Annals of Statistics*, 11:783–792, 1983b. [p191]

R. J. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, 2002. [p190]

J. A. Machado and J. Santos Silva. Quantiles via moments. *Journal of Econometrics*, 213: 145–173, 2019. [p189]

J. A. Machado and J. Santos Silva. `xtqreg`: Stata module to compute quantile regression with fixed effects, 2021. URL https://EconPapers.repec.org/RePEc:boc:bocode:s458523. [p189]

J. A. Machado, P. Parente, and J. Santos Silva. `qreg2`: Stata module to perform quantile regression with robust and clustered standard errors, 2021. URL https://EconPapers.repec.org/RePEc:boc:bocode:s457369. [p189]

M. F. Marino and A. Farcomeni. Linear quantile regression models for longitudinal experiments: an overview. *METRON*, 73:229–247, 2015. [p188]

M. F. Marino, N. Tzavidis, and M. Alfò. Mixed hidden markov quantile regression models for longitudinal data with possibly incomplete sequences. *Statistical Methods in Medical Research*, 27:2231–2246, 2018. [p188, 192, 194]

M. F. Marino, M. Alfo', N. Salvati, and M. G. Ranalli. *lqmix: Linear Quantile Mixture Models*, 2025. URL https://CRAN.R-project.org/package=lqmix. R package version 1.2. [p188]

M. Pons and B. Melly. Stata Commands to Estimate Quantile Regression with Panel and Grouped Data. Swiss Stata Conference 2022 05, Stata Users Group, Nov. 2022. [p189]

R Core Team. R: *A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019. URL https://www.R-project.org. [p188]

D. B. Rubin. Inference and missing data. *Biometrika*, 63:581–592, 1976. [p190]

L. R. Welch. Hidden Markov models and the Baum-Welch algorithm. *IEEE Information Theory Society Newsletter*, 53:10–13, 2003. [p194]

K. Yu and R. A. Moyeed. Bayesian quantile regression. *Statistics and Probability Letters*, 54: 437–447, 2001. [p190]

K. Yu, Z. Lu, and J. Stander. Quantile regression: applications and current research areas. *Journal of the Royal Statistical Society D: the Statistician*, 52:331–350, 2003. [p188]

W. Zucchini and I. MacDonald. *Hidden Markov models for time series*, volume 110 of *Monographs on Statistics and Applied Probability*. CRC Press, 2009. [p189]

Marco Alfó
Department of Statistics, Sapienza, University of Rome
Piazzale A. Moro 5, 00185 Rome
Italy
(ORCiD 7651-6052)
marco.alfo@uniroma1.it

Maria Francesca Marino
Department of Statistics, Computer Science, Applications, University of Florence
Viale Morgagni 59, 50134 Florence
Italy
(ORCiD 5841-7613)
Corresponding author – mariafrancesca.marino@unifi.it

Maria Giovanna Ranalli
Department of Political Science, University of Perugia
Via Pascoli 25, 06123 Perugia
Italy
(ORCiD 7005-8572)
giovanna.ranalli@unipg.it

Nicola Salvati
Department of Economics, University of Pisa
Via Ridolfi 10, 56124 Pisa Italy
(ORCiD 4160-9257)
nicola.salvati@unipi.it

Exploring Image Analysis in R: Applications and Advancements

by Tim Brauckhoff, Julius Rublack, and Stefan Rödiger

Abstract This review offers an overview of image processing packages in R, covering applications such as multiplex imaging, cell tracking, and general-purpose tools. We found 38 R packages for image analysis, with adimpro and EBImage being the oldest, published in 2006, and biopixR among the newest, released in 2024. Of these packages, over 90 % are still active, with two-thirds receiving updates within the last 1.5 years. The pivotal role of bioimage informatics in life sciences is emphasized in this review, along with the ongoing advancements of R's functionality through novel code releases. It focuses on complete analysis pipelines for extracting valuable information from biological images and includes real-world examples. Demonstrating how researchers can use R to tackle new scientific challenges in image analysis, the review provides a comprehensive understanding of R's utility in this field.

1 Introduction

Advancements in microscopy and computational tools have become pivotal to biological research, facilitating detailed investigation of cellular and molecular processes previously inaccessible. Consequently, imaging methodologies, staining protocols, and fluorescent labeling — particularly those employing genetically encoded fluorescent proteins and immunofluorescence — have resulted in a substantial increase in the capacity to examine cellular structures, dynamics, and functions (Swedlow et al., 2009; Peng et al., 2012; Chessel, 2017; Moen et al., 2019; Schneider et al., 2019).

As with any significant advance in today's world, software is required to facilitate the acquisition, analysis, management, and visualization of image data resulting from these techniques. The current techniques have allowed the capture of biological phenomena with an unparalleled level of complexity and resolution (Eliceiri et al., 2012). As a result, an ever-growing amount of image data is being generated (Peng et al., 2012). Alongside the three spatial dimensions, images now encompass additional dimensions like time and color channels. Biomedical images exhibit this high level of complexity, as evidenced by the analysis of dense cell turfs where cells may partially overlap (Peng, 2008; Swedlow et al., 2009). The increase in complexity demands computational approaches. Nevertheless, the challenge posed is not solely due to complexity. As imaging technology advances, the volume of image data generated from experiments also sees a steep rise (Peng, 2008; Caicedo et al., 2017).

The need for quantitative information from images to understand and develop new biological concepts has led to the emergence of bioimage informatics as a specialized field of study (Eliceiri et al., 2012; Murphy, 2014). Bioimage informatics is primarily concerned with the extraction of quantitative information from images to interpret biological concepts or develop new ones (Chessel, 2017; Moen et al., 2019; Schneider et al., 2019). Bioimage informatics focuses on the automation of objective and reproducible image data analysis, while concurrently developing tools for the visualization, storage, processing, and analysis of such data (Swedlow and Eliceiri, 2009; Peng et al., 2012). Crucial advancements range from cell phenotype screening, drug discovery, and cancer diagnosis to gene function, metabolic pathways, and protein expression patterns. The basic operations in bioimage informatics are feature extraction and selection, segmentation, registration, clustering, classification, annotation, and visualization (Peng, 2008).

Due to recent advancements, the utilization of microscopy in biology has evolved into a quantitative approach, as opposed to solely a visual one. Thus, various essential open-source platforms, applications, and languages have emerged, which have now become

well-established within the life science community (Paul-Gilloteaux, 2023). Python, R, and MATLAB are among the most favored programming languages in bioinformatics (Giorgi et al., 2022), with Python and R being extensively used in biomedicine (Roesch et al., 2023). R plays a pivotal role in the fields of statistics, bioinformatics, and data science. It is a versatile statistical software that is used in various assays, for example, in gene expression analyses (Rödiger et al., 2013, 2015b; Burdukiewicz et al., 2022; Chilimoniuk et al., 2024). Furthermore, it is one of the top ten most prevalent programming languages across the globe, with a thriving community that has developed numerous extensions and packages for various applications (Giorgi et al., 2022). Originally developed for statistical analysis, R and its packages now offer robust capabilities for image analysis and automation (Chessel, 2017; Haase et al., 2022). The growing demand for automation and data-driven analysis underscores the necessity for flexible and integrated computational tools. R's expanding ecosystem of packages, ranging from general-purpose image processing to specialized, domain-specific workflows, facilitates the creation of customized solutions tailored to diverse research needs. The extensible framework and robust statistical capabilities support seamless integration of image analysis with downstream data interpretation, promoting reproducibility and efficiency across the entire analytical pipeline (Rödiger et al., 2015a; Chessel, 2017; Giorgi et al., 2022; Haase et al., 2022).

R can integrate with other programming languages through the use of packages such as `reticulate` (Ushey et al., 2024) for Python, which enables users to leverage the strengths of multiple languages within their research workflows, enhancing flexibility across diverse domains. Another example of this is Bio7. Bio7 is an open-source platform designed for ecological modeling, scientific image analysis, and statistical analysis. It provides an R development environment and integration with the ImageJ application (Austenfeld and Beyschlag, 2012). ImageJ is a widely-used, public-domain Java-based software suite specifically developed for biological image processing and analysis, that supports various file formats, advanced image manipulation techniques, and a vast array of plugins and scripts (Schneider et al., 2012).

A common difficulty in bioinformatics is the large number of file formats, some of which are proprietary. A lack of standardization means that general tools must deal with this vast array of file formats. The open-source approach provides access to the code of applications, packages, and extensions, thereby facilitating modification and further development by the community. This enhances reproducibility and validation, offering flexibility and adaptability for scientific discovery. This makes open-source methods ideally suited to the diverse and interdisciplinary field of biological imaging research (Swedlow and Eliceiri, 2009; Rödiger et al., 2015a). The Open Microscopy Environment (OME) offers a standardized, open-source framework for the management, analysis, and exchange of biological imaging data, with a particular focus on the integration and preservation of rich metadata — such as experimental conditions, cell types, acquisition parameters, microscope specifications, and quantification methods (Goldberg et al., 2005). A central objective of OME is to ensure lossless storage and interoperability across diverse proprietary and non-proprietary platforms. This objective addresses the common issue of metadata loss during format conversions within image analysis pipelines. By establishing standardized formats and protocols, OME fosters compatibility between proprietary systems and enhances reproducibility. The widely adopted OME-TIFF format extends the traditional TIFF structure by embedding metadata in XML, enabling efficient storage and retrieval of large, multidimensional datasets commonly encountered in fluorescence imaging (Linkert et al., 2010; Leigh et al., 2016; Besson et al., 2019). In addition, the OME-ZARR format, developed under the Next-Generation File Format (NGFF) initiative, has been optimized for scalable, cloud-based storage of large N-dimensional arrays, with metadata stored in human-readable JSON. The system's capacity for partial data access is a notable feature, contributing to enhanced performance in distributed workflows by combining formats such as OME-TIFF, Hierarchical Data Format 5 (HDF5), and Zarr (Moore et al., 2021, 2023)[1]. Increasing adoption of these formats by commercial imaging software vendors further strengthens their relevance and sustainability (Linkert et al., 2010). In the

[1] https://ngff.openmicroscopy.org/about/index.html, accessed 07/13/2025

context of R-based workflows, the RBioFormats package provides a native interface to the OME Bio-Formats Java library. This enables the reading of proprietary file formats and associated metadata, output to OME-TIFF, and seamless integration of image acquisition with downstream analysis (Andrzej Oleś, John Lee, 2023). This facilitates the establishment of flexible, standardized, and reproducible image analysis pipelines within the R ecosystem.

The heterogeneous and dynamic nature of images presents a constant challenge for image analysis. Capturing precise and high-quality images that accurately represent the changing characteristics of an experiment can be difficult, even for experienced researchers (Swedlow et al., 2009). Additionally, visualizing and analyzing multi-gigabyte data sets requires substantial computational power. The process of detailed analysis of image sequences, which involves identifying and tracking objects, followed by the presentation of the resulting data and the exploration of the underlying biological mechanisms, adds further complexity (Swedlow and Eliceiri, 2009). To at least simplify the process of selecting the appropriate software, this review provides an overview of R packages suitable for image analysis and outlines their applications in biological laboratory settings.

2 Methods

In this study, a review of the literature was conducted over the period September 2023 to March 2024. The objective was to identify and analyze R packages that are suitable for bioimage informatics applications. The primary resources included the Comprehensive R Archive Network (CRAN)[2], GitHub repositories[3], rOpenSci's r-universe[4], the Bioconductor repository[5], OpenAlex database, PubMed, and Google Scholar. The chosen sources allowed for an extensive coverage of R package repositories while also providing access to relevant scientific literature. By combining these resources, the study aimed to provide a comprehensive overview of available tools and techniques within the domain of bioimage informatics using R.

The search strategy centered around pertinent keywords, including "bioimage," "biomedical image analysis," "imaging," "microscopy," "histology," and "pathology" and the following search strings:

- https://openalex.org/works?page=1&filter=title_and_abstract.search%3AR%20packages%20for%20image%20analysis
- https://openalex.org/works?page=1&filter=title_and_abstract.search%3Aimage%20processing%20in%20R
- https://openalex.org/works?page=1&filter=title_and_abstract.search%3Amicroscopy%20imaging%20in%20R%20packages
- https://pubmed.ncbi.nlm.nih.gov/?term=biomedical+image+analysis&filter=dates.1963%2F1%2F1-2025%2F3%2F26&filter=pubt.review&filter=other.excludepreprints
- https://scholar.google.de/scholar?hl=de&as_sdt=0%2C5&q=image+analysis+in+R&btnG=
- https://scholar.google.de/scholar?hl=de&as_sdt=0%2C5&q=bioimage+analysis+in+R&btnG=
- https://scholar.google.de/scholar?hl=de&as_sdt=0%2C5&q=microscopy+imaging+analysis+in+R&btnG=

[2] https://cran.r-project.org/, accessed 04/17/2025
[3] https://github.com/, accessed 04/17/2025
[4] https://ropensci.r-universe.dev/builds, accessed 04/17/2025
[5] https://www.bioconductor.org/, accessed 04/17/2025

The identified packages were then subjected to an analysis to understand their usage, dependencies on other libraries, repository hosting platforms, and licensing terms.

The examples provided, along with this review, were created using `RMarkdown`. All computations were performed using the `R` programming language, version 4.3.3, on a 64-bit x86_64-pc-linux-gnu platform with the Ubuntu 22.04.3 LTS operating system. We utilized the RStudio Integrated Development Environment (IDE, 2023.09.0+463 "Desert Sunflower", Ubuntu Jammy).

This review will examine a variety of `R` packages designed for image analysis, including both general-purpose tools and those crafted for specific applications. This overview aims to demonstrate the diverse capabilities and adaptability of these tools within and beyond biological research contexts. Given the significant interest in the localization of microplastics in cells and the environment, our examples will primarily focus on the analysis of microbead particles made of polymethylmethacrylate (PMMA), which measure approximately 12 µm and fall within the microplastic size range (Geithe et al., 2024). As microbeads are round, spherical objects in images, they visually resemble other commonly imaged objects such as seeds and cells.

3 Dividing to conquer - advanced segmentation strategies

Image segmentation is a crucial preliminary step in image analysis and interpretation. It involves dividing an image into distinct regions by assigning a label to each pixel. The primary objective is to delineate regions pertinent to the specific task (Peng, 2008; Ghosh et al., 2019; Niedballa et al., 2022a). This process frequently employs features such as pixel intensity, gradient magnitude, or texture measures. Based on these features, segmentation techniques can be classified into three categories: region-based, edge-based, or classification-based. Classification-based methods assign class labels to pixels based on their feature values, whereas region-based and edge-based techniques focus on within-region homogeneity and between-region contrast. One straightforward method of segmentation is thresholding, which involves comparing pixel values against one or more intensity thresholds. This process typically separates the image into foreground and background regions (Sonka and Fitzpatrick, 2000; Jähne, 2002).

Another image segmentation method was proposed by Ren and Malik (2003). This approach integrates a preprocessing step that segments the image into superpixels, feature extraction based on Gestalt cues, evaluation of the extracted features, and the training of a linear classifier. Superpixels are clusters of pixels that are similar with respect to properties such as color and texture, resulting in larger subregions of the image. The primary objective of this preprocessing step is to simplify the image and reduce the number of regions considered for segmentation. Previously, this involved evaluating every single pixel. The division of the image into regions larger than pixels but smaller than objects allows for the superpixels to encompass a greater quantity of information, adhere to the boundaries of natural image objects, reduce the presence of noise and outliers, and enhance the speed of the subsequent segmentation process. In summary, this method can be described as segmentation based on low-level pixel grouping (Ren and Malik, 2003; Hossain and Chen, 2019; Mouselimis et al., 2023).

However, segmentation is not limited to the differentiation of the foreground and background. Pixel classification plays a critical role in a number of applications, including visual question answering, object counting, and tracking. In these applications, classification occurs not just spatially but also temporally. These applications are diverse, encompassing fields such as traffic analysis and surveillance, medical imaging, and cell biology (Ghosh et al., 2019). While a relatively straightforward technique, thresholding has inherent limitations in distinguishing between background, noise, and foreground. Therefore, the next section will offer a more sophisticated approach, by presenting a package that utilizes deep learning for image segmentation (Smith et al., 2021).

3.1 imageseg: a deep learning package for forest structure analysis

By venturing beyond the traditional laboratory setting, the imageseg package offers a unique approach to analyzing forest structures through deep learning-based image segmentation, utilizing TensorFlow (https://www.tensorflow.org/). This R package employs the power of convolutional neural networks with the U-Net architecture to streamline image segmentation tasks (Niedballa et al., 2022a). According to the authors, this R package has been designed to be user-friendly, with pre-trained models that require only input images, making it accessible even to those without specialist knowledge. A comprehensive vignette accompanies the package, which provides detailed instructions on how to set up the software and explains how to utilize its functions effectively (Niedballa et al., 2022c). Developed primarily for forestry and ecology applications, imageseg includes pre-trained data sets representing various aspects of forest structure, such as canopy and understory vegetation density. Its flexibility allows for customization with different training data, enabling users to develop customized image segmentation workflows for other fields such as microscopy and cell biology. The package supports both binary and multiclass segmentation. For image processing within the R programming environment, the imageseg package integrates with the magick package (Niedballa et al., 2022a).

3.2 EBImage: specialized segmentation strategy for touching objects

The segmentation of closely adjacent objects, which is particularly prevalent in cell microscopy, represents a common challenge that is addressed by the EBImage package, which is equipped with a variety of segmentation algorithms. A typical approach involves the application of either global or adaptive thresholding, followed by connected set labeling, with the objective of distinguishing individual objects. To achieve more precise segmentation of touching objects, techniques such as watershed transformation or Voronoi segmentation are employed (Pau et al., 2010).

The watershed algorithm is employed to delineate touching microbeads (Figure 1A-C). Initially, the image is transformed into a binary image by applying a threshold (Figure 1B). After utilizing the watershed function the result is visualized by assigning distinct colors to the microbeads, effectively illustrating the algorithm's capacity to differentiate between touching objects (Figure 1C).

```
# Load necessary library
library(EBImage)

# Load the image from the specified path
image <- readImage("figures/beads.png")

# Display the original image
EBImage::display(image)

# Apply a threshold to the original image to create a binary image
img_thresh <- thresh(image, offset = 0.05)

# Read the binary image and display it
EBImage::display(img_thresh)

# Perform watershed segmentation on the distance map of the thresholded image
segmented <- EBImage::watershed(distmap(img_thresh))

# Color the labels of the segmented image
segmented_col <- colorLabels(segmented)

# Display the resulting image after watershed segmentation
```

```
EBImage::display(segmented_col)
```

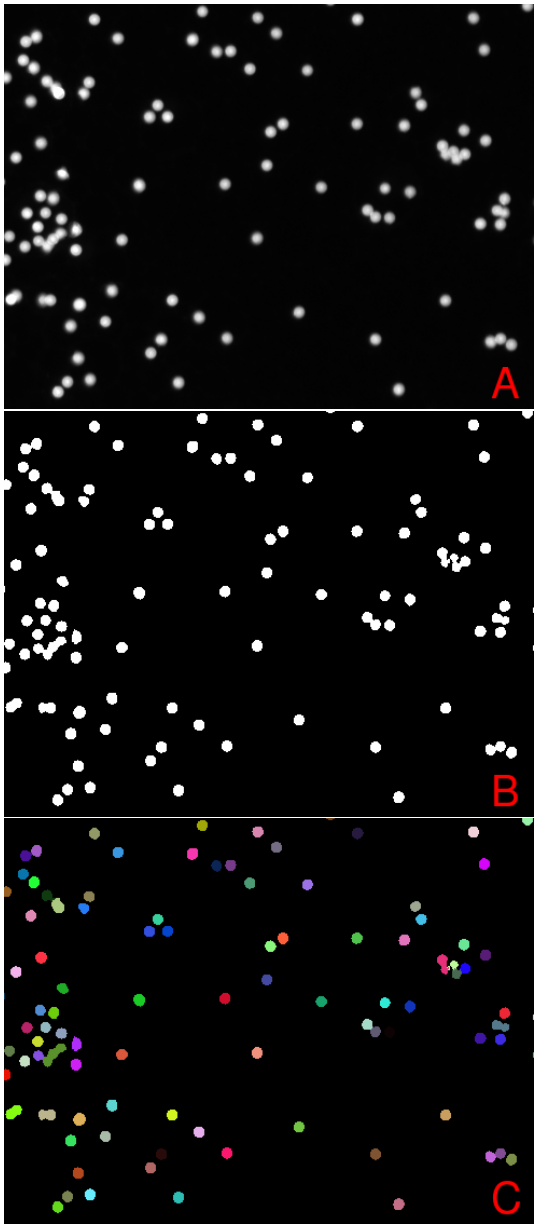


Figure 1: Watershed Segmentation in `EBImage`: **A**) Original image used for watershed segmentation in `EBImage`. **B**) The `thresh()` function was employed to generate a binary image with the objective of effectively separating the foreground from the background. The binary representation of the image facilitates further segmentation processes by simplifying the image. **C**) Presents the result of the watershed segmentation, which is visually represented by the assignment of a distinct color to each object. This technique is particularly effective in differentiating touching objects, as evidenced by the clear separation of microbeads in the image.

4 Unveiling the hidden - feature extraction

The primary objective of feature extraction is to condense the original data into significant objects that encapsulate crucial information pertinent to each specific image (Jude Hemanth and Anitha, 2012). Feature extraction may be applied to a predefined region of interest (ROI) or may involve the identification of the ROI, a process often referred to as segmentation, which was reviewed in the previous sections. Within any given ROI, a multitude of attributes typically exist, representing different states of the object under analysis. These attributes,

or features, are of vital importance for the interpretation of the detected objects and can enable applications such as disease diagnosis or the identification of promising candidates. Features related to individual pixels may include aspects such as neighborhood relationships, connectivity, and gradients, which are one-dimensional descriptions. Nevertheless, more intelligible and interpretable information is frequently derived from descriptions of regions or objects (Sonka and Fitzpatrick, 2000; Shirazi et al., 2018). Object-level features encompass a range of characteristics, including size, shape, texture, intensity, and spatial distribution. Shape features can be further categorized into specific characteristics, including perimeter, radius, circularity, and area. It is crucial to acknowledge that the successful extraction of object features is dependent on the quality and accuracy of the image segmentation process (Shirazi et al., 2018).

This section is devoted to an examination of `R` packages that enable the automated extraction of quantitative features. The `biopixR` package offers automated and interactive object detection strategies. The `pliman` package, initially developed for the analysis of plant images, has the potential to be adaptable to a range of different domains. The `FIELDimageR` package is capable of supporting the analysis of drone-captured images from agricultural field trials as well as images from pollen, which exhibit similar characteristics to cellular images. These tools provide novel perspectives for interdisciplinary research, facilitating the adaptation of methodologies across diverse fields.

4.1 `biopixR`: versatile biological image processing

The `biopixR` package is a comprehensive toolbox developed primarily for microbead analysis. It encompasses a range of functions, including image importation, preprocessing, segmentation, feature extraction, and clustering. The primary objective is to enable the detection of objects and the extraction of quantitative data, including intensity values, shape, and texture characteristics. These functionalities are integrated into user-friendly pipelines that support batch processing, thereby enhancing accessibility. The preprocessing capabilities include edge restoration and a variety of filter functions (Brauckhoff et al., 2024).

To illustrate the feature extraction process, the analysis focuses on a microbead image (Figure 2A). The image is initially converted to grayscale. Afterwards the `objectDetection()` function is applied to detect image objects. The extracted objects are then represented visually by plotting the highlighted contours of the objects and enumerating the microbeads according to their cluster IDs, thus distinguishing them as individual entities (Figure 2B).

```
# Loading necessary package
library(biopixR)

# Importing the image
beads <- importImage("figures/beads2.jpg")

# Plot original image
beads |> plot(axes = FALSE)

# Converting the image to grayscale
beads <- grayscale(beads)

# Detecting objects in the image using edge detection
objects <-
  objectDetection(beads,                # Image to process
                  method = 'edge',      # Method for object detection
                  alpha = 1,            # Threshold adjustment factor
                  sigma = 0)            # Smoothing factor

# Displaying internal visualization of object detection with marked contours
# and centers
```

```
objects$marked_objects |> plot(axes = FALSE)

# Adding text annotations at the centers of detected objects
text(objects$centers$mx,     # x-coordinates of object centers
     objects$centers$my,     # y-coordinates of object centers
     objects$centers$value,  # Text to display (value of the object center)
     col = "green",          # Color of the text
     cex = 1.5)
```

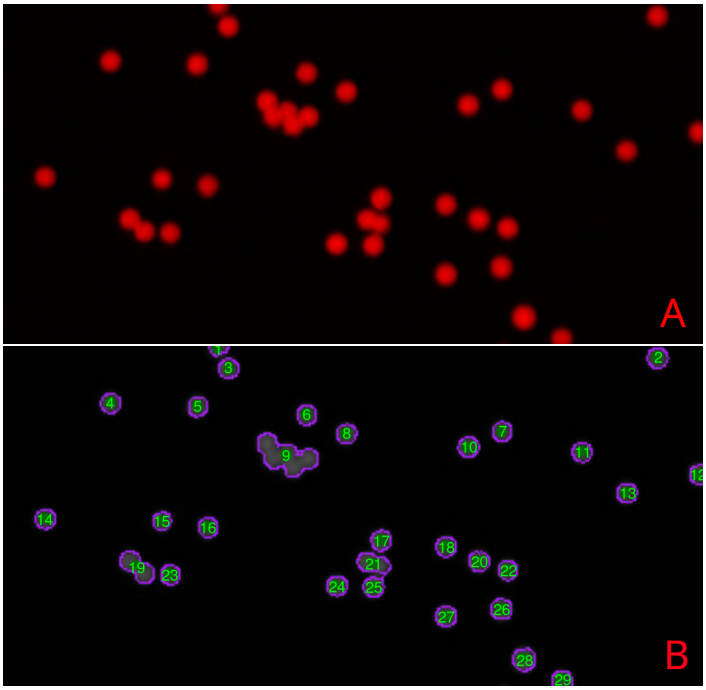


Figure 2: Microbead Detection using `biopixR`: **A**) The original image shows red fluorescent microbeads, with the majority appearing as isolated, round, spherical objects. Some microbeads are clustered together or overlapping, forming aggregated structures, while others are partially captured within the image frame. **B**) In the grayscale microbead image, edges of the microbeads are highlighted in purple, and the labeling ID (value) is displayed at the center of each object in green.

4.2 `pliman`: an R package for plant image analysis

`pliman` is designed to analyze plant images, particularly leaves and seeds, to help identify disease states, lesion shapes, and quantify objects. It supports various functions, including image transformation, binarization, segmentation, and detailed analysis, all facilitated by a detailed vignette.[6] A key feature of `pliman` is its automation of quantitative feature extraction (Figure 3 and 4), which traditionally requires manual, time-consuming, and error-prone methods. The features of this package are versatile, encompassing a range of segmentation strategies, the analysis of shape and contour characteristics of leaves and seeds, the counting of objects, and the quantification of disease states from leaf images. While the primary focus is on plant imaging, the techniques used are applicable to other fields such as cellular imaging. This cross-applicability is further emphasized by the package's batch processing capabilities, which allow for autonomous analysis of multiple images, critical for high-throughput phenotyping tasks (Olivoto, 2022).

```
# Loading necessary package
library(pliman)
```

[6] https://tiagoolivoto.github.io/pliman/index.html, accessed 07/11/2024

```
# Import requires EBImage:
# Importing the main image
beads <- EBImage::readImage("figures/beads2.jpg")

# Importing additional images for background and foreground
foreground <- EBImage::readImage("figures/foreground.jpg")
background <- EBImage::readImage("figures/background.jpg")

# Displaying the microbead image
EBImage::display(beads)

# Combining the foreground and background images and arranging them in 2 rows
pliman::image_combine(foreground, background, nrow = 2, col = "transparent")
```

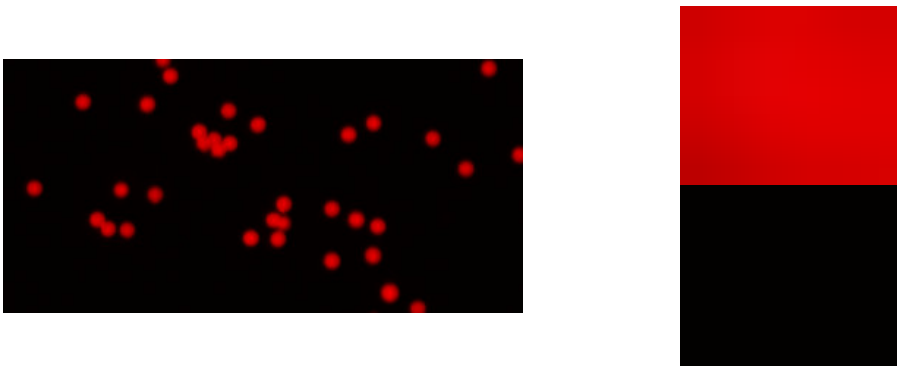


Figure 3: Preparing Segmentation using `pliman`: The image comprises two sections. On the left, an image of microbeads is displayed. On the right, a cropped view from the same image illustrates two states for segmentation: the microbead (foreground) in red, and the background is shown in black, emphasizing the clear division needed for segmentation analysis.

```
# Performing segmentation based on provided background and foreground images
analyze_objects(
  img = beads,               # Main image of microbeads
  background = background,   # Background sample image
  foreground = foreground,   # Foreground sample image
  marker = "id",             # Displaying enumeration
  contour_col = "yellow"     # Color for the contour of the segmented objects
)
```

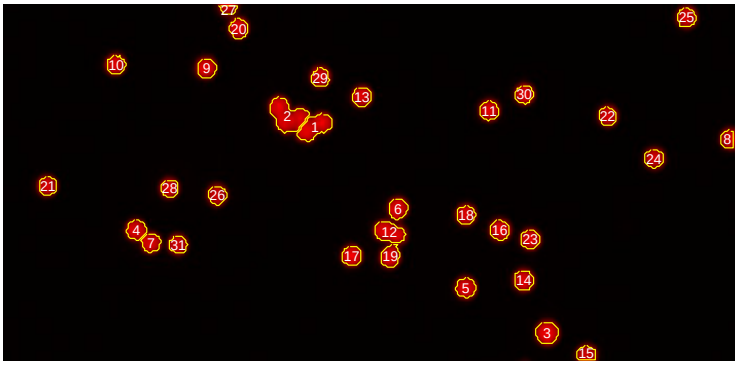


Figure 4: Segmentation Results using `pliman`: The image depicts the segmentation results obtained via the `pliman` `analyze_objects()` function. It displays the contours of the segmented objects, outlined in yellow. Each distinct object within the segmentation is numbered, facilitating its identification.

4.3 `FIELDimageR`: an R package for the analysis of drone-captured images

The `FIELDimageR` package, is an `R` package designed for the specific purpose of analyzing drone-captured images from agricultural field trials. The package offers a variety of functions for ROI selection, the extraction of foregrounds (Figure 5), watershed segmentation, quantification and shape analysis (Matias et al., 2020). The developers have applied this package to analyze pollen, which visually resembles cells under a microscope. This suggests that `FIELDimageR` may be applicable for use in microbiological image analysis. For the spatial analysis, the package utilizes the `terra` package (Matias et al., 2020).[7]

To showcase the functionalities of the `FIELDimageR` package and its parallels with biological applications, the same microbead image is subjected to analysis. The image is initially transformed into a 'SpatRaster' object and then segmented using an intensity threshold (Figure 5). The microbeads are correctly identified as the foreground objects by the `fieldMask()` function. Subsequently, a distinct labeling ID is assigned to each microbead, as illustrated by a color gradient. Moreover, the contours of each individual object are displayed (Figure 6). The results of the segmentation and the extraction of shape-related information are presented in the interactive `leaflet` interface (Figure 7). Presenting information like cluster ID, size, perimeter and width of the detected objects.

```
# Loading necessary packages
library(FIELDimageR)
library(FIELDimageR.Extra)
library(terra)
library(sf)
library(leafsync)
library(mapview)

# Using the same image as imported in the previous example
# Creating a SpatRaster object using the 'terra' package
EX.P <- rast("figures/beads2.jpg")
EX.P <- imgLAB(EX.P)

#> [1] "3 layers available"

# Removing background based on a vegetation index
```

[7] https://github.com/OpenDroneMap/FIELDimageR, accessed 05/07/2024

```
EX.P.R1 <-
  fieldMask(
    mosaic = EX.P,     # Input SpatRaster object
    index = "BIM",     # Index representing vegetation
    cropValue = 5,     # Threshold value for the index
    cropAbove = F      # Indicates to remove values below the threshold
  )

# Displaying the original, background, and foreground images
EX.P.R1$newMosaic
```

Figure 5: Displaying the original, background, and foreground Images: The original image (left) shows the fluorescent microbeads. The middle image displays the background in white (TRUE) and all objects detected by segmentation in black (FALSE). The right image shows only the foreground (microbeads) after detection through segmentation using the `fieldMASK()` function.

```
# Labeling of all microbeads
EX.P.Total <- fieldCount(mosaic = EX.P.R1$mask, plot = T)
```

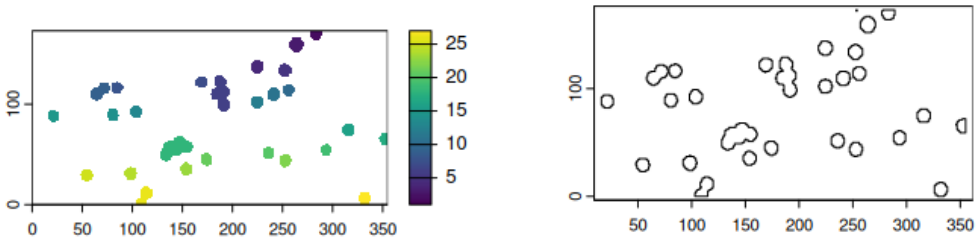


Figure 6: Labeling of Microbeads: The `fieldCount()` function is used to label individual microbeads. This function utilizes the mask produced in the previous section to identify the objects. The left image displays the labeling with a color gradient indicating distinct objects. On the right, the object contours are shown. The output of the function includes more than just the labeling value (named ID in this package); it also provides information on area, perimeter, width, and geometry of the detected objects.

```
# Combining the 'FIELDimageR.Extra', 'mapview' and 'leafsync' to create an
# interactive view
m1 <- fieldView(EX.P, r = 1, g = 2, b = 3)
m2 <- mapview(EX.P.Total)
sync(m1, m2)
```

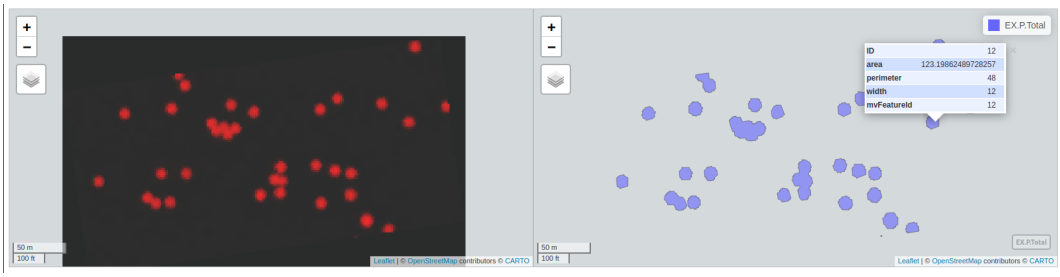


Figure 7: Displaying Results with an Interactive `leaflet` Tool: The tool displays the original image on the left. For comparison, the cursor is mirrored to the corresponding image (only visible in HTML format). The left image provides detailed information interactively. Hovering over the objects reveals their labeling ID. Performing a left-click opens a detailed window providing information for the individual object, such as area, perimeter, width, and shape. The packages `FIELDimageR.Extra`, `mapview`, and `leafsync` are used to create the interactive display.

In summary, packages such as `EBImage` and `biopixR` provide direct pipelines for the extraction of features from images, including shape, size, radius, and perimeter, as well as texture information through the calculation of Haralick texture features (Haralick et al., 1973; Pau et al., 2010; Brauckhoff et al., 2024). The `biopixR` package employs the `imager` and `magick` packages for image processing (Brauckhoff et al., 2024), whereas `pliman` and `FIELDimageR` rely on `EBImage` for direct image analysis, with `FIELDimageR` also utilizing `terra` and `raster` for spatial data exploration (Matias et al., 2020; Olivoto, 2022). In comparison to the other packages discussed in this section, `biopixR` facilitates the process of object detection by eliminating the necessity for the generation of masks or the provision of representative sample images of the foreground and background. Nevertheless, in contrast to the other packages, `biopixR` lacks the functionality of watershed segmentation for the enhanced handling of touching objects (Figure 2B and Figure 4) (Matias et al., 2020; Olivoto, 2022; Brauckhoff et al., 2024).

5 Decoding complexity - clustering, classification and annotation

The automation of measuring cellular phenomena and the effects of compounds, which started in the late 1990s, is now increasingly significant owing to the progress of machine learning (ML) algorithms and computing power. These advancements are enhancing the field of bioinformatics' accessibility to these techniques. Consequently, they are being more commonly employed with the aim of gaining novel biological insights (Murphy, 2014; Moen et al., 2019; Weiss et al., 2022). One of the latest methods of image analysis involves comparing the morphological characteristics of cells from captured images with pre-classified training data that represent a specific state (Moen et al., 2019). Bioimage informatics methods aim to generate fully automated models for biological systems (Murphy, 2014).

A major challenge in handling new data sets is the need to label images, which is critical to assigning meaning to the objects within them. This is particularly important in medical imaging, where expert knowledge is essential for accurate labeling (Boom et al., 2012; Weiss et al., 2022). In ML, two common techniques that can be used to categorize data into distinct groups are clustering and classification. Clustering, an unsupervised learning method, is used to discover underlying structures or patterns in unlabeled data by assessing similarities between data points (Mostafa and Amano, 2019). Classification, a form of supervised learning, involves building a model from previously labeled training data to make predictions about new data (Mostafa and Amano, 2019; Kumar Dubey et al., 2022). This requires prior labeling of the data to determine the characteristics of each group, a process known as annotation. However, manual annotation is time-consuming and labor-intensive, requiring significant human effort to identify relevant details in an image (Yao et al., 2016; Weiss et al., 2022). Because images often require multi-label annotation - the assignment of multiple semantic concepts to a single image - there has been a growing

demand for automated image annotation systems that aim to reduce the burden of manual labeling and increase the efficiency of data processing (Nasierding et al., 2009).

To effectively analyze complex image data sets, researchers require advanced pattern recognition techniques that can extract meaningful biological insights from these images. This enables them to transform visual data into actionable scientific knowledge (Behura, 2021). Some of the most widely used clustering algorithms for this purpose include:

- k-means: is a centroid-based algorithm that partitions n observations into k clusters by minimizing within-cluster sum of squares. It does require specifying the number of clusters beforehand (Struyf et al., 1996).
- Partitioning Around Medoids (PAM): a k-means relative, seeks to identify k representative objects from the data set, which are robust representations of the clusters' center and are also referred to as medoids. Clusters are formed by assigning each object to its nearest medoid, with the objective of optimizing within-cluster similarity (Kaufman and Rousseeuw, 1990; Van der Laan et al., 2003).
- c-means: also known as Fuzzy C-Means (FCM), extends the concept of k-means to allow each data point to belong to more than one cluster (Bezdek et al., 1984).
- Density-Based Spatial Clustering of Applications with Noise (DBSCAN): is a density-based clustering algorithm that groups together points that are closely packed together and separates them from points that lie alone in low-density regions. It does not require specifying the number of clusters beforehand (Ester et al., 1996; Schubert et al., 2017).
- Self-Organizing Maps (SOM): are a type of neural network architecture that systematically organizes input features into a spatially coherent representation. This method can be utilized for clustering based on various object features, thereby facilitating the discovery of patterns within these objects (Kohonen, 1990, 2013).

5.1 `pixelclasser`: a simplified support vector machine approach for pixel classification

The `pixelclasser` package is a tool for classifying image pixels into user-defined color categories using a simplified version of the Support Vector Machine (SVM) technique. It includes functions that allow users to visualize image pixels, define classification rules, classify pixels, and store the resulting information.[8] Users must provide a test set that captures the variation between categories, as the package requires manual placement of rules for each category - automatic rule construction methods are not included. In addition, `pixelclasser` provides quality control of the classifications and comes with a detailed vignette to facilitate the use of this classification tool.[9] The classification on the pixel-level can be used for image segmentation via pixel clustering.

5.2 `biopixR`: pattern recognition of shape- and texture-related features

The `biopixR` package incorporates two unsupervised ML clustering algorithms: SOM and PAM. PAM organizes a distance matrix into clusters, identifying medoids as robust representatives of each cluster, typically specified with a predefined number of groups (k) (Kaufman and Rousseeuw, 1990; Van der Laan et al., 2003; Park and Jun, 2009). This approach clusters Haralick texture features extracted from multiple images within a directory, thereby enabling image classification based on these features (Haralick et al., 1973). The optimal number of clusters (k) is automatically determined using silhouette analysis (Rousseeuw, 1987; Brauckhoff et al., 2024). SOM is used to cluster object features related to object shape and intensity, thereby facilitating the identification of patterns within these characteristics (Brauckhoff et al., 2024).

The capacity for pattern recognition within the `biopixR` package is demonstrated by the clustering of shape-related and pixel-intensity information from an example image of

[8] https://github.com/ropensci/pixelclasser, accessed 07/11/2024

[9] https://cloud.r-project.org/web/packages/pixelclasser/vignettes/pixelclasser.html, accessed 07/11/2024

microbeads (Figure 8A). The image depicts both single and aggregated microbeads, wherein the former exhibit a round, spherical shape, while the latter appear more oval. The extracted features and the corresponding cluster are depicted in Figure 8B, which showcases the identification of patterns within these objects based on their shape characteristics.

```
# Load the 'biopixR' package
library(biopixR)

# Import an image from the specified path
img <- importImage("figures/beads.png")

# Set seed for reproducibility
set.seed(123)

# Extract shape features from the image
result <- shapeFeatures(
  img,
  alpha = 0.8,
  sigma = 0.7,
  xdim = 2,
  ydim = 1,
  SOM = TRUE,
  visualize = FALSE
)

# Define colors for plotting points based on classes
colors <- c("darkgreen", "darkred")

# Plot the image without axes and add colored points representing the classes
img |> plot(axes = FALSE)
with(result,
     points(
       result$x,
       result$y,
       col = colors[factor(result$class)],
       pch = 19,
       cex = 1.2
     ))
text(c(471), c(354), c("A"), col = "darkred", cex = 5)

# Create a data frame with various shape features and the pixel-intensity
df <- data.frame(
  size = result$size,
  intensity = result$intensity,
  perimeter = result$perimeter,
  circularity = result$circularity,
  eccentricity = result$eccentricity,
  radius = result$mean_radius,
  aspectRatio = result$aspect_ratio
)

# Min-Max Normalization Function
min_max_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}
```

```
# Applying the function to each column
df_normalized <- as.data.frame(lapply(df, min_max_norm))

# Create a boxplot of the normalized data
boxplot(
  df_normalized,
  ylab = "normalized values",
  xaxt = "n",
  cex.lab = 1.25,
  cex.axis = 1.25
)

# Add axis ticks and diagonal labels
axis(1, at = 1:ncol(df), labels = FALSE)  # Add axis ticks but no labels
text(
  cex = 1.2,
  x = seq_len(ncol(df_normalized)),
  y = -0.07,
  labels = colnames(df_normalized),
  adj = 0,
  srt = -45,
  xpd = TRUE
)

# Highlight specific rows based on class
highlight_rows <-
  which(result$class == 2)  # Example row indices to highlight

# Add points for the specific rows
# Adding points for each column
for (col in 1:ncol(df_normalized)) {
  points(
    rep(col, length(highlight_rows)),
    df_normalized[highlight_rows, col],
    col = "red",
    pch = 19,
    cex = 1.5
  )
}

text(c(0.5),
     c(0.98),
     c("B"),
     col = "darkred",
     cex = 5)
```

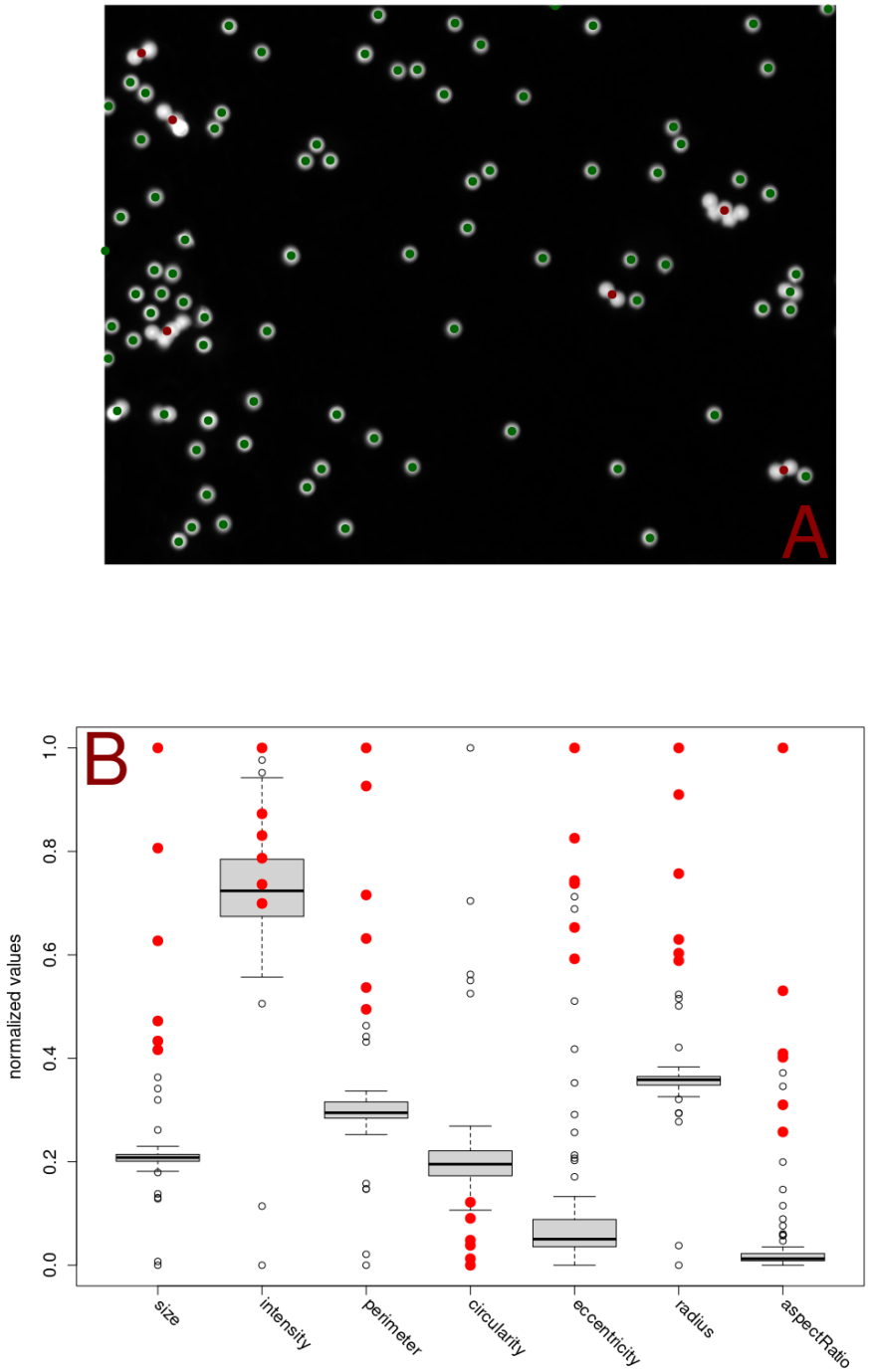


Figure 8: Clustering Microbeads Based on Shape and Intensity Features: **A**) The utilization of Self-Organizing Maps (SOM) enables the clustering of microbeads into two distinct groups based on shape and intensity features extracted using the `shapeFeatures()` function. This method enables the precise clustering of microbeads according to a range of properties, including intensity, area, perimeter, circularity, radius, and aspect ratio. This facilitates a deeper understanding of the morphological variations observed in the microbeads. **B**) The attributes utilized as input for the SOM algorithm are illustrated in this plot. To ensure comparability, the different parameters have been normalized using a min-max normalization procedure. The points highlighted in red represent the microbeads that are also highlighted in red in Figure A. Notably, these highlighted points differ from the most commonly occurring values in all attributes except for the intensity.

6 Harmonizing visions - techniques and approaches in image registration

The process of image registration plays a pivotal role in the analysis of medical images, as it enables the comparison of multiple images representing different conditions (Jenkinson and Smith, 2001). This process, which can be described as image alignment, entails aligning a series of images within a single coordinate system, thereby ensuring consistency across images (Peng, 2008; Rittscher, 2010). A variety of techniques are employed in image registration, including mutual information registration, spline-based elastic registration, and invariant moment feature-based registration, among others (Peng, 2008). These methods are of particular significance in the field of medical imaging, where they are employed to enhance the analysis of images obtained by techniques such as computed tomography (CT) and magnetic resonance imaging (MRI) (Sonka and Fitzpatrick, 2000).

6.1 `RNiftyReg`: interface for the 'NiftyReg' image registration tools

The `RNiftyReg` package provides an interface to the 'NiftyReg' image registration library, which supports both linear and non-linear registration in two and three dimensions (Clayden et al., 2023). This package has been utilized in research on brain connectivity (Clayden et al., 2013), and it includes a comprehensive README that introduces its features and capabilities.[10]

7 Jack of all trades - general purpose `R` packages for broad-spectrum analysis

Five principal image processing packages for `R` offer a broad range of algorithms and capabilities for complete image analysis, rendering them suitable as general-purpose tools. These packages are `imager`, `magick`, `EBImage`, `OpenImageR` and `SimpleITK`. This section will introduce each of these key packages and their roles in image analysis.

7.1 `imager`: wrapper for the 'CImg' C++ image processing library

The `imager` `R` package, created by Barthelmé and Tschumperlé (2019), integrates the functionality of the 'CImg' library, developed by David Tschumperlé, into `R`.[11] This allows users to edit and create images. The package uses two primary data structures: raster images, known as *cimg*, and pixel sets, referred to as *pixelset*. These structures, encoded as four-dimensional numeric or logical arrays, permit the execution of basic `R` functions such as `plot()`, `print()`, or `as.data.frame()`, as well as the processing of hyperspectral images and videos (Barthelmé and Tschumperlé, 2019). The 4D arrays encompass two spatial dimensions (width and height), one temporal or depth dimension, and one color dimension (Barthelme et al., 2024). `imager` offers over 100 standard commands for tasks such as loading, saving, resizing, and denoising of images.[12] The `imager` package supports the file formats JPEG, PNG, and BMP and is available on CRAN (Barthelme et al., 2024).

7.2 `EBImage`: image processing and analysis for biological imaging data in `R`

The `EBImage` package, established in 2006, is one of the oldest image processing tools available in `R` and can be accessed via the Bioconductor repository. It is primarily written in `R` and C/C++ (Andrzej Oleś, 2017). `EBImage` provides a suite of general tools for image processing and analysis, particularly excelling in microscopy-based cell assays. It features specialized commands for cell segmentation and the extraction of quantitative data from images (Pau et al., 2010). The package employs the RGB color system for color detection,

[10] https://github.com/jonclayden/RNiftyReg, accessed 07/11/2024
[11] https://github.com/asgr/imager, accessed 07/11/2024
[12] https://asgr.github.io/imager/, accessed 07/11/2024

which is based on pixel intensities. The incorporation of the EBImage package into the R workflow facilitates the automation and objectivity of the image analysis procedure (Heineck et al., 2019). Images in EBImage are managed as an extension of R's base array, specifically the package-specific Image class. As images are treated as multidimensional arrays, algebraic operations are possible. This class structure includes various slots, with the *.data* slot holding the numeric pixel intensity array and the *colorMode* slot managing the image's color information. Adjusting the *colorMode* setting changes the image's rendering mode (Andrzej Oleś, 2017; Heineck et al., 2019). Typically, the first two dimensions of an image carry spatial information, while additional dimensions are variable and can represent color channels, time points, replicas, or depth. EBImage also features an interactive display interface through GTK+, and offers a set of functions for automated image-based phenotyping in biology, including cell segmentation, feature extraction, statistical analysis, and visualization (Pau et al., 2010). It supports a range of file formats, including JPEG, PNG, and TIFF, and can handle additional formats through integration with the 'ImageMagick' image-processing library (Pau et al., 2010; Andrzej Oleś, 2017).

7.3 magick: advanced image processing in R using 'ImageMagick'

This package is built upon 'Magick++', the C++ API for the 'ImageMagick' image processing library.[13] The R package provides access to 'ImageMagick' functionalities, enabling both basic and complex image manipulations directly in R. Notably, images in magick are automatically displayed in the RStudio console, creating a dynamic and interactive editing environment. The wide variety of functions made available through this package are impressive. The possibilities range from functions that are rather 'just for fun', such as implosion or introduction of noise, to more advanced processing techniques, including different segmentation techniques, edge detection, and a toolbox for morphology operations. The magick package is compatible with a diverse range of image formats and encompasses the functionalities required for format conversion. This includes the conversion to the formats supported by the EBImage package. It also handles multiple frames, facilitating the creation and processing of animated graphics. Each operation in magick creates a new, altered version of the image, preserving the original (Ooms, 2024a).[14] Recent developments include the introduction of a shiny application that enables users to interactively perform basic image processing tasks such as blurring and edge detection.[15] The magick package is compatible with a range of popular file formats, including PNG, BMP, TIFF, PDF, SVG, and JPEG, and is available through the CRAN repository (Ooms, 2024a).[16]

7.4 OpenImageR: a general-purpose image processing library

OpenImageR is a lesser known but highly versatile general-purpose image processing library that integrates both the R and C++ programming languages. This package offers a comprehensive array of functions for preprocessing, filtering, and feature extraction. Images are treated as two- or three-dimensional objects, represented by matrices, data frames, or arrays, with the third dimension representing color information. The functionalities within OpenImageR are organized into three main categories: basic functions, which include importing, displaying, cropping, and thresholding; filter functions, which feature augmentation and various edge detection algorithms; and image recognition, which incorporates functions from the 'ImageHash' Python library. In recent updates, a number of new features have been incorporated, including Gabor feature extraction, which was originally developed in MATLAB and based on code by Haghighat et al. (2015). The most recent version incorporates image segmentation techniques that utilize superpixels and clustering. Images can be visualized through the shiny application or the grid package. OpenImageR is capable of

[13] https://imagemagick.org/script/magick++.php, accessed 07/11/2024
[14] https://www.imagemagick.org/Magick++/ImageDesign.html, accessed 07/11/2024
[15] https://georgestagg.github.io/shinymagick/, accessed 07/11/2024
[16] https://imagemagick.org/, accessed 07/11/2024

handling a multitude of image formats, including PNG, TIFF, and JPG (Mouselimis et al., 2023).[17] [18]

7.5 `SimpleITK`: a streamlined wrapper for ITK in biomedical image analysis

The following section will introduce a prominent tool in biomedical image analysis, the wrapper for the Insight Segmentation and Registration Toolkit (ITK), known as `SimpleITK` (Rittscher, 2010). `SimpleITK` represents a streamlined version of the original ITK, an open-source C++ library that features a wide array of imaging algorithms and frameworks (Lowekamp et al., 2013; Yaniv et al., 2017). This library has been in development for approximately two decades and is particularly favored in the medical image analysis community (Lowekamp et al., 2013; Beare et al., 2018). The objective of `SimpleITK` is to simplify the accessibility of ITK algorithms by reducing their complexity, thereby making these sophisticated tools more approachable for a broader audience (Lowekamp et al., 2013). Adapted for the `R` programming language through SWIG, `SimpleITK` offers over 250 image processing algorithms that function across various scripting and prototyping environments (Lowekamp et al., 2013; Yaniv et al., 2017; Beare et al., 2018). In contrast to other general-purpose image processing packages, which treat images as mere arrays, `SimpleITK` treats images as objects within a physical space, thereby providing a set of metadata about image and voxel geometry in world coordinates (Lowekamp et al., 2013; Yaniv et al., 2017; Beare et al., 2018). This nuanced representation is of particular importance for specific medical imaging applications. Additionally, `SimpleITK` incorporates metadata such as the origin, pixel spacing, and a matrix defining the physical orientation of image axes (Yaniv et al., 2017). However, the complexity of the underlying ITK library may impede customization and necessitate familiarity with C++. Another challenge for `R` developers arises from the fact that the documentation is also based on C++ (Beare et al., 2018). To facilitate the learning process, Yaniv et al. (2017) has developed a series of Jupyter notebooks that provide an introduction to the package and its capabilities for both Python and `R` users. These notebooks serve as educational tools and a resource for research, providing full coverage of the entire spectrum of image analysis processes (Beare et al., 2018).[19] In combination with `R`, `SimpleITK` enables detailed image processing and facilitates the subsequent statistical evaluation of quantified data. The software is compatible with a range of digital image formats, including JPEG, BMP, PNG, and TIFF, and is capable of analyzing 2D and 3D images (Beare et al., 2018). The package is obtained through the GitHub repository.[20]

In summary, these packages and their associated libraries offer a vast array of algorithms that can be accessed in `R`. This includes features from the 'CImg', 'ImageMagick' and ITK libraries, along with the diverse algorithms encoded in the `EBImage` package. These flexible packages provide the foundation for the development of numerous tailored applications.

8 Exploring the facets of complexity - multiplexed imaging in `R`

Multiplexed imaging is a crucial technology for analyzing complex biological processes at the single-cell level, especially in tissue-based cancers and autoimmune diseases (Harris et al., 2022a). This technique enables the simultaneous assessment of multiple protein and DNA molecules, overcoming limitations that hinder advancements in understanding biological interactions and phenomena (Gerdes et al., 2013; Goltsev et al., 2018). Multiplex imaging is the result of a multiplex experiment, in which multiple species (Aherne et al., 2024), biomolecules (Damond et al., 2019), or cell types (Creed et al., 2021) are labeled with different probes, dyes, or antibodies simultaneously. This technique allows for the differentiation of components within the resulting image (Eling et al., 2020). In comparison to standard immunofluorescence experiments, the number of distinct targets is significantly

[17] https://github.com/mlampros/OpenImageR, accessed 07/11/2024
[18] https://mlampros.github.io/OpenImageR/index.html, accessed 07/11/2024
[19] https://github.com/InsightSoftwareConsortium/SimpleITK-Notebooks, accessed 07/11/2024
[20] https://github.com/SimpleITK/SimpleITKRInstaller, accessed 07/11/2024

increased, reaching up to 50 different target molecules (Damond et al., 2019; Einhaus et al., 2023). This can be used to distinguish between species in a biofilm (Aherne et al., 2024), or to obtain an overview of the biomarker distribution or tissue composition in a sample (Damond et al., 2019; Yang et al., 2020). The technique has the capacity to reveal the positions and interactions of individual cells, provide insight into the activities of biomolecules, and holds the potential for the reconstruction of the three-dimensional tissue architecture of a given sample (Harris et al., 2022b; Cho et al., 2023; Zhao and Germain, 2023). Several imaging techniques are used to obtain detailed insights into the spatial interactions between cells, including Co-Detection by indEXing (CODEX) (Goltsev et al., 2018), Multiplex Ion Beam Imaging (MIBI) (Angelo et al., 2014), and Multiplexed Immunofluorescence Imaging (MxIF) (Gerdes et al., 2013; Harris et al., 2022a; Feng et al., 2023). These methods generate vast amounts of imaging data, often terabytes across hundreds of slides, which necessitates sophisticated image analysis pipelines (Harris et al., 2022b).

8.1 `mxnorm`: normalize multiplexed imaging data

Managing technical variability within these pipelines is crucial, and intensity normalization is one approach to address this issue (Harris et al., 2022b). The `R` package `mxnorm` addresses this by providing tools for implementing, evaluating, and visualizing various normalization techniques (Harris, 2023). These tools aid in measuring technical variability and evaluating the efficacy of various normalization methods. They enable users to apply customized methods to improve image consistency by reducing technical variations while preserving biological signals. `mxnorm` provides an analysis pipeline for multiplex images, incorporating normalization algorithms inspired by the ComBat paper, the `fda` package, and the tidyverse framework (Harris et al., 2022a). For researchers who want to effectively standardize multiplexed imaging data, these features make `mxnorm` a powerful resource (Harris, 2023).

8.2 `DIMPLE`: manipulation and exploration of multiplex images

To assess patient outcomes, understand disease mechanisms, and develop effective cancer therapies, the `DIMPLE` `R` package is designed to extract critical information from the tumor microenvironment (TME). `DIMPLE` facilitates quantification and visualization of cellular interactions within the TME using spatial data. It also enables correlation of these interactions and phenotypic data with patient outcomes through sophisticated statistical modeling. `DIMPLE` provides researchers with an extensive toolkit to analyze cellular interactions and transform raw multiplex imaging data into actionable biological insights, potentially identifying prognostic indicators for cancer research and therapy development. To support the analysis process, a `shiny` application is provided (Masotti et al., 2023).[21]

8.3 `cytomapper`: visualization of multiplex images and cell-level information

The `cytomapper` package is designed to visualize multiplexed read-outs and cell-level information obtained by multiplex imaging technologies (Nils Eling, Nicolas Damond, Tobias Hoch, 2020). It offers various functions to view pixel-level information across multiple channels and display expression data for individual cells. Additionally, `cytomapper` includes features to gate cells based on their expression values, enhancing the analysis of complex data sets. It is compatible with data from various multiplex imaging technologies and requires single-cell read-outs, multi-channel TIFF stacks, and segmentation masks. The `cytomapper` package is a versatile tool for researchers working with advanced imaging data sets to explore cellular behaviors and properties (Eling et al., 2020).

[21] https://github.com/nateosher/DIMPLE, accessed 07/11/2024

8.4 SPIAT: analyzing spatial properties of tissues

The SPIAT package, standing for **Sp**atial **I**mage **A**nalysis of **T**issues, is among the most comprehensive tools for multiplex image analysis (Trigos et al., 2022). Developed with compatibility for multiplex imaging technologies like CODEX and MIBI, SPIAT facilitates the analysis of spatial data by using X and Y coordinates of cells, their marker intensities, and phenotypes. It features six analysis modules that support a variety of functions including visualization, cell co-localization, distance measurements between cell types, categorization of the immune microenvironment in relation to tumor areas, analysis of cellular neighborhoods and clusters, and quantification of spatial heterogeneity (Yang et al., 2020; Trigos et al., 2022). To use SPIAT, images must be pre-segmented and cells phenotyped, typically using external software like HALO and InForm to prepare the correct input format (Yang et al., 2020). The package provides a shiny application that assists the user in formatting spatial data from the aforementioned sources in a manner that ensures compatibility with the functions of the SPIAT package.[22] SPIAT is designed to be user-friendly, making complex spatial analysis accessible to researchers with varying computational skills (Feng et al., 2023).

8.5 Seurat: spatially resolved transcriptomics (SRT)

Spatially resolved transcriptomics (SRT) is a commonly used approach for the quantification of gene expression levels in tissue sections while preserving positional information (Larsson et al., 2023). The Seurat package (Hao et al., 2024) is a package for spatial transcriptomics and multiplexed imaging analysis. It shares some similarities with the SPIAT and spatialTIME packages. For assays with cell segmentation, Seurat facilitates the visualization of individual cell boundaries or centroids, thereby enabling more precise mapping of molecular signals to cells. In contrast to other reviewed packages, Seurat's unique feature is its integration of spatial and molecular data for spatial data analysis. In particular, it enables the joint analysis of spatially-resolved gene expression data alongside traditional single-cell RNA-seq, allowing researchers to map cell types and states within their native tissue context, along with metadata. Notably, Seurat supports the analysis and visualization of spatial omics data at both single-cell and subcellular resolution. Seurat deliberately supports a broad range of spatial technologies, including the Akoya CODEX/Phenocycler™ platform and sequencing-based platforms such as Visium Spatial Gene Expression, 10x Genomics and Slide-seq. To achieve these capabilities, Seurat offers statistical methods to identify genes or features with spatially structured expression patterns, which facilitate the uncovering of region-specific biological processes. Since its first publication in 2015 (Satija et al., 2015), its functionality has expanded to include support for image-based spatial transcriptomics (highly multiplexed imaging technologies). Seurat uses image data (e.g., raw, masked, processed images, 10X Genomics Visium Image).

8.6 spatialTIME: spatial analysis of Vectra immunofluorescence data

The spatialTIME package has been designed for the analysis of immunofluorescence data with the objective of identifying spatial patterns within the TME. The package appears to be designed to work with data acquired by the Vectra Polaris™ imaging system.[23] It facilitates the spatial analysis of multiplex immunofluorescence data, enabling spatial characterization and architectural reconstruction. Additionally, the package includes a shiny application, iTIME, which offers a user-friendly point-and-click interface that mirrors many of the capabilities found in spatialTIME (Creed et al., 2021).[24] The package also comes with a detailed vignette to help users get started with its features (Creed et al., 2024).

[22] https://github.com/TrigosTeam/SPIAT-shiny, accessed 07/11/2024

[23] https://web.archive.org/web/20250125194642/https://www.akoyabio.com/wp-content/uploads/2021/11/Vectra_Polaris_Product_Note_with_MOTiF_Akoya.pdf, accessed 07/14/2025

[24] https://fridleylab.shinyapps.io/iTIME/, accessed 07/11/2024

In summary, R offers a range of tools for analyzing multiplex imaging data. However, it is important to note that these packages, except for the `cytomapper` package, require image preprocessing and use the resulting data frames as input for analysis.

9 Tracing the dance - R packages for analyzing cellular movement dynamics

Cellular migration is essential for various physiological and pathological functions, including development, immune responses, wound healing, and tumor progression (Bise et al., 2011; Yamada and Sixt, 2019; Hossian and Mattheolabakis, 2020), making it a crucial field in disciplines such as neuroscience, oncology, and regenerative medicine (Kaiser and Bruinink, 2004; Hu et al., 2023). To gain insight into these biological processes, researchers can track cell movement by manually tracing their positions in sequential images for 2D coordinates or by incorporating the z coordinate for 3D analysis (Hu et al., 2023). By studying cell migration at multiple levels - from the molecular components and the behavior of individual cells to the dynamics of cell populations - researchers can unravel the complex interactions that influence the movement of cells (Maheshwari and Lauffenburger, 1998). Such wide studies are crucial in advancing our understanding of phenomena such as cancer metastasis, which could lead to new therapeutic strategies (Um et al., 2017).

9.1 `celltrackR`: analyzing motion in two or three dimensions

The `celltrackR` package is intended for analyzing motion in two or three dimensions, primarily using data from time-lapse microscopy or x-y-(z) coordinates. It is useful in both biological settings for tracking cells and in non-biological contexts for object tracking (Textor et al., 2024). Additionally, the package provides a web user interface to facilitate the analysis process.[25] The package contains standard analytical tools, such as mean square displacement and autocorrelation, as well as algorithms for simulating artificial tracks using various models, such as Brownian motion and the Beauchemin model of lymphocyte migration (Textor et al., 2024). Furthermore, `celltrackR` provides a complete pipeline for track analysis, including data management, quality control, and methods for detecting tracking errors, such as track interpolation and drift correction (Wortel et al., 2021). The package is well-documented, providing detailed vignettes that guide users through the migration analysis process (Textor et al., 2024).

10 Mapping the unseen - exploring spatial properties in bioimage data

In this section, we explore the use of R tools for analyzing spatial properties in applications such as transcriptomics. One notable package is the `MoleculeExperiment` package (noa, 2024), which can be used to analyze molecular data within image-based data sets. This package builds upon other popular packages like `EBImage`, focusing on raster analysis, and `terra` (Hijmans, 2024) for handling geographic information systems (GIS) tasks. Raster or gridded data are spatial data structures that divide regions into rectangles called cells or pixels, storing one or more values. These grids contrast with vector data representing points, lines, and polygons in GIS contexts. Each pixel represents an area on a surface, making color image rasters unique due to their multiple bands containing reflectance values for specific colors or light spectra.

The `terra` package (formerly known as `raster/sp`) offers fast operations through optimized back-end C++ code. Users can perform various raster tasks such as creating objects, executing spatial/geometric functions like re-projections and resampling, filtering, and conducting calculations. Functions within the package facilitate extracting essential statistics from entire SpatRaster data sets, including mean values, maximum values, value ranges, or

[25] https://github.com/ingewortel/celltrackR, accessed 07/11/2024

counts of NA cells. In addition to these analytical capabilities, `terra` provides functionality for visualizing data and interacting with rasters, enhancing user experience when working with gridded spatial information. This versatility makes the package an essential tool in analyzing transcriptomic data within image-based data sets using R tools (Hijmans, 2020).

11 Numbers game - simplifying scientific image data representation

The R environment offers multiple additional tools for the extraction of information from data, with a particular focus on the extraction of measuring points in scientific diagrams. This task is of particular significance when data is available exclusively in image format, for instance from publications or other sources.

11.1 `digitize`: use data from published plots or images

The `digitize` package is a well-established and mature tool that simplifies importing data from digital images by providing a user-friendly interface for calibration and point location. It leverages the `readbitmap` package to read various bitmap formats such as BMP, JPEG, PNG, and TIFF. When reading these image files, digitize relies on the magic number embedded within each file rather than solely relying on the file extension. For seamless integration with JPEG and PNG images, this package depends on external libraries like 'libjpg' and 'libpng' (Poisot, 2011). Interestingly, the packages can be used for other purposes as well. For example, Figure 9 demonstrates that the `digitize` package can quantify certain structures in images. This example illustrates how fluorescent objects in an image can be identified by their position and subsequently quantified by their number.

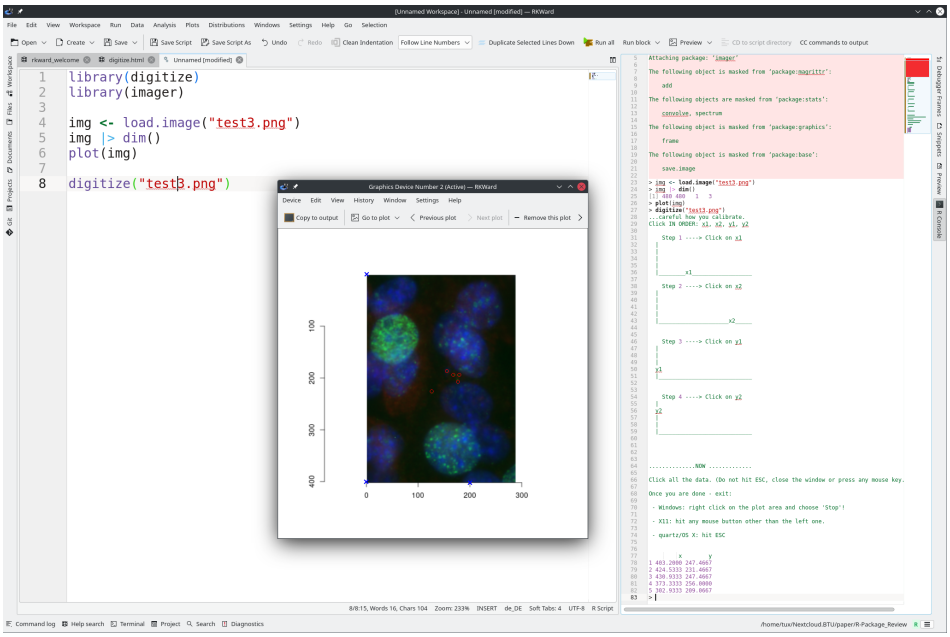


Figure 9: Counting using `digitize`: The figure provided to `digitize`, consists of cells with DNA damage (similar to Rödiger et al. (2018)). The nucleus is colored with DAPI (blue) and the γH2AX histone, a marker for DNA double strand breaks, is stained with a specific antibody. The `digitize` package is used to interactively extract the coordinates (shown in the console) by using the cursor to define the region of interest (blue cross) and tag the objects within it (red circles). In the screenshot it is displayed how `digitize` is invoked in RKWard (0.7.5z+0.7.6+devel3, Linux, TUXEDO OS 2, (Rödiger et al., 2012)).

11.2 `juicr`: extraction of numerical data from scientific images

`juicr` is a tool designed to automate the extraction of numerical data from scientific images. It offers users a Tcl/Tk graphical user interface (GUI) that simplifies point-and-click manual extraction with advanced features such as image zooming, calibration capabilities, and classification options. Additionally, `juicr` provides semi-automated tools for fine-tuning extraction attempts. To ensure optimal performance, this package depends on the `EBImage` package, which must be installed and loaded prior to utilization. Once data is extracted using `juicr`, users can choose to save their results in various formats including comma-separated values (CSV) files or postscript (EPS) files for easy import into other software. Moreover, extractions can also be saved as fully-embedded and standalone HTML files, that preserve all extraction details, setup configurations, and image modifications. These HTML files provide a means of storing data while ensuring long-term accessibility and replicability for future reference and analysis purposes (Lajeunesse, 2021).

11.3 `image2data`: transforming images into data sets

In recent years, the conversion of images into data sets has emerged as an essential tool in various fields such as computer vision, healthcare, and geospatial analysis. The `image2data` R package provides functionality to convert images into data sets (Caron and Dufresne, 2022). The primary function `image2data()` takes an image file with extensions like .png, .tiff, .jpeg or .bmp as input and converts it into a data set. Each row of the resulting data set represents a pixel (or subject), while columns represent variables such as x-coordinate, y-coordinate, and hex color code. The `image2data()` function offers methods for reducing data sets, yielding results akin to pixelated images with adjustable precision values. Higher precision leads to more data points, while lower precision yields fewer. This example showcases a pixelated representation of a pixel-based image in PNG format, highlighting its unique visual attributes. Users have the ability to customize and modify various elements by adjusting their corresponding hex color codes for precise control over hues, saturation levels, and brightness.

```
# Loading the required packages
library(image2data)
library(data.table)

# Path to the image file
image <- "figures/test3.png"
img <- EBImage::readImage(image)

# Subsampling the image data
beads_subsample <- image2data(
  path = image,                    # Path to the image file
  reduce = .2,                     # Reduction factor for subsampling
                                   # (20 % of original number of pixels)
  seed = 42,                       # Seed for random number generation by
                                   # return (for reproducibility)
  showplot = FALSE               # Whether to show a plot of the subsampled data
) |> as.data.table()               # Converting the result to a data.table

# Display a part of the subsampled data
beads_subsample
```

```
#>                x          y       g
#>            <num>      <num>  <char>
#>     1:  0.1022393 -0.9263444 #2F5C61
#>     2: -0.1022393  0.4006978 #121D11
```

```
#>     3:  1.2449136 -0.5213380 #121B10
#>     4:  0.4871401 -1.6588028 #151E1C
#>     5: -0.3548305 -1.5381626 #0D1B0D
#>    ---
#> 23151: -1.1486884  1.1159219 #352B5E
#> 23152: -0.6074216  0.1508003 #252E60
#> 23153:  1.4975048  0.5988925 #14180B
#> 23154: -1.3651952  0.2025032 #2A306B
#> 23155:  0.3428023 -0.3231434 #112048

EBImage::display(img)

# Plotting the subsampled data
plot(beads_subsample$x,            # x-coordinates
     beads_subsample$y,            # y-coordinates
    col = beads_subsample$g,    # Color based on hex code extracted by image2data()
     pch = 19,                     # Plotting character (solid circle)
     xlab = "",
     ylab = "")
```

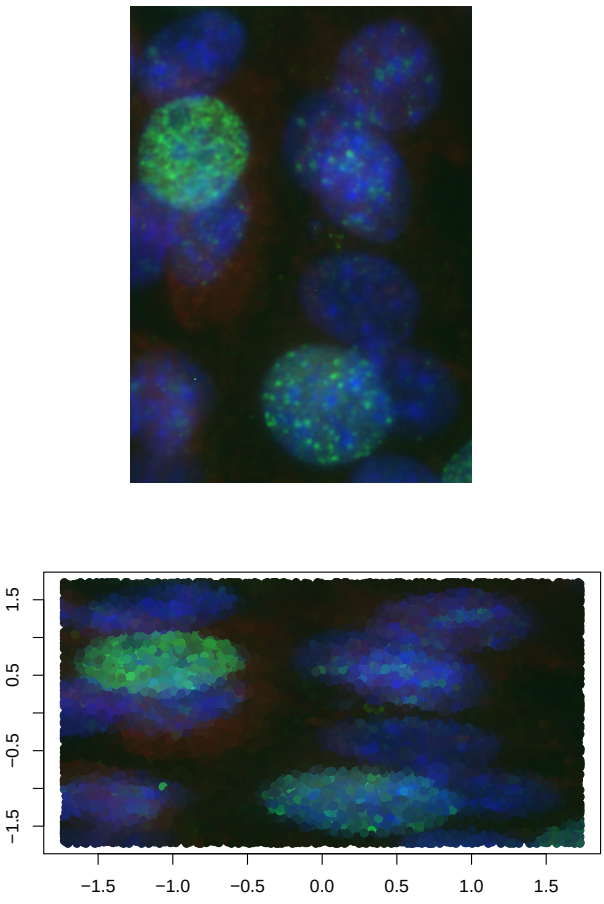


Figure 10: Application Example of the `image2data` Package: The image displays nuclei stained with DAPI (blue) and a quantitative marker for DNA double strand breaks, was labeled with a specific antibody (green). The `image2data` package extracted 20% of the pixels from the original image (top), creating a table with x | y coordinates and corresponding hex color codes. This data was then used to reassemble the image using R's base plot (bottom).

12 Engaging insights - interactive approaches to image analysis

The analysis and processing of images to extract useful information can be a challenging endeavor. Consequently, the implementation of interactive approaches accompanied by immediate visual feedback regarding parameter alterations represents a significant aid in simplifying image analysis. Therefore, this section will focus on interactive tools and functions from packages that facilitate the exploration of images and the extraction of useful insights.

12.1 `cytomapper`: a shiny application for hierarchical gating and visualization of multiplex images

The `cytomapper` package, designed for processing multiplex images, includes a `shiny` application that facilitates the hierarchical gating of cells using specific markers and allows for the visualization of selected cells. The graphical user interface (GUI) of this `shiny` application is designed to assist in the process of cell labeling. Furthermore, the data from the selected cells can be saved as a *SingleCellExperiment*, thereby enabling various downstream processing methods (Eling et al., 2020; Nils Eling, Nicolas Damond, Tobias Hoch, 2020). The `cytomapper` package offers comparable functionality for feature extraction as described in the beginning, providing an algorithm for extracting morphological and intensity features from multiplex images (Nils Eling, Nicolas Damond, Tobias Hoch, 2020).

12.2 `colocr`: interactive ROI selection in image analysis through shiny app

The `colocr` package, which facilitates the exploration of fluorescent microscopic images, features a GUI accessible through a `shiny` app. This GUI can be invoked locally or accessed online. The process of image analysis frequently necessitates the input of manual labor, particularly in the selection of ROIs. This package streamlines the process of selecting ROIs by semi-automating it, thereby allowing users to review and interactively select one or more ROIs. Moreover, the app offers the option to interactively adjust parameters such as threshold, tolerance, denoising, and hole filling, thereby enhancing user control and precision in image analysis by providing immediate feedback (Ahmed et al., 2019; Ahmed, 2020).[26]

[26] https://mahshaaban.shinyapps.io/colocr_app2/, accessed 07/11/2024

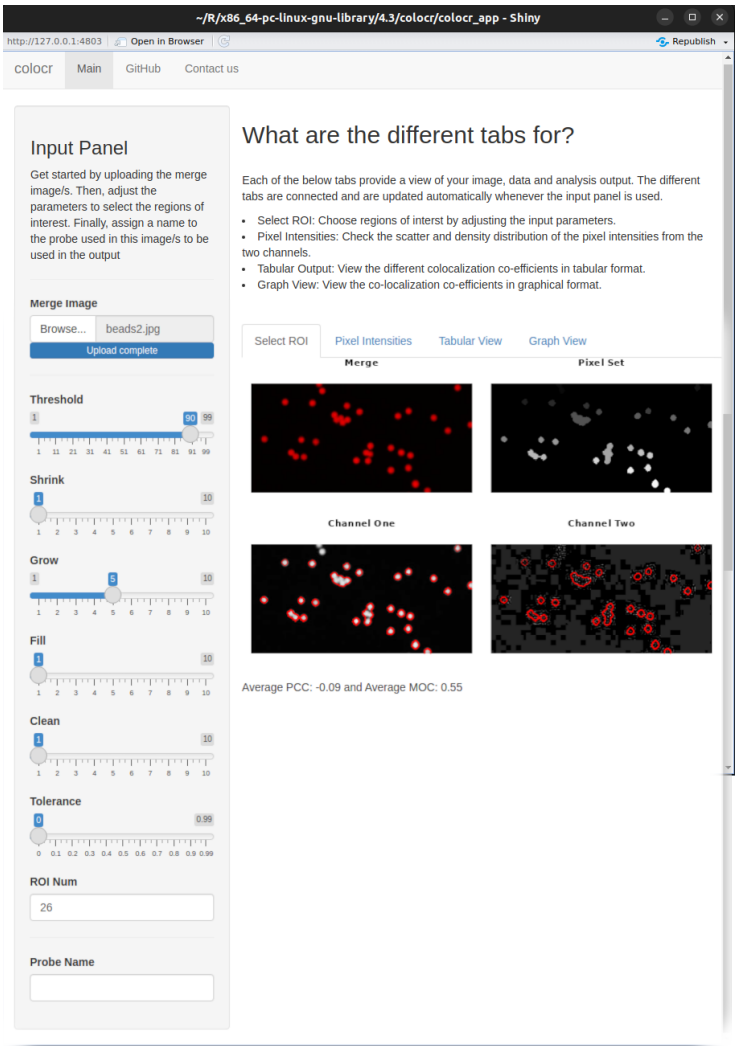


Figure 11: Shiny Application of the `colocr` Package: The figure depicts an interactive image analysis graphical user interface (GUI), invoked locally from the RStudio integrated development environment (IDE). It comprises multiple sliders for real-time parameter adjustments and supports the selection of multiple distinct regions of interest (ROIs). Users can interactively select ROIs and extract characteristics such as pixel intensity. Furthermore, the tool offers functionalities to compute co-localization, providing comprehensive analysis capabilities. Available at: https://mahshaaban.shinyapps.io/colocr_app2/ or run: `colocr::colocr_app()`.

12.3 `magick`: shiny and Tcl/Tk tools for interactive image exploration

A basic demo version of an interactive web interface for the `magick` R package is available via a `shiny` app. While it remains a demonstration version and does not encompass all the functionalities of the full package, it is not suitable for in-depth analysis of large-scale imaging data. In contrast, the app provides fundamental tools for image processing, including blurring, imploding, rotating, and more. This tool is designed to facilitate basic image processing tasks in an interactive environment.[27] Additionally, a distinct package is available that provides the functionality of `magick` in an interactive manner. This package, called `magickGUI`, was developed by Ochi (2023). The interactive features are based on the Tcl/Tk wrapper for R and include functions for thresholding, edge detection, noise reduction, and many more.

[27] https://github.com/jeroen/shinymagick, accessed 07/11/2024

12.4 `biopixR`: interactive Tcl/Tk function for feature extraction

In the `biopixR` package, the `tcltk` package — which enables Tcl/Tk integration in `R` — was employed to create an interactive function. This function initiates the launch of a GUI that streamlines the process of feature extraction by facilitating object detection and enabling users to select between edge detection and thresholding for segmentation. The GUI displays the currently detected edges (when using edge detector) or all detected coordinates (when using threshold) and the object centers within an image. The application includes sliders that allow users to adjust parameters and magnify the image. This interactive function is designed to facilitate the parameter selection process, as the chosen parameters affect the quality of image segmentation (Brauckhoff et al., 2024).

13 Tailored tools - specialized `R` packages for image processing

In contrast to the previously mentioned general-purpose tools, some packages have been designed with a specific focus on particular research areas. These specialized tools address the unique challenges encountered in those fields and offer versatile solutions for analyzing the data collected in those domains. While a complete survey of the available packages is outside the scope of this article, a concise overview of the most pertinent packages and their applications will be presented.

13.1 `fslr`: analysis of neuroimage data

The `fslr` package serves as a wrapper for the FSL software, enabling the use of the 'FMRIB' Software Library within the `R` environment. The FSL software is a widely utilized tool for the analysis and processing of neuroimaging data, including MRI. The package employs the use of `NIfTI` images to facilitate the execution of processing tasks, thereby introducing capabilities such as brain extraction and tissue segmentation, which were previously unavailable in `R` (Muschelli et al., 2015; Muschelli, 2022).

13.2 `colocr`: co-localization analysis of fluorescence microscopy images

A common application derived from fluorescence microscopy, which is extensively utilized in biological research, is co-localization analysis. This analysis assesses the distribution of signals across different color channels to determine whether the positioning of objects is correlated (Dunn et al., 2011; Ahmed et al., 2019). The objective of this software is to streamline the analysis process by providing tools for loading images, selecting regions of interest, and calculating co-localization statistics (Ahmed et al., 2019; Ahmed, 2020). It incorporates methods outlined by Dunn et al. (2011).[28]

CRAN offers a list of packages tailored to medical image analysis, accompanied by detailed descriptions of their applications. This list can be accessed via the following URL:

`https://cran.r-project.org/web/views/MedicalImaging.html`

Moreover, the Bioconductor repository contains a number of packages focused on single-cell analysis, as detailed by Amezquita et al. (2019). The Bioconductor project is an initiative dedicated to the collaborative development and the use of scalable software for computational biology and bioinformatics. Its objective is to reduce the entry barriers to interdisciplinary research and to improve the remote reproducibility of scientific findings (Gentleman et al., 2004). Other packages identified during the course of our research, though not explored in depth, are acknowledged in the forthcoming summary:

[28] `https://github.com/ropensci/colocr`, accessed 07/11/2024

Table 1: Overview of R packages for tailored applications in image processing. This table summarizes key aspects such as general application, repository (Repo) hosting (CRAN, Bioconductor (Bioc), GitLab), linked libraries, and package dependencies. It also includes information on licensing and current status. The current status is divided into the date of first publication on the corresponding repository (*). Active repository status is indicated by a circle, with the date of the latest update (°). Some packages that are no longer maintained are marked as archived (†).

	Application	Repo	based on	License	Status
adimpro by Polzehl and Tabelow (2007)	Adaptive Smoothing	CRAN	Image Magick	GPL ($\geq$ 2)	*2006-10-27 °2023-09-06
phenopix by Filippa et al. (2016)	Vegetation phenology	CRAN	jpeg	GPL-2	*2017-06-16 °2024-01-19
gitter by Wagih and Parts (2014)	Pinned Microbial Cultures	CRAN-archived	EBImage	LGPL	*2013-06-29 †2020-01-16
TCIApathfinder by Russell et al. (2018)	Cancer Imaging	CRAN	Rnifti	MIT	*2017-08-20 °2019-09-21
SPUTNIK by Inglese et al. (2018)	Mass Spectrometry Imaging	CRAN	imager	GPL ($\geq$ 3)	*2018-02-19 °2024-04-16
SAFARI by Fernández et al. (2022)	Shape analysis	CRAN	EBImage	GPL ($\geq$ 3)	*2021-02-25
pavo by Maia et al. (2019)	Spectral and Spatial analysis	CRAN	magick & imager	GPL ($\geq$ 2)	*2012-12-05 °2023-09-24
miet by Combès (2020)	Magnetic Resonance images	gitlab	Rnifti	MIT	*2019-09-06 °2023-12-20
scalpel by Petersen et al. (2017)	Calcium imaging	CRAN	-	GPL ($\geq$ 2)	*2017-03-14 °2021-02-03
ProFit by Robotham et al. (2016)	Galaxy images	CRAN-archived	EBImage	LGPL-3	*2016-09-29 †2022-08-08
fsbrain by Schäfer and Ecker (2020) & Schaefer (2024)	Neuroimaging	CRAN	magick	MIT	*2019-10-30 °2024-02-03
geomorph by Adams and Otárola-Castillo (2013)	Geometric morphometric shape analysis	CRAN	jpeg	GPL ($\geq$ 3)	*2012-10-26 °2024-03-05
imbibe	Medical images	CRAN	Rnifti	BSD-3-clause	*2020-10-26 °2022-11-09
opencv by Ooms and Wijffels (2024)	edge, body, face detection	CRAN	OpenCV	MIT	*2019-04-01 °2023-10-29
DRIP	jump regression, denoising, deblurring	CRAN	-	GPL ($\geq$ 2)	*2015-09-22 °2024-02-05
imagefluency by Mayer (2024)	image statistics based on fluency theory	CRAN	magick & OpenImageR	GPL-3	*2019-09-27 °2024-02-22
mand by Kawaguchi (2021)	Neuroimaging	CRAN	imager	GPL-2 GPL-3	*2020-05-06 °2023-09-12

	Application	Repo	based on	License	Status
recolorize by Weller et al. (2024)	Segmentation	CRAN	imager	CC BY 4.0	*2021-12-07
MaxContrastProjection by Jan Sauer (2017)	maximum contrast projection	Bioc	EBImage	Artistic-2.0	*2017-04-25 †2020-04-28

14 Combining forces - making use of the open-source approach

The majority of the aforementioned packages are designed to encompass all facets of image analysis, including preprocessing, quantification, and visualization. This integration is typically achieved through the utilization of one or more general-purpose packages (Table 1 and 2). The combination of existing packages or libraries with new code facilitates the development of specialized packages. R, as a package-based language, provides a convenient means of combining these specialized packages to meet the specific needs of the individual user. The following section illustrates the combination of packages to perform statistical analysis on quantified image data.

14.1 `biopixR` and `countfitteR`: quantitative analysis of DNA double strand breaks

DNA double strand breaks (DSBs) represent a particularly severe form of DNA damage, frequently resulting in apoptotic cell death in the absence of repair. The extent of DNA damage can be quantified through immunofluorescence staining, which employs antibodies against the phosphorylated histone protein H2AX (γH2AX). The staining process results in the formation of γH2AX foci, which serve as a quantitative representation of the number of DNA DSBs. It has been proposed that the number of DNA DSBs is indicative of the efficacy of an anti-tumor agent, thereby enabling the assessment of individual patient responses to therapies and the evaluation of the general cytotoxic effects of treatments *in vivo*. This enables more precise modulation of therapy according to the patient's individual needs (Rödiger et al., 2018; Ruhe et al., 2019; Schneider et al., 2019).

In the following example, the `biopixR` package was employed to quantify DNA double-strand breaks, resulting in an output of foci per cell (Figure 12). To achieve this objective, the green fluorescent foci were extracted by applying the `objectDetection()` function to the green color channel of the image (Figure 12A). The result of the foci extraction is illustrated in Figure 12B using the `changePixelColor()` function, whereby each of the distinct foci is highlighted in a different color. The DAPI-stained nuclei were extracted through the application of thresholding on the blue color channel. Subsequently, the resulting data frame was subjected to size filtering in order to eliminate any detected noise. The final quantification of foci per cell was achieved by comparing the coordinates of nuclei and foci in the obtained data frames. This result can then be further analyzed using the `countfitteR` package, which provides an automated evaluation of distribution models for count data (Burdukiewicz, 2019; Chilimoniuk et al., 2021). The resulting distribution is presented in Figure 13.

```
# Load the 'biopixR' package
library(biopixR)

# Import image from specified path
DSB_img <- importImage("figures/tim_242602_c_s3c1+2+3m4.tif")

# Extract the blue color channel representing the nuclei and
# the green color channel representing yH2AX foci
core <- as.cimg(DSB_img[, , , 3])
```

```
yH2AX <- as.cimg(DSB_img[, , , 2])

# Process the nuclei: thresholding, labeling, and converting to a data frame
cores <-
  threshold(core) |> label() |> as.data.frame() |> subset(value > 0)

# Calculate the center and size for the nuclei
DT <- as.data.table(cores)
cores_center <-
  DT[, list(mx = mean(x),
            my = mean(y),
            size = length(x)), by = value]

# Filter the nuclei based on size, to discard noise
cores_clean <-
  sizeFilter(cores_center,
             cores,
             lowerlimit = 150,
             upperlimit = Inf)

# Detect objects yH2AX foci in green color channel
DSB <- objectDetection(yH2AX, alpha = 1.1, sigma = 0)

# Function to compare coordinates from two data frames and count matches
compareCoordinates <- function(df1, df2) {
  # Create a single identifier for each coordinate pair
  df1$coord_id <- paste(round(df1$mx), round(df1$my), sep = ",")
  df2$coord_id <- paste(df2$x, df2$y, sep = ",")

  # Find matches by checking if coordinates from df2 exist in df1
  matches <- df2$coord_id %in% df1$coord_id

  # Convert df2 to a data table and add a column indicating matches
  DT <- data.table(df2)
  DT$DSB <- matches

  # Summarize the results
  result <-
    DT[, list(count = length(which(DSB == TRUE))), by = value]

  return(result)
}

# Compare coordinates between detected DSB centers and cleaned nuclei coordinates
count <- compareCoordinates(DSB$centers, cores_clean$coordinates)

# Extract the count column for further analysis
to_analyze <- count[, 2]
```

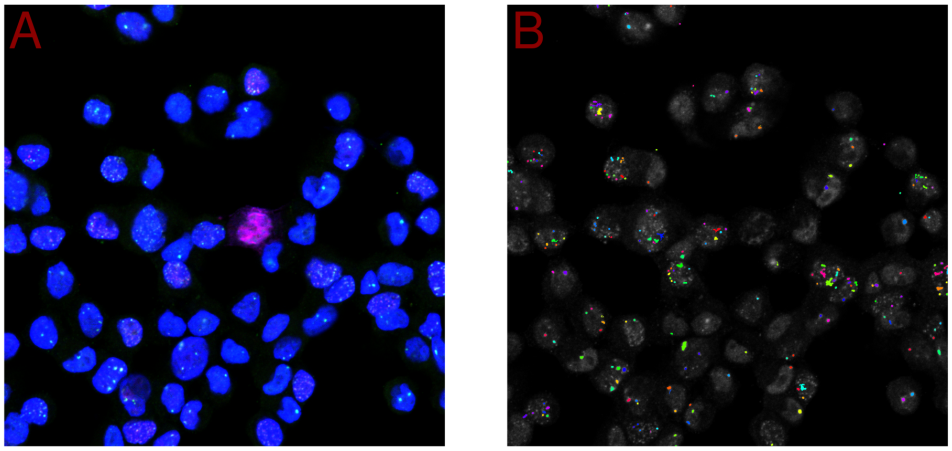


Figure 12: Quantification of DNA Double Strand Breaks: **A**) The image displays cells with nuclei stained using DAPI. The quantitative marker for DNA double strand breaks, γH2AX, targeted with a specific antibody, is visible as green fluorescent foci. The experimental procedure follows the method described by Rödiger et al. (2018). **B**) The γH2AX foci are quantified using the `biopixR` package. The detected foci are highlighted in different colors using the `changePixelColor()` function.

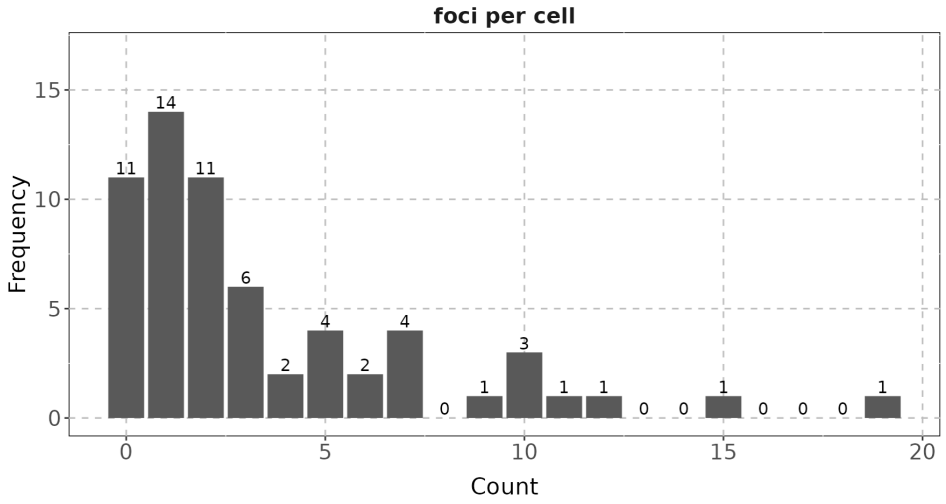


Figure 13: Analyzing Count Data with the `countfitteR` Package: The data representing the number of foci per cell obtained from the `biopixR` analysis were imported into the interactive `shiny` interface of the `countfitteR` package. This package analyzed the distribution and summarized the results. One outcome is illustrated in this figure, which shows the frequency distribution of a specific count of foci per cell.

15 Exploring the blank spot - z-stack imaging in R

Z-stack imaging refers to the capture of images that possess a third dimension, specifically image depth, which enables the spatial capture of molecules or the reconstruction of the three-dimensional architecture of tissues. One method for achieving z-stacking involves capturing multiple two-dimensional images at uniform intervals over the depth of an object by changing the focal plane. The individual 2D images are then reconstructed to create a 3D model (Trivedi and Mills, 2020; Kim et al., 2022).

The only packages currently available in the `R` programming language for dealing with z-stack imaging are `spatialTIME` and `MaxContrastProjection`. However, the `spatialTIME`

package necessitates preprocessing and is therefore unable to handle the images directly (Creed et al., 2021). The other package, `MaxContrastProjection`, has unfortunately been removed from Bioconductor. The package is capable of performing maximum contrast projection, whereby the z-stacks of a 3D image are merged into a 2D image (Jan Sauer, 2017). To the best of our knowledge, these are the only packages in `R` that address the topic of z-stack imaging.

16 Scaling new heights - high throughput analysis in the era of small and big data?

The exponential growth of data, which reached levels of zettabytes (10^{21} bytes) as early as 2012 (Sagiroglu and Sinanc, 2013), is accompanied by a significant increase in image generation due to advancements in imaging technologies such as microscopy. High-resolution images produced in a single experiment can result in data sets exceeding terabytes (Peng et al., 2012; Eliceiri et al., 2012). This surge in data generation across various fields has initiated the era of Big Data, which presents considerable challenges in the handling and interpretation of massive data sets (Cui et al., 2015). In automated microscopy, the rapid acquisition of large image volumes facilitates extensive screening processes but complicates the conversion of image stacks into actionable information and discoveries, resulting in a critical need for analytical pipelines that can efficiently identify regions of interest, compute relevant features, and perform statistical analysis, ensuring reproducibility and reliability (Wollman and Stuurman, 2007).

The extraction of quantitative information from images is a common practice, but it is becoming increasingly complex and error-prone when performed manually. This complexity requires the implementation of high-throughput methods capable of autonomously processing multiple images (Olivoto, 2022). These developments are crucial not only in specialized fields such as immunohistochemistry, fluorescence *in situ* hybridization (Ollion et al., 2013), drug discovery, and cell biology (Shariff et al., 2010), but also in promoting a data-driven approach to biological research, thereby accelerating tasks and enhancing research productivity (Rittscher, 2010).

The `R` programming language has limitations in handling large data sets. Since `R` places temporary copies of data in the random access memory (RAM) to access objects, it can lead to memory overload when processing data sets that exceed the available RAM. Additionally, `R` uses RAM to store generated data, so large lists of imported images can easily overwhelm the RAM. Moreover, `R` typically executes code on a single thread, not utilizing the full capabilities of the central processing unit (CPU). Several packages address issues such as file-based access and parallel computing, thereby enhancing `R`'s capability to handle big data. One approach is to combine `R` with the 'Hadoop' library (Prajapati, 2013; Oussous et al., 2018). Another effective method for managing big data is the use of the HDF5, which efficiently manages data storage and access, provides multicore reading and writing, and is well-suited for organizing complex data collections. The `cytomapper` package utilizes HDF5 to optimize file management (Nils Eling, Nicolas Damond, Tobias Hoch, 2020; Folk et al., 2011; Koranne, 2011).

Other packages, such as `pliman`, `biopixR`, and `FIELDimageR`, include features for optimized batch processing, such as parallel processing, by utilizing the `foreach` package for multi-core processing (Olivoto, 2022; Brauckhoff et al., 2024; Matias et al., 2020). However, these packages are not fully optimized for big data. The `biopixR` package simplifies image processing by providing a pipeline that scans entire directories and verifies image uniqueness using Message Digest 5 (MD5) sums. It enables the application of specific filters to batches of images and generates an `RMarkdown` log file detailing the operations performed. The results are saved in a manageable CSV format, enhancing the efficiency of handling whole image directories (Brauckhoff et al., 2024).

In conclusion, while R offers a range of options for handling big data, these options are not widely implemented in image processing packages. Consequently, the optimization and creation of workflows capable of handling big data is left to the end-user.

17 Summary

In conclusion, we present a summary of the major R packages previously discussed. This summary provides an overview of the general applications, published repositories, and licensing information associated with these packages. Furthermore, it includes a list of the dependencies or libraries that these packages rely on. The status column indicates both the initial publication date and the date of the most recent update, thereby demonstrating the ongoing commitment to maintaining these packages (Table 2).

Table 2: Summary of key characteristics of major R packages for image processing. The table details general applications, repository (Repo) sources (CRAN, Bioconductor (Bioc), and GitHub), primary package or library dependencies, and licensing information. The status column indicates the date of first publication (*) and the most recent update (°) for each package.

	Application	Repo	based on	License	Status
imager by Barthelmé and Tschumperlé (2019)	general purpose	CRAN	Cimg	LGPL-3	*2015-08-26 °2024-04-26
magick by Ooms (2024b)	general purpose	CRAN	Image Magick	MIT	*2016-07-24 °2024-02-18
EBImage by Pau et al. (2010)	general purpose	Bioc	-	LGPL	*2006-04-27 °2024-05-01
biopixR by Brauckhoff et al. (2024)	bioimages	CRAN	imager & magick	LGPL ($\geq$ 3)	*2024-03-25 °2024-11-11
pliman by Olivoto (2022)	plant images	CRAN	EBImage	GPL ($\geq$ 3)	*2021-05-15 °2023-10-14
mxnorm by Harris et al. (2022a)	multiplex images	CRAN	-	MIT	*2022-02-22 °2023-05-01
DIMPLE by Masotti et al. (2023)	multiplex images	GitHub	-	MIT	*2023-09-07
cytomapper by Eling et al. (2020)	multiplex images	Bioc	EBImage	GPL ($\geq$ 2)	*2020-10-28 °2024-05-01
SPIAT by Yang et al. (2020)	spatial data	Bioc	Spatial Experiment	Artistic-2.0	*2022-11-02 °2024-05-01
spatialTIME by Creed et al. (2021)	spatial data	CRAN	-	MIT	*2021-05-14 °2024-03-11
celltrackR by Wortel et al. (2021)	motion analysis	CRAN	-	GPL-2	*2020-03-31 °2024-03-26
FIELDimageR by Matias et al. (2020)	agricultural field trails	GitHub	EBImage	GPL-3	*2019-11-01 °2024-05-03
fslr by Muschelli et al. (2015)	MRI of the brain	CRAN	FMRIB library	GPL-3	*2014-06-13 °2022-08-25

	Application	Repo	based on	License	Status
colocr by Ahmed et al. (2019)	fluorescence microscopy	CRAN	imager & magick	GPL-3	*2019-05-31 °2020-05-08
imageseg by Niedballa et al. (2022a)	image segmentation	CRAN	magick	MIT	*2021-12-09 °2022-05-29
SimpleITK by Beare et al. (2018)	general purpose	GitHub	Simple ITK	Apache 2.0	*2015-11-16 °2020-09-17
pixelclasser by Real (2024)	image segmentation	CRAN	jpeg & tiff	GPL-3	*2021-10-21 °2023-10-18
OpenImageR	general purpose	CRAN	Rcpp	GPL-3	*2016-07-09 °2023-07-08
RniftyReg	image registration	CRAN	Rcpp & Rnifti	GPL-2	*2010-09-06 °2023-07-18

The packages outlined in Table 2 are examined in terms of their individual dependencies. A minimal number of dependencies is essential for ensuring long-term stability and functionality. The packages are organized according to their dependencies and imports, which were extracted from the `DESCRIPTION` files to facilitate the identification of similarities between the packages. The relationships between the packages are illustrated in the form of a dendrogram (Figure 14).

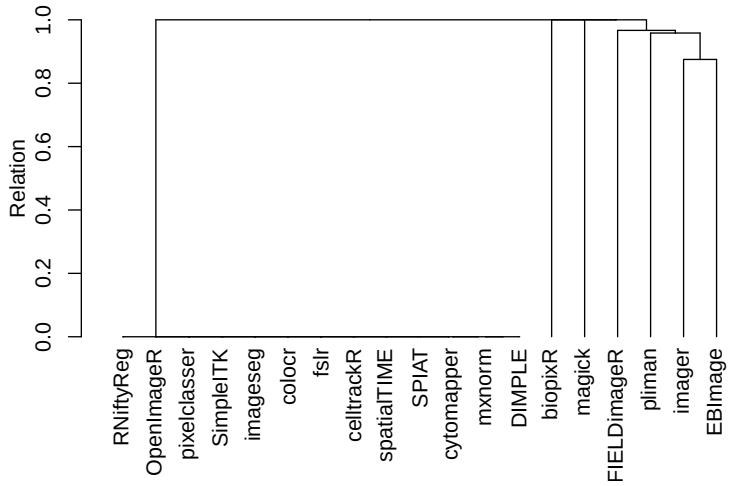


Figure 14: Dendrogram of Hierarchically Clustered Package Dependencies: The dendrogram depicts the outcomes of a hierarchical clustering of various image analysis packages, based on their named dependencies and imports, as extracted from their respective `DESCRIPTION` files. Each branch represents a distinct package, and the proximity between branches reflects the degree of similarity in their dependencies and imports. The required distance matrix was calculated using the binary method, also known as Jaccard distance. To perform the hierarchical clustering, the complete linkage clustering method was employed (R Core Team, 2023).

18 Conclusion

The Tables 1 and 2 highlight an array of `R` packages employed within bioimage informatics. These tools cater to diverse applications such as adaptive smoothing, vegetation phenology analysis, microbial culture imaging, cancer imaging, mass spectrometry imaging, shape

analysis, spectral and spatial analysis, magnetic resonance image processing, calcium imaging, galaxy image analysis, neuroimaging, geometric morphometric shape analysis, medical image processing, edge detection, body and face recognition, jump regression, denoising, and deblurring.

Many of these packages rely on common image processing libraries such as 'ImageMagick' and 'CImg' or specialized libraries like 'RNifti' for neuroimaging data and OpenCV for computer vision tasks. Some notable examples include `adimpro`, `gitter`, `SAFARI`, `pavo`, `rental`, `scalpel`, `ProFit`, and `fsbrain`.

The majority of these packages are hosted on CRAN, which serves as the primary repository for R packages. Notably, one package, rental, is hosted on GitLab, indicating that some packages may also be developed and distributed through alternative platforms. R is an open-source, free, and cross-platform programming language that extends these values to its packages (R Core Team, 2023). The CRAN Repository Policy states that package authors "should make all reasonable efforts to provide cross-platform portable code," typically requiring packages to run on at least two major R platforms.[29] Similarly, the standard tests employed by Bioconductor encompass evaluations on all major platforms, including Linux, macOS, and Windows.[30] Thus, it can be concluded that the majority of packages in these repositories are compatible across multiple platforms.

The most commonly used license in this domain is the GNU General Public License (GPL), particularly versions 2 and 3. Other licenses employed include the Lesser GNU General Public License (LGPL), MIT, Apache License 2.0, and others. The prevalence of open-source licenses reflects the collaborative nature of R package development. It's essential to ensure compatibility when combining code from different packages with varying licenses; otherwise, legal considerations might arise.

As previously outlined, the most fundamental image processing packages in R are `imager`, `magick`, `EBImage`, `OpenImageR`, and `SimpleITK`. Primarily, `imager`, `magick`, and `EBImage` form the foundation for the majority of the specialized packages reviewed. These packages support various formats, with JPEG and PNG being the most common and supported by all five packages. BMP and TIFF are also widely supported, while PDF and SVG formats are exclusively supported by `magick`.

Table 3: Supported File Formats by Main Image Processing Packages

	`imager`	`magick`	`EBImage`	`OpenImageR`	`SimpleITK`
JPEG	+	+	+	+	+
PNG	+	+	+	+	+
BMP	+	+	-	-	+
TIFF	-	+	+	+	+
PDF	-	+	-	-	-
SVG	-	+	-	-	-

The ongoing development of new code by the R community significantly enhances the capabilities of image analysis, fostering both growth and adaptability within the community. This ensures that R remains well-equipped to address emerging challenges effectively. The result is a diverse range of image processing packages, including versatile general-purpose tools and specialized pipelines designed for intricate analyses of biological images. This extensive array of tools in R not only demonstrates the versatility and applicability of these packages across different scientific disciplines but also solidifies R's position as an invaluable resource for researchers interested in leveraging image analysis to uncover novel insights. This review provides a concise overview of the current landscape of image processing packages available in R, emphasizing the pivotal role these tools play in advancing scientific

[29] https://cran.r-project.org/web/packages/policies.html, accessed 06/10/2024

[30] https://contributions.bioconductor.org/bioconductor-package-submissions.html, accessed 06/10/2024

research and discovery. The comprehensive toolkit, R, empowers researchers to drive forward innovations and enrich the scientific community. Finally, it is noteworthy that 92% of the 38 discovered packages are active in their respective repositories and thus considered up to date. Furthermore, 66% of these packages have been actively maintained with updates in the past 1.5 years. Among the identified packages, 14 provide users with GUIs or interactive functions. These packages include: `FIELDimageR`, `cytomapper`, `colocr`, `biopixR`, `EBImage`, `magick`, `imager`, `pavo`, `pliman`, `imagefluency`, `geomorph`, `fsbrain`, `scalpel`, and `adimpro`. The majority of the 38 packages identified during the research can be considered autonomous, offering all the necessary features for extensive image data analysis, including image import, processing, and visualization. However, some packages related to multiplex imaging necessitate preprocessing, rendering them unable to provide a complete analysis within the R environment.

All mentioned packages are open source and available either on CRAN, Bioconductor or GitHub.

Predicting the future is challenging, yet here we provide some opinions on trends in bioimage informatics, which ultimately will also be seen in R. Publications and conferences in the fields of image processing and computer vision show that advances are driven by artificial intelligence (AI), deep learning (particularly Convolutional Neural Networks (CNNs), Large Language Models (LLMs), and Vision Transformer models (VTs)), and data visualization (Ye et al., 2024; Belcher et al., 2023; Rabbani et al., 2021; Hameed et al., 2021; van der Velden et al., 2022). One example of deep learning is `imageseg`, which is using a CNN (U-Net and U-Net++ architectures) for general purpose image segmentation (Niedballa et al., 2022b). Another development is the deeper integration of R with advanced deep learning frameworks, which will enable users to build and deploy models, with applications like image classification, segmentation, and object detection. An example of such integration is `ellmer`, which makes various LLMs accessible from R for output streaming, tool calling, and structured data extraction.

The question arises: Is AI merely a buzzword, or is it here to stay? Given that AI is grounded in science and we already see applications in R, the latter is more probable. Consequently, R bioimage packages will be developed that combine image data with other multimodal data types, such as text and sensor data. Generative AI and advanced visualization techniques are also one topic due to the availability of generative models like diffusion models and Generative Adversarial Networks (GANs). These technologies open new possibilities for image augmentation and enhanced data visualization. It is important that such technologies stick to one of R's strengths, which is explainability, in particular focusing on transparent, understandable, and explainable AI (xAI).

19 Funding

This review was partially funded by the project Rubin: NeuroMiR (03RU1U051A, federal ministry of education and research, Germany).

20 Conflict of interest

The authors declare no conflict of interest.

21 Acknowledgements

We would like to express our gratitude to Dr. Coline Kieffer for providing the microbead images used in this review. We thank Robert M Flight at codeberg.org for reading and improving the manuscript.

References

MoleculeExperiment, 2024. URL http://bioconductor.org/packages/MoleculeExperiment/. [p233]

D. C. Adams and E. Otárola-Castillo. geomorph: an R package for the collection and analysis of geometric morphometric shape data. *Methods in Ecology and Evolution*, 4(4):393–399, Apr. 2013. ISSN 2041-210X. doi: 10.1111/2041-210x.12035. [p240]

O. Aherne, M. Mørch, R. Ortiz, O. Shannon, and J. R. Davies. A novel multiplex fluorescent-labeling method for the visualization of mixed-species biofilms in vitro. *Microbiology Spectrum*, May 2024. ISSN 2165-0497. doi: 10.1128/spectrum.00253-24. [p230, 231]

M. Ahmed. colocr: Conduct co-localization analysis of fluorescence microscopy images, 2020. URL https://CRAN.R-project.org/package=colocr. R package version 0.1.1. [p237, 239]

M. Ahmed, T. H. Lai, and D. R. Kim. colocr: an R package for conducting co-localization analysis on fluorescence microscopy images. *PeerJ*, 7:e7255, jul 2019. doi: 10.7717/peerj.7255. [p237, 239, 246]

R. A. Amezquita, A. T. L. Lun, E. Becht, V. J. Carey, L. N. Carpp, L. Geistlinger, F. Marini, K. Rue-Albrecht, D. Risso, C. Soneson, L. Waldron, H. Pagès, M. L. Smith, W. Huber, M. Morgan, R. Gottardo, and S. C. Hicks. Orchestrating single-cell analysis with bioconductor. *Nature Methods*, 17(2):137–145, Dec. 2019. ISSN 1548-7105. doi: 10.1038/s41592-019-0654-x. [p239]

G. P. Andrzej Oleś. Ebimage, 2017. [p228, 229]

Andrzej Oleś, John Lee. Rbioformats, 2023. [p214]

M. Angelo, S. C. Bendall, R. Finck, M. B. Hale, C. Hitzman, A. D. Borowsky, R. M. Levenson, J. B. Lowe, S. D. Liu, S. Zhao, Y. Natkunam, and G. P. Nolan. Multiplexed ion beam imaging of human breast tumors. *Nature Medicine*, 20(4):436–442, Mar. 2014. ISSN 1546-170X. doi: 10.1038/nm.3488. [p231]

M. Austenfeld and W. Beyschlag. A graphical user interface for R in a rich client platform for ecological modeling. *Journal of Statistical Software*, 49:1–19, June 2012. ISSN 1548-7660. doi: 10.18637/jss.v049.i04. URL https://doi.org/10.18637/jss.v049.i04. [p213]

S. Barthelmé and D. Tschumperlé. imager: an R package for image processing based on CImg. *Journal of Open Source Software*, 4(38):1012, jun 2019. doi: 10.21105/joss.01012. [p228, 245]

S. Barthelme, D. Tschumperle, J. Wijffels, H. E. Assemlal, S. Ochi, A. Robotham, and R. Tobar. imager: Image processing library based on 'cimg', 2024. URL https://CRAN.R-project.org/package=imager. R package version 1.0.1. [p228]

R. Beare, B. Lowekamp, and Z. Yaniv. Image segmentation, registration and characterization in R with simpleitk. *Journal of Statistical Software*, 86(8), 2018. doi: 10.18637/jss.v086.i08. [p230, 246]

A. Behura. The cluster analysis and feature selection: Perspective of machine learning and image processing, Jan. 2021. [p224]

B. T. Belcher, E. H. Bower, B. Burford, M. R. Celis, A. K. Fahimipour, I. L. Guevara, K. Katija, Z. Khokhar, A. Manjunath, S. Nelson, S. Olivetti, E. Orenstein, M. H. Saleh, B. Vaca, S. Valladares, S. A. Hein, and A. M. Hein. Demystifying image-based machine learning: a practical guide to automated analysis of field imagery using modern machine learning tools. *Frontiers in Marine Science*, 10, June 2023. ISSN 2296-7745. doi: 10.3389/fmars.2023.1157370. URL https://www.frontiersin.org/journals/marine-science/articles/10.3389/fmars.2023.1157370/full. Publisher: Frontiers. [p248]

S. Besson, R. Leigh, M. Linkert, C. Allan, J.-M. Burel, M. Carroll, D. Gault, R. Gozim, S. Li, D. Lindner, J. Moore, W. Moore, P. Walczysko, F. Wong, and J. R. Swedlow. *Bringing Open Data to Whole Slide Imaging*, pages 3–10. Springer International Publishing, 2019. ISBN 9783030239374. doi: 10.1007/978-3-030-23937-4_1. [p213]

J. C. Bezdek, R. Ehrlich, and W. Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2–3):191–203, Jan. 1984. ISSN 0098-3004. doi: 10.1016/0098-3004(84)90020-7. [p224]

R. Bise, T. Kanade, Z. Yin, and S. il Huh. Automatic cell tracking applied to analysis of cell migration in wound healing assay. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, Aug. 2011. doi: 10.1109/iembs.2011.6091525. [p233]

B. J. Boom, P. X. Huang, J. He, and R. B. Fisher. *Supporting ground-truth annotation of image datasets using clustering*. IEEE, Piscataway, NJ, 2012. ISBN "9781467322164". URL https://ieeexplore.ieee.org/abstract/document/6460437. Includes bibliographical references and author index. [p223]

T. Brauckhoff, C. Kieffer, and S. Rödiger. biopixr: Extracting insights from biological images. *Journal of Open Source Software*, 9(102):7074, Oct. 2024. ISSN 2475-9066. doi: 10.21105/joss.07074. [p218, 223, 224, 239, 244, 245]

M. Burdukiewicz. countfitter: Comprehensive automatized evaluation of distribution models for count data, Feb. 2019. [p241]

M. Burdukiewicz, A.-N. Spiess, D. Rafacz, K. Blagodatskikh, and S. Rödiger. PCRedux: A quantitative PCR machine learning toolkit. *Journal of Open Source Software*, 7(76):4407, Aug. 2022. ISSN 2475-9066. doi: 10.21105/joss.04407. URL https://joss.theoj.org/papers/10.21105/joss.04407. Number: 76. [p213]

J. C. Caicedo, S. Cooper, F. Heigwer, S. Warchal, P. Qiu, C. Molnar, A. S. Vasilevich, J. D. Barry, H. S. Bansal, O. Kraus, M. Wawer, L. Paavolainen, M. D. Herrmann, M. Rohban, J. Hung, H. Hennig, J. Concannon, I. Smith, P. A. Clemons, S. Singh, P. Rees, P. Horvath, R. G. Linington, and A. E. Carpenter. Data-analysis strategies for image-based cell profiling. *Nature Methods*, 14(9):849–863, sep 2017. doi: 10.1038/nmeth.4397. [p212]

P.-O. Caron and A. Dufresne. image2data: An R package to turn images in data sets. *The Quantitative Methods for Psychology*, 18(2):186–195, July 2022. ISSN 2292-1354. doi: 10.20982/tqmp.18.2.p186. URL http://www.tqmp.org/RegularArticles/vol18-2/p186. [p235]

A. Chessel. An overview of data science uses in bioimage informatics. *Methods*, 115:110–118, feb 2017. doi: 10.1016/j.ymeth.2016.12.014. [p212, 213]

J. Chilimoniuk, A. Gosiewska, J. Słowik, R. Weiss, P. M. Deckert, S. Rödiger, and M. Burdukiewicz. countfitteR: efficient selection of count distributions to assess DNA damage. *Annals of Translational Medicine*, 9(7):528–528, apr 2021. doi: 10.21037/atm-20-6363. [p241]

J. Chilimoniuk, A. Erol, S. Rödiger, and M. Burdukiewicz. Challenges and opportunities in processing NanoString nCounter data. *Computational and Structural Biotechnology Journal*, 23:1951–1958, Dec. 2024. ISSN 2001-0370. doi: 10.1016/j.csbj.2024.04.061. URL https://www.sciencedirect.com/science/article/pii/S2001037024001454. [p213]

W. Cho, S. Kim, and Y.-G. Park. Towards multiplexed immunofluorescence of 3d tissues. *Molecular Brain*, 16(1), May 2023. ISSN 1756-6606. doi: 10.1186/s13041-023-01027-9. [p231]

J. Clayden, M. Modat, B. Presles, T. Anthopoulos, and P. Daga. Rniftyreg: Image registration using the 'niftyreg' library, 2023. URL https://CRAN.R-project.org/package=RNiftyReg. R package version 2.8.1. [p228]

J. D. Clayden, M. Dayan, and C. A. Clark. Principal networks. *PLoS ONE*, 8(4):e60997, Apr. 2013. ISSN 1932-6203. doi: 10.1371/journal.pone.0060997. [p228]

B. Combès. miet: an R package for region of interest analysis from magnetic reasonance images. *Journal of Open Source Software*, 5(45):1862, Jan. 2020. ISSN 2475-9066. doi: 10.21105/joss.01862. [p240]

J. Creed, R. Thapa, C. Wilson, A. Soupir, O. Ospina, J. Wrobel, B. Fridley, and F. Lab. spatialtime: Spatial analysis of vectra immunoflourescent data, 2024. URL https://CRAN.R-project.org/package=spatialTIME. R package version 1.3.4-3. [p232]

J. H. Creed, C. M. Wilson, A. C. Soupir, C. M. Colin-Leitzinger, G. J. Kimmel, O. E. Ospina, N. H. Chakiryan, J. Markowitz, L. C. Peres, A. Coghill, and B. L. Fridley. spatialtime and itime: R package and shiny application for visualization and analysis of immunofluorescence data. *Bioinformatics*, 37(23):4584–4586, Nov. 2021. ISSN 1367-4811. doi: 10.1093/bioinformatics/btab757. [p230, 232, 244, 245]

S. Cui, G. Schwarz, and M. Datcu. Remote sensing image classification: No features, no clustering. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(11):5158–5170, Nov. 2015. ISSN 2151-1535. doi: 10.1109/jstars.2015.2495267. [p244]

N. Damond, S. Engler, V. R. Zanotelli, D. Schapiro, C. H. Wasserfall, I. Kusmartseva, H. S. Nick, F. Thorel, P. L. Herrera, M. A. Atkinson, and B. Bodenmiller. A map of human type 1 diabetes progression by imaging mass cytometry. *Cell Metabolism*, 29(3):755–768.e5, Mar. 2019. ISSN 1550-4131. doi: 10.1016/j.cmet.2018.11.014. [p230, 231]

K. W. Dunn, M. M. Kamocka, and J. H. McDonald. A practical guide to evaluating colocalization in biological microscopy. *American Journal of Physiology-Cell Physiology*, 300(4):C723–C742, Apr. 2011. ISSN 1522-1563. doi: 10.1152/ajpcell.00462.2010. [p239]

J. Einhaus, A. Rochwarger, S. Mattern, B. Gaudillière, and C. M. Schürch. High-multiplex tissue imaging in routine pathology—are we there yet? *Virchows Archiv*, 482(5):801–812, Feb. 2023. ISSN 1432-2307. doi: 10.1007/s00428-023-03509-6. [p231]

K. W. Eliceiri, M. R. Berthold, I. G. Goldberg, L. Ibáñez, B. S. Manjunath, M. E. Martone, R. F. Murphy, H. Peng, A. L. Plant, B. Roysam, N. Stuurman, J. R. Swedlow, P. Tomancak, and A. E. Carpenter. Biological imaging software tools. *Nature Methods*, 9(7):697–710, jun 2012. doi: 10.1038/nmeth.2084. [p212, 244]

N. Eling, N. Damond, T. Hoch, and B. Bodenmiller. cytomapper: an r/bioconductor package for visualization of highly multiplexed imaging data. *Bioinformatics*, 36(24):5706–5708, dec 2020. doi: 10.1093/bioinformatics/btaa1061. [p230, 231, 237, 245]

M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996. URL https://dl.acm.org/doi/10.5555/3001460.3001507. [p224]

Y. Feng, T. Yang, J. Zhu, M. Li, M. Doyle, V. Ozcoban, G. T. Bass, A. Pizzolla, L. Cain, S. Weng, A. Pasam, N. Kocovski, Y.-K. Huang, S. P. Keam, T. P. Speed, P. J. Neeson, R. B. Pearson, S. Sandhu, D. L. Goode, and A. S. Trigos. Spatial analysis with SPIAT and spaSim to characterize and simulate tissue microenvironments. *Nature Communications*, 14(1), may 2023. doi: 10.1038/s41467-023-37822-0. [p231, 232]

E. Fernández, S. Yang, S. H. Chiou, C. Moon, C. Zhang, B. Yao, G. Xiao, and Q. Li. Safari: shape analysis for ai-segmented images. *BMC Medical Imaging*, 22(1), July 2022. ISSN 1471-2342. doi: 10.1186/s12880-022-00849-8. [p240]

G. Filippa, E. Cremonese, M. Migliavacca, M. Galvagno, M. Forkel, L. Wingate, E. Tomelleri, U. Morra di Cella, and A. D. Richardson. Phenopix: A R package for image-based vegetation phenology. *Agricultural and Forest Meteorology*, 220:141–150, Apr. 2016. ISSN 0168-1923. doi: 10.1016/j.agrformet.2016.01.006. [p240]

M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson. An overview of the hdf5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, EDBT/ICDT '11. ACM, Mar. 2011. doi: 10.1145/1966895.1966900. [p244]

C. Geithe, B. Zeng, C. Schmidt, F. Dinter, D. Roggenbuck, W. Lehmann, G. Dame, P. Schierack, K. Hanack, and S. Rödiger. A multiplex microchamber diffusion assay for the antibody-based detection of micrornas on randomly ordered microbeads. *Biosensors and Bioelectronics: X*, 18:100484, June 2024. ISSN 2590-1370. doi: 10.1016/j.biosx.2024.100484. [p215]

R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. Yang, and J. Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology*, 5(10):R80, 2004. ISSN 1465-6906. doi: 10.1186/gb-2004-5-10-r80. [p239]

M. J. Gerdes, C. J. Sevinsky, A. Sood, S. Adak, M. O. Bello, A. Bordwell, A. Can, A. Corwin, S. Dinn, R. J. Filkins, D. Hollman, V. Kamath, S. Kaanumalle, K. Kenny, M. Larsen, M. Lazare, Q. Li, C. Lowes, C. C. McCulloch, E. McDonough, M. C. Montalto, Z. Pang, J. Rittscher, A. Santamaria-Pang, B. D. Sarachan, M. L. Seel, A. Seppo, K. Shaikh, Y. Sui, J. Zhang, and F. Ginty. Highly multiplexed single-cell analysis of formalin-fixed, paraffin-embedded cancer tissue. *Proceedings of the National Academy of Sciences*, 110(29):11982–11987, July 2013. ISSN 1091-6490. doi: 10.1073/pnas.1300136110. [p230, 231]

S. Ghosh, N. Das, I. Das, and U. Maulik. Understanding deep learning techniques for image segmentation. *ACM Computing Surveys*, 52(4):1–35, Aug. 2019. ISSN 1557-7341. doi: 10.1145/3329784. [p215]

F. M. Giorgi, C. Ceraolo, and D. Mercatelli. The R language: An engine for bioinformatics and data science. *Life*, 12(5):648, apr 2022. doi: 10.3390/life12050648. [p213]

I. G. Goldberg, C. Allan, J.-M. Burel, D. Creager, A. Falconi, H. Hochheiser, J. Johnston, J. Mellen, P. K. Sorger, and J. R. Swedlow. The open microscopy environment (ome) data model and xml file: open tools for informatics and quantitative analysis in biological imaging. *Genome Biology*, 6(5), May 2005. ISSN 1474-760X. doi: 10.1186/gb-2005-6-5-r47. [p213]

Y. Goltsev, N. Samusik, J. Kennedy-Darling, S. Bhate, M. Hale, G. Vazquez, S. Black, and G. P. Nolan. Deep profiling of mouse splenic architecture with codex multiplexed imaging. *Cell*, 174(4):968–981.e15, Aug. 2018. ISSN 0092-8674. doi: 10.1016/j.cell.2018.07.010. [p230, 231]

R. Haase, E. Fazeli, D. Legland, M. Doube, S. Culley, I. Belevich, E. Jokitalo, M. Schorb, A. Klemm, and C. Tischer. A hitchhiker's guide through the bio-image analysis software universe. *FEBS Letters*, 596(19):2472–2485, jul 2022. doi: 10.1002/1873-3468.14451. [p213]

M. Haghighat, S. Zonouz, and M. Abdel-Mottaleb. Cloudid: Trustworthy cloud-based and cross-enterprise biometric identification. *Expert Systems with Applications*, 42(21):7905–7916, Nov. 2015. ISSN 0957-4174. doi: 10.1016/j.eswa.2015.06.025. [p229]

I. M. Hameed, S. H. Abdulhussain, and B. M. Mahmmod. Content-based image retrieval: A review of recent trends. *Cogent Engineering*, 8(1):1927469, Jan. 2021. ISSN null. doi: 10.1080/23311916.2021.1927469. URL https://doi.org/10.1080/23311916.2021.1927469. Publisher: Cogent OA _eprint: https://doi.org/10.1080/23311916.2021.1927469. [p248]

Y. Hao, T. Stuart, M. H. Kowalski, S. Choudhary, P. Hoffman, A. Hartman, A. Srivastava, G. Molla, S. Madad, C. Fernandez-Granda, and R. Satija. Dictionary learning for integrative, multimodal and scalable single-cell analysis. *Nature Biotechnology*, 42(2):293–304, Feb. 2024. ISSN 1546-1696. doi: 10.1038/s41587-023-01767-y. URL

https://www.nature.com/articles/s41587-023-01767-y. Publisher: Nature Publishing Group. [p232]

R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, Nov. 1973. ISSN 2168-2909. doi: 10.1109/tsmc.1973.4309314. [p223, 224]

C. Harris. mxnorm: Apply normalization methods to multiplexed images, 2023. URL https://cran.r-project.org/package=mxnorm. R package version 1.0.3. [p231]

C. Harris, J. Wrobel, and S. Vandekar. mxnorm: An R package to normalize multiplexed imaging data. *Journal of Open Source Software*, 7(71):4180, mar 2022a. doi: 10.21105/joss.04180. [p230, 231, 245]

C. R. Harris, E. T. McKinley, J. T. Roland, Q. Liu, M. J. Shrubsole, K. S. Lau, R. J. Coffey, J. Wrobel, and S. N. Vandekar. Quantifying and correcting slide-to-slide variation in multiplexed immunofluorescence images. *Bioinformatics*, 38(6):1700–1707, Jan. 2022b. ISSN 1367-4811. doi: 10.1093/bioinformatics/btab877. [p231]

G. C. Heineck, I. G. McNish, J. M. Jungers, E. Gilbert, and E. Watkins. Using r-based image analysis to quantify rusts on perennial ryegrass. *The Plant Phenome Journal*, 2(1):1–10, jan 2019. doi: 10.2135/tppj2018.12.0010. [p229]

R. J. Hijmans. terra: Spatial data analysis, Mar. 2020. [p234]

R. J. Hijmans. *terra: Spatial Data Analysis*, 2024. URL https://CRAN.R-project.org/package=terra. R package version 1.7-71. [p233]

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, Apr. 2019. ISSN 0924-2716. doi: 10.1016/j.isprsjprs.2019.02.009. [p215]

A. K. M. N. Hossian and G. Mattheolabakis. *Cellular Migration Assay: An In Vitro Technique to Simulate the Wound Repair Mechanism*, pages 77–83. Springer US, Aug. 2020. ISBN "9781071608456". doi: 10.1007/978-1-0716-0845-6_8. [p233]

Y. Hu, M. L. Becker, and R. K. Willits. Quantification of cell migration: metrics selection to model application. *Frontiers in Cell and Developmental Biology*, 11, May 2023. ISSN 2296-634X. doi: 10.3389/fcell.2023.1155882. [p233]

P. Inglese, G. Correia, Z. Takats, J. K. Nicholson, and R. C. Glen. Sputnik: an R package for filtering of spatially related peaks in mass spectrometry imaging data. *Bioinformatics*, 35 (1):178–180, July 2018. ISSN 1367-4811. doi: 10.1093/bioinformatics/bty622. [p240]

B. F. Jan Sauer. Maxcontrastprojection, 2017. [p241, 244]

M. Jenkinson and S. Smith. A global optimisation method for robust affine registration of brain images. *Medical Image Analysis*, 5(2):143–156, June 2001. ISSN 1361-8415. doi: 10.1016/s1361-8415(01)00036-6. [p228]

D. Jude Hemanth and J. Anitha. *Image Pre-processing and Feature Extraction Techniques for Magnetic Resonance Brain Image Analysis*, pages 349–356. Springer Berlin Heidelberg, 2012. ISBN "9783642355943". doi: 10.1007/978-3-642-35594-3_47. [p217]

B. Jähne. *Digital image processing*. Engineering online library. Springer, Berlin, 5., rev. and extended ed. edition, 2002. ISBN "3540677542". doi: 10.1088/0957-0233/13/9/711. Dt. Ausg. u.d.T.: Digitale Bildverarbeitung. - CD-ROM-Ausg. u.d.T.: Digital image processing, idn 10527414. [p215]

J.-P. Kaiser and A. Bruinink. Investigating cell–material interactions by monitoring and analysing cell migration. *Journal of Materials Science: Materials in Medicine*, 15(4):429–435, Apr. 2004. ISSN 0957-4530. doi: 10.1023/b:jmsm.0000021115.55254.a8. [p233]

L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, Mar. 1990. ISBN "9780470316801". doi: 10.1002/9780470316801. [p224]

A. Kawaguchi. *Multivariate Analysis for Neuroimaging Data*. CRC Press, Milton, 2021. ISBN "9781000369861". doi: https://doi.org/10.1201/9780429289606. Description based on publisher supplied metadata and other sources. [p240]

D. Kim, R. Burkhardt, S. A. Alperstein, H. N. Gokozan, A. Goyal, J. J. Heymann, A. Patel, and M. T. Siddiqui. Evaluating the role of z-stack to improve the morphologic evaluation of urine cytology whole slide images for high-grade urothelial carcinoma: Results and review of a pilot study. *Cancer Cytopathology*, 130(8):630–639, May 2022. ISSN 1934-6638. doi: 10.1002/cncy.22595. [p243]

T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990. ISSN 0018-9219. doi: 10.1109/5.58325. [p224]

T. Kohonen. Essentials of the self-organizing map. *Neural Networks*, 37:52–65, Jan. 2013. ISSN 0893-6080. doi: 10.1016/j.neunet.2012.09.018. [p224]

S. Koranne. *Handbook of Open Source Tools*. SpringerLink. Springer US, Boston, MA, 1 edition, 2011. ISBN "9781441977199". doi: https://doi.org/10.1007/978-1-4419-7719-9. [p244]

A. Kumar Dubey, U. Gupta, and S. Jain. Medical data clustering and classification using tlbo and machine learning algorithms. *Computers, Materials & Continua*, 70(3):4523–4543, 2022. ISSN 1546-2226. doi: 10.32604/cmc.2022.021148. [p223]

M. J. Lajeunesse. *juicr: Automated and Manual Extraction of Numerical Data from Scientific Images*, 2021. URL https://CRAN.R-project.org/package=juicr. R package version 0.1. [p235]

L. Larsson, L. Franzén, P. L. Ståhl, and J. Lundeberg. Semla: a versatile toolkit for spatially resolved transcriptomics analysis and visualization. *Bioinformatics*, 39(10):btad626, Oct. 2023. ISSN 1367-4811. doi: 10.1093/bioinformatics/btad626. URL https://doi.org/10.1093/bioinformatics/btad626. [p232]

R. Leigh, D. Gault, M. Linkert, J.-M. Burel, J. Moore, S. Besson, and J. R. Swedlow. Ome files - an open source reference library for the ome-xml metadata model and the ome-tiff file format. Nov. 2016. doi: 10.1101/088740. [p213]

M. Linkert, C. T. Rueden, C. Allan, J.-M. Burel, W. Moore, A. Patterson, B. Loranger, J. Moore, C. Neves, D. MacDonald, A. Tarkowska, C. Sticco, E. Hill, M. Rossner, K. W. Eliceiri, and J. R. Swedlow. Metadata matters: access to image data in the real world. *Journal of Cell Biology*, 189(5):777–782, May 2010. ISSN 0021-9525. doi: 10.1083/jcb.201004104. [p213]

B. C. Lowekamp, D. T. Chen, L. Ibáñez, and D. Blezek. The design of SimpleITK. *Frontiers in Neuroinformatics*, 7, 2013. doi: 10.3389/fninf.2013.00045. [p230]

G. Maheshwari and D. A. Lauffenburger. Deconstructing (and reconstructing) cell migration. *Microscopy Research and Technique*, 43(5):358–368, Dec. 1998. ISSN 1097-0029. doi: 10.1002/(sici)1097-0029(19981201)43:5<358::aid-jemt2>3.0.co;2-d. [p233]

R. Maia, H. Gruson, J. A. Endler, and T. E. White. pavo2: New tools for the spectral and spatial analysis of colour in R. *Methods in Ecology and Evolution*, 10(7):1097–1107, Apr. 2019. ISSN 2041-210X. doi: 10.1111/2041-210x.13174. [p240]

M. Masotti, N. Osher, J. Eliason, A. Rao, and V. Baladandayuthapani. Dimple: An R package to quantify, visualize, and model spatial cellular interactions from multiplex imaging with distance matrices. *Patterns*, 4(12):100879, Dec. 2023. ISSN 2666-3899. doi: 10.1016/j.patter.2023.100879. [p231, 245]

F. I. Matias, M. V. Caraza-Harter, and J. B. Endelman. Fieldimager: An R package to analyze orthomosaic images from agricultural field trials. *The Plant Phenome Journal*, 3(1), Jan. 2020. ISSN 2578-2703. doi: 10.1002/ppj2.20005. [p221, 223, 244, 245]

S. Mayer. stm/imagefluency: imagefluency 0.2.5, 2024. [p240]

E. Moen, D. Bannon, T. Kudo, W. Graf, M. Covert, and D. V. Valen. Deep learning for cellular image analysis. *Nature Methods*, 16(12):1233–1246, may 2019. doi: 10.1038/s41592-019-0403-1. [p212, 223]

J. Moore, C. Allan, S. Besson, J.-M. Burel, E. Diel, D. Gault, K. Kozlowski, D. Lindner, M. Linkert, T. Manz, W. Moore, C. Pape, C. Tischer, and J. R. Swedlow. Ome-ngff: a next-generation file format for expanding bioimaging data-access strategies. *Nature Methods*, 18(12):1496–1498, Nov. 2021. ISSN 1548-7105. doi: 10.1038/s41592-021-01326-w. [p213]

J. Moore, D. Basurto-Lozada, S. Besson, J. Bogovic, J. Bragantini, E. M. Brown, J.-M. Burel, X. Casas Moreno, G. de Medeiros, E. E. Diel, D. Gault, S. S. Ghosh, I. Gold, Y. O. Halchenko, M. Hartley, D. Horsfall, M. S. Keller, M. Kittisopikul, G. Kovacs, A. Küpcü Yoldaş, K. Kyoda, A. le Tournoulx de la Villegeorges, T. Li, P. Liberali, D. Lindner, M. Linkert, J. Lüthi, J. Maitin-Shepard, T. Manz, L. Marconato, M. McCormick, M. Lange, K. Mohamed, W. Moore, N. Norlin, W. Ouyang, B. Özdemir, G. Palla, C. Pape, L. Pelkmans, T. Pietzsch, S. Preibisch, M. Prete, N. Rzepka, S. Samee, N. Schaub, H. Sidky, A. C. Solak, D. R. Stirling, J. Striebel, C. Tischer, D. Toloudis, I. Virshup, P. Walczysko, A. M. Watson, E. Weisbart, F. Wong, K. A. Yamauchi, O. Bayraktar, B. A. Cimini, N. Gehlenborg, M. Haniffa, N. Hotaling, S. Onami, L. A. Royer, S. Saalfeld, O. Stegle, F. J. Theis, and J. R. Swedlow. Ome-zarr: a cloud-optimized bioimaging file format with international community support. *Histochemistry and Cell Biology*, 160(3):223–251, July 2023. ISSN 1432-119X. doi: 10.1007/s00418-023-02209-1. [p213]

S. Mostafa and H. Amano. Effect of clustering data in improving machine learning model accuracy. *Journal of Theoretical and Applied Information Technology*, 97(21):2973–2981, Nov. 2019. ISSN 1992-8645. URL https://www.jatit.org/volumes/Vol97No21/7Vol97No21.pdf. [p223]

L. Mouselimis, S. Machine, J. Buchner, M. Haghighat, R. Achanta, and O. Onyshchak. Openimager: An image processing toolkit, 2023. URL https://CRAN.R-project.org/package=OpenImageR. R package version 1.3.0. [p215, 230]

R. F. Murphy. A new era in bioimage informatics. *Bioinformatics*, 30(10):1353–1353, apr 2014. doi: 10.1093/bioinformatics/btu158. [p212, 223]

J. Muschelli. fslr: Wrapper functions for 'fsl' ('fmrib' software library) from functional mri of the brain ('fmrib'), 2022. URL https://CRAN.R-project.org/package=fslr. R package version 2.25.2. [p239]

J. Muschelli, E. Sweeney, M. Lindquist, and C. Crainiceanu. fslr: Connecting the FSL software with R. *The R Journal*, 7(1):163, 2015. doi: 10.32614/rj-2015-013. [p239, 245]

G. Nasierding, G. Tsoumakas, and A. Z. Kouzani. Clustering based multi-label classification for image annotation and retrieval. In *2009 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, Oct. 2009. doi: 10.1109/icsmc.2009.5346902. [p224]

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, oct 2022a. doi: 10.1111/2041-210x.13984. [p215, 216, 246]

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, 2022b. ISSN 2041-210X. doi: 10.1111/2041-210X.13984. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13984. _eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.13984. [p248]

J. Niedballa, J. Axtner, L. I. for Zoo, and W. Research. imageseg: Deep learning models for image segmentation, 2022c. URL https://CRAN.R-project.org/package=imageseg. R package version 0.5.0. [p216]

Nils Eling, Nicolas Damond, Tobias Hoch. cytomapper, 2020. [p231, 237, 244]

S. Ochi. magickgui: Gui tools for interactive image processing with 'magick', 2023. URL https://CRAN.R-project.org/package=magickGUI. R package version 1.3.1. [p238]

T. Olivoto. Lights, camera, pliman! an R package for plant image analysis. *Methods in Ecology and Evolution*, 13(4):789–798, feb 2022. doi: 10.1111/2041-210x.13803. [p219, 223, 244, 245]

J. Ollion, J. Cochennec, F. Loll, C. Escudé, and T. Boudier. TANGO: a generic tool for high-throughput 3d image analysis for studying nuclear organization. *Bioinformatics*, 29(14): 1840–1841, may 2013. ISSN 1367-4803. doi: 10.1093/bioinformatics/btt276. [p244]

J. Ooms. *magick: Advanced Graphics and Image-Processing in R*, 2024a. URL https://docs.ropensci.org/magick/. R package version 2.8.3, https://CRAN.R-project.org/package=magick/, https://github.com/ropensci/magick and https://ropensci.r-universe.dev/magick/. [p229]

J. Ooms. *magick: Advanced Graphics and Image-Processing in R*, 2024b. URL https://ropensci.r-universe.dev/magick. R package version 2.8.3. [p245]

J. Ooms and J. Wijffels. *opencv: Bindings to 'OpenCV' Computer Vision Library*, 2024. URL https://ropensci.r-universe.dev/opencv. R package version 0.4.9001. [p240]

A. Oussous, F.-Z. Benjelloun, A. Ait Lahcen, and S. Belfkih. Big data technologies: A survey. *Journal of King Saud University - Computer and Information Sciences*, 30(4):431–448, Oct. 2018. ISSN 1319-1578. doi: 10.1016/j.jksuci.2017.06.001. [p244]

H.-S. Park and C.-H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2):3336–3341, Mar. 2009. ISSN 0957-4174. doi: 10.1016/j.eswa.2008.01.039. [p224]

G. Pau, F. Fuchs, O. Sklyar, M. Boutros, and W. Huber. EBImage — an R package for image processing with applications to cellular phenotypes. *Bioinformatics*, 26(7):979–981, mar 2010. doi: 10.1093/bioinformatics/btq046. [p216, 223, 228, 229, 245]

P. Paul-Gilloteaux. Bioimage informatics: Investing in software usability is essential. *PLOS Biology*, 21(7):e3002213, jul 2023. doi: 10.1371/journal.pbio.3002213. [p213]

H. Peng. Bioimage informatics: a new area of engineering biology. *Bioinformatics*, 24(17): 1827–1836, jul 2008. doi: 10.1093/bioinformatics/btn346. [p212, 215, 228]

H. Peng, A. Bateman, A. Valencia, and J. D. Wren. Bioimage informatics: a new category in bioinformatics. *Bioinformatics*, 28(8):1057–1057, mar 2012. doi: 10.1093/bioinformatics/bts111. [p212, 244]

A. Petersen, N. Simon, and D. Witten. Scalpel: Extracting neurons from calcium imaging data, 2017. [p240]

T. e. Poisot. The digitize package: Extracting numerical data from scatterplots. *The R Journal*, 3(1):25–26, June 2011. URL http://journal.r-project.org/archive/2011-1/RJournal_2011-1_Poisot.pdf. Number: 1. [p234]

J. Polzehl and K. Tabelow. Adaptive smoothing of digital images: The R package adimpro. *Journal of Statistical Software*, 19(1), 2007. doi: 10.18637/jss.v019.i01. [p240]

V. Prajapati. *Big data analytics with R and Hadoop*. Packt Publishing, Birmingham, online-ausg. edition, 2013. ISBN "9781782163282". URL https://api.pageplace.de/preview/DT0400.9781782163299_A24165845/preview-9781782163299_A24165845.pdf. Includes index. - Description based on online resource; title from PDF (ebrary, viewed December 30, 2013). [p244]

R Core Team. R: A language and environment for statistical computing. 2023. URL https://www.R-project.org/. [p246, 247]

A. Rabbani, A. M. Fernando, R. Shams, A. Singh, P. Mostaghimi, and M. Babaei. Review of data science trends and issues in porous media research with a focus on image-based techniques. *Water Resources Research*, 57 (10):e2020WR029472, 2021. ISSN 1944-7973. doi: 10.1029/2020WR029472. URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2020WR029472. _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2020WR029472. [p248]

C. Real. *pixelclasser: Classifies Image Pixels by Colour*, 2024. URL https://github.com/ropensci/pixelclasser. R package version 1.0.0. [p246]

Ren and Malik. Learning a classification model for segmentation. In *Proceedings Ninth IEEE International Conference on Computer Vision*. IEEE, 2003. doi: 10.1109/iccv.2003.1238308. [p215]

J. Rittscher. Characterization of biological processes through automated image analysis. *Annual Review of Biomedical Engineering*, 12(1):315–344, July 2010. ISSN 1545-4274. doi: 10.1146/annurev-bioeng-070909-105235. [p228, 230, 244]

A. S. G. Robotham, D. S. Taranu, R. Tobar, A. Moffett, and S. P. Driver. Profit: Bayesian profile fitting of galaxy images. *Monthly Notices of the Royal Astronomical Society*, 466(2): 1513–1541, Nov. 2016. ISSN 1365-2966. doi: 10.1093/mnras/stw3039. [p240]

E. Roesch, J. G. Greener, A. L. MacLean, H. Nassar, C. Rackauckas, T. E. Holy, and M. P. H. Stumpf. Julia for biologists. *Nature Methods*, 20(5):655–664, apr 2023. doi: 10.1038/s41592-023-01832-z. [p213]

P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, Nov. 1987. ISSN 0377-0427. doi: 10.1016/0377-0427(87)90125-7. [p224]

M. Ruhe, W. Dammermann, S. Lüth, M. Sowa, P. Schierack, P. M. Deckert, and S. Rödiger. Effect of cryopreservation on the formation of dna double strand breaks in human peripheral blood mononuclear cells. *Journal of Cellular Biotechnology*, 4(1–2):67–73, Jan. 2019. ISSN 2352-3697. doi: 10.3233/jcb-189006. [p241]

P. Russell, K. Fountain, D. Wolverton, and D. Ghosh. Tciapathfinder: An R client for the cancer imaging archive rest api. *Cancer Research*, 78(15):4424–4426, July 2018. ISSN 1538-7445. doi: 10.1158/0008-5472.can-18-0678. [p240]

S. Rödiger, T. Friedrichsmeier, P. Kapat, and M. Michalke. RKWard: a comprehensive graphical user interface and integrated development environment for statistical analysis with R. *Journal of Statistical Software*, 49(9):1–34, 2012. doi: 10.18637/jss.v049.i09. URL https://www.jstatsoft.org/article/view/v049i09/v49i09.pdf. Number: 9 00058. [p234]

S. Rödiger, A. Böhm, and I. Schimke. Surface melting curve analysis with R. *The R Journal*, 5(2):37–53, 2013. doi: 10.32614/RJ-2013-024. URL https://journal.r-project.org/archive/2013-2/roediger-bohm-schimke.pdf. Number: 2 00011. [p213]

S. Rödiger, M. Burdukiewicz, K. Blagodatskikh, M. Jahn, and P. Schierack. R as an environment for reproducible analysis of DNA amplification experiments. *The R Journal*, 7(1):127, 2015a. doi: 10.32614/rj-2015-011. [p213]

S. Rödiger, M. Burdukiewicz, K. A. Blagodatskikh, and P. Schierack. R as an environment for the reproducible analysis of DNA amplification experiments. *The R Journal*, 7(2):127–150, 2015b. URL https://journal.r-project.org/archive/2015-1/RJ-2015-1.pdf. Number: 2 00015. [p213]

S. Rödiger, M. Liefold, M. Ruhe, M. Reinwald, E. Beck, and P. M. Deckert. Quantification of dna double-strand breaks in peripheral blood mononuclear cells from healthy donors exposed to bendamustine by an automated γh2ax assay—an exploratory study. *Journal of Laboratory and Precision Medicine*, 3:47–47, May 2018. ISSN 2519-9005. doi: 10.21037/jlpm.2018.04.10. [p234, 241, 243]

S. Sagiroglu and D. Sinanc. Big data: A review. In *2013 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, May 2013. doi: 10.1109/cts.2013.6567202. [p244]

R. Satija, J. A. Farrell, D. Gennert, A. F. Schier, and A. Regev. Spatial reconstruction of single-cell gene expression data. *Nature Biotechnology*, 33(5):495–502, May 2015. ISSN 1546-1696. doi: 10.1038/nbt.3192. URL https://www.nature.com/articles/nbt.3192. Publisher: Nature Publishing Group. [p232]

T. Schaefer. fsbrain: an R package for the visualization of structural neuroimaging data, 2024. [p240]

C. A. Schneider, W. S. Rasband, and K. W. Eliceiri. NIH Image to ImageJ: 25 years of image analysis. *Nature Methods*, 9(7):671–675, July 2012. ISSN 1548-7105. doi: 10.1038/nmeth.2089. [p213]

J. Schneider, R. Weiss, M. Ruhe, T. Jung, D. Roggenbuck, R. Stohwasser, P. Schierack, and S. Rödiger. Open source bioimage informatics tools for the analysis of DNA damage and associated biomarkers. *Journal of Laboratory and Precision Medicine*, 4(0):1–27, 2019. doi: 10.21037/jlpm.2019.04.05. URL http://jlpm.amegroups.com/article/view/5008. Number: 0. [p212, 241]

E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu. Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Transactions on Database Systems*, 42(3):1–21, July 2017. ISSN 1557-4644. doi: 10.1145/3068335. [p224]

T. Schäfer and C. Ecker. fsbrain: an R package for the visualization of structural neuroimaging data. Sept. 2020. doi: 10.1101/2020.09.18.302935. [p240]

A. Shariff, J. Kangas, L. P. Coelho, S. Quinn, and R. F. Murphy. Automated image analysis for high-content screening and analysis. *SLAS Discovery*, 15(7):726–734, Aug. 2010. ISSN 2472-5552. doi: 10.1177/1087057110370894. [p244]

S. H. Shirazi, S. Naz, M. I. Razzak, A. I. Umar, and A. Zaib. *Automated Pathology Image Analysis*, pages 13–29. Elsevier, 2018. doi: 10.1016/b978-0-12-813087-2.00026-9. [p218]

B. Smith, M. Hermsen, E. Lesser, D. Ravichandar, and W. Kremers. Developing image analysis pipelines of whole-slide images: Pre- and post-processing. *Journal of Clinical and Translational Science*, 5(1):e38, 2021. doi: 10.1017/cts.2020.531. [p215]

M. Sonka and J. M. Fitzpatrick, editors. *Handbook of medical imaging*. Number PM80 in SPIE Press monograph. SPIE Press, Bellingham, Wash., 2000. ISBN 081948119X. doi: https://doi.org/10.1117/3.831079. Includes bibliographical references and index. - Print version record. [p215, 218, 228]

A. Struyf, M. Hubert, and P. Rousseeuw. Clustering in an object-oriented environment. *Journal of Statistical Software*, 1(4), 1996. ISSN 1548-7660. doi: 10.18637/jss.v001.i04. [p224]

J. R. Swedlow and K. W. Eliceiri. Open source bioimage informatics for cell biology. *Trends in Cell Biology*, 19(11):656–660, nov 2009. doi: 10.1016/j.tcb.2009.08.007. [p212, 213, 214]

J. R. Swedlow, I. G. Goldberg, and K. W. Eliceiri. Bioimage informatics for experimental biology. *Annual Review of Biophysics*, 38(1):327–346, jun 2009. doi: 10.1146/annurev.biophys.050708.133641. [p212, 214]

J. Textor, K. Dannenberg, J. Berry, G. Burger, A. Liu, M. Miller, and I. Wortel. celltrackr: Motion trajectory analysis, 2024. URL https://CRAN.R-project.org/package=celltrackR. R package version 1.2.0. [p233]

A. Trigos, Y. Feng, T. Yang, M. Li, J. Zhu, V. Ozcoban, and M. Doyle. Spiat, 2022. [p232]

M. M. Trivedi and J. K. Mills. Centroid calculation of the blastomere from 3d z-stack image data of a 2-cell mouse embryo. *Biomedical Signal Processing and Control*, 57:101726, Mar. 2020. ISSN 1746-8094. doi: 10.1016/j.bspc.2019.101726. [p243]

E. Um, J. M. Oh, S. Granick, and Y.-K. Cho. Cell migration in microengineered tumor environments. *Lab on a Chip*, 17(24):4171–4185, 2017. ISSN 1473-0189. doi: 10.1039/c7lc00555e. [p233]

K. Ushey, J. J. Allaire, and Y. Tang. reticulate: Interface to 'python', 2024. URL https://CRAN.R-project.org/package=reticulate. R package version 1.36.1. [p213]

M. Van der Laan, K. Pollard, and J. Bryan. A new partitioning around medoids algorithm. *Journal of Statistical Computation and Simulation*, 73(8):575–584, Aug. 2003. ISSN 1563-5163. doi: 10.1080/0094965031000136012. [p224]

B. H. M. van der Velden, H. J. Kuijf, K. G. A. Gilhuijs, and M. A. Viergever. Explainable artificial intelligence (XAI) in deep learning-based medical image analysis. *Medical Image Analysis*, 79:102470, July 2022. ISSN 1361-8415. doi: 10.1016/j.media.2022.102470. URL https://www.sciencedirect.com/science/article/pii/S1361841522001177. [p248]

O. Wagih and L. Parts. gitter: A robust and accurate method for quantification of colony sizes from plate images. *G3 Genes|Genomes|Genetics*, 4(3):547–552, mar 2014. doi: 10.1534/g3.113.009431. [p240]

R. Weiss, S. Karimijafarbigloo, D. Roggenbuck, and S. Rödiger. Applications of neural networks in biomedical data analysis. *Biomedicines*, 10(7):1469, June 2022. ISSN 2227-9059. doi: 10.3390/biomedicines10071469. [p223]

H. I. Weller, A. E. Hiller, N. P. Lord, and S. M. Van Belleghem. recolorize: An R package for flexible colour segmentation of biological images. *Ecology Letters*, 27(2), Feb. 2024. ISSN 1461-0248. doi: 10.1111/ele.14378. [p241]

R. Wollman and N. Stuurman. High throughput microscopy: from raw images to discoveries. *Journal of Cell Science*, 120(21):3715–3722, Nov. 2007. ISSN 0021-9533. doi: 10.1242/jcs.013623. [p244]

I. M. Wortel, A. Y. Liu, K. Dannenberg, J. C. Berry, M. J. Miller, and J. Textor. CelltrackR: An R package for fast and flexible analysis of immune cell migration data. *ImmunoInformatics*, 1-2:100003, oct 2021. doi: 10.1016/j.immuno.2021.100003. URL https://ingewortel.github.io/celltrackR/. [p233, 245]

K. M. Yamada and M. Sixt. Mechanisms of 3d cell migration. *Nature Reviews Molecular Cell Biology*, 20(12):738–752, Oct. 2019. ISSN 1471-0080. doi: 10.1038/s41580-019-0172-9. [p233]

T. Yang, V. Ozcoban, A. Pasam, N. Kocovski, A. Pizzolla, Y.-K. Huang, G. Bass, S. P. Keam, P. J. Neeson, S. K. Sandhu, D. L. Goode, and A. S. Trigos. SPIAT: An R package for the spatial image analysis of cells in tissues. may 2020. doi: 10.1101/2020.05.28.122614. [p231, 232, 245]

Z. Yaniv, B. C. Lowekamp, H. J. Johnson, and R. Beare. SimpleITK image-analysis notebooks: a collaborative environment for education and reproducible research. *Journal of Digital Imaging*, 31(3):290–303, nov 2017. doi: 10.1007/s10278-017-0037-8. [p230]

W. Yao, C. O. Dumitru, O. Loffeld, and M. Datcu. Semi-supervised hierarchical clustering for semantic sar image annotation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(5):1993–2008, May 2016. ISSN 2151-1535. doi: 10.1109/jstars.2016.2537548. [p223]

Y. Ye, J. Hao, Y. Hou, Z. Wang, S. Xiao, Y. Luo, and W. Zeng. Generative AI for visualization: State of the art and future directions. *Visual Informatics*, 8(2):43–66, June 2024. ISSN 2468-502X. doi: 10.1016/j.visinf.2024.04.003. URL https://www.sciencedirect.com/science/article/pii/S2468502X24000160. [p248]

C. Zhao and R. N. Germain. Multiplex imaging in immuno-oncology. *Journal for ImmunoTherapy of Cancer*, 11(10):e006923, Oct. 2023. ISSN 2051-1426. doi: 10.1136/jitc-2023-006923. [p231]

Tim Brauckhoff
Brandenburg University of Technology Cottbus - Senftenberg
Institute of Biotechnology
Senftenberg, Germany
Eberhard Karls Universität Tübingen
Department of Computer Science
Tübingen, Germany
ORCiD: 0009-0002-0142-7017
brauctile@disroot.org

Julius Rublack
University of Münster
Faculty of Biology
Münster, Germany
juliusrublack@web.de

Stefan Rödiger
Brandenburg University of Technology Cottbus - Senftenberg
Institute of Biotechnology
Senftenberg, Germany
ORCiD: 0000-0002-1441-6512
stefan.roediger@b-tu.de

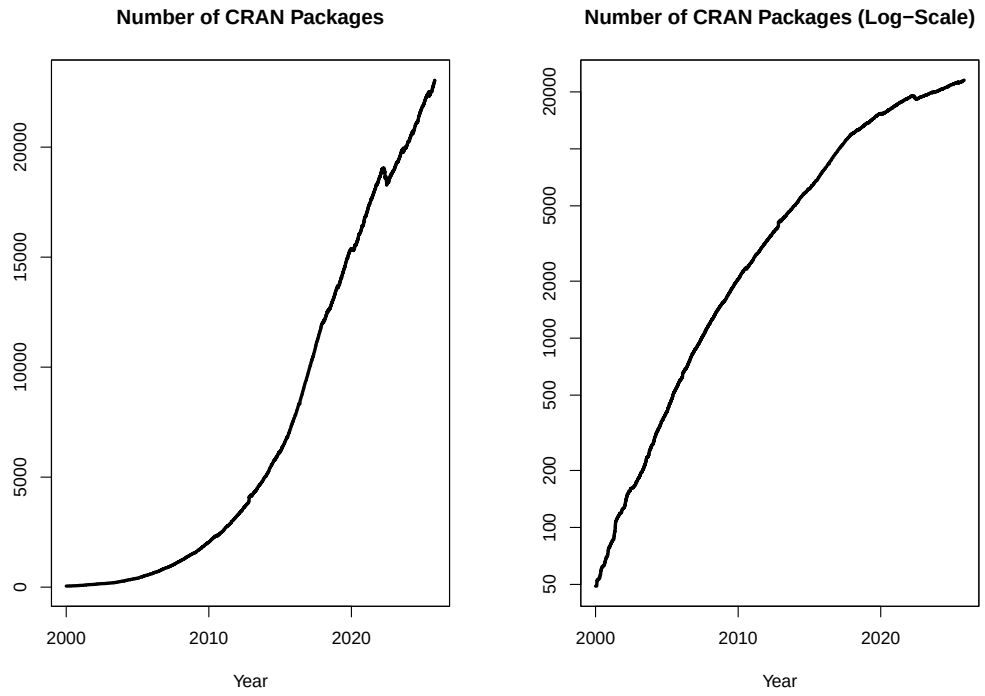
Changes on CRAN

2025-07-01 to 2025-09-30

by Kurt Hornik, Uwe Ligges, and Achim Zeileis

1 CRAN growth

In the past 3 months, 562 new packages were added to the CRAN package repository. 165 packages were unarchived, 287 were archived and 1 had to be removed. The following shows the growth of the number of active packages in the CRAN package repository:



On 2025-09-30, the number of active packages was around 22791.

2 CRAN package submissions

From July 2025 to September 2025 CRAN received 7923 package submissions. For these, 12326 actions took place of which 9289 (75%) were auto processed actions and 3037 (25%) 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	2422	673	1657	107	0	2856	981	392
manual	1212	0	20	9	19	1406	289	80

These include the final decisions for the submissions which were

	archive	publish
auto	2348 (30.1%)	2607 (33.4%)
manual	1204 (15.4%)	1652 (21.1%)

where we only count those as *auto* processed whose publication or rejection happened automatically in all steps.

3 CRAN mirror security

Currently, there are 93 official CRAN mirrors, 77 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).

4 CRAN Task View Initiative

There is one new task view:

- Anomaly Detection: Maintained by Priyanga Dilini Talagala, Rob J. Hyndman, Gaetano Romano.

Currently, there are 49 task views (see `https://CRAN.R-project.org/web/views/`), with median and mean numbers of CRAN packages covered 112 and 124, respectively. Overall, these task views cover 5051 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

Donations and members

Membership fees and donations received between 2025-09-17 and 2025-11-06.

Donations

Qualitas AG (Switzerland); Charles Geyer (United States)

Supporting benefactors

Zubin Dowlaty (United States)

Supporting institutions

Roseburg Forest Products, Springfield (United States)

Supporting members

Douglas Adamoski (Brazil); Jeremy Allen (United States); Tim Appelhans (Germany); Chris Aragao (United States); Gordon Blunt (United Kingdom); Terry Cox (United States); Gergely Daroczi (Hungary); David Freedman (United States); Keita Fukasawa (Japan); Arthur Gailes (United States); Yehonatan Mordechai Gidean (Israel); Rob Hyndman (Australia); Jan Herman Kuiper (United Kingdom); Amanuel Medhanie (United States); Bogdan-Alexandru Micu (Luxembourg); Guido Möser (Germany); Benjamin Schneider (United States); David Sides (United States); Rachel Smith-Hunter (United States); Matteo Starri (Italy); Samuel Stewart (Canada); Derrick Um (United States).

Torsten Hothorn
Universität Zürich
Switzerland
ORCiD: 0000-0001-8301-0471
Torsten.Hothorn@R-project.org