# The R Journal

Volume 16/4, December 2024

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.



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

# Editorial

*by Mark P.J. van der Loo*

On behalf of the editorial board, I am pleased to present Volume 16 Issue 4 of the R Journal.

## In this issue

News from CRAN, RForwards, and Bioconductor are included in this issue.

This issue features 10 contributed research articles relating to R packages on a diverse range of topics. All packages are available on CRAN. Supplementary material with fully reproducible code is available for download from the Journal website. Topics covered in this issue are the following.

### Time series and Survival Analyses

- Disaggregating Time-Series with Many Indicators: An Overview of the DisaggregateTS Package
- survivalSL: an R Package for Predicting Survival by a Super Learner

### Modeling and Inference

- IRTest: An R Package for Item Response Theory with Estimation of Latent Distribution
- Canonical Correlation Analysis of Survey Data: The SurveyCC R Package
- LMest: An R Package for Estimating Generalized Latent Markov Models
- MultiStatM: Multivariate Statistical Methods in R
- Calculating Standardised Indices Using SEI

### Computational Statistics

- GPUmatrix: Seamlessly harness the power of GPU computing in R
- Splinets – Orthogonal Splines for Functional Data Analysis

### Graphics

- UpAndDownPlots: An R Package for Displaying Absolute and Percentage Changes

*Mark P.J. van der Loo*
*Statistics Netherlands and Leiden University*

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

# survivalSL: an R Package for Predicting Survival by a Super Learner

*by Camille Sabathe and Yohann Foucher*

**Abstract** The R package **survivalSL** contains a variety of functions to construct a super learner in the presence of censored times-to-event and to evaluate its prognostic capacities. Compared to the available packages, we propose additional learners, loss functions for the parameter estimations, and user-friendly functions for evaluating prognostic capacities and predicting survival curves from new observations. We performed simulations to describe the value of our proposal. We also detailed its usage by an application in multiple sclerosis. Because machine learning is increasingly being used in predictive studies with right-censoring, we believe that our solution can be useful for a large community of data analysts, beyond this clinical application .

## 1 Introduction

In clinical practice, the prediction of the probability that a subject will experience an event is often of interest. For instance, for patients with multiple sclerosis under first-line treatment, the prediction of disease progression would result in the early identification of non-responders and help in deciding the switch to second-line treatment. However, the time-to-disease progression is often not observable for all subjects because of a lack of follow-up, i.e., right-censoring.

Several regressions can be used for prediction from right-censored data. The most popular is probably the proportional hazards (PH) model (Cox, 1972). The corresponding baseline hazard function can be obtained by assuming parametric distributions (Andersen et al., 1985), nonparametric estimators (Lin, 2007) or other flexible approaches such as splines (Rosenberg, 1995). Penalized PH models have also been proposed, particularly useful for high-dimensional data (Goeman, 2009). The accelerated failure times (AFT) approach also constitutes an alternative to the PH assumption (Wei, 1992).

In parallel to these regression-based methods, machine learning is increasingly used. Support-vector machines (Shivaswamy et al., 2007), neural networks (Faraggi and Simon, 1995), or even random forests (Ishwaran et al., 2008) have been successfully developed for censored time-to-event data. Ensemble learning allows us to combine regressions and algorithms by minimizing the cross-validated loss function (Breiman, 1996). SL was first proposed by van der Laan et al. (2007) and van der Laan and Dudoit (2003) from the theory of unified loss-based estimation.

Polley et al. (2011) and Polley and van der Laan (2011) extended the SL to right-censored data. Two R-based solutions are available on GitHub repositories. The **SuperLearner-Survival** repository is related to the work by Golmakani and Polley (2020). It allows us to obtain the linear predictor of a PH regression. The **survSuperLearner** package was developed by Westling (2021) with additional learners: several parametric PH models (Exponential, Weibull, log-logistic, and piecewise constant hazard), generalized additive Cox regression, and random survival forest.

In this paper, we aim to extend these solutions by i) additional learners (accelerated failure times, neural networks, penalized regressions), ii) several loss functions for the estimation of the parameters (Brier score, negative binomial log-likelihood, etc.) and iii) user-friendly S3 methods for evaluating the predictive capacities, as well as predicting survival curves from new observations (Section 1). In the second section, we propose a simulation-based study to compare the performances of our proposal with that of the alternative solutions. In the third section, we detail its usage by an application in multiple sclerosis. Finally, we discuss the strengths, weaknesses and perspectives of our solution.

## 2 Methods

### 2.1 Notations

For a subject $i$ in a sample of $N$ independent subjects $(i = 1, ..., N)$, we denoted the time-to-event by $T_i^\star$. Right-censoring leads the observation of $T_i = \min(T_i^\star, C_i)$, where $C_i$ is the censoring time. Let $D_i = \mathbb{1}\{T_i^\star \leq C_i\}$ be the event indicator. The survival function at time $t$ for a subject with the characteristics $Z_i$ at baseline is defined by $S(t \mid Z_i) = \mathbb{P}(T_i^\star > t \mid Z_i)$.

### 2.2 Definition of the super learner

Let $S_{sl}(.)$ be the survival function obtained by the SL such as $S_{sl}(t \mid Z_i) = \sum_{m=1}^{M} w_m \times S_m(t \mid Z_i)$, where $S_m(.)$ is the survival function obtained by its $m$th learner ($m = 1, ..., M$), and $w = (w_1, ..., w_M)$ are the corresponding weights with respect to $\sum_1^M w_m = 1$ and $0 \leq w_m \leq 1$. The estimations of the learners and the weights can be obtained by respecting the following steps (Polley and van der Laan, 2010):

1. Estimate the $M$ models or algorithms on the entire sample as usual. The procedure of estimation can be different for each model. For instance, a parametric model may be fitted by maximizing the full likelihood, a Cox regression by maximizing the partial likelihood and estimating the baseline survival function with Breslow method, or a random survival forest by maximizing the between-node survival differences according to the log-rank statistic.
2. By using the $M$ models or algorithms estimated in the step #1 ($m = 1, ..., M$), predict the values $\hat{S}_m(u \mid Z_i)$ for each subject $i = 1, ..., N$ and for a vector of $U$ times ($u = u_1, ..., u_U$) and stack the values in a 3-dimensional array of size ($N \times U \times M$).
3. Split the sample into a training and validation sample according to a V-fold cross-validation, i.e., $V$-equal size and randomly selected groups. Let $T(v)$ be the $v$th training sample and $V(v)$ be the corresponding validation sample ($v = 1, .., V$).
4. For the $v$th fold, estimate the $M$ models or algorithms from $T(v)$. The learners based on the estimation of hyperparameters should be tuned for each of the $V$ folds. Note that this tunning step is often based on cross-validation and should be performed for each of the $V$ folds. Regarding the corresponding computational burden, a compromise consists of using the hyperparameters estimated at the step #1 (Le Borgne et al., 2021).
5. Compute the corresponding predicted survival rates for the individuals of the validation sample $V(v)$ and the $U$ times.
6. Stack the predictions $\tilde{S}_m(u \mid Z_i)$ obtained in the step #5 in a 3-dimensional array of size ($N \times U \times M$).
7. Determine the loss function $\mathbb{L}(w \mid \tau)$ for quantifying the gap of the predictions $\tilde{S}_{sl}(\tau \mid Z_i) = \sum_{m=1}^{M} w_m \tilde{S}_m(\tau \mid Z_i)$ and the observations $(T_i, D_i)$ for a prognostic up to $\tau$. The loss functions available in the package are listed in the next subsection.
8. Estimate the weights $w$ by minimizing the loss function for a prognostic up to a time $\tau$:

$$\hat{w} = \arg\min_{w} \mathbb{L}(w \mid \tau)$$

9. Combine $\hat{w}_m$ with $\hat{S}_m(u \mid Z_i)$ to obtain the final super learner:

$$\hat{S}_{sl}(t \mid Z_i) = \sum_{m=1}^{M} \hat{w}_m \hat{S}_m(t \mid Z_i)$$

### 2.3 Available loss functions for estimating the weights

Several loss functions $\mathbb{L}(w \mid \tau)$ are available in the package. The Brier score (BS) for right-censored data and a prediction at time $\tau$ is defined by:

$$BS(w \mid \tau) = \frac{1}{N} \sum_i \left[ \frac{\tilde{S}_{sl}(\tau \mid Z_i)^2 \mathbb{1}\{T_i \leq \tau, D_i = 1\}}{\hat{G}(T_i)} + \frac{(1 - \tilde{S}_{sl}(\tau \mid Z_i))^2 \mathbb{1}\{T_i > \tau\}}{\hat{G}(\tau)} \right]$$

where $\hat{G}(u) = \mathbb{P}(C_i > u)$ is the survival function of the censoring time (van Houwelingen et al., 2012). The corresponding $\tau$-restricted integrated BS is $RIBS(w \mid \tau) = \int_0^\tau BS(u)du$. The integrated BS equals $RIBS(w \mid \max(T_i^\star))$. We also considered the negative binomial log-likelihood (BLL), which is defined for a prognostic at time $\tau$ as follows:

$$BLL(w \mid \tau) = \frac{1}{N} \sum_i \left[ \frac{\log(1 - \tilde{S}_{sl}(\tau \mid Z_i)) \mathbb{1}\{T_i \leq \tau, D_i = 1\}}{\hat{G}(T_i)} + \frac{\log(\tilde{S}_{sl}(\tau \mid Z_i)) \mathbb{1}\{T_i > \tau\}}{\hat{G}(\tau)} \right]$$

We also proposed the corresponding integrated and $\tau$-restricted integrated versions. Additionally, we implemented the concordance index (CI) as defined by Uno et al. (2011):

$$CI(w \mid \tau) = \frac{\sum_i \sum_j \mathbb{1}\{T_j < T_i, \tilde{S}_{sl}(\tau \mid Z_j) > \tilde{S}_{sl}(\tau \mid Z_i)\} D_j}{\sum_i \sum_j \mathbb{1}\{T_j < T_i\} D_j}$$

Finally, we considered the area under the time-dependent ROC curve, i.e., the sensitivity (SE) against one minus the specificity (SP) at each threshold $\eta$ (Hung and Chiang, 2010):

$$SE_\eta(w \mid \tau) = \frac{\sum_i D_i \mathbb{1}\{T_i \leq \tau, \tilde{S}_{sl}(\tau \mid Z_i) > \eta\}/(1-\hat{G}(T_i))}{\sum_i D_i \mathbb{1}\{T_i \leq \tau\}/(1-\hat{G}(T_i))}$$

$$SP_\eta(w \mid \tau) = \frac{\sum_i \mathbb{1}\{T_i > \tau, \tilde{S}_{sl}(\tau \mid Z_i) \leq \eta\}}{\sum_i \mathbb{1}\{T_i > \tau\}}$$

### 2.4 Available learners

Several models or algorithms are proposed in the **survivalSL** package:

- `Parametric AFT models.` They can be considered with Weibull, Gamma or generalized Gamma distributions (**flexsurv** package). The parameters are estimated by likelihood maximization.
- `Parametric PH models.` They can be considered with Exponential or Gompertz distributions (**flexsurv** package). The parameters are estimated by likelihood maximization.
- `Spline-based PH models.` We considered the spline-based survival model proposed by Royston and Parmar (2002). To estimate the baseline distribution, it involves one hyperparameter: the number of internal knots of the natural cubic spline, with a default grid search $k = \{1, 2, 3, 4\}$. The parameters are estimated by likelihood maximization (**flexsurv** package).
- `Semi-parametric PH models.` It corresponds to a PH model (**coxph** function of the **survival** package) with a non-parametric baseline hazard function estimated by using the Breslow estimator (Lin, 2007). This approach can be used with or without a forward selection of covariates by using the AIC.
- `Penalized PH models.` For high-dimensional data and/or covariate selection, three penalties can be used for the previous semiparametric PH model: Lasso, Ridge, or Elastic-Net (**glmnet** package). The quantitative covariates are transformed with B-splines to relax the log-linear assumption. The hyperparameters $\lambda$ of the Lasso regression allows for selection of the covariates, while it allows for the shrinkage of the regression coefficients in the Ridge regression. The Elastic-Net allows the two penalties. By default, we implemented the grid search proposed by Simon et al. (2011) for $\lambda$ and $\{0.1, 0.2, ..., 0.9\}$ for $\alpha$. The Lasso penalization corresponds to $\alpha = 0$, while the Ridge penalization corresponds to $\alpha = 1$.
- `Random survival forests.` This ensemble tree method extends Breiman's random forest framework to right-censored data (Ishwaran et al., 2008). It allows estimation of the cumulative hazard function in each node by using the nonparametric estimator of survival proposed by Aalen (1978). It involves three hyperparameters (**randomSRC** package): number of covariates to test for splitting the node, minimal number of events per terminal node, and number of trees. The default grid search is `mtry` = $\{1, 2, ..., 2 + 0.5 \times$ number of covariates$\}$, `nodesize` = $\{2, 4, 6, 10, 20, 30, 50, 100\}$ and `ntree` = 500, respectively.
- `Survival neural networks.` We implemented the method proposed by Biganzoli et al. (1998). It allows using the **nnet** package for estimating a feed forward neural network model for partial logistic regression. It involves four hyperparameters: the length of the time intervals, the number of units in the hidden layer, the weight decay, the maximum number of iterations, and the maximum allowable number of weights. The optimal combination is estimated by cross-validation and the previous possible loss functions. The default grid search is composed by `inter` = 1, `size` = $\{2, 4, 6, 8, 10\}$, `decay` = $\{0.001, 0.01, 0.02, 0.05\}$, `maxit` = 100, and `MaxNWts` = 10000, respectively.

## 3 Simulations

### 3.1 Design

We simulated 1000 data sets for each scenario. The times-to-event were generated from Weibull distributions with PH assumption. The censoring times were generated from uniform distributions to obtain a 40% censoring rate. We studied two sample sizes for learning (200 and 500), while the validation samples were composed of 500 subjects. We proposed two contrasting scenarios (Figure **??** in Appendix):

- A simple scenario with 6 independent covariates, 2 were continuous and 4 qualitative. Among them, 1 continuous and 2 qualitative covariates were associated with the times-to-event distribution without any log-linearity issue.

- A complex scenario with 23 correlated covariates: 12 continuous covariates (several nonlinear relationships, two effects were step functions, two were quadratic functions, five were log-linear), 11 qualitative covariates and 1 interaction.

We compared our proposed SL with each included learner, the perfectly specified PH model (i.e., the model used to generate the times-to-event, the only estimation consisting of the regression coefficients), and the SL obtained by the **survSuperLearner** package (Westling, 2021). We decided to use the package proposed by Westling (2021) as a comparator because it includes additional learners compared to the functions proposed by Golmakani and Polley (2020). The SL weights were estimated by minimizing the integrated BS (IBS) for all the methods.

In the SL proposed by Westling (2021), we included the 5 possible learners: the PH model with the Breslow estimator, the Exponential PH model, the random survival forests, the Lasso PH model and the PH model with the univariate selection of covariates ($p < 0.05$). We used the default grid of the hyperparameters.

In our proposed SL, we included the 7 possible learners: the PH model with the Breslow estimator, the same model with forward selection based on AIC minimization, the Elastic-Net PH model with B-splines on quantitative covariates, the random survival forest, the Exponential PH model, the Gamma distribution-based AFT model, and the survival neural network. Note that we did not consider the Weibull PH model since it was used for data generation. We used the default grid of the hyperparameters.

### 3.2 Results of the simple scenario

Figure 1 presents the distributions of the IBS over the learning (on the left) and validation samples (on the right). The related `R` code is available in Appendix. Consider the situation where $N = 200$ for learning and $N = 500$ for validation (the first two plots on the top). The perfectly specified model had the lower IBS. This result was expected since it constitutes the gold standard to reach by the other methods. The Elastic-Net PH regression outperformed all the other approaches in the learning samples. This result explained the important weight of this method, approximately 25% in our SL, as illustrated in Figure 5 in Appendix. Nevertheless, this apparent result was not observed in the validation samples, illustrating overfitting.

In both the learning and the validation samples, our proposed SL, the PH model with the Breslow estimator, and the same model with forward selection based on the AIC had the best performances, higher than that of the SL proposed by Westling (2021).

The survival neural network and the PH model with an Exponential distribution were associated with larger values of the IBS in both the learning and the validation samples. Comparatively, the results were closed across the other methods. In the validation samples, the highest variability of the IBS was observed for the survival neural network.

When $N = 500$ in both the learning and the validation samples (the two bottom plots of Figure 1), the results were close. However, one can note that the means and variances of the IBS were lower.

#### Results of the complex scenario

The related `R` code is available in Appendix. As illustrated in Figure 2, regarding the results in the validation samples, our proposed SL and the Elastic-Net PH regression outperformed the other models/algorithms. More precisely, the mean of the IBS was lower for the Elastic-Net PH model for a learning sample size of 200. The value of our proposed SL was close. When $N = 500$ for learning, the mean of the IBS was lower for our SL, but that of the Elastic-Net PH model was close. Nevertheless, regardless of the sample size for learning, our SL was associated with lower variance in the IBS.

Among the other differences compared to the simple context, the random survival forests and the survival neural networks performed better than the other model-based predictors, except the Elastic-Net PH model.

In both the learning and the validation samples, and regardless the sample size for learning, our proposed SL had slightly higher performances compared to the results of the SL proposed by Westling (2021).

### 3.3 Running times

We further explored the time required to estimate a SL depending on the sample size and the number of predictors. For this purpose, we simulated data according to the complex scenario (Figure **??** in Appendix). We used a 20-fold cross-validation, tested sample sizes from 500 to 3500, and reduced

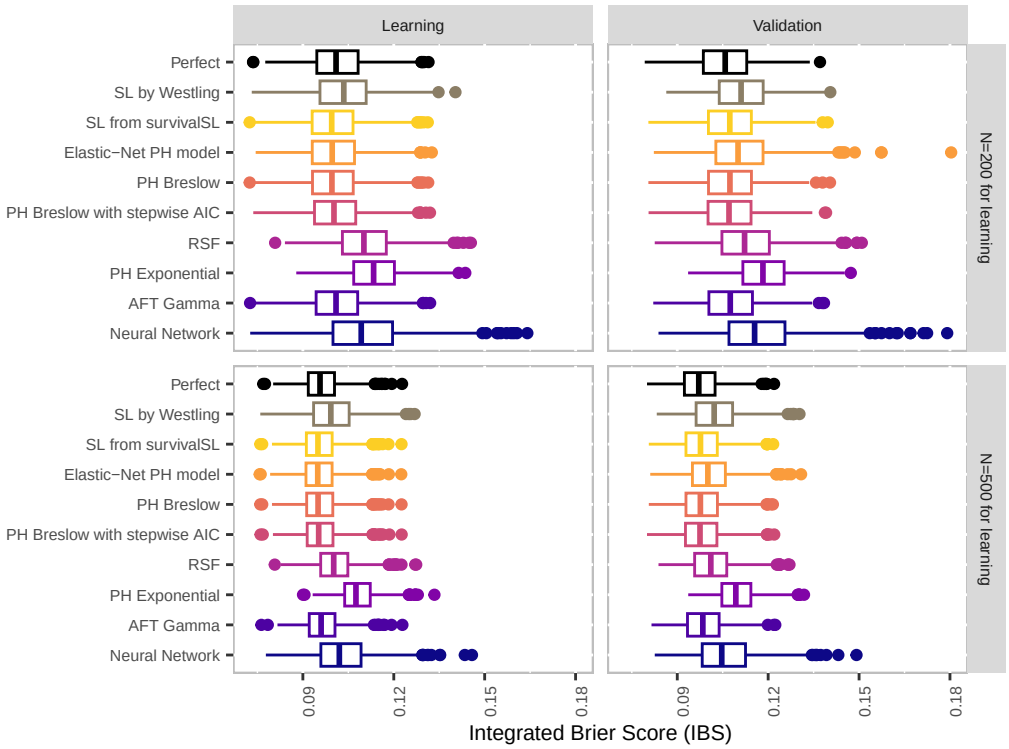



**Figure 1:** Simulation results in the simple context.

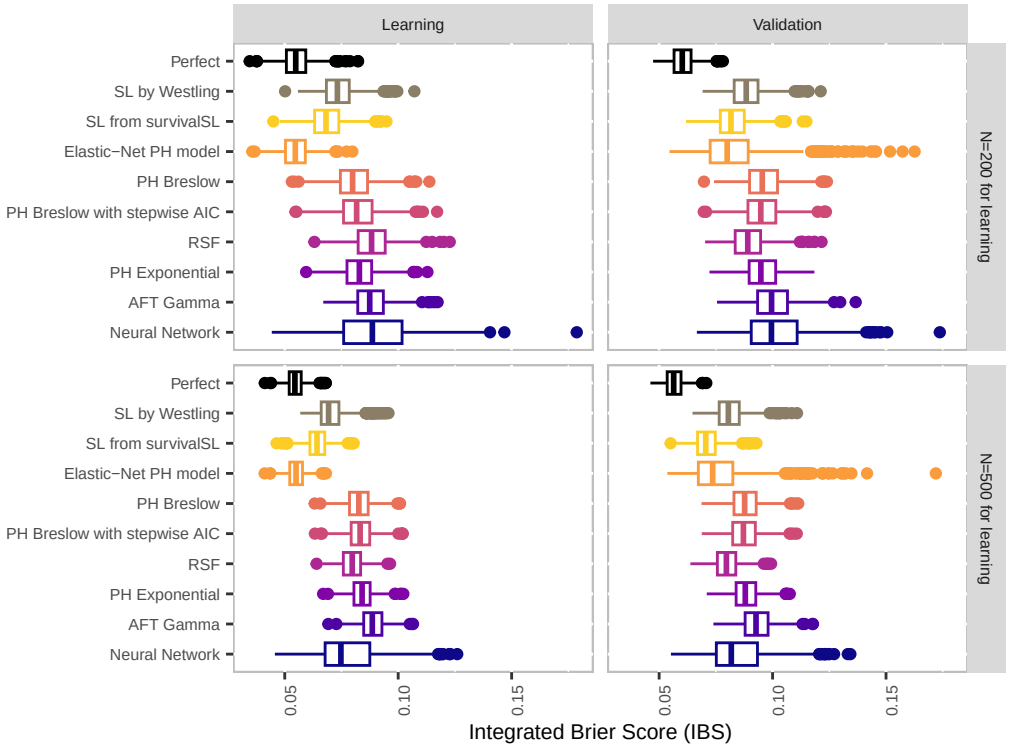



**Figure 2:** Simulation results in the complex context.

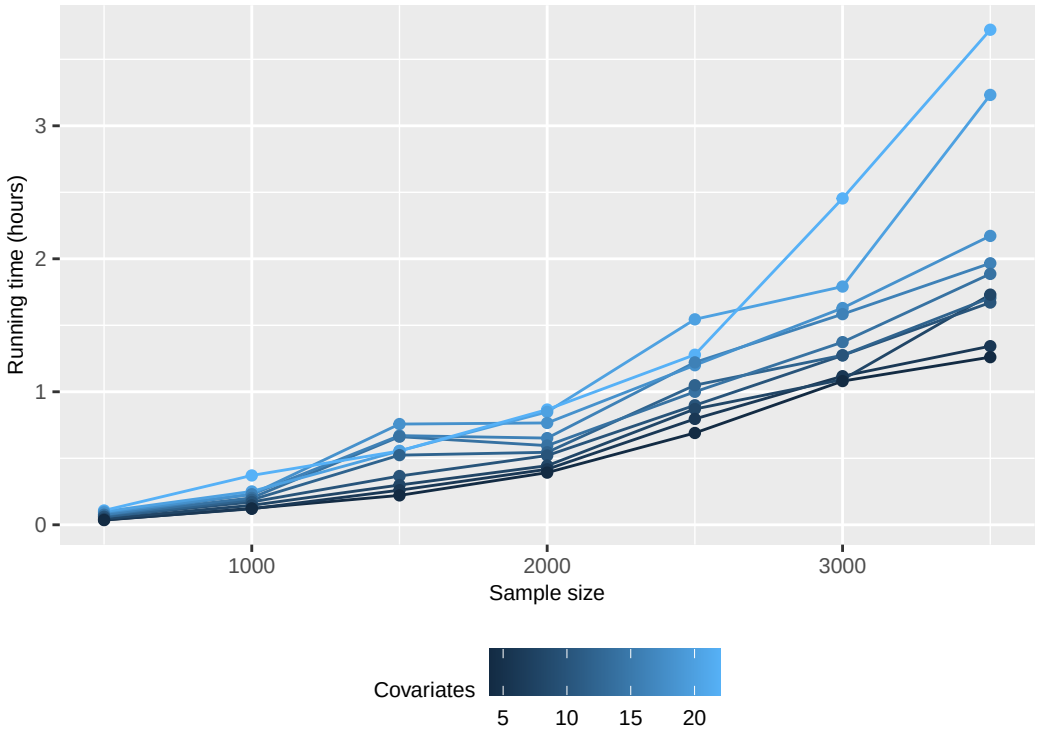



**Figure 3:** Running times according to the sample size and the number of covariates. Results obtained for a MacBook Pro 2.6 GHz Intel Core i7 6 cores.

the number of predictors from 22 to 4. The results are illustrated by Figure 3. If we considered 20 predictors, the running times increased exponentially from 0.10 ($N = 500$) to 3.23 hours ($N = 3500$). For 10 predictors, the times increased from 0.06 to 1.67 hours.

## 4 Usage

### 4.1 The main functions of the package

The `survivalSL` function fits the SL. It is mainly based on the following arguments:

- `formula`: A formula object, with the response on the left, and the predictors on the right. The response must be a survival object as returned by the `Surv` function..
- `methods`: The names of the learners included in the SL. At least two models/algorithms must be included. The complete list of candidates is described in Table 1.
- `data`: The data frame in which to look for the variables related to the follow-up time, the event indicator, and the covariates.
- `metric`: The loss function used to estimate the weights of the learners.
- `pro.time`: The prognostic time used for the Brier score, the negative binomial log-likelihood or the corresponding restricted versions.
- `cv`: The number of cross-validated samples to estimate the weights of the learners.
- `param.tune`: Optional list to define the grid search of the hyperparameter(s) or to set the value(s) of the hyperparameter(s). If `NULL`, the default grid is used.
- `seed`: A numeric value for random seed for reproducibility. The default is `NULL`.

Note that a method based on hyperparameters can be used several times with different values of hyperparameters. When the hyperparameters are specified by the user, the argument `param.tune` must be a list with their value(s) and name(s). For a user-friendly manipulation of the resulting `sltime` object, we proposed several S3 methods listed in Table 2. They allow for evaluating the calibration and discrimination capacities, or predicting survival curves. These functions can be applied to new data sets. Similar S3 methods are available for other learners (`LIB_AFTgamma`, `LIB_COXlasso`, etc.).

**Table 1:** List of possible learners (#: Breslow's estimation of the baseline hazard function). The last column lists the learners included in the survSuperLearner proposed by Westling (2021). This package additionally considers generalized additive Cox regression and piecewise constant hazard regression.

| Name | Description | Westling |
|---|---|---|
| LIB_COXall | Proportional hazards (PH) model with all covariates (#) | Yes |
| LIB_COXaic | PH model with covariate selection by AIC minimization (#) | No |
| LIB_COXen | PH model with B-spline for the quantitative covariates and Elastic-Net penalization (#, hyperparameters: alpha and lambda) | Yes |
| LIB_COXlasso | PH model with B-spline for the quantitative covariates and Lasso penalization (#, hyperparameter: lambda) | Yes |
| LIB_COXridge | PH model with B-spline for the quantitative covariates and Ridge penalization (#, hyperparameter: lambda) | Yes |
| LIB_AFTgamma | Accelerated failure times (AFT) model with Gamma distribution | No |
| LIB_AFTggamma | AFT model with generalized Gamma distribution | No |
| LIB_AFTllogis | AFT model with log-logistic distribution | No |
| LIB_AFTweibull | AFT model with Weibull distribution | Yes |
| LIB_PHexponential | Parametric PH model with Exponential distribution | Yes |
| LIB_PHgompertz | Parametric PH model with Gompertz distribution | Yes |
| LIB_PHspline | PH model with natural cubic spline as baseline distribution (hyperparameters: k) | No |
| LIB_RSF | Random survival forest (hyperparameters: nodesize, mtry and ntree) | Yes |
| LIB_PLANN | One-layer survival neural network (hyperparameters: n.nodes, decay, batch.size and epochs) | No |

**Table 2:** List of S3 functions applicable to an sltime object

| Function | Description |
|---|---|
| plot.sltime | To obtain a calibration plot. |
| predict.sltime | To predict the survival probabilities from the SL and the included learners. |
| summary.sltime | To obtain metrics describing the prognostic capacities of the SL and the included learners. |
| print.sltime | To print the learners and their weights |

### 4.2 Application to multiple sclerosis

This section is not intended to propose guidelines for data analysis. It only consists of illustrating the use of the **survivalSL** package. Beside the present application, a recent work proposed recommendations for implementing such a SL (Phillips et al., 2023).

The following application is related to the prediction of the time to disease progression from prognostic factors collected one year after the initiation of a first-line treatment, which is the baseline of the cohort ($T^\star = 0$). We also compared the prognostic capacities of the obtained SL with those of the modified Rio score (Sormani et al., 2013).

`Data description.` We detailed the implementation of the previous function with an application based on the OFSEP cohort, which was composed of 1300 simulated patients with multiple sclerosis. The simulations were performed from the observed cohort to obtain the present shared data set which complies with the General Data Protection Regulation of the European Union. More precisely, the prognostic factors were first simulated one by one, i.e., using the factors already simulated to obtain the next one. These simulations were based on linear or logistic models for continuous or categorical factors, respectively. Finally, the time to disease progression was simulated by a Weibull PH model, which explains why it was not considered among the learners.

```
library("survivalSL")
data(dataOFSEP); head(dataOFSEP)
```

```
#>        time event age duration period gender relapse edss t1 t2 rio
#> 1 2.1195051     1  34     1113      0      1       1  low  0  0   1
#> 2 0.8160995     1  37     1296      0      1       1 high 1+  0   1
#> 3 1.9709546     1  33      995      1      1       0  low  0  0   0
#> 4 2.5881311     1  35      858      1      1       0  low  0  0   0
#> 5 1.4726224     1  31      759      1      0      2+ miss 1+  0   2
#> 6 1.6970962     1  40     1642      1      1       0 high  0  0   0
```

The first two columns of the `dataOFSEP` table correspond to the time to disease progression (`time`) and the `event` indicator equals 1 for patients with events and 0 for right-censored patients. The time to disease progression was defined by the minimum among the time to first relapse, the time to increase of Expanded Disability Status Scale (EDSS), and the time to switch for inefficacy. The EDSS ranges from 0 to 10 (the higher levels of disability) and is based on an examination by a neurologist.

The other columns were the available predictors one year after the initiation of first-line therapy. The third and fourth columns were two quantitative predictors: patient `age` (in years) and disease `duration` (in days). The other columns were qualitative predictors: calendar `period` (1 if between 2014 and 2018, and 0 otherwise), `gender` (1 if women), diagnosis of `relapse` since treatment initiation (1 if at least one event, and 0 otherwise), and EDSS level (`miss` if missing, `low` if level between 0 and 2, and `high` otherwise). The variables `t1` and `t2` were related to the lesions observed from magnetic resonance imaging, namely, the new gadolinium-enhancing T1 lesion (`miss` if missing, `1+` if at least one lesion, and `0` otherwise) and the new T2 lesion (1 if at least one lesion, and 0 otherwise), respectively.

The last column corresponded to the modified Rio score, i.e., the sum of the relapses (0 for none, 1 for one event, and 2 for at least 2 events) and the new T2 lesion (0 for none, and 1 for at least one lesion) observed since the treatment initiation (Sormani et al., 2013). We transformed the categorical predictors into binary variables.

```
dataOFSEP$relapse.1 <- 1*(dataOFSEP$relapse=="1")
dataOFSEP$relapse.2 <- 1*(dataOFSEP$relapse=="2+")
dataOFSEP$edss.1 <- 1*(dataOFSEP$edss=="low")
dataOFSEP$edss.2 <- 1*(dataOFSEP$edss=="high")
dataOFSEP$t1.1 <- 1*(dataOFSEP$t1=="0")
dataOFSEP$t1.2 <- 1*(dataOFSEP$t1=="1+")
```

`Super learner training.` We randomly selected two-thirds of the sample to train the SL, and the other third was further used for validation in the next subsection. We set the seed to ensure the results replicability.

```
set.seed(117)
dataOFSEP$train <- 1*rbinom(n=dim(dataOFSEP)[1], size=1, prob=2/3)
dataTRAIN <- dataOFSEP[dataOFSEP$train==1,]
dataVALID <- dataOFSEP[dataOFSEP$train==0,]
```

We fitted the SL with the following learners: PH model with Gompertz baseline hazard function, AFT model with generalized Gamma distribution, Elastic-Net PH model with nonparametric baseline hazard function (Breslow estimator) and B-spline transformations of the quantitative variables, and random survival forest. We used the default grids to estimate the hyperparameters. The weights were estimated to minimize the concordance index at 2 years with a 30-fold cross-validation. Note that 851 patients were included in the training sample, and 651 events were observed. In such a situation, Phillips et al. (2023) recommended at least 20 folds.

```
.f <- Surv(time, event) ~ age + duration + period + gender +
   relapse.1 + relapse.2 + edss.1 + edss.2 + t1.1 + t1.2
sl1 <- survivalSL(formula=.f, metric="uno_ci",  data=dataTRAIN,
      methods=c("LIB_COXen", "LIB_PHspline", "LIB_AFTggamma", "LIB_RSF"),
      cv=30, optim.local.min=TRUE, show_progress=FALSE, seed=117)
print(sl1, digits=4)

#> The contribution of the learners:
#>
#>         leaners weights
#> 1     LIB_COXen  0.1775
#> 2  LIB_PHspline  0.2519
#> 3 LIB_AFTggamma  0.3039
#> 4       LIB_RSF  0.2667
#>
#> Minimum of the 30-fold CV of the metric uno_ci:0.6851.
```

Moreover, the package allows the user to propose a more precise grid for the hyperparameter-based methods. For instance for the spline-based PH model, the user may precise his/her own grid search:

```
.tune <- vector("list",4)
.tune[[2]] <- list(k=1:6)
sl2 <- survivalSL(formula=.f, metric="uno_ci",  data=dataTRAIN,
      methods=c("LIB_COXen", "LIB_PHspline", "LIB_AFTggamma", "LIB_RSF"),
      cv=30, optim.local.min=TRUE, param.tune=.tune, show_progress=FALSE, seed=117)
print(sl2, digits=4)

#> The contribution of the learners:
#>
#>         leaners weights
#> 1     LIB_COXen  0.2486
#> 2  LIB_PHspline  0.3340
#> 3 LIB_AFTggamma  0.1773
#> 4       LIB_RSF  0.2401
#>
#> Minimum of the 30-fold CV of the metric uno_ci:0.684.
```

On a MacBook Pro 2.6 GHz Intel Core i7 6 cores, the running times to estimate `sl1` and `sl2` were 11.0 and 13.2 minutes, respectively. Note that the optional argument related to the definition of the grid may be used to estimate the hyperparameters from the entire sample and use the related values in each fold of the steps #4 and #5 of the super learner estimation. This strategy may be associated with overfitting, but an important decrease in the computation time and the fluctuation of the results. Indeed, regarding the random allocation of the individuals in the folds at the steps #1 (for estimating the learners from the entire training sample) and #3 (for estimating the prediction matrix includes in the loss function), the results depend on the generating pseudo random numbers. To illustrate this fluctuation, the `sl2` estimation was performed for several seeds in Appendix. The weights of the learners significantly varied but the predictive performances of the super learners were stable.

`Super learner validation.` We further studied the prognostic capacities of this algorithm, always up to 2 years. First, we observed that the metrics between the training and validation samples were close, illustrating the absence of overfitting.

```
rbind(train = summary(sl2, method="sl", pro.time=2, digits=4)$metrics,
      test =  summary(sl2,  newdata=dataVALID, method="sl",
                      pro.time=2, digits=4)$metrics)
```

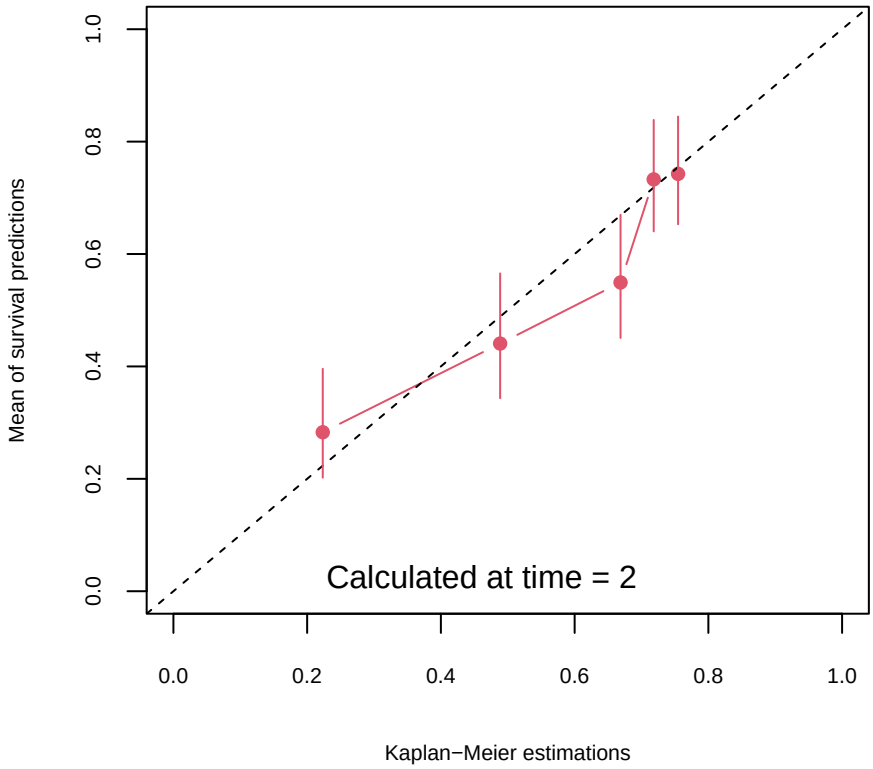



**Figure 4:** Calibration plot at 2 years for the validation sample.

```
#>         p_ci uno_ci    auc     bs    ibs   ribs    bll ibll ribll        ll
#> train 0.7463 0.7444 0.8056 0.1838 0.0813 0.0840 0.5487  NaN   NaN -465.7561
#> test  0.6737 0.6713 0.7036 0.2170 0.0924 0.0969 0.6232  NaN   NaN -595.7444
```

The user may also want to evaluate the prognostic capacities of the Elastic-Net PH model since it mainly contributes to the SL. It illustrates a marginal increase of the prognostic capacities related to the SL:

```
rbind(train = summary(sl2, method="LIB_COXen", pro.time=2, digits=4)$metrics,
      test =  summary(sl2,newdata=dataVALID, method="LIB_COXen",
                      pro.time=2, digits=4)$metrics)
```

```
#>         p_ci uno_ci    auc     bs    ibs   ribs    bll   ibll  ribll        ll
#> train 0.6782 0.6762 0.7204 0.1990 0.0875 0.0903 0.5843 0.2780 0.2921 -566.8255
#> test  0.6620 0.6588 0.6874 0.2201 0.0929 0.0974 0.6296 0.2952 0.3146 -664.7764
```

As illustrated by Figure 4, which is simply obtained by applying the `plot.sltime` function to the `sl2` object, the calibration of the SL was acceptable for patients of the validation sample. It represents the observed survival and the related 95% confidence intervals, which are respectively obtained by the Kaplan and Meier estimator and the Greenwood formula, against the mean of the predicted values, for individuals stratified into groups of the same size according to the percentiles of the predicted values. The identity line is usually included for reference.

```
plot(sl2, newdata=dataVALID, cex.lab=0.70,
     cex.axis=0.70, n.groups=5, pro.time=2, col=2)
```

The `predict.sltime` function returns predicted survival probabilities. With this function, one can also investigate the calibration by comparing the mean of the predictions of the patients included in the validation sample and the observed relapse-free survival, as illustrated in Figure 6 (Appendix). The SL did not improve the discrimination capacities offered by the modified Rio score for a prognostic up to 2 years after the treatment initiation (Figure 7 in Appendix).

## 5 Conclusion

The present paper introduced the **survivalSL** package to estimate a SL from right-censored data, to evaluate its prognostic capacities, and to predict survival curves for new individuals. The solution allows a larger set of learners (models or algorithms) and several loss functions to estimate their contributions. The user can define a grid for estimating the hyperparameters by cross-validation, or even fix their values.

The simulation study illustrated the SL performances. More precisely, in a simple context (several independent covariates, respect of the log-linearity assumption, no interaction), its performances are equivalent to more simple PH models. In contrast, in a complex context, the performances of the simple PH models decreased, and the SL resulted in the best performances: both the means and the variances of the loss functions were comparatively small relative to other methods considered.

We illustrated the usage of the package for predicting the non-response of patients with multiple sclerosis treated with a first-line therapy. The S3 methods allow a simple evaluation of the prognostic capacities, which can be compared with those of existing scoring systems. In this simple application, comparable to the simple scenario of the simulation study, the SL did not result in higher prognostic capacities compared to the Elastic-net PH model or even the existing modified Rio score.

As machine learning is increasingly being used in predictive studies, we believe that our proposal will be useful for a large community of data analysts. Nevertheless, we can point to three limitations. Firstly, even if we considered many learners, the current version of the package does not allow the users to include their own algorithms or models. An important perspective of development is to integrate this functionality. Secondly, one can also note that the estimation of learners and their weights assume non-informative censoring. Introducing the inverse probability of censoring weighting (IPCW) method, to correct for dependent censoring, would be of great interest, but challenging for some learners such as neural networks or random survival forests.

## 6 Acknowledgments

We would like to thank David Laplaud, neurologist and specialist in multiple sclerosis, for his clinical advice. The OFSEP cohort is supported by a grant provided by the French National Agency for Research, within the framework of the Investments for Future (ANR-10-COHO-002), and by the Eugène Devic EDMUS Foundation against multiple sclerosis and the ARSEP Foundation.

## 7 Bug reports

To improve future versions of the package, you can report errors at the following address:

`https://github.com/chupverse/survivalSL/issues`

# 8 Supplementary materials

## 8.1 R code to perform the simulations of the simple scenario

```
library(survivalSL)

# definition of the parameters related to the simulated data
n.valid <- 500 # sample size for validation
n.learn <- 500 # sample size for training

n <- n.valid + n.learn # overall sample size

max.time <- 50 # maximum follow-up time

mean.x <- 0; sd.x <- 1 # normal distribution of the quantitative predictors
proba.x <- .5 # proportion of the binary predictors

a <- 2; b <- .05 # Weibull baseline distribution of the PH model
beta <- c(log(1.8), log(1.8), log(1.3), 0, 0, 0) # regression coefficients

# simulation of  the training and validation samples
x1 <- rnorm(n, mean.x, sd.x)
x2 <- rbinom(n, 1, proba.x)
x3 <- rbinom(n, 1, proba.x)
x4 <- rnorm(n, mean.x, sd.x)
x5 <- rbinom(n, 1, proba.x)
x6 <- rbinom(n, 1, proba.x)
x <- cbind(x1, x2, x3, x4, x5, x6) # matrix of the potential predictors

u <- runif(n, 0, 1)
times <- 1/b*((-exp(-1*(x %*% beta))*(log(1-u)))**(1/a)) # time to event

censoring <- runif(n, min=0, max=max.time)

status <- ifelse(times <= censoring, 1, 0) # event status
obs.times <- ifelse(times <= censoring, times, censoring) # follow-up times

data <- cbind(obs.times, status, as.data.frame(x))

data.simul <- list(data[1:n.valid,], data[(n.valid+1):n,])

# definition of the grid search related to the hyperparameters

slres <- survivalSL(
  formula=Surv(obs.times, status) ~ x1 + x2 + x3 + x4 + x5 + x6,
  methods=c("LIB_COXen", "LIB_COXall", "LIB_RSF", "LIB_PLANN",
            "LIB_AFTgamma", "LIB_PHexponential"),
  metric="ibs",  data=data.simul[[1]], show_progress=FALSE, cv=10)

# loss functions estimated from training sample (Figure 1)
summary(slres)

# from validation sample (Figure 1)
summary(slres, newdata=data.simul[[2]])
```

### 8.2 R code to perform the simulations of the complex scenario

```
library(survivalSL)

# definition of the parameters related to the simulated data
n.valid <- 500 # sample size for validation
n.learn <- 200 # sample size for training
n <- n.valid + n.learn # overall sample size

max.time <- 50 # maximum follow-up time

b0.t <- (-0.4) # intercept related to the predictors' distribution
b1.t <- log(2) # slope related to the predictors' distribution

a <- 2; b <- .05 # Weibull baseline distribution of the PH model
b1.o <- 0.69 # regression coefficients of the PH model

# simulation of  the training and validation samples
.x1 <- rnorm(n, 0, 1)
.x2 <- rnorm(n, b0.t + b1.t * .x1, 1)
.x3 <- rnorm(n, b0.t - b1.t * .x1 - b1.t * .x2, 1)
.x4 <- rnorm(n, b0.t + b1.t * .x3, 1)
.x5 <- rnorm(n, 0, 1)
.x6 <- 1 * (rnorm(n, 0, 1) > 0.66)
.x7 <- 1 * (rnorm(n, b0.t - b1.t * .x5, 1) > (-0.40))
.x8 <- rnorm(n, b0.t - b1.t * .x6, 1)
.x9 <- 1 * (rnorm(n, b0.t + b1.t * .x7, 1) > (-0.80))
.x10 <- rnorm(n, b0.t + b1.t * .x8, 1)
.x11 <- rnorm(n, 0, 1)
.x12 <- 1 * (rnorm(n, b0.t + b1.t * .x9, 1) > (0.84))
.x13 <- 1 * (rnorm(n, b0.t + b1.t * .x10, 1) > (-0.09))
.x14 <- rnorm(n, b0.t - b1.t * .x12 - b1.t * .x11, 1)
.x15 <- rnorm(n, b0.t - b1.t * .x12, 1)
.x16 <- 1 * (rnorm(n, 0, 1) > (-0.66))
.x17 <- 1 * (rnorm(n, b0.t - b1.t * .x16, 1) > (-0.92))
.x18 <- rnorm(n, 0, 1)
.x19 <- 1 * (rnorm(n, 0, 1) > (0.66))
.x20 <- 1 * (rnorm(n, 0, 1) > (0.66))
.x21 <- rnorm(n, 0, 1)
.x22 <- 1 * (rnorm(n, 0, 1) > (0.66))

data.obs <- data.frame(x1=.x1, x2=.x2, x3=.x3, x4=.x4, x5=.x5, x6=.x6,
    x7=.x7, x8=.x8, x9=.x9, x10=.x10, x11=.x11, x12=.x12, x13=.x13,
    x14=.x14, x15=.x15, x16=.x16, x17=.x17, x18=.x18, x19=.x19,
    x20=.x20, x21=.x21, x22=.x22)

bx <- b0.t + b1.t*data.obs$x1 - b1.t*data.obs$x3 + b1.t*data.obs$x5 -
  b1.t*data.obs$x7 + b1.t*data.obs$x9 - b1.t*data.obs$x11 +
  b1.t*data.obs$x13 - b1.t*data.obs$x15 - b1.t*data.obs$x17 +
  b1.t*data.obs$x19 - b1.t*data.obs$x21

pr.t <- (exp(bx) / (1 + exp(bx)))
data.obs$t.obs <- rbinom(n, 1, prob = pr.t)

bx <- b1.o*(data.obs$x2>-0.40) - b1.o*data.obs$x3 +
  b1.o*0.5*(data.obs$x3^2) + b1.o*data.obs$x6 + b1.o*data.obs$x7 +
  b1.o*data.obs$x10 + b1.o*0.5*(data.obs$x11^2) - b1.o*data.obs$x14 -
  b1.o*(data.obs$x15>-0.57) + b1.o*data.obs$x18 + b1.o*data.obs$x19 +
  b1.o*data.obs$t.obs + b1.o*0.5*data.obs$t.obs*data.obs$x18

u <- runif(n,0,1)
times <- 1/b*((-exp(-bx)*(log(1-u)))**(1/a))

censoring <- runif(n, min=0, max=max.time)
```

```
status <- ifelse(times <= censoring, 1, 0)
obs.time <- ifelse(times <= censoring, times, censoring)

data <- cbind(obs.time, status, data.obs)

data.simul <- list(data[1:n.valid,], data[(n.valid+1):n,])

slres <- survivalSL(
  formula=Surv(obs.time, status) ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
  x10 + x11 + x12 + x13 + x14 + x15 + x16 + x17 + x18 + x19 + x20 + x21 + x22,
  data=data.simul[[1]],  metric="ibs",  cv=10, show_progress=FALSE,
  methods=c("LIB_COXen", "LIB_COXall", "LIB_RSF", "LIB_PLANN",
            "LIB_AFTgamma", "LIB_PHexponential"))

# loss functions estimated from training sample (Figure 2)
summary(slres)

# from validation sample (Figure 2)
summary(slres, newdata=data.simul[[2]])
```

### 8.3 Fluctuation of the results according to the seed

We estimate three additional SL with 3 alternative initializations of the pseudo-random number generator.

```
sl2.1 <- survivalSL(formula=.f, data=dataTRAIN, metric="uno_ci",
      methods=c("LIB_COXen", "LIB_PHspline", "LIB_AFTggamma", "LIB_RSF"),
      cv=30, optim.local.min=TRUE, param.tune=.tune, show_progress=FALSE, seed=118)

sl2.2 <- survivalSL(formula=.f, data=dataTRAIN, metric="uno_ci",
      methods=c("LIB_COXen", "LIB_PHspline", "LIB_AFTggamma", "LIB_RSF"),
      cv=30, optim.local.min=TRUE, param.tune=.tune, show_progress=FALSE, seed=119)

sl2.3 <- survivalSL(formula=.f, data=dataTRAIN, metric="uno_ci",
      methods=c("LIB_COXen", "LIB_PHspline", "LIB_AFTggamma", "LIB_RSF"),
      cv=30, optim.local.min=TRUE, param.tune=.tune, show_progress=FALSE, seed=120)

rbind(
  seed.121 = summary(sl2,   newdata=dataVALID, method="sl", pro.time=2, digits=4),
  seed.122 = summary(sl2.1, newdata=dataVALID, method="sl", pro.time=2, digits=4),
  seed.123 = summary(sl2.2, newdata=dataVALID, method="sl", pro.time=2, digits=4),
  seed.124 = summary(sl2.3, newdata=dataVALID, method="sl", pro.time=2, digits=4))

#>          metrics       method pro.time ROC.precision
#> seed.121 data.frame,10 "sl"   2        numeric,99
#> seed.122 data.frame,10 "sl"   2        numeric,99
#> seed.123 data.frame,10 "sl"   2        numeric,99
#> seed.124 data.frame,10 "sl"   2        numeric,99

rbind(
  seed.121 = sl2$weights$values,
  seed.122 = sl2.1$weights$values,
  seed.123 = sl2.2$weights$values,
  seed.124 = sl2.3$weights$values)

#>               [,1]      [,2]       [,3]      [,4]
#> seed.121 0.2486286 0.3339517 0.17734499 0.2400747
#> seed.122 0.2713981 0.3002995 0.17599652 0.2523059
#> seed.123 0.1721614 0.5182388 0.08864614 0.2209536
#> seed.124 0.2427707 0.2397718 0.27035706 0.2471004
```

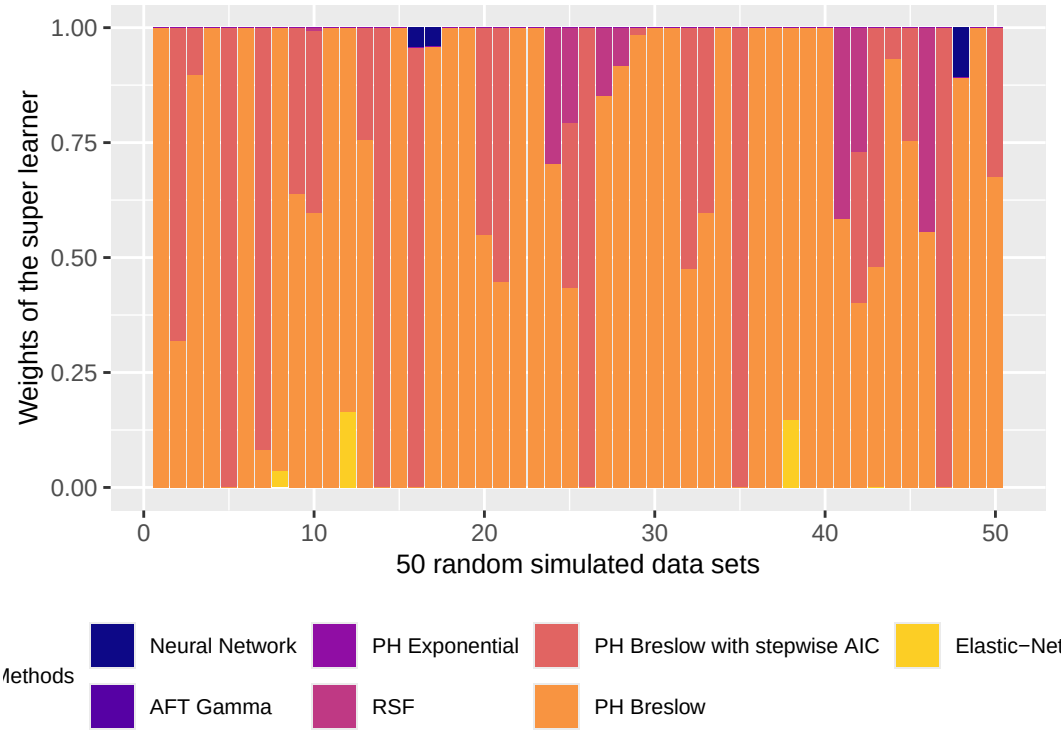



**Figure 5:** Weight distribution among 50 randomly estimated super learners.

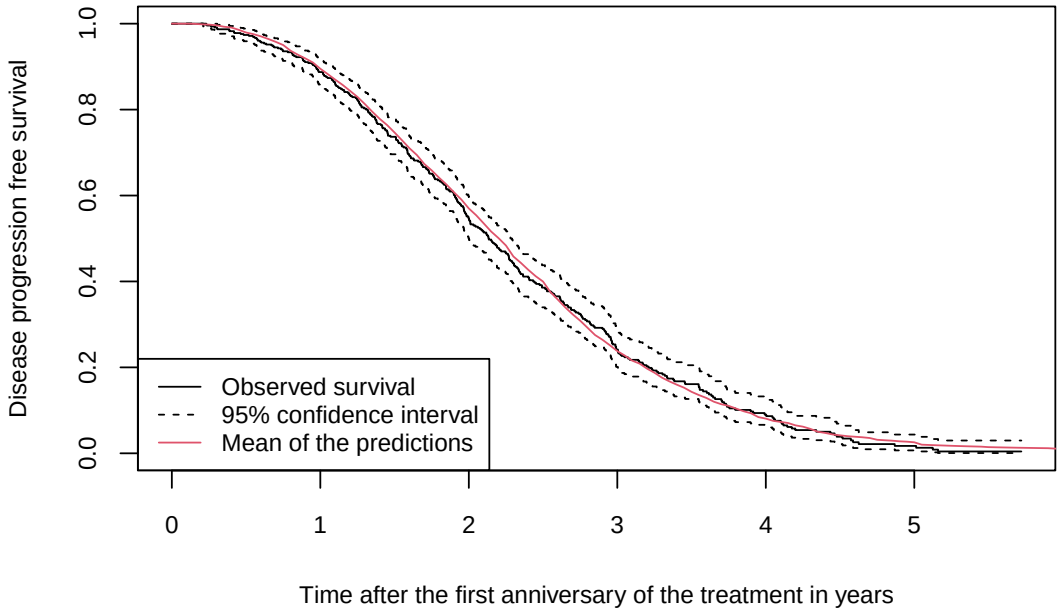



**Figure 6:** Observed relapse-free survival of the OFSEP validation sample (Kaplan-Meier estimator) compared to the mean of the individual predictions.

```
plot(survfit(Surv(time, event) ~ 1, data = dataVALID),
     ylab="Disease progression free survival",
     xlab="Time after the first anniversary of the treatment in years",
     cex.lab = 0.8, cex.axis=0.8)

.pred.matrix <- predict(sl2, newdata=dataVALID,
                        newtimes = seq(0, 6, by=0.05))$predictions$sl

.pred <- apply(.pred.matrix, MARGIN=2, FUN="mean")

lines(x=seq(0, 6, by=0.05), y=.pred, col=2)

legend("bottomleft", c("Observed survival", "95% confidence interval",
        "Mean of the predictions"), col=c(1, 1, 2), lty=c(1,2,1), cex=0.8)
```

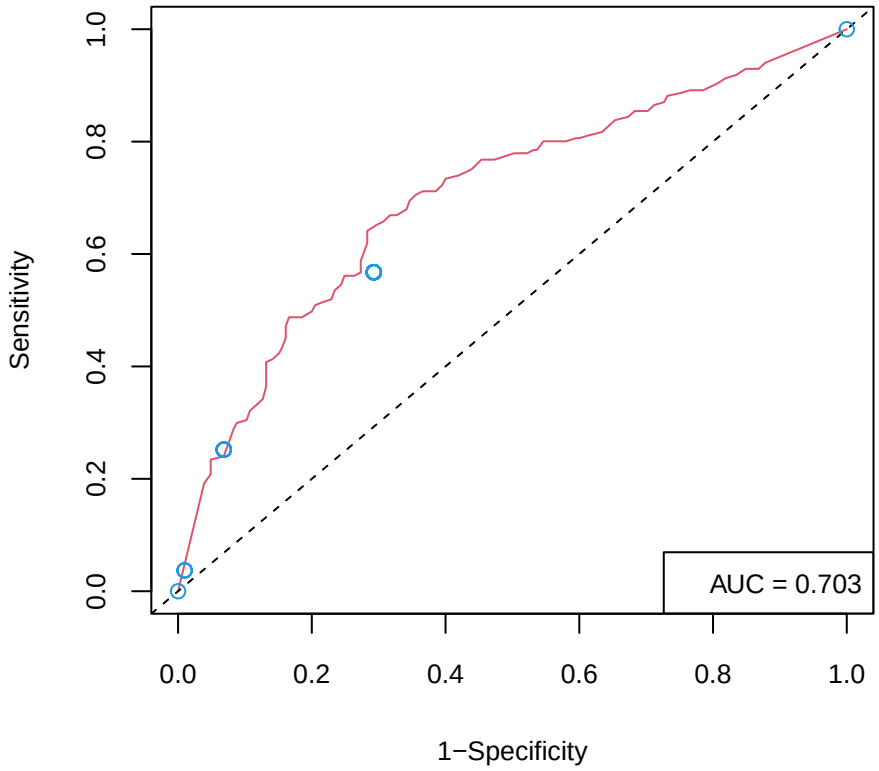



**Figure 7:** ROC curve of the SL (line in red) compared to the sensitivity and specificity of the modified Rio score (points in blue) for a prognostic up to 2 years in the OFSEP validation sample.

```
library("RISCA")

.pred <- predict(sl2, newdata=dataVALID)

dataVALID$sl <- 1 - .pred$predictions$sl[,sum(.pred$times<2)]

roc.sl <- roc.time(times="time", failures="event", variable="sl",
                   confounders=~1, data=dataVALID, pro.time=2,
                   precision=seq(0.1, 0.9, by=0.01))
# Note: “confounders=~1” for usual time-dependent ROC curves,
# i.e., without considering confounder (doi:10.1177/0962280217702416)

roc.rio <- roc.time(times="time", failures="event", variable="rio",
                    confounders=~1, data=dataVALID, pro.time=2)

plot(roc.sl, col=2, type="l", xlab="1-Specificity", ylab="Sensitivity",
     cex.lab=0.8, cex.axis=0.8)

points(x=1-roc.rio$table$sp, y=roc.rio$table$se, col=4)

legend("bottomright", legend=
  paste("AUC =", round(roc.sl$auc, 3)), cex=0.8 )
```

## References


O. Aalen. Nonparametric Inference for a Family of Counting Processes. *The Annals of Statistics*, 6(4): 701–726, July 1978. ISSN 0090-5364. doi: 10.1214/aos/1176344247. [p6]

P. K. Andersen, Ø. Borgan, N. L. Hjort, E. Arjas, J. Stene, and O. O. Aalen. Counting Process Models for Life History Data: A Review [with Discussion and Reply]. *Scandinavian Journal of Statistics*, 12(2): 97–158, 1985. [p4]

E. Biganzoli, P. Boracchi, L. Mariani, and E. Marubini. Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach. *Statistics in Medicine*, 17(10):1169–1186, May 1998. ISSN 0277-6715. doi: 10.1002/(sici)1097-0258(19980530)17:10<1169::aid-sim796>3.0.co;2-d. [p6]

L. Breiman. Stacked regressions. *Machine Learning*, 24:49–64, 1996. doi: 10.1007/BF00117832. [p4]

D. R. Cox. Regression Models and Life-Tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, Jan. 1972. ISSN 00359246. doi: 10.1111/j.2517-6161.1972.tb00899.x. [p4]

D. Faraggi and R. Simon. A neural network model for survival data. *Statistics in Medicine*, 14(1):73–82, Jan. 1995. ISSN 02776715, 10970258. doi: 10.1002/sim.4780140108. [p4]

J. J. Goeman. $L_1$ Penalized Estimation in the Cox Proportional Hazards Model. *Biometrical Journal*, pages NA–NA, Nov. 2009. ISSN 03233847, 15214036. doi: 10.1002/bimj.200900028. [p4]

M. K. Golmakani and E. C. Polley. Super Learner for Survival Data Prediction. *The International Journal of Biostatistics*, 16(2), Feb. 2020. ISSN 1557-4679. doi: 10.1515/ijb-2019-0065. [p4, 7]

H. Hung and C.-T. Chiang. Estimation methods for time-dependent AUC models with survival data. *Canadian Journal of Statistics*, 38(1):8–26, 2010. ISSN 1708-945X. doi: 10.1002/cjs.10046. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cjs.10046. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cjs.10046. [p6]

H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. Random survival forests. *The Annals of Applied Statistics*, 2(3), Sept. 2008. ISSN 1932-6157. doi: 10.1214/08-AOAS169. [p4, 6]

F. Le Borgne, A. Chatton, M. Léger, R. Lenain, and Y. Foucher. G-computation and machine learning for estimating the causal effects of binary exposure statuses on binary outcomes. *Scientific Reports*, 11(1):1435, Dec. 2021. ISSN 2045-2322. doi: 10.1038/s41598-021-81110-0. [p5]

D. Y. Lin. On the Breslow estimator. *Lifetime Data Analysis*, 13(4):471–480, Dec. 2007. ISSN 1380-7870, 1572-9249. doi: 10.1007/s10985-007-9048-y. [p4, 6]

R. V. Phillips, M. J. Van Der Laan, H. Lee, and S. Gruber. Practical considerations for specifying a super learner. *International Journal of Epidemiology*, 52(4):1276–1285, Aug. 2023. ISSN 0300-5771, 1464-3685. doi: 10.1093/ije/dyad023. URL https://academic.oup.com/ije/article/52/4/1276/7076266. [p11, 12]

E. C. Polley and M. J. van der Laan. Super Learner In Prediction. In *U.C. Berkeley Division of Biostatistics Working Paper Series*, May 2010. [p5]

E. C. Polley and M. J. van der Laan. Chapter 16. Super Learning for Right-Censored Data. In M. J. van der Laan and S. Rose, editors, *Targeted Learning*, Springer Series in Statistics. Springer New York, New York, NY, 2011. ISBN 978-1-4419-9781-4 978-1-4419-9782-1. doi: 10.1007/978-1-4419-9782-1. [p4]

E. C. Polley, S. Rose, and M. van der Laan. Chapter 3. Super Learning. In M. J. van der Laan and S. Rose, editors, *Targeted Learning*, Springer Series in Statistics. Springer New York, New York, NY, 2011. ISBN 978-1-4419-9781-4 978-1-4419-9782-1. doi: 10.1007/978-1-4419-9782-1. [p4]

P. S. Rosenberg. Hazard Function Estimation Using B-Splines. *Biometrics*, 51(3):874, Sept. 1995. ISSN 0006341X. doi: 10.2307/2532989. [p4]

P. Royston and M. K. B. Parmar. Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. *Statistics in Medicine*, 21(15):2175–2197, Aug. 2002. ISSN 0277-6715. doi: 10.1002/sim.1203. [p6]

P. K. Shivaswamy, W. Chu, and M. Jansche. A Support Vector Approach to Censored Targets. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 655–660, Omaha, NE, USA, Oct. 2007. IEEE. ISBN 978-0-7695-3018-5. doi: 10.1109/ICDM.2007.93. [p4]

N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent. *Journal of Statistical Software*, 39(5), 2011. ISSN 1548-7660. doi: 10.18637/jss.v039.i05. [p6]

M. Sormani, J. Rio, M. Tintorè, A. Signori, D. Li, P. Cornelisse, B. Stubinski, M. Stromillo, X. Montalban, and N. De Stefano. Scoring treatment response in patients with relapsing multiple sclerosis. *Multiple Sclerosis Journal*, 19(5):605–612, Apr. 2013. ISSN 1352-4585, 1477-0970. doi: 10.1177/1352458512460605. [p11]

H. Uno, T. Cai, M. J. Pencina, R. B. D'Agostino, and L. J. Wei. On the C-statistics for Evaluating Overall Adequacy of Risk Prediction Procedures with Censored Survival Data. *Statistics in medicine*, 30(10): 1105–1117, May 2011. ISSN 0277-6715. doi: 10.1002/sim.4154. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3079915/. [p5]

M. J. van der Laan and S. Dudoit. Unified cross-validation methodology for selection among estimators and a general cross-validated adaptive epsilon-net estimator: Finite sample oracle inequalities and examples. Technical report, Division of Biostatistics, University of California, Bercley, 2003. [p4]

M. J. van der Laan, E. C. Polley, and A. E. Hubbard. Super Learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1), Jan. 2007. ISSN 1544-6115, 2194-6302. doi: 10.2202/1544-6115.1309. [p4]

H. van Houwelingen, H. Putter, and J. C. van Houwelingen. *Dynamic Prediction in Clinical Survival Analysis*. Number 123 in Monographs on Statistics and Applied Probability. CRC Press, Taylor & Francis, Boca Raton, Fla., 2012. ISBN 978-1-4398-3543-2 978-1-4398-3533-3. [p5]

L. J. Wei. The accelerated failure time model: A useful alternative to the cox regression model in survival analysis. *Statistics in Medicine*, 11(14-15):1871–1879, 1992. ISSN 1097-0258. doi: 10.1002/sim.4780111409. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.4780111409. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sim.4780111409. [p4]

T. Westling. \pkg{survSuperLearner}: Super Learning for Conditional Survival Functions with Right-Censored Data, 2021. [p4, 7]

*Camille Sabathe*
*Nantes University, INSERM U1246 SPHERE, France*
*22, Bd Benoni Goullin, F-44200 Nantes*
*ORCiD: 0000-0001-9353-691X*
camille.sabathe@univ-nantes.fr

*Yohann Foucher*
*University and Hospital of Poitiers, CIC INSERM 1402*
*2, rue de la Miletrie, F-86000 Poitiers, France*
*ORCiD: 0000-0003-0330-7457*
yohann.foucher@univ-poitiers.fr

# IRTest: An R Package for Item Response Theory with Estimation of Latent Distribution

*by Seewoo Li*

**Abstract** Item response theory (IRT) models the relationship between respondents' latent traits and their responses to specific items. One key aspect of IRT is the assumption about the distribution of the latent variables, which can influence parameter estimation accuracy. While a normal distribution has been conventionally assumed, this may not always be appropriate. When the assumption of normality is violated, latent distribution estimation (LDE) can enhance parameter estimation accuracy by accommodating non-normal characteristics. Despite there being several methods proposed for LDE in IRT, there is a lack of software designed to handle their implementations. This paper introduces IRTest, a software program developed for IRT analysis that incorporates LDE procedures. It outlines the statistical foundation of LDE, details the functionalities of IRTest, and provides examples of IRT analyses to demonstrate the software's applications.

## 1 Introduction

Item response theory (IRT) is a widely used statistical framework for modeling the probabilistic relationship between examinees' latent traits (i.e., ability parameters) and their responses to specific items (de Ayala, 2009; Hambleton et al., 1991; van der Linden, 2016). These latent traits, an essential component of IRT, typically represent unobservable human characteristics in educational and psychological assessments, such as academic ability, depression severity, or extroversion levels.

In the estimation process of IRT models, particularly when using marginal maximum likelihood (MML), the latent trait's distribution can impact parameter estimation (Woods, 2015). MML is obtained by marginalizing a joint likelihood with respect to the latent variable, and any misspecification in the latent distribution can lead to biased parameter estimates. It is also worth noting that IRT necessarily assumes that the latent variables of interest are continuous. In some cases, a discrete latent distribution may better reflect observed data, such as in located latent class (LLC) models (see Clogg, 1981; Follmann, 1988; Haberman, 2005; McCutcheon, 1987; Xu and von Davier, 2008).

The conventional assumption of normality in latent distributions has been questioned, and empirical evidence and potential drawbacks of violating this assumption have been addressed (Dudley-Marling, 2020; Li, 2022; Mislevy, 1984; Sass et al., 2008; Seong, 1990; Woods and Lin, 2009). Previous studies have identified factors that can result in a skewed and/or bimodal latent distribution (Harvey and Murry, 1994; Ho and Yu, 2015; Woods, 2015; Yadin, 2013): disparities between high-achieving and low-achieving groups, the presence of an extreme group, difficulties of test items, and an innate human tendency to be inclined to one side of a latent ability scale. Potential problems of this assumption being violated include biases in parameter estimates and errors in ensuing decision-making processes. In this case, latent distribution estimation (LDE) can effectively reduce the biases in parameter estimates by capturing non-normal characteristics of the latent distribution.

Several methods have been proposed for LDE: the empirical histogram method (EHM: Bock and Aitkin, 1981; Mislevy, 1984), a mixture of two normal components (2NM: Li, 2021; Mislevy, 1984), the Ramsay-curve method (RCM: Woods, 2006b), the Davidian-curve method (DCM: Woods and Lin, 2009), the log-linear smoothing method (LLS: Casabianca and Lewis, 2015; Xu and von Davier, 2008), and the kernel density estimation method (KDM: Li, 2022).

**IRTest** is an R package for unidimensional IRT analyses which aims to handle LDE in IRT. **IRTest** is available from the Comprehensive R Archive Network (CRAN) at `https://cran.r-project.org/package=IRTest`. In **IRTest**, the model-fitting functions estimate the latent distribution through MML estimation using the EM algorithm (MML-EM: Bock and Aitkin, 1981). Along with the conventional method of assuming normality, five LDE methods (EHM, 2NM, DCM, LLS, and KDM) are currently available as model-fitting procedures. Extensions of these methods to multidimensional settings will be updated in accordance with advancements in theoretical research, as the current lack of research in this area necessitates ongoing development.

Currently, there are not many software programs to implement LDE, and their choices of LDE methods are somewhat limited since LDE may not be one of their main concerns. Most of them offer only EHM which may be the most straightforward way to carry out LDE (e.g., BILOG-MG, Zimowski et al., 2003; flexMIRT, Cai, 2022), some software programs such as RCLOG (Woods, 2006a), LLSEM

(Casabianca and Lewis, 2011), and **sirt** (Robitzsch, 2024) focus on one particular LDE method, and **mirt** (Chalmers, 2012) is equipped with EHM and DCM.

This paper details the statistical foundation of **IRTest** and its implementation. The remainder of this paper is organized as follows: Section 2 explains the basic statistical concepts of the IRT parameter estimation and the LDE procedure within the MML-EM framework. Section 3 discusses the LDE methods. Section 4 validates **IRTest** by comparing it with existing packages. Section 5 demonstrates the **IRTest** implementations. Lastly, Section 6 presents a discussion on the package.

## 2 LDE in IRT

This section provides a brief overview of the statistical aspects of LDE in IRT, and specific LDE methods will be discussed in detail in the next section. The objectives of this section are 1) to introduce the basic concepts of IRT, 2) to summarize the role of the latent distribution within the estimation process, and 3) to explain how LDE can enhance parameter estimation accuracy.

IRT is a statistical framework widely used in educational and psychological measurement to analyze how examinees' responses to test items reflect their underlying traits. IRT models offer mathematical functions to estimate the probability of examinees' responses based on item parameters and their ability levels. Among the estimation methods, MML is often preferred over conditional maximum likelihood (CML) and joint maximum likelihood (JML) for its versatility in model selection and statistical consistency of estimation. MML incorporates the latent distribution when integrating out the latent variable from the joint likelihood.

Statistically, LDE extends the normality assumption models by adding distribution parameters. While the marginal likelihood of the normality assumption models is dependent only on item parameters, LDE incorporates both item and distribution parameters in the marginal likelihood. It is worth noting that some LDE methods may not follow the maximum likelihood estimation (MLE) approach. Instead, they may utilize another index to estimate distribution parameters.

### 2.1 IRT models

Most IRT models follow a monotonically increasing probabilistic form and specify a functional relationship between item parameters and ability parameters. This section presents five well-known and widely-used IRT models, all of which are available in **IRTest**. For brevity, the discussion is mainly focused on three-parameter logistic model (3PLM: Birnbaum, 1968) and generalized partial credit model (GPCM: Muraki, 1992), since 3PLM can be reduced to one-parameter logistic model (1PLM: Rasch, 1960) or two-parameter logistic model (2PLM: Birnbaum, 1968) and GPCM can be reduced to partial credit model (PCM: Masters, 1982). The former handles dichotomous item responses and the latter handles polytomous item responses.

More than one model can be applied to item response data. For example, when analyzing a test of dichotomous items, the 2PLM can be applied to short-answer items, while the 3PLM can be applied to multiple-choice items. This differentiation allows researchers to reflect the differences in guessing behaviors observed between the two types of items. Also, the pair of one dichotomous response model and one polytomous response model could be used for a mixed-format test that comprises both dichotomous and polytomous items (Baker and Kim, 2004).

**Three-parameter logistic model (3PLM)** Let $u \in \{0, 1\}$ be a dichotomous item response, $\theta$ be the ability parameter of an examinee, and $a$, $b$, and $c$ $(0 < c < 1)$ be the item discrimination, difficulty, and guessing parameters, respectively. The item response function of the 3PLM can be expressed as,

$$\Pr(u = 1 \mid \theta, a, b, c) = c + (1 - c)\frac{\exp\left(a(\theta - b)\right)}{1 + \exp\left(a(\theta - b)\right)}, \tag{1}$$

where $u = 1$ indicates the correct response of the examinee. The probability ranges from $c$ to 1 because of the guessing parameter determining the lower bound. Given the nature of dichotomous items, $1 - \Pr(u = 1 \mid \theta, a, b, c)$ represents the probability of an incorrect response. The model reduces to the 2PLM when $c = 0$, and to the 1PLM when $c = 0$ and the same $a$ value is assumed across all items.

**Generalized partial credit model (GPCM)** The GPCM can be regarded as a polytomous form of the 2PLM. Let $u \in \{0, 1, \ldots, M\}$ $(M \geq 2)$ be a polytomous item response, $b_v$ $(v = 1, 2, \ldots, M)$ be the boundary parameters, and the rest be the same as previously defined. The item response function of

the GPCM can be expressed as,

$$\Pr(u = k \mid \theta, a, b_1, \ldots, b_M) = \frac{\exp \sum_{v=0}^{k} (a(\theta - b_v))}{\sum_{m=0}^{M} \exp \sum_{v=0}^{m} (a(\theta - b_v))}, \tag{2}$$

where $\exp \sum_{v=0}^{0} (a(\theta - b_v)) = 1$ for notational convenience. Equation (2) represents the probability of providing a response of $u = k$. The GPCM reduces to the PCM when the same $a$ value is assumed across all items. When $M = 1$, they are reduced to their dichotomous counterparts: the 1PLM and the 2PLM, respectively.

### 2.2 Role of the latent distribution

In the implementation of MML-EM, quadrature schemes are used to numerically approximate the integral with respect to the latent variable (Baker and Kim, 2004; Bock and Aitkin, 1981). A quadrature scheme transforms a continuous latent variable $\theta$ into a discrete variable $\theta^*$; the domain of $\theta$ is divided into non-overlapping $Q$ grids, each of which is assigned to a certain value of $\theta^*$ called a quadrature point. Typically, quadrature points are set to the middle of the grids. The default option of **IRTest** is to set quadrature points from $-6$ to 6 with an increment of 0.1, resulting in 121 quadrature points. Within the estimation functions of **IRTest**, the `range` and `q` arguments determine the range and the number of quadrature points, respectively. The corresponding probability mass function (PMF) of the latent variable can be expressed as,

$$A\left(\theta_q^*\right) = \frac{g\left(\theta_q^*\right)}{\sum_{q=1}^{Q} g\left(\theta_q^*\right)}, \tag{3}$$

where $g(\theta)$ is the probability density function (PDF) of the latent variable and $\theta_q^*$ is the $q$th quadrature point ($q = 1, 2, \ldots, Q$) (Baker and Kim, 2004). In this paper, the term latent distribution indicates either $g(\theta)$ or $A(\theta^*)$ depending on the context.

The marginal log-likelihood of the model is the quantity to be maximized in the estimation procedure, which can be expressed as follows (Baker and Kim, 2004):

$$\begin{aligned} \log L &= \sum_{j=1}^{N} \log \int_{\theta} L_j(\theta) g(\theta)\, d\theta \\ &\approx \sum_{j=1}^{N} \log \sum_{q=1}^{Q} L_j\left(\theta_q^*\right) A\left(\theta_q^*\right). \end{aligned} \tag{4}$$

In the equation above, the integral is approximated by the summation to facilitate the EM algorithm. The quantity $L_j\left(\theta_q^*\right)$ is the $j$th examinee's ($j = 1, 2, \ldots, N$) likelihood for their item responses given that their ability parameter is $\theta_q^*$.

Equation (4) shows that the latent distribution is one of the components of the marginal log-likelihood. Consequently, the specification of the latent distribution influences the value of the marginal log-likelihood of a model, potentially impacting the accuracy of parameter estimates.

Additionally, the latent distribution plays a role in model identification and affects the convergence of the MML-EM procedure. In **IRTest**, a scale is assigned to the latent variable by setting the mean and standard deviation of the latent distribution to 0 and 1, respectively. Meanwhile, for LDE methods such as LLS and DCM, where a hyperparameter determines the number of distributional parameters, the MML-EM procedure may not converge with a small sample size and a large number of distributional parameters.

### 2.3 LDE in the MML-EM procedure

The decomposition of the marginal log-likelihood would help explicate the separate estimation of the item and distribution parameters, which can be expressed as follows (Li, 2021):

$$\begin{aligned} \log L &\approx \sum_{q=1}^{Q} \sum_{j=1}^{N} \gamma_{jq} \log L_j\left(\theta_q^*\right) + \sum_{q=1}^{Q} \sum_{j=1}^{N} \gamma_{jq} \log A\left(\theta_q^*\right) - \sum_{q=1}^{Q} \sum_{j=1}^{N} \gamma_{jq} \log \gamma_{jq} \\ &= \log L_{\text{item}} + \log L_{\text{distribution}} - (\text{constant}). \end{aligned} \tag{5}$$

The quantity $\gamma_{jq} = E\left(\Pr_j\left(\theta_q^*\right)\right)$, calculated through Bayes' theorem in the expectation-step (E-step) of the EM algorithm, represents the expected probability of $j$th examinee's ability parameter belonging to the $q$th grid (see Baker and Kim, 2004). Then, in the maximization-step (M-step), the item and distribution parameters are estimated. Since $\gamma_{jq}$ is a function of the latent distribution, precise specification of the latent distribution would enhance the accuracy of $\gamma_{jq}$. Thus, the parameter estimates are implicitly affected by the latent distribution through $\gamma_{jq}$.

In equation (5), regarding $\gamma_{jq}$ as a constant, the $L_j\left(\theta_q^*\right)$ in the first term depends only on item parameters, while the $A\left(\theta_q^*\right)$ in the second term depends only on distribution parameters. This probabilistic independence allows the separate estimation of the item and distribution parameters, from which a selection of estimation methods of the distribution parameters may emerge.

To elaborate more on the second term of equation (5), it can be rewritten and simplified as,

$$\begin{aligned} \log L_{\text{distribution}} &= \sum_{q=1}^{Q}\sum_{j=1}^{N}\gamma_{jq}\log A\left(\theta_q^*\right) \\ &= \sum_{q=1}^{Q}\hat{f}_q\log A\left(\theta_q^*\right), \end{aligned} \tag{6}$$

where $f_q$ is an unknown true frequency at the $q$th grid and $\hat{f}_q = E(f_q) = \sum_{j=1}^{N}\gamma_{jq}$ is the expected frequency at the $q$th grid by the definition of $\gamma_{jq}$. In the E-step, the latent distribution is involved in calculating $\hat{f}_q$, then, in the M-step, the distribution parameters are estimated and updated by using the quantity $\hat{f}_q$. This E and M cycle iterates until the algorithm converges. The estimated parameters in the last iteration would be the final output, and the corresponding distribution of the final output becomes the estimated latent distribution.

Unlike the item parameter estimation being aligned with the MLE approach of the MML-EM procedure, the distribution parameters are not always estimated by maximizing equation (6). With the MLE approach still being the dominant choice, different approaches, such as minimizing the *approximate mean integrated squared error* (Li, 2022), can be applied to estimate distribution parameters, which is discussed in the next section. In all case, every strategy for the estimation of distribution parameters utilizes $\gamma_{jq}$ in its estimation procedure.

## 3 LDE methods

In principle, almost every density estimation method can be used for LDE in IRT. However, existing studies have selectively inspected and developed some methods that would enhance the effectiveness of practical applications of IRT and/or benefit the researchers working on IRT. This section focuses on four LDE methods to highlight their methodological diversity. The choice and order of the methods in this section are not intended to imply any superiority of one method over the other.

At the time of writing this article, there is a lack of research comparing LDE methods. Given that comparing models in meaningful ways is crucial for practical applications, further studies examining the evaluation criteria of LDE methods would be beneficial for practitioners to select the most suitable method for their analyses.

### 3.1 Empirical histogram method (EHM)

One simple LDE strategy would be to directly employ the outputs obtained from the E-step. EHM does this by simply calculating the expected probabilities for each grid of the quadrature scheme, which can be considered either as the normalized expected sample size or nonparametric maximum likelihood estimates (Bock and Aitkin, 1981; Laird, 1978; Mislevy, 1984). The entire estimation process can be easily portrayed in the form of an equation as follows:

$$\hat{A}_q = E(A_q) = \frac{\sum_{j=1}^{N} E\left(\Pr_j\left(\theta_q^*\right)\right)}{N} = \frac{\sum_{j=1}^{N}\gamma_j q}{N} = \frac{\hat{f}_q}{N}, \tag{7}$$

where $A_q$ denotes $A\left(\theta_q^*\right)$ for notational brevity. It can be seen that the estimates are simply the expected frequencies $\hat{f}_q$'s divided by the total population $N$. EHM can be implemented in the estimation functions of **IRTest** by specifying the argument as `latent_dist="EHM"`.

Alternatively, the MLE solution can be derived in the following manner using Lagrangian multi-

pliers. With Lagrangian multipliers, the quantity to be maximized becomes,

$$\mathcal{L} = \sum_{q=1}^{Q} \hat{f}_q \log A_q - \lambda \left( \sum_{q=1}^{Q} A_q - 1 \right), \tag{8}$$

where the second term is introduced from the constraint for a proper distribution (i.e., $\sum_q A_q = 1$). Differentiating $\mathcal{L}$ with respect to $A_q$ and equating it to zero yields

$$\frac{\partial \mathcal{L}}{\partial A_q} = \frac{\hat{f}_q}{A_q} - \lambda = 0. \tag{9}$$

Then, $A_q = \frac{\hat{f}_q}{\lambda}$ for all $q = 1, 2, \ldots, Q$, which results in $\lambda = N$ by the constraint. This shows that $\hat{A}_q = \frac{\hat{f}_q}{N}$ maximizes the likelihood $\mathcal{L}$.

In addition to its simplicity and expediency, EHM has been shown to be effective in reducing biases in parameter estimates when the normality assumption is violated. However, the performance of EHM could be limited to some extent when it fails to screen out the random noise of the data, thereby producing less accurate parameter estimates (Li, 2021; Woods, 2015; Woods and Lin, 2009). To address this issue, some methods incorporate smoothing procedures to alleviate the impacts of the random noise, which are addressed later in this section.

### 3.2 Two-component normal mixture distribution (2NM)

The 2NM is made up of two normal components added up together to form a single distribution. As a natural extension of the normality assumption, the 2NM could be thought of as a non-normal distribution caused by two different latent groups where each group is assumed to follow a normal distribution. The addition of a normal component imparts flexibility to the 2NM to reflect bimodality and skewness of the latent distribution. The 2NM method can be implemented in the estimation functions of **IRTest** by specifying the argument as `latent_dist="2NM"`.

Letting $\tau = [\pi, \mu_1, \mu_2, \sigma_1, \sigma_2]'$ be the vector of five original parameters of the 2NM, the PDF of the 2NM can be expressed as follows (Li, 2021):

$$g(\theta \mid \tau) = \pi \times \phi(\theta \mid \mu_1, \sigma_1) + (1 - \pi) \times \phi(\theta \mid \mu_2, \sigma_2), \tag{10}$$

where $\phi(\theta)$ is a normal component.

Proportionality holds when $A\left(\theta_q^*\right)$ in equation (6) is substituted with $g(\theta_q^* \mid \tau)$, resulting in $\log L_{\text{distribution}} \propto \sum_{q=1}^{Q} \hat{f}_q \log g(\theta_q^* \mid \tau)$. The MLE results for the 2NM parameters can be obtained by introducing another EM algorithm. In this paper, this additional optimization algorithm would be referred to as *the secondary EM algorithm* nested in the M-step of the primary EM algorithm.

To estimate the 2NM parameters, another quantity $\eta_q$ is calculated in the E-step of the secondary EM algorithm, which represents the expected probability of $\theta_q$ belonging to the first normal component. With the quantity $\eta_q$, the likelihood of the M-step of the secondary EM algorithm can be expressed as follows (Li, 2021):

$$\begin{aligned} \log L_{\text{distribution}} \propto & \sum_{q=1}^{Q} \hat{f}_q \eta_q \log \left[ \pi \phi(\theta_q \mid \mu_1, \sigma_1) \right] \\ & + \sum_{q=1}^{Q} \hat{f}_q (1 - \eta_q) \log \left[ (1 - \pi) \phi(\theta_q \mid \mu_2, \sigma_2) \right]. \end{aligned} \tag{11}$$

The closed-form solution for the 2NM parameters can be obtained by differentiating the likelihood with respect to each parameter and setting the first derivatives equal to zero (Li, 2021):

$$\hat{\pi} = \frac{\sum_{q=1}^{Q} \hat{f}_q \eta_q}{\sum_{q=1}^{Q} \hat{f}_q} = \frac{\sum_{q=1}^{Q} \hat{f}_q \eta_q}{N}, \tag{12}$$

$$\hat{\mu_1} = \frac{\sum_{q=1}^{Q} \hat{f}_q \eta_q \theta_q^*}{\sum_{q=1}^{Q} \hat{f}_q \eta_q}, \tag{13}$$

$$\hat{\mu_2} = \frac{\sum_{q=1}^{Q} \hat{f}_q (1-\eta_q)\theta_q^*}{\sum_{q=1}^{Q} \hat{f}_q (1-\eta_q)}, \tag{14}$$

$$\hat{\sigma_1^2} = \frac{\sum_{q=1}^{Q} \hat{f}_q \eta_q \left(\theta_q^* - \hat{\mu_1}\right)^2}{\sum_{q=1}^{Q} \hat{f}_q \eta_q}, \tag{15}$$

and

$$\hat{\sigma_2^2} = \frac{\sum_{q=1}^{Q} \hat{f}_q (1-\eta_q) \left(\theta_q^* - \hat{\mu_2}\right)^2}{\sum_{q=1}^{Q} \hat{f}_q (1-\eta_q)}. \tag{16}$$

Both advantages and disadvantages of the 2NM method stem from the parametric nature of the 2NM. The parameters of the 2NM render the estimated latent distribution interpretable. Also, the reparameterization of the 2NM parameters offers an inherent way to fix the mean and variance of the latent distribution to constants (see Li, 2021), which is a typical way to assign a scale to the latent variable in the MML-EM procedures. On the other hand, compared with its nonparametric counterparts, the flexibility of the 2NM is limited to some extent. For example, the 2NM is incapable of forming a wiggly-shaped distribution.

### 3.3 Davidian-curve method (DCM)

DCM uses a semi-nonparametric distribution, where the hyperparameter $h = 1, 2, \ldots, 10$ determines the complexity of the density (Woods and Lin, 2009). When $h = 1$, the distribution reduces to the normal distribution. In general, the search for an optimal hyperparameter involves balancing the flexibility of the latent distribution with the prevention of overfitting. DCM can be implemented in the estimation functions of **IRTest** by specifying the argument as `latent_dist="DC"`.

In DCM, the latent distribution can be expressed as follows:

$$g(\theta \mid h, \mathbf{m}) = \{P_h(\theta)\}^2 \varphi(\theta) = \left\{ \sum_{k=0}^{h} m_k \theta^k \right\}^2 \varphi(\theta), \tag{17}$$

where $\mathbf{m} = [m_0, m_1, \ldots, m_h]^{'}$ is a vector of coefficients, $P_h$ is a polynomial of order $h$, $\varphi$ is the standard normal distribution, and $m_h \neq 0$ (Woods and Lin, 2009; Zhang and Davidian, 2001). The following constraint guarantees that the function is a proper distribution (Zhang and Davidian, 2001):

$$\begin{aligned} E\left(P_h(Z)^2\right) &= E\left((\mathbf{mZ})^2\right) \\ &= \mathbf{m}^{'} E\left(\mathbf{ZZ}^{'}\right)\mathbf{m} \\ &= \mathbf{m}^{'}\mathbf{Mm} \\ &= \mathbf{m}^{'}\mathbf{B}^{'}\mathbf{Bm} \\ &= \mathbf{c}^{'}\mathbf{c} \\ &= 1. \end{aligned} \tag{18}$$

In the constraint above, $Z \sim N(0,1)$, $\mathbf{Z} = \left[1, Z^1, Z^1, \ldots, Z^h\right]^{'}$, $\mathbf{M} = E\left(\mathbf{ZZ}^{'}\right)$, $\mathbf{B}^{'}\mathbf{B} = \mathbf{M}$ by eigenvalue decomposition, and $\mathbf{c} = \mathbf{Bm}$ is a $h+1$ dimensional vector. By applying a polar coordinate transformation of $\mathbf{c}$, the constraint is always satisfied (see Woods and Lin, 2009; Zhang and Davidian, 2001).

As DCM follows the MLE approach, the quantity to be maximized for LDE can be expressed as,

$$\log L_{\text{distribution}} \propto \sum_{q=1}^{Q} \hat{f}_q \log \left[ \left\{ \left[\mathbf{B}^{-1}\mathbf{c}\right]^{'} \begin{bmatrix} \left(\theta_q^*\right)^0 \\ \left(\theta_q^*\right)^1 \\ \vdots \\ \left(\theta_q^*\right)^h \end{bmatrix} \right\}^2 \varphi\left(\theta_q^*\right) \right]. \tag{19}$$

Since the elements of $\mathbf{B}^{-1}$ are constants, the latent distribution is estimated by finding $\mathbf{c}$ that maximizes the likelihood above.

In the implementation of DCM, ten models are typically fitted according to each value of the hyperparameter ($h = 1, 2, \ldots, 10$). Then, the best model is selected by Hannan-Quinn (HQ) criterion (Hannan and Quinn, 1979):

$$HQ = -2 \log L + 2p \left( \log \left( \log N \right) \right), \tag{20}$$

where $N$ is the total number of examinees and $p$ is the number of parameters to be estimated. Focusing on whether HQ criterion selects $h = 1$ or not, this model selection procedure in DCM can be used to examine the normality of the latent distribution (Woods and Lin, 2009).

This paper does not go into details of LLS (Casabianca and Lewis, 2015; Xu and von Davier, 2008) because of the similarities between DCM and LLS: both of them take the MLE approach for parameter estimation, and the hyperparameter $h$ determines the number of distribution parameters to control the degree of smoothing.

### 3.4 Kernel density estimation method (KDM)

KDM is a nonparametric method for conducting the LDE procedure, and it can be implemented in the estimation functions of **IRTest** by specifying the argument as `latent_dist="KDE"`. In general, a kernel function is assigned to every observation to be stacked up all together and form a density function. In the context of LDE in IRT, the former statement means that $\hat{f}_q$ kernels are assigned to $\theta_q^*$, which can be expressed as,

$$g(\theta \,|\, h) = \frac{1}{Nh} \sum_{q=1}^{Q} \hat{f}_q K\left( \frac{\theta - \theta_q^*}{h} \right), \tag{21}$$

where $K(\cdot)$ is a kernel function and $h$ is a hyperparameter often referred to as the *bandwidth*. The following discussion assumes the Gaussian kernel for $K(\cdot)$ as a general default choice (Gramacki, 2018; Silverman, 1986).

KDM takes a different approach from the previously discussed methods in carrying out the LDE procedure. Instead of the log-likelihood ($\log L_{\text{distribution}}$), the approximate mean integrated squared error (AMISE) is used, which is the Tayor-series approximation of the mean integrated squared error:

$$AMISE(\hat{g_h}) = \frac{1}{2Nh\sqrt{\pi}} + \frac{h^4}{4} R(g''). \tag{22}$$

In the equation above, $\hat{g_h}$ is the estimated latent distribution using the bandwidth $h$ and $R(g'') = \int g''(x)\, dx$. Note that this is a simplified version of AMISE by the adoption of the Gaussian kernel. To find $h$ that minimizes equation (22), another equation is obtained by differentiating AMISE with respect to $h$ and equating it to 0 (Gramacki, 2018; Silverman, 1986; Wand and Jones, 1995):

$$h_{AMISE} = \left[ 2N\sqrt{\pi} R(g'') \right]^{-\frac{1}{5}}. \tag{23}$$

Unfortunately, this solution cannot be immediately employed in estimating the bandwidth, because $R(g'')$ still depends on the unknown density $g(\theta)$. For the details of the methods that deal with this situation, refer to Silverman (1986), Gramacki (2018), Sheather (2004), and Wand and Jones (1995). Utilizing the built-in R function `stats::density()` for the KDM procedure, the default option of **IRTest** is `bandwidth="SJ-ste"`, a recommended method for bandwidth estimation (Jones et al., 1996; Sheather and Jones, 1991). Other available options for the `bw` argument of the `stats::density()` can also be passed to the `bandwidth` argument of **IRTest**.

On the one hand, KDM and DCM are similar in that their hyperparameters determine the degree of smoothing. On the other hand, once the $\gamma_{jq}$ is calculated and treated as a constant, the hyperparameter of KDM is the only parameter to influence the LDE results, whereas the LDE results of DCM depends on both the hyperparameter and the corresponding $h + 1$ density parameters. The absence of distribution parameter in KDM, except for the hyperparameter (bandwidth) itself, allows KDM to estimate the hyperparameter in a single model-fitting procedure, thereby obviating the need for a model selection process. Compared with other methods having a model selection step, this advantage may decrease computation time and expedite the analysis (Li, 2022).

## 4 Package validation

This section examines and validates the estimation performance of **IRTest** by comparing its results with those from **mirt** (Chalmers, 2012) and **ltm** (Rizopoulos, 2006). **mirt** and **ltm** are among the widely used R packages for IRT analyses. To this end, a dataset of ten dichotomous items and 1,000 examinees is generated using the 2PLM and under a right-skewed latent distribution, which is used throughout

**Table 1:** Item parameters and their estimates

| | parameter | IRTest::IRTest_Dich() | IRTest::IRTest_Poly() | mirt::mirt() | ltm::ltm() |
|---|---|---|---|---|---|
| **Item discrimination parameter (a)** | | | | | |
| Item1 | 0.82 | 0.9264 | 0.9264 | 0.9263 | 0.9263 |
| Item2 | 1.26 | 1.0174 | 1.0174 | 1.0174 | 1.0174 |
| Item3 | 1.88 | 1.7614 | 1.7614 | 1.7613 | 1.7614 |
| Item4 | 1.41 | 1.3730 | 1.3730 | 1.3729 | 1.3729 |
| Item5 | 2.33 | 1.7300 | 1.7300 | 1.7296 | 1.7301 |
| Item6 | 1.34 | 1.2904 | 1.2904 | 1.2903 | 1.2903 |
| Item7 | 1.19 | 1.2593 | 1.2593 | 1.2592 | 1.2593 |
| Item8 | 1.15 | 1.3579 | 1.3579 | 1.3578 | 1.3579 |
| Item9 | 2.42 | 2.1221 | 2.1221 | 2.1218 | 2.1221 |
| Item10 | 2.31 | 1.7667 | 1.7667 | 1.7665 | 1.7668 |
| **Item difficulty parameter (b)** | | | | | |
| Item1 | 1.23 | 1.1484 | 1.1484 | 1.1482 | 1.1481 |
| Item2 | -0.72 | -0.7642 | -0.7642 | -0.7645 | -0.7645 |
| Item3 | -0.19 | -0.1788 | -0.1788 | -0.1791 | -0.1791 |
| Item4 | -0.14 | -0.1565 | -0.1565 | -0.1568 | -0.1568 |
| Item5 | -1.16 | -1.2970 | -1.2970 | -1.2975 | -1.2973 |
| Item6 | 0.14 | 0.2154 | 0.2154 | 0.2151 | 0.2151 |
| Item7 | 0.29 | 0.2619 | 0.2619 | 0.2616 | 0.2616 |
| Item8 | 1.28 | 1.1198 | 1.1198 | 1.1196 | 1.1195 |
| Item9 | 0.12 | 0.2919 | 0.2919 | 0.2916 | 0.2916 |
| Item10 | -0.96 | -0.9906 | -0.9906 | -0.9909 | -0.9909 |

this section.

The following functions are used to fit the models: `IRTest::IRTest_Dich()` and `IRTest::IRTest_Poly()` from **IRTest**, `mirt::mirt()` from **mirt**, and `ltm::ltm()` from **ltm**. The standard normal latent distribution, the 2PLM, and 61 quadrature points are applied to all functions, and the rest of the options are set to the default. Note that `IRTest::IRTest_Poly()` can also be applied to binary data, but with lower computational efficiency compared to `IRTest::IRTest_Dich()`.

Table 1 shows the item parameters and their estimates. The result shows that the parameter estimates from **IRTest** are almost identical to those from **mirt** and **ltm**. For example, the root mean square error (RMSE) of the $a$ (item discrimination) parameter estimates between `IRTest::IRTest_Dich()` and `mirt::mirt()` is $2.04\times10^{-4}$, and that between `IRTest::IRTest_Dich()` and `ltm::ltm` is $0.29\times10^{-4}$. Similarly, the RMSE of the $b$ (item difficulty) parameter estimates between `IRTest::IRTest_Dich()` and `mirt::mirt()` is $3.23\times10^{-4}$, and that between `IRTest::IRTest_Dich()` and `ltm::ltm()` is $3.21\times10^{-4}$. This shows that the estimates from the four functions are practically identical.

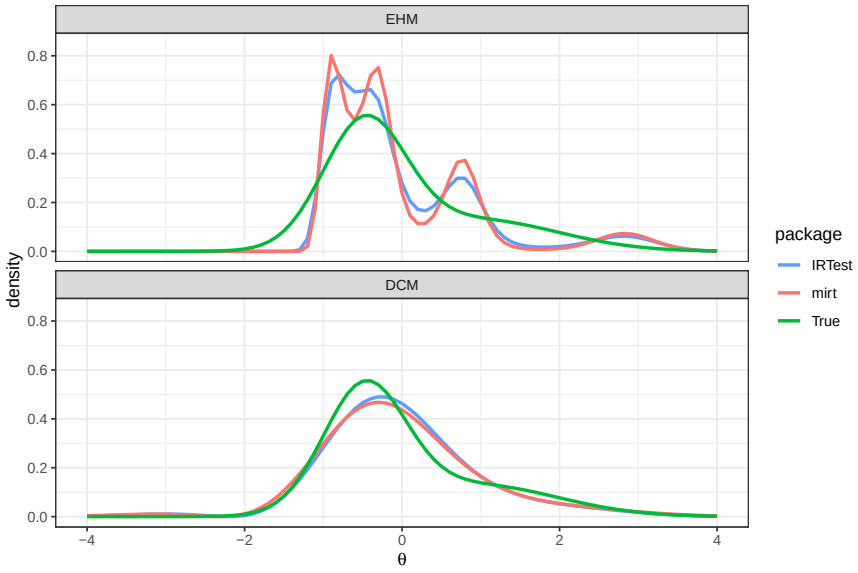



**Figure 1:** Estimated latent distributions from IRTest and mirt

To validate the LDE procedures of **IRTest**, the estimated latent distributions of **IRTest** are compared with those estimated from **mirt**, using EHM and DCM. The hyperparameter $h$ of DCM is set to $h = 3$ for an illustrative purpose by considering the complexity of the true distribution and the size of the

data. Total 121 quadrature points are used for the MML-EM procedure. Integrated squared error (ISE), one of the widely used criterion in comparing two distributions, is used as an index to measure the closeness of the estimated distributions of **IRTest** and **mirt** (for the usages of ISE, see Jones, 1991):

$$\begin{aligned} ISE\left(\hat{g}^{\text{IRTest}}, \hat{g}^{\text{mirt}}\right) &= \int_{-\infty}^{\infty} \left(\hat{g}(\theta)^{\text{IRTest}} - \hat{g}(\theta)^{\text{mirt}}\right)^2 d\theta \\ &\approx \sum_{q=1}^{Q} \left(\hat{g}\left(\theta_q^*\right)^{\text{IRTest}} - \hat{g}\left(\theta_q^*\right)^{\text{mirt}}\right)^2 \Delta\theta^*, \end{aligned} \tag{24}$$

where $\Delta\theta^*$ is the distance between quadrature points, and $\hat{g}^{\text{IRTest}}$ and $\hat{g}^{\text{mirt}}$ are the estimated latent distributions of **IRTest** and **mirt**, respectively. In equation (24), the integral is approximated by the summation using the quadrature scheme of the MML-EM procedure.

Figure 1 illustrates the estimated latent distributions from both packages and the true latent distribution. Note that values outside of $|\theta| \leq 4$ are truncated for visualization, but they are included in the ISE calculation. The figure shows that the estimated distribution of **IRTest** (blue line) and the one from **mirt** (red line) closely resemble each other. The ISE values for EHM and DCM are 0.0107 and 0.0009, respectively, which numerically validate the similarities of the estimated distributions of the two packages.

## 5 Implementations of IRTest

The primary purpose of this section is to demonstrate the usages of **IRTest** with examples. Section 5.1 provides an example using a simulated dataset. In doing so, it illustrates the effect of LDE when the normality assumption is violated. Section 5.2 performs an IRT analysis using the Generic Conspiracist Beliefs Scale (GCBS) data (Brotherton et al., 2013) available from the Open-Source Psychometric Project at `http://openpsychometrics.org/_rawdata/GCBS.zip`.

### 5.1 The effect of LDE

This section illustrates the effect of LDE by showing an improvement in estimation accuracy. To this end, an artificial item response dataset is generated from a non-normal latent distribution. The parameters of this dataset are used as the true values in evaluating errors.

**Data generation** Using the function `IRTest::DataGeneration()`, a dataset of 40 dichotomous items and 2,000 respondents is generated by

```
Alldata <- IRTest::DataGeneration(N = 2000,
                                  nitem_D = 40,
                                  latent_dist = "2NM",
                                  d = 1.414,
                                  sd_ratio = 2,
                                  prob = 2/3)
simulated_data <- Alldata$data_D
true_item <- Alldata$item_D
true_theta <- Alldata$theta
```

where `simulated_data` is the item response matrix, `true_item` is the item parameter matrix, and `true_theta` is the vector of ability parameters. The 2PLM is employed in generating the dataset, and a 2NM distribution is employed to simulate a non-normal latent distribution (`latent_dist = "2NM"`) with its parameters being `d = 1.414`, `sd_ratio = 2`, and `prob = 2/3` (for the reparameterization of the 2NM, see Li, 2021, 2024).

The highly skewed distribution is chosen to clearly demonstrate the effect of LDE, which is likely to be rare in practice (see Figure 2). Therefore, cautions are needed in interpreting and understanding the magnitude of the effect.

**Model fitting** The function `IRTest::IRTest_Dich()` is applied to the dichotomous data for the model-fitting, and an LDE method is specified by `latent_dist` argument. Two types of ability parameter estimates are illustrated: expected *a posteriori* (EAP) and maximum likelihood estimate (MLE). Note that *MLE* is also used to abbreviate maximum likelihood estimation. In **IRTest**, estimation methods of

ability parameters are determined by specifying the argument `ability_method`, where the default option is EAP. Either a model-fitting function (e.g., `IRTest::IRTest_Dich()`) or `IRTest::factor_score()` can be used to estimate ability parameters.

```
model_normal <- IRTest::IRTest_Dich(data = simulated_data,
                                    latent_dist = "Normal")
model_KDM <- IRTest::IRTest_Dich(data = simulated_data,
                                 latent_dist = "KDE")
theta_eap_normal <- IRTest::factor_score(model_normal)
theta_mle_normal <- IRTest::factor_score(model_normal, ability_method = "MLE")
theta_eap_KDM <- IRTest::factor_score(model_KDM)
theta_mle_KDM <- IRTest::factor_score(model_KDM, ability_method = "MLE")
```

Among these two models, `model_normal` assumes normality, whereas `model_KDM` estimates the latent distribution using KDM during the MML-EM procedure. KDM is arbitrarily selected for illustrative purposes.

**Estimated latent distribution** **IRTest** provides two ways to draw a density curve of the estimated latent distribution. The first is to use `plot()`, and the second is to use `IRTest::latent_distribution()`. The `plot()` can be considered as a shortcut for using `IRTest::latent_distribution()`. Note that `IRTest::latent_distribution()` is a PDF, and, thus, can only be applied to LDE methods that estimate a PDF. For example, since EHM (or LLS) estimates a *PMF*, a message will be printed without evaluated density values if an EHM-based (or LLS-based) object is passed to `IRTest::latent_distribution()`. The `plot()` can be utilized in all cases.

The following code can be an example for utilizing `IRTest::latent_distribution()`, where `IRTest::dist2()` is a density function of the 2NM and `stats::dnorm()` is the built-in function for a normal distribution.

```
density_plot <- ggplot2::ggplot() +
  ggplot2::stat_function(fun = IRTest::dist2,
                         args = list(d = 1.414, sd_ratio = 2, prob = 2/3),
                         linewidth = 1,
                         mapping = aes(color = "True")) +
  ggplot2::stat_function(fun = stats::dnorm,
                         linewidth = 1,
                         mapping = aes(color = "Normal")) +
  ggplot2::stat_function(fun = IRTest::latent_distribution,
                         args = list(model_KDM),
                         linewidth = 1,
                         mapping = aes(color = "KDM"))
```

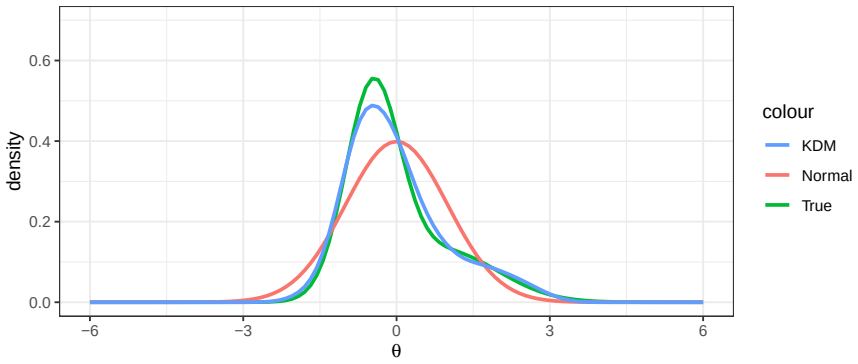



**Figure 2:** The true, normal, and estimated latent distributions

Figure 2 is drawn with a slight addition of aesthetic codes to the object `density_plot`. It shows the density curves of the true latent distribution, the standard normal distribution, and the estimated latent distribution. Notably, even with the discrepancies between the true distribution (green line) and the standard normal distribution (red line), the estimated distribution (blue line) almost recovered the shape of the true latent distribution.

**Table 2:** RMSEs from the normality assumption model and KDM

| | Normal | KDM |
|---|---|---|
| a | 0.181 | 0.123 |
| b | 0.124 | 0.079 |
| EAP | 0.272 | 0.252 |
| MLE | 0.285 | 0.272 |

**Parameter estimates** Differences in parameter estimates caused by the LDE procedure are portrayed in Figure 3, where the $x$-axis represents the true values (parameters) and the $y$-axis shows the corresponding errors. It shows that the parameter estimates from KDM (blue dots) are generally located closer to the "Error $= 0$" line than those from the normality assumption model (red dots), which indicates that the estimates from KDM are more accurate on average. RMSE is used as an evaluation criterion to assess the accuracy of the parameter estimates, where the lower RMSE indicates higher accuracy. Table 2 shows that, for all types of parameters, estimates from the KDM model are more accurate with lower RMSE values than those from the normality assumption model. Especially, the considerable amounts of decreases in RMSEs for $\hat{a}$ and $\hat{b}$ substantiates the effectiveness of LDE in enhancing the estimation accuracy of item parameters, where the magnitudes in the RMSE reduction are originated from the highly skewed latent distribution. As stated in Section 2.2, the results show that the appropriate specification of the latent distribution can have a positive impact on the accuracy of parameter estimates.

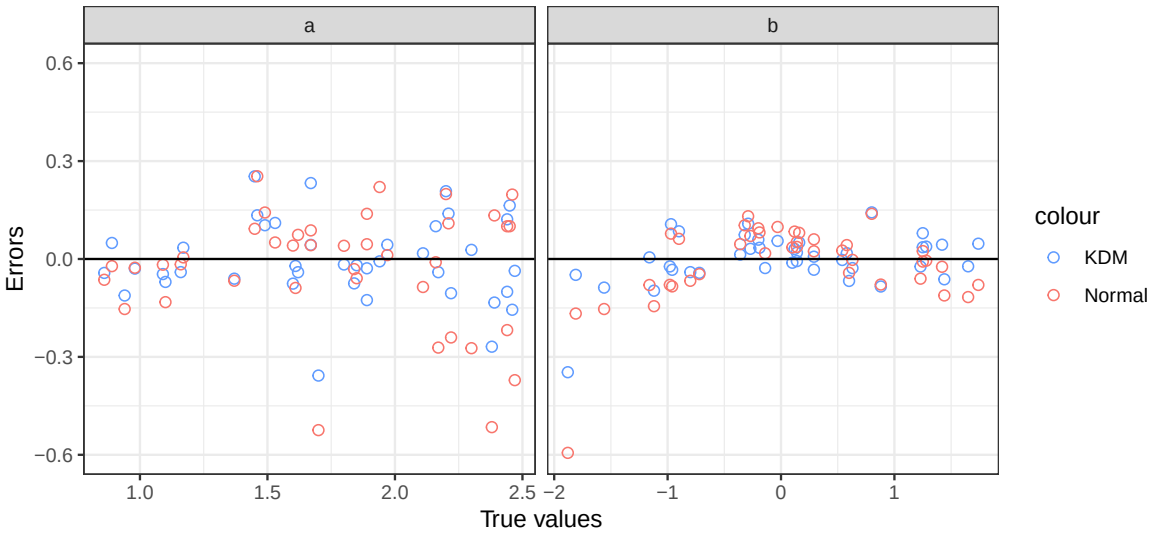



**Figure 3:** Differences in the errors of the item parameter estimates caused by LDE

## 5.2 An empirical example

This section performs an IRT analysis using the GCBS data. Along with the data analysis, one of the objectives of the section is to illustrate the usages of the functions of **IRTest**, such as those for calculating reliability coefficients and item/test information.

**Data** The GCBS data contains responses from 15 polytomous items and 2,391 respondents, where each item has five categories scored from one to five. The data can be loaded in the following manner.

```
data_GCBS <- read.csv("data/data_GCBS.csv")
```

There are 108 missing values in the data. **IRTest** uses a full-information maximum likelihood (FIML) approach in handling missing data.

**Model selection** DCM and KDM are used in performing LDE for illustrative purposes, where the DCM's $h$ is the hyperparameter indicating the complexity of the latent distribution. The PCM, GPCM,

and graded response model (GRM: Samejima, 1969) are available IRT models of `IRTest::IRTest_Poly()`, where the default is the GPCM. Models are fitted as follows:

```
# Davidian-curve method
DCMs <- list()
for(i in 1:10){
    DCMs[[i]] <- IRTest::IRTest_Poly(data = data_GCBS, latent_dist = "DC", h = i)
}
names(DCMs) <- paste0("DC", 1:10)

# kernel density estimation method
KDM <- IRTest::IRTest_Poly(data = data_GCBS, latent_dist = "KDE")
```

A model-selection step is required for DCM: the "best-DCM" can be selected by the HQ criterion (Hannan and Quinn, 1979) presented in equation (20). After ten models are fitted, the best model can be selected with the function `IRTest::best_model()` (`stats::anova()` can also be used). The `IRTest::best_model()` function uses a specific criterion to determine the best model, with options including `"logLik"`, `"deviance"`, `"AIC"`, `"BIC"`, and `"HQ"`. The default is `criterion = "HQ"`.

```
do.call(what = IRTest::best_model, args = DCMs)
```

```
#> The best model: DC8
#>
#>            HQ
#> DC1  91232.30
#> DC2  91236.40
#> DC3  91237.18
#> DC4  91220.93
#> DC5  91217.55
#> DC6  91210.21
#> DC7  91209.33
#> DC8  91207.85
#> DC9  91209.87
#> DC10 91213.24
```

The result indicates that `DC8` is the best DCM.

The `DC8` and `KDM` can also be compared by the function `IRTest::best_model()`.

```
IRTest::best_model(DCMs$DC8, KDM, criterion = "logLik")
```

```
#> The best model: KDM
#>
#>             logLik
#> DCMs$DC8 -45433.63
#> KDM      -45433.02
```

The rest of this section looks into the details of `KDM` by following the model-comparison result.

**Summary of the model** A brief summary of the model-fitting results can be printed with `summary()`. It presents the convergence status, model-fit indices, and the number of parameters and items. Also, it displays a cursory shape of the estimated latent distribution.

```
summary(KDM)
```

```
#> Convergence:
#> Successfully converged below the threshold of 1e-04 on 63rd iterations.
#>
#> Model Fit:
#>  log-likeli   -45433.02
#>    deviance   90866.05
#>         AIC   91018.05
#>         BIC   91457.48
```

```
#>          HQ   91177.92
#>
#> The Number of Parameters:
#>        item   75
#>        dist   1
#>       total   76
#>
#> The Number of Items:  15
#>
#> The Estimated Latent Distribution:
#> method - KDE
#> ----------------------------------------
#>
#>                     @ @ .
#>                   @ @ @ @
#>                 . @ @ @ @ @
#>               . @ @ @ @ @ @ .
#>               @ @ @ @ @ @ @ @
#>           . @ @ @ @ @ @ @ @ @ @
#>         . @ @ @ @ @ @ @ @ @ @ @ .
#>     . @ @ @ @ @ @ @ @ @ @ @ @ @ @ .
#> . @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ .
#> +---------+---------+---------+---------+
#> -2        -1        0         1         2
```

Alternatively, the shape of the estimated latent distribution can be easily visualized by the `plot()` function which produces a `ggplot`-class object. Figure 4 shows the estimated latent distribution of the GCBS data which is left-skewed. Figure 4 is produced by adding aesthetic codes to `plot()` and the standard normal distribution. If a PDF is estimated, additional arguments in `plot()` are passed to `ggplot2::stat_function()` of **ggplot2** (Wickham, 2016). Otherwise, if a PMF is estimated (e.g., EHM or LLS), they are passed to `ggplot2::geom_line()` of **ggplot2**.

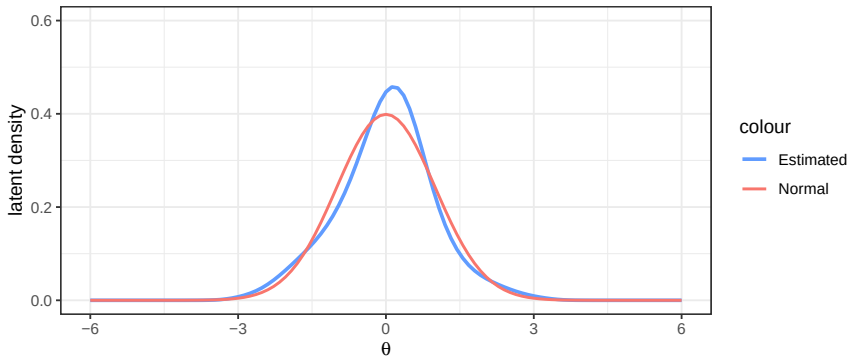



**Figure 4:** Estimated latent distribution for the GCBS data

**Parameter estimates** Users can access to item parameter estimates and corresponding standard errors with `stats::coef()` and `IRTest::coef_se()`, respectively.

```
stats::coef(KDM)
```

```
#>                a         b_1          b_2          b_3        b_4
#> Item1  0.9689201 -0.76302183 -0.391283148 -0.855328244  0.3535741
#> Item2  1.0789839 -0.53792382  0.009783414  0.005704736  0.7242562
#> Item3  0.7387147  1.67238829  0.366676394  0.966121767  1.3765605
#> Item4  1.3101921 -0.13464756  0.161325353  0.204687878  1.3519716
#> Item5  0.7816865 -0.54657035 -0.191682964 -0.831793749  0.7254415
#> Item6  1.1236494 -0.46965441 -0.187144001 -0.214049525  0.6453753
#> Item7  1.1752710 -0.07562134  0.301472014  0.146375779  0.9623761
#> Item8  0.6667551  1.54879257 -0.087408217  0.644569978  0.3019026
#> Item9  1.0243926  0.53049910  0.622129709  0.614778365  1.3307188
#> Item10 0.5422634 -0.67411059 -0.728342737 -1.374793623  0.4975367
```

```
#> Item11 1.0969859 -0.85223166 -0.647107727 -0.193551333  0.7969608
#> Item12 1.6043792 -0.13515994  0.138427510  0.313100514  0.9981246
#> Item13 0.8859117  1.15644479  0.243555061  1.098831006  1.3876873
#> Item14 1.0693468 -0.36458700 -0.054602648 -0.080358875  0.8726474
#> Item15 0.8699800 -2.01564618 -1.551317019 -1.872535217 -0.6573178

IRTest::coef_se(KDM)

#>                 a        b_1        b_2        b_3        b_4
#> Item1  0.03678565 0.08584960 0.08654712 0.08019167 0.05779205
#> Item2  0.03966144 0.06329931 0.06584207 0.06508447 0.06232969
#> Item3  0.03208946 0.11778055 0.10831653 0.11904930 0.13613987
#> Item4  0.04714700 0.05119141 0.05476176 0.05471713 0.06194283
#> Item5  0.03074447 0.09686522 0.09959919 0.09509046 0.07449349
#> Item6  0.04106021 0.06583399 0.06718341 0.06322656 0.05630556
#> Item7  0.04324230 0.05617174 0.06338652 0.06467895 0.06362815
#> Item8  0.02800753 0.12930642 0.12010949 0.12179173 0.12290590
#> Item9  0.04025272 0.06541014 0.07425989 0.08161815 0.09079149
#> Item10 0.02420022 0.15487167 0.14764212 0.13523579 0.09904333
#> Item11 0.04013028 0.07370413 0.06734274 0.05764279 0.05617594
#> Item12 0.05686980 0.04318188 0.04603551 0.04652548 0.04914336
#> Item13 0.03648830 0.09029916 0.08542261 0.09660685 0.11458529
#> Item14 0.03930729 0.06541102 0.06780640 0.06541428 0.06253578
#> Item15 0.03830840 0.16985780 0.13629512 0.10927582 0.06689059
```

Likewise, `IRTest::factor_score()` returns ability parameter estimates (`IRTest::factor_score()$theta`) and their standard errors (`IRTest::factor_score()$theta_se`) which are not printed here for their lengths being 2397.

```
IRTest::factor_score(KDM, ability_method = "EAP")
```

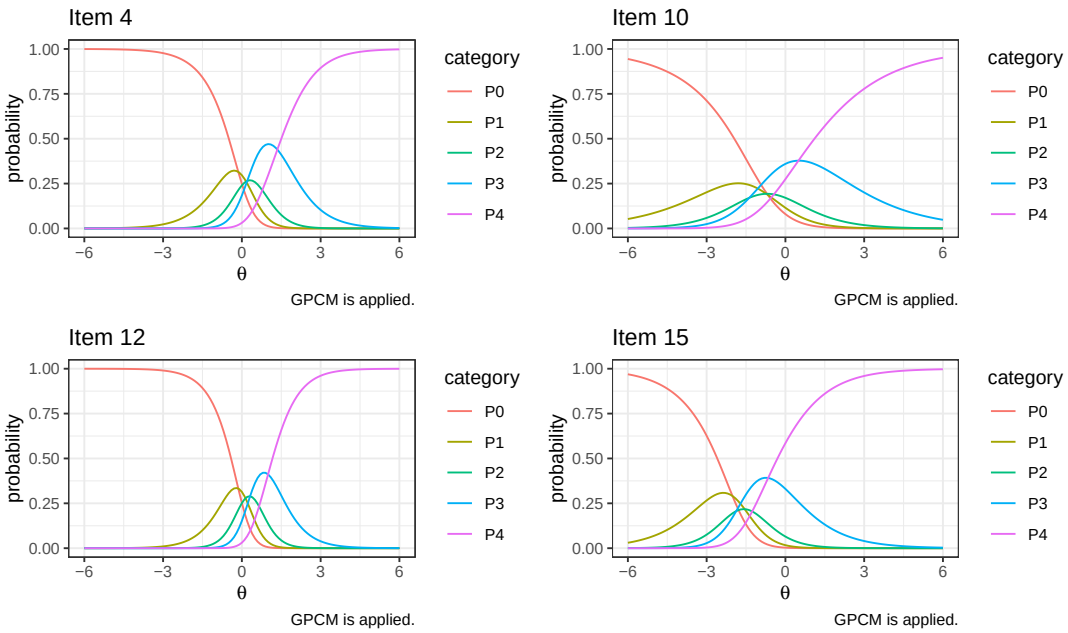



**Figure 5:** Item response functions of Item 4, 10, 12, and 15

**Plotting item response functions** Figure 5 shows item response functions of the four items (Item 4, 10, 12, and 15) by using `IRTest::plot_item()`. For example, a plot of the item response function of Item 11 can be drawn with `IRTest::plot_item(x = KDM, item.number = 11)`.

**Reliability coefficient** Among various types of IRT reliability coefficients, **IRTest** applies those discussed by Green et al. (1984) and May and Nicewander (1994). The coefficient from Green et al.

(1984) is calculated on the latent variable $\theta$ scale, and the one from May and Nicewander (1994) is calculated on the summed-score scale. May and Nicewander (1994)'s approach has an advantage of providing reliability coefficients for individual items. The function `IRTest::reliability()` returns the coefficients mentioned above: item reliability coefficients, and test reliability coefficients on the $\theta$ scale and the summed-score scale.

```
IRTest::reliability(KDM)

#> $summed.score.scale
#> $summed.score.scale$test
#> test reliability
#>        0.9293983
#>
#> $summed.score.scale$item
#>     Item1     Item2     Item3     Item4     Item5     Item6     Item7     Item8
#> 0.5039295 0.5248032 0.3818564 0.5740997 0.4227400 0.5468987 0.5526063 0.3795133
#>     Item9    Item10    Item11    Item12    Item13    Item14    Item15
#> 0.4929443 0.2898281 0.5196563 0.6437762 0.4386300 0.5232466 0.4021941
#>
#>
#> $theta.scale
#> test reliability
#>        0.9159476
```

**Item and test information functions** In **IRTest**, `IRTest::inform_f_item()` and `IRTest::inform_f_test()` evaluate item and test information, respectively. Figure 6 visually illustrates how each item contributes to the test information and how item information functions are added up all together to form the test information function. The test information function is drawn with a black line and the item information functions are colored.

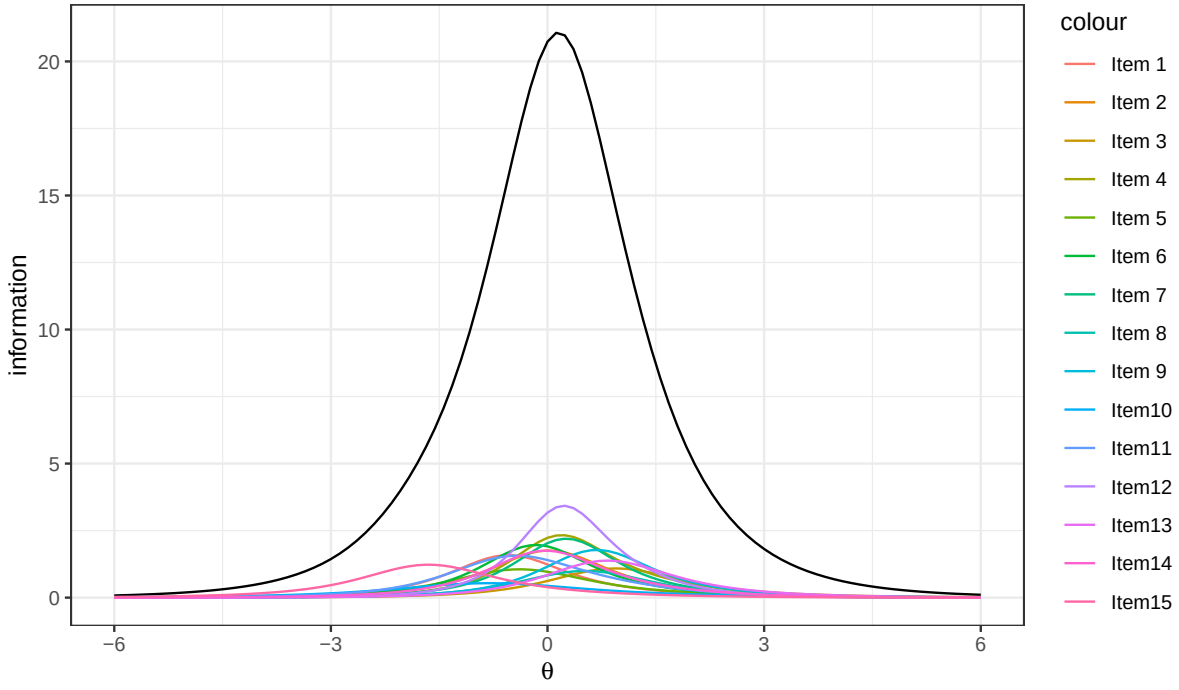



**Figure 6:** Item and test information functions

## 6 Discussion

While some may not find the magnitude of biases resulting from the deviations from the normality assumption particularly significant in the context of IRT, LDE remains an appealing option. Within IRT, increasing the number of items and respondents may require additional time and resources to improve parameter estimation accuracy. Meanwhile, an appropriate usage of LDE would increase

the estimation accuracy with the cost of adding a few distribution parameters. Therefore, unless the addition of distribution parameters compromises the model-data fit significantly, LDE can be a reasonable strategy for reducing estimation biases.

Although the current literature on LDE in IRT lacks clear guidelines on how and when to implement the LDE procedure, practitioners may choose to start with either KDM or DCM, even when there is limited or no information available about the shape of the latent distribution. Both approaches not only perform well in estimating non-normal latent distributions, but also effectively recover normal distributions when the normality assumption holds (Li, 2022; Woods and Lin, 2009).

The **IRTest** package offers the choice of LDE methods for conducting IRT analyses, and its functions utilize the estimated latent distribution for the subsequent analyses. **IRTest** is actively being updated, and some potential enhancements can be made for a better application of the package, which may include expanding the range of available IRT models, decreasing computation time, and imposing constraints on parameters. User feedback would also guide a way for the package maintenance and enhancement. Yet, there is much more to explore in the field of LDE beyond what has already been studied. Thus, further research on LDE is expected to provide valuable guidance to both package users and the developer.

## Acknowledgments

Special thanks to Nagap Park, Hyesung Shin, Ryan Lerch, and anonymous reviewers for their insightful suggestions that improved the article.

## Additional features

**Analysis of continuous response data** Continuous item responses, ranging from 0 to 1, are often encountered in test datasets. Such item responses can also be handled by the **IRTest** package with `IRTest::IRTest_Cont()`. All features of **IRTest** presented in this paper are applicable to continuous item responses. An example code is shown below:

```
# Generating a continuous item response data
data <- IRTest::DataGeneration(N = 1000, nitem_C = 10)$data_C

# Analysis
model_continuous <- IRTest::IRTest_Cont(data)
```

**Analysis of mixed-format data** An IRT analysis of mixed-format data can also be conducted with `IRTest::IRTest_Mix()`. The difference between `IRTest::IRTest_Mix()` and `IRTest::IRTest_Dich()` (or `IRTest::IRTest_Poly()`) is that `IRTest::IRTest_Mix()` requires two separate data: one for dichotomous items and the other for polytomous items. An example code is shown below:

```
model_mixed.format <- IRTest::IRTest_Mix(data_D = dichotomous_data,
                                         data_P = polytomous_data,
                                         model_D = rep(c("2PL", "3PL"), each = 5),
                                         model_P = "GPCM")
```

In this case, the 2PLM is applied to the first five dichotomous items, the 3PLM is applied to the rest of the dichotomous items, and the GPCM is applied to the polytomous items. The rest are the same with `IRTest::IRTest_Dich()` and `IRTest::IRTest_Poly()`.

The `IRTest::IRTest_Dich()` can also take a vector for the argument `model` to apply multiple IRT models to different types of items.

**Item-fit statistic** An item-fit statistic can be calculated with `IRTest::item_fit()`. Currently, Bock (1960)'s $\chi^2$ and Yen (1981)'s $Q_1$ are available.

## References

F. B. Baker and S.-H. Kim. *Item Response Theory: Parameter Estimation Techniques*. CRC press, 2nd edition, 2004. [p24, 25, 26]

A. Birnbaum. Some latent trait models and their use in inferring an examinee's ability. In F. M. Lord and M. R. Novick, editors, *Statistical Theories of Mental test Scores*, chapter 17, pages 397–479. Addison-Wesley, 1968. [p24]

D. R. Bock. Methods and applications of optimal scaling. University of North Carolina Psychometric Laboratory Research Memorandum, No. 25, 1960. [p38]

D. R. Bock and M. Aitkin. Marginal maximum likelihood estimation of item parameters: Application of an em algorithm. *Psychometrika*, 46(4):443–459, 1981. doi: 10.1007/BF02293801. [p23, 25, 26]

R. Brotherton, C. C. French, and A. D. Pickering. Measuring belief in conspiracy theories: The generic conspiracist beliefs scale. *Frontiers in Psychology*, 4, 2013. doi: 10.3389/fpsyg.2013.00279. [p31]

L. Cai. *flexMIRT version 3.65: Flexible Multilevel Multidimensional Item Analysis and Test Scoring [Computer software]*. Vector Psychometric Group, 2022. Chapel Hill, NC. [p23]

J. M. Casabianca and C. Lewis. *LLSEM 1.0: LogLinear Smoothing in an Expectation Maximization Algorithm for Item Response Theory Item Parameter Estimation [Computer software]*. Washington University in St. Louis, 2011. Bronx, NY. [p24]

J. M. Casabianca and C. Lewis. Irt item parameter recovery with marginal maximum likelihood estimation using loglinear smoothing models. *Journal of Educational and Behavioral Statistics*, 40(6): 547–578, 2015. doi: 10.3102/1076998615606112. [p23, 29]

P. R. 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. URL https://www.jstatsoft.org/index.php/jss/article/view/v048i06. [p24, 29]

C. C. Clogg. New developments in latent structure analysis. In D. J. Jackson and E. F. Borgatta, editors, *Factor analysis and measurement in sociological research*. Sage, 1981. [p23]

R. J. de Ayala. *The Theory and Practice of Item Response Theory*. The Guilford Press, New York, 2009. [p23]

C. Dudley-Marling. The tyranny of the normal curve: How the "bell curve" corrupts educational research and practice. In D. M. Allen and J. W. Howell, editors, *Groupthink in Science: Greed, Pathological Altruism, Ideology, Competition, and Culture*, chapter 17, pages 201–210. Springer, Cham, 2020. doi: 10.1007/978-3-030-36822-7_17. [p23]

D. Follmann. Consistent estimation in the rasch model based on nonparametric margins. *Psychometrika*, 53(4):553–562, 1988. [p23]

A. Gramacki. *Nonparametric Kernel Density Estimation and Its Computational Aspects*. Springer, 2018. [p29]

B. F. Green, D. R. Bock, L. G. Humphreys, R. L. Linn, and M. D. Reckase. Technical guidelines for assessing computerized adaptive tests. *Journal of Educational measurement*, 21(4):347–360, 1984. doi: 10.1111/j.1745-3984.1984.tb01039.x. [p36]

S. J. Haberman. Latent-class item response models (RR-05-28), 2005. [p23]

R. K. Hambleton, H. Swaminathan, and H. J. Rogers. *Fundamentals of Item Response Theory*. Sage, 1991. [p23]

E. J. Hannan and B. G. Quinn. The determination of the order of an autoregression. *Journal of the Royal Statistical Society B*, 41(2):190–195, 1979. doi: 10.1111/j.2517-6161.1979.tb01072.x. [p29, 34]

R. J. Harvey and W. D. Murry. Scoring the myers-briggs type indicator: Empirical comparison of preference score versus latent-trait methods. *Journal of Personality Assessment*, 62(1):116–129, 1994. doi: 10.1207/s15327752jpa6201_11. [p23]

A. D. Ho and C. C. Yu. Descriptive statistics for modern test score distributions: Skewness, kurtosis, discreteness, and ceiling effects. *Educational and Psychological Measurement*, 75(3):365–388, 2015. doi: 10.1177/0013164414548576. [p23]

M. C. Jones. The roles of ISE and MISE in density estimation. *Statistics & Probability Letters*, 12(1):51–56, 1991. doi: 10.1016/0167-7152(91)90163-L. [p31]

M. C. Jones, J. S. Marron, and S. J. Sheather. A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association*, 91(433):401–407, 1996. doi: 10.1080/01621459.1996.10476701. [p29]

N. Laird. Nonparametric maximum likelihood estimation of a mixing distribution. *Journal of the American Statistical Association*, 73(364):805–811, 1978. doi: 10.1080/01621459.1978.10480103. [p26]

S. Li. Using a two-component normal mixture distribution as a latent distribution in estimating parameters of item response models. *Journal of Educational Evaluation*, 34(4):759–789, 2021. doi: 10.31158/JEEV.2021.34.4.759. [p23, 25, 27, 28, 31]

S. Li. The effect of estimating latent distribution using kernel density estimation method on the accuracy and efficiency of parameter estimation of item response models, 2022. [p23, 26, 29, 38]

S. Li. *IRTest: Parameter Estimation of Item Response Theory with Estimation of Latent Distribution*, 2024. URL https://CRAN.R-project.org/package=IRTest. R package version 2.0.0. [p31]

G. N. Masters. A rasch model for partial credit scoring. *Psychometrika*, 47(2):149–174, 1982. doi: 10.1007/BF02296272. [p24]

K. May and A. W. Nicewander. Reliability and information functions for percentile ranks. *Journal of Educational Measurement*, 31(4):313–325, 1994. doi: 10.1111/j.1745-3984.1994.tb00449.x. [p36, 37]

A. L. McCutcheon. *Latent class analysis*. Sage, 1987. [p23]

R. J. Mislevy. Estimating latent distributions. *Psychometrika*, 49(3):359–381, 1984. doi: 10.1007/BF02306026. [p23, 26]

E. Muraki. A generalized partial credit model: Application of an em algorithm. *Applied Psychological Measurement*, 16(2):159–176, 1992. doi: 10.1177/014662169201600206. [p24]

G. Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests.* Danish Institute for Educational Research, Copenhagen, 1960. [p24]

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. [p29]

A. Robitzsch. *sirt: Supplementary Item Response Theory Models*, 2024. URL https://CRAN.R-project.org/package=sirt. R package version 4.1-15. [p24]

F. Samejima. *Estimation of Latent Ability Using a Response Pattern of Graded Scores (Psychometric Monograph No. 17)*. Psychometric Society, Richmond, VA, 1969. URL https://www.psychometrika.org/journal/online/MN17.pdf. [p34]

D. A. Sass, T. A. Schmitt, and C. M. Walker. Estimating non-normal latent trait distributions within item response theory using true and estimated item parameters. *Applied Measurement in Education*, 21(1):65–88, 2008. doi: 10.1080/08957340701796415. [p23]

T.-J. Seong. Sensitivity of marginal maximum likelihood estimation of item and ability parameters to the characteristics of the prior ability distributions. *Applied Psychological Measurement*, 14(3):299–311, 1990. doi: 10.1177/014662169001400307. [p23]

S. J. Sheather. Density estimation. *Statistical Science*, 19(4):588–597, 2004. doi: 10.1214/088342304000000297. [p29]

S. J. Sheather and M. C. Jones. A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society B*, 53(3):683–690, 1991. doi: 10.1111/j.2517-6161.1991.tb01857.x. [p29]

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, Boca Raton, FL, 1986. [p29]

W. J. van der Linden. *Handbook of Item Response Theory: Three Volume Set*. CRC Press, 2016. [p23]

M. P. Wand and C. M. Jones. *Kernel Smoothing*. Chapman & Hall, Boca Raton, FL, 1995. [p29]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p35]

C. M. Woods. *RCLOG v.2: Software for Item Response Theory Parameter Estimation with the Latent Population Distribution Represented Using Spline-based Densities [Computer software]*. Washington University in St. Louis, 2006a. St. Louis, MO. [p23]

C. M. Woods. Ramsay-curve item response theory (rc-irt) to detect and correct for nonnormal latent variables. *Psychological Methods*, 11(3):253–270, 2006b. doi: 10.1037/1082-989X.11.3.253. [p23]

C. M. Woods. Estimating the latent density in unidimensional irt to permit non-normality. In S. P. Reise and D. A. Revicki, editors, *Handbook of Item Response Theory Modeling: Applications to Typical Performance Assessment*, chapter 4, pages 60–84. Routledge, New York, 2015. doi: 10.4324/9781315736013-4. [p23, 27]

C. M. Woods and N. Lin. Item response theory with estimation of the latent density using davidian curves. *Applied Psychological Measurement*, 33(2):102–117, 2009. doi: 10.1177/0146621608319512. [p23, 27, 28, 29, 38]

X. Xu and M. von Davier. Fitting the structured general diagnostic model to naep data (RR-08-27), 2008. [p23, 29]

A. Yadin. Using unique assignments for reducing the bimodal grade distribution. *ACM Inroads*, 4(1): 38–42, 2013. doi: 10.1145/2432596.2432612. [p23]

W. M. Yen. Using simulation results to choose a latent trait model. *Applied Psychological Measurement*, 5 (2):245–262, 1981. doi: 10.1177/014662168100500212. [p38]

D. Zhang and M. Davidian. Linear mixed models with flexible distributions of random effects for longitudinal data. *Biometrics*, 57(3):795–802, 2001. doi: 10.1111/j.0006-341X.2001.00795.x. [p28]

M. F. Zimowski, E. Muraki, R. J. Mislevy, and D. R. Bock. *BILOG-MG: Multiple-group IRT Analysis and Test Maintenance for Binary Items [Computer software]*. Scientific Software International, 2003. [p23]

*Seewoo Li*
*University of California, Los Angeles*
*Social Research Methodology, Department of Education*
*ORCiD: 0000-0002-6290-2777*
seewooli@g.ucla.edu

# Splinets – Orthogonal Splines for Functional Data Analysis

*by Rani Basna, Hiba Nassar, and Krzysztof Podgórski*

**Abstract** This study introduces an efficient workflow for functional data analysis in classification problems, utilizing advanced orthogonal spline bases. The methodology is based on the flexible Splinets package, featuring a novel spline representation designed for enhanced data efficiency. The focus here is to show that the novel features make the package a powerful and efficient tool for advanced functional data analysis. Two main aspects of spline implemented in the package are behind this effectiveness: 1) Utilization of Orthonormal Spline Bases – the workflow incorporates orthonormal spline bases, known as splinets, ensuring a robust foundation for data representation; 2) Consideration of Spline Support Sets – the implemented spline object representation accounts for spline support sets, which refines the accuracy of sparse data representation. Particularly noteworthy are the improvements achieved in scenarios where data sparsity and dimension reduction are critical factors. The computational engine of the package is the dyadic orthonormalization of B-splines that leads the so-called splinets – the efficient orthonormal basis of splines spanned over arbitrarily distributed knots. Importantly, the locality of *B*-splines concerning support sets is preserved in the corresponding splinet. This allows for the mathematical elegance of the data representation in an orthogonal basis. However, if one wishes to traditionally use the *B*-splines it is equally easy and efficient because all the computational burden is then carried in the background by the splinets. Using the locality of the orthogonal splinet, along with implemented algorithms, the functional data classification workflow is presented in a case study in which the classic Fashion MINST dataset is used. Significant efficiency gains obtained by utilization of the package are highlighted including functional data representation through stable and efficient computations of the functional principal components. Several examples based on classical functional data sets, such as the wine data set, showing the convenience and elegance of working with Splinets are included as well.

## 1 Introduction

In functional data analysis (FDA), it is often desired that functions considered are continuous or even differentiable up to a certain degree . From this perspective, spline functions given over a set of knots form convenient finite-dimensional functional spaces. There exist many R-packages that handle splines, see (Perperoglou et al., 2019). However, none of them consistently treat splines as elements of functional spaces and provide convenient functionally represented orthogonal bases. The recent package, **Splinets** (Basna et al., 2025), approaches splines exactly like this by using an object that represents a set of functional splines and provides efficient orthogonal bases as such objects. This focus on the functional form of splines and functional analysis approach to them makes **Splinets** particularly suitable for functional data analysis. Care was taken to make this functional treatment efficient by first accounting for support sets of spline, then using efficient and novel orthogonal bases.

While the locality of the representation, expressed by the support sets, makes the package suitable for the analysis of data distributed sparsely over their domain, its most important feature is a direct access to orthogonal spline bases. The *B*-spline bases, (de Boor, 1978; Schumaker, 2007), are available as well, and efficiency is achieved by explicitly using the support in implemented algebra and calculus of splines. However, there is a fundamental problem with the *B*-splines that introduces a computational burden to functional decomposing – the *B*-splines are not orthogonal. Since any basis in a Hilbert space can be orthogonalized, it is to the point of considering orthonormalization of the *B*-splines, (Nguyen, 2015; Goodman, 2003; Cho and Lai, 2005). In (Liu et al., 2022), a new natural orthogonalization method for the *B*-splines has been introduced and an efficient dyadic algorithm has been proposed for this purpose. The orthogonalization was argued to be the most appropriate since it, firstly, preserves most from the original structure of the *B*-splines and, secondly, obtains computational efficiency both in the basis element evaluations and in the spectral decomposition of a functional signal.

The main engine of the package is a new orthogonalization method of he *B*-splines described in (Liu et al., 2022). The method is fundamentally different from and superior to one-sided and two-sided orthogonalization discussed in (Mason et al., 1993) that are also implemented in the package. Since the constructed basis spreads a net of splines rather than a sequence of them, the term *splinet* was coined when referring to such a base. It is formally shown that the splinets, similarly to the *B*-splines, feature a locality due to the small size of the total support, an example of such a splinet spanned on irregularly placed knots is shown in Figure 1 *(Left)*. The figure shows the splinet on the dyadic graph. Using the dyadic presentation with different levels allows for a better insight into the structure of the data and connects to the dyadic algorithm used for generation of the graph, see (Liu et al.,

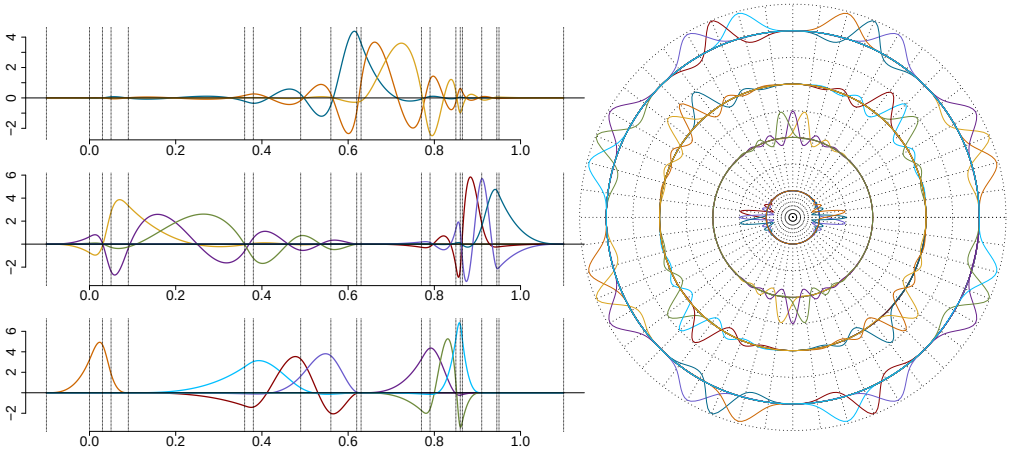


**Figure 1:** *Splinets* - the orthogonal cubic spline bases presented on dyadic levels: *(Left)* A splinet made on irregularly spaced knots on an interval, not a complete dyadic case; *(Right)* Periodic splinet on regularly spaced knots presented on circular dyadic levels, a complete dyadic case of 48 O-splines.

2022). However, the presentation of the basis elements on a single coordinate linearly is also possible and perhaps more natural from the interpretational point of view, see also comparison of the two graphical representation shown in Figure 3. The dyadic graphical format finds its root in the dyadic algorithm used for orthogonalization of the *B*-splines, see (Liu et al., 2022). Here, it is used mostly for its visualization value. Beyond splines defined on a subset of real line one can also consider splines defined on a circle, i.e. periodic splines, see (Nassar and Podgórski, 2025). These are also implemented in the package, as shown in Figure 1 *(Right)* but the focus here is on the functional data on the real line. The following code shows how one can easily access both splinets and their graphical representation

```
xi <- c(-0.1, 0.0, 0.03, 0.05, 0.09, 0.36, 0.38, 0.49, 0.56, 0.62,
        0.63, 0.77, 0.79, 0.85, 0.86, 0.865, 0.91, 0.945, 0.95, 1.1) # Knots
so <- splinet(xi)  # Orthonormal basis
plot(so$os)
N <- 4  # Number of levels in the dyadic structure
k <- 3  # Spline degree
n_knots <- 2^N * 3 - 1  # Number of internal knots in the dyadic case
pxi <- seq(0, 1, length.out = n_knots + 2)  # Knots
pso <- splinet(xi, periodic = TRUE)  # Periodic orthonormal basis
plot(pso$os)
```

Our goal is to demonstrate that all these features of the **Splinets** package make it a powerful and effective tool for functional data analysis (FDA). Not only does the package allow for standard methods to be easily implemented, but it also facilitates the exploration of new methodologies that are specific to the choice of splines as functional spaces for analysis. In fact, the workflow for the classification problem is presented in this paper to demonstrate the capabilities of the **Splinets** package in various aspects of functional data. The approach is used in a case study of classifying images.

The organization of the material is as follows. The paper starts with the Taylor expansion based functional representation of splines used in the package implementation. A discussion of the importance of the spline space selection for the data at hand follows. The knot selection is highlighted in this discussion. Once a proper spline space is chosen, the projection to the space becomes the main tool in the data analysis as it produces the isometry between functional spaces and Euclidean spaces. Thus a presentation of the projection of functions to the space of splines by means of the developed spline bases is discussed next. Applications of these basic methods to perform a standard functional analysis through some classical data sets illustrate the methodology. The most advanced implementation of the FDA is presented in the final section, where the workflow for the classification problem is elaborated and applied through a case study based on the fashion MNIST data.

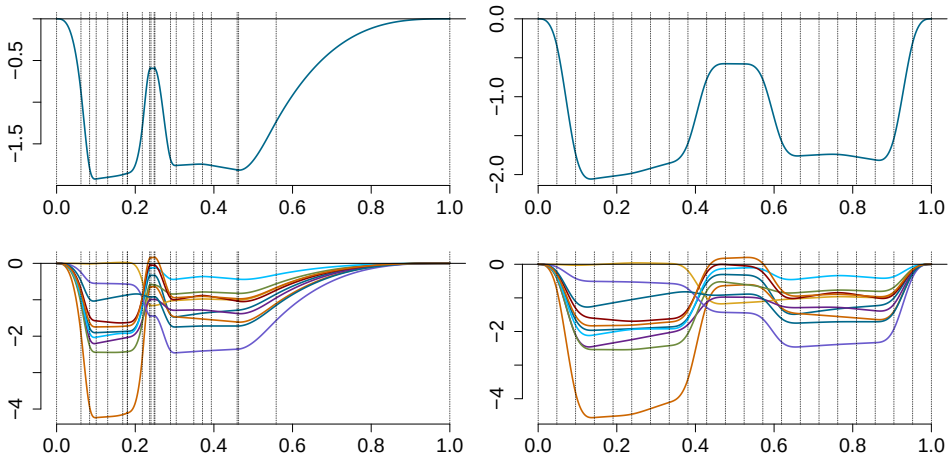



**Figure 2:** Examples of splines for non-equally spaced knots (left) and equally spaced knots (right). In the top graphs, we see individual splines while in the bottom ones random spline samples are generated around the top one by the random spline generator function implemented in the package. The vertical dashed lines in all figures are placed at the locations of the knots.

## 2 Splinets implementation

The splines can be represented in a variety of ways. In Qin (2000), a general matrix representation was proposed that allows for efficient numerical processing of the operations on the splines. This was utilized in Zhou et al. (2008) and Redd (2012) to represent and orthogonalize *B*-splines that were implemented in the R-package **orthogonalsplinebasis** (Redd, 2022). In our approach, we propose to represent a spline in a different way. Namely, we focus on the values of the derivatives at knots and the support of a spline. The goal is to achieve better numerical stability as well as to utilize the discussed efficiency of base splines having support only on a small portion of the considered domain. In fact, there are three sources of the efficiency and computational stability of the package **Splinets**:

**Taylor expansion representation** Representing a spline by the matrix of the derivatives at the knots makes use the local Taylor expansion.

**Sparse domains** Implementing partial support sets for splines as well as the spline algebra and calculus that is based on support sets.

**Efficient orthogonal bases** Using the dyadic algorithm to orthogonalize the *B*-splines and utilize them in the projection methods.

The first two features are implemented in the `"Splinets"` class, the third one is programmed through an algorithm of intelligent orthogonalization of *B*-splines. The efficiency and stability of the representation based on the local Taylor expansions is mathematically justifiable since the matrix of the derivatives at the knots translate directly to the function behaviour locally between knots. However, we did not explicitly compare computational efficiency with other spline implementations because these features are unique and, to our best knowledge, they are not covered in other packages. Some details are given in what follows, while the topic is more completely elaborated in (Podgórski, 2021).

In the following code, a random sample of splines is generated using spline generator implemented in the package. The result is shown in Figure 2 and demonstrates that the knot distribution is fundamental in shaping geometry of functional spaces. The difference in the nature of the two linear functional spaces is obvious: in the left graphs we see more focus on the localized variability and detail, the capacity that is lacked in splines in the right-hand side. From the analyst point of view, the challenge is to choose a space that matches geometrical characteristics observed in the data.

```
n <- 20  # Number of knots
k <- 3   # Degree of splines
set.seed(10)
xi <- sort(rbeta(n + 2, 2, 5))  # Randomly distributed knots
xi[1] <- 0
xi[n + 1] <- 1
S <- matrix(rnorm((n + 2) * (k + 1)), ncol = (k + 1))  # Random matrix of derivatives
spl <- construct(xi, k, S)  # A spline object with corrected derivatives matrix
plot(spl)
y <- rspline(spl, 10)  # Random spline generator
```

```
plot(y)
xi2 <- seq(0, 1, by = 1 / (n + 1))
spl2 <- construct(xi2, k, S)  # Another spline object
plot(spl2)
y2 <- rspline(spl2, 10)  # Another random sample
plot(y2)
```

The implementation of a sparse domain is a novel feature not present, as far as we know, in other spline implementations. The numerical gains are obvious since we avoid numerical evaluations of operations that are known to be zero. In this sense it is similar to the packages for sparse matrices, where zero terms are omitted in the matrix representations. The benefits for sparse functional data is illustrated in Figure 4- *(Bottom-right)*. Additional efficiency in handling functional basis with locality features such as *B*-splines and splinets lies in using the sparse domain feature. Indeed, the elements of such bases are non-zero only on small portions of the domain and in evaluating the projection to the spline spaces is significantly faster if the unnecessary operations on zeros are omitted.

Finally, the package implements, for the first time, an innovative orthogonalization through the numerical efficiency of the dyadic algorithm. The orthogonal bases obtained in the process are discussed in more detail further in the paper while the algorithm itself and its computational benefits have been demonstrated in (Liu et al., 2022).

### 2.1 Splines with zero-boundary conditions at the endpoints

Splines are piecewise polynomials of a given degree with continuous derivatives up to the degree (exclusive) at the points they interconnect, which are called *internal knots*. The domain of a spline will be called its *range* and is assumed in the paper to be a closed finite interval. Additional knots called the *initial knots* and the *terminal knots* are located at the beginning and the end of the spline range, respectively. For a given set of knots, the space of splines is finite-dimensional, with the inner product of the Hilbert space of the square-integrable functions.

It is worth mentioning two extensions of functional spaces that are handled by the package. The first are splines that do not have full support over the entire range of the considered domain. This can be of special utility for the sparse functional data, i.e. the functional data that are nonzero only on a small number of locations. A good example of such data are mass spectra in proteomics or metabolomics analyses. The implementation of the `"splinet"` object allows specifying a support of functional spline to be only on a union of disjoint intervals inside of the entire range of domain. The second generalization extends beyond the splines since the `Splinets` object can be used to represent any piecewise polynomial function of a given degree. All the relevant functions work properly even if the smoothness at the knots is not preserved.

Due to the continuity requirements, the behaviour of a spline between two given knots is necessarily affected by the form of polynomials at the neighboring between-knots subintervals. Since the between-knots intervals with terminal knots have only one neighboring between-knots interval, the influence of values over other intervals is not the same. To mitigate this biased terminal knots effect, it is natural and, as it will be seen, also mathematically elegant to introduce the zero boundary conditions at the terminal knots for all the derivatives except the highest degree one. It can be shown that to remove the zero boundary effect, one has to consider splines over the knots obtained through extending by a certain number of knots from both ends of the complete set of knots. More specifically, the number of knots that have to be added at each end is equal to the degree of the splines. Often the knots are added by replicating the terminal knots although there are some serious disadvantages of such an approach.

The splines involve knots at which the polynomials smoothly connect. A set of such knots is represented as a vector $\boldsymbol{\xi}$ of $n+2$ ordered values. As already mentioned, there are two alternatives, but in a certain sense equivalent, requirements on the behaviour of a spline at the endpoints of its range. In the first, no boundary conditions are imposed. The main problem in this unrestricted approach is that the polynomials at both ends of the considered range do not 'sense' the same restrictions from the neighbors as the polynomial residing further from both endpoints. This is due to the fact that at the endpoints the 'neighbors' are only present from one side. Another approach, favored in this work, is by putting zero boundary restrictions on the derivatives at the endpoints. This approach is mathematically equivalent to the first one in a limiting sense, when the $k$ initial knots and the $k$ terminal knots converge to the beginning and the end of the range, respectively. Moreover and most importantly, the approach is structurally elegant and thus easier to implement. For all these reasons, it is used in our package. However, the main object of the package which is `"splinets"`.

We impose on a spline and all its derivatives of the degree smaller than the spline degree the value of zero at both the endpoints of the range. In this case, if we consider knots $\boldsymbol{\xi} = (\xi_0, \ldots, \xi_{n+1})$ it is important to assume that $n \geq k$, in order to have enough knots to define at least one non-zero spline

with the $2k$ zero-boundary conditions at the endpoints. Indeed, if $n = k - 1$, then we have $k + 1$-knots yielding $k$ between knot intervals. On each such interval, a spline is equal to a polynomial of degree $k$. The dimension of the space of such piecewise polynomial functions is $k(k + 1)$. However, at each internal knot, there are $k$ equations to make derivative up to degree $k$ (excluding) continuous. This reduces the initial unrestricted polynomials dimension by $(k - 1)k$ dimensions to $2k$, but there are $2k$ equations for the derivatives to be zero at the endpoints. We conclude that the dimension of the spline space is eventually reduced to zero, meaning that there is only a function trivially equal to zero in this space. From now on $\mathcal{S}_k^{\boldsymbol{\xi}}$ stands for the $n - k + 1$ dimensional space of the $k$-smoothed splines with the zero boundary conditions at the terminal knots of in the ordered knots given in vector $\boldsymbol{\xi}$. When the dependence on either $k$ or $\boldsymbol{\xi}$ or both is not important, they will be dropped from the notation.

The fundamental fact that we use here is that for a given degree , say $k$, and a vector of knot points $\boldsymbol{\xi} = (\xi_0, \ldots, \xi_{n+1})$, a spline $S \in \mathcal{S}$ is uniquely defined by the values of the first $0, \ldots, k$ derivatives at the knots. The values of derivatives at the knots allow for the Taylor expansions at the knots but they cannot be taken arbitrarily due to the smoothness at the knots. The matrix of the derivative values is the main component of an object belonging to our main class in the package. For any spline function $S \in \mathcal{S}_k^{\boldsymbol{\xi}}$, we consider $\mathbf{s}_j = (s_{0j}, \ldots, s_{n+1j})$ is an $n + 2$-dimensional vector (column) of values of the $j^{\text{th}}$-derivative of $S$, $j = 0, \ldots, k$, at the knots given in vector $\boldsymbol{\xi} = (\xi_0, \ldots, \xi_{n+1})$ that, as a general convention for all vectors (also the convention in R, is treated as a $(n + 2) \times 1$ column matrix. These columns are kept in a $(n + 2) \times (k + 1)$ matrix

$$\mathbf{S} := [\mathbf{s}_0\ \mathbf{s}_1\ \ldots\ \mathbf{s}_k] \tag{1}$$

More specifically, the main object `"Splinets"` implemented through the S4 system for the OOP in R is defined through `setClass` function with the following slots

```
representation(knots = "vector", degree = "numeric", equid = "logical", type =
    "character", supp = "list", der = "list", taylor = "matrix", epsilon = "numeric")
```

which represents a collection of splines, all built over the same knots given in `knots`, of the degree $k$ given in `degree`. Further `supp` is the list of matrices having row-wise pairs of the endpoints of the intervals, the union of which constitutes the support set of a particular spline, and the flag `equid` informs about the equally placed knots, for which the computation can be significantly accelerated. The matrices of the derivatives at the knots inside the support sets are given in the list `der` of matrices, where an element in the list refers to a particular spline in our collection and the length of the list corresponds to the number of splines in the object. Descriptions of other fields are given in the **Splinets** package but are not crucial for this presentation.

Consequently, the above object corresponds to a spline function $S$ of degree $k$ over knots in $\boldsymbol{\xi} = (\xi_0, \ldots, \xi_{n+1})$ that is identified as

$$S = \{k, \boldsymbol{\xi}, \mathcal{I}, \mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_k\},$$

where $\mathcal{I} = \{(i_1, i_1 + m_1 + 1), \ldots, (i_N, i_N + m_N + 1)\}$ is a sequence of ordered pairs of indexes in $\{1, \ldots, n + 2\}$ representing the intervals, the union of which is the support of a spline, i.e. the minimal closed set outside of which spline vanishes.

## 3 Spline spaces for FDA

One of the fundamental aspects of functional data analysis is the decision in which functional space the functional data (FD) in hand are assumed to be located. FD are not observed as continuous objects, but high-frequency sampling and mathematical efficiency enable us to see these data as samples of curves, surfaces, or anything else varying over a continuum. The fundamental step in FDA is to convert this discrete recorded data to a truly functional form, which allows each function to be evaluated at any value of its continuous argument. In order to utilize the topology of such data for dimension reduction, one must perform data conversion. Typically, one represents a functional object as a linear combination of a suitably chosen basis functions. For this purpose, one of the standard bases such as trigonometric, wavelet, or polynomial is typically chosen. Then the efficiency is accomplished by using smoothing through regression or roughness penalty for estimating the coefficients of the basis expansions.

One can argue that the spline bases with their flexibility that comes from the knot selection and the degree of splines make these spaces ideal for efficiency in retrieving the functional structure of a studied model, see also (Basna et al., 2022). The splines involve knots at which the polynomials smoothly connect. A set of such knots is represented as a vector $\boldsymbol{\xi}$ of $n + 2$ ordered values. In this work, we consider the zero-boundary version of splines by putting zero boundary restrictions on the

derivatives at the endpoints. This approach is mathematically equivalent to the first one in a limiting sense, when the $k$ initial knots and the $k$ terminal knots converge to the beginning and the end of the range, respectively. Moreover and most importantly, the approach is structurally elegant and thus easier to implement. For all these reasons, it is used in our package. The central object of the package is `"splinets"`, which represents a sequence of splines and holds it in the object by coefficients of the Taylor expansion of the function at the knots, which uniquely defines a spline using a matrix of derivatives at the knots.

The fundamental feature of the package and of the approach to FDA are the efficient orthogonal bases, the so-called splinets, which allow for a very efficient data representation. These bases that were introduced in (Liu et al., 2022) are discussed next.

### 3.1 Splinets – orthonormal bases of splines

The direct approach to building splines requires a lot of care and often can be cumbersome. A more efficient approach is through functional bases of splines. There are many possible choices of such bases but the most popular are the $B$-splines. Despite having many advantages, the $B$-splines do not constitute an orthogonal basis, which is a computational problem as the projection to the non-orthogonal bases adds to the numerical complexity. This challenge occurs in implementing the orthogonal projection operator discussed in the next subsection. On the other hand, using the orthogonality of the splinet allows an efficient representation of the data also in $B$-splines because the matrix conversion from one basis to another is sparse. The main feature of the package is to implement an optimal orthogonalization of the $B$-splines into the orthogonal bases called *splinets* introduced in Liu et al. (2022). The graphical presentation of the spline basis visually benefits from organizing them in the form of a dyadic net that corresponds to the dyadic orthogonalization procedure applied to the $B$-splines, the technical definition of the dyadic net and further details can be found in Liu et al. (2022). However, this dyadic net representation is used mostly for the visualization and one can equally use the sequential representation if preferred. The package facilitates simple switching between the two representations. In Figure 3, one can see $B$-spline basis in the left graphs and the corresponding splinet in the right. In the top graphs, we use the sequential graphical presentation and in the bottom ones the dyadic form that allows for a better inspection of the basis functions.

The orthonormalized bases implemented in this package are obtained by one of the following three orthogonalization procedures applied to $B$-splines. The first one is simply the Gram-Schmidt orthogonalization performed on the $B$-splines ordered by their locations, the second one is a symmetric (with respect to the knot locations) version of the Gram-Schmidt, and, finally, the dyadic orthogonalization into a *splinet* which is our preferred method. We will not discuss the first two orthonormalization methods as they have been included in the package mostly because of historical reasons. In the object representation of collections of splines, i.e. in the `"Splinets"` class, the field `type` specifies which of the orthonormal basis one deals with. The function `splinet()` is generating the proper basis as illustrated the code that creates the $B$-splines, the corresponding splinet and the graphical illustration shown in Figure 3

```
k <- 3        # Degree of splines
N <- 5        # Number of layers
n <- k * 2^N - 1  # Number of knots
set.seed(2)
xi <- cumsum(runif(n + 2, min = 0.2))  # Random knots
so <- splinet(xi, k)  # Evaluation of the B-splines and the splinet
plot(so$bs, type = "dyadic")  # B-splines on the dyadic net
plot(so$os)                   # Splinet on the dyadic net
```

One can observe that `splinet()` is returning a list of two `Splinets` objects, `so$bs` and `so$os` build over the ordered knots `xi`. The first object represents the basis of the standard cubic $B$-splines and is thus not orthogonal. The second one represents the recommended orthonormal basis, which is referred to as a cubic splinet. In Figure 3 *(Right)*, one can see the splinet obtained from the $B$-splines given in the right-hand side.

### 3.2 Projection to space of splines

Splines with a given knot selection constitute finite-dimensional subspaces of the Hilbert space of square-integrable functions. Thus, any such function can be projected orthogonally into the space of splines defined by a particular set of knots. Functional data analysis typically begins by projecting discretized data onto such a finite-dimensional functional space, making the projection a fundamental operation in statistical analysis. Projections can also serve as a smoothing step, and various methods are available for this purpose.

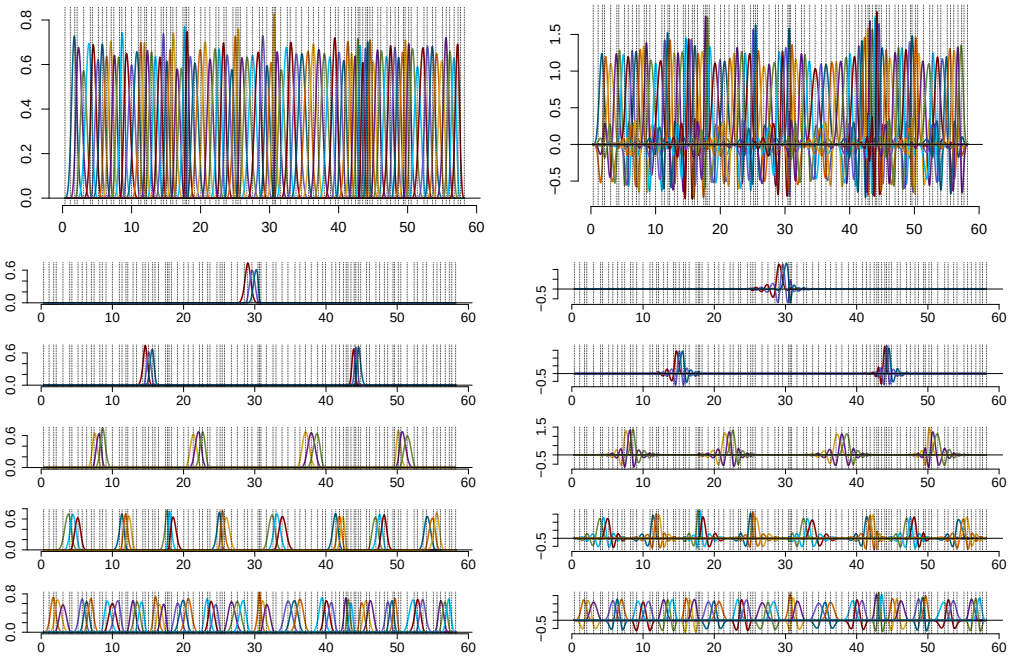


**Figure 3:** Cubic spline bases presented graphically in the sequential form (top) and on the dyadic net (bottom). The case of $n = k2^N - 1 = 95$, $k = 3$, $N = 5$ which is the number of layers in the dyadic structure seen in the figures. *(Left)*: The *B*-spline basis; *(Right)*: The corresponding splinet.

In the package, orthogonal projection is implemented via the function `project()`. Since functional data may be represented in different ways, the actual projection depends on the input format, especially how the inner product with a spline is evaluated numerically. The input to `project()` can either be `Splinets` objects or a matrix with two columns representing the arguments and values of discretized functional data.

The output of the function is a list with three components:

`projsp$coeff` A matrix of coefficients for the decomposition in the selected basis;

`projsp$basis` The `Splinets` object representing the selected basis;

`projsp$sp` The `Splinets` object representing the projection of the input data.

Additional details—such as the knots, degree, and basis type—can be retrieved from `projsp$basis`. Many algebraic operations on splines are more efficiently performed on the coefficient matrix rather than directly on the `Splinets` objects. The matrix `projsp$coeff` can be used for such tasks, while linear combinations of `projsp$basis` yield the corresponding functional forms.

To highlight the simplicity and utility of **Splinets**, we apply it to mass spectrometry data in a basic numerical example. This class of data is particularly suited to the package due to its sparse nature and the computational advantages of localized spline bases. We consider a low-resolution SELDI-TOF mass spectrum available from the National Cancer Institute, Center for Cancer Research, Clinical Proteomics Program (`https://home.ccr.cancer.gov/ncifdaproteomics/ppatterns.asp`, accessed 2024-05-26). For related work using such data, see Petricoin et al. (2002).

In Figure 4, we present flexible ways of representing the original signal by different **Splinets** objects. From these simple illustrations, one can see dramatic reduction from the original 15154 dimension of the data to three different 200 dimensional spline representations that emphasize different and subtle feature in the data. The focus here is on the efficiency and flexibility by showing that 200 dimensional spline spaced can be molded toward particular features of interest. We conclude that even without any in-depth FDA, one can get a significant simplification and dimension reduction of the data at the data representation level. In this work, we show that also advanced FDA can be easily implemented with the help of the package.

The simplicity of using the **Splinets** package can be seen again in the following code examples. If the `Ovarian` data and the **Splinets** package are loaded, then the projection to splines with equally spaced knots is simply obtained by

```
xi1 <- seq(min(Ovarian\$ms), max(Ovarian\$ms), length.out = 200) # Equally spaced knots
```

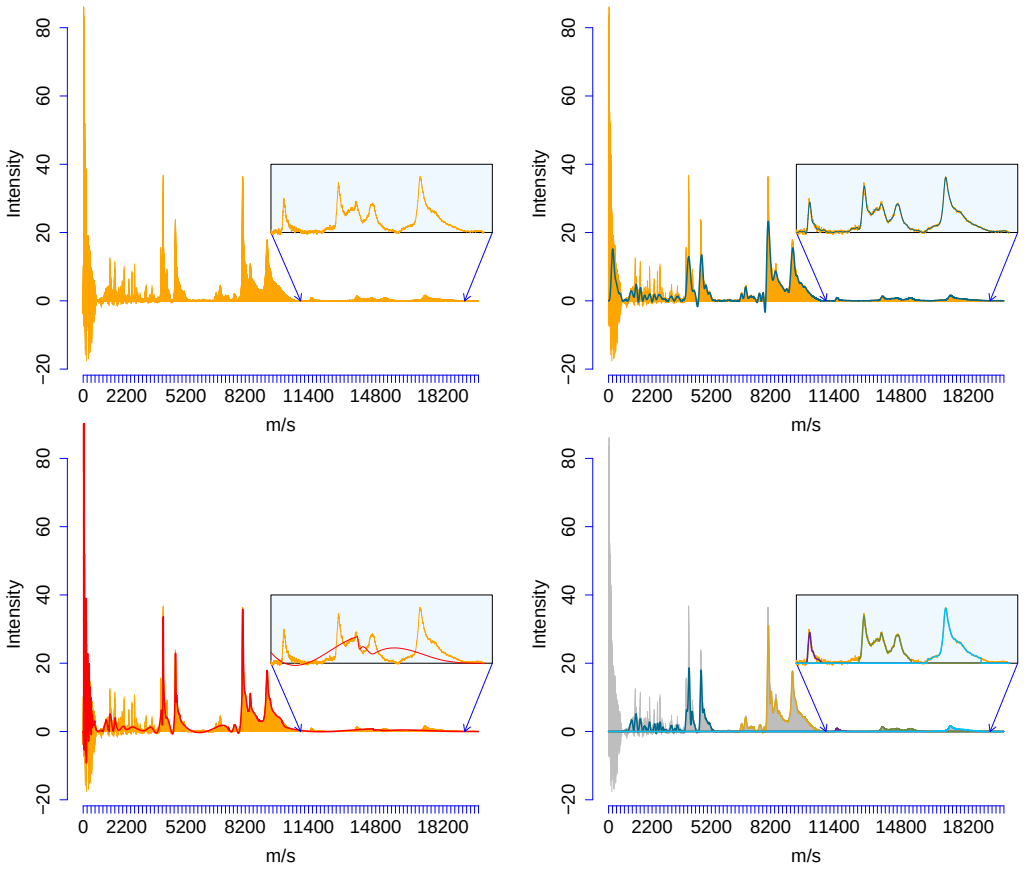



**Figure 4:** An ovarian cancer tissue low-resolution SELDI-TOF mass spectrum (in orange) and its various representations as *Splinet*-objects in approximately 200 dimensional spline spaces: *(Top-Left)* The original sample (orange) consisting of 15154 values; *(Top-Right)* The projection (navy-blue) to the spline space spanned on 200 equally spaced knots; *(Bottom-Left)* The projection (red) to the spline space spanned on 200 non-equally spaced knots with their locations chosen by the spectrum; *(Bottom-Right)* The projection to the splines with a sparse domain obtained by specifying five importance regions, with the graph above each of the five support intervals drawn by a different color.

```
so1 <- splinet(xi1)                                          # Orthogonal basis of splines
OvSpl1 <- project(as.matrix(Ovarian), basis = so1\$os)          # Projection to the basis
```

the projection to the data driven choice of knots by

```
wghts <- abs(Ovarian\$Intensity)/sum(abs(Ovarian\$Intensity))  # Knot selection weights
xi2 <- sort(sample(Ovarian\$ms, 200, prob = wghts))             # Random knots
so2 <- splinet(xi2)                                          # Orthogonal basis of splines
OvSpl2 <- project(as.matrix(Ovarian), basis = so2\$os)          # Projection to the basis
```

and the representation by a spline `OvSpl4` with a sparse domain with five interval components, which requires a bit more coding, is presented only on the graphs while the code is available in the package vignette at https://ranibasna.github.io/R-Splinets/articles/MassSpectrometry.html.

The omission of the first group of knots from the support is due to its noisy character that typically is not of interest in the mass-spectrometry data analysis.

The package provides a plot method for `Splinet` objects, so all the plots from Figure 4 can be simply obtained through:

```
plot(Ovarian\$ms, Ovarian\$Intensity, type = "h")
lines(OvSpl1\$sp)
lines(OvSpl2\$sp)
lines(OvSpl3)
```

The complete code, which facilitates experimentation, demonstrates data analysis, performs the classification procedure, and allows for the recreation of all figures and results, is available as R files with comments. The R code is provided in the supplementary materials on the GitHub page

https://github.com/ranibasna/R-Splinets with the accompanied package site https://ranibasna.github.io/R-Splinets/index.html.

In the discussion of the projection to the spline spaces, we discuss four different types of the projection: embedding, basis decomposition, spline projection, and discretized data projection. This functionality is illustrated by a numerically simulated example.

*Embedding a spline into higher-dimensional spaces of splines.* The first function that we discuss is an embedding rather than a projection. One of the most interesting aspects of spline spaces is their agility attributed to different choices of the knots. Yet, when employing splines for functional data analysis, the focus is typically on splines with a fixed, often equally spaced, set of knots, rather than exploring this adaptability. In the proposed package, we provide tools to thoroughly explore the properties of splines under various knot choices. Any spline remains a spline of the same *degree* when considered on a set of knots larger than the original. However, this changes the `Splinets` object representation of the refined spline. It is thus important to have a function that embeds a given spline into a larger space of splines defined on a refined set of knots. In the package, the function `refine()` allows splines from smaller spaces to be conveniently represented in larger, more refined spaces.

*Basis decomposition.* The simplest projection obtained through `project()` is also not, strictly speaking, a projection but rather a decomposition of a `Splinets` object into coefficients in the given basis. If `sp` is a `Splinets` object, then the code

```
bdsp <- project(sp)
bdsp2 <- project(sp, type = "bs")
```

have as its main output the matrices of coefficients $a_{ji}$, such that the $j^{\text{th}}$ input-spline $S_j$ that is in the input `sp` has the form

$$S_j = \sum_{i=1}^{n-k+1} a_{ji} B_i,$$

where $j$ indexes the input splines, $n$ is the number of the internal knots (the knots not including the two endpoints of the domain), $k$ is the degree of splines, and $B_i$'s are the elements of the selected basis of splines controlled by the input `type`, so in the above example it is either the splinet (`bdsp`) or the $B$-spline basis (`bdsp2`). The bases are built on the same knots as the input spline.

*Projecting splines.* The projection of `Splinets`-objects over a given set of knots to the space of splines over a different set of knots is obtained through the orthogonal projection:

```
bdsp <- project(sp, knots)
bdsp2 <- project(sp, knots, type = "bs")
```

The results are represented in the spline space built over `knots`, and thus the `Splinets`-object in the output representing the projection spline satisfies `bdsp$bs@knots = knots`. Namely, if $S_j$ is the input `Splinets`-object, then the output is denoted as $\mathbf{P}S_j$, where $\mathbf{P}$ is the orthogonal projection to the space spanned by the spline basis constructed from the second set of knots:

$$\mathbf{P}S_j = \sum_{i=1}^{n-k+1} a_{ji} B_i, \quad (S_j - \mathbf{P}S_j) \perp \mathbf{P}S_j, \tag{2}$$

where $B_i$ are elements of the specified spline basis, and `knots` can differ from `sp@knots`.

This is an extension of the previous case since the output functions may belong to a different space than the input functions. The output is obtained by embedding both the input splines and the projection space to the space of splines that contains both built over the union of the two sets of knots.

*Projecting discretized functional data.* The function `project()` works also when the input is not a spline object but discretized functional data. This is the most important projection for FDA. In this case, the input is a matrix having in the first column a set of arguments and in the remaining ones the corresponding values of a sample of functional data. The input data are treated as piecewise constant functions with the value over two subsequent arguments equal to the value in the input corresponding to the left-hand side argument. In this way, the discretized data are viewed as functions, and their inner products with any spline are well defined. Consequently, the projection $\mathbf{P}S_j$ of the functional data in $S_j$ satisfies (2), if $S_j$ represents the discretized datum as a piece-wise constant function.

### 3.3 Spectral decomposition of the wine dataset – an example of FDA

We use the `Splinets` R package to perform Functional Principal Component Analysis (FPCA) on a wine dataset that includes 124 samples, with absorbance measurements taken across 256 wave numbers in the mid-infrared spectrum (4000 to 400 $\text{cm}^{-1}$). These spectral characteristics capture key aspects of the chemical composition of the wines, such as alcohol content and sugar content.

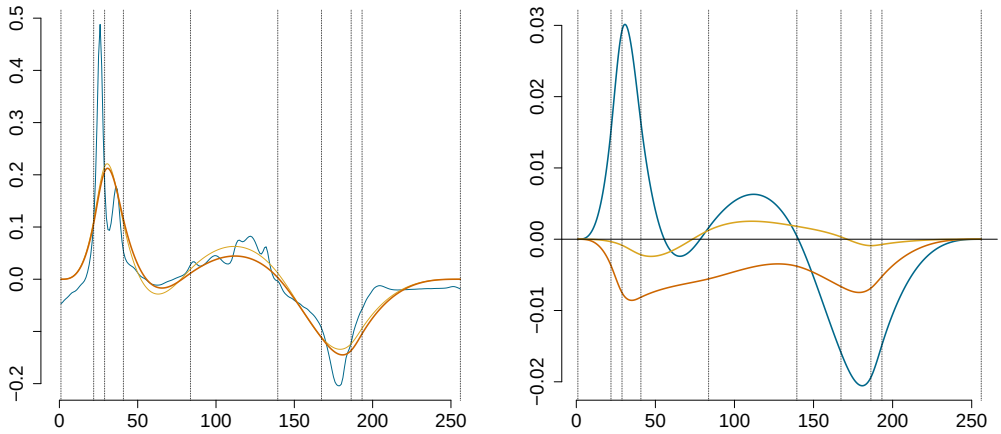



**Figure 5:** *Left:* One sample of the data (blue curve), the data projected onto splinets (yellow curve) built over a set of selected knots, with the location of the knots indicated by vertical dashed lines, and the data decomposed using only the first eigenfunction (orange curve); *Right:* the first three eigenfunctions scaled by the square roots of their corresponding eigenvalues.

The wine dataset is provided by Professor Marc Meurens of the Université Catholique de Louvain (Benoudjit et al., 2004). To initiate the analysis, we project the discrete data onto orthogonal spline bases using the `project()` function. This step transforms the discrete absorbance values into a functional representation utilizing a selected set of knots.

```
WineProj <- project(WineData, knots) # Project wine data into the spline space.
```

It is assumed here that the `WineData` data object is processed. The preprocessing steps of the data can be found at the *wine data example package vignette page* https://ranibasna.github.io/R-Splinets/articles/FunctionalPrincipalValueDecomposition.html.

The use of an orthogonalized functional basis provides significant computational advantages, reducing the complexity of performing FPCA on the functional data. By projecting the data onto an orthogonal basis, we benefit from simplifying the process of finding the functional principal components. This is because, in an orthogonal basis, the functional principal components can be efficiently derived by performing PCA on the projection coefficients.

Without orthogonality of the basis (like in the B-spline case), doing so would not be correct. The functional principal components can be computed as a linear combination of the eigenvectors and the spline basis.

```
Sigma <- cov(WineProj\$coeff)    # Covariance matrix of the projection coefficients
Spect <- eigen(Sigma)            # Spectral decomposition of the covariance
EigenSp <- lincomb(WineProj\$basis, t(Spect\$vec))  # Functional eigenfunctions
```

The extracted principal components enable the identification of significant spectral differences between the samples, offering a more interpretable and efficient representation. The `plot()` function is then used to visualize both the original and projected data, as well as the principal components. The detailed plotting code can be found at wine data example package vignette page https://ranibasna.github.io/R-Splinets/articles/FunctionalPrincipalValueDecomposition.html.

This approach, leveraging the flexibility and efficiency of the `Splinets` package, provides a robust framework for analyzing functional wine spectra, ensuring that essential spectral features are preserved without overfitting. Figure 5 (*Left*) shows a sample from the dataset, comparing the original wine data (blue curve), the data projected onto splinets over the selected knots (yellow curve), and the data decomposed and reconstructed using only the first eigenfunction (orange curve). The vertical dashed lines indicate the locations of the selected knots. Figure 5 (*Right*) shows the first three eigenfunctions scaled by the square roots of their respective eigenvalues.

## 4 Classification problem - a case study using Splinets

The package **Splinets** provides a comprehensive toolbox to analyze functional data. However, for a specific goal, one has to form a workflow that assures all steps have been performed to form the final

conclusions following the analysis. Here, we introduce a workflow for the classification problems. The focus here is on functional data that have a one-dimensional domain. In the future, we plan to extend the analytical tools to functional data with domains in higher dimensions such as images, movies, etc. For this reason, we have chosen a classic data set of images; however, our approach to their two-dimensional nature involves converting them into functions of one argument; for a more detailed discussion of the 2D extensions see (Basna et al., 2024). Before introducing the actual workflow, we explain the fundamentals of our methodology, highlighting the innovations that are specific to our use of spline spaces.

### 4.1 Methodology

In our approach to the classification of functional data, we first search for a suitable functional space of splines that captures important features of the data while maintaining a low dimensionality. These two seemingly opposing goals—fidelity and parsimony—must be balanced to ensure both efficiency and precision.

The main idea is to place knots in locations where the reduction in total squared error of the data approximation by 0-`degree` splines is largest. One implementation of this idea is available in the **ddk** package, written in R and accessible on GitHub: `https://github.com/ranibasna/ddk`, as discussed in (Basna et al., 2022).

For classification, spline spaces are selected separately for each class. The training data are then projected into the respective spaces, and these projections are referred to as functional data points. Functional principal component analysis (FPCA) is performed on the functional data points for each class in the training set. Specifically, the means are calculated for each class, and the centered data is decomposed into eigenvalues and eigenfunctions, yielding a spectral decomposition for each class separately.

This process involves estimating the eigenvalues $\lambda_{i1} > \lambda_{i2} > \cdots > \lambda_{in_i}$ for the $i$th class, along with the corresponding set of eigenfunctions $e_{i1}, e_{i2}, \ldots, e_{in_i}$, where $n_i$ is the number of eigenfunctions for class $i$, and $i = 1, 2, \ldots, K$, with $K$ representing the number of classes. A validation method is used to determine the optimal number of eigenvalues and eigenfunctions. Theoretical validation of the spectral decomposition is through Karhune-Loève's representation of the functional data which states that a function in each class is sampled independently from the following model

$$X_i(t) = \mu_i(t) + \sum_{k=1}^{\infty} \sqrt{\lambda_{ki}} Z_{ki} e_{ki}(t),\ i = 1, \ldots, K, \tag{3}$$

where $Z_{ki}$'s, mean-zero variance-one variables that are uncorrelated within a class and independent between classes.

Subsequently, each original discretized data point, say $x_l$, from a test set is projected onto all $K$ spline spaces based on the original knot selection per class. Thus one obtains $K$ functional data points $f_{li}, i = 1, \ldots, K$. The core principle of classification is to determine how closely the $f_{li}$ test data point matches the projection to the eigenspaces for a given class. The mathematical techniques used in this process for image data include feature extraction, dimensionality reduction, and classification algorithms, which are used to obtain meaningful information from functional data points and classify them efficiently. Let us describe this classification procedure in some further detail.

The first step involves computing the mean for each of the ten classes using our training data sets. For a given class such a mean function of all elements in the training set belonging to the $i$th class is denoted by $\hat{\mu}_i$ and is viewed as an estimated value of $\mu_i$. It is obtained by projection of the averaged data per class into a space of splines that has been determined by the knot selection process specifically for the mean.

For each original data point $x_l$ in the test set, one evaluates its representations $f_{li}, i = 1, \ldots, K$ in the spline spaces corresponding to each class. Our objective is to assess the proximity to each of the $K$ classes by projecting $f_{li} - \hat{\mu}_i$ onto the eigenspaces (the space spanned by eigenfunctions) corresponding to that class. Thus, we obtain $K$ distinctive projections. We denote them by $\hat{f}_{l1}, \hat{f}_{l2}, \ldots, \hat{f}_{lK}$. Explicitly,

$$\hat{f}_{li} = \hat{\mu}_i + \sum_{j=1}^{n_i} \langle f_{li} - \hat{\mu}_i, \hat{e}_{ji} \rangle \hat{e}_{ji},\ i = 1, \ldots, K \tag{4}$$

where $\langle \cdot \rangle$ stands for the inner product in the functional spaces, i.e. integral over the product of two functions, and $\hat{e}_{ij}$ are estimates of eigenfunctions $e_{ij}$ obtained in the training phase and discussed above. Formally, we obtain the following spectral decompositions

$$f_{li} = \hat{f}_{li} + \hat{\varepsilon}_{li},$$

where $\hat{\varepsilon}_{li}$ is the residual of the projection. If the functional point $x_l$ belongs to the $i$th class and $\hat{\mu}_i$ and $\hat{e}_{ij}$ are approximately equal to the true values, we would have based on (3):

$$\hat{\varepsilon}_{li} \approx \sum_{j=n_i+1}^{\infty} \sqrt{\lambda_{ki}} Z_{ki} e_{ji}(t),$$

which should be rather small in the squared norm $\|\cdot\|$ of the functional spaces. Indeed, we have

$$\|\hat{\varepsilon}_{li}\|^2 \approx \left\| \sum_{j=n_i+1}^{\infty} \sqrt{\lambda_{ki}} Z_{ki} e_{ji} \right\|^2 = \sum_{j=n_i+1}^{\infty} \lambda_{ki} Z_{ki}^2$$

and thus

$$\mathbb{E}(\|\hat{\varepsilon}_{li}\|^2) \approx \sum_{j=n_i+1}^{\infty} \lambda_{ki},$$

which can be made small as long as the selection of $n_i$ targets values so that $\sum_{j=1}^{\infty} \lambda_{ji} \approx \sum_{j=1}^{n_i} \lambda_{ji}$.

On the other hand, if $x_l$ does not belong to the $i$th class, the residual $\hat{\varepsilon}_{li}$ should be large, given that the classes are sufficiently distinguished by projections to the eigenspaces of their largest eigenvectors. This justifies the following classification rule

$$I(x_l) = \underset{i=1,\ldots,K}{\arg\min} \|\hat{\varepsilon}_{li}\| = \underset{i=1,\ldots,K}{\arg\min} \|x_l - \hat{f}_{li}\|, \tag{5}$$

where $I(x_l)$ stands for the chosen class and $x_l$ is treated as a piecewise constant function. One can enhance the outcome of classification by providing the squared normalized distances

$$\left(w_1^l, \ldots, w_K^l\right) = \frac{\left(\|x_l - \hat{f}_{l1}\|^2, \ldots, \|x_l - \hat{f}_{lK}\|^2\right)}{\sum_{i=1}^{K} \|x_l - \hat{f}_{li}\|^2}. \tag{6}$$

*Remark 1.* This rule may be improved to account for the classes that differ not by eigenvectors but by the corresponding eigenvalues. Namely, one could consider the path of residuals over an increasing number of eigenvectors and consider the sizes of the residuals along this path. We do not investigate this enhancement of the proposed classification.

Having outlined the classification procedure, the pivotal role of selecting the appropriate eigenvalues/eigenfunctions for each class becomes evident. These hyper-parameters have a great impact on our classification results. To enhance our classification's precision, we ascertain the ideal number of eigenvalues or eigenfunctions. Using the validation set technique, our primary objective is to identify the optimal count of eigenvalues/eigenfunctions.

### 4.2 Workflow

We describe, step-by-step, the workflow for the above-described methodology and, along with this, we perform these steps on the well-known Fashion MNIST dataset. The data set consists of a training set of 60,000 examples and a test set of 10,000 examples of Zalando's article images. The test dataset is divided into two distinct subsets: the validation dataset and the test dataset. Each data entry is represented by a grayscale image of $28 \times 28$ pixels. The problem can be viewed as a classification problem on the 784-dimensional Euclidean space. Items belong to one of 10 types of clothing, such as shoes, dresses, etc. The name of each class and its corresponding label are: *0 – T-shirt/top, 1 – Trouser, 2 – Pullover, 3 – Dress, 4 – Coat, 5 – Sandal, 6 – Shirt, 7 – Sneaker, 8 – Bag, 9 – Ankle boot*.

1. **Data preparation:** In this segment of the workflow, the package **Splinets** is not used but the way we represent and preprocess our data plays a critical role in subsequent analyses and outcomes. The remaining steps of the workflow assume that the data represent discretized functions, i.e. are matrices of columns representing arguments and values of the functional data. Two variants of this format are allowed: one with the common arguments for all data points, where the vector of arguments stands as the last column; the second one allows different arguments for different data points, in which the input should be a list of two-column matrices, with the vector of the arguments as the first column and the vector of the corresponding values as the second one. We note that in the second case, it is allowed to have a varying number of rows in the elements of the list.

   Once the data are properly formatted, it should be divided into three parts. The first and largest part (typically at least 50% of the data) constitutes the training data, and the remaining are split into two, the validation and testing. Alternatively, one can perform training and cross-validation on a data set of size, for example, 75%, and test the approach on the remaining testing portion of the data.

   **Example:** (Fashion MNIST dataset). For the Fashion MNIST dataset, we deal with two-dimensional

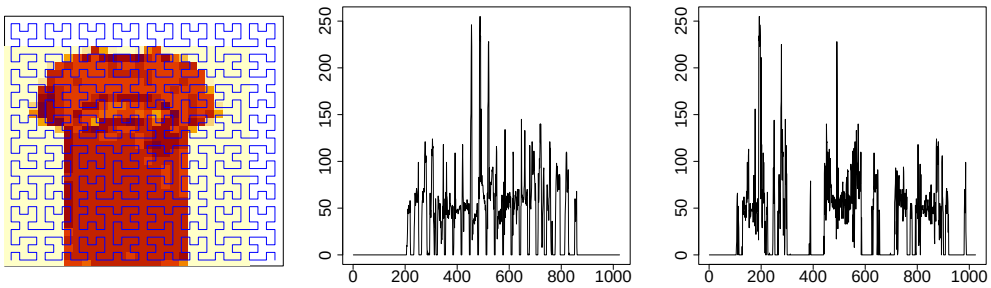



**Figure 6:** One- and two-dimensional representation of a T-shirt . *Left:* original image with the Hilbert curve laid over; *Middle:* column-major order representation; *Right:* Hilbert curve representation.

images, and transforming the data to one dimension is a critical process in which information will be lost. One of the tasks, we settled with the analysis of this data set is to compare different approaches to this transformation process. For this reason, we elucidate our approaches to data representation and transformation.

Images are typically represented as matrices of pixels, where the value of each matrix element indicates the color and intensity of the respective part of the image. The matrices need to be transformed into vector form. A prevailing approach for such a transformation involves stacking the image's columns (or rows) consecutively to create a vectorized representation. However, this technique often neglects the local spatial correlations existing between image pixels, ensuring only the preservation of vertical (or horizontal) correlations.

To more effectively retain these local correlations, we turn to the Hilbert curve transformation. A Hilbert curve is a continuous fractal space-filling curve that traverses every point in a square grid sized to any 2-power magnitude. For a detailed discussion of Hilbert curves, we refer to (Bader, 2012). Hilbert curves are particularly interesting for their ability to group pixels locally, see also R-package **gghilbertstrings** on CRAN (Valdez, 2021). The additional materials accompanying the paper show the explicit definition of the Hilbert curve used for our data obtained from the R-code of the Hilbert space-filling algorithm. This specific case of the Hilbert curve is shown in Figure 6 *(Left)* laid over the picture of a shirt.

Given that our image dimensions do not align with a 2-power magnitude, we have implemented zero padding, adjusting the image to a $2^5 \times 2^5$ pixel matrices. Figure 6 presents two illustrative examples from the MNIST dataset: a shirt and a boot. The middle illustrations provide vectorized representations of these items, derived from consecutively stacking image columns. The right-hand-side depictions visualize vectors formed through the Hilbert space-filling curve. It is easy to notice that the locality-preserving properties of Hilbert's curve make the vectorization based on it a better choice in comparison to the column-wise vectorization. We also see that the Hilbert curve vectorization lead to more sparse data which could be accounted for using disjoint-interval domains featured by the package, which would lead to more efficient computations. However, this aspect of the data has not been implemented as we did not face any major computational burden in our studies.

2. **Projection into spline spaces built on selected knots:** In this step, we consider $K$ distinct third-degree data-driven splinet bases, each built on knots specifically chosen for the corresponding class among the $K$ available. The number of knots for each class, as well as their placement, is determined using the DDK algorithm to effectively capture changes in the data. We then project the discrete training data points onto these splinets, constructed based on the class-specific knots. We considered separated spaces for the mean discrete data and the centered discrete data. We note that since the knots for the mean data are a subset of the knots for the centered data, the projections of the means are in the space of the projections of the centered data. However, they are not the same as their projections of the centered data space.

   These projections establish an isomorphism between the Euclidean vectors made of coefficients of the projection and the space of splines into which the original data have been projected. More specifically, if $N_i$ is the number of knots chosen of the $i$th class and $k$ is the degree of the splines, then the splines spanned on these knots constitute a $N_i - k - 1$-dimensional Hilbert space with $N_i - k - 1$ elements of the corresponding splinet so that

$$x_l \overset{proj}{\longmapsto} f_{li} \overset{isom}{\longleftrightarrow} \mathbf{a}_{li} \in \mathbb{R}^{N_i-k-1}, \tag{7}$$

   where $x_l$ is either the original centered discrete data point, $f_{li}$ is the corresponding functional

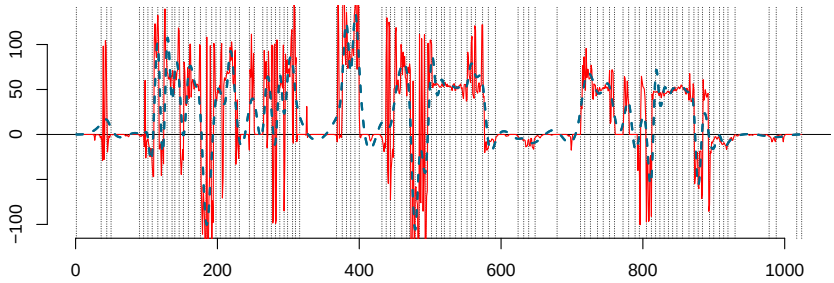



**Figure 7:** Fitting the centered 'T-shirt' by a spline with selected knots marked by vertical dotted lines.

data point when projected on the knots selected for the $i$th class, $\mathbf{a}_{li}$ is the vector of coefficients corresponding to the splinet expansion of $f_{li}$. This isomorphism allows us to perform the next part of the analysis simply by doing standard multivariate analysis on $\mathbf{a}_{li}$'s. The projection process is facilitated using the `project()` functions in the R-**Splinets** package. Other functions of the package assist exploration and visualization of various features of $f_{li}$'s.

**Example:** (Fashion MNIST dataset, cont.). The original images that were transformed through the Hilbert curve transform to the 1D discrete data are next projected to spline spaces, where the number of knots for the respective classes are

$$(N_1, N_2, N_3, N_4, N_5, N_6, N_7, N_8, N_9, N_{10}) = (106, 78, 100, 86, 94, 100, 106, 78, 128, 102).$$

Considering the number of knots and since we consider the third degree splines the dimension of the spline spaces are to be computed and put here according to the formula $(N_i - 4)$. We note the initial dimension reduction from 784 to the reported values of the given knot numbers in the classes. Additional further dimension reduction will be achieved through FPCA later on in the workflow. We see the projection of the 'T-Shirt' mean and an example of a centered 'T-Shirt' in Figure 7 *(Top-Middle)*. The locations of knots for these two cases are shown by vertical dashed lines.

3. **FPCA on training data:** To facilitate the classification procedure in the next step but also to gain a deeper understanding of the complexity of the data, the FPCA is performed within each class of the functional data points of the training set. In this task, one utilizes the isometry given in (7), and first the sample covariance matrix $\mathbf{\Sigma}_i$ is evaluated

$$\mathbf{\Sigma}_i = \overline{\mathbf{a}_{\cdot i}\mathbf{a}_{\cdot i}^\top} - \overline{\mathbf{a}_{\cdot i}}\,\overline{\mathbf{a}_{\cdot i}}^\top,$$

where averaging $\bar{\cdot}$ is made within the class $i$ of the training data. One simply applies `cov()` on the matrix of column vectors $\mathbf{a}_{li}$'s. Then the spectral decomposition of $\mathbf{\Sigma}_i$ into eigenvalues and eigenvectors is performed using `eigen()` of the R - *base* package. The obtained eigenvectors correspond to eigenfunctions through the isometry.

**Example:** (Fashion MNIST dataset, cont.). This rather standard step when performed on cloth image classes is illustrated in Figure 8. The first three eigenfunctions of 'T-shirts' scaled by the square roots of the respective eigenvalues are shown in the *(top-left)* graph and projections to the spaces based on 3 and 20 eigenfunctions are presented in the remaining graphs including the projection of 'Boots' to the 'T-Shirt' space, the *(bottom-right)* graph.

4. **Determining the significant eigenfunctions:** In this step, the classification procedure (5) as a function of the number of considered eigenfunctions is implemented. Then the number $n_i$ of the eigenfunctions for the $i$th class is based on its accuracy on the validation data set. In Figure 8 *(Bottom)*, we see the illustration of the classification principle. There are shown approximation of two data points, a 'T-Shirt' *(Left)*, and a 'Boot' *(Right)*, based on the projection to 20 eigenvalues in the 'T-Shirt spline space. It can be seen clearly that the approximation of 'T-shirt' is better than the approximation of 'Boot'. The functional $L_2$-norm of the functional spaces can be utilized to measure the approximation.

The approach to data classification relies on projecting the data into a subspace defined by a set of numbers $n_i$, $i = 1, \ldots, K$, of eigenfunctions of the distinct classes. We can see from (4) and the classification rule (5) that if $n_i$'s are taken too big, then it may result in overfitting of the data and a smaller dimension reduction, on the other hand, if the values are too small the precision of distinguishing the features in the data of different classes may be not sufficient. In this framework, the numbers of eigenfunctions for individual classes are of paramount significance and are treated as hyper-parameters. The $n_i$'s, $i = 1, \ldots, K$ are chosen through the following cross-validation procedure. From the training data, from each class, we exclude at random 10%-data points, denoted by $\mathcal{C}_i$,

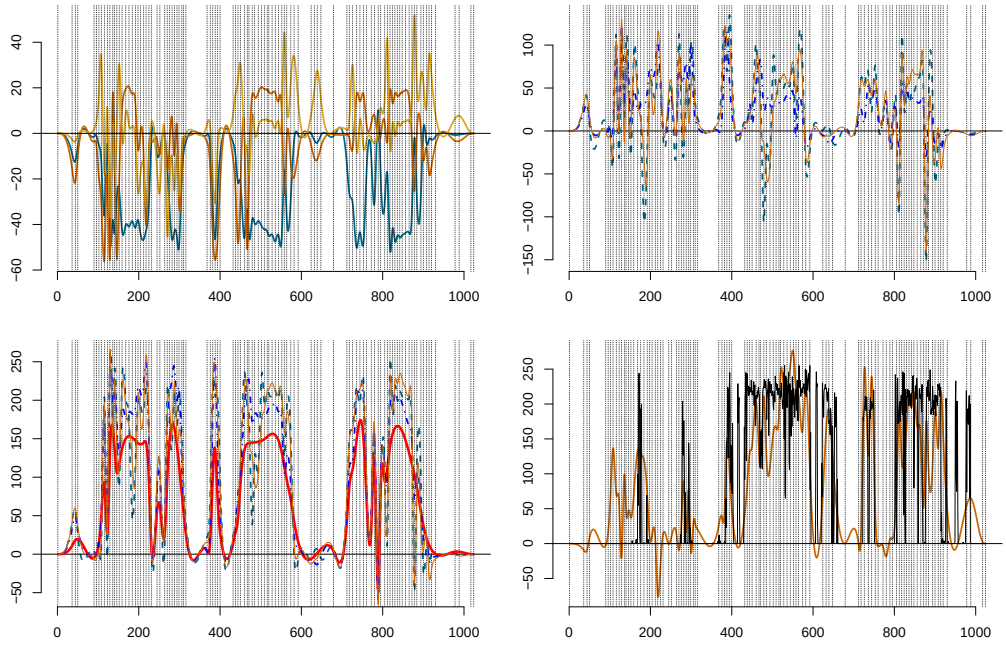


**Figure 8:** The spectral decomposition of the training data. *Top-Left:* The first three eigenfunctions for the 'T-shirt' class scaled by the square roots of the respective eigenvalues, *Top-Right:* Two approximations of the centered functional 'T-shirt' data point *(NavyBlue-Dashed Line)*: 1) by the first three eigenfunctions *(Blue-DottedDashed Line)* ; 2) by the first twenty eigenfunctions *(Orange-ThinSolid Line)*, *Bottom-Left:* The same approximation but centered around the 'T-Shirt' class mean $\hat{\mu}_1$ *(Red-ThickSolid Line)*, *Bottom-Right:* Projection of a 'Boot' data point to the 'T-Shirt' spectrum with 20 eigenfunctions after centering around the 'T-Shirt' class mean, the 'Boot' data point *(Black-Rough Line)* vs. its projection *(Orange-Solid Line)*.

$i = 1, \ldots, K$. Consider the remaining 90% for the $i$th class and build spectral decomposition of the data as explained in the previous step. Set the initial vector of numbers of the eigenvalues $\mathbf{n}^0 = (n_1^0, \ldots, n_{10}^0)$ in each class. For example, we can set it to zero to start with the classification based only on the mean $\hat{\mu}_i$, i.e. the closest mean decides for the class to which a data point belongs to. Let the classification distances (evaluated through (6)) for a given discrete data point $x$ be denoted by

$$\mathbf{w}(x, \mathbf{n}^0) = \mathbf{w}(x; n_1^0, \ldots, n_{10}^0).$$

Evaluate classification success rate through

$$\mathbf{s}(\mathbf{n}^0) = \left( \frac{\sum_{x \in \mathcal{C}_1} w_1(x, \mathbf{n}^0)}{|\mathcal{C}_1|}, \ldots, \frac{\sum_{x \in \mathcal{C}_K} w_K(x, \mathbf{n}^0)}{|\mathcal{C}_K|} \right).$$

where $|\mathcal{C}_i|$ is the number of elements in $\mathcal{C}_i$, or through the classification accuracy rate

$$\mathbf{a}(\mathbf{n}^0) = \left( \frac{\#\{x \in \mathcal{C}_1; I(x, \mathbf{n}^0) = 1\}}{|\mathcal{C}_1|}, \ldots, \frac{\#\{x \in \mathcal{C}_K; I(x, \mathbf{n}^0) = K\}}{|\mathcal{C}_K|} \right).$$

Small values in $\mathbf{s}$ or large values in $\mathbf{a}$ indicate good classification. Thus, taking averages of these two vectors can be used to assess the overall quality of a classification rule. Since the accuracy $\mathbf{a}$ is more interpretable, we focus on it and denote the average of its entries by $\bar{\mathbf{a}}$.

In general, the function $\mathbf{n} \mapsto \bar{\mathbf{a}}(\mathbf{n})$ is a non-convex function of $K$ arguments, and our goal is to find its optimum when evaluated over the validation data set. We adopt a simple iterative marginal gradient method, where we increase by one that coordinates in $\mathbf{n}$ that produces the large increase in $\bar{\mathbf{a}}$. Allow for a specific number of negative increases $L$ and conclude with the location $\mathbf{n}_{opt}$ of the maximum over-searched path. The number $L$ is a hyperparameter and is data-specific. The algorithm is repeated by a number of random initial knot distributions and the final result is the maximum over these runs.

**Example:** (Fashion MNIST dataset, cont.). In Figure 9 *(Top)*, the different trajectories of the accuracy obtained for randomly picked $\mathbf{n}_0$ are shown in thin-lines. The procedure was run on the validation set of the functional using the classification procedure (5) (also used in the next step of the workflow). The baseline for the performance is made of the rates of correct classification per class when it is

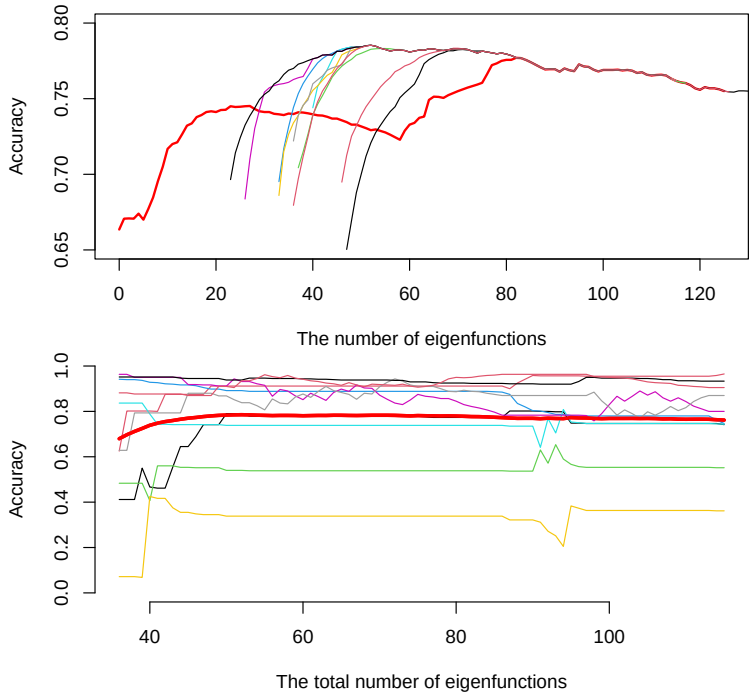



**Figure 9:** Optimization of the accuracy function in the validation phase (Step 4). *Top:* The trajectory of the average accuracy along the initial optimization path (thick-line) together with subsequently chosen random samples of the initial $\mathbf{n}_0$ that are shown in thin-lines; *Bottom:* The class-wise accuracies (thin-lines) against the average accuracy (thick-line). 'Pullovers' (green line) and 'Shirts' (yellow line) are notoriously difficult to classify.

only based on the distance of a data point from the spline projection of the mean, i.e. $\|x - \hat{\mu}_i\|$, so no eigenfunctions are considered. The obtained rates of correct classification rates per class are

$$(0.67, 0.88, 0.33, 0.73, 0.66, 0.75, 0.23, 0.80, 0.76, 0.86)$$

with the overall average performance of 66% of correct classification. From this, one sees that 'Pullover' and 'Shirt' seem to be difficult to classify. Overall accuracy is assessed using the approach as described above with random starting vector $\mathbf{n}_0$ of the initial allocation of the numbers of the eigenvalues in the class and searching for the optimal allocation. The procedure is illustrated in Figure 9 *(Top)*, where several subsequent trajectories of the total accuracy are presented by the thin-lines against the initial selection, the thick-line. In Figure 9 *(Bottom)*, the trajectories of the classwise accuracies and the average accuracies based on the above optimization procedure are presented. They lead to the following numbers of significant eigenvectors $(8, 4, 5, 6, 4, 8, 2, 4, 5, 6)$ for the ten cloth classes and, with this choice, the average accuracy in the validation process is 78.5%. We conclude that the original dimension of 784 vectors has been reduced more than tenfold (based on the total number 52 of all eigenfunctions). However, we also see that 'Pullovers' (y and 'Shirt' remain hard to classify, with the latter being properly classified less than 40% of the time. The final classification based on the obtained sizes of the functional spaces is performed on the testing data set in the next step.

5. **Testing the classification procedure:** This step is essentially repeating on the testing data set, most of the workflow except the validation steps used for the knot selection and deciding for the number of eigenfunctions. Thus for each data point $x_l$ in the testing sets, one projects it to ten functional spaces obtaining splines $f_{li}$, $i = 1, \ldots, K$. Those in turn are used to evaluate the classification rule (5) and decide on the class of the object. In fact, in the algorithm, the following convenient representation of the squared distance between a discrete data point $x_l$ and its approximation $\hat{f}_{li}$ given in (4) is used

$$\|x_l - \hat{f}_{li}\|^2 = \|x_l\|^2 - \|f_{li}\|^2 + \|f_{li} - \hat{\mu}_i\|^2 - \sum_{j=1}^{n_i} \left| \langle f_{li} - \hat{\mu}_i, \hat{e}_{ji} \rangle \right|^2 .$$

The results are compared with actual class memberships and summarized in the confusion matrix and other standard measurements of efficiency.

**Example:** (Fashion MNIST dataset, cont.). The leading example illustrating the workflow is using

Hilbert curve approach to transform from two-dimensional images to one-dimensional discrete data points. The classification method (5) with optimally chosen numbers of eigenvectors was performed through all testing data points and several characteristics have been evaluated. The average accuracy (over all classes) is 77.0%, which is close to the one in the cross-validation step and the classwise accuracies are

$$(73.4\%, 89.1\%, 52.3\%, 87.7\%, 72.0\%, 91.7\%, 32.9\%, 82.2\%, 94.5\%, 92.9\%),$$

which are also consistent with the results in the cross-validation experiment seen in Figure 9 *(Bottom)*. Another important summary is to report the averaged normalized distances $\bar{\mathbf{w}}$, of the elements of the class to their projection to that class and they are

$$(0.049, 0.049, 0.053, 0.046, 0.047, 0.051, 0.060, 0.033, 0.042, 0.030).$$

We observe expected negative correlations with the accuracies. The values are significantly lower from 0.1 (the case that the distance does not differentiate between classes) and are a major improvement over the original relative distances to the spaces.

6. **Final evaluation and conclusions:** This part depends on the goal for which the classification procedure has been used and thus is case-specific. One may need simply a classification method and its accuracy, then little is needed beyond the details shared in the previous step. It is also recommended to examine the statistics of misclassified data points and the features of these points as expressed by FPCA. This may give insight into the reasons for failing to identify data points properly and, in consequence, give an idea of how the classification can be further improved. If classification needs to be applied to some data without labels, then the classification should be run on it and summarized. This workflow can also serve as a benchmark to compare various classification methods. Then a comparison and a discussion should be carried out through reporting confusion matrices, along with other pertinent evaluation metrics. Again checking which data points have been misclassified can be important to see if a hybrid classification method could further improve the success rate.
   **Example:** (Fashion MNIST dataset, cont.). In our illustrative example, the goal was to show the workflow and the corresponding methodology. Since we focused on a single method, it is natural to present here more detailed characteristics of the method and comment on the obtained outcomes. It is clear that the method performs poorly in classifying 'Shirts' and, somewhat better, 'Pullovers'. We observe that a 'Shirt' is often classified as a 'T-Shirt' and a 'Coat'. Additionally, a 'Pullover' is often classified as a 'Coat'. Generally, it seems that the main responsibilities for the misclassifications are the classes: 'T-shirt', 'Pullover', 'Coat', and 'Shirt'.
   To improve on the method one could try to use a lower degree of splines that could help due to the noisiness of the Hilbert curve data points. By just looking at the data, it appears that the first-degree splines should suffice. Improving the search for an optimal number of eigenfunctions could be another possible source of improvement. We have seen that the performance from the validation step carries over to the testing results. Thus, enhancing the validation process should lead to more accurate classification results. Nevertheless, it seems that the achieved performance will be hard to significantly improve unless the full 2D character of the data is considered. This is planned in a future 2D extension of the spline-based method for the FDA.

**Table 1:** Confusion matrix

| | | | | | TARGET | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **PREDICT.** | T-shirt | Trouser | Pullover | Dress | Coat | Sandal | Shirt | Sneaker | Bag | Boot |
| T-shirt | 73.4% | 0.8% | 4.5% | 4.6% | 0.8% | 0.0% | 25.5% | 0.0% | 2.2% | 0.1% |
| Trouser | 0.1 | 89.1% | 0.3% | 1.7% | 0.1% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| Pullover | 1.8% | 0.5% | 52.3% | 1.5% | 10.4% | 0.0% | 8.6% | 0.0% | 0.7% | 0.0% |
| Dress | 12.1% | 8.7% | 2.0% | 87.7% | 8.4% | 0.0% | 8.9% | 0.0% | 1.2% | 0.0% |
| Coat | 0.3% | 0.6% | 20.2% | 1.7% | 72.0% | 0.0% | 17.9% | 0.0% | 0.2% | 0.0% |
| Sandal | 0.7% | 0.0% | 0.0% | 0.1% | 0.0% | 91.7% | 0.0% | 9.2% | 1.0% | 3.3% |
| Shirt | 5.0% | 0.1% | 17.5% | 1.8% | 6.8% | 0.0% | 32.9% | 0.0% | 0.0% | 0.0% |
| Sneaker | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 4.7% | 0.0% | 82.2% | 0.1% | 3.7% |
| Bag | 6.6% | 0.2% | 3.0% | 0.8% | 1.4% | 0.4% | 6.2% | 0.0% | 94.5% | 0.0% |
| Boot | 0.0% | 0.0% | 0.2% | 0.1% | 0.1% | 3.2% | 0.0% | 8.6% | 0.1% | 92.9% |

### 4.3 Comparing different 1D-methods functional methods

We employ the proposed workflow to investigate how different methods of data preparation influence the efficiency of our classification method. Our main goal is to assess the impact on the efficiency of

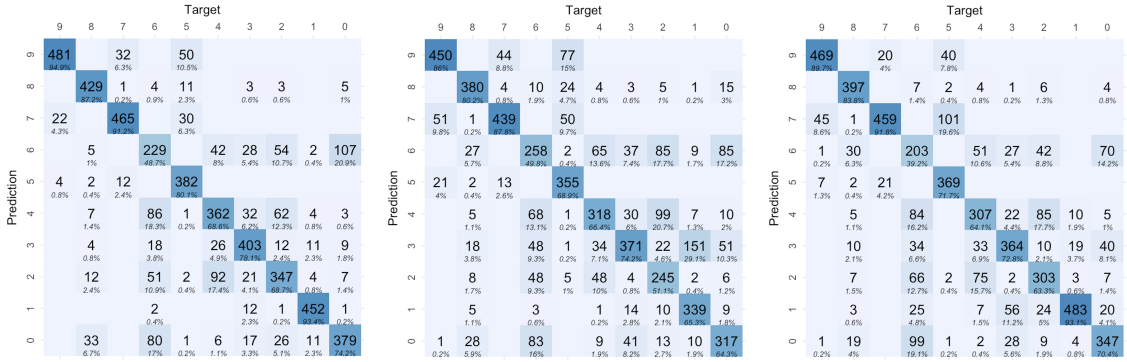



**Figure 10:** Confusion matrix for S1, S2, and S3 from right to left.

two factors: the Hilbert curve-based transformation method and the data targeted knots selection.

(that can be obtained, for example, by using DDK described in Basna et al. (2022)). We put forward three distinct scenarios to be examined using our established workflow:

S1: Hilbert curve transformation with 100 data-driven knot selections.

S2: By-row data with 100 data-driven knot selections.

S3: By-row data with 100 equidistant knots.

This choice ensures both the validity and fairness of our comparisons, thus enhancing the reliability and interpretability of our results.

The choice of these scenarios stems largely from our belief that the data targeted knot selection method leads to a more efficient dimension reduction, especially in the context of sparse data. As evidenced in Figure 6, the pixel distribution in the by-row data does not exhibit sparsity. This suggests that we might not see significant performance gains if we were to deploy the data based knot selection methodology on it. However, the introduction of the Hilbert curve transformation imparts a noticeable sparsity in the data's curvature.

As detailed above, we project the data into a subspace spanned by a number of eigenfunctions. For each scenario, we chose a number of eigenfunctions that achieved the highest accuracy on the validation data set. The validation phase of the analysis suggests that the optimal number of eigenvalues for S1, S2, and S3 are 15, 10, and 15, respectively. After the testing phase, we carry the classification problem described earlier across the three scenarios. To evaluate the performance of our classification model, we define the following metrics

$$\text{Accuracy} = \frac{T_P + T_N}{T_P + T_N + F_P + F_N}, \quad \text{Precision} = \frac{T_P}{T_P + F_P}, \quad \text{Recall} = \frac{T_P}{T_P + F_N}, \quad \text{F1} = 2\,\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},$$

where $T_P, T_N, F_P, F_N$ are true positives, true negatives, false positives and false negatives, respectively. Table 2 presents the outcomes obtained from analyzing the classification problem within the three suggested scenarios.

**Table 2:** Accuracy, Precision, Recall and F1 Score of the classification problem in each scenario.

| | **Scenario 1** | **Scenario 2** | **Scenario 3** |
|---|---|---|---|
| **Accuracy** | 78.6% | 69.44% | 73.6% |
| **Precision** | 79.22% | 71.3% | 74.1% |
| **Recall** | 78.5% | 69.4% | 73.9% |
| **F1** | 78.74% | 69.8% | 73.6% |

The results from the first scenario (S1) clearly demonstrate its superior performance, affirming our proposition. To further understand the algorithm's misclassifications, we provide the confusion matrix for each scenario in Figure 10. The confusion matrix illustrates the comparison between the true labels and the predicted labels of the test dataset. The confusion matrices show the uncertainty mostly between the classes 0,2,4,6 in the fashion image dataset, as discussed in the previous section . This makes sense because t-shirts, pullovers, coats, and shirts look similar and might be confusing.

In the absence of the Hilbert curve transformation, it is observed that the classification performance of the data-driven knot selection approach alone is the least accurate among the three scenarios. This is consistent with expectations given that the data exhibits a dense and non-sparse format with highly repetitive patterns. The equidistant knot selection classification strategy shows a modest increment

in the accuracy results. However, the application of data-driven knot selection on the transformed data significantly improves the accuracy of the results. It is important to note that our aim is not to outperform or compete with state-of-the-art machine learning and deep learning methodologies in image classification. Although our method may not achieve the the optimal accuracy levels that can be seen in the convolution neural network approach, it does reach a commendable accuracy rate of approximately 80%. This is comparable to simpler neural networks, such as decision tree classifiers and MLP classifiers. For further benchmarking of machine learning algorithm classifiers on the Fashion MNIST dataset, see (Xiao et al., 2017). The implementation of the Hilbert curve transformation highlights the effectiveness of the data-driven method for representing data in a functional format, resulting a noticeable increase in accuracy by $5-6\%$. This enhancement is attributed to the efficient representation of locality features in the data through the chosen basis, ensuring the preservation of space curvature.

There are two notable observations to consider. First, the data-driven knot selection method demonstrates superior performance compared to the traditional equidistance method, especially when handling sparse data that exhibits distinct spatial dependencies. This difference in performance can be observed by comparing the class-wise accuracy between S1, S2, and S3. Second, leveraging the 2D characteristics of the images has a positive impact on the performance of Functional Principal Component Analysis (FPCA) when combined with the data-driven knot selection method. A transformation such as the Hilbert curve, can reveal the two-dimensional characteristics of the original image by preserving the locality relationships among the pixels. More precisely, in the flattened one-dimensional vector, consecutive elements (pixels) were likely neighbors in the original two-dimensional space. It also creates a sparser representation of the image, which makes it more amenable to our method. Therefore, when we apply our FPCA data-driven analysis techniques to this one-dimensional vector, these techniques approximately analyze the local neighborhoods of the original two-dimensional image. This enhancement suggests, as expected, that considering the 2D structure of the images contributes to improved results and should be further investigated.

## 5 Summary

We have presented tools from the **Splinets**-package that facilitate an efficient analysis of functional data. The benefits using the orthogonal basis implemented in the package are illustrated both by a simple example of the FPCA performed on the classical wine data set and by a more advanced workflow for classification of functional data. This workflow is designed to enhance the efficiency of dimension reduction while retaining functional dependencies in the data's characterizing features, which may not necessarily be local. The workflow was benchmarked on the Fashion MNIST data set. The findings from this example emphasize the pivotal importance of the data's two-dimensional nature in the functional principal component analysis approach. The Hilbert curve proved to be the most efficient and it is partially due to its capacity of capturing dependencies beyond one dimension. These observations suggest that adopting the method further toward two-dimensional analysis, for example, by considering gradient images or, even better, two-dimensional FPCA may lead to significant gains in the efficiencies.

## References

M. Bader. *Space-filling curves: an introduction with applications in scientific computing*, volume 9. Springer Science & Business Media, 2012. [p54]

R. Basna, H. Nassar, and K. Podgórski. Data driven orthogonal basis selection for functional data analysis. *Journal of Multivariate Analysis*, 189:104868, 2022. [p46, 52, 59]

R. Basna, H. Nassar, and K. Podgórski. Spline-based methods for functional data on multivariate domains. *Journal of Mathematics in Industry*, 14(1):13, 2024. [p52]

R. Basna, X. Liu, H. Nassar, K. Podgórski, and H. Wu. Splinets: Functional data analysis using splines and orthogonal spline bases, 2025. URL https://CRAN.R-project.org/package=Splinets. R package version 1.5.1. [p42]

N. Benoudjit, E. Cools, M. Meurens, and M. Verleysen. Chemometric calibration of infrared spectrometers: selection and validation of variables by non-linear models. *Chemometrics and intelligent laboratory systems*, 70(1):47–53, 2004. [p51]

O. Cho and M. J. Lai. A class of compactly supported orthonormal B-spline wavelets. *Splines and Wavelets*, pages 123–151, 2005. [p42]

C. de Boor. A practical guide to splines. In *Applied Mathematical Sciences*, 1978. [p42]

T. N. Goodman. A class of orthogonal refinable functions and wavelets. *Constructive approximation*, 19 (4):525–540, 2003. [p42]

X. Liu, H. Nassar, and K. Podgórski. Dyadic diagonalization of positive definite band matrices and efficient B-spline orthogonalization. *Journal of Computational and Applied Mathematics*, 414:114444, 2022. [p42, 43, 45, 47]

J. Mason, G. Rodriguez, and S. Seatzu. Orthogonal splines based on B-splines – with applications to least squares, smoothing and regularisation problems. *Numerical Algorithms*, 5(1):25–40, 1993. [p42]

H. Nassar and K. Podgórski. Periodic splinets. *Communications in Statistics - Simulation and Computation*, pages 1–16, 2025. [p43]

T. Nguyen. Construction of spline type orthogonal scaling functions and wavelets. Honors Project Paper 19, Illinois Wesleyan University, 2015. [p42]

A. Perperoglou, W. Sauerbrei, M. Abrahamowicz, and M. Schmid. A review of spline function procedures in R. *BMC Medical Research Methodology*, 19(1):46, 2019. [p42]

E. F. Petricoin, A. M. Ardekani, B. A. Hitt, P. J. Levine, V. A. Fusaro, S. M. Steinberg, G. B. Mills, C. Simone, D. A. Fishman, E. C. Kohn, and L. A. Liotta. Use of proteomic patterns in serum to identify ovarian cancer. *Lancet*, 359(9306):572–577, Feb 2002. [p48]

K. Podgórski. Splinets – splines through the Taylor expansion, their support sets and orthogonal bases, 2021. URL https://arxiv.org/abs/2102.00733. [p44]

K. Qin. General matrix representations for B-splines. *Vis. Comput.*, 16:177–186, 2000. [p44]

A. Redd. A comment on the orthogonalization of B-spline basis functions and their derivatives. *Stat. Comput*, 22:251–257, 2012. [p44]

A. Redd. orthogonalsplinebasis: Orthogonal b-spline basis functions, 2022. URL https://CRAN.R-project.org/package=orthogonalsplinebasis. R package version 0.1.7. [p44]

L. Schumaker. *Spline functions: basic theory*. Cambridge University Press, 2007. [p42]

A. C. Valdez. gghilbertstrings: A fast 'ggplot2'-based implementation of hilbert curves, 2021. URL https://CRAN.R-project.org/package=gghilbertstrings. R package version 0.3.3. [p54]

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://arxiv.org/abs/1708.07747. [p60]

L. Zhou, J. Huang, and R. Carroll. Joint modeling of paired sparse functional data using principal components. *Biometrika*, 95:601–619, 2008. [p44]

*Rani Basna*
*Department of Clinical Sciences*
*Lund University*
*Sweden*
Rani.Basna@med.lu.se

*Hiba Nassar*
*Cognitive Systems, Department of Applied Mathematics and Computer Science*
*Technical University of Denmark*
*Denmark*
hibna@dtu.dk

*Krzysztof Podgórski*
*Department of Statistics*
*Lund University*
*Sweden*
Krzysztof.Podgorski@stat.lu.se

# Disaggregating Time-Series with Many Indicators: An Overview of the DisaggregateTS Package

*by Luke Mosley, Kaveh Salehzadeh Nobari, Giuseppe Brandi, and Alex Gibberd*

**Abstract** Low-frequency time-series (e.g., quarterly data) are often treated as benchmarks for interpolating to higher frequencies, since they generally exhibit greater precision and accuracy in contrast to their high-frequency counterparts (e.g., monthly data) reported by governmental bodies. An array of regression-based methods have been proposed in the literature which aim to estimate a target high-frequency series using higher frequency indicators. However, in the era of big data and with the prevalence of large volumes of administrative data-sources there is a need to extend traditional methods to work in high-dimensional settings, i.e., where the number of indicators is similar or larger than the number of low-frequency samples. The package DisaggregateTS includes both classical regressions-based disaggregation methods alongside recent extensions to high-dimensional settings. This paper provides guidance on how to implement these methods via the package in R, and demonstrates their use in an application to disaggregating CO2 emissions.

## 1 Introduction

Economic and administrative data, such as recorded surveys and consensus, are often disseminated by international governmental agencies at low or inconsistent frequencies, or irregularly-spaced intervals. To aid the forecasting of the evolution of the dynamics of these macroeconomic and socioeconomic indicators, as well as their comparison with higher resolution indicators provided by international agencies, statistical agencies rely on signal extraction, interpolation and temporal distribution adjustments of the low-frequency data to provide high precision and uninterrupted historical data. Although, temporal distribution, interpolation and benchmarking are closely associated with one another, this article and its respective package (**DisaggregateTS**, Mosley and S. Nobari, 2024), expend particular attention to interpolation and temporal distribution (disaggregation) techniques, where the latter is predicated on regression-based methods[1]. These regression-based temporal distribution techniques rely on high-frequency indicators to estimate (relatively) accurate high-frequency data points. With the prevalence of large volume of high-frequency administrative data, a great body of literature pertaining to statistical and machine learning methods has been dedicated to taking advantage of these additional resources for forecasting purposes (see Fuleky, 2019, for an overview of macroeconomic forecasting in the presence of big data). Additionally, one may wish to utilize these abundant indicators to generate high-frequency estimates of low-frequency time-series with greater precision. However, in high-dimensional linear regression models where the number of dimensions surpass that of the observations, consistent estimates of the parameters is not possible without imposing additional structure (see Wainwright, 2019). Hence, this article and the package **DisaggregateTS** adapt recent contributions in high-dimensional temporal disaggregation (see Mosley et al., 2022) to extend previous work within this domain (see the package **tempdisagg** Sax et al., 2023, and its corresponding article Sax and Steiner (2013)) to high-dimensional settings.

As noted by Dagum and Cholette (2006), time-series data reported by most governmental and administrative agencies tend to be of low-frequency and precise, but not particularly timely, whereas their high-frequency counterparts seldom uphold the same degree of precision.

The aim of temporal distribution techniques is to generate high-frequency estimates that can track shorter term movements, than directly observable with the direct low-frequency observations. While interpolation problems are generally encountered in the context of stock series, where say, the quarterly value of the low-frequency series must coincide with the value of third month of the high-frequency data (of the same quarter), temporal distribution problems often concern flow series, where instead the value of the low-frequency quarterly series must agree with the sum (or weighted combination) of the values of the high-frequency months in that quarter. The latter approach is generally accomplished by identifying and taking advantage of a number of high-frequency indicators which are deemed to behave in a similar manner to the low-frequency series, and by estimating the high-frequency series through a linear combination of such indicators.

In the last few decades, a significant number of articles have been published within this domain—

[1]See Dagum and Cholette (2006) for an overview of benchmarking, interpolation, temporal distribution and calendarization techniques.

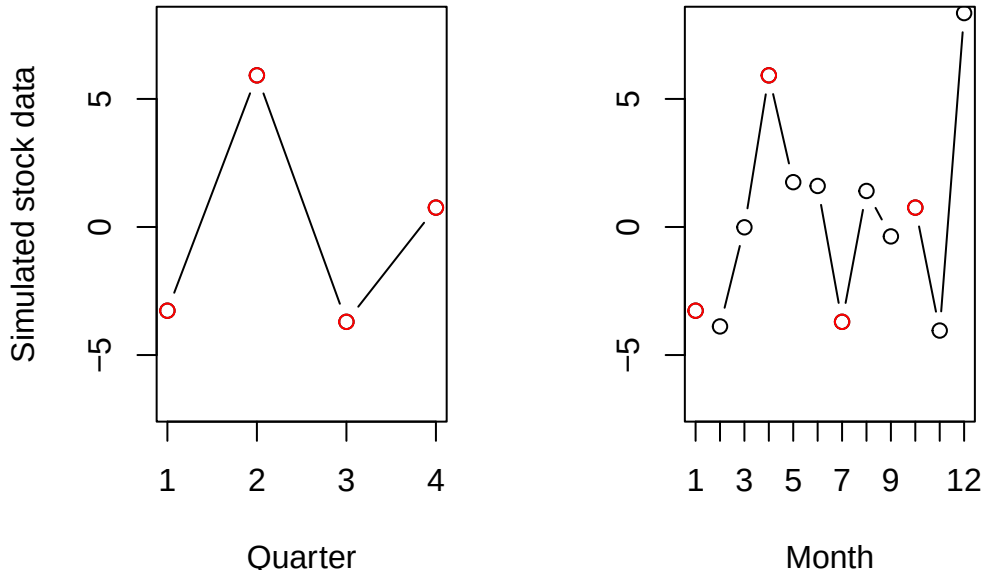



**Figure 1:** Quarterly and monthly simulated stock data

see Dagum and Cholette (2006) for a detailed review of these techniques. Notable studies within this context include the additive regression-based benchmarking methods of Denton (1971) and Chow and Lin (1971), Chow and Lin (1976), as well as those proposed by Fernández (1981) and Litterman (1983) in the presence of highly persistent error processes. More recently these methods have been extended to the high-dimensional setting by Mosley et al. (2022), where prior information on the structure of the linear regression model is used to enable estimation, and better condition the regression problem. Specifically, this is accomplished by "least absolute shrinkage and selection operator" (LASSO hereafter) proposed by Tibshirani (1996), which in principle selects an appropriate model by penalizing the coefficients (in scale) of the high-dimensional regression, in effect discarding the irrelevant indicators from the model. In what follows, we demonstrate how to apply these methods using **DisaggregateTS** to easily estimate high-frequency series of interest.

The remainder of the paper is organized as follows: Section 2 presents the methodologies behind key temporal disaggregation techniques included in the **DisaggregateTS** package, along with their extensions to high-dimensional settings. Section 3 introduces the **DisaggregateTS** package and highlights its key functions. Sections 4 and 5 provide examples based on simulations (using a function in the package that generates synthetic data) and empirical data to demonstrate the package's functionality. Finally, Section 6 concludes the paper.

## 2 Sparse temporal disaggregation

### 2.1 Classical regression-based techniques

The data in figures 1 and 2 are generated using the `TempDisaggDGP()` function from the **DisaggregateTS** package, representing simulated stock and flow data. For the simulated stock data (Figure 1, each quarter's low-frequency data should match the first month's value of the corresponding high-frequency series, denoted with red dots. For the simulated flow data (Figure 2, the quarterly figures should equal the sum of the sub-quarterly values

Suppose we observe a low-frequency series, say, quarterly GDP, encoded as the vector $\mathbf{y}_q \in \mathbb{R}^n$, containing $n$ quarterly observations. We desire to disaggregate this series to higher frequencies (say monthly), where the disaggregated series is denoted $\mathbf{y}_m \in \mathbb{R}^p$, with $p = 3n$. Furthermore, we wish that the disaggregated series be temporally consistent without exhibiting any jumps between quarters (see Section 3.4 of Dagum and Cholette, 2006, for examples of such inconsistencies between the periods). The challenge is to identify an approach that distributes the variation between each observed quarterly point to the monthly level. A method that has been extensively studied in the literature concerns finding high-frequency (e.g., monthly) indicator series that are thought to exhibit similar inter-quarterly movements as the low-frequency variable of interest. Let us denote a set of $p$ observations from these $d$ indicators as the matrix $\mathbf{X}_m \in \mathbb{R}^{p\times d}$. A classical approach to provide high-frequency estimates is the regression-based temporal disaggregation technique proposed by Chow and Lin (1971) whereby the unobserved monthly series $\mathbf{y}_m$ are assumed to follow the regression:

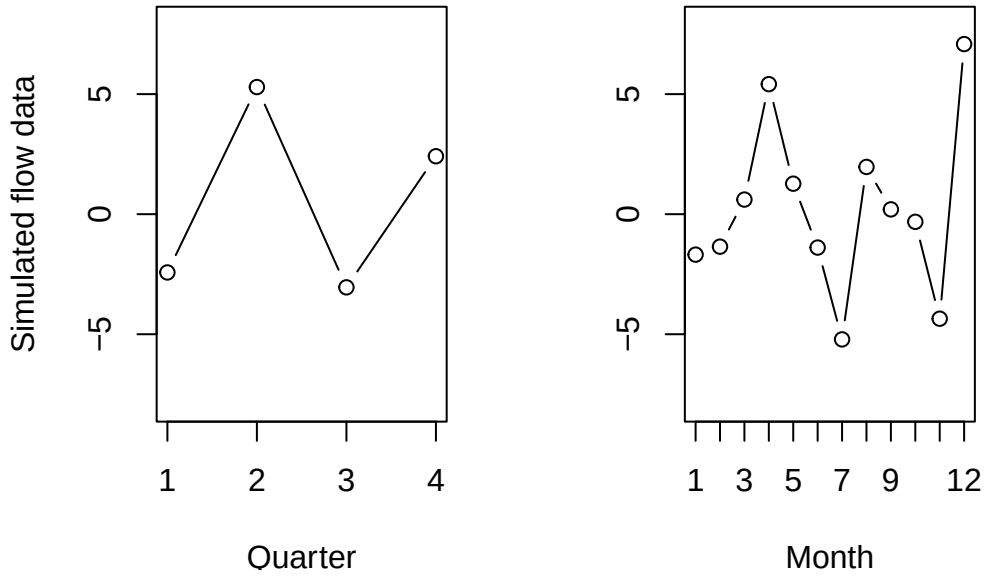



**Figure 2:** Quarterly and monthly simulated flow data

$$\mathbf{y}_m = \mathbf{X}_m\beta + \mathbf{u}_m, \quad \mathbf{u}_m \sim N(\mathbf{0}_m, \mathbf{V}_m) \tag{1}$$

where $\beta \in \mathbb{R}^d$ is a vector of regression coefficients to be estimated (noting that $\mathbf{X}_m$ may contain deterministic terms) and $\mathbf{u}_m \in \mathbb{R}^p$ is a vector of residuals. Chow and Lin (1971) assume that $\mathbf{u}_m$ follows as AR(1) process of the form $u_t = \rho u_{t-1} + \varepsilon_t$ with $\varepsilon_t \sim N(0, \sigma^2)$ and $|\rho| < 1$. The assumption of stationary residuals allows for a cointegrating relationship between $\mathbf{y}_m$ and $\mathbf{X}_m$ when they are integrated of the same order. Thus, the covariance matrix has a well-known Toeplitz structure as follows:

$$\mathbf{V}_m = \frac{\sigma^2}{1-\rho^2}\begin{pmatrix} 1 & \rho & \cdots & \rho^{p-1} \\ \rho & 1 & \cdots & \rho^{p-2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho^{p-1} & \rho^{p-2} & \cdots & 1 \end{pmatrix} \tag{2}$$

where $\rho$ and $\sigma$ are unknown parameters that need to be estimated. The dependent variable $\mathbf{y}_m$ in (1) is unobserved, hence the regression is premultiplied by the $n \times p$ aggregation matrix $\mathbf{C}$, where:

$$\begin{aligned} \mathbf{C} &= \mathbf{I}_n \otimes (1,1,1) \\ &= \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}_{n \times p} \end{aligned} \tag{3}$$

where $\otimes$ is the Kronecker operator, and the vector of ones in (3) is used for flow data (e.g., GDP), such that the sum of the monthly GDPs coincides with its quarterly counterpart[2]. The premultiplication yields the quarterly counterpart of (1):

$$\mathbf{C}\mathbf{y}_m = \mathbf{C}\mathbf{X}_m\beta + \mathbf{C}\mathbf{u}_m, \quad \mathbf{C}\mathbf{u}_m \sim N(\mathbf{C}\mathbf{0}_m, \mathbf{C}\mathbf{V}_m\mathbf{C}^\top). \tag{4}$$

The GLS estimator for $\beta$ is thus expressed as follows:

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^d} \left\{ \left| \mathbf{V}_q^{-\frac{1}{2}}(\mathbf{y}_q - \mathbf{X}_q\beta) \right|_2^2 \right\} \tag{5}$$

$$= \left(\mathbf{X}_q^\top \mathbf{V}_q^{-1}\mathbf{X}_q\right)^{-1} \mathbf{X}_q^\top \mathbf{V}_q^{-1}\mathbf{y}_q \tag{6}$$

where $\mathbf{X}_q = \mathbf{C}\mathbf{X}_m$, $\mathbf{y}_q = \mathbf{C}\mathbf{y}_m$ and $\mathbf{V}_q = \mathbf{C}\mathbf{V}_m\mathbf{C}^\top$. Note that estimating $\beta$ requires the knowledge of the unknown parameters $\sigma$ and $\rho$ in $\mathbf{V}_m$ which are unknown. We employ the profile-likelihood maximization technique of Bournay and Laroque (1979), which involves first estimating $\hat{\beta}$ and $\mathbf{V}_q$

[2]For alternative aggregations see Quilis (2018) and Sax et al. (2023). For instance, if quarterly values correspond to averages of monthly values, then the vector in equation (3) assumes the form $(0.33, 0.33, 0.33)$.

conditional on a given value of $\rho$, and maximizing the log-likelihood function by conducting a grid search over $\rho \in (-1, 1)$ for the autoregressive parameter. However, in practical applications, including well-known implementations such as the **tempdisagg** package and the `Gretl` econometric software Cottrell and Lucchetti (2023), the grid search is often restricted to $\rho \in [0, 1)$, reflecting a non-negative constraint on the autoregressive parameter. Our method is specifically designed to search within $\rho \in [0, 1)$, reflecting this non-negative constraint.

Chow and Lin (1971) show the optimal solution is obtained by:

$$\hat{\mathbf{y}}_m = \mathbf{X}_m\hat{\beta} + \hat{\mathbf{V}}_m\mathbf{C}\hat{\mathbf{V}}_q^{-1}\left(\mathbf{y}_q - \mathbf{X}_q\hat{\beta}\right), \tag{7}$$

where $\mathbf{X}_m\hat{\beta}$ is the conditional expectation of $\mathbf{y}_m$ given $\mathbf{X}_m$ and the estimate of the monthly residuals are obtained by disaggregating the quarterly residuals $\mathbf{y}_q - \mathbf{X}_q\hat{\beta}$ to attain temporal consistency between $\hat{\mathbf{y}}_m$ and $\mathbf{y}$.

Other variants of the regression-based techniques include those proposed by Denton (1971), Fernández (1981), Litterman (1983), and Cholette (1984), with the latter two addressing scenarios where $\mathbf{y}_m$ and $\mathbf{X}_m$ are not cointegrated. Although these traditional techniques are included in the **DisaggregateTS** package, a comprehensive overview of different temporal disaggregation techniques and distribution matrices can be found in Table 2 of Sax and Steiner (2013). The **tempdisagg** package implements several standard methods for temporal disaggregation, each with distinct approaches for estimating high-frequency series. These include the Denton, Denton-Cholette, Chow-Lin, fernández, and Litterman methods. As summarized in Table 2 of Sax and Steiner (2013), Denton and Denton-Cholette focus on movement preservation, while regression-based methods like Chow-Lin, fernández, and Litterman perform generalized least squares regressions on the low-frequency series using one or more high-frequency indicators. We have implemented the Chow-Lin method with a non-negative autoregressive parameter, constrained within the range $[0, 1)$, following the approach of Bournay and Laroque (1979) and Cottrell and Lucchetti (2023). This offers flexibility in different disaggregation scenarios.

### 2.2 Extension to high-dimensional settings

The shortcoming of Chow and Lin (1971) becomes evident in data-rich environments, where the number of indicators $d \gg n$ surpass that of the time-stamps for the low-frequency data. Let us once again recall the GLS estimator (6). When $d < n$ and the columns of $\mathbf{X}_q^\top\mathbf{V}_q^{-1}\mathbf{X}_q$ are independent, the estimator is well-defined. However, when $d > n$, the matrix is rank-deficient - i.e., $\text{rank}(\mathbf{X}_q^\top\mathbf{V}_q^{-1}\mathbf{X}_q) \leq \min(n, d)$, the matrix $\mathbf{X}_q^\top\mathbf{V}_q^{-1}\mathbf{X}_q$ has linearly dependent columns, and thus is not invertible. In moderate dimensions, where $d \approx n$, $\mathbf{X}_q^\top\mathbf{V}_q^{-1}\mathbf{X}_q$ has eigenvalues close to zero, leading to high variance estimates of $\hat{\beta}$.

Mosley et al. (2022) resolve this problem by adding a regularizing penalty (e.g., $\ell_1$ regularizer) onto the GLS cost function (5):

$$\hat{\beta}(\lambda_n \mid \rho) = \arg\min_{\beta\in\mathbb{R}^d}\left\{\left\|\mathbf{V}_q^{-\frac{1}{2}}(\mathbf{y}_q - \mathbf{X}_q\beta)\right\|_2^2 + \lambda_n\|\beta\|_1\right\}. \tag{8}$$

Unlike the GLS estimator (6), the regularized estimator corresponding to the cost function (8) is a function of $\lambda_n$ and the autoregressive parameter $\rho$. Henceforth, it is important to nominate the most suitable $\lambda_n$ and $\rho$ to correctly recover the parameters. In (8), we denote the estimator as $\hat{\beta}(\lambda_n \mid \rho)$ to highlight that the solution paths of the estimator for different values of $\lambda_n$, say, $\lambda_n^{(1)}, \lambda_n^{(2)}, \cdots, \lambda_n^{(k)}$ are generated for (i.e. conditional on) a fixed $\rho$. The solution paths are obtained using the LARS algorithm proposed by Efron et al. (2004), the benefits of which have been extensively discussed in Mosley et al. (2022).

LASSO estimators inherently exhibit a small bias, such that $\|\hat{\beta}\|_2^2 \leq \|\beta^*\|_2^2$, where $\beta^*$ denotes the true coefficient vector. To alleviate this issue, Mosley et al. (2022) further follow Belloni and Chernozhukov (2013), by performing a refitting procedure using least squares re-estimation. The latter entails generating a new $n \times d^{(l)}$ sub-matrix $\mathbf{X'}_q$, where $d^{(l)} \leq d$ from the original $n \times d$ matrix $\mathbf{X}_q$, with $\mathbf{X'}_q$ corresponding to the columns of $\mathbf{X}_q$ supported by $\hat{\beta}(\lambda_n^{(l)} \mid \rho)$, for solutions $l = 1, \cdots, k$ obtained from the LARS algorithm[3]. We then perform a usual least squares estimation on $(\mathbf{y}_q, \mathbf{X'}_q)$ to obtain debiased solution paths for each $\lambda_n^{(l)}$.

Finally, Mosley et al. (2022) choose the optimal estimate from $\hat{\beta}(\lambda_n^1 \mid \rho), \cdots, \hat{\beta}(\lambda_n^k \mid \rho)$ using the Bayesian Information Criterion (BIC hereafter) proposed by Schwarz (1978). The motivation for nomi-

[3]noting the LARS algorithm produces solutions evaluated at a series of $\{\lambda_l\}_{l=1}^k$ points.

nating this statistic over resampling methods, such as cross-validation or bootstrapping techniques, stems from the small sample size in the low-frequency observations. The optimal regularization is chosen conditional on $\rho$ according to

$$\hat{\lambda}_n(\rho) = \arg\min_{\lambda_n(\rho)\in\{\lambda_n^{(1)}(\rho),\cdots,\lambda_n^{(k)}(\rho)\}} \left\{-2\mathcal{L}\left(\hat{\beta}(\lambda_n \mid \rho), \hat{\sigma}^2\right) + \log(n) K_{\lambda_n(\rho)}\right\}, \tag{9}$$

where $K_{\lambda_n(\rho)} = |\{r : (\hat{\beta}(\lambda_n \mid \rho)_r \neq 0)\}|$ is the degrees of freedom and $\mathcal{L}(\hat{\beta}(\lambda_n \mid \rho), \hat{\sigma}^2)$ is the log-likelihood function of the GLS regression (6), which in the presence of Gaussian errors, is given by:

$$\mathcal{L}(\hat{\beta}, \hat{\sigma}^2) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2}\log(|\mathbf{S}|) - \frac{1}{2\sigma^2}(\mathbf{y}_q - \mathbf{X}_q\beta)^\top(\mathbf{y}_q - \mathbf{X}_q\beta), \tag{10}$$

where $|\mathbf{S}|$ is the determinant of the Toeplitz matrix $\mathbf{S}$ depending on $\rho$, such that $\mathbf{V}_q = \sigma^2\mathbf{S}$ (recalling that $\mathbf{V}_q = \mathbf{C}\mathbf{V}_m\mathbf{C}^\top$).

## 3 The package

In this Section, we showcase the main functions that has been included in the **DisaggregateTS** package. Following Sax et al. (2023), we first introduce the main function of the package and it its features, and subsequently we will showcase other functions that allow the practitioner to conducting simulations and analyses.

### 3.1 Functions

The main function of the package which performs the sparse temporal disaggregation method proposed by Mosley et al. (2022) is `disaggregate()`. This function is of the following form:

```
disaggregate(Y, X, aggMat, aggRatio, method, ...)
```

where the first argument of the function, `Y`, corresponds to the $n \times 1$ vector of low-frequency series $\mathbf{y}_q$ that we wish to disaggregate, and the second argument, `X`, is the $p \times d$ matrix of high-frequency indicator series $\mathbf{X}_m$. In the event that there is no input `X`, the disaggregation matrix $\mathbf{X}_m$ is replaced with an $n \times 1$ vector of ones.

The argument `aggMat` coincides with the aggregation matrix $\mathbf{C}$ in (3), and it has been set to `"sum"` by default, rendering it suitable for flow data. Alternative options include `"first"`, `"last"` and `"average"`. The aggregation (distribution) matrices that are utilized in this function are summarized in table 2 of Sax et al. (2023).

The argument `aggRatio` has been set to `4` by default, which represents the ratio of annual to quarterly data. In general, this argument should be set to the ratio of the high-frequency to low-frequency series. For instance, in the examples considered in the preceding Sections, we had considered quarterly data as the low-frequency series, and monthly data as its high-frequency counterpart. Thus, in this setting `aggRatio = 3`. At first glance, the presence of the `aggRatio` argument may seem redundant. However, if $n \geq n_l \times$ aggRatio, then extrapolation is done up to $n$.

Finally, the argument `method` refers to the method of disaggregation under consideration. This argument has been set to `"Chow-lin"` method by default, which is the classical regression-based disaggregation technique introduced in Section 2.1. Other classical low-dimensional options include `"Denton"`, `"Denton-Cholette"`, `"fernandez"`, and `"Litterman"`, where these techniques have been extensively discussed in Dagum and Cholette (2006) and Sax and Steiner (2013). The main contribution of this package stems from the `"spTD"` and `"adaptive-spTD"` options pertaining to sparse temporal disaggregation and adaptive sparse temporal disaggregation, which are Mosley et al. (2022)'s high-dimensional extension of the regression-based techniques proposed by Chow and Lin (1971). In a high-dimensional regression, the adaptive LASSO is relevant when, for instance, the columns of the design matrix $\mathbf{X}$ exhibit multicollinearity, and the *Irrepresentability Condition* (IC hereafter) is violated (see Zou, 2006, for details). In such settings, the regularization parameter $\lambda$ does not satisfy the oracle property, which can lead to inconsistent variable selection. The adaptive counterpart of the regularized GLS cost function (8), can be expressed as follows:

$$\hat{\beta}(\lambda_n \mid \rho) = \arg\min_{\beta\in\mathbb{R}^d} \left\{ \left\| \mathbf{V}_q^{-\frac{1}{2}}(\mathbf{y}_q - \mathbf{X}_q\beta) \right\|_2^2 + \lambda_n \sum_{j=1}^{d} \frac{|\beta_j|}{|\hat{\beta}_{\text{init},j}|} \right\}, \tag{11}$$

where $\hat{\beta}_{\text{init},j}$ is an initial estimator, predicated on $\hat{\beta}(\hat{\rho})$ from the regularized (LASSO) temporal disaggregation. See Mosley et al. (2022), for the details of the proposed methodology, and Zou (2006) and

Van de Geer et al. (2011) to yield variable selection consistency using the OLS estimator and LASSO as $\hat{\beta}_{\text{init},j}$ when the IC condition is violated.

The second main function of the **DisaggregateTS** package is `TempDisaggDGP()`, which generates synthetic data that can be used for conducting simulations using the `disaggregate()` function. The main arguments of this function are as follows:

```
TempDisaggDGP(n_l, n, aggRatio, p, beta, sparsity, method, aggMat, rho, ...)
```

where the first argument corresponds to the size of low-frequency series and `n` to that of the high-frequency series. Moreover, `aggRatio` and `aggMat` are defined as before, in turn representing the ratio of the high-frequency to low-frequency series, as well as the aggregation matrix (3). A minor difference in the DGP function is that if $n \geq n_l \times \text{aggRatio}$, then the last $n - \text{aggRatio} \times n_l$ columns of the aggregation matrix are set to zero, such that $Y$ is observed only up to $n_l$. Argument `p` sets the dimensionality of high-frequency series (set to 1 by default), `beta` which has been set to `1` by default is the positive and negative elements of the coefficient vector, `sparsity` is the sparsity percentage of the coefficient vector, and `rho` is the autocorrelation of the error terms, which has been set to `0` by default. Finally, the `method` argument determines the data generating process of the error terms, corresponding to methods discussed earlier in this Section.

A number of optional arguments included in the function determine the mean vector and the standard deviation of the design matrix, as well as options such a setting seed for running the simulations, where the design matrix and the coefficient vectors are fixed.

In what follows, we showcase a simple example of the function and its respective outputs:

```
# Generate low-frequency quarterly series and its high-frequency monthly counterpart
SynthethicData <- TempDisaggDGP(n_l = 2,
                                n = 6,
                                aggRatio = 3,
                                p = 6,
                                beta = 0.5,
                                sparsity = 0.5,
                                method  = 'Chow-Lin',
                                rho = 0.5)
```

In the example above, we generate low-frequency series $\mathbf{y}_q \in \mathbb{R}^2$ corresponding to two quarters, and consequently, its high-frequency monthly counterpart $\mathbf{y}_m \in \mathbb{R}^6$. It is further assumed that the data is generated using six monthly indicators - i.e., $\mathbf{X}_m^{6\times 6}$, with a coefficient vector $\beta \in \mathbb{R}^6$, where $\beta_j \in \{-0.5, 0, +0.5\}$. Since, the sparsity argument is set to 0.5, only half of $\beta$'s elements are non-zero. Finally, the error vector $\mathbf{u}_m$ is assumed to follow the AR(1) structure of Chow and Lin (1971), with an autocorrelation parameter of $\rho = 0.5$.

## 4 Simulations

In this Section, we show a simulation exercise to demonstrate the implementation of the temporal disaggregation method via the **DisaggregateTS** package.

### 4.1 Classical setting

We start by simulating the dependent variable $Y \in \mathbb{R}^{17}$ and the set of high-frequency exogenous variables $X \in \mathbb{R}^{68\times 5}$ by using the command:

```
# Set seed for reproducibility
set.seed(27)
# Generate low-frequency yearly series and its high-frequency quarterly counterpart
n_l <- 17 # The number of low-frequency data points - annual
n <- 68 # The number of high-frequency (quarterly) data points.
p_sim <- 5 # The number of the high-frequency exogenous variables.
rho_sim <- 0.8 # autocorrelation parameter
Sim_data <- TempDisaggDGP(n_l,
                          n,
                          aggRatio = 4,
                          p = p_sim,
```

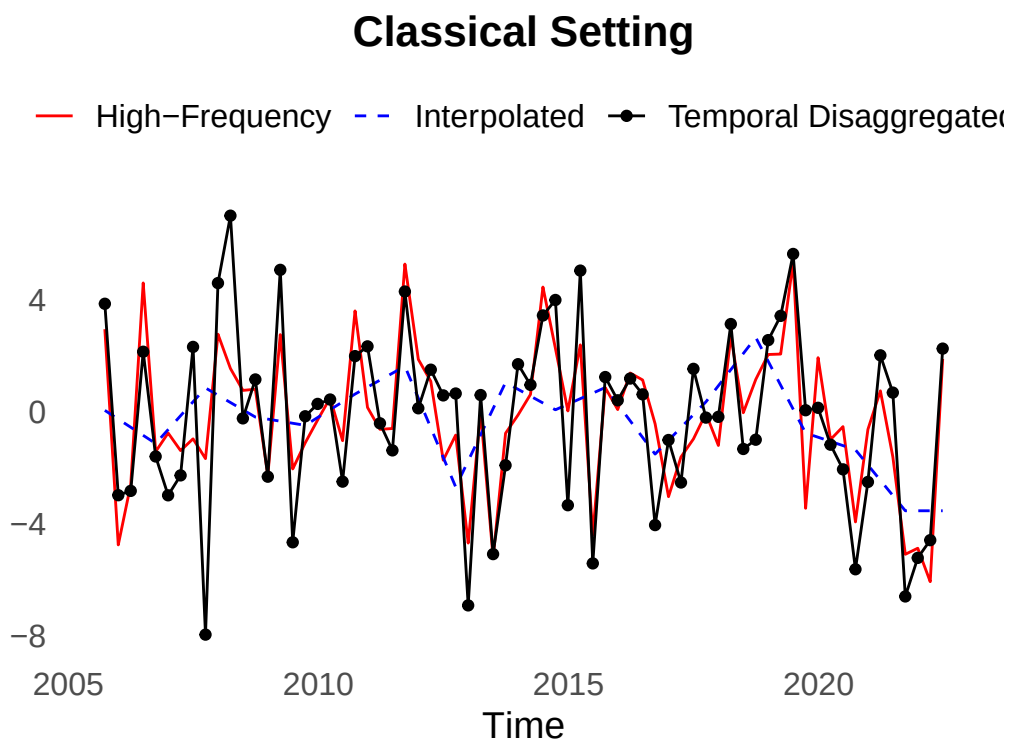



**Figure 3:** Temporal disaggregated and interpolated observations for the estimation under the classical setting. The plot is built using the snippet code provided in this subsection. As we used the setting `aggMat = sum`, the sum of every four disaggregated observations corresponds to an actual low-frequency observation.

```
                          rho = rho_sim)
Y_sim <- matrix(Sim_data$Y_Gen) # Extract the simulated dependent low-frequency variable
X_sim <- matrix(Sim_data$X_Gen) # Extract the simulated dependent
# (low-frequency) variable
Y_sim_HF_obs <- matrix(Sim_data$y_Gen) #  HF simulated observations
```

In this example, we are generating a set of low-frequency data, i.e. 17 annual datapoints and a set of high-frequency (quarterly) exogenous variables that we want to use to infer the high-frequency counterpart of the low-frequency data. We now want to temporally disaggregate the low-frequency time series by using the information encapsulated in the high-frequency time series. In this case, since the number of time observations is larger than the number of exogenous variables, we can use standard methodologies to estimate the temporal disaggregation model. To do so, we use the `disaggregate()` function setting `method="Chow-Lin"`. The code is as follows:

```
C_sparse_SIM <- disaggregate(Y_sim,
                             X_sim,
                             aggMat = "sum",
                             aggRatio = 4,
                             method = "Chow-Lin")
C_sparse_SIM$beta_Est

#> 1 x 1 Matrix of class "dgeMatrix"
#>           [,1]
#> [1,] 0.3193439

Y_HF_SIM <- C_sparse_SIM$y_Est[ ,1] # Extract the temporal disaggregated
# dependent variable estimated through the function disaggregate()
```

We show in Figure 3 the results, where we depict the high-frequency observation computed via standard interpolation and estimated through the Chow-Lin temporal disaggregation method.

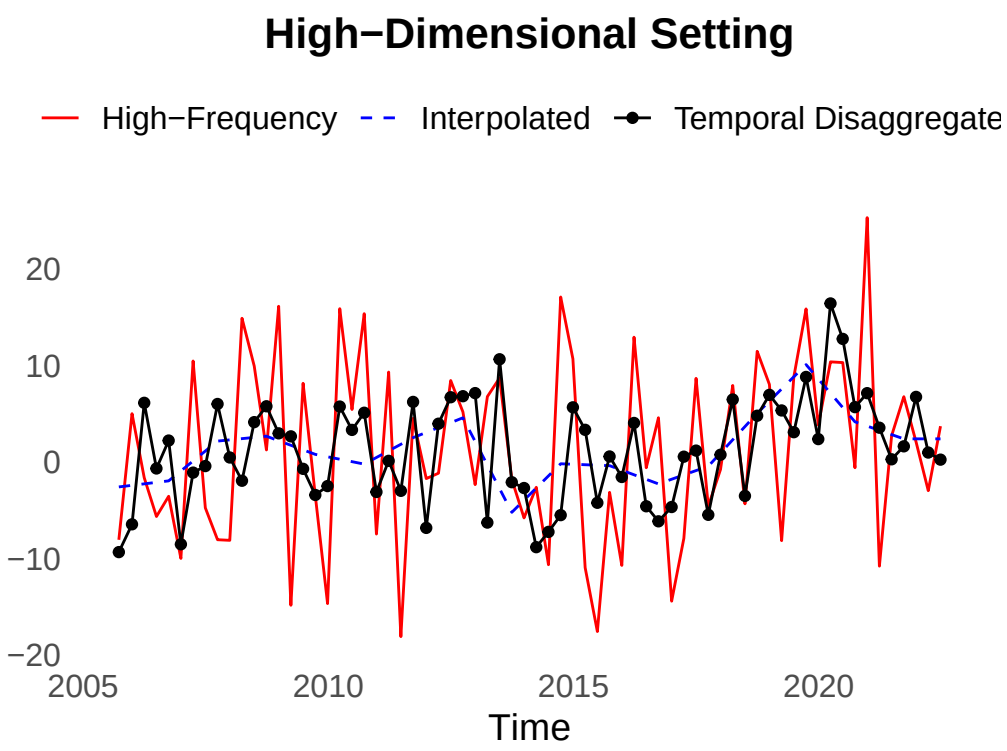



**Figure 4:** Temporal disaggregated and interpolated observations for the estimation under the high-dimensional setting. The plot is built using the snippet code provided in this subsection. As we used the setting `aggMat = sum`, the sum of every four disaggregated observations corresponds to an actual low-frequency observation.

### 4.2 High-dimensional setting

We now repeat the simulation experiment in a high-dimensional setting, where the number of temporal observations is lower than the number of exogenous variables. In this case, standard methods like Chow-Lin cannot be applied. To do so, we simulate the dependent variable $Y \in \mathbb{R}^{17}$ as before, but now the set of high-frequency exogenous variables is of dimension $X \in \mathbb{R}^{68\times 100}$. Similarly, as before, we can use the following command:

```
# Generate low-frequency yearly series and its high-frequency quarterly counterpart
set.seed(27)
n_l <- 17 # The number of low-frequency data points
n <- 68 # The number of high-frequency data points - quarterly
p_sim <- 100 # The number of the high-frequency exogenous variables.
rho_sim <- 0.8 # autocorrelation parameter
Sim_data <- TempDisaggDGP(n_l,
                          n,
                          aggRatio = 4,
                          p = p_sim,
                          rho = rho_sim)
Y_sim <- matrix(Sim_data$Y_Gen) #Extract the simulated dependent
# (low-frequency) variable
X_sim <- Sim_data$X_Gen #Extract the simulated exogenous variables - high-frequency
```

In this case, we cannot use a standard technique, and we need to estimate a sparse model to overcome the curse of dimensionality. The `disaggregate()` function can handle the high-dimensional setting by choosing the method to be `"spTD"` or `"adaptive-spTD"`. In the following example, we use the latter:

As we can see from both Figures 3 and 4, the standard interpolation cannot reproduce the fluctuations of the data, making the result overly smooth. On the other hand, the temporal disaggregation methods demonstrate fluctuations in line with the actual observations. We remark that the degree of success in recovering the short-term fluctuations depends considerably on the setting, and as one may expect, performance is not as strong in the high-dimensional scenario where the estimator must simultaneously search for appropriate indicators and estimate the parameters.

We would in general caution against throwing sets of non-curated (i.e. possibly irrelevant) indicators into any of the Chow-Lin methods. Instead, we recommend that where possible, indicators that are designed to track the outcome of interest are used, e.g., we may wish to benchmark (align) monthly flash-indicators of GDP against more reliable yearly/quarterly observations, in this case, the flash estimates can be used as an indicator within a standard Chow-Lin approach. If there is a larger set of indicators which practitioners find hard to decide amongst, that is all indicators could be feasible from the practitioner view, then we recommend users may try the model-selection based `"spTD"`, and `"adaptive-spTD"` routines. In some settings, use of the classical Chow-Lin procedure becomes impossible due to the lack of low-frequency responses, in such settings it is necessary to either apply regularization or perform model-selection.

To conclude the simulation exercise, we present a set of statistics from 1,000 simulations for both the linearly interpolated and temporally disaggregated time series across the two simulation settings. The results are summarized in Table 1. As it can be observed, the temporally disaggregated time series provides a much better representation of the real data's level of dispersion, while interpolation consistently underestimates it. In terms of MSE and MAE, the findings are mixed: for the classic case, temporal disaggregation shows lower values for both metrics, whereas in the high-dimensional setting, the opposite trend is observed.

**Table 1:** Mean and standard deviation (in parentheses) of key statistical measures for the high-frequency simulated observations, temporal disaggregated observations, and interpolated observations across 1000 Monte Carlo simulations. MAE and MSE are computed with respect to the high-frequency simulated observations.

| Statistic | Classical_Obs | Classical_Temporal_Disaggregation | Classical_Interpolation | HighDim_Obs | HighDim_Temporal_Disaggregation | HighDim_Interpolation |
|---|---|---|---|---|---|---|
| Standard Deviation | 2.716 (0.258) | 2.888 (0.597) | 1.488 (0.321) | 10.089 (0.888) | 8.478 (1.696) | 4.18 (0.817) |
| Kurtosis | 2.897 (0.519) | 2.902 (0.527) | 2.689 (0.694) | 2.898 (0.558) | 2.890 (0.523) | 2.892 (0.772) |
| MSE | - (-) | 2.405 (1.733) | 5.158 (0.977) | - (-) | 104.463 (28.519) | 86.095 (16.258) |
| MAE | - (-) | 1.185 (0.386) | 1.809 (0.178) | - (-) | 8.136 (1.101) | 7.403 (0.735) |

## 5 Empirical application

In this Section, we show how temporal disaggregation can be used in a real-world problem.

The urgent need to address climate change has propelled the scientific community to explore innovative approaches to quantify and manage greenhouse gas (GHG) emissions. Carbon intensity, a crucial metric that measures the amount of carbon dioxide equivalent emitted per unit of economic activity (e.g. sales), plays a pivotal role in assessing the environmental sustainability of industries, countries, and global economies. By focusing on emissions per unit of economic output, carbon intensity accounts for the fact that larger organizations or economies may naturally produce more emissions simply due to scale. This allows for a fair comparison of sustainability performance across different entities, regardless of size. Accurate and timely carbon accounting and the development of robust measurement frameworks are essential for effective emission reduction strategies and the pursuit of sustainable development goals. While carbon accounting frameworks offer valuable insights into emissions quantification, they are not without limitations. One of those limitations is the frequency with which this information is released, generally at an annual frequency, while most companies' economic indicators are made public on a quarterly basis. This is a perfect example in which temporal disaggregation can be used to bridge the gap between data availability and prompt economic and financial analyses. In this application, the variable of interest is the GHG emissions for IBM between Q3 2005 and Q3 2021, at annual frequency, resulting in 17 datapoints, i.e. $Y \in \mathbb{R}^{17}$. For the high-frequency data, we used the balance sheet, income statement, and cash flow statement quarterly data between Q3 2005 and Q3 2021, resulting in 68 datapoints for the 128 variables. We remove variables that have a pairwise correlation higher than 0.99, resulting in a filtered dataset with 112 variables, i.e. $X \in \mathbb{R}^{68\times 112}$. In this example, we employed the adaptive LASSO method (`method = "adaptive-spTD"`) as a way to select the best variables that can be used to recover the high-frequency observations and we applied the `aggMat = "sum"` aggregation method. The rationale for using this method, in conjunction with `aggRatio = 4` is to ensure that the disaggregated quarterly carbon intensity values are consistent with the overall annual figures as the goal is to break down the emissions and sales data such that the sum of the quarterly carbon intensities equals the yearly total. The adaptive LASSO procedure select only seven non-zero coefficients, which are Actual sales per employee, trailing 12-month net sales, Enterprise value, Current liabilities, total liabilities and equity, total line of credit and net debt. Actual sales per employee, trailing 12-month net sales and Enterprise value indicate company sales and size, both linked to the company's economic activity, operational intensity, and commitment to sustainability, and hence potential emissions. Current liabilities and total liabilities and equity reflect the company's financial position and operational scale, both of which might influence emissions. Total

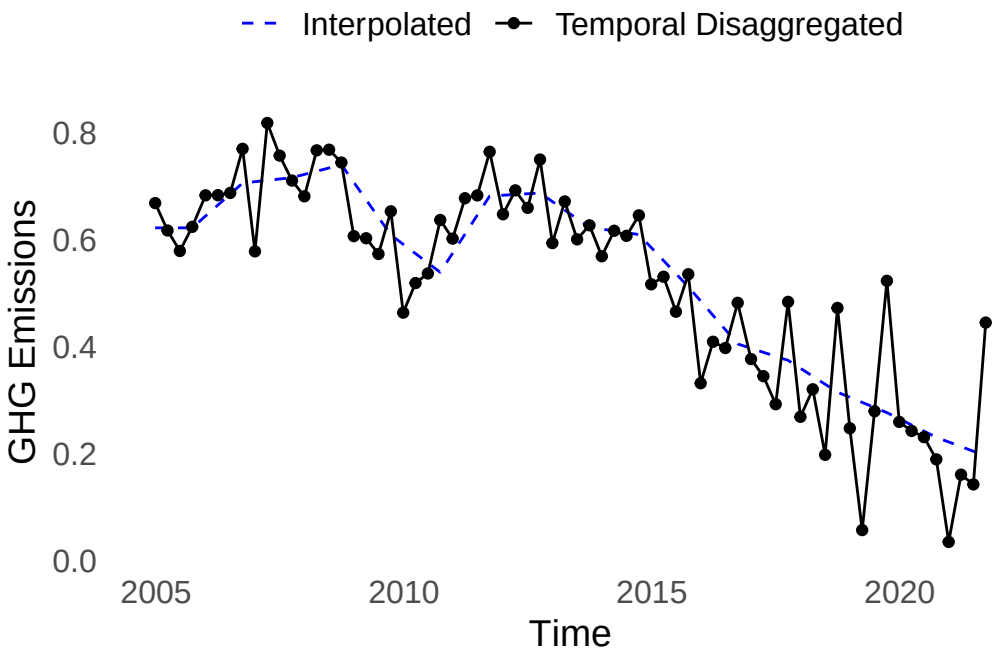



**Figure 5:** Temporal disaggregated and interpolated GHG emissions observations. In this example, we used the setting `aggMat = sum`, so the sum of the quarterly disaggregated GHG sums to the actual annual observation.

line of credit and net debt highlight the company's ability to borrow, which could impact investment in emissions-reducing projects. We show the results in Figure 5 alongside a linear interpolation method.

As it is possible to observe from the plot, the interpolated data do not fluctuate as we would expect from real GHG emissions, as the method is not conditional on the variability of the high-frequency variables. In this respect, the temporal disaggregated observations show more realistic dynamics. This result can then be used to compute the GHG intensity, computing the ratio between GHG emissions and the sales for the corresponding quarter.

## 6 Summary

In this paper, we have given an overview of the key functionality for the **DisaggregateTS** R package. The package builds on features of the existing **tempdisagg** package allowing the user to easily apply the regularized Chow-Lin type procedure of Mosley et al. (2022) and compare this with classical methods based on interpolation via smoothing (e.g. Denton, 1971), or the Fernández (1981) and Fernández (1981) methods in the case where series are not co-integrated. The package may be extended in future to allow cross-sectional constraints, the disaggregation of panel time-series data, and the accommodation of factor (latent-variable) structures in the temporal-disaggregation problem.

It is important to point out that when performing disaggregation, one should be careful to select an appropriate set of indicator time-series, and not purely rely on the model-selection procedures deployed here. Whilst these methods can help pick from a set of indicators, there is still room for them to pick irrelevant/spurious series, especially given the short nature of the aggregate series. To this end, we do not recommend just relying on one method alone, but rather a user deploy several methods of disaggregation. The user should then attempt to obtain some form of (potentially qualitative) external validation of the resultant series in their application of choice. One should always remember, that as the high-frequency series is not observed, there can be no direct empirical validation of the methods other than at the aggregate scale.

## References

A. Belloni and V. Chernozhukov. Least squares after model selection in high-dimensional sparse models. *Bernoulli*, 19(2):521–547, 2013. [p65]

J. Bournay and G. Laroque. Réflexions sur la méthode d'elaboration des comptes trimestriels. In *Annales de l'INSEE*, volume 36, pages 3–30. JSTOR, 1979. [p64, 65]

P.-A. Cholette. Adjusting sub-annual series to yearly benchmarks. *Survey Methodology*, 10(1):35–49, 1984. [p65]

G. C. Chow and A.-l. Lin. Best linear unbiased interpolation, distribution, and extrapolation of time series by related series. *The review of Economics and Statistics*, 53(4):372–375, 1971. [p63, 64, 65, 66, 67]

G. C. Chow and A.-L. Lin. Best linear unbiased estimation of missing observations in an economic time series. *Journal of the American Statistical Association*, 71(355):719–721, 1976. [p63]

A. Cottrell and R. Lucchetti. *Gretl User's Guide: Gnu Regression, Econometrics and Time-series Library*, 2023. URL https://gretl.sourceforge.net/gretl-help/gretl-guide.pdf. Accessed: 2023-10-08. [p65]

E. B. Dagum and P. A. Cholette. *Benchmarking, Temporal Distribution, and Reconciliation Methods for Time Series*. Springer, 2006. [p62, 63, 66]

F. T. Denton. Adjustment of monthly or quarterly series to annual totals: an approach based on quadratic minimization. *Journal of the american statistical association*, 66(333):99–102, 1971. [p63, 65, 71]

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2): 407–451, 2004. [p65]

R. B. Fernández. A methodological note on the estimation of time series. *The Review of Economics and Statistics*, 63(3):471–476, 1981. [p63, 65, 71]

P. Fuleky. *Macroeconomic forecasting in the era of big data: Theory and practice*, volume 52. Springer, 2019. [p62]

R. B. Litterman. A random walk, markov model for the distribution of time series. *Journal of Business & Economic Statistics*, 1(2):169–173, 1983. [p63, 65]

L. Mosley and K. S. Nobari. *DisaggregateTS: High-Dimensional Temporal Disaggregation*, 2024. URL https://CRAN.R-project.org/package=DisaggregateTS. R package version 3.0.1. [p62]

L. Mosley, I. A. Eckley, and A. Gibberd. Sparse Temporal Disaggregation. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 185(4):2203–2233, 10 2022. ISSN 0964-1998. doi: 10.1111/rssa.12952. URL https://doi.org/10.1111/rssa.12952. [p62, 63, 65, 66, 71]

E. M. Quilis. Temporal disaggregation of economic time series: The view from the trenches. *Statistica Neerlandica*, 72(4):447–470, 2018. [p64]

C. Sax and P. Steiner. Temporal disaggregation of time series. *The R Journal*, 5(2):80–87, 2013. [p62, 65, 66]

C. Sax, P. Steiner, and T. Di Fonzo. *'tempdisagg': Methods for Temporal Disaggregation and Interpolation of Time Series*, 2023. URL https://CRAN.R-project.org/package=tempdisagg. R package version 1.1.1. [p62, 64, 66]

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. [p65]

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. [p63]

S. Van de Geer, P. Bühlmann, and S. Zhou. The adaptive and the thresholded lasso for potentially misspecified models (and a lower bound for the lasso). *Electronic Journal of Statistics*, 5:688–749, 2011. [p67]

M. J. Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge university press, 2019. [p62]

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101 (476):1418–1429, 2006. [p66]

*Luke Mosley*
*Lancaster University*
*School of Mathematical Sciences*
*Lancaster, United Kingdom*
lukemos313@gmail.com

*Kaveh Salehzadeh Nobari*
*Imperial College LondonLondon School of Economics and Political Science*
*Centre for Climate Finance & Investment, London, United Kingdom*
*Department of Psychological and Behavioural Science, London, United Kingdom*
k.salehzadeh-nobari@imperial.ac.uk

*Giuseppe Brandi*
*Imperial College LondonNortheastern University London*
*Centre for Climate Finance & Investment, London, United Kingdom*
*Khoury College of Computer Sciences, London, United Kingdom*
g.brandi@imperial.ac.uk

*Alex Gibberd*
*Lancaster University*
*School of Mathematical Sciences*
*Lancaster, United Kingdom*
a.gibberd@lancaster.ac.uk

# LMest: An R Package for Estimating Generalized Latent Markov Models

*by Fulvia Pennoni, Silvia Pandolfi, and Francesco Bartolucci*

**Abstract** We provide a detailed overview of the updated version of the R package LMest, which offers functionalities for estimating Markov chain and latent or hidden Markov models for time series and longitudinal data. This overview includes a description of the modeling structure, maximum-likelihood estimation based on the Expectation-Maximization algorithm, and related issues. Practical applications of these models are illustrated using real and simulated data with both categorical and continuous responses. The latter are handled under the assumption of the Gaussian distribution given the latent process. When describing the main functions of the package, we refer to potential applicative contexts across various fields. The LMest package introduces several key novelties compared to previous versions. It now handles missing responses under the missing-at-random assumption and provides imputed values. The implemented functions allow users to display and visualize model results. Additionally, the package includes functions to perform parametric bootstrap for inferential procedures and to simulate data with complex structures in longitudinal contexts.

## 1 Introduction

Markov chain (MC) and latent or hidden Markov (HM) models are gaining increasing attention due to possibility of handling the temporal structure of many observed phenomena (Meyn and Tweedie, 2012; Mor et al., 2021). In particular, with only one categorical response, an MC model allows studying the initial distribution of this response variable and its conditional distribution given the previous response. On the other hand, HM models offer a practical model-based dynamic clustering and classification method (Bouveyron et al., 2019). This class of models finds application in the analysis of time-series (Ephraim and Merhav, 2002; Zucchini et al., 2016) and longitudinal data (Wiggins, 1973; Bartolucci et al., 2013, 2022). The model formulation involves the introduction of a sequence of individual and time specific discrete latent (hidden) variables. This results in a hidden process assumed to follow a Markov chain of first order. The states of this chain correspond to latent clusters or subpopulations of individuals with similar latent characteristics. The approach can handle responses of different types, whether continuous or categorical. In the latter case, the conditional distribution of these variables is freely parameterized on the basis of conditional response probabilities.

In this article we provide a practical introduction to MC and HM models through an overview of the **LMest** package, focusing in particular on the new features of this package compared to the previous versions, such as the one illustrated in Bartolucci et al. (2017). The package allows analyzing time-series and longitudinal data by the models at issues. In the longitudinal context, recurring measurements over time on possibly multiple characteristics are taken on several sampling units (e.g., individuals), in such a way that it is possible to describe the evolution of these characteristics over time.

The package in its updated version 3.2.5 is available on CRAN at **LMest** and it is also described with detailed vignettes, including applicative examples. Each function is well documented in the help provided within the package, and several datasets from surveys and other sources are included in the package in both wide and long formats.

The current version of the package has the following features, which were also present in the previous versions:

- it is designed to estimate different model formulations, such as MC, HM, and mixed HM models, through the Expectation-Maximization (EM) algorithm (Dempster et al., 1977);
- it allows the estimation of the effect of covariates under different parameterizations and model specifications;
- model selection procedures using the Akaike Information Criterion (AIC, Akaike, 1973) and the Bayesian Information Criterion (BIC, Schwarz, 1978) are included, relying on different parameter initialization strategies;
- functions to produce local and global decoding are implemented to perform dynamic clustering;
- standard errors for the parameter estimates are obtained either through exact computation or reliable approximations of the observed information matrix; parametric bootstrap procedures (Davison and Hinkley, 1997) are also available for different model specifications;
- in order to increase the computational efficiency, Fortran routines are used to perform numerical operations.

Some important extensions have been introduced in the most recent version of the package. The features of these new functions can be summarized as follows:

- in addition to categorical data, the package can handle continuous data, assuming a Gaussian distribution for the response variables conditional on the latent process. It also accommodates missing responses, drop-out, and intermittent missingness under the missing-at-random assumption (MAR, Little and Rubin, 2020);
- when dealing with continuous outcomes, covariates may be included under different model specifications, similar to the approach used for categorical outcomes;
- simulations from almost all the available model specifications may be carried out through a suitable method;
- data in long format can be provided as input, and the model specification follows the common R style; formulas for the response variables, as well as for the initial and transition probabilities can be specified using the package **Formula** (Zeileis and Croissant, 2010);
- parameter estimates can be graphically displayed to enhance the interpretation of the results.

Overall, the current version of the package offers great flexibility in model formulation and estimation.

In the following sections, after covering the main theoretical aspects of the models, we provide an overview of the **LMest** software package through examples using univariate and multivariate data with and without missing values. The package uses the S3 object-oriented system, defining three main classes of objects. We use both synthetic and real data to illustrate the models, where the synthetic data are generated using functions available within the package. These data are similar to those used in previous applied works published in the authors' research articles or by other researchers in the field. However, we prefer to use simulated versions because the original data are not always publicly available. Additionally, using such data allows us to incorporate specific features that are useful for illustrating certain functions of the package. When describing the data, we will refer to potential real-world applications to characterize the research questions of interest, thereby better motivating the adopted models. Specifically:

- the MC model is used to analyze longitudinal data about labor market careers after graduation;
- the HM model for categorical longitudinal responses is adopted to examine the customer's purchase behavior over time;
- the HM model for continuous time-series data is used to discover financial market phases;
- the HM model for continuous longitudinal responses is used to investigate the state progression of illness in patients after treatment and the progression of students performance in different types of school.

In the R examples we use `set.seed(x)` both when we generate the data and fit the models so that all the output can be replicated. The code for replicability of the simulated data can be provided upon request.

In the following we introduce the models, examples, and codes in an expository style to demonstrate the use of specific functions of **LMest**. More details about MC and HM models are provided in Bartolucci et al. (2013, 2014b). The remainder of the paper is structured as follows: the next section presents the MC model, while the third section introduces the HM model for categorical longitudinal data. The fourth section presents the HM model for continuous responses, which allows for missing values and individual covariates. Finally, the fifth section provides a summary, mentions some of other similar packages, and discusses additional issues related to future package extensions.

## 2 Markov chain model

In the following we first illustrate the assumptions of the MC model for categorical responses and then an application based on a longitudinal categorical response and covariates.

### 2.1 Model assumptions

In the context of longitudinal data, the MC model is referred to as the transition model (Anderson, 1954) since it is of interest to estimate the transition between states of the stochastic process. With only one categorical response variable, let $\mathbf{Y}_i = (Y_{i1}, \ldots, Y_{iT})'$ denote the vector of such variables for individual $i$, with $i = 1, \ldots, n$, where $Y_{it}$ has $k$ categories labeled from 1 to $k$. Let $\boldsymbol{x}_{it}$ denote the vector of individual covariates available at the $t$-th time occasion for individual $i$, with $t = 1, \ldots, T$.

The main assumption of the model is that, for $t = 3, \ldots, T$, $Y_{it}$ is conditionally independent of $Y_{i1}, \ldots, Y_{i,t-2}$ given $Y_{i,t-1}$ when a first-order dependence structure is formulated. Moreover, in its basic version, the model is characterized by initial and transition probabilities that, for every $i = 1, \ldots, n$, are denoted by

$$\pi_u = p(Y_{i1} = u), \quad u = 1, \ldots, k$$

and

$$\pi_{uv} = p(Y_{it} = v | Y_{i,t-1} = u), \quad t = 2, \ldots, T, \; u, v = 1, \ldots, k,$$

respectively.

In presence of individual time-fixed and time-varying covariates, the initial and transition probabilities are denoted as $\pi_{i,u} = p(Y_{i1} = u | \boldsymbol{x}_{i1})$, $u = 1, \ldots, k$, and $\pi_{it,uv} = p(Y_{it} = v | Y_{i,t-1} = u, \boldsymbol{x}_{it})$, $t = 2, \ldots, T$, $u, v = 1, \ldots, k$, respectively. Note that these probabilities are now individual specific, and their dependence on the vector of covariates is formulated on the basis of multinomial logit models (Azzalini, 1994). In particular, for the initial probabilities we have

$$\log \frac{\pi_{i,u}}{\pi_{i,1}} = \beta_{0u} + \boldsymbol{x}_{i1}' \boldsymbol{\beta}_{1u}, \quad u = 2, \ldots, k. \tag{1}$$

Note that, as a reference category, the multinomial logit model in (1) has the first state. For the transition probabilities we assume

$$\log \frac{\pi_{it,uv}}{\pi_{it,uu}} = \gamma_{0uv} + \boldsymbol{x}_{it}' \boldsymbol{\gamma}_{1uv}, \quad t = 2, \ldots, T, \; u, v = 1, \ldots, k, \; u \neq v, \tag{2}$$

where the logits have, as reference state, that corresponding to the row of the transition matrix. In the above expressions, $\boldsymbol{\beta}_u = (\beta_{0u}, \boldsymbol{\beta}_{1u}')'$ and $\boldsymbol{\gamma}_{uv} = (\gamma_{0uv}, \boldsymbol{\gamma}_{1uv}')'$ are parameter vectors to be estimated.

Denoting by $\boldsymbol{\theta}$ the vector of all model parameters, the log-likelihood can be defined as

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} f(y_{i1}, \ldots, y_{iT} | \boldsymbol{x}_{i1}, \ldots \boldsymbol{x}_{iT}),$$

where $f(y_{i1}, \ldots, y_{iT} | \boldsymbol{x}_{i1}, \ldots \boldsymbol{x}_{iT})$ is the probability of the observed vector of response variables for unit $i$ and $y_{it}$ is a realization of $Y_{it}$. Under the assumptions formulated for the MC model, and in presence of individual covariates, this probability may be expressed as

$$f(y_{i1}, \ldots, y_{iT} | \boldsymbol{x}_{i1}, \ldots \boldsymbol{x}_{iT}) = \pi_{i,y_{i1}} \prod_{t=2}^{T} \pi_{it,y_{i,t-1}y_{it}}.$$

For the basic MC model without covariates, explicit formulas are available for the maximum likelihood estimation of the parameters. Under more complex models, for example when individual covariates are included as in Equations (1) and (2), an iterative algorithm, such as the Newton-Raphson or the Fisher-scoring, is required to maximize the above log-likelihood. As is well known, at each step of these algorithms, the parameter vector is updated by adding to its current value the inverse of the observed or expected information matrix, which is multiplied by the score vector, that is, the vector of the first derivatives of $\ell(\boldsymbol{\theta})$. These steps are performed until a suitable convergence criterion is satisfied. The corresponding asymptotic standard errors can be obtained in the usual way from the diagonal elements of the information matrix; see Bartolucci et al. (2014b) for more details.

Finally note that, although we illustrate the MC model only for the case of balanced longitudinal data, this model can obviously be applied also to unbalanced data, when we have a specific number of observations $T_i$ for each individual $i$. However, for this aim, it is convenient to use the function of **LMest** for HM models under a specific constraint, as will be clarified in the following.

### 2.2 Application to longitudinal data with covariates

The present example illustrates an application of the MC model with a univariate categorical response and time-fixed covariates. The data provided within the package, named `data_employment_sim`, are simulated supposing they come from a survey context, where interviews are conducted among a nationally representative sample of graduates about their employment status after graduation; for analyses of similar data see Bartolucci and Pennoni (2011). In this scenario, the binary response variable (`emp`) is recorded for three time occasions corresponding to one, two, and three years after graduation. Individual covariates, which are assumed to be time-fixed, are the geographical location of the university where individuals graduated, assuming the country is divided into two main areas (`area`: 1 for South, 2 for North), the grade at graduation categorized into three levels (`grade`: 1 for low, 2 for medium, and 3 for high), and the indicator for parents' educational university qualification (`edu`:

1 for university degree, 0 otherwise). In simulating these data, we set the sample size equal to $n = 585$, and we consider $T = 3$ time occasions.

According to the simulated context, analyzing such data with the MC model allows us to identify the employment patterns immediately after graduation and evaluate trends over time. Additionally, the research question is whether the probability of employment in the first period after graduation depends on the geographical area or on the final degree grade. Then, we can explore if and how family background influences the probability of employment.

The type of data structures that can be given in input to the estimation function is in long format, that is, a data frame with one row for each combination of unit $i$ and time occasion $t$, with $i = 1, \ldots, n$ and $t = 1, \ldots, T$, so that the total number of rows of this object is $nT$. All columns should have names; one column must correspond to the unit identifier, typically named with `id`, and another column must correspond to the time occasions, typically named with `time`.

The data stored in the long format can be described by the usual `summary()` method after they have been prepared by function `lmestData()`, as follows:

```
data("data_employment_sim")
dt <- lmestData(responsesFormula = emp ~ area + grade + edu,
                id = "id", time = "time",
                data = data_employment_sim)
```

In particular, the frequency distribution of the response variable for each wave is reported below:

```
summary(dt, dataSummary = "responses", varType = rep("d", ncol(dt$Y)))
```

```
#>
#> Data Info:
#> ----------
#>
#> Observations:        585
#> Time occasions:      3
#> Variables:           1
#>
#>
#> Proportion:
#> ----------
#>
#>  time    emp
#>  1:585   0:0.411396
#>  2:585   1:0.588604
#>  3:585
#>
#> Proportion by year:
#> ----------
#>
#>
#> Time =  1
#>
#>  time    emp
#>  1:585   0:0.4615385
#>          1:0.5384615
#>
#> Time =  2
#>
#>  time    emp
#>  2:585   0:0.4034188
#>          1:0.5965812
#>
#> Time =  3
#>
#>  time    emp
#>  3:585   0:0.3692308
#>          1:0.6307692
```

To estimate an MC model with covariates we can use the `lmestMc()` function. In this example, the interest is in estimating the effect, on the employment at the first interview, of the area where the students graduated and of their grade, and in evaluating the impact of the parent's educational level only for the periods following the first interview. Accordingly, covariates `area` and `grade` are included in `responsesFormula` argument so as to affect the initial probabilities, while `edu` is included in the parameterization of the transition probabilities. The covariates for initial and transition probabilities are separated by the symbol "`|`". Note that we use the **Formula** package for all formula operators (Zeileis and Croissant, 2010). Moreover, the argument `index` is required, which is a character vector to specify the name of the unit identifier (first element) and that of the time occasions (second element). Standard errors for the parameter estimates are provided by setting `out_se = TRUE` as in the following:

```
mod1 <- lmestMc(responsesFormula = emp ~
                as.factor(area)  + as.factor(grade) | as.factor(edu),
                index = c("id", "time"),
                data = dt, out_se = TRUE,
                output = TRUE, seed = 345)
```

Note that, in the above code, the data object `dt` returned by the `lmestData()` function can be directly passed to the estimation function. The tolerance level to check convergence of the EM algorithm and the maximum number of iterations of this algorithm are set by default as `tol = 10^-8` and `maxit = 1000`.

Using option `output = TRUE`, the `lmestMc()` function also returns the subject-specific estimated initial probabilities, collected in object `Piv`, and the estimated subject- and time-specific transition probabilities, collected in `PI`. To summarize these results, it is convenient to calculate suitable averages of these probabilities. Provided that the estimation output is stored in object `mod1`, for the initial probabilities we have:

```
print(round(colMeans(mod1$Piv), 3))
```

```
#>     0     1
#> 0.462 0.538
```

These estimates indicate that the 46% of individuals is unemployed at the beginning of the period of observation, that is, one year after graduation.

The array `PI` containing the estimates of the transition probabilities has dimension $2 \times 2 \times 585 \times 3$, where 2 is the number of response categories, 585 is the sample size, and 3 is the number of time occasions. The averaged transition probabilities, over all individuals and time occasions, are computed as follows starting from object `mod1` where such probabilities are stored:

```
print(round(apply(mod1$PI[,,,2:3], c(1,2), mean), 3))
```

```
#>          category
#> category     0     1
#>        0 0.761 0.239
#>        1 0.088 0.912
```

From these results we observe that unemployed individuals have a probability close to 0.76 of remaining without a job between two successive time occasions. Employed individuals have a higher persistence in the same employment status. The probability to find a job from one wave to another is around 0.24.

A summary of the output of the `lmestMc()` function including parameter estimates can be printed using the `summary()` method:

```
summary(mod1)
```

```
#> Call:
#> lmestMc(responsesFormula = emp ~ as.factor(area) + as.factor(grade) |
#>     as.factor(edu), data = dt, index = c("id", "time"), out_se = TRUE,
#>     output = TRUE, seed = 345)
#>
#> Coefficients:
#>
```

```
#>  Be - Parameters affecting the logit for the initial probabilities:
#>                    logit
#>                          2
#>   (Intercept)       -0.4576
#>   as.factor(area)2   0.4382
#>   as.factor(grade)2  0.1536
#>   as.factor(grade)3  1.6676
#>
#>  Standard errors for Be:
#>                    logit
#>                         2
#>   (Intercept)       0.1681
#>   as.factor(area)2  0.1847
#>   as.factor(grade)2 0.1988
#>   as.factor(grade)3 0.2529
#>
#>  p-values for Be:
#>                    logit
#>                        2
#>   (Intercept)       0.006
#>   as.factor(area)2  0.018
#>   as.factor(grade)2 0.440
#>   as.factor(grade)3 0.000
#>
#>  Ga - Parameters affecting the logit for the transition probabilities:
#>                   logit
#>                         1       2
#>   (Intercept)     -1.6015 -2.1789
#>   as.factor(edu)1  2.1834 -2.6251
#>
#>  Standard errors for Ga:
#>                   logit
#>                        1      2
#>   (Intercept)     0.1257 0.1423
#>   as.factor(edu)1 0.3128 1.0141
#>
#>  p-values for Ga:
#>                   logit
#>                    1    2
#>   (Intercept)     0 0.00
#>   as.factor(edu)1 0 0.01
```

Note that the `summary()` method also returns the standard errors for the parameter estimates and the corresponding significance level when option `out_se = TRUE` is included in the main estimation function. The argument `Be` returned by the function provides the estimated regression parameters of the logistic model for the probability of belonging to category 1 (employed) versus category 0 (unemployed) at the first interview. For interpretation, a higher degree grade positively affects the employment probability one year after graduation. Specifically, the log-odds ratios is equal to 0.154 for those with a medium grade and 1.668 for those with a high grade, with respect to individuals with a low grade, all the other covariates held fixed. Moreover, the log-odds ratio for `area` is positive, indicating that, at the beginning of the study, individuals who graduated from a university located in the North of the country are more likely to be employed than those who graduated from a university located in the South, with other covariates held constant.

Regarding the estimates referred to the transition probabilities reported in the output `Ga`, we may conclude that, apart for the intercept, a higher level of parents' education positively affects the probability of transition from the first to the second category (being employed after the first interview). In terms of odds ratio, we have

```
round(exp(mod1$Ga[2,1]), 3)
```

```
#> [1] 8.877
```

compared to individuals with low educated parents. Moreover, having highly educated parents has a negative effect on the transition from the second to the first category:

```
round(exp(mod1$Ga[2,2]),3)

#> [1] 0.072
```

## 3 Hidden markov models for categorical data

We outline the main notation and assumptions of the HM models useful to understand the output of the illustrated estimation functions of the **LMest** package.

### 3.1 Model assumptions

For a sample of $n$ individuals and $T_i$ time occasions, let $\boldsymbol{Y}_{it}$, $i = 1, \ldots, n$, $t = 1, \ldots, T_i$, be the observable vector of response variables with elements $Y_{ijt}$, $j = 1, \ldots, r$, where $r$ denotes the number of response variables. Note that the number of time occasions $T_i$ is specific for each individual $i$. In this way, we allow for unbalanced panels that may be due to a different number of time occasions for every individual. Let also $\boldsymbol{U}_i = (U_{i1}, \ldots, U_{iT_i})'$ denote the latent process for each individual $i$ that affects the distribution of the response variables and assumed to follow a first-order Markov chain with a certain number of latent (or hidden) states equal to $k$. The Markov properties of $U_{it}$ are the same as those illustrated in Section 2.1. However, here we use the notation $U_{it}$, which stands for "unobserved", to emphasize that it is latent/hidden, whereas $Y_{it}$ in Section 2.1 is observable. The response variables are conditionally independent given this latent process; this assumption is referred to as *local independence*. It means that the latent process fully explains the underlying phenomenon together with possible covariates that may be time-fixed or time-varying.

The HM model is characterized by two components: the *measurement (sub)model*, which describes the conditional distribution of the response variables given the latent process, and the *structural or latent (sub)model*, which describes the distribution of the latent process. When dealing with categorical outcomes, the formulation of the measurement (sub)model without covariates is based on the parameters

$$\phi_{jy|u} = p(Y_{ijt} = y | U_{it} = u), \quad j = 1, \ldots, r, \; t = 1, \ldots, T_i, \; u = 1, \ldots, k, \; y = 0, \ldots, l_j - 1, \tag{3}$$

where $l_j$ corresponds to the number of response categories of the $j$-variable in $\boldsymbol{Y}_{it}$.

Regarding the latent (sub)model, the typical assumption is that the latent Markov chain is of first order and, for the initial and transition probabilities, we can use the same notation previously adopted for the MC model that now is formulated with reference to the latent variables rather than the observable variables. More precisely, we introduce the initial and transition probabilities

$$\begin{aligned} \pi_u &= p(U_{i1} = u), \quad u = 1, \ldots, k, \\ \pi_{uv} &= p(U_{it} = v | U_{i,t-1} = u), \quad t = 2, \ldots, T_i, \; u, v = 1, \ldots, k. \end{aligned}$$

Note that, in the basic version of the HM model, the transition probabilities are assumed to be time homogeneous to reduce the number of free parameters. However, this assumption can be relaxed if it proves to be too restrictive.

When available, individual explanatory variables, collected in the vectors $\boldsymbol{x}_{it}$, may be included in the measurement model, so as to account for the unobserved heterogeneity between units, or in the latent model, so that they affect the initial and transition probabilities of the Markov chain. In the latter case, these probabilities may be defined on the basis of the same multinomial logit parameterizations adopted for the MC model; see in particular Equations (1) and (2). The parameterization of the transition probabilities in Equation (2) is referred to as `paramLatent = "multilogit"` in the corresponding argument of the estimation function `lmest()` and is illustrated with an application to health related data in Bartolucci et al. (2017).

In order to make the model more parsimonious, an alternative differential logit parameterization can be used for the transition probabilities, denoted as `paramLatent = "difflogit"`. Such a parameterization relies on logits based on the difference between two vectors of parameters, for $t = 2, \ldots, T_i$:

$$\log \frac{\pi_{it,uv}}{\pi_{it,uu}} = \gamma_{0uv} + \boldsymbol{x}_{it}'(\gamma_{1v} - \gamma_{1u}), \quad u, v = 1, \ldots, k, \; u \neq v, \tag{4}$$

where $\gamma_{11} = \boldsymbol{0}$ to ensure model identifiability; see Bartolucci et al. (2015) for an example of application of this parameterization.

The **LMest** package also allows including individual covariates in the measurement model through a suitable parameterization of the conditional distribution of the response variables given the latent states; see Bartolucci et al. (2017). This formulation can only be used for univariate data by setting

argument `modManifest = "LM"` in the `lmest()` function. It is also possible to indicate an alternative model specification by setting the option `modManifest = FM`, which relies on the assumption that the latent process has a distribution given by a mixture of AR(1) processes with common variance and specific correlation coefficients. See Bartolucci et al. (2014a) for an illustration of this model; see also Pennoni and Vittadini (2013).

### 3.2 Maximum likelihood inference

We illustrate maximum likelihood estimation in the general case in which covariates are available and are included in the distribution of the latent process. In this case, for a sample of $n$ independent units, the model log-likelihood has the following expression:

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} f(\boldsymbol{y}_{i1}, \ldots, \boldsymbol{y}_{iT_i} | \boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{iT_i}),$$

where $\boldsymbol{\theta}$ is the vector of all free model parameters and $f(\boldsymbol{y}_{i1}, \ldots, \boldsymbol{y}_{iT_i} | \boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{iT_i})$ corresponds to the manifest distribution, that is, the probability of the responses provided by subject $i$ given the covariates. This manifest probability has expression

$$f(\boldsymbol{y}_{i1}, \ldots, \boldsymbol{y}_{iT_i} | \boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{iT_i}) = \sum_{u_1=1}^{k} \cdots \sum_{u_{T_i}=1}^{k} \pi_{i,u_1} \prod_{t=2}^{T_i} \pi_{it,u_{t-1}u_t} \prod_{t=1}^{T_i} f(\boldsymbol{y}_{it} | u_t), \tag{5}$$

where $f(\boldsymbol{y}_{it}|u)$ refers to the conditional distribution of the response variables for unit $i$ at time occasion $t$ given the latent process that, in the case of categorical response variables, is freely parameterized by the conditional response probabilities in Equation (3).

In order to compute $f(\boldsymbol{y}_{i1}, \ldots, \boldsymbol{y}_{iT_i} | \boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{iT_i})$ while avoiding the sum in Equation (5), which has a computational cost that exponentially increases in $T_i$, we rely on the Baum and Welch recursion (Baum and Petrie, 1966) that is described in Bartolucci et al. (2013), among others. The above log-likelihood function can be maximized by the EM algorithm based on the complete-data log-likelihood (Baum et al., 1970; Dempster et al., 1977). The EM algorithm is characterized by a series of iterations consisting of two steps, named E- and M-step, which are repeated until convergence. The E-step computes the conditional expected value of the complete-data log-likelihood given the current value of the parameters and the observed data. The M-step consists in maximizing this expected value so as to update the model parameters. The convergence of the EM algorithm is assessed on the basis of a suitable convergence criterion relying on the relative log-likelihood difference between two consecutive iterations. From this iterative algorithm we obtain an estimate of $\boldsymbol{\theta}$, denoted by $\hat{\boldsymbol{\theta}}$.

For HM models, initialization of the estimation algorithm is an important issue as the model log-likelihood is typically multimodal. The **LMest** package allows performing a multi-start initialization strategy, based on both deterministic and random rules, which is aimed at suitably exploring the parameter space so as to increase the chance to reach the global maximum of the model log-likelihood. With some differences, this is done by functions `lmest()` and `lmestSearch()`; the latter will be illustrated in the following.

Once the parameter estimates are computed, standard errors may be obtained in the usual way on the basis of the observed information matrix. This matrix is provided by **LMest** using either the exact computation method proposed in Bartolucci and Farcomeni (2015) or the numerical method of Bartolucci and Farcomeni (2009), depending on the complexity of the model of interest. The package also provides the `bootstrap()` method to obtain standard errors by a parametric bootstrap procedure, that is, by drawing samples from the estimated model and computing the maximum likelihood estimates for every bootstrap sample. The standard errors are obtained by computing, in a suitable way, the standard deviation of the empirical distribution so obtained; see among others, Visser and Speekenbrink (2022). An alternative nonparametric bootstrap procedure can also be used, based on drawing a large number of samples with replacement from the observed data.

Selection of the number of states is performed according to information criteria such as AIC (Akaike, 1973) and BIC (Schwarz, 1978). They are defined as follows:

$$\begin{aligned} AIC &= -2\ell(\hat{\boldsymbol{\theta}}) + 2\,\#par, \\ BIC &= -2\ell(\hat{\boldsymbol{\theta}}) + \log(n)\,\#par, \end{aligned}$$

where $\ell(\hat{\boldsymbol{\theta}})$ denotes the maximized log-likelihood of the model of interest, $n$ is the sample size, and $\#par$ denotes the number of free parameters to be estimated. Other criteria, such as those based on entropy may be used; see, among others, Bacci et al. (2014).

Once the number of states is selected, dynamic clustering is performed by assigning every unit to

a latent state at each time occasion by means of the estimated posterior probabilities of the $U_{it}$. These probabilities are directly provided by the EM algorithm and are defined as

$$\hat{a}_{it,u} = p(U_{it} = u|\boldsymbol{y}_{i1}, \ldots, \boldsymbol{y}_{iT_i}, \boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{iT_i}),\ t = 1, \ldots, T_i,\ u = 1, \ldots, k, \tag{6}$$

$$\hat{b}_{it,uv} = p(U_{i,t-1} = u, U_{it} = v|\boldsymbol{y}_{i1}, \ldots, \boldsymbol{y}_{iT_i}, \boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{iT_i}),\ t = 2, \ldots, T_i,\ u, v = 1, \ldots, k.$$

Prediction of the latent states of every unit $i$ at each time occasion $t$ is known as *local decoding* and it is obtained as the value of $u$ that maximizes the posterior probabilities in Equation (6). As an alternative to the local decoding, the *global decoding* may be performed, which is based on the Viterbi algorithm (Viterbi, 1967; Juang and Rabiner, 1991) to obtain the prediction of the latent trajectories of a unit across time, that is, the *a posteriori* most likely sequence of hidden states. The package provides the `lmestDecoding()` method to perform both local and global decoding based on the different model specifications, as illustrated in the following sections. Additional details on the model formulations, backward and forward recursions, and estimation can be found in Bartolucci et al. (2013).

### 3.3 Application to longitudinal data with covariates

As an illustration of the **LMest** package for estimation of HM models with multivariate responses and covariates, we consider simulated data provided in the package, named `data_market_sim`. These data refer to a hypothetical marketing context where a sample of $n = 200$ customers of four different brands is observed along with the price of each transaction, defined for $T = 5$ occasions, in Euros per purchase within the following ranges: [0.1, 10], (10, 30], (30, 60], (30, 100], (100, 500]. For a similar application see, among others, Paas et al. (2007); Bassi et al. (2021). Accordingly, we consider $r = 2$ response variables (items), the first variable with $l_1 = 4$ categories, one for each brand, and the second variable with $l_2 = 5$ categories, one for each range of price. The initial part of the dataset, in long format, is displayed below:

```
data("data_market_sim")
head(data_market_sim)
```

```
#>   id time brand price age income
#> 1  1    1     2     2  34     25
#> 2  1    2     0     2  35     25
#> 3  1    3     1     0  36     25
#> 4  1    4     2     2  37     25
#> 5  1    5     3     1  38     25
#> 6  2    1     1     3  35     27
```

The research questions in this context concern the evolution of customer's purchase behavior over time by exploring market segments, while also considering the role of socio-demographic characteristics defined by the available individual covariates. In particular, we include age, as time-varying continuous covariate, and income of the customer at the time of the first purchase, as time-fixed continuous covariate.

As already mentioned, the package provides function `lmestSearch()`, which searches for the global maximum of the log-likelihood of different models and selects the optimal number of hidden states. This function combines deterministic and random initializations with a rather wide tolerance level. Moreover, starting from the best solution obtained from these preliminary runs, a final run is performed, with a smaller tolerance level, in order to increase the chance of reaching the global maximum. The argument `nrep` can be provided to set the number of random initializations for each number of hidden states. The range of states to be considered may be specified in argument `k` as a vector of integers. Generally, model selection is performed by estimating the multivariate HM model without covariates, which is specified within the model formula, indicated in the argument `responsesFormula`, with `~ NULL` stating that there are no predictors for the two response variables as follows:

```
hmm <- lmestSearch(responsesFormula = brand + price ~ NULL,
                   latentFormula =  ~ NULL,
                   version = "categorical",
                   index = c("id", "time"),
                   data = data_market_sim,
                   k = 1:4, fort = TRUE,
                   seed = 12345)
```

By adding the optional `fort = TRUE` argument, a faster estimation procedure is achieved using Fortran routines. The `summary()` method returns the estimation results, displaying the AIC and BIC values for the sequence of estimated hidden states:

```
summary(hmm)
```

```
#> Call:
#> lmestSearch(responsesFormula = brand + price ~ NULL, latentFormula = ~NULL,
#>     data = data_market_sim, index = c("id", "time"), k = 1:4,
#>     version = "categorical", seed = 12345, fort = TRUE)
#>
#>  states        lk np      AIC      BIC
#>       1 -2811.001  7 5636.003 5659.091
#>       2 -2520.131 17 5074.263 5130.334
#>       3 -2445.538 29 4949.075 5044.727
#>       4 -2434.262 43 4954.524 5096.352
```

In this case, the BIC, used by default for model selection, supports a model with three hidden states. Once the model is selected, we estimate the parameters of the latent model with covariates by fixing the parameter values of the conditional response probabilities obtained under the model chosen above. In this way, the estimated subpopulations are held fixed. These conditional response probabilities are displayed below:

```
Psi <- hmm$out.single[[3]]$Psi
print(round(Psi, 2))
```

```
#> , , item = 1
#>
#>          state
#> category    1    2    3
#>        0 0.69 0.08 0.10
#>        1 0.10 0.45 0.12
#>        2 0.17 0.40 0.08
#>        3 0.05 0.07 0.70
#>        4   NA   NA   NA
#>
#> , , item = 2
#>
#>          state
#> category    1    2    3
#>        0 0.30 0.04 0.00
#>        1 0.43 0.19 0.05
#>        2 0.10 0.64 0.05
#>        3 0.12 0.11 0.30
#>        4 0.04 0.02 0.60
```

From these results, we notice that the three states may represent distinct customer segments that can be labeled as follows: low-cost market segment (1st), ordinary segment (2nd), and luxury segment (3rd). These segments are described in more detail in the following.

Function `lmest()` estimates the HM model with covariates that can affect the latent structure in various ways. In the following, we assume that age and income may influence only the transition probabilities, and do not affect the composition of the market segments at the beginning of the survey. This can be set through the `latentFormula` argument, which includes `age + income` in the parameterization of the transition probabilities. For the initial probabilities, we ignore the effects of covariates and estimate only the intercept of the multinomial logit by including `NULL` in the formula before the symbol "`|`". Note that this formulation is the same as the one expressed in Equation (1) but without the vector of covariates, whereas for the transition probabilities we consider the parametrization defined in Equation (4) obtained by including the argument `paramLatent = "difflogit"`. Moreover, as an input to the `lmest()` function, we also use the optional arguments `parInit`, which is a list of initial model parameters that also includes argument `fixPsi = TRUE` to avoid estimation of the conditional response probabilities. In this way, the EM algorithm is performed to estimate the parameters of the structural model while these probabilities are kept fixed at the value displayed above. The same arguments can be used to require the estimation of an MC model for unbalanced data as a constrained version of an HM model with a number of states equal to the number of response categories.

The HM model with three hidden states and fixed conditional response probabilities, chosen as described above, is estimated as follows:

```
mod2 <- lmest(responsesFormula = brand + price ~ NULL,
              latentFormula =  ~ NULL | age + income,
              k = 3, data = data_market_sim,
              index = c("id", "time"),
              parInit = list(Psi = Psi, fixPsi = TRUE),
              paramLatent = "difflogit",
              seed = 12345)
```

The `print()` method provides an overview of the estimation results including the maximum log-likelihood, number of free parameters, and AIC and BIC values that can be used for model selection.

```
print(mod2)
```

```
#>
#> Basic Latent Markov model with covariates in the latent model
#> Call:
#> lmest(responsesFormula = brand + price ~ NULL, latentFormula = ~NULL |
#>     age + income, data = data_market_sim, index = c("id", "time"),
#>     k = 3, paramLatent = "difflogit", parInit = list(Psi = Psi,
#>         fixPsi = TRUE), seed = 12345)
#>
#> Available objects:
#>  [1] "lk"          "Be"          "Ga"          "Psi"         "Piv"
#>  [6] "PI"          "np"          "k"           "aic"         "bic"
#> [11] "lkv"         "n"           "TT"          "paramLatent" "ns"
#> [16] "yv"          "Lk"          "Bic"         "Aic"         "call"
#> [21] "data"
#>
#> Convergence info:
#>      LogLik np k      AIC      BIC   n TT
#>   -2444.437 12 3 4912.874 4952.454 200  5
```

The estimated HM model has a maximum log-likelihood of -2,444.437 with 12 parameters so that AIC and BIC indexes are equal to 4,912.874 and 4,952.454, respectively.

The `plot()` method associated with `lmest()` provides a variety of displays. For example, a plot of the estimated conditional response probabilities can be obtained as:

```
par(mar = c(5,4,4,2) + 0.1)
plot(mod2, what = "CondProb")
```

as shown in Figure 1. According to the simulated context, from this figure we observe that the 1st state (low-cost segment), which includes the highest frequency of customers at the first time occasion (39%), is related to those who primarily purchase products of the first brand (category 0 of item 1) with low prices (categories 0 and 1 of item 2). Customers in the 2nd state (ordinary segment), corresponding to around 30% of all customers, tend to buy products of the second and third brands with medium prices. On the other hand, customers in the 3rd state (luxury segment) tend to purchase products of brand four (category 3 of item 1) with relatively high prices (categories 3 and 4 of item 2). This state includes around 31% of customers at the beginning of the period of observation.

The averaged estimated transition probabilities may be obtained by the following command:

```
print(round(apply(mod2$PI[,,2:5], c(1,2), mean), 3))
```

```
#>      state
#> state     1     2     3
#>     1 0.092 0.486 0.422
#>     2 0.028 0.878 0.093
#>     3 0.000 0.106 0.894
```

A plot showing the corresponding path diagram of the estimated transition matrix can be obtained as

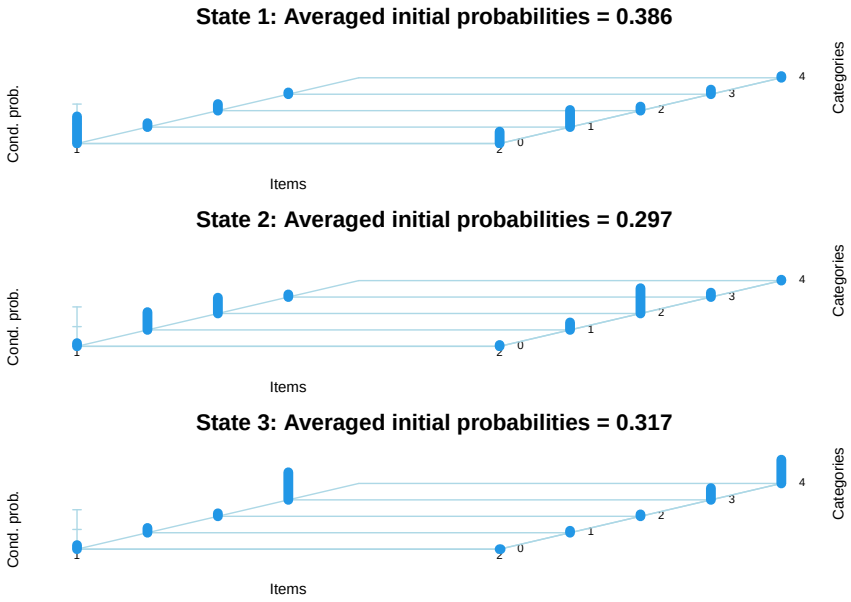



**Figure 1:** Plot of the estimated conditional response probabilities: on the left side there is item 1 (brand), and on the right side item 2 (price), respectively. The top, middle, and bottom panels correspond to the estimates for the 1st state (low-cost segment), 2nd state (ordinary segment), and 3rd state (luxury segment), respectively.

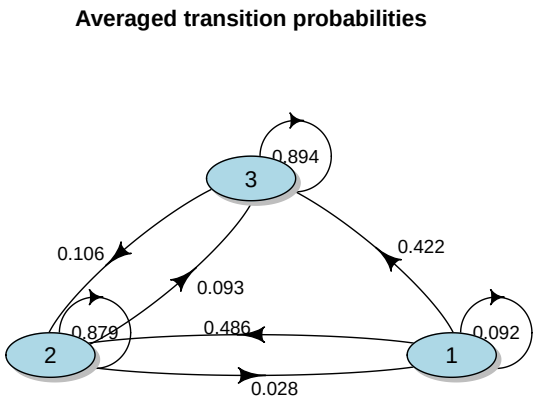



**Figure 2:** Path diagram of averaged estimated transition probabilities: nodes represent hidden states, and arrows are present when the estimated transition probability between two states is not null. Self-arrows indicate persistence in the same state if estimated. The reported numbers refer to the estimated values of the transition probability matrix.

```
par(mar = c(2,1,5,1))
plot(mod2, what = "transitions")
```

as illustrated in Figure 2. From the results shown in this figure, we observe that customers have a fairly high probability of persistence in the 2nd and 3rd state from one time period to the next, as the estimated diagonal elements of the transition probability matrix are fairly high (0.88 and 0.89, respectively). This suggests that these two market segments are quite stable across customers over years. Conversely, the highest estimated probabilities outside of the main diagonal refer to the transition from the 1st state (low-cost segment) to the 2nd state (ordinary) and from the 1st to the 3rd (luxury). This shows that customers of the 1st segment have a greater tendency to change their behavior over time.

The `summary()` method returns the main estimation results:

```
summary(mod2)

#> Call:
#> lmest(responsesFormula = brand + price ~ NULL, latentFormula = ~NULL |
#>     age + income, data = data_market_sim, index = c("id", "time"),
```

```
#>     k = 3, paramLatent = "difflogit", parInit = list(Psi = Psi,
#>         fixPsi = TRUE), seed = 12345)
#>
#> Coefficients:
#>
#>  Be - Parameters affecting the logit for the initial probabilities:
#>              logit
#>                    2      3
#>   (Intercept) -0.2604 -0.196
#>
#>  Ga0 - Intercept affecting the logit for the transition probabilities:
#>            logit
#> (Intercept)       2       3
#>           1 -3.2126 -3.9261
#>           2  1.4353 -2.8156
#>           3 -5.4099 -1.5630
#>
#>  Ga1 - Regression parameters affecting the logit for the transition probabilities:
#>         logit
#>               2      3
#>   age    0.0836 0.0897
#>   income 0.0563 0.0674
#>
#>  Psi - Conditional response probabilities:
#> , , item = 1
#>
#>         state
#> category      1      2      3
#>        0 0.6875 0.0816 0.1031
#>        1 0.1000 0.4512 0.1216
#>        2 0.1674 0.4010 0.0759
#>        3 0.0450 0.0663 0.6993
#>        4     NA     NA     NA
#>
#> , , item = 2
#>
#>         state
#> category      1      2      3
#>        0 0.3016 0.0353 0.0000
#>        1 0.4344 0.1895 0.0505
#>        2 0.1033 0.6407 0.0489
#>        3 0.1172 0.1113 0.2981
#>        4 0.0436 0.0232 0.6025
```

The output `Ga` contains the estimated parameters affecting the distribution of the transition probabilities based on the difflogit parameterization. More in detail, `Ga0` refers to the intercepts, while `Ga1` refers to the regression coefficients. Each column of `Ga1` can be interpreted as a general measure of attraction of the corresponding state. From these results, we observe that the estimated effects of the age and income of the customer are relatively small for each logit. Both covariates have a positive impact, indicating that as age or income increases, the probability of moving to more valuable segments also increases, while holding other parameters constant. Thus, older customers or those with higher income are more likely to move to higher-value segments.

## 4 Hidden markov models for continuous data assuming a conditional gaussian distribution

In this section, we illustrate the main notation used for HM models for continuous outcomes and discuss how to handle missing responses under the MAR assumption (Little and Rubin, 2020). We also illustrate the inclusion of covariates in both the latent and measurement models. Finally, we introduce three examples of the application of these models using the appropriate functions of the **LMest** package.

### 4.1 Model assumptions and inference

When dealing with continuous outcomes, let $\boldsymbol{Y}_{it} = (Y_{i1t}, \ldots, Y_{irt})'$ denote the vector of $r$ continuous response variables measured at time $t$ for subject $i$. As usual, under the local independence assumption, the response vectors $\boldsymbol{Y}_{i1}, \ldots, \boldsymbol{Y}_{iT_i}$ are assumed to be conditionally independent given the latent process $\boldsymbol{U}_i$. Moreover, for the measurement model, we assume a conditional Gaussian distribution, that is,

$$\boldsymbol{Y}_{it}|U_{it} = u \sim N(\boldsymbol{\mu}_u, \boldsymbol{\Sigma}_u), \quad u = 1, \ldots, k. \tag{7}$$

The parameters of the measurement (sub)model are the conditional means $\boldsymbol{\mu}_u$, $u = 1, \ldots, k$, which are state specific, and the variance-covariance matrices $\boldsymbol{\Sigma}_u$, which may be assumed to be constant across states under the assumption of homoschedasticity, that is, $\boldsymbol{\Sigma}_1 = \cdots = \boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$. This assumption, which reduces the number of estimated parameters and may be reasonable in many applied contexts, is adopted in the **LMest** package.

The parameters of the latent (sub)model are again the initial and transition probabilities of the Markov chain, as specified in the previous sections. As usual, when individual covariates collected in vectors $\mathbf{x}_{it}$ are available, they may be included in the distribution of the latent variables, so as to affect the initial and transition probabilities. It is also possible to assume that individual covariates affect the distribution of the response variables given the latent process, thereby accounting for unobserved heterogeneity. The latter formulation is based on the assumption that

$$E(Y_{it}|U_{it} = u, \boldsymbol{x}_{it}) = \alpha_u + \boldsymbol{x}_{it}'\boldsymbol{\beta}, \quad u = 1, \ldots, k,$$

which naturally extends to the multivariate case.

The manifest distribution may be expressed with a formula that closely recalls the one used in Equation (5) for HM models for categorical responses, but with $f(\boldsymbol{y}_{it}|u)$ that here denotes the density of the multivariate Gaussian distribution based on assumption in Equation (7), possibly dependent on the covariates. Maximum likelihood estimation is carried out as illustrated in Section 3.2 through the EM algorithm and its extended version when individual covariates are included in the model. The **LMest** package can also handle missing values in the response variables. When the outcomes are continuous, we consider two different types of missing pattern under the MAR assumption: partially missing outcomes at a given time occasion and completely missing outcomes, that is, when individuals do not respond at one or more time occasions (intermittent pattern). Following Pandolfi et al. (2023), in the presence of missing responses, it is convenient to partition each response vector $\boldsymbol{Y}_{it}$ as $(\boldsymbol{Y}_{it}^o, \boldsymbol{Y}_{it}^m)'$, where $\boldsymbol{Y}_{it}^o$ is the (sub)-vector of observed variables, and $\boldsymbol{Y}_{it}^m$ refers to the missing data. The conditional mean vectors and variance-covariance matrix may be decomposed into observed and missing components as follows:

$$\boldsymbol{\mu}_u = \begin{pmatrix} \boldsymbol{\mu}_u^o \\ \boldsymbol{\mu}_u^m \end{pmatrix}, \quad \boldsymbol{\Sigma}_u = \begin{pmatrix} \boldsymbol{\Sigma}_u^{oo} & \boldsymbol{\Sigma}_u^{om} \\ \boldsymbol{\Sigma}_u^{mo} & \boldsymbol{\Sigma}_u^{mm} \end{pmatrix},$$

where, for instance, $\boldsymbol{\Sigma}_u^{om}$ is the block of $\boldsymbol{\Sigma}_u$ collecting covariances between each observed and missing response. In this way, for the observed responses, we have that:

$$\boldsymbol{Y}_{it}^o|U_{it} = u \sim N(\boldsymbol{\mu}_u^o, \boldsymbol{\Sigma}_u^{oo}), \quad u = 1, \ldots, k.$$

The manifest distribution of the responses is now expressed with reference to the observed data. Model inference is carried out using an extended version of the EM algorithm based on suitable recursions, as illustrated in Pandolfi et al. (2023). In particular, at the E-step the algorithm also computes additional expected values arising from the MAR assumption for the missing observations. Under this model formulation, it is also possible to perform a type of multiple imputation, which allows us to predict the missing responses either conditionally or unconditionally on the predicted states. Within the **LMest** package, when the missing data are dealt with under the MAR assumption, the unconditional prediction of the missing responses is performed by computing

$$\tilde{\boldsymbol{y}}_{it} = \sum_{u=1}^{k} \hat{a}_{it,u} E(\boldsymbol{Y}_{it}|\boldsymbol{y}_{it}^o, u),$$

where $\hat{a}_{it,u} = p(U_{it}|\boldsymbol{y}_{i1}^o, \ldots, \boldsymbol{y}_{iT_i}^o, \boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{iT_i})$ and $E(\boldsymbol{Y}_{it}|\boldsymbol{y}_{it}^o, u)$ are posterior expected values computed at the E-step.

Finally, both local and global decoding may be performed as usual. In this context, it is important to note that the prediction of the latent states is also carried out for units with missing responses.

### 4.2 Application to time-series

This section illustrates an application of the HM model to multivariate time-series data; for more details, see Pennoni et al. (2022). Specifically, we use real data from Yahoo Finance, covering S&P 500 Index (SP500TR) and Intel stock prices (INTC). These data are sourced from the R package **quantmod** (Ryan and Ulrich, 2022). The dataset includes the closing prices for each trading day from March to August 2022. For the following application, we then compute the percentage returns based on these closing prices.

The analysis of financial time-series data is typically focused on identifying market phases associated with the volatility of returns. In this context, latent states are referred to as regimes, and it is valuable to detect any abrupt and persistent changes in these regimes.

```
require(quantmod)
SP500_22 <- getSymbols("^SP500TR",
                       env = NULL,
                       from = "2022-03-01",
                       to = "2022-08-31",
                       periodicity = "daily")
dim(SP500_22)

#> [1] 127   6

INCT_22 <- getSymbols("INTC",
                      env = NULL,
                      from = "2022-03-01",
                      to = "2022-08-31",
                      periodicity = "daily")

sp_sreturns <- dailyReturn(SP500_22)*100
intc_sreturns <- dailyReturn(INCT_22)*100

data_fin <- cbind(1, 1:length(sp_sreturns), sp_sreturns, intc_sreturns)

names(data_fin) <- c("id", "time",  "SP", "INCT")
head(round(data_fin, 3))

#>            id time     SP   INCT
#> 2022-03-01  1    1 -1.304 -1.515
#> 2022-03-02  1    2  1.868  4.378
#> 2022-03-03  1    3 -0.513 -1.923
#> 2022-03-04  1    4 -0.786  0.292
#> 2022-03-07  1    5 -2.951 -0.811
#> 2022-03-08  1    6 -0.721 -0.378

data_fin[which.min(data_fin$SP),]

#>            id time        SP      INCT
#> 2022-05-18  1   56 -4.016448 -4.617124

data_fin[which.min(data_fin$INCT),]

#>            id time       SP      INCT
#> 2022-07-29  1  105 1.431582 -8.562069
```

The two series are displayed in Figure 3 and 4, while descriptive statistics are reported in the following:

```
#>      Index                  SP                INCT
#>  Min.   :2022-03-01   Min.   :-4.01645   Min.   :-8.5621
#>  1st Qu.:2022-04-13   1st Qu.:-0.89097   1st Qu.:-1.8511
#>  Median :2022-05-31   Median :-0.06888   Median :-0.1265
#>  Mean   :2022-05-30   Mean   :-0.05265   Mean   :-0.2769
#>  3rd Qu.:2022-07-16   3rd Qu.: 1.09352   3rd Qu.: 1.1456
#>  Max.   :2022-08-30   Max.   : 3.05815   Max.   : 6.9401
```

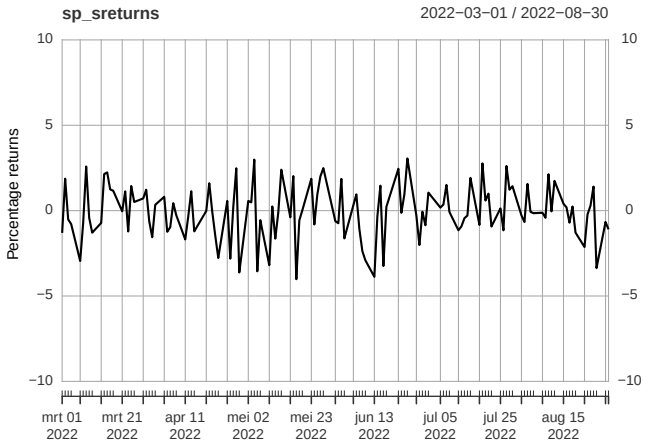



**Figure 3:** Observed percentage returns of Standard and Poor's 500 Index from March to August 2022 (127 days).

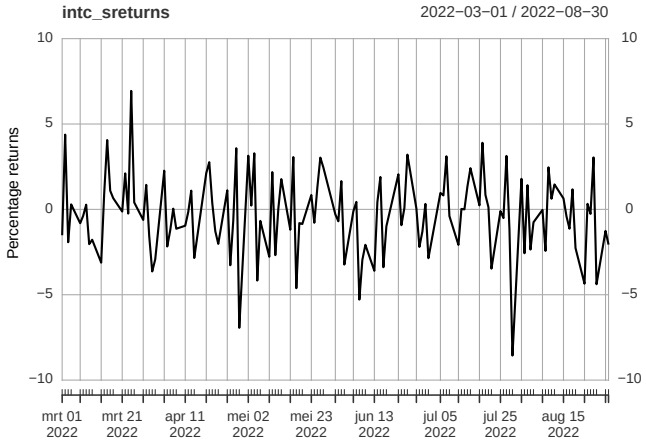



**Figure 4:** Observed percentage returns of Intel stock from March to August 2022 (127 days).

We can easily estimate the HM model for continuous data with three hidden states, assuming they may represent bear, bull, and intermediate market regimes. The function of the package that can be used to estimate this model is `lmestCont()`; the basic version of the model, without covariates, can be specified by setting the argument `responsesFormula = SP + INCT ~ NULL`.

As for the other estimation functions, the argument `index` is required to specify the name of the unit identifier and the indicator of the time occasions. The model can be fitted with the constraint of time-homogeneity, by setting the argument `modBasic = 1`, which ensures that the transition probabilities do not depend on $t$. Argument `tol = 10^-6` is used to set the tolerance level for the convergence of the algorithm; by default, this is set to `tol = 10^-10`. We can also specify `maxit`, which is an integer that sets the maximum number of EM iterations; by default, this is set to 5,000.

```
mod3 <- lmestCont(responsesFormula = SP + INCT ~ NULL,
                  data = data_fin,
                  index = c("id", "time"),
                  k = 3, modBasic = 1, tol = 10^-6)
```

A summary of the estimation results is obtained with the `summary()` method:

```
summary(mod3)
```

```
#> Call:
#> lmestCont(responsesFormula = SP + INCT ~ NULL, data = data_fin,
#>     index = c("id", "time"), k = 3, modBasic = 1, tol = 10^-6)
#>
#> Coefficients:
#>
#> Initial probabilities:
#>      est_piv
#> [1,]       1
#> [2,]       0
#> [3,]       0
#>
#> Transition probabilities:
#>      state
#> state      1      2      3
#>     1 0.1265 0.2715 0.6020
#>     2 0.0389 0.9610 0.0001
#>     3 0.5113 0.0001 0.4887
#>
#>  Mu - Conditional response means:
#>       state
#> item         1       2      3
#>   SP   -2.5609  0.2353 0.6531
#>   INCT -2.9240 -0.0996 1.1140
#>
#>  Si - Variance-covariance matrix:
#>        [,1]   [,2]
#> [1,] 1.5253 1.7211
#> [2,] 1.7211 4.3718
```

This summary output shows the estimated initial and transition probabilities of the three latent states, the estimated cluster means, and the variance-covariance matrix. The three states represent different market regimes that can be interpreted according to the estimated means as follows: the 1st state identifies negative returns, the 2nd state corresponds to positive returns for S&P 500 (SP) and negative for Intel stock (INCT), and the 3rd state identifies positive returns for both. The main transitions are from the 1st to the 2nd state (0.272), from the 1st to the 3rd (0.602) state, and from the 3rd to the 1st state (0.511).

The contour plot of the estimated overall density, with weights given by the estimated marginal probabilities of the latent states, is shown in Figure 5. It is obtained by the `plot()` method using the argument `what = "density"` as follows:

```
plot(mod3, what = "density")
```

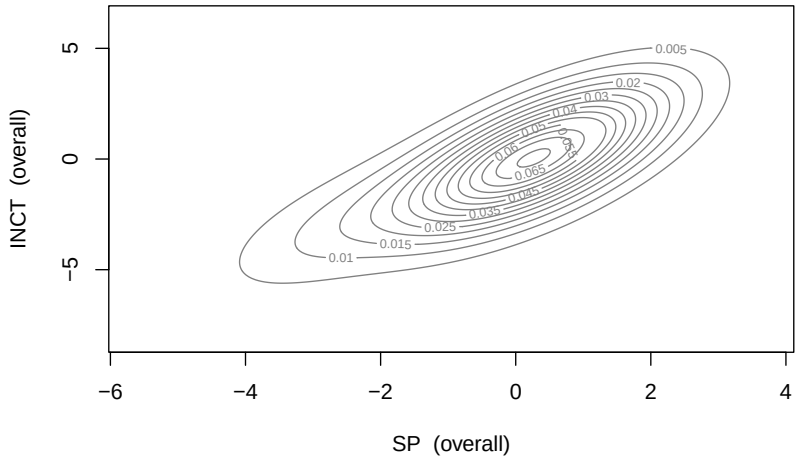



**Figure 5:** Estimated density surface of the bivariate distribution of Standard and Poor's 500 Index and Intel stock prices observed from March to August 2022 (127 days)

The density regions of each estimated component (regime), represented as a contour plot, can be visualized by setting the argument `components = c(1, 2, 3)`. The resulting plot is shown in Figure 6.

```
par(mar = c(3,4,2,2), oma = c(0,1,0,1))
plot(mod3, what = "density", components = c(1,2,3))
```

In the context of financial time-series, a relevant issue is path prediction, that is, finding the most likely sequence of latent states for a given unit on the basis of the observed values. As already mentioned, both local and global decoding are implemented in the `lmestDecoding()` method, which allows us to predict the sequence of latent states for a sample unit on the basis of the output of the different estimation functions, as follows:

```
deco3 <- lmestDecoding(mod3)
```

Object `deco3` contains the local (object `Ul`) and global (object `Ug`) decoding (obtained through the Viterbi algorithm). For example, for the local decoding, we have:

```
deco3$Ul
```

```
#>   [1] 1 3 3 3 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
#>  [38] 1 3 1 3 3 1 3 3 3 1 3 1 3 1 3 3 3 3 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 3
#>  [75] 3 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
#> [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2
```

These sequences can be depicted as follows:

```
data_fin$Ul <- deco3$Ul
plot(data_fin$Ul, ylab="State")
```

From Figure 7 we can observe distinct periods characterized by different market regimes, as indicated by the decoded state sequence. Initially, a neutral market regime is observed, followed by alternating bullish and bearish market regimes, and finally, a neutral market regime predominates during the last period.

### 4.3 Application to longitudinal data with missing values and covariates in the latent (sub)models

Data provided in the package, named `data_heart_sim`, are simulated from a hypothetical observational retrospective study designed to assess the progression of health states of individuals after treatment. The dataset includes observations on systolic and diastolic blood pressure (variables `sap` and `dap` in mmHg) and heart rate (variable `hr` in bpm), as well as covariates such as fluid administration (variable `fluid` in ml/kg/h), gender (variable `gender`: 1 for male, 2 for female), and age (variable `age` in years). Note that `fluid` is a time-varying covariate. The initial part of the dataset is as follow:

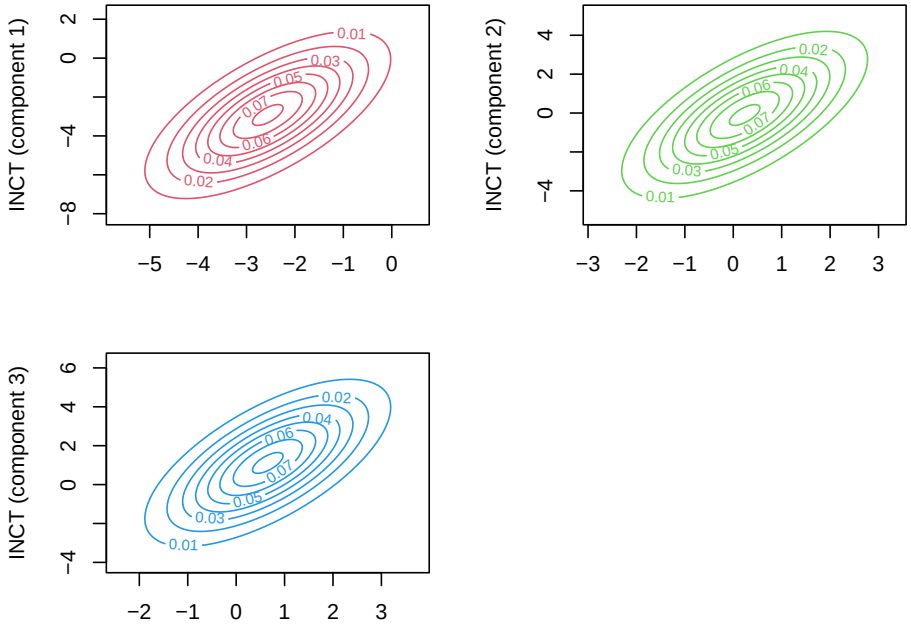



**Figure 6:** Estimated density surfaces for each hidden state (component) represented as contour levels of Standard and Poor's 500 Index and Intel stock prices observed from March to August 2022 (127 days).

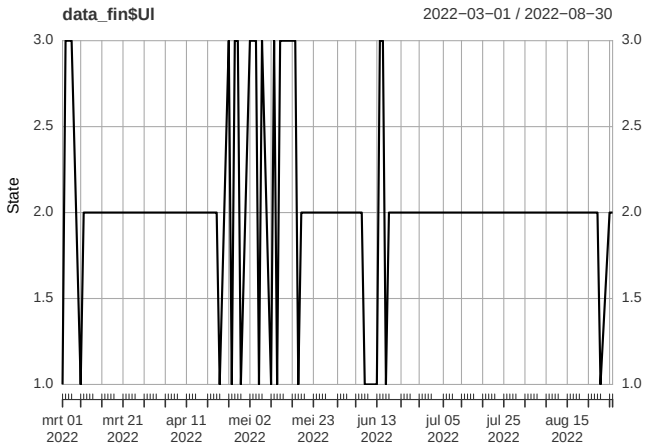



**Figure 7:** Predicted sequence of hidden states from March to August 2022 (127 days) under the HM model with $k = 3$ estimated for Standard and Poor's 500 Index and Intel stock prices.

```
data("data_heart_sim")
head(data_heart_sim)

#>   id time sap dap  hr fluid gender age
#> 1  1    1 117  67  75  2800      1  56
#> 2  1    2 111  69  93  1750      1  56
#> 3  1    3 102  60 108  2320      1  56
#> 4  1    4 123  78 105  2600      1  56
#> 5  1    5 102  57  99  1700      1  56
#> 6  1    6  86  60  96  2210      1  56
```

We assume that these measurements are recorded daily after a particular surgery for up to six days, so that $T = 6$, and some patients have missing records for one or more outcomes. The number of patients is $n = 125$. Missing responses are denoted with `NA` in the data provided to the estimation function. Note that the package does not support missing values in the covariates, which should be imputed separately. The data are simulated to include a proportion of missing values of around 15%.

We estimate an HM model with three hidden states, including covariates that affect both the initial and transition probabilities, and where missing data are handled under the MAR assumption. This model is specified using the full set of responses through the function `lmestCont()` as follows:

```
mod4 <- lmestCont(responsesFormula = sap + dap + hr  ~ NULL,
                  latentFormula = ~ fluid + as.factor(gender) + age,
                  data = data_heart_sim,
                  index = c("id", "time"),
                  k = 3, miss.imp = FALSE)
```

Note that it is possible to specify how to deal with missing responses by setting the logical argument `miss.imp`. The MAR assumption is adopted if this argument is set to `FALSE` (the default value). Otherwise, missing outcomes are imputed using the `imp.mix()` function from the package **mix** (Schafer, 2024) before starting the estimation.

The following command provides a summary of the estimation results:

```
summary(mod4)

#> Call:
#> lmestCont(responsesFormula = sap + dap + hr ~ NULL, latentFormula = ~fluid +
#>     as.factor(gender) + age, data = data_heart_sim, index = c("id",
#>     "time"), k = 3, miss.imp = FALSE)
#>
#> Coefficients:
#>
#>  Be - Parameters affecting the logit for the initial probabilities:
#>                     logit
#>                           2       3
#>   (Intercept)        -0.0348  0.0258
#>   fluid               0.0003  0.0001
#>   as.factor(gender)2 -0.4704  0.3428
#>   age                -0.0290 -0.0096
#>
#>  Ga - Parameters affecting the logit for the transition probabilities:
#> , , logit = 1
#>
#>                     logit
#>                           2       3
#>   (Intercept)        -2.1318 -2.3075
#>   fluid              -0.0007  0.0001
#>   as.factor(gender)2  2.1343  0.0536
#>   age                -0.0047  0.0076
#>
#> , , logit = 2
#>
#>                     logit
#>                           2       3
```

```
#>   (Intercept)        -0.9958 -1.9856
#>   fluid              -0.0025 -0.0001
#>   as.factor(gender)2  4.3148  2.4297
#>   age                -0.0493 -0.0641
#>
#> , , logit = 3
#>
#>                     logit
#>                         2       3
#>   (Intercept)        -2.2754 -2.4416
#>   fluid               0.0008  0.0007
#>   as.factor(gender)2  0.3736 -1.9600
#>   age                -0.0700 -0.0342
#>
#>
#>  Mu - Conditional response means:
#>      state
#>              1        2        3
#>   sap 101.0068 104.9304 127.5081
#>   dap  61.9488  66.9759  73.6769
#>   hr   74.2144 102.1859  83.0373
#>
#>  Si - Variance-covariance matrix:
#>          [,1]     [,2]     [,3]
#> [1,] 197.0381  55.0938  12.7835
#> [2,]  55.0938 107.8801  21.0515
#> [3,]  12.7835  21.0515 123.8274
```

Considering the estimated conditional means under the HM model with $k = 3$ latent states, we observe that these states are ordered increasingly according to the values of the estimated means for both systolic (`sap`) and diastolic (`dap`) blood pressure. The estimated regression parameters related to the initial probabilities are stored in object `Be`. The log-odds ratio for gender is positive for the 3rd state and negative for the 2nd, indicating that females are more likely to be in the 3rd and 1st states one day after surgery, holding other covariates constant.

The output `Ga` refers to the estimated regression parameters on the transition probabilities. For example, the values in the 1st column of `Ga[,,3]` measure the influence of each covariate on the transition from the 3rd to the 1st state. For females, the probability of this transition is higher than for males and age has a negative effect on this transition.

Parametric bootstrap standard errors for the parameter estimates can be obtained using the `bootstrap()` method by specifying, as input argument, the object returned by the call of `lmestCont()`. The number of bootstrap samples can be specified through argument `B`. The call is as follows:

```
boot <- bootstrap(mod4, B = 20)
```

Results of function `bootstrap()` include the average of bootstrap estimates and the standard errors for the model parameters. For example, the estimated standard errors for the regression parameters affecting the logits on initial and transition probabilities can be obtained as

```
round(boot$seBe, 3)
```

```
#>                     logit
#>                         2     3
#>   (Intercept)        0.014 0.031
#>   fluid              0.000 0.000
#>   as.factor(gender)2 0.206 0.402
#>   age                0.012 0.011
```

```
round(boot$seGa, 3)
```

```
#> , , logit = 1
#>
#>                     logit
#>                         2     3
```

```
#>   (Intercept)        0.011 0.036
#>   fluid              0.000 0.001
#>   as.factor(gender)2 0.099 0.520
#>   age                0.015 0.026
#>
#> , , logit = 2
#>
#>                     logit
#>                         2     3
#>   (Intercept)        4.240 0.705
#>   fluid              0.006 0.004
#>   as.factor(gender)2 4.477 2.196
#>   age                0.159 0.301
#>
#> , , logit = 3
#>
#>                     logit
#>                         2     3
#>   (Intercept)        0.031 0.042
#>   fluid              0.004 0.001
#>   as.factor(gender)2 0.245 0.629
#>   age                0.318 0.069
```

### 4.4 Application to longitudinal data with imputed missing values and covariates in the measurement (sub)model

We examine data collected by the Minnesota Department of Education for all Minnesota schools during the period 2008-2010, as illustrated in Roback and Legler (2021). The dataset is available in the GitHub repository at the link provided in the following chunk.

```
urlfile <- "https://raw.githubusercontent.com/proback/BeyondMLR/master/data/chart_wide_condense.csv"

data <- read.csv(url(urlfile))
n <- nrow(data); TT <- 3
data_school <- data.frame(id = rep(1:n, each = TT), time = rep(1:TT, n),
                          chart = rep(data$charter, each = TT),
                          sped = rep(data$schPctsped, each = TT),
                          math = c(t(data[,c("MathAvgScore.0",
                                             "MathAvgScore.1", "MathAvgScore.2")])))
```

The final part of the dataset in long format is displayed in the following:

```
round(tail(data_school), 3)

#>       id time chart  sped  math
#> 1849 617    1     1 0.105    NA
#> 1850 617    2     1 0.105    NA
#> 1851 617    3     1 0.105 651.4
#> 1852 618    1     1 0.455    NA
#> 1853 618    2     1 0.455    NA
#> 1854 618    3     1 0.455 631.2
```

The data refer to 618 schools, and the aim is to compare student performance in charter schools versus public schools by analyzing Minnesota Comprehensive Assessment average math scores from 6th to 8th graders in each school. The response variable (`math`) is recorded at three time occasions corresponding to years 2008, 2009, and 2010. Type of school (`chart`: coded as 1 for a charter school, and 0 for a public school), and the proportion of special education students in a school (`sped`), based on 2010 figures, are time-fixed covariates. In this example, we investigate the effects of the two covariates on the test performance, specifically examining whether there are differences between charter and public schools.

We observe that the school with id 618 has missing records for the math scores at both the first and second occasions. Additionally, there are 121 missing math scores out of a total of 1,854 records.

```
table(complete.cases(data_school$math))

#>
#> FALSE  TRUE
#>   121  1733
```

We estimate an HM model with two latent states, handling missing values through imputation, and accounting for the effect of covariates on the measurement model. This is done using the `lmestCont()` function with the following options:

```
mod5 <- lmestCont(responsesFormula = math ~ chart + sped,
                  data = data_school,
                  index = c("id", "time"),
                  k = 2, miss.imp = TRUE)
```

The summary output presents the estimated model parameters, including the following details:

```
summary(mod5)

#> Call:
#> lmestCont(responsesFormula = math ~ chart + sped, data = data_school,
#>     index = c("id", "time"), k = 2, miss.imp = TRUE)
#>
#> Coefficients:
#>
#> Initial probabilities:
#>      est_piv
#> [1,]  0.1652
#> [2,]  0.8348
#>
#> Transition probabilities:
#> , , time = 2
#>
#>      state
#> state      1      2
#>     1 0.9233 0.0767
#>     2 0.0130 0.9870
#>
#> , , time = 3
#>
#>      state
#> state      1      2
#>     1 0.7419 0.2581
#>     2 0.0085 0.9915
#>
#>
#>  Al - Intercepts:
#>
#> state     [,1]
#>     1 644.8250
#>     2 656.8341
#>
#>  Be - Regression parameters:
#>          [,1]
#> [1,]  -2.8784
#> [2,] -12.8279
#>
#>  Si - Variance-covariance matrix:
#>         [,1]
#> [1,] 24.0767
```

The estimated intercepts, collected in object `Al`, indicate that the 1st state corresponds to lower performing schools. Additionally, the analysis reveals a negative effect of attending a non-charter

public school on the average math score, while controlling for the percentage of special education students. Covariate sped has also a negative effect on the math score. Specifically, an increase of 1% in the proportion of special education students at a school is associated with a decrease of 12.83 points in the estimated mean math scores.

From the estimated initial probabilities, we infer that, in 2008, around 17% of schools belong to the 1st state. The two estimated transition matrices indicate a significant increase in the average math score from 2009 to 2010, as evidenced by the transition probability from the 1st to the 2nd state, which is 0.26.

The output collected in `mod5` also includes an object named `Yimp`, which is an array of dimension $n \times T \times k$ containing the imputed data. For instance, the imputed values for unit 618 at the first and second time occasion are 662.966 and 661.465, respectively.

```
round(mod5$Yimp[618,,], 3)

#> [1] 662.966 661.465 631.200
```

## 5 Discussion

The **LMest** package is designed for Markov chain (MC) and hidden Markov (HM) models. It includes functions for maximum likelihood estimation of various versions of these models, both with and without covariates, and under different constraints. In particular, the package allows analyzing longitudinal and time-series data through MC models for univariate categorical responses and HM models for both categorical and continuous responses, so covering a wide range of applications.

The primary features of the models at issue, and of HM models in particular, are the great flexibility and the capability of performing dynamic model-based clustering with each cluster corresponding to a support point of the Markov chain. HM models are estimated using the Expectation-Maximization (EM) algorithm. Given that the likelihood is often multimodal, the package provides appropriate criteria to assess the convergence and choose different sets of starting values for the model parameters. Moreover, in order to optimize computational efficiency, the Fortran language is used for many numerical operations. The package also includes functions for simulating data from different versions of the HM models, as well as for displaying and visualizing parameter estimates. Parametric bootstrap can also be applied to provide standard errors for the parameters estimates, offering an alternative to using the information matrix.

In addition to the previous features, the **LMest** package also allows estimation of a model for longitudinal categorical data based on a mixture of latent AR(1) processes to dynamically account for unobserved heterogeneity. Each mixture component has a specific mean and correlation coefficient, but these components share a common variance. Therefore this model is based on a continuous latent process and is tailored for univariate data, in which the response variable has an ordinal nature. As another extension, it is possible to estimate mixed HM models (van de Pol and Langeheine, 1990) through function `lmestMixed()`, which is formulated to account for additional sources of time-fixed dependency in the data. The parameters of the latent process can vary across different latent subpopulations defined by an additional latent variable.

It is worth mentioning that, within the package, it is possible to handle intermittent, entirely or partially, missing values in the responses under the missing-at-random assumption and the EM algorithm for parameter estimation is suitable adapted also for providing an imputation of the missing responses. Weighted maximum likelihood can also be performed, which can be useful in many applied contexts, such as when longitudinal survey data are analyzed. In this case, individual survey weights (Kaplan and Ferguson, 1999), which refer to the probability of each unit being sampled in the reference population, are available; see among others, Pennoni and Genge (2020) and Pennoni and Nakai (2023). By using suitable weights, it is also possible to carry out causal inference with observational longitudinal data (Robins, 1997), in a potential outcome framework (Rubin, 1974), on the basis of a propensity score method. For applications of this type see Bartolucci et al. (2016, 2023) and Pennoni et al. (2023).

As alternative packages to **LMest**, we mention **march** (Maitre et al., 2020), which provides tools for fitting discrete-time Markov chains, semi-Markov chains, and higher-order models. Moreover, we mention the **mstate** package (Putter et al., 2007) designed for modeling and analyzing multi-state models, which are generalizations of Markov chains, useful in the context of survival analysis, and the **ctmcd** package (Pfeuffer, 2024), which provides methods for parameter estimation, simulation, and detailed analysis of continuous-time Markov models.

For the HM model, it is also worth considering the **depmixS4** package (Visser and Speekenbrink, 2022). **LMest** and **depmixS4** differ from others packages mainly in how covariates are parameterized

in the latent and the manifest models. Additionally, **depmixS4** can handle a more general class of distributional assumptions when covariates are included in the measurement model. We also highlight the recent proposal of Turner (2024) with the **eglhmm** package, which allows us to estimate extended generalized linear HM models for data conforming to various distributions. Another available package of interest is **msm** (Christopher H. Jackson, 2011), which is designed for estimating continuous-time Markov and HM models for longitudinal data. It provides both maximum likelihood estimation and Bayesian inference under various models, including multi-state models. Additionally, the **HiddenMarkov** package (Harte, 2021) offers tools for modeling time series data and supports various distributions for the observations, including Gaussian and Poisson distributions. Lastly, package **seqHMM** (Helske and Helske, 2019) is designed for fitting HM models and mixture HM models specifically for social sequence data and other types of categorical data.

The extensive development and availability of packages for HM models on CRAN highlights the broad applicability across diverse fields of such models. Their flexibility and robustness in handling various types of data, combined with their capability to uncover underlying unobserved processes, demonstrate the popularity of these models and their value among researchers and practitioners. Finally, as possible extension of the **LMest** package we consider that to handle informative missing data, as in the case of dropout from a longitudinal study. As described in Pandolfi et al. (2023), dropout can be accounted for by considering an absorbing state, which allows modeling survival time without specifying a separate survival model component. This extension will be included in future updates of the package. The package will also be extended to allow for variable selection, as recently proposed in Pennoni et al. (2024), where a greedy search algorithm is implemented to identify the most important response variables. Other forthcoming extensions include functions to estimate models for data with a multilevel structure, as proposed in Bartolucci et al. (2011), additional parameterizations for the covariates affecting the latent (sub)model, as described in Bartolucci et al. (2024b), and methods for dealing with compositional data, as proposed in Bartolucci et al. (2024a).

**Acknowledgments**

The authors are grateful for the financial support from the grant "*Hidden Markov Models for Early Warning Systems*" of Ministero dell'Università e della Ricerca (PRIN 2022TZEXKF) funded by the European Union - Next Generation EU, Mission 4, Component 2, CUP J53D23004990006.

## References

H. Akaike. Information theory as an extension of the maximum likelihood principle. In B. N. Petrov and C. F., editors, *Second International symposium on information theory*, pages 267–281, Budapest, 1973. Akademiai Kiado. [p74, 81]

T. Anderson. Probability models for analyzing time changes in attitudes. In P. F. Lazarsfelsd, editor, *Mathematical Thinking in the Social Science*, pages 17–66. New York; Free Press, 1954. [p75]

A. Azzalini. Logistic regression for autocorrelated data with application to repeated measures. *Biometrika*, 81:767–775, 1994. [p76]

S. Bacci, S. Pandolfi, and F. Pennoni. A comparison of some criteria for states selection in the latent Markov model for longitudinal data. *Advances in Data Analysis and Classification*, 8:125–145, 2014. [p81]

F. Bartolucci and A. Farcomeni. A multivariate extension of the dynamic logit model for longitudinal data based on a latent Markov heterogeneity structure. *Journal of the American Statistical Association*, 104:816–831, 2009. [p81]

F. Bartolucci and A. Farcomeni. Information matrix for hidden Markov models with covariates. *Statistics and Computing*, 25:515–526, 2015. [p81]

F. Bartolucci and F. Pennoni. Impact evaluation of job training programs by a latent variable model. In *New Perspectives in Statistical Modeling and Data Analysis: Proceedings of the 7th Conference of the Classification and Data Analysis Group of the Italian Statistical Society, Catania, September 9-11, 2009*, pages 65–73. Springer-Verlag, 2011. [p76]

F. Bartolucci, F. Pennoni, and G. Vittadini. Assessment of school performance through a multilevel latent Markov Rasch model. *Journal of Educational and Behavioural Statistics*, 36:491–522, 2011. [p98]

F. Bartolucci, A. Farcomeni, and F. Pennoni. *Latent Markov Models for Longitudinal Data*. Chapman & Hall/CRC Press, Boca Raton, FL, 2013. [p74, 75, 81, 82]

F. Bartolucci, S. Bacci, and F. Pennoni. Longitudinal analysis of self-reported health status by mixture latent auto-regressive models. *Journal of the Royal Statistical Society, Series C*, 63:267–288, 2014a. [p81]

F. Bartolucci, A. Farcomeni, and F. Pennoni. Latent Markov models: A review of a general framework for the analysis of longitudinal data with covariates. *TEST*, 23:433–465, 2014b. [p75, 76]

F. Bartolucci, G. E. Montanari, and S. Pandolfi. Three-step estimation of latent Markov models with covariates. *Computational Statistics & Data Analysis*, 83:287–301, 2015. [p80]

F. Bartolucci, F. Pennoni, and G. Vittadini. Causal latent Markov model for the comparison of multiple treatments in observational longitudinal studies. *Journal of Educational and Behavioral Statistics*, 41: 146–179, 2016. [p97]

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. [p74, 80]

F. Bartolucci, S. Pandolfi, and F. Pennoni. Discrete latent variable models. *Annual Review of Statistics and its Application*, 9:425–452, 2022. [p74]

F. Bartolucci, F. Pennoni, and G. Vittadini. A causal latent transition model with multivariate outcomes and unobserved heterogeneity: Application to human capital development. *Journal of Educational and Behavioral Statistics*, 48:387–419, 2023. [p97]

F. Bartolucci, M. Greenacre, S. Pandolfi, and F. Pennoni. Hidden Markov and related discrete latent variable models: An application to compositional data. In G. Giordano, M. La Rocca, B. Niglio, M. Restaino, and M. Vichi, editors, *Studies in Classification, Data Analysis and Knowledge Organization*, CLADAG 2023, pages 1–8. Springer, 2024a. [p98]

F. Bartolucci, S. Pandolfi, and F. Pennoni. Parsimonious parametrizations of transition matrices of markov chain and hidden markov models. *Submitted*, pages 1–25, 2024b. [p98]

F. Bassi, F. Pennoni, and L. Rossetto. Market segmentation and dynamic analysis of sparkling wine purchases in italy. *Journal of Wine Economics*, 16:283–304, 2021. [p82]

L. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, 37:1554–1563, 1966. [p81]

L. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41:164–171, 1970. [p81]

C. Bouveyron, G. Celeux, T. B. Murphy, and A. E. Raftery. *Model-based Clustering and Classification for Data Science: with Applications in R*. Cambridge University Press, Cambridge, UK, 2019. [p74]

Christopher H. Jackson. Multi-state models for panel data: The msm package for R. *Journal of Statistical Software*, 8:1–29, 2011. doi: 10.18637/jss.v038.i08. [p98]

A. C. Davison and D. V. Hinkley. *Bootstrap Methods and their Application*. Cambridge University Press, Cambridge, MA, 1997. [p74]

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977. [p74, 81]

Y. Ephraim and N. Merhav. Hidden Markov processes. *IEEE Transactions on Information Theory*, 48: 1518–1569, 2002. [p74]

D. Harte. *HiddenMarkov: Hidden Markov Models*. Statistics Research Associates, Wellington, 2021. URL https://www.statsresearch.co.nz/dsh/sslib/. R package version 1.8-13. [p98]

J. Helske and S. Helske. seqhmm: Mixture hidden Markov models for social sequence data and other multivariate, multichannel categorical time series. *Journal of Statistical Software*, 88:1–32, 2019. [p98]

B. Juang and L. Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33:251–272, 1991. [p82]

D. Kaplan and A. J. Ferguson. On the utilization of sample weights in latent variable models. *Structural equation modeling: A Multidisciplinary Journal*, 6:305–321, 1999. [p97]

R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley Sons, Hoboken, NJ, 2020. [p75, 86]

O. Maitre, K. Emery, with contributions from O. Buschor, and A. Berchtold. *march: Markov Chains*, 2020. URL https://CRAN.R-project.org/package=march. R package version 3.3.2. [p97]

S. P. Meyn and R. L. Tweedie. *Markov Chains and Stochastic Stability*. Springer Science & Business Media, London, 2012. [p74]

B. Mor, S. Garhwal, and A. Kumar. A systematic review of hidden Markov models and their applications. *Archives of Computational Methods in Engineering*, 28:1429–1448, 2021. [p74]

L. J. Paas, J. K. Vermunt, and T. H. A. Bilmolt. Discrete time, discrete state latent Markov modelling for assessing and predicting household acquisitions of financial products. *Journal of the Royal Statistical Society, Series A*, 170:955–974, 2007. [p82]

S. Pandolfi, F. Bartolucci, and F. Pennoni. A hidden Markov model for continuous longitudinal data with missing responses and dropout. *Biometrical Journal*, 5:1–28, 2023. [p87, 98]

F. Pennoni and E. Genge. Analysing the course of public trust via hidden Markov models: a focus on the Polish society. *Statistical Methods & Applications*, 29:399–425, 2020. [p97]

F. Pennoni and M. Nakai. Exploring heterogeneity in happiness: Evidence from a Japanese longitudinal survey. In *Facets of Behaviormetrics: The 50th Anniversary of the Behaviormetric Society*, pages 193–217. Springer, 2023. [p97]

F. Pennoni and G. Vittadini. Two competing models for ordinal longitudinal data with time-varying latent effects: An application to evaluate hospital efficiency. *Quaderni di Statistica*, pages 53–68, 2013. [p81]

F. Pennoni, F. Bartolucci, G. Forte, and F. Ametrano. Exploring the dependencies among main cryptocurrency log-returns: A hidden Markov model. *Economic Notes*, 51:1–27, 2022. [p88]

F. Pennoni, L. J. Paas, and F. Bartolucci. A causal hidden Markov model for assessing effects of multiple direct mail campaigns. *TEST*, pages 1–29, 2023. [p97]

F. Pennoni, F. Bartolucci, and S. Pandolfi. Variable selection for hidden Markov models with continuous variables and missing data. *Journal of Classification*, pages 1–28, 2024. doi: https://doi.org/10.1007/s00357-024-09464-4. [p98]

M. Pfeuffer. *ctmcd: Estimating the Parameters of a Continuous-Time Markov Chain from Discrete-Time Data*, 2024. URL https://CRAN.R-project.org/package=ctmcd. R package version 1.4.4. [p97]

H. Putter, R. B. Geskus, and M. Fiocco. Tutorial in biostatistics: Competing risks and multi-state models. *Statistics in Medicine*, 26:2389–2430, 2007. [p97]

P. Roback and J. Legler. *Beyond Multiple Linear Regression: Applied Generalized Linear Models and Multilevel Models in R*. Chapman and Hall/CRC Press, New York, 2021. [p95]

J. Robins. Causal inference from complex longitudinal data. In M. Berkame, editor, *Latent Variable Modeling and Applications to Causality*, volume 120 of *Lecture Notes in Statistics*, pages 69–117. Springer, New York, 1997. [p97]

D. B. Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66:688–701, 1974. [p97]

J. A. Ryan and J. M. Ulrich. *quantmod: Quantitative Financial Modelling Framework*, 2022. URL https://CRAN.R-project.org/package=quantmod. R package version 0.4.26. [p88]

J. L. Schafer. *mix: Estimation/Multiple Imputation for Mixed Categorical and Continuous Data*, 2024. URL https://CRAN.R-project.org/package=mix. R package version 1.0-12. [p93]

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978. [p74, 81]

R. Turner. *eglhmm: Extended Generalised Linear Hidden Markov Models*, 2024. URL https://CRAN.R-project.org/package=eglhmm. R package version 0.1-3. [p98]

F. van de Pol and R. Langeheine. Mixed Markov latent class models. *Sociological Methodology*, 20:213–247, 1990. [p97]

I. Visser and M. Speekenbrink. *Mixture and Hidden Markov Models with R*. Springer, Cham, CH, 2022. [p81, 97]

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967. [p82]

L. Wiggins. *Panel Analysis: Latent Probability Models for Attitude and Behaviour Processes*. Elsevier, Amsterdam, 1973. [p74]

A. Zeileis and Y. Croissant. Extended model formulas in R: Multiple parts and multiple responses. *Journal of Statistical Software*, 34:1–13, 2010. doi: 10.18637/jss.v034.i01. [p75, 78]

W. Zucchini, I. L. MacDonald, and R. Langrock. *Hidden Markov Models for Time Series: An Introduction using R*. CRC press, Boca Raton, FL, 2016. [p74]

*Fulvia Pennoni*
*University of Milano-Bicocca*
*Department of Statistics and Quantitative Methods*
*Milan, Italy*
https://sites.google.com/view/fulviapennoni/home
*ORCiD: 0000-0002-6631-7211*
fulvia.pennoni@unimib.it

*Silvia Pandolfi*
*University of Perugia*
*Department of Economics*
*Perugia, Italy*
https://sites.google.com/site/spandolfihome/home
*ORCiD: 0000-0002-6631-1211*
silvia.pandolfi@unipg.it

*Francesco Bartolucci*
*University of Perugia*
*Department of Economics, Via A. Pascoli 8*
*Perugia, Italy*
https://sites.google.com/site/bartstatistics/
*ORCiD: 0000-1721-1511-1101*
francesco.bartolucci@unipg.it

# Calculating Standardised Indices Using SEI

*by Sam Allen and Noelia Otero*

**Abstract** Standardised indices are measurements of variables on a standardised scale. The standardised scale facilitates comparisons between different variables, and its probabilistic interpretation means the indices are effective for risk management and decision making. Standardised indices have become popular in weather and climate settings, for example within operational drought monitoring systems, and have also been applied in other contexts, such as to energy variables. To facilitate their implementation in practice, the SEI package in R allows the flexible calculation of standardised indices. This paper discusses the theory underlying well-known standardised indices, outlines the general framework to construct them, and provides implementation details for the SEI package. Two case studies are presented whereby standardised indices are applied to climate and energy variables.

## 1 Introduction

Monitoring the values of a variable over time allows practitioners to not only identify long term patterns in the variables of interest, but also to analyse abnormal situations with potentially adverse socio-economic consequences. However, it can be difficult to compare time series of variables defined on different scales. For example, different spatial regions have different climates, meaning a rainfall event that is impactful at one location may not be at another. Similarly, energy demand will depend on the local climate, as well as on installed energy capacities. To account for this, it is convenient to transform variables onto the same, standardised scale. The resulting standardised variables are often referred to as *standardised indices*.

Well-known examples of standardised indices include the Standardised Precipitation Index (SPI) (McKee et al., 1993) and Standardised Precipitation Evapotranspiration Index (SPEI) (Vicente-Serrano et al., 2010), which are ubiquitously used to monitor hydrometeorological droughts (Begueria et al., 2014). The framework used to construct the SPI and SPEI is simple and interpretable, framing the observed values in terms of their similarity to past values. As such, while other methods of standardisation exist, this framework has also been used to construct many other weather and climate indices.

This includes, for example, a Standardised Runoff Index (Shukla and Wood, 2008), a Standardised Streamflow Index (Vicente-Serrano et al., 2012), a Standardised Groundwater Index (Bloomfield and Marchant, 2013), and a Standardised Temperature Index (Zscheischler et al., 2014). Multivariate extensions of these standardised indices have also been proposed (Erhardt and Czado, 2018). Hao et al. (2019), for example, define a Standardised Compound Event Index, which is used by Li et al. (2021) to construct a Standardised Compound Drought and Heat Index. Outside of a weather and climate context, Allen and Otero (2023) introduced analogues of the SPI and SPEI that can be used to monitor energy supply and demand. This general framework for constructing standardised indices can be applied to any variable of interest, not just those previously considered in the literature.

There are several benefits to these standardised indices. The indices are easy to interpret, and are defined on a scale with a probabilistic interpretation, making them useful tools for risk management and decision making. Standardised indices therefore provide an appealing framework with which to define shortages, or droughts, in the variable of interest. Since the standardisation can be performed separately in different conditions, for example in different seasons and locations, these shortages can be defined in a relative sense, allowing their intensity in the different conditions to be compared. As summarised by Zargar et al. (2011), standardised indices provide a "pragmatic way to assimilate large amounts of data into a quantitative information that can be used in applications such as drought forecasting, declaring drought levels, contingency planning and impact assessment."

As a result, several software packages have been developed to apply standardised indices in practice. The **SCI** package (Gudmundsson and Stagge, 2016) in the programming language R provides the functionality to calculate several standardised climate indices. The **SPEI** package (Beguería and Vicente-Serrano, 2023), also in R, contains more comprehensive functionality for the SPI and SPEI, and the **standard-precip** package (Nussbaumer, 2021) has transferred this functionality to the programming language Python (Python, 2017). The **spei** package (Vonk, 2017) in Python additionally allows computation of the SPI and SPEI, as well as the Standardised Runoff and Groundwater Indices, while the **climate-indices** package in Python also provides functionality to compute the SPI, SPEI, and other drought indices. The developmental package **standaRdized** (Maetens, 2019) can also be used for this purpose, while the **SPIGA** package (Ayala-Bizarro and Zuniga-Mendoza, 2016) in R allows

the SPI to be calculated using genetic algorithms.

However, existing software packages are limited in their flexibility, with functionality typically restricted to particular standardised indices (generally the SPI and SPEI), and particular assumptions about the reference data or parametric distributions used to calculate the indices (see next section for details). In this paper, we introduce the **SEI** R package, which seeks to assimilate the advantages of these various packages to provide a flexible and comprehensive software package with which to calculate standardised indices. In particular, the package is not designed solely for climate indices, and is therefore applicable in much broader generality. The package can handle variables defined on any time scale and with any (possibly non-stationary) underlying distribution, permits user-specified reference data when calculating the indices, and additionally provides plot capabilities to visualise the resulting standardised indices.

The following section provides theoretical background on standardised indices and describes the general formula to construct them. Section 3 discusses diagnostic checks that confirm the standardisation was appropriate, while Section 4 demonstrates how the **SEI** package can be used to calculate standardised indices in practice. Two applications are then presented in Section 5, whereby the **SEI** package is used to calculate standardised energy indices based on time series of renewable energy production, as well as standardised wind speed indices. Section 6 concludes the paper and discusses possible extensions to the package.

## 2 How to construct standardised indices

There is no unique way to define standardised indices. However, several widely-adopted indices are constructed using the same, simple procedure. Using this procedure, standardised indices are straightforward to calculate, can be defined on any timescale, and behave in a relative sense, meaning they can readily be compared for different variables, spatial regions, or points in time.

The general approach is to choose a univariate variable $X$ that measures the quantity of interest, and then to estimate $X$'s distribution: in the SPI, $X$ represents the precipitation accumulation aggregated over the time scale of interest, which is usually assumed to follow a gamma distribution; in the SPEI, $X$ represents the difference between precipitation and evapotranspiration, which is usually assumed to follow a log-logistic distribution; in the standardised temperature index, $X$ represents the temperature, which is usually assumed to follow a normal distribution. More complicated variables have also been considered. Hao et al. (2019), for example, consider $X$ to be the probability of a simultaneously hot and dry event, as derived from a copula analysis.

The distribution of $X$ is typically estimated from an archive of realisations, $x_1, \ldots, x_n$. These could represent precipitation accumulations in different months, for example, or energy demand on different days. It is common to make parametric assumptions about the distribution, as in the examples above, though non- and semi-parametric density estimation methods could also be employed. The choice of distribution is discussed in detail in the following section. In any case, the distributional assumptions must be verified when the index is calculated.

Let $F$ denote the distribution function of $X$, and let $\hat{F}$ denote an estimate of $F$ obtained from $x_1, \ldots, x_n$. If $F$ is continuous, then the *probability integral transform* (PIT) $F(X)$ will be standard uniformly distributed. Hence, this transformation provides a convenient approach with which to standardise a (possibly new) random observation $X^*$ (with realisation denoted $x^*$): the PIT value $\hat{F}(x^*)$ denotes the relative position of the observation $x^*$ within the distribution $\hat{F}$. If $x^*$ is large relative to $\hat{F}$, then the PIT value will be close to one; if $x^*$ is small relative to $\hat{F}$, then the PIT value will be close to zero.

This *PIT variable* $\hat{F}(X^*)$ is bounded between 0 and 1, with values intrinsically linked to probabilities. This therefore itself provides a standardised index of sorts. This PIT variable can also be rescaled using $2\hat{F}(X^*) - 1$ to get an index that is centred at zero and bounded between -1 and 1, though it is more common to use the Gaussian quantile function $\Phi^{-1}$ to transform $\hat{F}(X^*)$. In this case, if $X^*$ is identically distributed to $X$, and $\hat{F}$ is a good approximation of $F$, then the standardised index will follow a standard normal distribution.

That is, for any variable of interest, the probability integral transform can be used to define three different types of standardised indices:

- *Normal indices*: $\Phi^{-1}(\hat{F}(X^*))$
- *Probability indices*: $\hat{F}(X^*)$
- *Bounded indices*: $2\hat{F}(X^*) - 1$

All three types of indices can be calculated from observations of the variable $X$ and an estimate of its distribution $F$. Since they are defined as increasing functions of PIT values, these standardised

indices all quantify the relative position of an observation $x^*$ within the previously observed sample $x_1, \ldots, x_n$, making them easy to interpret.

In practice, it is most common to use normal indices, though one could argue that probability and bounded indices have a more direct probabilistic interpretation. For example, if we observe the $90^{th}$ percentile of $X$, then (assuming $F$ is correctly estimated) the corresponding normal index will be 1.28, the $90^{th}$ percentile of the standard normal distribution, whereas the probability index will be 0.9, and the bounded index will be 0.8. Similarly, for the $10^{th}$ percentile of $X$, the normal index will be -1.28, the probability index will be 0.1, and the bounded index will be -0.8. For the median of $X$, the normal and bounded indices will be zero, whereas the probability index will be 0.5.

The general framework can therefore be summarised as follows.

- Estimate $F$ from the observations $x_1, \ldots, x_n$.
- Check that the estimate $\hat{F}$ correctly fits the data.
- Apply $\hat{F}$ to an observation $x^*$ (or several observations).
- Transform $\hat{F}(x^*)$ so that the index is on the desired scale.

The probabilistic interpretation of these standardised indices has made them a useful tool when analysing shortages and excesses of the variable $X$. This was first proposed by McKee et al. (1993) in the context of hydrometeorological droughts, and Allen and Otero (2023) demonstrate that the same framework can be employed to define droughts in energy systems. In this case, droughts are defined when the standardised index exceeds or falls below a relevant threshold. Different classes of droughts can be defined using different thresholds, with a more severe drought corresponding to a more extreme threshold.

Following McKee et al. (1993), it is most common to study three categories of droughts, typically labelled "Moderate", "Severe", and "Extreme" droughts. These often correspond to the thresholds 1, 1.5, and 2 of the normal standardised indices, respectively, though the thresholds 1.28, 1.64, and 1.96 are also often used, which correspond to quantiles of the standard normal distribution. This assumes that a drought is defined as an exceedance of a threshold, though the negative of these values could analogously be used when interest is on values that fall below a threshold. Table 1 gives corresponding threshold values for the other two types of indices.

**Table 1:** Possible thresholds used to define droughts when using each of the three types of standardised indices. Values are shown for droughts defined when the index falls below (left) and exceeds (right) the threshold.

| Index Type | Extreme | Severe | Moderate | Moderate | Severe | Extreme |
|---|---|---|---|---|---|---|
| Normal (`normal`) | -1.96 | -1.64 | -1.28 | 1.28 | 1.64 | 1.96 |
| Probability (`prob01`) | 0.025 | 0.05 | 0.1 | 0.9 | 0.95 | 0.975 |
| Bounded (`prob11`) | -0.95 | -0.9 | -0.9 | 0.8 | 0.9 | 0.95 |

The **SEI** package allows the computation of all three types of indices, and additionally allows drought occurrences and characteristics to be computed from a time series of standardised indices.

## 3 The choice of distribution

### 3.1 Stationary distribution estimation

The construction of standardised indices relies on an estimate of the distribution function $F$. This has led to much debate about what distributional assumptions are most appropriate. Consider the SPI, for example. McKee et al. (1993) initially introduced the SPI using the gamma distribution to model $F$. However, Guttman (1999) argued that the Pearson III distribution is more appropriate, due to the additional flexibility afforded by an extra parameter, while results in Stagge et al. (2015) suggest the Weibull distribution fits the monthly precipitation accumulations in their study better than alternative parametric choices. Russo et al. (2013) also consider using non-stationary distributions that include parameters that depend on time, while Li et al. (2015) incorporate additional covariates based on other climate variables. In reality, there is generally no true parametric distribution underlying the data, and the optimal assumptions will change depending on the data under consideration.

As an alternative, it has been argued (e.g. Erhardt and Czado, 2018; Hao et al., 2019; Allen and Otero, 2023) that the choice of a parametric distribution can be circumvented by using the empirical

distribution function of the observations,

$$\hat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} 1\{x_i \leq x\}, \tag{1}$$

where $1\{\cdot\}$ is equal to one if the statement inside the brackets is true, and zero otherwise. It is common to rescale $\hat{F}_n(x)$ so that it is never equal to zero or one, thereby ensuring that the (normal) standardised indices are real-valued; for example, using $(n\hat{F}_n(x)+1)/(n+2)$. The empirical distribution function returns a standardised index that can only take on $n+1$ possible values. This is not an issue in practice if $n$ is large, but can become problematic with smaller sample sizes, particularly when interest is on extreme events.

Allen and Otero (2023) propose the use of semi-parametric density estimation methods, which provide a compromise between parsimonious parametric distributions and the empirical distribution function. For example, kernel density estimators estimate the density function $f$ corresponding to $F$ using

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right),$$

where $K$ is a kernel function, and $h > 0$ is a smoothing parameter called the bandwidth. The **SEI** package provides functionality to perform kernel density estimation using a Gaussian kernel. Non- and semi-parametric density estimation methods are considerably more flexible than parametric methods, though they require more data to accurately model $F$. Further details about density estimation methods can be found, for example, in Silverman (2018).

**Table 2:** Distributions available in **SEI**, and their supports.

| Distribution | Argument Code | Support |
|---|---|---|
| Empirical | `"empirical"` | range$(x_1, \ldots, x_n)$ |
| Kernel density estimation | `"kde"` | $(-\infty, \infty)$ |
| Normal | `"norm"` | $(-\infty, \infty)$ |
| Log-normal | `"lnorm"` | $[0, \infty)$ |
| Logistic | `"logis"` | $(-\infty, \infty)$ |
| Log-logistic | `"llogis"` | $[0, \infty)$ |
| Exponential | `"exp"` | $[0, \infty)$ |
| Gamma | `"gamma"` | $[0, \infty)$ |
| Weibull | `"weibull"` | $[0, \infty)$ |

Table 2 lists the distributions currently available in the **SEI** package for estimating $F$. The choice of distribution should align with the variable of interest. For example, if interest is on precipitation, then we should not choose a distribution that assigns positive probability density to negative values, such as the normal distribution. Conversely, if interest is on temperature (in Celsius), then we should not choose a distribution that has support $[0, \infty)$, such as the exponential distribution. Table 2 additionally contains the support of the available distributions; the support of the empirical distribution is determined by the archive of observations $x_1, \ldots, x_n$.

### 3.2 Non-stationary distribution estimation

In many practical applications, the variable of interest may be non-stationary. For example, several weather variables have exhibited strong temporal trends in recent decades as a result of changes in the climate. If these trends are ignored when estimating the distribution $F$, then the resulting index values may be misleading: if the variable is increasing in time, then larger index values will generally occur for later observations, meaning extreme events, as defined using the standardised indices, will be clustered in time.

This could be circumvented by estimating $F$ using an adaptive subset of the observations $x_1, \ldots, x_n$ that accounts for these trends, rather than using all available data. For example, $\hat{F}$ could be constructed from a moving window that only considers (or assigns higher weight to) the most recent observations, so that the standardised indices, as well as the corresponding drought and extreme events, are defined in terms of the recent climate. This would require estimating a separate distribution for each new observation $x^*$, but provides an example of how the data used to construct standardised indices affects their interpretation.

Alternatively, the trends could be accounted for by estimating $F$ as a function of time, so that time is included as a predictor in the distribution (Russo et al., 2013). The distribution could similarly be

made to depend on other predictor variables (Li et al., 2015). For example, precipitation accumulations are strongly influenced by changes in the temperature. By incorporating temperature as a predictor within the distribution of precipitation, the PIT values would quantify how extreme a precipitation observation is, given the corresponding temperature. The problem of estimating the distribution of the observations $x_1, \ldots, x_n$ then becomes estimating the conditional distribution of the observations given corresponding realisations of a set of predictor variables.

There are many ways to estimate this conditional distribution (see e.g. Kneib et al., 2023), but one flexible and general approach is the Generalized Additive Models for Location, Scale and Shape (GAMLSS) framework (Rigby and Stasinopoulos, 2005). Several recent studies have proposed to construct standardised indices from non-stationary distributions estimated using GAMLSS (e.g. Wang et al., 2015; Rashid and Beecham, 2019; Shao et al., 2022). The general approach is to choose a parametric distribution and then assume that its location, scale and shape parameters depend additively (but possibly non-linearly) on the set of predictor variables. GAMLSS is a well-known framework for which many resources are available, to which readers are referred for further details. The **SEI** package allows non-stationary distributions to be implemented to construct standardised indices by calling functions from the **gamlss** R package (Stasinopoulos and Rigby, 2008; Stasinopoulos et al., 2017).

Incorporating predictors into standardised indices implicitly assumes that the impacts associated with the variable of interest are somehow mitigated by the predictors. For example, a society's capability to deal with hydrometeorological droughts typically becomes better as these events become more common, so that an unusually low precipitation event today leads to a lower impact than it would have done 50 years ago, say; instead, a yet more extreme precipitation event is required to generate the same impact. Including time, or other relevant predictors, within the distribution $F$ allows this information to be incorporated into the standardised indices. Conversely, if we cannot assume that the predictors help to mitigate the impacts associated with the variable being monitored, then there is little reason to include it as a predictor when constructing standardised indices.

### 3.3 Checking distribution fit

Regardless of the method used to estimate $F$, it is necessary to check that the estimated distribution fits the data. Several goodness-of-fit tests exist for this purpose. The **SEI** package implements the Kolmogorov-Smirnov test (Massey Jr, 1951) applied to the PIT values $\hat{F}(x_1), \ldots, \hat{F}(x_n)$. If the distribution fits the data, these PIT values should resemble a sample from a standard uniform distribution. The Kolmogorov-Smirnov test statistic then calculates the maximum difference between the empirical distribution function of the PIT values and the distribution function of the standard uniform distribution. If this statistic is close to zero, then it suggests that the hypothesised distribution fits the data well.

The fit of the distribution could also be visualised graphically by plotting the distribution of the PIT values and checking whether they resemble a sample from a standard uniform distribution; a histogram of these values should therefore be flat, up to sampling variation, for example. Similarly, the corresponding normal standardised indices should resemble a sample from a standard normal distribution. An example of how this can be checked using the **SEI** package is presented in Section 5.

## 4 Package functionality

The **SEI** package is designed to facilitate the computation of standardised indices. The package is available on CRAN (`CRAN.R-project.org/package=SEI`) and a developmental version is available on GitHub (`github.com/noeliaof/SEI`). In this section, we demonstrate the functionality of the **SEI** package when implementing standardised indices in practice.

### 4.1 Standardised indices

The principal function in the **SEI** package is `std_index()`, which takes a time series or vector `x_new` as an input, and returns a time series or vector of corresponding standardised index values.

```
std_index(
  x_new,
  x_ref = x_new,
  dist = "empirical",
  preds_new = NULL,
  preds_ref = preds_new,
  method = "mle",
```

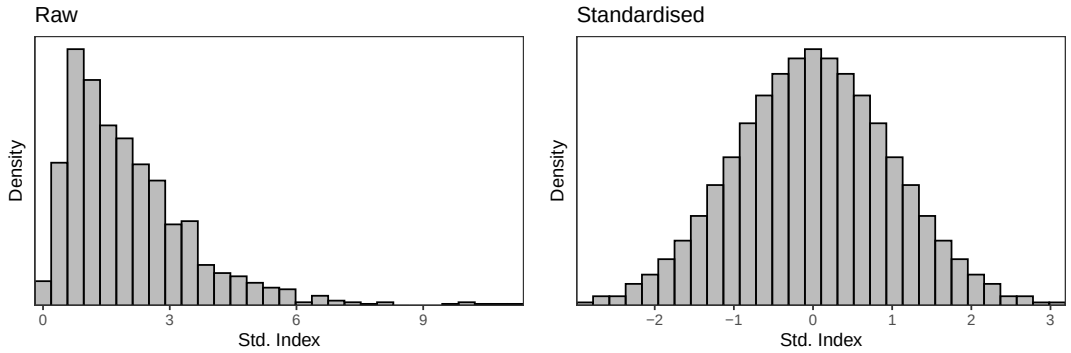



**Figure 1:** Histogram of values sampled from a gamma distribution, and corresponding standardised values.

```
  return_fit = FALSE,
  index_type = "normal",
  gr_new = NULL,
  gr_ref = gr_new,
  timescale = NULL,
  moving_window = NULL,
  window_scale = NULL,
  agg_period = NULL,
  agg_scale = NULL,
  agg_fun = "sum",
  rescale = NULL,
  rescale_fun = "sum",
  ignore_na = FALSE,
  n_thres = 10,
  na_thres = 10,
  lower = -Inf,
  upper = Inf,
  cens = index_type,
  ...
)
```

The input `x_new` must be either a vector or an **xts** time series object. The argument `x_ref` is similarly a vector or time series, containing the archive $x_1, \ldots, x_n$ of past observations that are used to estimate the distribution $F$. This distribution is then applied to each value in `x_new`, which therefore correspond to realisations of $X^*$ in the previous section. The default is that `x_ref = x_new`, in which case the standardised indices are calculated *in-sample*.

As a very simple example, consider a vector of observations drawn from a gamma distribution. These can be converted to (normal) standardised indices as follows.

```
x <- rgamma(1000, shape = 2, scale = 1)
x_std <- std_index(x)
```

Histograms of the raw and standardised values are displayed in Figure 1. While the raw values are heavily skewed, the standardised values closely resemble a standard normal distribution.

```
plot_raw <- plot_sei(x, type = "hist", title = "Raw")
plot_std <- plot_sei(x_std, type = "hist", title = "Standardised")
grid.arrange(plot_raw, plot_std, nrow = 1)
```

The argument `index_type` in `std_index()` specifies which type of standardised index should be constructed from the estimated distribution. This must be one of `"normal"` (the default), `"prob01"`, or `"prob11"`, corresponding respectively to the definitions of normal, probability, and bounded standardised indices defined in the previous section.

The `dist` argument specifies how the distribution $F$ will be estimated from `x_ref`. The default is to use the empirical distribution of `x_ref` (Equation 1). To accurately estimate $F$, the empirical distribution requires a sufficiently large archive, and a warning is therefore returned if the length of `x_ref` is smaller than 100. An error message is returned if the distribution support does not align with

the data; for example, if a distribution with support $[0, \infty)$ is chosen while the data contains negative values.

Information about the distribution fit, including estimated parameters and goodness-of-fit statistics, can be returned by setting `return_fit = TRUE`; the default is `return_fit = FALSE`. If `return_fit = TRUE`, the output of `std_index()` is a list containing the time series of standardised indices (`si`), the estimated distribution parameters (`params`), and a named vector containing properties of the model fit (`fit`); this includes the number of observations used to estimate the distribution, the number and percentage of `NA` values in the data, the Akaike Information Criterion (AIC) value, and the p-value of the Kolmogorov-Smirnov test that the distribution correctly fits the data; a small p-value provides evidence to suggest that the distribution does not fit the data.

The distribution fitting is performed using `fit_dist()`, which is essentially a wrapper for `fitdist()` in the **fitdistrplus** package, along with extensions that are relevant for the construction of standardised indices. The `fit_dist()` function is also exported by **SEI**, and further details can be found on the associated help page. Parameters of the distributions (if any) are estimated by default using maximum likelihood estimation, though many other estimation techniques, including method of moments and method of L-moments, can be employed by changing the `method` argument. Other arguments to `fitdist()` can be specified as variable arguments `...`.

### 4.2 Non-stationary distributions

The default is to estimate a non-stationary distribution from the data in `x_ref`. However, non-stationary distributions can be estimated via the GAMLSS framework by inputting a data frame of predictor variables in `preds_ref`. This data frame should have the same number of rows as the length of `x_ref`, with each column representing a separate predictor variable. Similarly, `preds_new` should be a data frame containing the same predictor variables as `preds_ref` (with the same column names), but with rows corresponding to the entries of `x_new`. The `std_index()` function then estimates the non-stationary distribution by calling `gamlss()` from the **gamlss** package. The model assumes that the location parameter of the distribution depends linearly on all predictors in `preds_ref`. Relationships between the predictors and other distribution parameters, as well as other optional arguments to `gamlss()`, can again be applied using `...`.

For example, consider observations drawn from a normal distribution whose mean and standard deviation both increase over time.

```
N <- 1000
t <- seq(10, 20, length.out = N)
x <- rnorm(N, mean = t, sd = exp(t/10))
```

The underlying distribution of these values can be estimated both with and without including a trend in the location and scale parameters of the distribution. Note that `sigma.formula` is an optional argument to `gamlss()` that allows the scale parameter of the distribution to depend on predictors.

```
preds <- data.frame(t = t)
# standardised indices without trend
si_st <- std_index(x, dist = "norm")
# standardised indices with trend in mean
si_nst <- std_index(x, dist = "norm", preds_new = preds)
# standardised indices with trend in mean and sd
si_nst2 <- std_index(x, dist = "norm", preds_new = preds, sigma.formula = ~ t)
```

Figure 2 displays the time series of standardised indices corresponding to these values under three different assumptions: there is no trend, there is a linear trend in the mean but not the standard deviation of the distribution, and there is a trend in both the mean and standard deviation. When the trend is not included, higher index values unsurprisingly occur at higher time points, and the variation of the indices also increases in time. If a trend is included in the mean of the distribution, then the standardised indices do not increase in time, but still become more variable. If a trend is included in both the mean and the standard deviation, then the index values become stationary.

### 4.3 Grouping

Rather than estimating one distribution across all observations, whether dependent on predictors or not, the variable of interest may have a distribution that changes in different subsets of the data. These subsets could correspond to different seasons, for example. Standardised indices corresponding

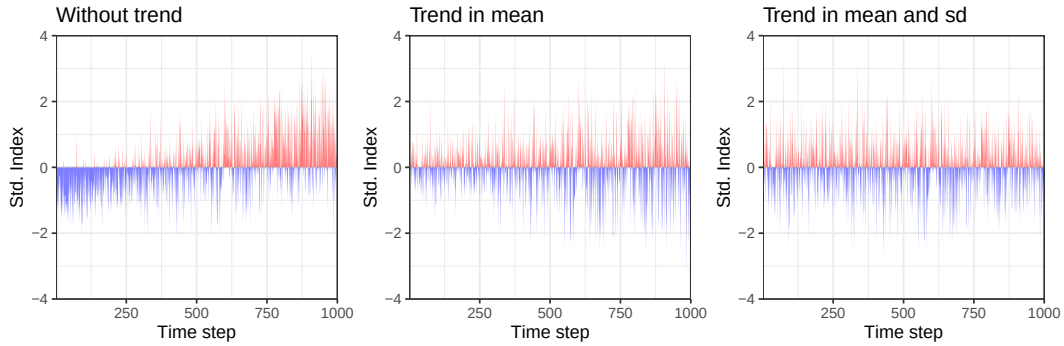



**Figure 2:** Standardised indices corresponding to a time series of values drawn from a non-stationary normal distribution. Results are shown when the estimated normal distribution does and does not include a trend.

to different distributions can be derived from `std_index()` via the `gr_new` and `gr_ref` arguments. These should be factor vectors of the same length as `x_new` and `x_ref` respectively, with each factor corresponding to a different subset of the data from which a distribution should be estimated. The values in `x_ref` are stratified according to `gr_ref`, and a separate distribution is estimated from each subset; the values in `x_new` are then stratified according to `gr_new`, and the distribution corresponding to each factor level is applied to the subset of values in `x_new` that correspond to the same level. The factor levels in `x_new` and `x_ref` should therefore be the same. In particular, `x_new` should not contain any levels that are not also present in `x_ref`, since in this case there would be no data in `x_ref` from which to estimate the relevant distribution. This returns an error.

Non- and semi-parametric distributions can flexibly adapt to situations where the shape of the distribution changes within the different subsets of the data. However, if a large number of factor levels are present in `x_ref` and `x_new`, then the amount of data to estimate the distributions can become small. To counteract this, the `dist` argument in `std_index()` can be a vector of the same length as the number of levels in `gr_ref`. The first element of `dist` denotes the distribution to be fit to the data corresponding to the first factor of `gr_ref`, and so on. Currently, for non-stationary distributions, the predictors do not depend on the grouping, and thus the same relationship between the variable of interest and the predictors must be assumed for each subset.

### 4.4 Censoring

The theory underlying standardised indices in Section 2 relies on the variable of interest being continuous, and the distributions available in **SEI** are therefore also all continuous. However, standardised indices may also be informative when the variable of interest is not continuous. A particular example is when the variable is censored; that is, it is bounded either above or below (or both), with positive probability of being exactly equal to the boundary points. Precipitation, for example, is censored below at zero.

Depending on how they are used, the definition of standardised indices may need to be adjusted to account for the censoring. This has been considered, for example, by Stagge et al. (2015). How to deal with censored variables is discussed in detail in the appendix. Censored data can be handled in **SEI** using the `lower`, `upper`, and `cens` arguments in `std_index()`: `lower` and `upper` specify the lower and upper bounds of the data, respectively, while `cens` determines the method used to address the censoring. The available methods are outlined in the appendix.

### 4.5 Time series manipulations

The remaining arguments of `std_index()` are only applicable when `x_new` and `x_ref` are time series, rather than vectors, since they all involve manipulations based on the date of the data. For example, the argument `moving_window` can be used to calculate standardised indices using a moving window. In this case, `x_ref` is updated for each observation in `x_new` so that it contains the previous `moving_window` (an integer) observations in the time series; the indices are therefore determined relative to recently observed values. The argument `window_scale` specifies the timescale that `moving_window` refers to. For example, setting `moving_window = 10` and `window_scale = "days"` would calculate standardised indices using a moving window of length 10 days.

If `moving_window` is specified but `window_scale = NULL`, then it is assumed that the units of `moving_window` correspond to the timescale of `x_new`. If the time difference between consecutive

observations is consistent throughout the time series, then this can be determined automatically within `std_index()`. Alternatively, and more robustly, it can be specified by the user using the `timescale` argument. This (and `window_scale`) must be one of `"mins"`, `"hours"`, `"days"`, `"weeks"`, `"months"`, or `"years"`.

If the original time series contains, say, hourly data, but the time series of standardised indices is desired on a different timescale, then the `rescale` argument can be used to rescale the data. This must also be one of the timescales listed above. By default, `std_index()` assumes the sum of the values should be returned when rescaling. For example, if interest is on precipitation accumulations, then converting hourly data to daily data will return the daily aggregations. However, alternative rescaling could also be performed using the `rescale_fun` argument, which is a function specifying what operation to perform over the aggregated data. If `rescale_fun = "mean"` or `rescale_fun = "max"`, for example, then the daily (or weekly, monthly, etc) mean or maximum would be provided, respectively, in place of the hourly data.

Similarly, the `agg_period` argument can be used to aggregate data across multiple time steps. Like `moving_window` and `window_scale`, `agg_period` is an integer representing the number of previous time steps over which the data should be aggregated, while `agg_scale` specifies the timescale of the aggregation period, which is by default assumed to be the timescale of the data. The argument `agg_fun` represents the function used to aggregate the data over the aggregation period.

This differs from `rescale` in the following way. If we want to convert a time series of daily observations to a time series of weekly indices, then we can use `rescale = "weeks"` (and `timescale = "days"`). On the other hand, if we set `agg_period = 1` and `agg_scale = "weeks"`, then we get a daily time series of weekly aggregated data.

To perform the rescaling, `std_index()` calls one of the **xts** functions `apply.daily`, `apply.weekly`, `apply.monthly`, `apply.quarterly`, and `apply.yearly`, alongside the argument `rescale_fun`. To perform the aggregation, `std_index()` uses the function `aggregate_xts()`, which is additionally exported by **SEI**.

```
aggregate_xts(
  x,
  agg_period = 1,
  agg_scale = c("days", "mins", "hours", "weeks", "months", "years"),
  agg_fun = "sum",
  timescale = c("days", "mins", "hours", "weeks", "months", "years"),
  na_thres = 10
)
```

The interpretation of the arguments is equivalent to in the discussion above: `x` represents the **xts** time series to be aggregated, `agg_period` and `agg_scale` are the length and time scale of the aggregation period, `agg_fun` is the function used in the aggregation, and `timescale` is the time scale of `x`. The final argument `na_thres` represents the maximum proportion of values that can be missing (i.e. `NA`) in the aggregation period, before the aggregation itself returns `NA`. For example, if 23 hours of precipitation accumulations are missing on one day so that only one hourly accumulation is available, this sole observation is not representative of the daily accumulation. By default, `NA` is returned if more than 10% of the values in the aggregation period are missing.

### 4.6 Plotting standardised indices

To visualise the indices, **SEI** includes the `plot_sei()` function.

```
plot_sei(
  x,
  type = c("ts", "hist", "bar"),
  title = NULL,
  lab = "Std. Index",
  xlims = NULL,
  ylims = NULL,
  n_bins = 30
)
```

The argument `x` is either a vector or an **xts** time series object that contains the index values to be displayed. This function can either be used to plot a time series of the values (`type = "ts"`), a histogram (`type = "hist"`), or a bar plot (`type = "bar"`). If `type = "hist"`, the function is essentially a

wrapper for `geom_histogram()` in **ggplot2**. The histogram and bar plot both display the distribution of the values in `x`. The histogram is particularly useful when superimposing a density onto the histogram, while the bar plot is also made available since it can be more robust when the data in `x` is bounded. Here, `n_bins` can be used to specify the number of bins in the plot.

Additional aspects of the plot can be specified using `title`, `lab`, `xlims` and `ylims`. For the time series plot, `lab` refers to the label of the y-axis, whereas it corresponds to the x-axis label of the histogram and bar plot.

### 4.7 Drought definitions

Having obtained a time series of standardised indices, one might wish to perform an analysis of shortages, or droughts, in the variable of interest. As discussed, this can be achieved using standardised indices with the definitions of droughts in Table 1. To facilitate such analyses, the `get_drought()` function takes a time series or vector `x` as input, and outputs a dataframe containing information regarding drought occurrences and characteristsics.

```
get_drought(
  x,
  thresholds = c(1.28, 1.64, 1.96),
  exceed = TRUE,
  cluster = 0,
  lag = NULL
)
```

The argument `thresholds` is a vector containing the thresholds used to define the droughts; by default, these correspond to the $90^{th}$, $95^{th}$, and $97.5^{th}$ percentiles of a standard normal distribution. It is also assumed by default that a drought is defined when the standardised indices exceed these thresholds. This can be specified using the `exceed` argument. If a drought should instead correspond to an instance where the indices fall below the given thresholds, `exceed` should be `FALSE`.

When defining meteorological droughts, it is often common to introduce a lag such that the drought does not end when the standardised index no longer exceeds the drought threshold, but rather when the index falls below a lower threshold (such as zero for the SPI) (e.g. McKee et al., 1993). This accounts for cases where there are small fluctuations in the index around the drought threshold, classing this as one persistent drought rather than several shorter droughts. This can be implemented in **SEI** by specifying the `lag` argument, which is a numeric value denoting the weaker threshold to use when lagging the indices.

Alternatively, these fluctuations could be avoided by clustering together drought events separated by a small number of days (e.g. Otero et al., 2022a). The number of days between which drought events should be clustered can be specified via the `cluster` argument in `get_drought()`. The default is to not cluster droughts.

As an example, consider the previous time series of standardised indices corresponding to a random sample from a gamma distribution.

```
head(get_drought(x_std))
```

```
#>             x ins occ dur mag
#> 1 -0.54344079   0   0   0   0
#> 2  0.06007512   0   0   0   0
#> 3  0.07511934   0   0   0   0
#> 4 -0.60540462   0   0   0   0
#> 5  0.25024846   0   0   0   0
#> 6 -0.29438586   0   0   0   0
```

The output is a dataframe containing the original vector or time series, as well as four additional columns: the intensity (`ins`), which corresponds to the category of drought, a higher value referring to a more severe drought category; the occurrence (`occ`), which corresponds to the intensity being higher than zero; the duration (`dur`), which provides the number of consecutive time steps that are in a drought state; and the drought magnitude (`mag`), which is the sum of all indices within the drought event (see McKee et al., 1993, for details). If only one threshold is used to define a drought, rather than the three used in Table 1, then the intensity is equivalent to the occurrence, and is therefore not returned.

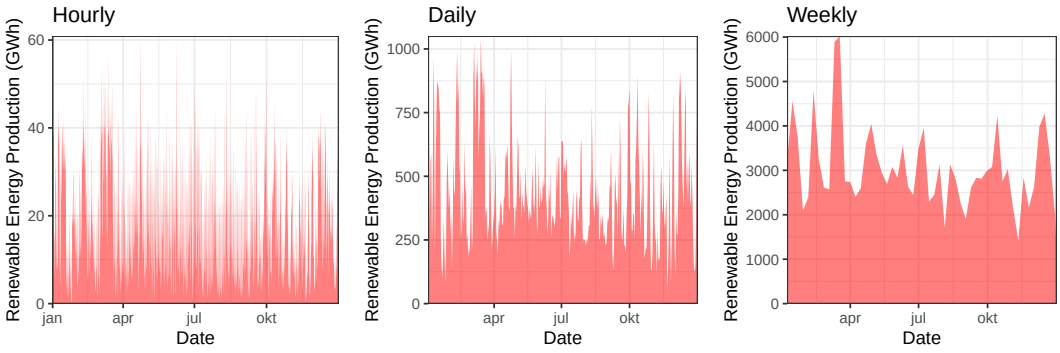



**Figure 3:** Time series of 2019 renewable energy production in Germany at hourly, daily, and weekly time scales.

## 5 Applications

### 5.1 Renewable energy production

Consider an application of standardised indices to renewable energy production in Europe. Hourly time series of wind and solar power generation are publicly available for 27 European countries at https://researchdata.reading.ac.uk/275/ (see Bloomfield et al., 2020, for details). A subset of this data corresponding to production in 2019 can be accessed from **SEI** using

```
data("data_supply", package = "SEI")
head(data_supply)
```

```
#>                  date country        PWS
#> 1 2019-01-01 00:00:00 Austria 0.07823861
#> 2 2019-01-01 01:00:00 Austria 0.05330285
#> 3 2019-01-01 02:00:00 Austria 0.06773745
#> 4 2019-01-01 03:00:00 Austria 0.09971320
#> 5 2019-01-01 04:00:00 Austria 0.18676895
#> 6 2019-01-01 05:00:00 Austria 0.29001299
```

The dataframe `data_supply` contains a `POSIXct` time series of dates, along with the corresponding country and wind and solar power production (`PWS`). The units of the production are Gigawatt hours (GWh).

For concision, we restrict attention here to renewable energy production in Germany.

```
de_supply_h <- subset(data_supply, country == "Germany")
de_supply_h <- xts::xts(de_supply_h$PWS, de_supply_h$date) # convert to xts
```

This can be rescaled from hourly to daily or weekly time series using **xts** functionality.

```
de_supply_d <- xts::apply.daily(de_supply_h, "sum")    # daily data
de_supply_w <- xts::apply.weekly(de_supply_h, "sum")   # weekly data
```

These time series of renewable energy production in Germany can be visualised using the `plot_sei()` function. Figure 3 demonstrates that all three time series display the same patterns, though the weekly time series removes the hourly and daily fluctuations.

```
lab <- "Renewable Energy Production (GWh)"
plot_h <- plot_sei(de_supply_h, lab = lab, title = "Hourly")
plot_d <- plot_sei(de_supply_d, lab = lab, title = "Daily")
plot_w <- plot_sei(de_supply_w, lab = lab, title = "Weekly")
grid.arrange(plot_h, plot_d, plot_w, nrow = 1)
```

The raw renewable energy production values can be transformed to a standardised index using `std_index()`.

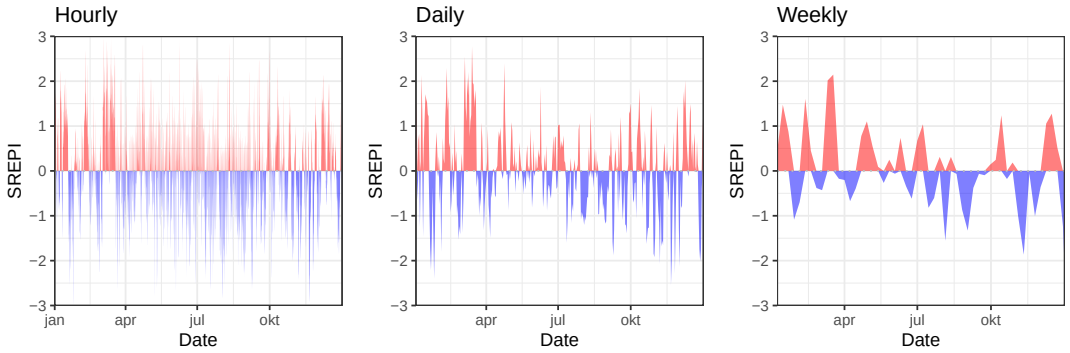



**Figure 4:** Time series of 2019 SREPI values in Germany at hourly, daily, and weekly time scales.

```
srepi_h <- std_index(de_supply_h)
srepi_d <- std_index(de_supply_d)
srepi_w <- std_index(de_supply_w, dist = "kde")
```

Following, Allen and Otero (2023), we refer to this as the standardised renewable energy production index (SREPI). Time series of hourly, daily, and weekly SREPI values are presented in Figure 4.

```
lab <- "SREPI"
ylims <- c(-3, 3)
plot_h <- plot_sei(srepi_h, lab = lab, ylims = ylims, title = "Hourly")
plot_d <- plot_sei(srepi_d, lab = lab, ylims = ylims, title = "Daily")
plot_w <- plot_sei(srepi_w, lab = lab, ylims = ylims, title = "Weekly")
grid.arrange(plot_h, plot_d, plot_w, nrow = 1)
```

In this case, the reference data `x_ref` is assumed to be the input time series itself, meaning the indices are calculated relative to this input data. By default, `std_index()` returns normal indices (rather than probability or bounded indices), and estimates the distribution of the renewable energy production using the empirical distribution. This returns a warning for the weekly data, since there are fewer than 100 weekly production values in the time series. Instead, for weekly data, we estimate the distribution of production values using kernel density estimation (`dist = "kde"`). Alternative distributions could be implemented by changing the `dist` argument in `std_index()`.

While these indices are created by standardising the respective hourly, daily, and weekly time series of raw production values, they could all be obtained from the original hourly time series by specifying the `rescale` argument in `std_index()`.

```
z <- std_index(de_supply_h, rescale = "days")
all.equal(srepi_d, z)

#> [1] TRUE
```

By default, the `plot_sei()` function displays the time series of input values. However, by specifying `type = "hist"` (rather than the default `type = "ts"`), this function can additionally be used to plot a histogram of the input values. An example of this for the daily data production and corresponding SREPI values is presented in Figure 5. As discussed in Section 3, such a plot can be used to validate the distributional assumptions made when calculating the indices. While the raw renewable energy production follows a heavily skewed distribution, the SREPI values resemble a sample from a standard normal distribution, suggesting the empirical distribution function is appropriate in this example. If probability or bounded indices were used, then the histogram of SREPI values should be flat, rather than normal. This is illustrated in Figure 6.

Finally, the **SEI** package additionally allows shortages or drought characteristics of the time series to be assessed. In this case, an energy supply drought could occur if the renewable energy production is low compared to previously observed values (Allen and Otero, 2023). This corresponds to a low SREPI. As in Table 1, we define three categories of droughts, each defined using increasing thresholds of the SREPI. The function `get_drought()` can then be used to obtain a time series of drought occurrences, intensities, durations, and magnitudes.

```
thresholds <- qnorm(c(0.1, 0.05, 0.025)) # -1.28, -1.64, -1.96
drought_df <- get_drought(srepi_d, thresholds, exceed = F)
```

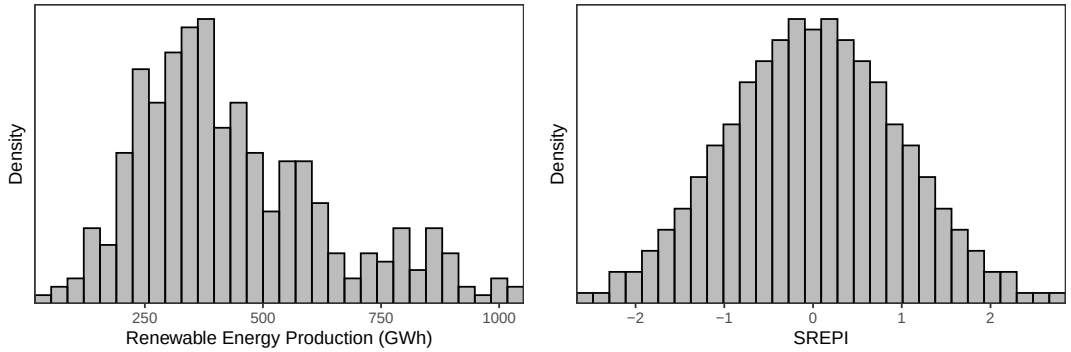



**Figure 5:** Histogram of 2019 daily renewable energy production and SREPI values in Germany.

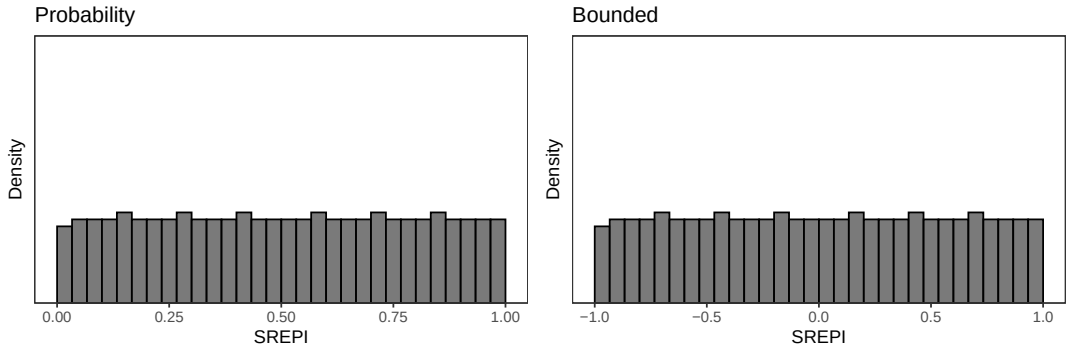



**Figure 6:** Histogram of 2019 daily renewable energy production and SREPI values in Germany, where the indices are probability and bounded indices.

From the dataframe `drought_df`, it is straightforward to analyse the properties of these drought events. For example, we can list the frequency that a moderate, severe, or extreme drought event occurs

```
num_ev <- table(drought_df$ins)
names(num_ev) <- c("None", "Moderate", "Severe", "Extreme")
print(num_ev)

#>     None Moderate   Severe  Extreme
#>      330       18        9        8
```

we could display the relative frequency of drought durations

```
table(drought_df$dur[drought_df$dur > 0])

#>
#>  1  2  3
#> 11  3  6
```

or we could calculate the average drought magnitude

```
mean(drought_df$mag[drought_df$mag != 0])

#> [1] 2.985504
```

These are just simple examples of analyses that could be performed using the output of `get_drought()`. For a more thorough analysis, readers are referred to Allen and Otero (2023).

### 5.2 Wind speed

Consider now a second application of standardised indices to weather and climate variables. While standardised indices are most commonly used to construct the SPI and SPEI, they can also be applied

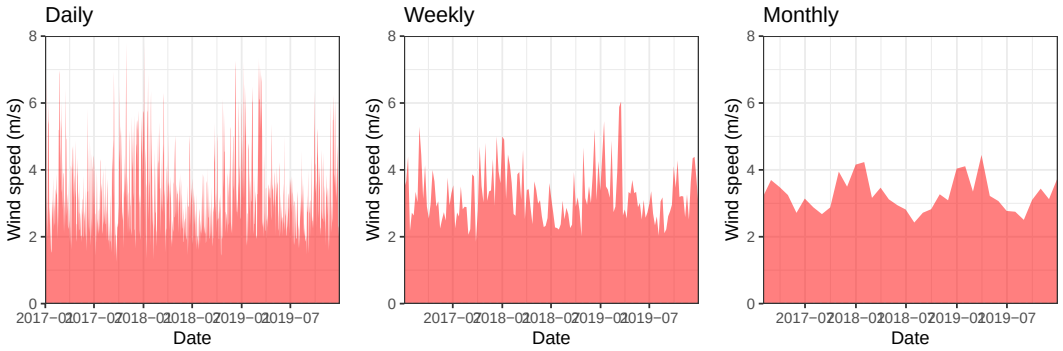



**Figure 7:** Daily, weekly and monthly time series of average wind speed in Germany between 2017 and 2019.

to other climate variables (see Section 1 for examples). In this section, we use the framework discussed herein to define standardised wind speed indices. Monitoring wind speed is crucial in many applications, including aviation safety, renewable energy planning, and the design and insurance of infrastructure.

Hourly estimates of near-surface (10m) wind speed are publicly available for several European countries between 1979 and 2019 (Hersbach et al., 2023). As in Otero et al. (2022b), we aggregate these wind speeds to get daily nationwide averages. Consistent with the previous application to renewable energy production, the analysis is restricted to Germany, and this subset of data is available from **SEI**. For clarity of visualisation, the following analysis is limited to the three year period between 2017 and 2019.

```
data("data_wind_de", package = "SEI")
data_wind_de <- subset(data_wind_de, format(date, "%Y") >= 2017)
head(data_wind_de)
```

```
#>             date   wsmean
#> 13881 2017-01-01 3.461528
#> 13882 2017-01-02 3.479744
#> 13883 2017-01-03 5.391327
#> 13884 2017-01-04 6.801471
#> 13885 2017-01-05 3.822558
#> 13886 2017-01-06 2.101849
```

As before, the data can be rescaled using **xts** functionality to obtain time series of the weekly and monthly average wind speed in Germany. We label the resulting daily, weekly, and monthly time series `de_wind_d`, `de_wind_w`, and `de_wind_m`, respectively, all of which are **xts** objects. These time series are displayed using `plot_sei()` in Figure 7.

```
lab <- "Wind speed (m/s)"
ylims <- c(0, 8)
plot_ws_d <- plot_sei(de_wind_d, lab = lab, ylims = ylims, title = "Daily")
plot_ws_w <- plot_sei(de_wind_w, lab = lab, ylims = ylims, title = "Weekly")
plot_ws_m <- plot_sei(de_wind_m, lab = lab, ylims = ylims, title = "Monthly")
grid.arrange(plot_ws_d, plot_ws_w, plot_ws_m, nrow = 1)
```

The raw data can be transformed to a time series of standardised indices using `std_index()`. As discussed, constructing standardised indices requires estimating the distribution of the variable of interest. While we argue that the empirical distribution provides a flexible means to achieve this, we demonstrate in this application how the **SEI** package allows parametric distributions to be used for this purpose.

Various distributions can be fit to the daily wind speeds using the `fit_dist()` function. Consider, for example, a gamma distribution, log-normal distribution, and Weibull distribution.

```
out_gamma <- fit_dist(data_wind_de$wsmean, dist = "gamma")
out_lnorm <- fit_dist(data_wind_de$wsmean, dist = "lnorm")
out_weibull <- fit_dist(data_wind_de$wsmean, dist = "weibull")
```

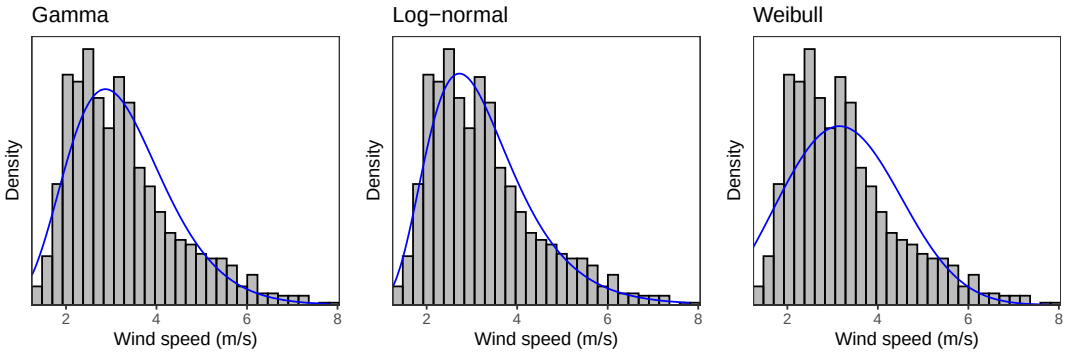



**Figure 8:** Distribution of daily wind speeds in Germany between 2017 and 2019, along with estimated gamma, log-normal, and Weibull densities.

The three distributions can be compared with respect to their AIC, which is output from `fit_dist()`.

```
aic_vec <- c(out_gamma$fit['aic'],
             out_lnorm$fit['aic'],
             out_weibull$fit['aic'])
names(aic_vec) <- c("Gamma", "Log-normal", "Weibull")
print(aic_vec)

#>      Gamma Log-normal    Weibull
#>   3278.268   3231.083   3443.914
```

The AIC provides a measure of distributional fit whilst also accounting for model complexity, with a smaller value indicating a better bit. The AIC values above suggest that the log-normal distribution provides the best fit to the data, while the Weibull distribution provides the worst fit.

This can be verified by looking at p-values corresponding to the Kolmogorov-Smirnov test.

```
ksp_vec <- c(out_gamma$fit['ks_pval'],
             out_lnorm$fit['ks_pval'],
             out_weibull$fit['ks_pval'])
names(ksp_vec) <- c("Gamma", "Log-normal", "Weibull")
print(round(ksp_vec, 4))

#>      Gamma Log-normal    Weibull
#>     0.0047     0.1127     0.0000
```

The null hypothesis is that the sample of data has been drawn from the parametric distribution. The Kolmogorov-Smirnov test is rejected at the 0.5% level for both the gamma and Weibull distributions, providing evidence to suggest that these distributions do not accurately describe the data. On the other hand, there is insufficient evidence to reject the null hypothesis that the wind speed data was drawn from a log-normal distribution.

The fit of the three distributions to the data is presented in Figure 8. The `plot_sei()` function can be used to plot histograms of the daily wind speeds. Since `plot_sei()` returns a **ggplot2** object, the estimated density functions can be added to the histograms. For example, for the gamma distribution

```
x <- seq(0, 9, length.out = length(de_wind_d))
xlab <- "Wind speed (m/s)"
pars_gam <- out_gamma$params

plt_gam <- plot_sei(de_wind_d, type = "hist", lab = xlab, title = "Gamma") +
  geom_line(aes(x = x, y = dgamma(x, pars_gam[1], pars_gam[2])), col = "blue")
```

Finally, the adequacy of the distributions can also be visualised by plotting a histogram of the standardised indices obtained using the three distributions. The standardised indices can be calculated using the `std_index()` function, with the `dist` argument corresponding to the distribution of interest. When probability indices are used (`index_type = "prob01"`), plotting a histogram of the indices is equivalent to displaying the probability integral transform (PIT) histogram commonly used to assess the fit of probabilistic models.

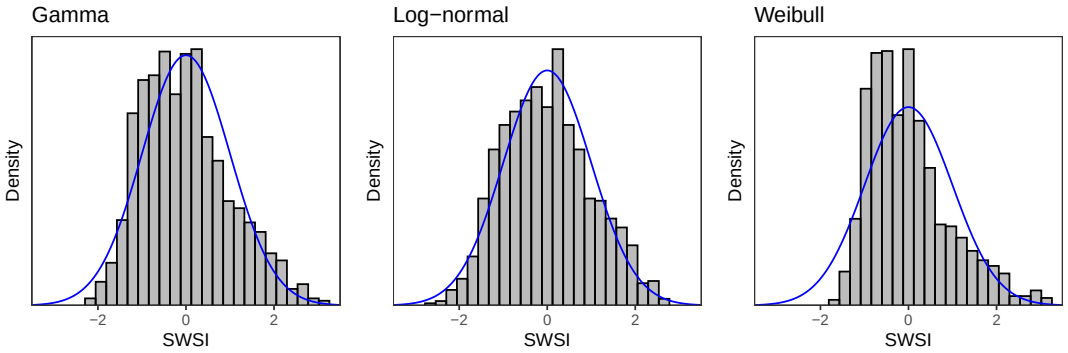



**Figure 9:** Distribution of standardised wind speed indices (SWSI) constructed using the gamma, log-normal, and Weibull distributions. The standard normal density is displayed in blue.

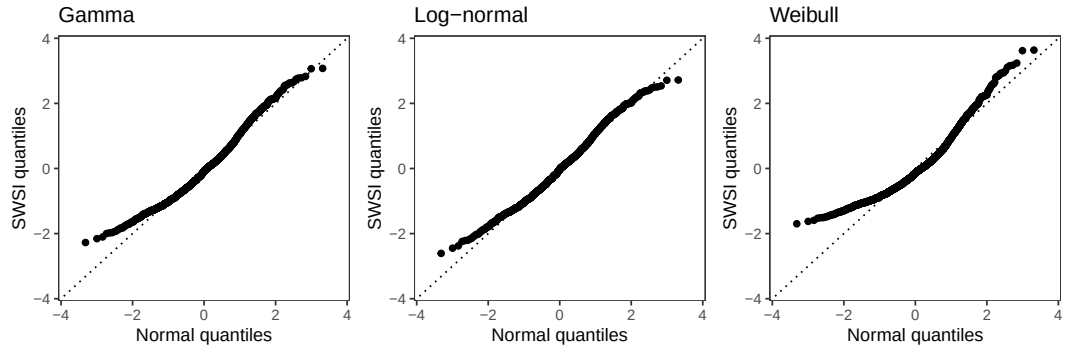



**Figure 10:** Normal quantile-quantile (qq) plots showing the fit of the standard normal distribution to the distribution of SWSI values.

```
sei_ws_gam <- std_index(de_wind_d, dist = "gamma")
sei_ws_lnorm <- std_index(de_wind_d, dist = "lnorm")
sei_ws_weib <- std_index(de_wind_d, dist = "weibull")
```

Histograms of the standardised indices, constructed using `plot_sei()`, are displayed in Figure 9. Similarly to before, a standard normal distribution is superimposed onto the histograms.

As expected, no parametric distribution fits the data perfectly. The standardised wind speed indices constructed using a log-normal distribution appear closer to a standard normal distribution, and this is reinforced by quantile-quantile plots in Figure 10, which compare the quantiles of the SWSI values to the quantiles of a standard normal distribution. This supports the conclusion that the log-normal distribution provides a more accurate fit to the daily wind speed data, which also agrees with previous studies that use this distribution when modelling wind speed (e.g. Kollu et al., 2012; Baran and Lerch, 2015).

Having constructed these wind speed indices, it would be straightforward to conduct an analysis of wind droughts (or extreme wind speeds) using the `get_drought()` function in **SEI**, akin to how renewable energy production droughts were studied in the previous application. This is omitted here for concision, but could be analysed in more detail in the future.

## 6 Summary

This paper documents the **SEI** package in R, which provides comprehensive functionality to calculate time series of standardised indices. Standardised indices are projections of variables onto an interpretable and probabilistically meaningful scale, and they have become popular tools when monitoring variables of interest over time, particularly in the context of drought analysis. While there is no unique way to define standardised indices, several widely-adopted indices are constructed using the same, simple procedure. We outline this general framework for constructing standardised indices, and reiterate that this approach can be applied to any variable of interest, not just those previously considered in the literature.

The **SEI** package converts a time series of observations to a time series of standardised indices. The package allows the time series to be aggregated and rescaled to different time scales, provides plot

capabilities to visualise the resulting indices, and offers a range of distributions with which to calculate the indices, including flexible non-parametric and non-stationary methods. The package additionally allows users to define and analyse shortages, or droughts, of the variable under consideration. Two examples are presented in Section 5 whereby the package is employed to calculate standardised indices to monitor shortages in renewable energy production, and to calculate standardised wind speed indices.

The package has been designed to facilitate applications of standardised indices in practice. While standardised indices have received considerable attention within the field of climate science, such indices would also be relevant in several other domains. The **SEI** package is applicable in broad generality, though the package could also be extended in light of more recent extensions of standardised indices that have been proposed in the literature. For example, the package currently only considers indices corresponding to univariate variables, though multivariate extensions have also been studied in the literature (e.g. Erhardt and Czado, 2018; Hao et al., 2019). Similarly, functionality could also be added to automatically implement methods that remove seasonality and temporal dependence in the data (Erhardt and Czado, 2018).

Moreover, while the package extends existing packages by providing flexible density estimation methods, alternative such methods could also be made available, particularly methods that are appropriate when estimating the distribution of bounded variables. Non-stationary distribution estimation is currently only available for parametric distributions via the GAMLSS framework, and further extensions could therefore incorporate non-parametric distributional regression methods such as isotonic distributional regression (Henzi et al., 2021). Finally, the functionality of the package could be transferred to software packages in other programming languages, facilitating the implementation of standardised indices in monitoring systems that have not been designed in R.

## 7 Acknowledgements

This package was created under funding by the Oeschger Centre for Climate Change Research and the Swiss Federal Office of Meteorology and Climatology (MeteoSwiss).

## 8 Appendix: Censored data

The definition of standardised indices relies on the well-known result that if $X$ is a continuous random variable with cumulative distribution function $F$, then the probability integral transform $F(X)$ is uniformly distributed between zero and one. If the random variable $X$ is not continuous, then the PIT will generally not be uniformly distributed. This becomes an issue when calculating standardised indices. Of particular interest is the case when the variable $X$ is censored; that is, the distribution of $X$ has a point mass at the boundary (or boundaries) of its support, but is continuous otherwise. Precipitation, for example, is censored below at zero.

While standardised indices for censored variables can be defined as in Section 2, Stagge et al. (2015) remark that the resulting indices may be misleading. For example, if $X$ denotes daily precipitation accumulations and no precipitation occurs on 90% of days in the time series, then $F(0) = 0.9$. The (normal) standardised index value corresponding to a precipitation of zero would therefore be $\Phi^{-1}(0.9) = 1.28$, the threshold often used to define excessively large values of a variable, despite zero precipitation clearly not being extreme; similarly, if the smallest precipitation value attains a standardised index of 1.28, then the system will never enter a drought state according to the drought definitions typically employed in practice, regardless of what precipitation occurs.

To circumvent this, Stagge et al. (2015) propose to replace the PIT values corresponding to zero precipitation events with an alternative constant, chosen such that the resulting standardised indices "maintain statistical interpretability". In particular, they propose to choose the constant so that the mean of the standardised indices is the same as it would be if $X$ were continuous (zero, in the case of normal indices).

We can formalise this as follows. Suppose we are interested in a random variable $X$ that is censored below at some value $a$, precipitation accumulations being a particular example with $a = 0$. While standardised indices are typically defined using $F(X)$, we want to introduce a real-valued constant $c$ and define an adapted variable, $\tilde{F}(X)$, such that

$$\tilde{F}(X) = \begin{cases} F(X), & X > a, \\ c, & X = a. \end{cases}$$

Note that for $X > a$, $F(X)$ can also be written as $F(X) = F(a) + (1 - F(a))F(X \mid X > a)$, which is

equivalent to the expression given in Equation 4 of Stagge et al. (2015).

For probability standardised indices (`index_type = 'prob01'`), the constant should be chosen such that $\mathsf{E}[\tilde{F}(X)] = 1/2$, since the expectation $\mathsf{E}$ of a standard uniform random variable is equal to one half. We have

$$\mathsf{E}[\tilde{F}(X)] = \mathsf{E}[F(X) \mid X > a]\mathsf{P}(X > a) + c\mathsf{P}(X = a).$$

Since $F(X) \mid X > a$ is uniformly distributed between $\mathsf{P}(X = a)$ and 1, the first expectation is equal to $(\mathsf{P}(X = a) + 1)/2$. Rearranging gives that $\mathsf{E}[\tilde{F}(X)] = 1/2$ when $c = \mathsf{P}(X = a)/2$. Hence, probability indices at the censored values should be replaced with $c = \mathsf{P}(X = a)/2$.

This is also the case for bounded standardised indices (`index_type = 'prob11'`). In this case, we want to choose $c$ such that $\mathsf{E}[2\tilde{F}(X) - 1] = 0$, which is exactly the same requirement as for the probability indices. Using analogous arguments, if $X$ is censored above at some value $b$ (rather than below at some value $a$), then the probability and bounded indices at the censored values should be replaced with $c = 1 - \mathsf{P}(X = b)/2$.

The constant proposed by Stagge et al. (2015) Equation 3 is a sample estimate of $\mathsf{P}(X = a)/2$. However, since the Gaussian inverse function $\Phi^{-1}$ is a non-linear function, for an arbitrary random variable $Y$ generally $\mathsf{E}[\Phi^{-1}(Y)] \neq \Phi^{-1}(\mathsf{E}[Y])$. Hence, normal standardised indices will generally not have expectation zero when the constant $c$ is chosen as above; this is also highlighted by Figure 1b in Stagge et al. (2015), where the mean of the normal standardised indices with their proposed censoring methodology is generally not zero.

Instead, we propose an alternative constant when normal standardised indices are used, which results in a mean index value equal to zero. We now want to find the constant $c$ such that

$$\mathsf{E}[\Phi^{-1}(\tilde{F}(X))] = \mathsf{E}[\Phi^{-1}(F(X)) \mid X > a]\mathsf{P}(X > a) + \Phi^{-1}(c)\mathsf{P}(X = a) = 0.$$

Since $F$ and $\Phi^{-1}$ are increasing functions, we have that

$$\mathsf{E}[\Phi^{-1}(F(X)) \mid X > a] = \mathsf{E}[\Phi^{-1}(F(X)) \mid \Phi^{-1}(F(X)) > \Phi^{-1}(F(a))],$$

which is just the expectation of a normal distribution truncated below at $l = \Phi^{-1}(F(a)) = \Phi^{-1}(\mathsf{P}(X = a))$. This is therefore equal to

$$\frac{\phi(l)}{1 - \Phi(l)},$$

where $\phi$ is the probability density function of the standard normal distribution. Similarly, $\mathsf{P}(X > a) = \mathsf{P}(\Phi^{-1}(F(X)) > l) = 1 - \Phi(l)$. Rearranging the above equality then yields

$$c = \Phi\left[\frac{-\phi(l)}{\mathsf{P}(X = a)}\right],$$

and the normal indices at the censored values should therefore be replaced with $\Phi^{-1}(c) = -\phi(l)/\mathsf{P}(X = a)$. If $X$ is instead bounded above at $b$, then the constant $c$ becomes

$$c = \Phi\left[\frac{\phi(u)}{\mathsf{P}(X = b)}\right],$$

where $u = \Phi^{-1}(1 - \mathsf{P}(X = b))$.

In practice, we observe a sample of realisations of the variable $X$. The terms $\mathsf{P}(X = a)$ and $\mathsf{P}(X = b)$ can be estimated from a sample using the relative number of realisations that are censored, and these can then be used to determine the constant $c$. Note that since these terms have been calculated on a population level, the sample mean of the standardised indices will generally not be exactly zero (or 1/2 for probability indices), but it will be close to zero in large enough samples.

Variables could also be censored both below at $a$ and above at $b$. For example, the fraction of cloud cover over a domain is bounded between $a = 0$ and $b = 1$, with positive probability of taking on either of the boundary values. In this case, we must choose how to assign values at both boundary points. This adds an extra degree of freedom, since there are infinite combinations of two points that would lead to the standardised indices having mean zero (or mean one half for probability indices). A second criterion could be introduced, such as the variance of the indices being equal to one (or 1/12). However, such calculations go beyond the scope of this paper, and we therefore leave this for future work.

To implement censoring within the **SEI** package, the arguments `lower` and `upper` to the functions `std_index()` and `get_pit()` allow the user to indicate that the data is censored below at $a$ (`lower`) and/or above at $b$ (`upper`). The argument `cens` then specifies the method used to deal with censoring. This can be one of four options: if `cens = 'none'`, then no censoring is performed, i.e. $\tilde{F} = F$; if `cens = 'prob'`, then the approach above is used that results in the probability indices having mean equal to

1/2, which is essentially the same as the original approach proposed by Stagge et al. (2015); if `cens = 'normal'`, then the approach above is used that results in the normal indices having mean equal to zero; alternatively, `cens` can be a single numeric value, allowing the user to manually specify the constant $c$. As discussed above, we recommend using `cens = 'prob'` when working with probability or bounded standardised indices, and `cens = 'normal'` when normal standardised indices are used. If the variable is bounded both below and above (i.e. `lower` and `upper` are both finite), then `cens` must be a numeric vector of length two, specifying the constants at which to assign values at the lower and upper boundary points, respectively.

## References


S. Allen and N. Otero. Standardised indices to monitor energy droughts. *Renewable Energy*, 217:119206, 2023. doi: 10.1016/j.renene.2023.119206. [p102, 104, 105, 113, 114]

I. Ayala-Bizarro and J. Zuniga-Mendoza. **SPIGA***: Compute SPI Index Using the Methods Genetic Algorithm and Maximum Likelihood*, 2016. URL https://CRAN.R-project.org/package=SPIGA. R package version 1.0.0. [p102]

S. Baran and S. Lerch. Log-normal distribution based ensemble model output statistics models for probabilistic wind-speed forecasting. *Quarterly Journal of the Royal Meteorological Society*, 141: 2289–2299, 2015. doi: 10.1002/qj.2521. [p117]

S. Begueria, S. M. Vicente-Serrano, F. Reig, and B. Latorre. Standardized precipitation evapotranspiration index (SPEI) revisited: Parameter fitting, evapotranspiration models, tools, datasets and drought monitoring. *International Journal of Climatology*, 34:3001–3023, 2014. doi: 10.1002/joc.3887. [p102]

S. Beguería and S. M. Vicente-Serrano. **SPEI***: Calculation of the Standardized Precipitation-Evapotranspiration Index*, 2023. URL https://CRAN.R-project.org/package=SPEI. R package version 1.8.1. [p102]

H. Bloomfield, D. Brayshaw, and A. Charlton-Perez. ERA5 derived time series of european country-aggregate electricity demand, wind power generation and solar power generation: Hourly data from 1979-2019. 2020. doi: 10.17864/1947.272. [p112]

J. Bloomfield and B. Marchant. Analysis of groundwater drought building on the standardised precipitation index approach. *Hydrology and Earth System Sciences*, 17:4769–4787, 2013. doi: 10.5194/hess-17-4769-2013. [p102]

T. M. Erhardt and C. Czado. Standardized drought indices: A novel univariate and multivariate approach. *Journal of the Royal Statistical Society Series C*, 67:643–664, 2018. doi: 10.1111/rssc.12242. [p102, 104, 118]

L. Gudmundsson and J. H. Stagge. **SCI***: Standardized Climate Indices such as SPI, SRI or SPEI*, 2016. URL https://CRAN.R-project.org/package=SCI. R package version 1.0-2. [p102]

N. B. Guttman. Accepting the standardized precipitation index: A calculation algorithm. *Journal of the American Water Resources Association*, 35:311–322, 1999. doi: 10.1111/j.1752-1688.1999.tb03592.x. [p104]

Z. Hao, F. Hao, V. P. Singh, and X. Zhang. Statistical prediction of the severity of compound dry-hot events based on El Nino-Southern Oscillation. *Journal of Hydrology*, 572:243–250, 2019. doi: 10.1016/j.jhydrol.2019.03.001. [p102, 103, 104, 118]

A. Henzi, J. F. Ziegel, and T. Gneiting. Isotonic distributional regression. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 83:963–993, 2021. doi: 10.1111/rssb.12450. [p118]

H. Hersbach, B. Bell, P. Berrisford, G. Biavati, A. Horanyi, J. Munoz Sabater, J. Nicolas, C. Peubey, I. Radu, Rand Rozum, D. Schepers, A. Simmons, C. Soci, D. Dee, and J.-N. Thepaut. Era5 hourly data on single levels from 1940 to present. copernicus climate change service (c3s) climate data store (cds). https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-single-levels?tab=overview, 2023. Accessed: 01-09-2022. [p115]

T. Kneib, A. Silbersdorff, and B. Säfken. Rage against the mean–a review of distributional regression approaches. *Econometrics and Statistics*, 26:99–123, 2023. doi: 10.1016/j.ecosta.2021.07.006. [p106]

R. Kollu, S. R. Rayapudi, S. Narasimham, and K. M. Pakkurthi. Mixture probability distribution functions to model wind speed distributions. *International Journal of Energy and Environmental Engineering*, 3:1–10, 2012. doi: 10.1186/2251-6832-3-27. [p117]

J. Li, Y. Wang, S. Li, and R. Hu. A nonstationary standardized precipitation index incorporating climate indices as covariates. *Journal of Geophysical Research: Atmospheres*, 120:12–082, 2015. doi: 10.1002/2015JD023920. [p104, 106]

J. Li, Z. Wang, X. Wu, J. Zscheischler, S. Guo, and X. Chen. A standardized index for assessing sub-monthly compound dry and hot conditions with application in China. *Hydrology and Earth System Sciences*, 25:1587–1601, 2021. doi: 10.5194/hess-25-1587-2021. [p102]

W. Maetens. **standaRdized**, 2019. URL https://github.com/WillemMaetens/standaRdized. GitHub repository release v.1.0. [p102]

F. J. Massey Jr. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46:68–78, 1951. doi: 10.1080/01621459.1951.10500769. [p106]

T. B. McKee, N. J. Doesken, and J. Kleist. The relationship of drought frequency and duration to time scales. In *Proceedings of the 8th Conference on Applied Climatology*, volume 17, pages 179–183. Boston, MA, USA, 1993. [p102, 104, 111]

E. Nussbaumer. **standard-precip***: Functions to Calculate SPI and SPEI*, 2021. URL https://pypi.org/project/standard-precip/. Python package version 1.0. [p102]

N. Otero, O. Martius, S. Allen, H. Bloomfield, and B. Schaefli. A copula-based assessment of renewable energy droughts across europe. *Renewable Energy*, 201:667–677, 2022a. doi: 10.1016/j.renene.2022.10.091. [p111]

N. Otero, O. Martius, S. Allen, H. Bloomfield, and B. Schaefli. Characterizing renewable energy compound events across europe using a logistic regression-based approach. *Meteorological Applications*, 29:e2089, 2022b. doi: 10.1002/met.2089. [p115]

Python. *Python Software, Version 3.6.4*. Beaverton, OR, 2017. URL https://www.python.org/. [p102]

M. M. Rashid and S. Beecham. Development of a non-stationary standardized precipitation index and its application to a south australian climate. *Science of the Total Environment*, 657:882–892, 2019. doi: 10.1016/j.scitotenv.2018.12.052. [p106]

R. A. Rigby and D. M. Stasinopoulos. Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 54:507–554, 2005. doi: 10.1111/j.1467-9876.2005.00510.x. [p106]

S. Russo, A. Dosio, A. Sterl, P. Barbosa, and J. Vogt. Projection of occurrence of extreme dry-wet years and seasons in europe with stationary and nonstationary standardized precipitation indices. *Journal of Geophysical Research: Atmospheres*, 118:7628–7639, 2013. doi: 10.1002/jgrd.50571. [p104, 105]

S. Shao, H. Zhang, V. P. Singh, H. Ding, J. Zhang, and Y. Wu. Nonstationary analysis of hydrological drought index in a coupled human-water system: Application of the gamlss with meteorological and anthropogenic covariates in the wuding river basin, china. *Journal of Hydrology*, 608:127692, 2022. doi: 10.1016/j.jhydrol.2022.127692. [p106]

S. Shukla and A. W. Wood. Use of a standardized runoff index for characterizing hydrologic drought. *Geophysical Research Letters*, 35:L02405, 2008. doi: 10.1029/2007GL032487. [p102]

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Routledge, New York, 2018. doi: 10.1201/9781315140919. [p105]

J. H. Stagge, L. M. Tallaksen, L. Gudmundsson, A. F. Van Loon, and K. Stahl. Candidate distributions for climatological drought indices (SPI and SPEI). *International Journal of Climatology*, 35:4027–4040, 2015. doi: 10.1002/joc.4267. [p104, 109, 118, 119, 120]

D. M. Stasinopoulos and R. A. Rigby. Generalized additive models for location scale and shape (gamlss) in r. *Journal of Statistical Software*, 23:1–46, 2008. doi: 10.18637/jss.v023.i07. [p106]

M. D. Stasinopoulos, R. A. Rigby, G. Z. Heller, V. Voudouris, and F. De Bastiani. *Flexible Regression and Smoothing: Using GAMLSS in R*. CRC Press, 2017. doi: 10.1201/b21973. [p106]

S. M. Vicente-Serrano, S. Begueria, and J. I. Lopez-Moreno. A multiscalar drought index sensitive to global warming: The standardized precipitation evapotranspiration index. *Journal of Climate*, 23: 1696–1718, 2010. doi: 10.1175/2009JCLI2909.1. [p102]

S. M. Vicente-Serrano, J. I. Lopez-Moreno, S. Begueria, J. Lorenzo-Lacruz, C. Azorin-Molina, and E. Moran-Tejeda. Accurate computation of a streamflow drought index. *Journal of Hydrologic Engineering*, 17:318–332, 2012. doi: 10.1061/(ASCE)HE.1943-5584.0000433. [p102]

M. Vonk. **spei***: A Simple Python Package to Calculate Drought Indices for Time Series such as the SPI, SPEI and SGI*, 2017. URL https://pypi.org/project/spei/. Python package version 2.0.0. [p102]

Y. Wang, J. Li, P. Feng, and R. Hu. A time-dependent drought index for non-stationary precipitation series. *Water Resources Management*, 29:5631–5647, 2015. doi: 10.1007/s11269-015-1138-0. [p106]

A. Zargar, R. Sadiq, B. Naser, and F. I. Khan. A review of drought indices. *Environmental Reviews*, 19: 333–349, 2011. doi: 10.1139/a11-013. [p102]

J. Zscheischler, A. M. Michalak, C. Schwalm, M. D. Mahecha, D. N. Huntzinger, M. Reichstein, G. Berthier, P. Ciais, R. B. Cook, B. El-Masri, et al. Impact of large-scale climate extremes on biospheric carbon fluxes: An intercomparison based on MsTMIP data. *Global Biogeochemical Cycles*, 28:585–600, 2014. doi: 10.1002/2014GB004826. [p102]

*Sam Allen*
*ETH Zürich*
*Seminar for Statistics*
*Rämistrasse 101*
*8092 Zurich, Switzerland*
*ORCiD: 0000-0003-1971-8277*
sam.allen@stat.math.ethz.ch

*Noelia Otero*
*University of Bern and Oeschger Centre for Climate Change Research*
*Institute of Geography*
*Hallerstrasse 12*
*3012 Bern, Switzerland*
*ORCiD: 0000-0003-3217-3945*
noelia.otero@unibe.ch

# MultiStatM: Multivariate Statistical Methods in R

*by György Terdik and Emanuele Taufer*

**Abstract** The package MultiStatM presents a vectorial approach to multivariate moments and cumulants. Functions provided concern algorithms to build set partitions and commutator matrices, multivariate d-Hermite polynomials; theoretical vector moments and vector cumulants of multivariate distributions and their conversion formulae. Applications discussed concern multivariate measures of skewness and kurtosis; asymptotic covariances for d-variate Hermite polynomials and multivariate moments and cumulants; Gram-Charlier approximations.

## 1 Introduction and background

The package **MultiStatM** (Terdik and Taufer (2024)) provides a set of tools such as set partitions, commutator matrices and vector Hermite polynomials which are needed in developing general computational and conversion formulae for multivariate moments and cumulants of any order. Applications are provided for theoretical moments and cumulants of some important symmetric and skew-symmetric multivariate distributions; measures of multivariate skewness and kurtosis; asymptotic covariance formulae for vector Hermite polynomials; last but not least, estimation functions for all theoretical results discussed are provided. All the methods implemented in **MultiStatM** are discussed in detail in Terdik (2021).

A careful study of the cumulants is a necessary and typical part of nonlinear statistics. Dealing with cumulants for multivariate distributions is made complicated by the index notations: McCullagh (2018) provides quite an elegant approach using tensor methods; however, tensor methods are not very well known and computationally not so simple. Kollo (2008) and Móri et al. (1994) provide formulae for cumulants in terms of matrices; however, retaining a matrix structure for all higher-order cumulants leads to high-dimensional matrices with special symmetric structures which are quite hard to follow notionally and computationally. Cumulant matrices, with special attention to the third and fourth order, are ubiquitous in the literature. The use of decompositions and approximations allow for dimensionality reduction and find several statistical applications; see Loperfido (2015) and Loperfido (2017) for analysis and discussion of skewness and kurtosis matrices and their applications.

**MultiStatM** offers an alternative method, which we believe is simpler to follow: the higher-order cumulants of the same degree for some random $d$-vector $\mathbf{X}$ are collected together and kept as a vector (see Terdik (2003)). In order to do so, a particular differential operator on a multivariate function, called the T-derivative, is introduced. Although the tensor product of Euclidean vectors is not commutative, it has the advantage of permutation equivalence and allows one to obtain general formulae for cumulants and moments of any order. Methods based on a matrix approach do not provide this type of result; see also Ould-Baba et al. (2015), which goes at most up to the sixth-order moment matrices, whereas there is no such limitation in our derivations and formulae. The vectorial approach has also been advocated by Holmquist (1996) and Chacón and Duong (2015); for further discussion, one can see also Kolda and Bader (2009) and Qi (2006).

More formally, with the symbol $\otimes$ denoting the Cartesian tensor product, consider the operator $D_{\boldsymbol{\lambda}}^{\otimes}$, which we refer to as the T-derivative; see Jammalamadaka et al. (2006) for details. For any function $\phi(\boldsymbol{\lambda})$, the T-derivative is defined as

$$D_{\boldsymbol{\lambda}}^{\otimes}\phi(\boldsymbol{\lambda}) = \operatorname{vec}\left(\left(\frac{\partial\phi(\boldsymbol{\lambda})}{\partial\boldsymbol{\lambda}'}\right)'\right) = \phi(\boldsymbol{\lambda}) \otimes \frac{\partial}{\partial\boldsymbol{\lambda}}. \tag{1}$$

If $\phi$ is $k$-times differentiable, its $k$-th T-derivative is simply $D_{\boldsymbol{\lambda}}^{\otimes k}\phi(\boldsymbol{\lambda}) = D_{\boldsymbol{\lambda}}^{\otimes}\left(D_{\boldsymbol{\lambda}}^{\otimes k-1}\phi(\boldsymbol{\lambda})\right)$. When $\phi$ is the characteristic function of some random $d$-vector, $k$-times application of the T-derivative will allow to retrieve the so-called multivariate T-moments vector of order $k$, which lists all moments of the same order and has dimension $d^k$. In the same way, multivariate T-cumulants and T-Hermite polynomials can be derived. **MultiStatM** aims at exploiting the advantages of this method for multivariate analysis and dealing with the high-dimensionality implied by this approach.

We could not find any **R** (R Core Team (2023)) package using this approach although there is a partial overlapping between functions in **MultiStatM** and functions that can be found in other **R** packages; when this happens, the functions in **MultiStatM** provide useful generalizations.

The package **arrangements** (Lai (2020)) provides, among other things, all possible partitions of a

set (or an integer) in lexicographical order. In **MultiStatM** all possible partitions in terms of partition matrices together with information about types, sizes and groups of partitions are given. Furthermore **MultiStatM** provides other partition functions which produce indecomposable partitions, partitions that bring to a closed graph, partitions in only two groups.

The package **matrixcalc** (Novomestky (2021)) provides the commutation, elimination and duplication matrices for Cartesian tensor products of two vectors, which are particular cases of those provided in the package **MultiStatM**. Furthermore, in **MultiStatM** a commutator matrix for any T-product of vectors of any size is given; also, a fast algorithm to produce a symmetrizer of T-products of vectors is implemented (see Holmquist (1996)). One of the problems of commutators matrices is their high-dimensionality which can become soon quite cumbersome; on the other hand, commutator matrices are useful if one needs to work with linear combinations of matrices. To deal with the high-dimensionality issue, index versions of the commutators are also provided (more details below).

The package **calculus** (Guidotti (2022)) dealing with high dimensional numerical and symbolic calculus provides, up to a given order, the list of all symbolic entries of the multivariate Hermite polynomials with given covariance matrix used here. **MultiStatM** does not deal with single symbolic expressions rather provides full numerical evaluation of multivariate T-Hermite polynomials suitable for immediate use in further formulas and applications.

**kStatistics** (Di Nardo and Guarino (2022a)) gives string expressions for converting single entries of multivariate cumulants or moments which need to be joined and evaluated for numerical computations, for details and examples see Di Nardo and Guarino (2022b). This differs to a large degree from the formulae provided in **MultiStatM** which are always in numerical form and ready for applications; for example, conversion of multivariate moments to multivariate cumulants and viceversa, given any list of (numerical) multivariate moments up to order $r$, or multivariate cumulants up to order $r$ are provided.

In **MultiStatM**, exploiting the conversion formulae and some results on uniform spherical and folded multivariate normal distributions due to Jammalamadaka et al. (2021b) and Terdik (2021), the multivariate moments and cumulants of any order of the multivariate skew normal and its generalization (Azzalini and Dalla Valle (1996) , Arellano-Valle and Genton (2005)) are provided. The package **sn** (Azzalini (2022)) provides moments and cumulants of any order for the univariate skew-normal and the skew-t distributions; statistical methods are provided for data fitting and model diagnostics, in the univariate and the multivariate case. In **MultiStatM** random numbers generator for multivariate skew distributions are provided also.

The package **moments** (Komsta and Novomestky (2022)) deals with functions to calculate, cumulants, Pearson's kurtosis, Geary's kurtosis and skewness; tests related to them from univariate data. However the multivariate approach is not considered. The packages **mpoly** (Kahle (2013)) and **orthopolynom** (Novomestky (2022)) deal with Hermite polynomials in the univariate case.

**MultiStatM** provides some common measures of cumulant-based multivariate skewness and kurtosis (Mardia (1970), Móri et al. (1994)) which are provided in a number of packages, for example **MaxSkew** (Franceschini and Loperfido (2017a)), **MultiSkew** (Franceschini and Loperfido (2017b)) and **Kurt** (Franceschini and Loperfido (2021)). Indeed the vectorial approach provides a unifying framework for several cumulant-based skewness and kurtosis indexes, as discussed in Jammalamadaka et al. (2021b) where, beyond the above mentioned ones, indexes proposed by Malkovich and Afifi (1973), Kollo (2008), Koziol (1987), Koziol (1989) and others are discussed. When going higher than the fourth order, the vectorial approach shows it full power; see Jammalamadaka et al. (2021a) for an application in deriving the asymptotic variances of skewness and kurtosis vectors and indexes which require cumulants up to the 8-th order. Below, exploiting **MultiStatM** applications to the construction of new measures of multivariate skewness and kurtosis and derivation of Gram-Charlier approximation to densities will be presented.

As far as non R-universe is concerned, the proprietary software Mathematica (Inc.) provides partial overlapping of the multivariate methods discussed in **MultiStatM**. There, functions to compute, numerically and symbolically, multivariate moments and cumulants of common distributions and general conversion functions from moments to cumulants and viceversa are available. Most of the multivariate distributions discussed in **MultiStatM** however are not available there. Mathematica also provides several functions for univariate Hermite polynomials which however need to be adapted for the multivariate case based on the T-derivative approach.

## 2 Package features

**MultiStatM** imports **arrangements**, **MASS** (Venables and Ripley (2002)), **Matrix** (Bates et al. (2024)), **mvtnorm** (Genz and Bretz (2009)) and **stats**. We discuss here some examples on the use of the functions; considerations about timing, objects size and algorithms will be collected in Section 4.

### 2.1 Set partitions

**MultiStatM** provides several functions dealing with set partitions. Such functions provide some basic tools used to build the multivariate formulae for moments and cumulants in the following sections.

Generally a set of $N$ elements can be split into a set of disjoint subsets, i.e. it can be partitioned. The set of $N$ elements will correspond to set $1:N=\{1,2,\ldots,N\}$. If $\mathcal{K}=\{b_1,b_2,\ldots,b_r\}$ where each $b_j\subset 1:N$, then $\mathcal{K}$ is a partition provided $\cup b_j=1:N$, each $b_j$ is non-empty and $b_j\cap b_i=\emptyset$ (the empty set) is disjoint whenever $j\neq i$. The subsets $b_j$, $j=1,2,\ldots,r$ are called the blocks of $\mathcal{K}$. We will call $r$ (the number of the blocks in partition $\mathcal{K}$), the size of $\mathcal{K}$, and denote it by $|\mathcal{K}|=r$, and a partition with size $r$ will be denoted by $\mathcal{K}_{\{r\}}$. Let us denote the set of all partitions of the numbers $1:N$ by $\mathcal{P}_N$.

Consider next a partition $\mathcal{K}_{\{r\}}=\{b_1,b_2,\ldots,b_r\}\in\mathcal{P}_N$, with size $r$. Denote the cardinality $k_j$ of a block in the partition $\mathcal{K}_{\{r\}}$, i.e. $k_j=|b_j|$. The type of a partition $\mathcal{K}_{\{r\}}$ is $l=[l_1,\ldots,l_N]$, if $\mathcal{K}_{\{r\}}$ contains exactly $l_j$ blocks with cardinality $j$. The type $l$ is with length $N$ always. A partition with size $r$ and type $l$ will be denoted by $\mathcal{K}_{\{r|l\}}$. It is clear that $l_j\geq 0$, and $\sum_j jl_j=N$, and $\sum_j l_j=r$. Naturally, some $l_j$'s are zero. A block constitutes a row vector of entries 0's and 1's with length $N$. The places of 1's correspond to the elements of the block. A partition matrix collects the rows of its blocks, it is an $r\times N$ matrix with column-sums 1.

The basic function is `PartitionTypeAll` which provides complete information on the partition of a set of `N` elements, namely:

- `S_N_r`: a vector with the number of partitions of size `r=1`, `r=2`, etc. (Stirling numbers of the second kind); `S_N_r[r]` denotes the number of partition matrices of size `r`.
- `Part.class`: the list of all possible partitions given as partition matrices. This list is enumerated according to `S_N_r[r]`, $r=1,2,\ldots N$, such that the partition matrices with size `R` are listed from $\sum_{r<R}$`S_N_r`$[r]+1$ up to $\sum_{r\leq R}$`S_N_r`$[r]$. The order of the partition matrices within a fixed size is called canonical.
- `S_r_j`: a list of vectors of number of partitions with given types grouped by partitions of size `r=1`, `r=2`, etc.; an entry is the number of partitions with that type.
- `eL_r`: a list of partition types with respect to partitions of size `r=1`, `r=2`, etc. ; since a partition type is a row vector with length `N` this list includes matrices of types (row vectors), the number of rows are the length of vectors of `S_r_j` of a given size `r`.

**Example 1**. Consider the case where `N=4` and run the following

```
PTA <- PartitionTypeAll(4)
```

`S_N_r` provides the number of partitions with `r=1` to `r=4` blocks:

```
PTA$S_N_r
```

```
#> [1] 1 7 6 1
```

From the results above we see that there are: 1 partition of 1 block (`r=1`); 7 partitions of two blocks (`r=2`); 6 partitions of 3 blocks and 1 partition of 4 blocks. All the partition matrices are listed in `Part.class`, for example the first partition matrix of size 2 among 7 is

```
PTA$Part.class[[2]]
```

```
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    1    1    0
#> [2,]    0    0    0    1
```

The second partition matrix of size 3 is

```
PTA$Part.class[[10]]
```

```
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    0    1    0
#> [2,]    0    1    0    0
#> [3,]    0    0    0    1
```

etc.. If one interested in the number of partitions with different types for `r=2` then consider the list `S_r_j`, i.e.

```
PTA$S_r_j[[2]]

#> [1] 4 3
```

That is, for partitions with `r=2` blocks, there are 2 possible types with 4 and 3 partitions each. These types will show up in the list `eL_r`:

```
PTA$eL_r

#> [[1]]
#> [1] 0 0 0 1
#>
#> [[2]]
#>      [,1] [,2] [,3] [,4]
#> [1,]    1    0    1    0
#> [2,]    0    2    0    0
#>
#> [[3]]
#> [1] 2 1 0 0
#>
#> [[4]]
#> [1] 4 0 0 0
```

From `PTA$eL_r[[2]]` we see that there are two types of partition with `r=2`: the first is of type $[1,0,1,0] = (l_1 = 1, l_2 = 0, l_3 = 1, l_4 = 0)$ with 4 partitions and the second is of type $[0,2,0,0] = (l_1 = 0, l_2 = 2, l_3 = 0, l_4 = 0)$ with 3 partitions.

Another general function in this class is `Partitions` which has a `Type` argument in order to specify the type of partition to compute: `2Perm` which provides the permutation of `N` elements according to a partition matrix `L`; `Diagram` which provides the list of partition matrices indecomposable with respect to L, representing diagrams without loops; `Indecomp`, which provides the list of all indecomposable partitions with respect to a partition matrix `L`; `Pairs`, which provides the list of partitions dividing into pairs the set of `N` elements.

### 2.2 Commutators, symmetrizer and selection matrices

The `Commutators` family of functions produce commutators and selection matrices. The use of matrices allows to use linear combinations to represent problems such as the permutation of powers of T-products or the selection of certain elements in a vector. On the other hand, the size of these matrices can quickly become quite important. To deal with this issues an option for sparse matrices is always provided; also, a corresponding `Indx` group of functions which give equivalent results to the corresponding functions in the group `Matr` is provided.

Two general functions in this family are `CommutatorMatr` and `CommutatorIndx`, both with a `Type` argument selected from: `Kmn`, `Kperm`, `Mixing`, `Moment`. `Kmn` produces a commutation matrix, with usual notation $\mathbf{K}_{m\cdot n}$, of dimension $mn \times mn$ such that, given a matrix $\mathbf{A}$ $m \times n$, $\mathbf{K}_{m\cdot n} \operatorname{vec} \mathbf{A} = \operatorname{vec} \mathbf{A}'$ (see Terdik (2021), p.8) while `Kperm` produce any permutation of Kronecker products of vectors of any length.

**Example 2**. Consider the product of vectors $\mathbf{a}_1 \otimes \mathbf{a}_2 \otimes \mathbf{a}_3$ of dimensions $d_1$ to $d_3$ respectively.

`CommutatorMatr(Type="Kperm",c(3,1,2),c(d1,d2,d3))` produces $\mathbf{a}_3 \otimes \mathbf{a}_1 \otimes \mathbf{a}_2$.

```
a1 <- c(1, 2)
a2 <- c(2, 3, 4)
a3 <- c(1, 3)
x <- a1 %x% a2 %x% a3
c(CommutatorMatr(Type = "Kperm", c(3, 1, 2), c(2, 3, 2)) %*% x)

#>  [1]  2  3  4  4  6  8  6  9 12 12 18 24

a3 %x% a1 %x% a2
```

```
#>  [1]  2  3  4  4  6  8  6  9 12 12 18 24
```

The same result can be obtained by using `CommutatorIndx`

```
x[CommutatorIndx(Type = "Kperm", c(3, 1, 2), c(2, 3, 2))]
```

```
#>  [1]  2  3  4  4  6  8  6  9 12 12 18 24
```

Note that in the example above `CommutatorMatr` produces the matrix $\mathbf{K}_p$ of dimension $12 \times 12$ by which $\mathbf{y} = \mathbf{K}_p\mathbf{x}$, while `CommutatorIndx` produces the permutation $p_\mathbf{y}$ of 1:12 such that $\mathbf{y} = \mathbf{x}[p_\mathbf{y}]$. Recall that from computational point of view `CommutatorIndx` is always preferable however, if working with linear combinations of matrices, like in Example 3 and in Section 3.1 and the discussion leading to formulae (10) and (11), the use of `CommutatorMatr` is unavoidable. For this reason **MultiStatM** offers both possibilities. The idea of avoiding the use of commutator, selection (and symmetrizer) matrices can be traced back to Chacón and Duong (2015).

The Elimination and Qplication matrices- related functions respectively eliminate and restore duplicated or q-plicated elements in powers of T-products. For examples of applications of these functions we defer to Section 2.4.

The symmetrizer matrix, a $d^n \times d^n$ matrix for the symmetrization of a T-product of $n$ vectors with the same dimension $d$ which overcomes the difficulties arising from the non commutative property of the Kronecker product, and simplifies considerably the computation formulae for multivariate polynomials and their derivatives (see Holmquist (1996) for details). The symmetrizer for a T-product of $q$ vectors of dimension $d$ is defined as

$$\mathbf{S}_{d\mathbf{1}q} = \frac{1}{q!}\sum_{p\in\mathcal{P}_q}\mathbf{K}_p \tag{2}$$

where $\mathcal{P}_q$ is the set of all permutations of the numbers $1:q$ and $\mathbf{K}_p$ is the commutator matrix for the permutation $p \in \mathcal{P}_{\text{II}}$, (i.e. the `CommutatorMatr` with `Type="Kperm"` of the package). For an application of the symmetrizer via `SymMatr` with $d$-variate Hermite polynomials, see formula (4) and Example 3. We defer to Section 4 for a discussion about computational time and memory allocation of these matrices.

### 2.3 Multivariate T-Hermite polynomials

Consider a Gaussian vector $\mathbf{X}$ of dimension $d$ with $\mathrm{E}\,\mathbf{X} = 0$ and $\mathbf{\Sigma} = \mathrm{Cov}(\mathbf{X}) = \mathrm{E}\,\mathbf{X}\mathbf{X}'$ and define the generator function

$$\begin{aligned}\psi(\mathbf{X};\mathbf{a}) &= \exp\left(\mathbf{a}'\mathbf{X} - \frac{1}{2}\mathbf{a}'\mathbf{\Sigma}\mathbf{a}\right)\\ &= \exp\left(\mathbf{a}'\mathbf{X} - \frac{1}{2}\boldsymbol{\kappa}_2^{\otimes\prime}\mathbf{a}^{\otimes 2}\right)\end{aligned} \tag{3}$$

where $\mathbf{a}$ is a $d$-vector of constants and $\boldsymbol{\kappa}_2^{\otimes} = \mathrm{vec}\,\mathbf{\Sigma}$. The vector Hermite polynomials is defined via the T-derivative of the generator function, viz.

$$\mathbf{H}_n(\mathbf{X}) = D_\mathbf{a}^{\otimes n}\psi(\mathbf{X};\mathbf{a})\big|_{\mathbf{a}=0}.$$

For example one has

$$\mathbf{H}_1(\mathbf{X}) = \mathbf{X}, \quad \mathbf{H}_2(\mathbf{X}) = \mathbf{X}^{\otimes 2} - \boldsymbol{\kappa}_2^{\otimes}.$$

Note that the multivariate T-Hermite polynomial $\mathbf{H}_n(\mathbf{X})$ is a vector of dimension $d^n$ which contains the $n$-th order polynomials of the vector $\mathbf{X}^{\otimes n}$. For example the entries of $\mathbf{H}_2(\mathbf{X})$ are the second order Hermite polynomials $H_2(X_i, X_j)$, $i, j = 1, 2, \ldots d$; for $d = 2$

$$\mathbf{H}_2(\mathbf{X}) = \left((X_1^2 - \sigma_{11}), (X_1X_2 - \sigma_{12}), (X_2X_1 - \sigma_{21}), (X_2^2 - \sigma_{22})\right)'.$$

Note that $\mathbf{H}_n(\mathbf{X})$ is $n$-symmetric, i.e. $\mathbf{H}_2(\mathbf{X}) = \mathbf{S}_{d\mathbf{1}_n}\mathbf{H}_2(\mathbf{X})$ where $\mathbf{S}_{d\mathbf{1}_n}$ is the symmetrizer defined in (2). From this one can get useful recursion formulae

$$\mathbf{H}_n(\mathbf{X}) = \mathbf{S}_{d\mathbf{1}_n}\left(\mathbf{H}_{n-1}(\mathbf{X})\otimes\mathbf{X} - (n-1)\mathbf{H}_{n-2}(\mathbf{X})\otimes\boldsymbol{\kappa}_2^{\otimes}\right). \tag{4}$$

For further details, consult Terdik (2021), Section 4.5.

The definition of the $d$-variate Hermite polynomial requires the variance-covariance matrix $\mathbf{\Sigma}$ of the vector $\mathbf{X}$. `HermiteN` and `HermiteN2X` with `Type=c("Univariate","Multivariate")` compute respectively the univariate and $d$-variate Hermite polynomials and their inverses up to a given order `N` evaluated at `x` for a given covariance matrix `Sig2`. By default `Sig2=`$I_{\mathbf{d}}$.

**Example 3**. Consider a vector $x$ of dimension $d = 2$ and compute the Hermite polynomials up to order `N`=3 with identity variance matrix $\mathbf{I}_2$; `HermiteN(x,3,Type="Multivariate")` will contain the list of $d$-variate Hermite polynomials up to order 3 at value $x$.

```
x <- c(1, 2)
HN <- HermiteN(x, 3, Type = "Multivariate")
```

Retrieve the third Hermite polynomial

```
HN[[3]]

#> [1] -2  0  0  3  0  3  3  2
```

Use now the symmetrizer in matrix form to verify formula (4) with $n = 2$. Note that in this case $\boldsymbol{\kappa}_2^{\otimes} = \operatorname{vec} \mathbf{I}_2$.

```
k2 <- c(1, 0, 0, 1)
c(SymMatr(2, 3) %*% (kronecker(HN[[2]], x) - (3 - 1) * kronecker(HN[[1]], k2)))

#> [1] -2  0  0  3  0  3  3  2
```

### 2.4 Multivariate T-moments and T-cumulants

Multivariate moments and cumulants of all orders of a random vector $\mathbf{X}$ in $d$-dimensions, with mean vector $\boldsymbol{\mu}$ and covariance matrix $\mathbf{\Sigma}$ can be obtained by applying the T-derivative respectively to the characteristic function and the log of the CF. More formally, let $\boldsymbol{\lambda}$ a $d$-vector of real constants; $\phi_{\mathbf{X}}(\boldsymbol{\lambda})$ and $\psi_{\mathbf{X}}(\boldsymbol{\lambda}) = \log \phi_{\mathbf{X}}(\boldsymbol{\lambda})$ denote, respectively, the characteristic function and the cumulant function of $\mathbf{X}$. Then the $k$-th order moments and cumulants of the vector $\mathbf{X}$ are obtained as

$$\boldsymbol{\mu}_{\mathbf{X},k}^{\otimes} = (-\mathbf{i})^k D_{\boldsymbol{\lambda}}^{\otimes k} \phi_{\mathbf{X}}(\boldsymbol{\lambda})\big|_{\boldsymbol{\lambda}=0}.$$

$$\boldsymbol{\kappa}_{\mathbf{X},k}^{\otimes} = \underline{\mathrm{Cum}}_k(\mathbf{X}) = (-\mathbf{i})^k D_{\boldsymbol{\lambda}}^{\otimes k} \psi_{\mathbf{X}}(\boldsymbol{\lambda})\big|_{\boldsymbol{\lambda}=0}.$$

Note that $\boldsymbol{\mu}_{\mathbf{X},k} = \mathrm{E}\, \mathbf{X}^{\otimes k}$ is a vector of dimension $d^k$ that contains all possible moments of order $k$ formed by $X_1, \ldots, X_d$. This approach has the advantage of being straightforwardly extendable to any $k$-th order moment. An analogous discussion can be done for cumulants. Note that one has $\boldsymbol{\kappa}_{\mathbf{X},2} = \operatorname{vec} \mathbf{\Sigma}$. **MultiStatM** contains functions which obtains moments from cumulants and vice-versa as well as functions which provide theoretical moments and cumulants for some important multivariate distributions.

The `Cum2Mom` and `Mom2Cum` either for the univariate or multivariate cases provide conversion formulae for cumulants from moments and viceversa given any list of moments (or cumulants).

**Example 4**. Consider the case of the 2-variate standard normal distribution with unit mean vector and covariance matrix with unit elements on the main diagonal and off-diagonal elements equal to 0.5; in this case the first four moments are given in the vector `mu` below

```
mu <- list(
  c(1, 1), c(2, 1.5, 1.5, 2), c(4, 3, 3, 3, 3, 3, 3, 4),
  +c(10, 7, 7, 6.5, 7, 6.5, 6.5, 7, 7, 6.5, 6.5, 7, 6.5, 7, 7, 10)
)
cum <- Mom2Cum(mu, Type = "Multivariate")
cum

#> [[1]]
#> [1] 1 1
#>
#> [[2]]
#> [1] 1.0 0.5 0.5 1.0
#>
```

```
#> [[3]]
#> [1] 0 0 0 0 0 0 0 0
#>
#> [[4]]
#>  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Getting back to moments

```
Cum2Mom(cum, Type = "Multivariate")
```

```
#> [[1]]
#> [1] 1 1
#>
#> [[2]]
#> [1] 2.0 1.5 1.5 2.0
#>
#> [[3]]
#> [1] 4 3 3 3 3 3 3 4
#>
#> [[4]]
#>  [1] 10.0  7.0  7.0  6.5  7.0  6.5  6.5  7.0  7.0  6.5  6.5  7.0  6.5  7.0  7.0
#> [16] 10.0
```

If one wishes to select only the distinct moments from the vector of third moments, then

```
r.mu <- mu[[3]][EliminIndx(2, 3)]
r.mu
```

```
#> [1] 4 3 3 4
```

Note that `EliminIndx` (and `EliminMatr`) does not correspond to the function `unique`, rather it individuates the distinct elements from the symmetry of the Kronecker product. Here the values 4 and 3 could appear to be duplicates however in general they could be different. This allows to recover the whole vector when needed by using `QplicIndx` (or `QplicMatr`)

```
mu3 <- r.mu[QplicIndx(2, 3)]
mu3
```

```
#> [1] 4 3 3 3 3 3 3 4
```

The `MomCum` functions provide theoretical moments and cumulants for some common multivariate distributions: Skew-normal, Canonical Fundamental Skew-normal (CFUSN), Uniform distribution on the sphere, central folded Normal distribution (univariate and multivariate); for detail on the multivariate formulae used see Jammalamadaka et al. (2021b). Some more details on the multivariate distributions considered are reported in the list below.

- A $d$-vector $\mathbf{U}$ having uniform distribution on the sphere $\mathbb{S}_{d-1}$. Moments and cumulants of all orders are provided for $\mathbf{U}$ by the function `MomCumUniS`; the function `EVSKUniS` can compute moments and cumulants (up to the 4th order), skewness, and kurtosis of $\mathbf{U}$ (`Type="Standard"`) and its modulus (`Type="Modulus"`). Recall that any $d$-vector, say $\mathbf{W}$, has a spherically symmetric distribution if that distribution is invariant under the group of rotations in $\mathbb{R}^d$. This is equivalent to saying that $\mathbf{W}$ has the stochastic representation $\mathbf{W} = R\mathbf{U}$ where $R$ is a non negative random variable. Moments and cumulants of $\mathbf{W}$ can be obtained by its stochastic representation as discussed in Jammalamadaka et al. (2021b), Theorem 1 and Lemma 1. Furthermore a $d$-vector $\mathbf{X}$ has an elliptically symmetric distribution if it has the representation

  $$\mathbf{X} = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\mathbf{W},$$

  where $\boldsymbol{\mu} \in \mathbb{R}^d$, $\boldsymbol{\Sigma}$ is a variance-covariance matrix and $\mathbf{W}$ is spherically distributed. Hence the cumulants of $\mathbf{X}$ are just constant times the cumulants of $\mathbf{W}$ except for the mean i.e.

  $$\underline{\mathrm{Cum}}_m(\mathbf{X}) = \left(\boldsymbol{\Sigma}^{1/2}\right)^{\otimes m} \underline{\mathrm{Cum}}_m(\mathbf{W}).$$

- If $\mathbf{Z}$ denotes a $d$-vector with $d$-variate standard normal distribution, the function `MomCumZabs` provides the moments and cumulants of $|\mathbf{Z}|$ respectively in the univariate (`Type="univariate"`) and multivariate case (`Type="Multivariate"`).
- The multivariate skew-normal distribution introduced by Azzalini and Dalla Valle (1996), whose marginal densities are scalar skew-normals. A $d$-dimensional random vector $\mathbf{X}$ is said to have a multivariate skew-normal distribution, $\mathrm{SN}_d(\boldsymbol{\mu}, \boldsymbol{\Omega}, \boldsymbol{\alpha})$ with shape parameter $\boldsymbol{\alpha}$ if it has the density function
$$2\varphi(\mathbf{X}; \boldsymbol{\mu}, \boldsymbol{\Omega})\, \Phi\left(\boldsymbol{\alpha}'(\mathbf{X} - \boldsymbol{\mu})\right), \quad \mathbf{X} \in \mathbb{R}^d,$$
where $\varphi(\mathbf{X}; \boldsymbol{\mu}, \boldsymbol{\Omega})$ is the $d$-dimensional normal density with mean $\boldsymbol{\mu}$ and correlation matrix $\boldsymbol{\Omega}$; here $\varphi$ and $\Phi$ denote the univariate standard normal density and the cdf. For a general formula for cumulants, see Jammalamadaka et al. (2021b), Lemma 4. For this distribution are available the functions `MomCumSkewNorm`, which computes the theoretical values of moments and cumulants up to the r-th order and `EVSKSkewNorm` which gives mean vector, covariance, skewness and kurtosis vectors and other features.
- Arellano-Valle and Genton (2005) introduced the CFUSN distribution (cf. their Proposition 2.3), to include all existing definitions of skew-normal distributions. The marginal stochastic representation of $\mathbf{X}$ with distribution $\mathrm{CFUSN}_{d,m}(\boldsymbol{\Delta})$ is given by
$$\mathbf{X} = \boldsymbol{\Delta}\,|\mathbf{Z}_1| + \left(\mathbf{I}_d - \boldsymbol{\Delta}\boldsymbol{\Delta}'\right)^{1/2} \mathbf{Z}_2,$$
where $\boldsymbol{\Delta}$, is the $d \times m$ skewness matrix such that $\|\boldsymbol{\Delta}\mathbf{a}\| < 1$, for all $\|\mathbf{a}\| = 1$, and $\mathbf{Z}_1 \in \mathcal{N}(0, \mathbf{I}_m)$ and $\mathbf{Z}_2 \in \mathcal{N}(0, \mathbf{I}_d)$ are independent (Proposition 2.2. Arellano-Valle and Genton (2005)). A simple construction of $\boldsymbol{\Delta}$ is $\boldsymbol{\Delta} = \boldsymbol{\Lambda}\left(\mathbf{I}_m + \boldsymbol{\Lambda}'\boldsymbol{\Lambda}\right)^{-1/2}$ with some real matrix $\boldsymbol{\Lambda}$ with dimensions $d \times m$. The $\mathrm{CFUSN}_{d,m}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Delta})$ can be defined via the linear transformation $\boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\mathbf{X}$. For a general formula for cumulants, see Jammalamadaka et al. (2021b), Lemma 5. For this distributions `MomCumCFUSN` provides moments and cumulants up to the $r$-th order.

**Example 5**. For a skew-normal distribution with $\alpha = (10, 5, 0)$ and correlation function $\Omega = \mathrm{diag}(1, 1, 1)$ the third moments and cumulants are

```
alpha <- c(10, 5, 0)
omega <- diag(3)
MSN <- MomCumSkewNorm(r = 3, omega, alpha, nMu = TRUE)
round(MSN$Mu[[3]], 3)

#>  [1] 1.568 0.073 0.000 0.073 0.570 0.000 0.000 0.000 0.711 0.073 0.570 0.000
#> [13] 0.570 0.996 0.000 0.000 0.000 0.355 0.000 0.000 0.711 0.000 0.000 0.355
#> [25] 0.711 0.355 0.000

round(MSN$CumX[[3]], 3)

#>  [1] 0.154 0.077 0.000 0.077 0.039 0.000 0.000 0.000 0.000 0.077 0.039 0.000
#> [13] 0.039 0.019 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
#> [25] 0.000 0.000 0.000
```

As another example, for the uniform distribution on the sphere, the kurtosis vector is:

```
EVSKUniS(3, Type = "Standard")$Kurt.U

#>  [1] -1.2  0.0  0.0  0.0 -0.4  0.0  0.0  0.0 -0.4  0.0 -0.4  0.0 -0.4  0.0  0.0
#> [16]  0.0  0.0  0.0  0.0  0.0 -0.4  0.0  0.0  0.0 -0.4  0.0  0.0  0.0 -0.4  0.0
#> [31] -0.4  0.0  0.0  0.0  0.0  0.0 -0.4  0.0  0.0  0.0 -1.2  0.0  0.0  0.0 -0.4
#> [46]  0.0  0.0  0.0  0.0  0.0 -0.4  0.0 -0.4  0.0  0.0  0.0 -0.4  0.0  0.0  0.0
#> [61] -0.4  0.0  0.0  0.0  0.0  0.0  0.0  0.0 -0.4  0.0 -0.4  0.0 -0.4  0.0  0.0
#> [76]  0.0 -0.4  0.0  0.0  0.0 -1.2
```

### 2.5 Analyzing multivariate skewness and kurtosis

A complete picture of multivariate skewness is provided by the third-order T-cumulant (skewness vector) of a standardized $\mathbf{X}$; set $\mathbf{Y} = \boldsymbol{\Sigma}^{-1/2}(\mathbf{X} - \boldsymbol{\mu})$, then the skewness vector is

$$\boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},3} = \underline{\mathrm{Cum}}_3(\mathbf{Y}) = \left(\boldsymbol{\Sigma}^{-1/2}\right)^{\otimes 3} \boldsymbol{\kappa}^{\otimes}_{\mathbf{X},3}. \tag{5}$$

A definition of skewness which is invariant under shifting and orthogonal transformations or, in other words affine invariant, is provided by the total skewness of $\mathbf{X}$, i.e. the square norm of the skewness vector:

$$\gamma_{1,d} = \|\boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},3}\|^2 = \beta_{1,d} \tag{6}$$

where $\beta_{1,d}$ is Mardia's multivariate skewness index (Mardia, 1970) which coincides with the total skewness $\gamma_{1,d}$ since the third-order central moments and third-order cumulants are equal. Móri et al. (1994) (MRSz's) skewness vector can also be recovered from the skewness vector as

$$\mathbf{b}(\mathbf{Y}) = \left(\mathrm{vec}'\,\mathbf{I}_d \otimes \mathbf{I}_d\right) \boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},3}. \tag{7}$$

Note that $\mathrm{vec}'\,\mathbf{I}_d \otimes \mathbf{I}_d$ is a matrix of dimensions $d \times d^3$, which contains $d$ unit values per-row, whereas all the others are 0; as a consequence, $\mathbf{b}(\mathbf{Y})$ does not take into account the contribution of cumulants of the type $\mathrm{Cum}_3(X_j, X_k, X_l)$, where all the three indices $j$, $k$, $l$ are different from each other. The corresponding scalar measure of multivariate skewness is $b(\mathbf{Y}) = \|\mathbf{b}(\mathbf{Y})\|^2$.

The fourth-order T-cumulant of the standardized $\mathbf{X}$, i.e. $\boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},4} = \mathbf{H}_4(\mathbf{Y})$, will be called kurtosis vector of $\mathbf{X}$; its square norm will be called the total kurtosis of $\mathbf{X}$,

$$\gamma_{2,d} = \|\boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},4}\|^2. \tag{8}$$

Mardia's kurtosis index $\beta_{2,d} = \mathrm{E}\left(\mathbf{Y}'\mathbf{Y}\right)^2$ ( Mardia (1970)) is related to the kurtosis vector by the formula

$$\beta_{2,d} = \left(\mathrm{vec}'\,\mathbf{I}_{d^2}\right) \boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},4} + d(d+2).$$

A consequence of this is that Mardia's measure does not depend on all the entries of $\boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},4}$ which has $d(d+1)(d+2)(d+3)/24$ distinct elements, while $\beta_{2,d}$ includes only $d^2$ elements among them. We note that if $\mathbf{X}$ is Gaussian, then $\boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},4} = \mathbf{0}$. Note that Koziol (1989), for $\mathbf{Y}$ and an independent copy $\tilde{\mathbf{Y}}$, considered $\mathrm{E}(\mathbf{Y}'\tilde{\mathbf{Y}})^4$ as an index of kurtosis which is the next higher degree analogue of Mardia's skewness index $\beta_{1,d}$. Note that (see Jammalamadaka et al. (2021b))

$$\mathrm{E}(\mathbf{Y}'\tilde{\mathbf{Y}})^4 = \gamma_{2,d} + 6\beta_{2,d} - d^2.$$

Móri et al. (1994) define what we will call the MRS kurtosis matrix

$$\mathbf{B}(Y) = \mathrm{E}\left(\mathbf{Y}\mathbf{Y}'\mathbf{Y}\mathbf{Y}'\right) - (d+2)\mathbf{I}_d,$$

which can be expressed in terms of the kurtosis vector as (note also that $\mathrm{tr}\,\mathbf{B}(Y) = \beta_{2,d}$),

$$\mathrm{vec}\,\mathbf{B}(Y) = \left(\mathbf{I}_{d^2} \otimes \mathrm{vec}'\,\mathbf{I}_d\right) \boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},4}.$$

$\mathbf{B}(Y)$ got a prominent role in multivariate statistics due to invariant coordinate selection (Tyler et al. (2009)), model-based clustering (Peña et al. (2010)), and multivariate outlier detection (Archimbaud et al. (2018)). For further connections between $\boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},3}$, $\boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},4}$ and other measures of skewness and kurtosis, see Terdik (2021), Section 6 and Jammalamadaka et al. (2021b). For Malkovich and Afifi (1973), where directional measures of multivariate skewness and kurtosis measures are proposed, one can refer to **MaxSkew**, **Kurt** and Loperfido (2015) for discussions and applications; see also Arevalillo and Navarro (2020) and Arevalillo and Navarro (2021).

For examples on computing skewness and kurtosis measures with **MultiStatM** and relate tests see the following sections.

### 2.6 Estimation

Estimating functions starting from a vector of multivariate data are available: multivariate Hermite polynomials, moments, cumulants, skewness and kurtosis vectors, Mardia's skewness and kurtosis indexes, MRSz's skewness vector and kurtosis matrices. The `Estimation` family of functions include `SampleEVSK`, `SampleGC`, `SampleHermiteN`, `SampleKurt`, `SampleSkew`, `SampleMomCum`, see the examples in this and the next section for their application.

The basic estimation method relies on sample averages: given a random sample of $d$-vectors $\mathbf{X}_1, \ldots, \mathbf{X}_n$, for any function $g$ let

$$\widehat{g(\mathbf{X})} = \frac{1}{n}\sum_{i=1}^{n} g(\mathbf{X}_i). \tag{9}$$

To estimate moments of any order $k = 1, \ldots, r$ the function `SampleMomCum` sets $g(\mathbf{X}_i) = \mathbf{X}_i^{\otimes k}$ in (9), while cumulants are obtained from moments via the function `Mom2Cum`. Options for centering, i.e. $(\mathbf{X_i} - \bar{\mathbf{X}})$

or scaling (default), i.e. $\mathbf{S}^{-1/2}(\mathbf{X_i} - \bar{\mathbf{X}})$, where $\bar{\mathbf{X}}$ and $\mathbf{S}$ are the usual sample estimates of mean and variance, are provided. For any sample $\mathbf{X}_1, \ldots, \mathbf{X}_n$ the mean zero and identity covariance matrix data $\mathbf{Z}_i = \mathbf{S}^{-1/2}(\mathbf{X_i} - \bar{\mathbf{X}})$ can always be obtained using `MultiStatM::MVStandardize`.

$d$-variate Hermite polynomials of any order $k = 1, \ldots, N$ are evaluated at standardized data points setting $g(\mathbf{X}_i) = \mathbf{H}_k(\mathbf{X}_i)$ in (9). The estimators in terms of standard Hermite polynomials proved to be useful for finding the asymptotic distribution of skewness and kurtosis estimates, see Jammalamadaka et al. (2021a). For estimation of skewness and kurtosis vectors, see Terdik (2021) (6.20), p. 331 and (6.28), p. 340. Formulae for evaluating the variance of estimators are also available. For example `SampleVarianceSkewKurt` provides estimates of the covariance matrix of the data-estimated skewness and kurtosis vectors; see (Terdik (2021), (6.13) and (6.22)).

**MultiStatM** does not make use of multivariate k-statistics in estimation. For these one can use **kStatistics**; for tutorials see Di Nardo and Guarino (2022b) and Smith (2020).

**Example 6**. Consider a multivariate data vector of dimension $d = 3$ and $n = 1000$ from the multivariate skew-normal distribution of Example 5. The estimated first four cumulants are listed in the object `EsMSN` obtained by `SampleEVSK`; the corresponding theoretical values are in the object `ThMSN` obtained by `MomCumSkewNorm`.

```
data <- rSkewNorm(1000, omega, alpha)
EsMSN <- SampleEVSK(data)
ThMSN <- EVSKSkewNorm(omega, alpha)
```

Compare the distinct elements of the estimated skewness vector and the theoretical ones using `EliminIndx`.

```
EsMSN$estSkew[EliminIndx(3, 3)]

#>  [1]  0.61047183  0.29410211 -0.03229834  0.20806541  0.01564957  0.01190094
#>  [7]  0.17297521  0.05908067  0.03803957 -0.09638842

ThMSN$SkewX[EliminIndx(3, 3)]

#>  [1] 0.68927167 0.34463583 0.00000000 0.17231792 0.00000000 0.00000000
#>  [7] 0.08615896 0.00000000 0.00000000 0.00000000
```

If one wishes to recover the estimated univariate skewness and kurtosis of the univariate components $X_1$, $X_2$ and $X_3$ of $X$, then, using `UnivMomCum`,

```
EsMSN$estSkew[UnivMomCum(3, 3)]

#> [1]  0.61047183  0.17297521 -0.09638842

EsMSN$estKurt[UnivMomCum(3, 4)]

#> [1]  0.4664381734  0.1013773614 -0.0005109212
```

An estimate of Mardia's skewness index is provided together with the p-value under the null hypothesis of normality. The theoretical value of Mardia's skewness can be recovered from the element `SkewX.tot` in the object `ThMSN`.

```
SampleSkew(data, Type = "Mardia")

#> $Mardia.Skewness
#> [1] 0.8210853
#>
#> $p.value
#> [1] 1.863394e-24

ThMSN$SkewX.tot

#> [1] 0.9279208
```

The MRSz skewness vector and index are provided together with the p-value for the skewness index under the null hypothesis of normality. The theoretical value, for the distribution at hand, can be computed using formula (7).

```
SampleSkew(data, Type = "MRSz")

#> $MRSz.Skewness.Vector
#> [1]  0.8304382  0.5051169 -0.0696061
#>
#> $MRSz.Skewness.Index
#> [1] 0.9496157
#>
#> $p.value
#> [1] 1.881808e-20

c(t(c(diag(3)) %x% diag(3)) %*% ThMSN$SkewX)

#> [1] 0.8615896 0.4307948 0.0000000
```

# 3 Applications

## 3.1 Weighted measures of multivariate skewness and kurtosis

The skewness and kurtosis vectors, respectively $\mathbf{H}_q(\mathbf{Y})$, $q = 3, 4$, contain $d^q$ elements, which are not all distinct. Just as the covariance matrix of a $d$-dimensional vector contains only $d(d+1)/2$ distinct elements, a simple computation shows that $\mathbf{H}_q(\underline{Y})$ contains $\binom{d+q-1}{q}$ distinct elements. To avoid duplicated information, it makes sense to work only with these *distinct* elements of the cumulant vector. The selection of distinct elements can be accomplished via linear transformations and through the so-called elimination matrix which we denote here as $\mathbf{E}^+_{d,q}$. The linear transformation

$$\mathbf{H}_{q,D}(\mathbf{Y}) = \mathbf{E}^+_{d,q}\mathbf{H}_q(\mathbf{Y}) \tag{10}$$

contains only the distinct values of $\mathbf{H}_q(\mathbf{Y})$ and has covariance matrix

$$\mathbf{C}_{\mathbf{H}_{q,D}} = \mathbf{E}^+_{d,q}\mathbf{C}_{\mathbf{H}_q}\mathbf{E}^{+\top}_{d,q}. \tag{11}$$

Given the availability of explicit formulae for the covariance matrices, it is natural to consider the weighted skewness and kurtosis measures based on $\boldsymbol{\kappa}_{\mathbf{Y}^\otimes,q,D} = \mathbf{E}^+_{d,q}\boldsymbol{\kappa}_{\mathbf{Y}^\otimes,q}$, $q = 3, 4$ the third and fourth order cumulant vectors with distinct elements, respectively; denote these as $\beta_{1,T,d}$ (skewness) and $\beta_{2,T,d}$ (kurtosis); then

$$\beta_{q-2,T,d} = \left\| \mathbf{C}^{-1/2}_{\mathbf{H}_{q,D}} \boldsymbol{\kappa}_{\mathbf{Y}^\otimes,q,D} \right\|^2, \quad q = 3, 4. \tag{12}$$

The corresponding sample versions, $b_{q-2,T,d}$, $q = 3, 4$, are simply obtained by replacing the cumulant vectors and the covariance with their estimated versions. See Jammalamadaka et al. (2021a) for detailed results about their asymptotic distributions which, under mild conditions are non-central $\chi^2$ distributions with $\binom{d+q-1}{q}$ degrees of freedom and non-centrality parameter $\beta_{q-2,T,d}$, $q = 3, 4$.

Since the distribution of $b_{1,T,d}$ with estimated covariance matrix is asymptotically a $\chi^2$ with $\binom{d+2}{3}$ degrees of freedom either in the Gaussian or in the more general case of elliptically symmetric multivariate distributions, a consistent test for elliptical symmetry can be based on $b_{1,T,d}$ with the estimated covariance $\mathbf{C}_{\mathbf{H}_{3,D}}$. Under the alternative hypothesis of asymmetry, the asymptotic distribution will involve a non-centrality parameter $\beta_{1,T,d}$.

**Example 7**. Suppose we have 3-variate data from a normal distribution with unknown mean and covariance matrix. We want to estimate $b_{1,T,d}$ from formula (12). Since we are considering the skewness vector $\mathbf{H}_3(\mathbf{Y})$, without loss of generality we can consider a null mean vector and $\boldsymbol{\Sigma} = \mathbf{I}_d$. Hence, in order to compute the covariance matrix of $\mathbf{H}_3(\mathbf{Y})$ we use `HermiteCov12`:

```
d <- 3
Cov <- diag(d)
CovH3 <- HermiteCov12(Cov, 3)
```

From formula (11) one can use `EliminMatr` to compute the covariance (and its inverse) of the distinct vectors elements of $\mathbf{Y}$ as

```
ME <- EliminMatr(3, 3)
CovH3D <- ME %*% CovH3 %*% t(ME)
InvCovH3D <- solve(CovH3D)
```

Given a sample of data, compute $b_{1,T,3}$. First estimate the skewness vector,

```
x <- MASS::mvrnorm(150, rep(0, d), diag(d))
EH3 <- SampleHermiteN(x, 3)
```

As a second step obtain the skewness vector with only distinct elements

```
EH3D <- EH3[EliminIndx(d, 3)]
```

In the third step we estimate $b_{1,T,3}$

```
b1T <- EH3D %*% InvCovH3D %*% EH3D
b1T

#>             [,1]
#> [1,] 0.06601879
```

If we compare $b_{1,T,3}$ with Mardia's skewness index $b_{1,3}$,

```
b1 <- SampleSkew(x, Type = "Mardia")$Mardia.Skewness
b1

#> [1] 0.3961127
```

We note that $b_{1,3} = 6 \cdot b_{1,T,3}$. This fact was already noted and formalized in Jammalamadaka et al. (2021a), i.e. Mardia's index can also be seen to be equivalent, up to a constant, to the weighted index $b_{1,T,d}$ for the Gaussian case.

One can also build a weighted indexes for more general cases, e.g. for an elliptically symmetric distribution, by computing the covariance matrices for $\mathbf{H}_3(\mathbf{Y})$ and $\mathbf{H}_4(\mathbf{Y})$ using the results of Theorem 1 in Jammalamadaka et al. (2021a) and the functions provided in **MultiStatM**.

### 3.2 A multivariate Gram-Charlier density

The Gram-Charlier expansion expresses the density function of a random variable in terms of its cumulants (see e.g. Brenn and Anfinsen (2017) and Section 4.7 in Terdik (2021)).

Given a $d$-vector from some distribution with density $f_{\mathbf{X}}(\mathbf{x})$ having mean $\mathbf{0}$ and covariance matrix $\mathbf{\Sigma}$, under appropriate moment conditions, the density can be expanded into a Gram-Charlier series as follows

$$f_{\mathbf{X}}(\mathbf{x}) = \left(1 + \sum_{k=3}^{\infty} \frac{1}{k!} \left(\mathbf{B}_k\left(0, 0, \boldsymbol{\kappa}_{\mathbf{Y},3}, \ldots \boldsymbol{\kappa}_{\mathbf{Y},k}\right)\right)^{\top} \mathbf{H}_k\left(\mathbf{\Sigma}^{-1/2}(\mathbf{x} - \boldsymbol{\mu})\right)\right) \varphi(\mathbf{x}; \mathbf{0}, \mathbf{\Sigma}) \tag{13}$$

with $\varphi$ denoting the $d$-variate standard normal density function and $\mathbf{B}_k$ indicating the (complete) T-Bell polynomial of $k$-th order (see, e.g. Terdik (2021), Section 1.4.2 and Section 4.7). The polynomials $\mathbf{B}_3$, $\mathbf{B}_4$, and $\mathbf{B}_5$ have the following simple forms

$$\begin{aligned}
\mathbf{B}_3\left(0, 0, \boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},3}\right) &= \boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},3}, \\
\mathbf{B}_4\left(0, 0, \boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},3}, \boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},4}\right) &= \boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},4}, \\
\mathbf{B}_5\left(0, 0, \boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},3}, \boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},4}, \boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},5}\right) &= \boldsymbol{\kappa}^{\otimes}_{\mathbf{Y},5}.
\end{aligned}$$

Using the above results and considering only the first few order terms in the expansion (13), one can provide an approximation to $f_{\mathbf{X}}(\mathbf{x})$:

$$f_{\mathbf{X}}(\mathbf{x}) = \left(1 + \sum_{k=3}^{4} \frac{1}{k!} \boldsymbol{\kappa}^{\top}_{\mathbf{Y},k} \mathbf{H}_k\left(\mathbf{\Sigma}^{-1/2}(\mathbf{x} - \boldsymbol{\mu})\right)\right) \varphi(\mathbf{x}, \mathbf{0}, \mathbf{\Sigma}) + \mathcal{O} \tag{14}$$

where $\mathcal{O}$ includes the remaining terms. Starting from $d$-variate data, a Gram-Charlier approximation to an unkonw multivariate density can be constructed using the multivariate function for cumulants and Hermite polynomials provided in **MultiStatM**. In the package, the function `SampleGC` is available in order to estimate the truncated Gram-Charlier density for $k$ from 3 to 8. For higher order generalized multivariate Bell polynomials see **kstatistics**. `SampleGC` can estimate directly the needed cumulants from the data or use a given list of cumulants to evaluate the Gram-Charlier density at the given data. If large data is used it will be more efficient to estimate the needed cumulants using `SampleMomCum` and then evaluate the density at the points needed for analysis; note that in (14) $\mathbf{\Sigma}$ refers to $\mathbf{X}$ while the higher order cumulants are computed for the standardized data $\mathbf{Y}$. These aspects can be handled efficiently with `SampleMomCum` which provides options for centering and scaling of the data.

**Example 8**. For $n = 500$ randomly generated data from a bivariate skew-normal distribution with $\alpha = (4, 12)$ and correlation function $\Omega = \text{diag}(1, 1)$, the Gram-Charlier approximation (for $k = 4$ and $k = 6$) is evaluated: a) using the cumulants estimated from data; b) using the true cumulants of the given skew-normal. For this distribution the mean can be computed as $\sqrt{2/\pi}\,\Omega\alpha\,(1 + \alpha'\Omega\alpha)^{-1/2}$.

```
alpha <- c(4, 12)
omega <- diag(2)
set.seed(1234)
X <- rSkewNorm(500, omega, alpha)
X <- X - sqrt(2 / pi) * c(omega %*% alpha) / c(sqrt(1 + alpha %*% omega %*% alpha))
```

For $k$ all the needed cumulants (in proper form as argument for `SampleGC`) can be efficiently estimated using (e.g. $k = 4$)

```
EC <- SampleMomCum(X, r = 4, centering = TRUE, scaling = FALSE)
```

The theoretical cumulants (up to any order) can be obtained using `MomCumSkewNorm`.

```
TC <- MomCumSkewNorm(r = 4, omega, alpha, nMu = FALSE)
```

Compare directly the theoretical and data-estimated skewness and kurtosis vectors (use only the distinct values) in the code below

```
EC$estCum.r[[3]][EliminIndx(2, 3)]

#> [1]  0.215266065 -0.007213623  0.032072834  0.207076236

TC$CumX[[3]][EliminIndx(2, 3)]

#> [1] 0.006830064 0.020490192 0.061470576 0.184411729

EC$estCum.r[[4]][EliminIndx(2, 4)]

#> [1] -0.028502396  0.047460791  0.010992685  0.001601845  0.010988540

TC$CumX[[4]][EliminIndx(2, 4)]

#> [1] 0.001133494 0.003400482 0.010201445 0.030604334 0.091813003
```

In order to produce the plots in Figure 1, the Gram-Charlier approximation with $k = 4$ (and $k = 6$) are evaluated using the estimated cumulants and the true ones; for example:

```
fy4 <- SampleGC(X, cum = EC$estCum.r)
```

Figure 1 shows the contours of true skew-normal density and the Gram-Charlier densities based on the estimated and true cumulants for the cases $k = 4$ and $k = 6$. The approximation, as expected, appears to improve when true cumulants are used and $k$ is larger.

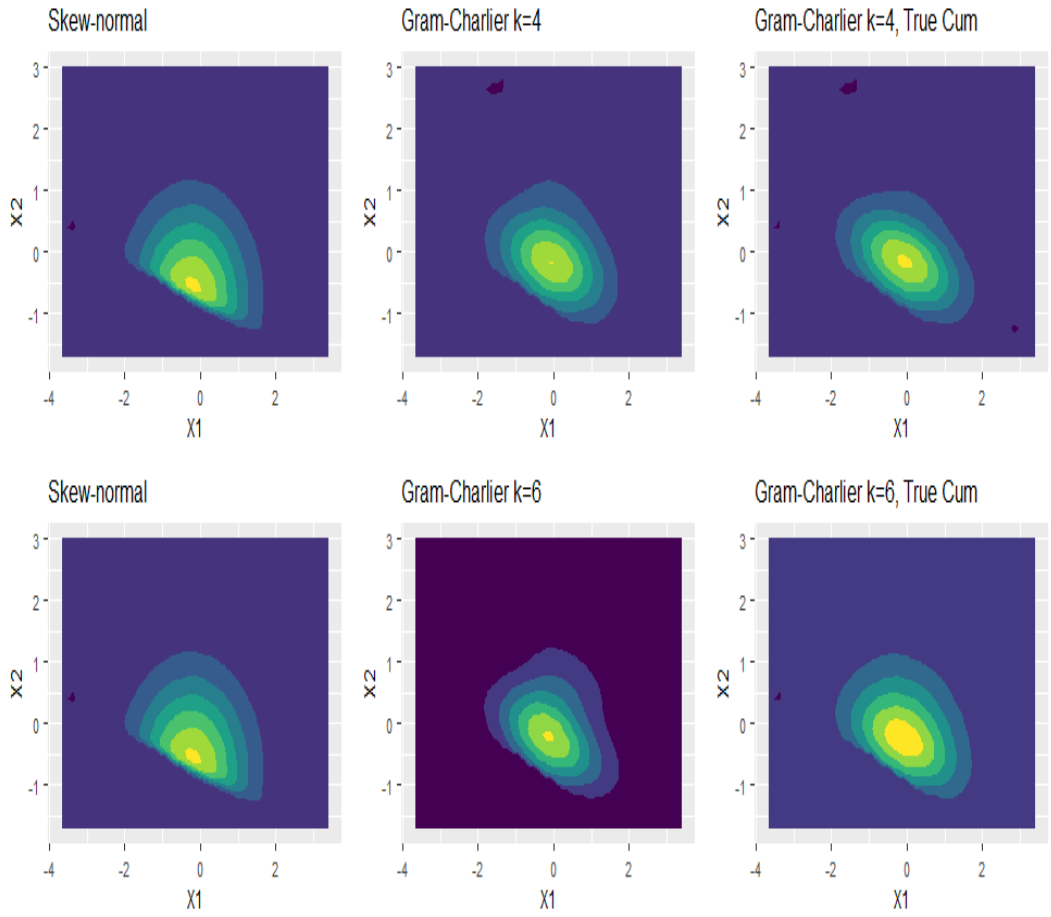



**Figure 1:** Contour plots of the skew-normal true density (left), the Gram-Charlier approximation with estimated cumulants (center) and the Gram-Charlier approximation with true cumulants (right). Case with $k = 4$ (first row) and $k = 6$ (second row).

## 4 Computational details

One of the basic functions of **MultiStatM** is `PartitionTypeAll` (Section 2.1) whose algorithm to generate all partitions of the set $1:N$ is based on the recursive generation of inclusive and exclusive partition matrices as described in Terdik (2021), section 1.4. For example, the number of partitions of a given type `S_r_j` and `eL_r` - see Section 2.1 produced by `PartitionTypeAll` are exploited in conversion formulae from moments to cumulants and viceversa. Recall that the conversion formula from moments to cumulants (Terdik (2021), Section 3.4)) is given by

$$\begin{aligned}\boldsymbol{\mu}_n^{\otimes} &= \sum_{\mathcal{K}\in\mathcal{P}_n} \mathbf{K}_{p(\mathcal{K})}^{-1} \prod_{b_j\in\mathcal{K}}^{\otimes} \boldsymbol{\kappa}_{|b_j|}^{\otimes} \\ &= \mathbf{S}_{d\mathbf{1}_n}\left(\sum_{r=1}^{n}\sum_{\sum l_j=r,\sum jl_j=n} \frac{n!}{\prod_{j=1}^{n} l_j!(j!)^{l_j}} \prod_{j=1:n-r+1}^{\otimes} \boldsymbol{\kappa}_j^{\otimes l_j}\right)\end{aligned} \tag{15}$$

where the summation is over all partitions $\mathcal{K}=\{b_1,b_2,\ldots,b_k\}$ of $1:n$; $|b_j|$ denotes the cardinality of block $b_j$. The simpler second formula, exploiting the symmetrizer matrix, derives from symmetry of $\boldsymbol{\mu}_n^{\otimes}$.

As far as the formula from cumulants to moments (Terdik (2021), Section 3.4) is concerned,

$$\boldsymbol{\kappa}_n^{\otimes} = \mathbf{S}_{d\mathbf{1}_n}\left(\sum_{r=1}^{n}(-1)^{r-1}(r-1)! \sum_{\sum l_j=r,\sum jl_j=n} \prod_{j=1:n-r+1}^{\otimes} \frac{1}{l_j!}\left(\frac{1}{j!}\boldsymbol{\mu}_j^{\otimes}\right)^{l_j}\right). \tag{16}$$

One can also use T-Bell polynomials $\mathbf{B}_n$, and incomplete T-Bell polynomial $\mathbf{B}_{n,r}$ (see, e.g. Section 1.4.2 and Section 4.7 in Terdik (2021)) to show the structure of partitions in (15) as

$$\boldsymbol{\mu}_n^{\otimes} = \mathbf{S}_{d\mathbf{1}_n}\mathbf{B}_n\left(\boldsymbol{\kappa}_1^{\otimes},\boldsymbol{\kappa}_2^{\otimes},\ldots,\boldsymbol{\kappa}_n^{\otimes}\right) \tag{17}$$

and in (16),

$$\boldsymbol{\kappa}_n^{\otimes} = \mathbf{S}_{d\mathbf{1}_n}\sum_{r=1}^{n}(-1)^{r-1}(r-1)!\mathbf{B}_{n,r}\left(\boldsymbol{\mu}_1^{\otimes},\ldots,\boldsymbol{\mu}_n^{\otimes}\right). \tag{18}$$

Consider now the symmetrizer matrix (Section 2.2), note that, by definition, its computation requires $q!$ operations; in the package, the computational complexity is overcome by exploiting the Chacón and Duong (2015) efficient recursive algorithms for functionals based on higher order derivatives and the prime factorization theorem. Note however that even though the computational requirements are greatly reduced, the symmetrizer is a square matrix of dimension $d^n$ where $d$ is the vector dimension and $n$ the power of the Kronecker products. Since `SymMatr` produces essentially sparse matrices, the option `useSparse=T,F` can be selected (`F` is the default). Overall however, if the use of matrices is not strictly necessary, the function `SymIndx` will directly produce a symmetrized version of the T-product of vectors. Table 1 reports the computational times as well as memory allocation of the objects created. Computation times are approximate and obtained on a desktop computer with an Intel-Core-i7-7700 CPU at 3.60GHz and 16GB RAM. From the table it can be noted that using sparse matrices can be much faster and more efficient in terms of memory allocation. The use of `SymIndx` should be the standard way however, as it much less demanding in terms of memory allocation. Recall Example 3 for an instance where the use of `SymMatr` is preferable. The numerical results are in line with those of Chacón and Duong (2015).

In general the functions of the group `Indx` should be used wherever possible, as they are typically much faster than the corresponding `Matr` group since the former avoids constructing possibly large matrices. All code in **MultiStatM** wherever possible, exploits the `Indx` group of functions.

**Table 1:** Approximate memory allocation (in Kb or Mb) and computational times in secs (s) or mins (m) of `SymMatr` and `SymIndx` under different settings. For $d=3$, $n$ indicates the power of the Kronecker products.

| Function | n | Time | Object size |
|---|---|---|---|
| SymMatr() | 4 | 0.003s | 55Kb |
| useSparse=F | 6 | 0.720s | 4Mb |
| | 8 | 9.2m | 344Mb |
| SymMatr() | 4 | 0.03s | 9Kb |
| useSparse=T | 6 | 0.12s | 0.4Mb |
| | 8 | 0.46s | 25Mb |

| Function | n | Time | Object size |
|---|---|---|---|
| Symindx() | 4 | 0.06 | 0.6Kb |
| | 6 | 0.02 | 0.6Kb |
| | 8 | 0.06 | 52Kb |

A further discussion is needed for `CommutatorMatr` with `Type="Kperm"` (Section 2.2) which produces permutations of any T-products of vectors of any dimensions: beyond the considerations concerning using `useSparse=T` or not, when using `CommutatorMatr` with `Type="Kperm"`, the dimensions of the vectors in the argument may have a large impact on computing time.

Recall that `CommutatorMatr` with `Type="Kperm"` produces a permutation `perm` of vectors of dimensions specified in `dims`. As an example consider the case of the product of $n = 8$ vectors all with the same dimension $d$; using `dims <- d` instead of `dims <- c(d,d,d,d,d,d,d,d)` will require much less computing time. In the evaluation below

```
CommutatorMatr(Type = "Kperm", perm = c(2, 4, 6, 1, 3, 8, 5, 7), dims = 3)
```

requires approximately 2.11 seconds while

```
CommutatorMatr(Type = "Kperm", perm = c(2, 4, 6, 1, 3, 8, 5, 7),
               dims = c(3, 3, 3, 3, 3, 3, 3, 3))
```

requires approximately 1326.47 seconds.

**Acknowledgement**. The work of Gy. H. Terdik has been supported by the project TKP2021-NKTA of the University of Debrecen, Hungary. Project no. TKP2021-NKTA-34 has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NKTA funding scheme.

## References

A. Archimbaud, K. Nordhausen, and A. Ruiz-Gazen. Ics for multivariate outlier detection with application to quality control. *Computational Statistics & Data Analysis*, 128:184–199, 2018. ISSN 0167-9473. doi: https://doi.org/10.1016/j.csda.2018.06.011. [p131]

R. B. Arellano-Valle and M. G. Genton. On fundamental skew distributions. *Journal of Multivariate Analysis*, 96(1):93–116, 2005. [p124, 130]

J. M. Arevalillo and H. Navarro. Data projections by skewness maximization under scale mixtures of skew-normal vectors. *Advances in Data Analysis and Classification*, 14(2):435–461, 2020. [p131]

J. M. Arevalillo and H. Navarro. Skewness-based projection pursuit as an eigenvector problem in scale mixtures of skew-normal distributions. *Symmetry*, 13(6):1056, 2021. [p131]

A. Azzalini. *The R package* sn*: The skew-normal and related distributions such as the skew-t and the SUN (version 2.1.0)*. Università degli Studi di Padova, Italia, 2022. URL https://cran.r-project.org/package=sn. Home page: http://azzalini.stat.unipd.it/SN/. [p124]

A. Azzalini and A. Dalla Valle. The multivariate skew-normal distribution. *Biometrika*, 83(4):715–726, 1996. [p124, 130]

D. Bates, M. Maechler, and M. Jagan. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2024. URL https://CRAN.R-project.org/package=Matrix. R package version 1.6-5. [p124]

T. Brenn and S. N. Anfinsen. A revisit of the Gram-Charlier and Edgeworth series expansions. Artikler, rapporter og annet (fysikk og teknologi), Department of Physics and Technology, University of Tromsø, The Arctic University of Norway,, 2017. [p134]

J. E. Chacón and T. Duong. Efficient recursive algorithms for functionals based on higher order derivatives of the multivariate gaussian density. *Statistics and Computing*, 25(5):959–974, 2015. [p123, 127, 137]

E. Di Nardo and G. Guarino. *kStatistics: Unbiased Estimators for Cumulant Products and Faa Di Bruno's Formula*, 2022a. URL https://CRAN.R-project.org/package=kStatistics. R package version 2.1.1. [p124]

E. Di Nardo and G. Guarino. kstatistics: Unbiased estimates of joint cumulant products from the multivariate faà di bruno's formula. *The R Journal*, 14:208–228, 2022b. ISSN 2073-4859. doi: 10.32614/RJ-2022-033. https://doi.org/10.32614/RJ-2022-033. [p124, 132]

C. Franceschini and N. Loperfido. *MaxSkew: Orthogonal Data Projections with Maximal Skewness*, 2017a. URL https://CRAN.R-project.org/package=MaxSkew. R package version 1.1. [p124]

C. Franceschini and N. Loperfido. *MultiSkew: Measures, Tests and Removes Multivariate Skewness*, 2017b. URL https://CRAN.R-project.org/package=MultiSkew. R package version 1.1.1. [p124]

C. Franceschini and N. Loperfido. *Kurt: Performs Kurtosis-Based Statistical Analyses*, 2021. URL https://CRAN.R-project.org/package=Kurt. R package version 1.1. [p124]

A. Genz and F. Bretz. *Computation of Multivariate Normal and t Probabilities*. Lecture Notes in Statistics. Springer-Verlag, Heidelberg, 2009. ISBN 978-3-642-01688-2. [p124]

E. Guidotti. calculus: High-dimensional numerical and symbolic calculus in R. *Journal of Statistical Software*, 104(5):1–37, 2022. doi: 10.18637/jss.v104.i05. [p124]

B. Holmquist. The d-variate vector Hermite polynomial of order. *Linear Alg. and its Appl.*, 237/238: 155–190, 1996. [p123, 124]

W. R. Inc. Mathematica, Version 14.0. URL https://www.wolfram.com/mathematica. Champaign, IL, 2024. [p124]

S. R. Jammalamadaka, T. Subba Rao, and G. H. Terdik. Higher order cumulants of random vectors and applications to statistical inference and time series. *Sankhya (Ser. A, Methodology)*, 68:326–356, 2006. [p123]

S. R. Jammalamadaka, E. Taufer, and G. H. Terdik. Asymptotic theory for statistics based on cumulant vectors with applications. *Scandinavian Journal of Statistics*, 48(2):708–728, 2021a. [p124, 132, 133, 134]

S. R. Jammalamadaka, E. Taufer, and G. H. Terdik. On multivariate skewness and kurtosis. *Sankhya A*, 83-A:607–644, 2021b. [p124, 129, 130, 131]

D. Kahle. mpoly: Multivariate polynomials in R. *The R Journal*, 5(1):162–170, 2013. URL https://journal.r-project.org/archive/2013-1/kahle.pdf. [p124]

T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009. [p123]

T. Kollo. Multivariate skewness and kurtosis measures with an application in ICA. *Journal of Multivariate Analysis*, 99(10):2328–2338, 2008. [p123, 124]

L. Komsta and F. Novomestky. *moments: Moments, Cumulants, Skewness, Kurtosis and Related Tests*, 2022. URL https://CRAN.R-project.org/package=moments. R package version 0.14.1. [p124]

J. A. Koziol. An alternative formulation of neyman's smooth goodness of fit tests under composite alternatives. *Metrika*, 34(1):17–24, 1987. [p124]

J. A. Koziol. A note on measures of multivariate kurtosis. *Biometrical journal*, 31(5):619–624, 1989. [p124, 131]

R. Lai. *arrangements: Fast Generators and Iterators for Permutations, Combinations, Integer Partitions and Compositions*, 2020. URL https://CRAN.R-project.org/package=arrangements. R package version 1.1.9. [p123]

N. Loperfido. Singular value decomposition of the third multivariate moment. *Linear Algebra and its Applications*, 473:202–216, 2015. ISSN 0024-3795. doi: https://doi.org/10.1016/j.laa.2014.05.043. Special issue on Statistics. [p123, 131]

N. Loperfido. A new kurtosis matrix, with statistical applications. *Linear Algebra and its Applications*, 512:1–17, 2017. ISSN 0024-3795. doi: https://doi.org/10.1016/j.laa.2016.09.033. [p123]

J. F. Malkovich and A. A. Afifi. On tests for multivariate normality. *Journal of the american statistical association*, 68(341):176–179, 1973. [p124, 131]

K. V. Mardia. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57: 519–530, 1970. [p124, 131]

P. McCullagh. *Tensor methods in statistics*. Chapman and Hall/CRC, 2018. [p123]

T. F. Móri, V. K. Rohatgi, and G. J. Székely. On multivariate skewness and kurtosis. *Theory of Probability & Its Applications*, 38(3):547–551, 1994. [p123, 124, 131]

F. Novomestky. *matrixcalc: Collection of Functions for Matrix Calculations*, 2021. URL https://CRAN.R-project.org/package=matrixcalc. R package version 1.0-5. [p124]

F. Novomestky. *orthopolynom: Collection of Functions for Orthogonal and Orthonormal Polynomials*, 2022. URL https://CRAN.R-project.org/package=orthopolynom. R package version 1.0-6.1. [p124]

H. Ould-Baba, V. Robin, and J. Antoni. Concise formulae for the cumulant matrices of a random vector. *Linear Algebra and its Applications*, 485:392–416, 2015. [p123]

D. Peña, F. J. Prieto, and J. Viladomat. Eigenvectors of a kurtosis matrix as interesting directions to reveal cluster structure. *Journal of Multivariate Analysis*, 101(9):1995–2007, 2010. [p131]

L. Qi. Rank and eigenvalues of a supersymmetric tensor, the multivariate homogeneous polynomial and the algebraic hypersurface it defines. *Journal of Symbolic Computation*, 41(12):1309–1327, 2006. [p123]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2023. URL https://www.R-project.org/. [p123]

K. D. Smith. A tutorial on multivariate $k$-statistics and their computation. *arXiv preprint arXiv:2005.08373*, 2020. [p132]

G. H. Terdik. Higher order statistics and multivariate vector Hermite polynomials for nonlinear analysis of multidimensional time series. *Theory of Probability and Mathematical Statistics*, (66): 165–186, 2003. [p123]

G. H. Terdik. *Multivariate Statistical Methods: Going Beyond the Linear*. Springer Nature, 2021. [p123, 124, 126, 127, 131, 132, 134, 137]

G. H. Terdik and E. Taufer. *MultiStatM: Multivariate Statistical Methods*, 2024. URL https://CRAN.R-project.org/package=MultiStatM. R package version 2.0.0. [p123]

D. E. Tyler, F. Critchley, L. Dümbgen, and H. Oja. Invariant co-ordinate selection. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 71(3):549–592, 2009. [p131]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL https://www.stats.ox.ac.uk/pub/MASS4/. ISBN 0-387-95457-0. [p124]

*György Terdik*
*University of Debrecen*
*Department of Informatics*
*Debrecen, Hungary*
*ORCiD: 0000-0002-9663-6892*
terdik.gyorgy@inf.unideb.hu

*Emanuele Taufer*
*University of Trento*
*Department of Economics and Management*
*Trento, Italy*
*ORCiD: 0000-0001-5140-9912*
emanuele.taufer@unitn.it

# Canonical Correlation Analysis of Survey Data: The SurveyCC R Package

*by Raul Cruz-Cano, Aaron Cohen, and Erin Mead-Morse*

**Abstract** Classic Canonical Correlation Analysis (CCA) is a popular statistical method that allows for the analysis of the associations between two sets of variables. However, currently it cannot be applied following the published methodological documentation to data sets collected using complex survey design (CSD), which includes factors, such as replicate weights, clusters, and strata, that are critical for the accurate calculation of the statistical significance of any correlation. To close this gap, we have developed the Survey CC algorithm and implemented it in an R package. We describe the theoretical foundations of our algorithm and provide a detailed report of the options of the function that performs it. Moreover, the application of our newly developed method to several national survey data sets shows the differences in conclusions that can be reached if the CSD elements are not taken into consideration during the calculation of the statistical significance of the canonical correlations.

## 1 Introduction

Classical Canonical Correlation Analysis (CCA) is a popular exploratory statistical method that allows for the analysis of relationships between two sets of variables, offering several advantages over other statistical techniques, including the ability to limit the probability of committing Type I errors when examining more than two variables (Hair et al., 2019). CCA has been applied to diverse fields, including ecology (Gittins, 2012), neuroscience (Zhuang and Yang, 2020), biomedical imaging (Correa et al., 2008), human geography (Clark, 1975), effect of work place regulations (Borland et al., 1991), illicit drug use (Leo and Wulfert, 2013) and tobacco use (Morris et al., 2018).

Current methods for CCA inference and interval estimation, however, depend on simple random sampling, and are, therefore, inaccurate when data is collected using a complex survey design (CSD). Or, they are based on methods that cannot be applied following the published CSD methodological documentation, hence, leading to results without practical significance. To overcome these issues, we will employ our recently developed, innovative Survey CC method.

Using CCA, investigators study the extent and nature of the correlation between two sets of variables, $X = \{X_1, X_2, ..., X_p\}$ and $Y = \{Y_1, Y_2, \ldots, Y_q\}$ (Hotelling, 1936). As described by (Clark, 1975), without loss of generality, it is assumed that $p \geq q$ and that each and every variable in $X$ and $Y$ have been standardized to have a mean zero and a variance equal to one. CCA is used to explore sample correlations between $X$ and $Y$ observed on the same experimental units by analyzing the coefficients, $A_1 = (a_1, a_2, \ldots, a_p)^T$ and $B_1 = (b_1, b_2, \ldots, b_q)^T$, which maximize the correlation between linear combinations of the variables $X$ and $Y$ subject to $Var(XA_1) = Var(YB_1) = 1$.

The "classic" canonical correlation can be defined as (Hotelling, 1936):

$$\rho_1 = max_{A_1,B_1} Corr(XA_1, YB_1) = max_{A_1,B_1} \frac{Cov(XA_1, YB_1)}{\sqrt{Var(XA_1) \cdot Var(YB_1)}} \tag{1}$$

subject to $Var(XA_1) = A_1^T X^T X A_1 = B_1^T Y^T Y B_1 = 1$, as described by (Hardoon and Shawe-Taylor, 2011). These linear combinations, $U_1 = XA_1$ and $V_1 = YB_1$, are called the first canonical variates. Their correlation, $\rho_1$, is the first canonical correlation, and the coefficients $A_1$ and $B_1$ are the first canonical coefficients.

The first canonical coefficients in each set will help to identify variables from $X$ and $Y$ that are associated with each other. Variables with a strong positive relationship will have coefficients with large magnitudes and similar signs (positive or negative), whereas inverse relationships will have coefficients with opposing signs. The coefficients' magnitudes are commonly examined to determine which variables are the most relevant as variables with small coefficients will have weak relationships.

The process can be duplicated to identify the coefficients, $A_2$ and $B_2$, that maximize the correlation, $\rho_2$, subject to $Var(XA_2) = Var(YB_2) = 1$, with the additional constraint that the new canonical variates, $U_2 = XA_2$ and $V_2 = YB_2$, are uncorrelated with the first pair of canonical variates. This process can be repeated $q$ times. These secondary canonical correlations are important because not all the significant relationships that exist among variables can be expressed in the first canonical correlation.

Many researchers base their conclusions regarding the relationships among the variables examined in a CCA on the canonical coefficients associated with statistically significant canonical correlations

(Boedeker and Henson, 2020) (Hill and Lewicki, 1997), hence, tools able to accurately estimate their $p$-values are necessary. For example, (Travis and Lagrosen, 2014) limited their conclusions to the information inferred from the first set of canonical coefficients because only the first canonical correlation had a significant $p$-value. In (Stowe et al., 1980), three canonical correlations have statistically significant $p$-values, hence all three sets of canonical coefficients are studied.

For the sample cross-correlation matrix, $\begin{pmatrix} S_{XX} & S_{XY} \\ S_{YX} & S_{YY} \end{pmatrix}$, canonical correlations can be calculated by finding the descending-ordered square roots of the matrix eigenvalues $\mathbf{M} = S_{YY}^{-1} S_{YX} S_{XX}^{-1} S_{XY}$. The canonical coefficients for $Y$ are the corresponding $q$ eigenvectors, and the elements of eigenvector $i$ are the canonical coefficients $B_i$. Correlations between canonical variates and original variables (called canonical factor loadings) offer information that is complementary to canonical coefficients and, hence, are preferred by some researchers (Meredith, 1964).

## 2 Weighted CCA

Broadly speaking, when data sets have been collected using CSD methods, two main types of factors are used to account for the fact that the sample is not a simple random sample:

- Survey weights are used to create a synthetic sample for which the distribution of covariates is similar to the population under study, i.e., they help to account for unequal selection probability (Hahs-Vaughn et al., 2011), leading to correct population parameter estimates (Lewis, 2016).
- Repetition weights, clusters, and strata are CSD elements essential to consider the nonindependence among the sampling units derived from the complex sampling design utilized to gather the data (Hahs-Vaughn et al., 2011). Hence, they are used during the calculation of standard errors to accurately estimate confidence intervals and $p$-values for any hypothesis tested.

Standard statistical methods that ignore structures, such as clustering or stratification, can give seriously misleading results (Holt et al., 1980). Analyzing a stratified sample as if it were a simple random sample tends to overestimate the standard errors, whereas analyzing a clustered sample as if it were a simple random sample tends to underestimate the standard errors, and analyzing the data without taking into consideration the survey weights will lead to incorrect point estimates (Lumley, 2004). Examples of how intracluster correlations can severely affect the effective sample size can be found in (Killip et al., 2004).

From (Winship and Radbill, 1994), we know that the formulas for weighted covariance and variance for two variables, $X_1$ and $X_2$, are:

$$Weighted\ Cov(X_1, X_2) = \frac{\sum_{i=1}^{n} W_i (X_{i1} - \bar{X_1})(X_{i2} - \bar{X_2})}{\sum_{i=1}^{n} W_i} \tag{2}$$

$$Weighted\ Var(X_1) = \frac{\sum_{i=1}^{n} W_i (X_{i1} - \bar{X_1})^2}{\sum_{i=1}^{n} W_i} \tag{3}$$

where $W$ is the vector that stores the survey weight for each of the $n$ subjects in the sample.

If we choose $A_j$ and $B_j$, such that $WeightedVar(U_j) = WeightedVar(V_j) = 1$, then substituting (2) and (3) into (1) reduces the weighted canonical correlation to

$$Weighted\ \rho_j = max_{A_j, B_j} \frac{\sum_{i=1}^{n} W_i ((XA_j)_i (YB_j)_i)}{\sum_{i=1}^{n} W_i} \tag{4}$$

It can be proven that the $q$ canonical coefficients $A_j$ are the $q$ eigenvectors of:

$$|(X^T diag(W) Y) \cdot (Y^T diag(W) Y)^{-1} \cdot (Y^T diag(W) X) \cdot (X^T diag(W) X)^{-1} - \lambda^2 I| = 0 \tag{5}$$

The $q$ canonical coefficients $B_j$ can be found similarly. As in the classic CCA, the $q$ eigenvalues, $\lambda$, are the weighted canonical correlations; hence, parameters $A$ and $B$ are not needed to find the weighted canonical variates and correlations, as this calculation derives their values.

Modern statistical software, including SAS 9.4 software's PROC CANCORR (SAS, 2012), Stata's canon command (StataCorp, 2023b), and R's `cancor()` function in the **candisc** package, apply these formulas and are able to consider survey weights in their calculations of weighted canonical correlations and their corresponding $p$-values according to the methods Wilks' lambda, Pillai's trace, and Hotelling-Lawley trace (Caliński et al., 2006) and Roy's largest root (Johnstone, 2009). However, these programs are unable to account for other factors, such as clusters and strata. As mentioned

previously, ignoring these complex survey design elements will lead to different estimates of $p$-values for weighted canonical correlations.

The SAS macro presented in (Nelson et al., 2020) accounts for the CSD factors by calculating a design-effect adjusted weight for each variable. The way in which these calibrated weights are calculated tends to result in design-effect adjusted sample sizes large enough to consistently provide trivially significant results, i.e., statistically significant $p$-values for canonical correlations regardless of their magnitude and practical meaning. The authors of this algorithm provide a recommendation to ameliorate this problem but their advice requires the user to choose a random variable as the dependent variable, hence leading to inconsistent results that are influenced by this selection. In contrast, our proposed algorithm leads to the same degrees of freedom recommended for linear regression models outlined in the complex surveys' guidelines.

## 3 Complex survey design CCA

The proposed Survey CC method is based on three well-known ideas related to data collected using CSD: (1) the survey weights are sufficient to calculate correct parameter estimates, e.g., the canonical coefficients; (2) the linear combination of variables in a complex survey data set follows the same CSD structure as the original variables; and (3) the $p$-values for a simple linear regression coefficient and a correlation using the same two variables are equal.

In the case of (3), due to the linearization of the variance estimator used by `survey::svyglm()`, the $p$-value for the slope of the regression of $U_j$ on $V_j$ will not be equal to the $p$-value for the regression of $V_j$ on $U_j$, as would be the case if there were no CSD elements. We addressed this issue by calculating both $p$-values and then selecting the largest of them, i.e., we took a conservative approach for the test of the hypothesis $H_0 : \rho_j = 0$ (Nelson et al., 2020).

**Survey CC step-by-step methodology:**

1. Select variable sets, $X$ and $Y$, from existing data sets.
2. Include survey weights in the calculation of the weighted canonical coefficients and correlations to find their point estimates.
3. For each $j$ canonical correlation where $1 \leq j \leq q$:
   (a) Use weighted canonical coefficients to calculate weighted canonical variates, $U_j$ and $V_j$.
   (b) Perform a simple linear regression of $U_j$ on $V_j$ that includes all complex survey design elements (e.g., survey weights, cluster, and strata) to find the $p$-values for their corresponding $r_j$ canonical correlation
   (c) Perform a simple linear regression of $V_j$ on $U_j$ that includes all complex survey design elements (e.g., survey weights, cluster, and strata) to find the $p$-values for their corresponding $r_j$ canonical correlation
   (d) Compare the $p$-values of the linear regression models in the previous two steps and select the largest.

In practice, existing computational CCA procedures that consider survey weights can be used to identify weighed canonical coefficients, which, in turn, can be used to calculate the weighted canonical variates. Once these canonical variates have been identified, given that they are just linear combinations of the original variables, $p$-values for canonical correlations can be estimated using complex survey linear regression procedures, as linear regression survey procedures in modern statistical software can include not only survey weights but also information regarding the repetition weights, clusters, and strata from the original data set. The values required for these CSD factors are provided in the user guides, codebooks, and manuals of the surveys that would be examined by the users of our package. In other words, the users of our package will not need to determine these values; they will be able to find these values using the help materials provided by the institutions that conduct the surveys and make them available to the public.

A further advantage of our Survey CC algorithm is that it will allow the assessment of the statistical significance of each canonical correlation, and hence the conclusions that might be drawn from the associated canonical coefficients or loadings, to be done separately. The reason is that a byproduct of the proposed algorithm are the $p$-values for each individual $r_j$ canonical correlation (i.e., $H_0 : \rho_j = 0$), whereas current software tests the hypothesis $H_0 : \rho_j = \rho_{j+1} = \cdots = \rho_{q-1} = \rho_q = 0$. This sequential testing approach followed by the classic statistical methods has been criticized **?**. Because the effective sample size might change for each canonical correlation depending on the canonical coefficients and the CSD factors, a larger correlation magnitude might not imply a smaller $p$-value, therefore secondary canonical correlations might be statistically significant even if the first one is not.

This difference on the hypotheses being tested also implies that Survey CC should be considered an alternative to the existing CCA methods, not an improvement on them. Moreover, this also

means that the $p$-values produced by Survey CC are not directly comparable to those provided by the classic statistical methods. Therefore, when it is mentioned that these statistical tests lead to different conclusions, it refers to the interpretation that a general scientist with basic training in statistics would give to the $p$-values, not to the formal interpretation that should be strictly given to them.

## 4 The R package

The package **SurveyCC** extends the functionality of `candisc::cancor()` by calculating the test statistics, degrees of freedom and $p$-values necessary to estimate and interpret the statistical significance of the canonical correlations according to the methods of Wilks' lambda, Pillai's trace, and Hotelling-Lawley trace (Caliński et al., 2006). The statistics for these secondary canonical correlations were, up until now, unavailable to the users of Stata's canon and SAS PROC CANCORR. In R, the repeated application of the function `p.asym()` in the **CCP** package, once for each of these classic methods, is needed as `candisc::cancor()` provides by the default only the results for the Wilks' lambda test. It is worth noting that the existing statistical software uses an $F$-distribution to calculate the $p$-values for these methods, while those provided by **SurveyCC** are based on a $\chi^2$ distribution. This difference leads to slightly different results from those provided by the existing CCA commands. Using a different distribution makes the information provided by **SurveyCC** complementary, instead of redundant, to that provided by the existing CCA commands.

Roy's largest root test statistics are also provided by **SurveyCC**. Our implementation is based on the idea described in (Johnstone, 2009), which posits the use of the first canonical correlation as a test statistic for all secondary canonical correlations and simply changing the degrees of freedom of the $F$-distribution used to assess it. Other existing CCA functions provide the Roy's largest root test statistics only for the first canonical correlation. Additionally, we found in the examples included in this paper and countless preliminary experiments, that Roy's largest root $p$-value was smaller than the rest of those calculated by **SurveyCC**. We hypothesize that this difference might be due to the upper-bound method used by Johnstone (2009), which is the base of our programming work for this test. Although this might limit the usefulness of this statistic, we decided to include it in our calculations and leave it to the user to determine how much importance to give to the results of this method.

Moreover, **SurveyCC** implements the proposed Survey CC algorithm described in the previous section. The results of the Survey CC algorithm appears in the tables displayed by `surveycc()` as `Complex Survey CC`. As mentioned previously, the core idea of the algorithm is to calculate the correlations among the canonical variates and their corresponding statistical significance via an equivalent sequence of univariate linear regressions. This transformation allows the user to take advantage of the existing theoretical and computational resources that integrate the CSD elements into these regression models (Valliant and Dever, 2018). Hence, **SurveyCC** can include the same complex design elements as `survey::svyglm()`. Indeed, for the main function `surveycc()`, one of the main required inputs is a `survey::svydesign()` or `survey::svrepdesign()` object (see section 4.1 below). The `Statistic` for our algorithm is the coefficient for independent variable in the univariate linear regression model and, given that the variances of the canonical variates have been setup to be 1, it is equal to the corresponding canonical correlation. The other values in the results table for **SurveyCC**, i.e. `df1`, `df2`, `Chi-Sq/F` and `p-val`, are those provided by R for the coefficient estimate in simple linear regression model (for more information see the help material for `survey::svyglm()` in the **survey** package).

Beside the $p$-value corresponding to the Survey CC, `surveycc()` also provides statistics for the case in which only the survey weights are taken into consideration, while the rest of the complex survey design elements are ignored. This additional set of results, which are listed as `Weighted Survey CC`, allows the user to appreciate the effect that the elements, which are dispensable to obtain correct point estimates, affect the calculation of the statistical significance of the correlations.

The units and variables graphs (Gittins, 2012) can be drawn by `candisc::cancor()`, further complementing the information listed by the existing canonical correlation commands. The variables graph uses the first and second canonical coefficients for each the variables in the original data as coordinates in a 2-dimensional graph. This graph allows the user to see the associations that might exist among the variables in the original data set by studying their positions around the unit circle. Variables in opposite sides are expected to have inverse relationships, i.e., when one increases the other decreases, while variables close to each other should have positive relationships, i.e., when one increases/decreases, the other also increases/decreases. The distance from the origin can be interpreted as a direct proportional measure of the strength of these relationships. The units graph is created by multiplying the values of the original variables of each observation by the canonical coefficients of the two canonical correlations selected by the user and then using the resulting values as coordinates in a two-dimensional plane described by the corresponding canonical variates. The

units graphs serves to identify possible clusters among the observations in the data set that share characteristics which might not have been apparent in the original variables (Cruz-Cano and Lee, 2014) and identify observations that might be considered outliers. For example, in a units graph, samples from grassland vegetation in a small area in North Wales formed three distinct clusters based on the community (limestone, neutral, and fertile grassland) to which they belonged (Gittins, 2012).

The **SurveyCC** package is currently available in the Comprehensive R Archive Network (CRAN): https://cran.r-project.org/web/packages/SurveyCC/index.html.

### 4.1 Syntax

The **SurveyCC** package has one main user-facing function, `surveycc()`. The arguments for this function are as follows:

- `design_object`
- `var.x, var.y`
- `howmany`
- `dim1, dim2`
- `selection`

The first item, the design object, needs to be created by using the **survey** functions `svydesign()` or `svrepdesign()`. This object will contain the actual data set, but will also keep track of all CSD characteristics, such as cluster, strata, etc. Note that, even if the data does not have a complex survey design structure, the design object still needs to be created and passed in (see "Simple Example" below).

The `var.x` and `var.y` arguments specify the names of the first and second sets of variables, respectively. These both should be character vectors, and the names should exactly match the desired variable names in the data set.

The `howmany` argument is a positive integer specifying how many canonical correlations one wants to test for. Note that this always must be between 1 and the cardinality of the largest set of variables (i.e., `var.x` or `var.y`).

The `selection` argument specifies if the effective sample size should simply be the number of observations/rows, or whether it should be based on the sum of the weights in the survey design object. Specifying `selection = "FREQ"` will result in the weight information being used, `selection = ROWS` will result in the number of rows being used. The effective sample size is used in the calculation of $p$-values and hence included in the result tables.

### 4.2 Output

The output of the call to `surveycc()` in R is a list with the following structure:

- `Stats.cancor`
  - `Stats.cancor.1`
  - `Stats.cancor.2`
  - `...`
- `cc_object`

The first item in this list is another list named `Stats.cancor`, and it contains a data frame for each canonical correlate (the number being obtained from the argument `howmany`). Each data frame is the table of test results, including the results of the new Survey CC method.

The second item in this list is another list named `cc_object`, which is actually the output from the call to `cancor()` in the **candisc** package. It is beyond the scope of the present article to describe everything in the `cancor()` output as it is already done extensively in its help files, but special attention should be paid to the object `cc_object$coef`, which gives the actual canonical variate coefficient estimates. This will be explored in some of the examples below.

The **SurveyCC** package contains a plotting method for the `surveycc` object. Simply pass the saved object to `plot()`, along with two arguments, `dim1` and `dim2`, which specify which variates to use as axes in the plots. The values must both be positive integers, cannot equal each other, and must be $\leq$ `howmany`. Please see the PATH example below (PATH Study Wave 1 Adult Questionnaire Example) for an example of this.

## 5 Examples

In this section we present four examples to illustrate different aspects of **SurveyCC** being exalted. Although all the data sets mentioned in this section are publicly available, we have also included them as supplementary files for the convenience of the readers who would like to run the code provided in this document.

These data sets are:

- **Simple Automobile data set**: This example shows that `surveycc()` can deal with data sets without CSD factors and provide information about the statistical significance of the secondary canonical correlations, which is currently difficult to obtain using exiting CCA software. In the past it has been used as an example for other canonical correlation packages (StataCorp, 2023a). The data set is available at `https://www.stata-press.com/data/r17/auto`.
- **2007-2010 National Health and Nutrition Examination Survey**: This case illustrates the application of **SurveyCC** to study a survey that includes a complete set of CSD factors (survey weights, cluster and strata). The data set is available at `https://wwwn.cdc.gov/nchs/nhanes/Default.aspx`. The creation of this data set required merging of the 2007-2008 and 2009-2010 surveys according to the specifications provided in the *National Health and Nutrition Examination Survey:Estimation Procedures, 2007–2010 Data Evaluation and Methods Research* of the CDC's National Center for Health Statistics.
- **PATH Study Wave 1 Adult Questionnaire**: This example is based on a national survey data set with replicate weights instead of cluster and strata for standard error estimation. This data set is available at `https://www.icpsr.umich.edu/web/NAHDAP/studies/36231/datadocumentation`. To reduce the PATH data set to a size that would not impede the installing of **SurveyCC** by persons with little memory space in their computers or unreliable or slow internet connections, we provide a subset of PATH Wave I Adult Questionnaire data set that can be downloaded from the PATH Study website. Only the rows that have no missing values for the variables used in the example, i.e., only the rows that are in fact used during the CCA calculations, were kept.
- **2021 National Youth Tobacco Survey**: The last of our national survey examples presents a case in which the `surveycc()` handles a larger set of variables and leads to conclusions different from those obtained if only the classic statistical tests are applied. The data set is available at `https://www.cdc.gov/tobacco/data_statistics/surveys/nyts/data/index.html`.

We recommend to the readers who want to replicate the results listed in this manuscript running the reproducibility scripts provided in the supplementary files as they include ancillary commands, e.g., `library`, needed for the correct execution of our examples.

### 5.1 Simple Automobile Example

The first of our examples serves to show how `surveycc()` can handle cases in which no CSD factors exist and the user might only be interested in examining the statistical significance of the secondary canonical correlations. For this purpose, we examine the relationships among several measures related to different models of automobiles. One group of variables express the size of the automobile (length, weight, headroom, and trunk size) while the others are related to the performance of it (displacement in cubic centimeters, miles per gallon, gear ratio, and turn ratio).

#### Simple Automobile Example Setup

As previously mentioned, it is necessary to create the CSD object using the **survey** package, which is then passed into the main function of the current package, `surveycc()`. This is the case even if, as in the present example, there are no CSD factors in the data set (see the design_object in the code below). Notice that the only necessary information for the creation of a survey design object is `ids`, set to `~ 1` here.

In addition to the CSD object, we define vectors for each set of variables, as well as an indicator of the desired number of canonical variates for which that statistical significance must be reported. These objects are all passed into the `surveycc()` function.

```
# Simple example
design_object <-
  survey::svydesign(
    ids = ~1,
    data = auto
```

```
  )
var.x <- c("length", "weight", "headroom", "trunk")
var.y <- c("displacement", "mpg", "gear_ratio", "turn")
howmany <- 3
out.auto <- surveycc(design_object = design_object, var.x = var.x,
  var.y = var.y, howmany = howmany, selection = "FREQ")
```

### Simple Automobile Example Results

Here `surveycc()` is used to estimate canonical correlations $p$-values for the Simple example.

See the output of the saved `out.auto` object below for the tables containing all statistics, degrees of freedom, test statistics, and $p$-values.

```
out.auto
```

```
#> $Stats.cancor
#> $Stats.cancor$Stats.cancor.1
#>                         Statistic df1 df2  Chi-Sq/F p-val
#> Wilks' Lambda             0.08973  16  NA 165.42306     0
#> Pillai's Trace            1.01956  15  NA  73.52406     0
#> Hotelling-Lawley Trace    8.93344  16  NA 581.68905     0
#> Roy's Greatest Root       0.89792   4  69 151.74255     0
#> Weighted Survey CC        0.94759  73  72  27.56469     0
#> Complex Survey CC         0.94759  73  72  27.56469     0
#>
#> $Stats.cancor$Stats.cancor.2
#>                         Statistic df1 df2  Chi-Sq/F   p-val
#> Wilks' Lambda             0.87907   9  NA   9.82963 0.36445
#> Pillai's Trace            0.12163   8  NA   9.58017 0.38553
#> Hotelling-Lawley Trace    0.13677   9  NA  10.08844 0.34337
#> Roy's Greatest Root       0.89792   4  69 151.74255 0.00000
#> Weighted Survey CC        0.34003  73  72   2.76224 0.00728
#> Complex Survey CC         0.34003  73  72   2.76224 0.00728
#>
#> $Stats.cancor$Stats.cancor.3
#>                         Statistic df1 df2  Chi-Sq/F   p-val
#> Wilks' Lambda             0.99399   4  NA   1.95328 0.74435
#> Pillai's Trace            0.00601   3  NA   1.95903 0.74329
#> Hotelling-Lawley Trace    0.00603   4  NA   1.94751 0.74541
#> Roy's Greatest Root       0.89792   4  69 151.74255 0.00000
#> Weighted Survey CC        0.06338  73  72   0.60644 0.54613
#> Complex Survey CC         0.06338  73  72   0.60644 0.54613
#>
#>
#> $cc_object
#>
#> Canonical correlation analysis of:
#>   4   X  variables:  length, weight, headroom, trunk
#>   with    4   Y  variables:  displacement, mpg, gear_ratio, turn
#>
#>      CanR   CanRSQ    Eigen  percent    cum                           scree
#> 1 0.94759 0.897924 8.796670 98.46902  98.47 ******************************
#> 2 0.34003 0.115619 0.130734  1.46343  99.93
#> 3 0.06338 0.004017 0.004033  0.04514  99.98
#> 4 0.04470 0.001998 0.002002  0.02241 100.00
#>
#> Test of H0: The canonical correlations in the
#> current row and all that follow are zero
#>
#>      CanR LR test stat approx F numDF  denDF Pr(> F)
#> 1 0.94759      0.08973  15.1900    16 202.27  <2e-16 ***
#> 2 0.34003      0.87907   0.9863     9 163.21  0.4534
#> 3 0.06338      0.99399   0.1026     4 136.00  0.9814
```

```
#> 4 0.04470      0.99800   0.1381     1  69.00  0.7113
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> attr(,"class")
#> [1] "surveycc" "list"
```

Notice that `Stats.cancor` does not include the canonical coefficients and hence it would not be useful to draw conclusions about the relationships among the sets of variables included in the CCA. To see the canonical correlations and coefficient estimates one would save the output of the function call, and then refer to the sub-object `cc_object`:

```
out.auto$cc_object$coef
```

```
#> $X
#>                 Xcan1        Xcan2         Xcan3         Xcan4
#> length   -0.009477879  0.144140174 -0.0329362330 -0.0211707964
#> weight   -0.001016175 -0.003663031  0.0009582644 -0.0006830296
#> headroom -0.035113185 -0.370142988 -1.5361072863  0.0440329417
#> trunk     0.002282333 -0.034273898  0.2135491362  0.3252799822
#>
#> $Y
#>                     Ycan1       Ycan2        Ycan3         Ycan4
#> displacement -0.005370441 -0.01254983 -0.019129726  0.0005364212
#> mpg           0.046148112 -0.04127765 -0.068339793 -0.2478094648
#> gear_ratio   -0.032958321  1.02797976 -3.659566938  1.0311435250
#> turn         -0.079392663  0.31126673 -0.003303379 -0.2240488108
```

### Simple Automobile Example Conclusion

In this example the weighted Survey CC provides accurate results, because weights (equal to 1 in the case of simple random sampling) are the only survey design factor. For the first canonical correlation, all methods conclude that it is statistically significant and hence it is reasonable to assume that the conclusions drawn might be true, i.e., a positive relationship between the headroom of the cars (negative coefficient -0.0351) and its turn ratio (negative coefficient -0.0794) but a negative one with its MPG (positive coefficient 0.0461) are likely to exist. The second canonical coefficients represent the negative association that exist between the headroom (negative coefficient -0.3701) and the gear ratio (positive coefficient 1.0279) combined with the turn ratio (positive coefficient 0.3112).

Although the $p$-values for the Survey CC method and most of those provided by the classic tests led to the same conclusions for the first (statistically significant) and third canonical correlation (not statistically significant), different conclusions are reached for the second canonical correlation. Considering all these results together can help the user of the commands to reach a more informed judgment about the robustness of their conclusions.

## 5.2 2007-2010 National Health and Nutrition Examination Survey (NHANES) Example

The Survey CC method was applied to the 2007-2010 NHANES data set to evaluate the existing relationship between select demographic factors (`ridageyr`: Age in years, at the time of the screening interview, and `indhhin2`: the estimated total household income) and body measurements related to obesity (`bmxbmi`: BMI, `bmxwaist`: Waist circumference, `bpxpls`: Resting pulse per minute, `bpxsy1`: Systolic blood pressure, and `bpxdi1`: Diastolic blood pressure) for people between 45 and 64 years old, which is the age group with the largest proportion of persons with undiagnosed diabetes, considering the appropriate CSD factors (survey weights, cluster and strata). Although many other demographic factors such are race/ethnicity and education would be considered in a serious study about the relationship between demography and body measure, the purpose of this example is to present a relatively simple case that includes CSD factors such as cluster, strata and survey weights.

### NHANES Example Setup

As previously mentioned, it is necessary to create the CSD object using the **survey** package, which is then passed into the main function of the current package, `surveycc()`.

```
reducedNHANESdata <- NHANESdata %>%
  dplyr::filter(ridageyr <= 64 & ridageyr >= 45)

design_object <-
  survey::svydesign(
    id = ~sdmvpsu,
    weights = ~wtmec4yr,
    strata = ~sdmvstra,
    nest = TRUE,
    data = reducedNHANESdata
  )
var.x <- c("bmxwaist", "bmxbmi", "bpxpls", "bpxdi1", "bpxsy1")
var.y <- c("ridageyr", "indhhin2")
out.NHANES <- surveycc(design_object, var.x = var.x, var.y = var.y,
                howmany = 2, selection = "ROWS")
```

### NHANES Example Results

Here `surveycc()` is used to estimate canonical correlations $p$-values for the NHANES example. Notice that the specification of the CSD elements was adapted from the section *Sample Code* webpage from the *Tutorials* tab of NHANES website https://wwwn.cdc.gov/nchs/nhanes/tutorials/samplecode.aspx.

See the output of the saved `out.NHANES` object below for the tables containing all statistics, degrees of freedom, test statistics, and $p$-values.

```
out.NHANES
```

```
#> $Stats.cancor
#> $Stats.cancor$Stats.cancor.1
#>                         Statistic  df1  df2  Chi-Sq/F p-val
#> Wilks' Lambda             0.89351   10   NA 131.69647     0
#> Pillai's Trace            0.10680    9   NA 125.23226     0
#> Hotelling-Lawley Trace    0.11884   10   NA 138.63868     0
#> Roy's Greatest Root       0.10384    5 1160  26.88222     0
#> Weighted Survey CC        0.30978 1165 1164   8.80881     0
#> Complex Survey CC         0.30978 1165   31   7.11893     0
#>
#> $Stats.cancor$Stats.cancor.2
#>                         Statistic  df1  df2 Chi-Sq/F   p-val
#> Wilks' Lambda             0.99704    4   NA  4.46159 0.34713
#> Pillai's Trace            0.00296    3   NA  4.46091 0.34721
#> Hotelling-Lawley Trace    0.00297    4   NA  4.46227 0.34704
#> Roy's Greatest Root       0.10384    4 1161 33.63175 0.00000
#> Weighted Survey CC        0.05603 1165 1164  1.95003 0.05141
#> Complex Survey CC         0.05603 1165   31  2.05636 0.04825
#>
#>
#> $cc_object
#>
#> Canonical correlation analysis of:
#>   5   X  variables:  bmxwaist, bmxbmi, bpxpls, bpxdi1, bpxsy1
#>   with    2   Y  variables:  ridageyr, indhhin2
#>
#>      CanR  CanRSQ   Eigen percent    cum                           scree
#> 1 0.30978 0.09597 0.10615  97.119  97.12 ******************************
#> 2 0.05603 0.00314 0.00315   2.881 100.00 *
#>
#> Test of H0: The canonical correlations in the
#> current row and all that follow are zero
#>
#>       CanR LR test stat approx F numDF denDF Pr(> F)
#> 1 0.309784      0.90120  12.3766    10  2318  <2e-16 ***
#> 2 0.056033      0.99686   0.9134     4  1160  0.4553
#> ---
```

```
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> attr(,"class")
#> [1] "surveycc" "list"
```

### NHANES Example Conclusion

Similarly to the 'Simple Automobile Example' above, we can save the output of `surveycc()` to extract the canonical coefficients themselves (one could also directly apply the `candisc::cancor()` function, as is done internally in the present package).

```
out.NHANES$cc_object$coef
```

```
#> $X
#>                 Xcan1        Xcan2
#> bmxwaist -0.070475488  0.022913764
#> bmxbmi    0.129348798 -0.171143233
#> bpxpls    0.007807571 -0.034438475
#> bpxdi1    0.054467924 -0.010895040
#> bpxsy1   -0.051782536  0.009062863
#>
#> $Y
#>                Ycan1      Ycan2
#> ridageyr -0.17278878 0.05291342
#> indhhin2  0.02498235 0.07249308
```

The canonical coefficients from CCA are shown in the above output. In the first set of canonical variables, we will focus on the coefficients with the greatest magnitudes: `bmxbmi` (first set) and `ridageyr` (second set). The results show an opposite relationship between Age (negative canonical coefficient -0.1728) and BMI (positive canonical coefficient 0.1294); that is, BMI decreases as age increases for people ages 45-64 years. This correlation is considered statistically significant regardless of the method and has been observed in previous studies such as (Yang et al., 2021), (Amies-Cull et al., 2022), (Sun et al., 2022) and (Jarrett et al., 2009). The dominant second canonical coefficients express a negative association between a combined increase in age (positive canonical coefficient 0.0529) and an increase in income (positive canonical coefficient 0.0725) and a decrease in BMI (negative canonical coefficient -0.1712). In other words, increases in income and age are associated with decreases in BMI. The negative relationship between income and BMI in adults has been seen before in the literature, especially among women (Garcia Villar and Quintana-Domeque, 2009).

Studying the results of this section, the conclusion about the statistical significance of the second canonical correlation would be different depending on whether a classic test or the Survey CC method is used for this purpose. For example, Wilks' lambda $p$-value is 0.3471 and Pillai's trace $p$-value is 0.3472, both leading to the conclusion that it is not statistically significant. In contrast, Survey CC, which accounts for the complex survey design elements, leads to the opposite conclusion that the correlation is statistically significant ($p$= 0.048).

### 5.3 PATH Study Wave 1 Adult Questionnaire Example

This example includes material about how the variables and units graphs can be used to examine the associations between certain smoking- and drinking-related variables. The evaluation of the statistical significance of the canonical correlations takes in to consideration the replicate weights needed for their accurate calculation. Once again, a serious examination between smoking and drinking alcohol would require a larger number of variables but given the purpose of this manuscript, we believe that it is acceptable to present this preliminary analysis to further illustrate how `surveycc()` can help researchers extract information from national survey data sets with replicate weights. Moreover, it is important to notice that on occasion the classic statistical tests and the proposed Survey CC method lead to the same conclusions as is the case presented in this subsection.

The smoking variables are:

- `R01_AC1022`: In past 30 days, number of days smoked cigarettes.
- `R01_AE1022`: In past 30 days, number of days used an e-cigarette.
- `R01_AG1022CG`: Number of days smoked cigarillos in past 30 days.

The variables related to alcohol drinking are:

- `R01_AX0075`: Number of days drank one or more drinks of an alcoholic beverage in past 30 days.
- `R01_AX0076`: Number of alcoholic drinks usually consumed each day on days drank in past 30 days.

### PATH Study Example Setup

As previously mentioned, it is necessary to create the CSD object using the **survey** package, which is then passed into the main function of the current package, `surveycc()`.

In addition to the CSD object, we also define here vectors for each set of variables, as well as an indicator for how many canonical variates we want. These objects are all passed into the `surveycc()` function.

```
design_object <-
 survey::svrepdesign(
 id = ~PERSONID,
 weights = ~R01_A_PWGT,
 repweights = "R01_A_PWGT[1-9]+",
 type = "Fay",
 rho = 0.3,
 data=reducedPATHdata,
 mse = TRUE
 )
var.x <- c("R01_AC1022", "R01_AE1022", "R01_AG1022CG")
var.y <- c("R01_AX0075", "R01_AX0076")
howmany <- 2
out.PATH <- surveycc(design_object, var.x, var.y, howmany = howmany,
  selection = "ROWS")
```

### PATH Study Example Results

Here `surveycc()` is used to estimate canonical correlations $p$-values for the PATH Study example. Notice that the specification of the CSD elements in the function `survey::svrepdesign()` comes directly from the *Population Assessment of Tobacco and Health (PATH) Study [United States] Public-Use Files ICPSR Public-Use Files User Guide*. These files can be found at the PATH Study website https://www.icpsr.umich.edu/web/NAHDAP/studies/36231. See the output of the saved `out.PATH` object below for the output table containing all statistics, degrees of freedom, test statistics, and $p$-values.

```
out.PATH
```

```
#> $Stats.cancor
#> $Stats.cancor$Stats.cancor.1
#>                         Statistic df1 df2 Chi-Sq/F   p-val
#> Wilks' Lambda             0.93636   6  NA  9.47145 0.14875
#> Pillai's Trace            0.06395   5  NA  9.33935 0.15537
#> Hotelling-Lawley Trace    0.06763   6  NA  9.60680 0.14222
#> Roy's Greatest Root       0.05866   3 128  2.65880 0.05106
#> Weighted Survey CC        0.24187 131 130  1.15077 0.25194
#> Complex Survey CC         0.24187 131  98  1.13501 0.25914
#>
#> $Stats.cancor$Stats.cancor.2
#>                         Statistic df1 df2 Chi-Sq/F   p-val
#> Wilks' Lambda             0.99471   2  NA  1.76100 0.41458
#> Pillai's Trace            0.00529   1  NA  1.76162 0.41445
#> Hotelling-Lawley Trace    0.00532   2  NA  1.76036 0.41471
#> Roy's Greatest Root       0.05866   2 129  4.01935 0.02026
#> Weighted Survey CC        0.07375 131 130  0.79444 0.42839
#> Complex Survey CC         0.07375 131  98  0.89128 0.37496
#>
#>
#> $cc_object
#>
```

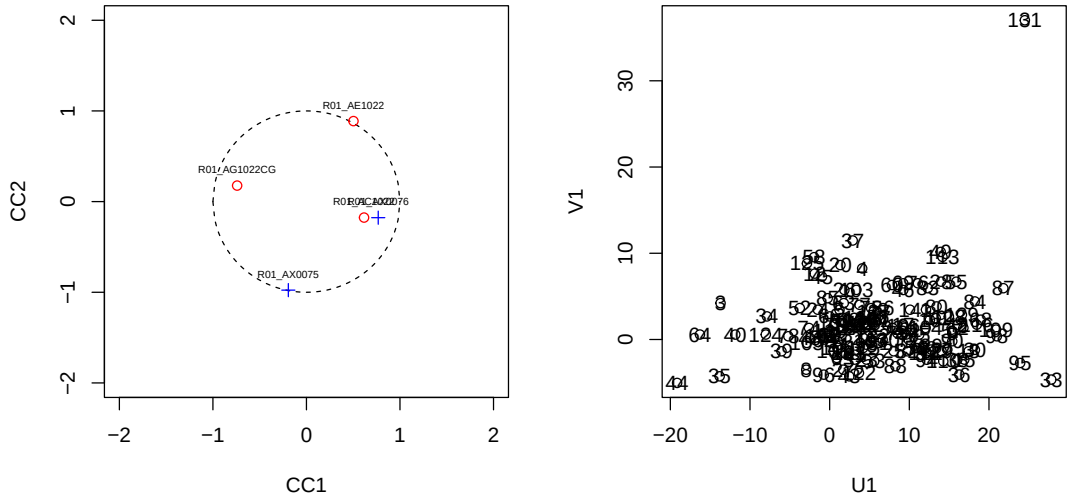



**Figure 1:** SurveyCC plot for the PATH example

```
#> Canonical correlation analysis of:
#>   3   X  variables:  R01_AC1022, R01_AE1022, R01_AG1022CG
#>   with    2   Y  variables:  R01_AX0075, R01_AX0076
#>
#>      CanR   CanRSQ    Eigen percent    cum                              scree
#> 1 0.24187 0.058499 0.062134  91.911  91.91 ******************************
#> 2 0.07375 0.005439 0.005469   8.089 100.00 ***
#>
#> Test of H0: The canonical correlations in the
#> current row and all that follow are zero
#>
#>       CanR LR test stat approx F numDF denDF Pr(> F)
#> 1 0.241867      0.93638   1.4145     6   254  0.2093
#> 2 0.073748      0.99456      NaN     2   NaN     NaN
#>
#> attr(,"class")
#> [1] "surveycc" "list"
```

### PATH Study Example Conclusion

Notice how the $p$-values for all tests are statistically insignificant, hence, the same conclusions are reached regardless of the method used. This might be especially true for national survey data sets since they have sample sizes so large that $p$-values for all tests might be considered statistically significant.

In this example the plotting method is demonstrated, using the following code:

```
par(mfrow = c(1,2))
dim1 <- 1
dim2 <- 2
plot(out.PATH, dim1 = dim1, dim2 = dim2)
```

The variables graph in Figure 1 shows that the number of cigarillo-smoking days in past 30 days (`R01_AG1022CG`) and the number of alcoholic drinks usually consumed each drinking day in past 30 days (`R01_AX0076`) lie on opposite sites of the unit circle and hence it should be expected that when one increases the other one should be expected to decrease. On the other hand, both variables are extremely close inside a unit circle, so this association is not strong and more work would be needed to explore it. Also, Figure 1 shows that, in this example, observation 131 is a unit that has an extraordinary value in the new two-dimensional plane described by the canonical variates, and may be an outlier.

## 5.4 2021 National Youth Tobacco Survey (NYTS) Example

This example also shows how yet another national survey with CSD elements can be handled by the `surveycc()` function further demonstrating its versatility. For illustrative purposes, the proposed

Survey CC method was applied to the 2021 NYTS data set to evaluate the relationships of multiple tobacco and/or e-cigarette product use with measures of exposure to e-cigarette Internet marketing and perceived addiction to various tobacco products relative to cigarettes among Asian Americans (n=1150). In this case **SurveyCC** handles a larger set of variables, providing evidence of its potential usefulness in a real-life research scenario. The definitions of the variables in this example is provided in a supplementary file. The detailed examination of the canonical structure produced by these commands is beyond the scope of this manuscript and more appropriate for a paper focused on tobacco and nicotine research.

### NYTS Example Setup

As previously mentioned, it is necessary to create the CSD object using the **survey** package, which is then passed into the main function of the current package, `surveycc()`.

In addition to the CSD object, here we define vectors for each set of variables, as well as an indicator for the desired number of canonical variates we want. These objects are all passed into the `surveycc()` function.

```
design_object <-
  survey::svydesign(
    ids = ~psu2,
    nest = TRUE,
    strata = ~v_stratum2,
    weights = ~finwgt,
    data = reducedNYTS2021data
  )
var.x <- c("qn9", "qn38", "qn40", "qn53", "qn54", "qn64", "qn69",
           "qn74","qn76", "qn78", "qn80", "qn82", "qn85", "qn88",
           "qn89")
var.y <- c("qn128", "qn129", "qn130", "qn131", "qn132", "qn134")
howmany <- 3
out.NYTS <- surveycc(design_object = design_object, var.x = var.x,
  var.y = var.y, howmany = howmany, selection = "ROWS")
```

### NYTS Example Results

Here `surveycc()` is used to estimate canonical correlations $p$-values for the NYTS example. Notice that the specification of the CSD elements in the function `survey::svrepdesign()` were adapted from the document *Guide to weighting harmonized NYTS data for the 1999 and 2000 survey years available at Integrated Public Use Microdata Series (IPUMS) NYTS* [https://www.ipums.org/projects/ipums-nyts].

See the output of the saved `out.NYTS` object below for the output table containing all statistics, degrees of freedom, test statistics, and $p$-values:

```
out.NYTS
```

```
#> $Stats.cancor
#> $Stats.cancor$Stats.cancor.1
#>                         Statistic  df1  df2  Chi-Sq/F   p-val
#> Wilks' Lambda             0.88398   90   NA 143.58266 0.00029
#> Pillai's Trace            0.12173   89   NA 142.94303 0.00032
#> Hotelling-Lawley Trace    0.12495   90   NA 144.22768 0.00025
#> Roy's Greatest Root       0.03663   15 1134   2.87474 0.00018
#> Weighted Survey CC        0.19387 1149 1148   1.56828 0.11709
#> Complex Survey CC         0.19387 1149   75   1.51094 0.13501
#>
#> $Stats.cancor$Stats.cancor.2
#>                         Statistic  df1  df2  Chi-Sq/F   p-val
#> Wilks' Lambda             0.91759   70   NA 102.88481 0.00641
#> Pillai's Trace            0.08509   69   NA 102.56555 0.00679
#> Hotelling-Lawley Trace    0.08692   70   NA 103.20553 0.00604
#> Roy's Greatest Root       0.03663   14 1135   3.08280 0.00010
#> Weighted Survey CC        0.17412 1149 1148   1.91871 0.05527
#> Complex Survey CC         0.17412 1149   75   1.86544 0.06603
```

```
#>
#> $Stats.cancor$Stats.cancor.3
#>                         Statistic  df1  df2 Chi-Sq/F   p-val
#> Wilks' Lambda             0.94624   52   NA 68.54747 0.06171
#> Pillai's Trace            0.05482   51   NA 68.44115 0.06277
#> Hotelling-Lawley Trace    0.05570   52   NA 68.65328 0.06066
#> Roy's Greatest Root       0.03663   13 1136  3.32286 0.00005
#> Weighted Survey CC        0.14872 1149 1148  2.02639 0.04296
#> Complex Survey CC         0.14872 1149   75  2.02187 0.04676
#>
#>
#> $cc_object
#>
#> Canonical correlation analysis of:
#>  15  X variables:  qn9, qn38, qn40, qn53, qn54, qn64, qn69, qn74, qn76, qn78, qn80, qn82, qn85, qn88, qn89
#>   with    6   Y  variables:  qn128, qn129, qn130, qn131, qn132, qn134
#>
#>      CanR   CanRSQ    Eigen percent    cum                              scree
#> 1 0.19387 0.037585 0.039053  30.904  30.90 ******************************
#> 2 0.17412 0.030317 0.031265  24.741  55.64 ************************
#> 3 0.14872 0.022118 0.022618  17.898  73.54 ******************
#> 4 0.11860 0.014066 0.014267  11.290  84.83 ***********
#> 5 0.10631 0.011301 0.011430   9.045  93.88 *********
#> 6 0.08762 0.007678 0.007737   6.123 100.00 ******
#>
#> Test of H0: The canonical correlations in the
#> current row and all that follow are zero
#>
#>       CanR LR test stat approx F numDF  denDF   Pr(> F)
#> 1 0.193869      0.88276  1.58342    90 6355.7 0.0003999 ***
#> 2 0.174117      0.91724  1.40840    70 5384.0 0.0144608 *
#> 3 0.148720      0.94591  1.21872    52 4382.5 0.1358675
#> 4 0.118602      0.96731  1.05132    36 3345.3 0.3859128
#> 5 0.106305      0.98111  0.98695    22 2266.0 0.4776264
#> 6 0.087625      0.99232  0.87743    10 1134.0 0.5539335
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> attr(,"class")
#> [1] "surveycc" "list"
```

**NYTS Example Conclusion**

The importance of our proposed Survey CC method is shown in all three tables produced by `surveycc`, which compare the $p$-values obtained using the classic tests available in the existing statistical software that consider only survey weights (weighted CCA) against those obtained using our newly developed method, Survey CC, which incorporates the remaining complex survey design elements. For the Asian-Americans in this data set, the Survey CC method results in different conclusions regarding the significance of the first canonical correlation and the relationships that can be inferred from the examination of canonical coefficients. For example, in the tables `Stats.cancor\$Stats.cancor.1` and `Stats.cancor\$Stats.cancor.2` the classic tests lead to $p$-values smaller then 0.05, e.g., Wilks' Lambda has $p$-value=0.00029, Pillai's Trace has $p$-value=0.00032, Hotelling-Lawley Trace has $p$-value=0.00025 and Roy's Greatest Root has $p$-value=0.00018 for the first canonical correlation, while the Survey CC method does the opposite ($p$-value=0.11709 taking into consideration only the survey weights and $p$-value=0.13501 considering all CSD factors).

## 6 Conclusions

The inclusion of all complex survey design elements, i.e., survey weights, cluster, strata, etc., is essential for the accurate calculation of any statistical significance, including that of canonical correlations. The incorporation of design elements is important as CCA is a multivariate method that it is especially suitable to study problems that involve sets of variables, such as those presented as examples in

this manuscript. Our proposed function `surveycc()` allows the user not only to address this issue but to test the statistical significance of each canonical correlation separately instead of a canonical correlation and all those that come after it. Consideration of these results with those from classic statistical tests allows the user of `surveycc()` to obtain a complete and accurate assessment of the statistical significance of each of the canonical correlations and hence of the conclusions obtained from the examination of the canonical structure.

Moreover, an example with simple random sample data, i.e., devoid of complex survey design elements, showed that `surveycc()` can be useful and provide additional information compared to the existing CCA R, Stata and SAS commands, even if the user is not examining national survey data, as it provides necessary information for assessing the statistical significance of secondary canonical correlations using different methods. As seen in the examples included in this manuscript, the examination of the secondary canonical coefficients leads to insights about the relationships that exist among the variables included in the problem not provided by the first canonical coefficients. Therefore, the accurate evaluation of the statistical significance of the corresponding secondary canonical correlations is essential to determine the validity of the conclusions drawn from these analyses. Also, the ability to provide the variables and units graphs makes **SurveyCC** a complement to the existing CCA packages in R.

The results of our examples show the limited usefulness of the upper limit of the Largest Root statistic as proposed in (Johnstone, 2009), which is a limitation of the current implementation of our package. For the secondary canonical correlations, it consistently provided $p$-values significantly smaller than those of all the other methods, making its interpretation exceedingly difficult. Another limitation of our package is that, in all national survey examples, the option `FREQ` led to sample sizes so large that all methods provided a $p$-value of <0.0001 for all canonical correlations. These trivially significant results are due to the same issues present in (Nelson et al., 2020). Although national survey data sets include a large number of subjects, when the option `FREQ` is omitted, the sample sizes are not always large enough to make all statistical tests lead to a small $p$-values, especially when examining sub-populations of interest, as shown in the last example presented in Section 5 of this paper. The user of our command should keep these issues in mind and refrain from declaring a canonical correlation statistically significant based solely on these criteria. Also, this paper focuses on introducing the Survey CC algorithm and the SurveyCC package, but it does not cover any preprocessing steps that may be required for proper analysis of specific datasets, such as adjustments to survey weights when analyzing a population subdomain. Finally, to our knowledge our method Survey CC is the first to incorporate CSD factors into the calculation of the statistical significance of all canonical correlations using the exact adjustments provided in the methodological documentation of the surveys being studied, but there might be other ways to combine or modify existing statistical procedures to reach this same objective.

We presented several examples coming from a diverse set of national surveys that show the versatility of the command `surveycc()` and its ability to work appropriately for a diverse group of national data sets. Given the wealth of information contained in these surveys, an examination of the survey data that appropriately accounts for their complex design is essential. Our package can become an important tool in this process through the accurate examination of the canonical correlations statistical significance. In addition to the national surveys listed in this manuscript, we expect our package to be able to handle data sets such as the Behavioral Risk Factor Surveillance System, the Medical Expenditures Panel Survey, the Health Information National Trends Survey, the Tobacco Use Supplement to the Current Population Survey, and other national and international surveys produced using complex survey design.

In conclusion, our package **SurveyCC** can help to extract accurate information about the relationships between sets of variables and the statistical significance of those relationships in data sets created using complex survey design, hence having the potential to help researchers interested in using this type of data sets to address scientific questions.

## References

B. Amies-Cull, J. Wolstenholme, L. Cobiac, and P. Scarborough. Estimating bmi distributions by age and sex for local authorities in england: a small area estimation study. *BMJ Open*, 2022. URL https://doi.org/10.1136/bmjopen-2022-060892. [p150]

P. Boedeker and R. Henson. *Canonical Correlation Analysis*. SAGE Publications Ltd, 2020. URL https://doi.org/10.4135/9781526421036883301. [p142]

R. Borland, N. Owen, D. Hill, and P. Schofield. Predicting attempts and sustained cessation of smoking after the introduction of workplace smoking bans. *Health Psychology*, 1991. URL https://doi.org/10.1037//0278-6133.10.5.336. [p141]

T. Caliński, M. Krzyśko, and W. WOłyński. A comparison of some tests for determining the number of nonzero canonical correlations. *Communications in Statistics - Simulation and Computation*, 2006. URL https://doi.org/10.1080/03610910600716290. [p142, 144]

D. Clark. Understanding canonical correlation analysis. *Norwich: Geo Abstracts Ltd*, 1975. URL https://www.qmrg.org.uk/catmog/index.html. [p141]

N. Correa, Y. Li, T. Adali, and V. Calhoun. Canonical correlation analysis for feature-based fusion of biomedical imaging modalities and its application to detection of associative networks in schizophrenia. *IEEE Journal of Selected Topics in Signal Processing*, 2008. URL https://doi.org/10.1109/JSTSP.2008.2008265. [p141]

R. Cruz-Cano and M.-L. Lee. Fast regularized canonical correlation analysis. *Computational Statistics and Data Analysis*, 2014. URL https://doi.org/10.1016/j.csda.2013.09.020. [p145]

J. Garcia Villar and C. Quintana-Domeque. Income and body mass index in europe. *Economics and Human Biology*, 2009. URL https://doi.org/10.1016/j.ehb.2009.01.006. [p150]

R. Gittins. Canonical analysis: a review with applications in ecology. 2012. URL https://link.springer.com/book/10.1007/978-3-642-69878-1. [p141, 144, 145]

D. Hahs-Vaughn, C. McWayne, R. Bulotsky-Shearer, X. Wen, and A. Faria. Methodological considerations in using complex survey data: an applied example with the head start family and child experiences survey. *Evaluation Review*, 2011. doi: https://doi.org/10.1177/0193841X11412071. [p142]

J. F. Hair, W. C. Black, B. J. Babin, and R. E. Anderson. *Multivariate data analysis*. Cengage Learning EMEA, 2019. URL https://www.cengage.com/c/multivariate-data-analysis-8e-hair-babin-anderson-black/9781473756540/. [p141]

D. Hardoon and J. Shawe-Taylor. Sparse canonical correlation analysis. *Machine Learning*, 2011. URL https://doi.org/10.1007/s10994-010-5222-7. [p141]

T. Hill and P. Lewicki. *Electronic Statistics Textbook*. StatSoft Inc., 1997. URL https://statsoft.com. [p142]

D. Holt, T. Smith, and P. Winter. Regression analysis of data from complex surveys. *Journal of the Royal Statistical Society: Series A (General)*, 143(4):474–487, 1980. URL https://doi.org/10.2307/2982065. [p142]

H. Hotelling. Relations between two sets of variates. *Biometrika*, 1936. URL https://doi.org/10.1093/biomet/28.3-4.321. [p141]

B. Jarrett, G. Bloch, D. Bennett, B. Bleazard, and D. Hedges. The influence of body mass index, age and gender on current illness: a cross-sectional study. *International Journal of Obesity*, 2009. URL https://doi.org/10.1038/ijo.2009.258. [p150]

I. M. Johnstone. Approximate null distribution of the largest root in multivariate analysis. *The annals of applied statistics*, 3(4):1616, 2009. URL https://doi.org/10.1214/08-AOAS220. [p142, 144, 155]

S. Killip, Z. Mahfoud, and K. Pearce. What is an intracluster correlation coefficient? crucial concepts for primary care researchers. *The Annals of Family Medicine*, 2(3):204–208, 2004. URL https://doi.org/10.1370/afm.141. [p142]

J. D. Leo and E. Wulfert. Problematic internet use and other risky behaviors in college students: An application of problem-behavior theory. *Psychol Addict Behav.*, 2013. URL https://doi.org/10.1037/a0030823. [p141]

T. H. Lewis. *Complex survey data analysis with SAS*. Chapman and Hall/CRC, 2016. URL https://doi.org/10.1201/9781315366906. [p142]

T. Lumley. Analysis of complex survey samples. *Journal of statistical software*, 9:1–19, 2004. URL https://doi.org/10.18637/jss.v009.i08. [p142]

W. Meredith. Canonical correlations with fallible data. *Psychometrika*, 1964. URL https://doi.org/10.1007/BF02289567. [p142]

D. Morris, A. Davis, K. Lauritsen, C. Rieth, M. Silvestri, J. Winters, and S. CHermack. Substance use consequences, mental health problems, and readiness to change among veterans seeking substance use treatment. *Journal of Substance Abuse Treatment*, 2018. URL https://doi.org/10.1016/j.jsat.2018.08.005. [p141]

D. R. Nelson, S. H. Wong-Jacobson, E. Lilly, and Company. %surveycorrcov macro: Complex survey data correlations for multivariate analysis and model building. 2020. URL https://api.semanticscholar.org/CorpusID:215747405. [p143, 155]

SAS. *Survey Weights: A Step-by-Step Guide to Calculation*. SAS Institute Inc., 2012. URL https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.4/statug/statug_cancorr_toc.htm. [p142]

StataCorp. *Stata Statistical Software: Release 18*. Stata Press, 2023a. URL https://www.stata-press.com/manuals/documentation-set. [p146]

StataCorp. *Stata 18 Multivariate Statistics Reference Manual*. Stata Press, 2023b. URL https://www.stata.com/manuals/mvcanon.pdf. [p142]

J. Stowe, C. Watson, and T. Robertson. Relationships between the two sides of the balance sheet: A canonical correlation analysis. *Journal of Finance*, 1980. URL https://doi.org/10.2307/2327214. [p142]

J.-Y. Sun, W. Huang, Y. Hua, Q. Qu, C. Cheng, H. Liu, X. Kong, Y. Ma, and W. Sun. Trends in general and abdominal obesity in us adults: Evidence from the national health and nutrition examination survey (2001–2018). *Frontiers in Public Health*, 2022. URL https://doi.org/10.3389/fpubh.2022.925293. [p150]

F. Travis and Y. Lagrosen. Creativity and brain-functioning in product development engineers: A canonical correlation analysis. *Creativity Research Journal*, 2014. URL https://doi.org/10.1080/10400419.2014.901096. [p142]

R. Valliant and J. Dever. *Survey Weights: A Step-by-Step Guide to Calculation*. Stata Press, 2018. URL https://www.stata-press.com/books/survey-weights. [p144]

C. Winship and L. Radbill. Sampling weights and regression analysis. *Sociological Methods & Research*, 23(2):230–257, 1994. URL https://doi.org/10.1177/0049124194023002004. [p142]

Y. Yang, C. Walsh, M. Johnson, D. Belsky, M. Reason, P. Curran, A. Aiello, M. Chanti-Ketterl, and K. Harris. Life-course trajectories of body mass index from adolescence to old age: Racial and educational disparities. *Proc Natl Acad Sci*, 2021. doi: 10.1073/pnas.2020167118. [p150]

X. Zhuang and Z. Yang. A technical review of canonical correlation analysis for neuroscience applications. *Hum Brain Mapp*, 2020. URL https://doi.org/10.1002/hbm.25090. [p141]

*Raul Cruz-Cano*
*Department of Epidemiology and Biostatistics, IU School of Public Health*
*Indiana University School of Public Health*
*Bloomington, IN*
*USA*
raulcruz@iu.edu

*Aaron Cohen*
*Department of Epidemiology and Biostatistics, IU School of Public Health*
*Biostatistics Consulting Center*
*Department of Epidemiology and Biostatistics*
*Indiana University School of Public Health*
*Bloomington, IN*
*USA*
cohenaa@iu.edu

*Erin Mead-Morse*
*Department of Public Health Sciences, UConn Health, School of Medicine*
*UConn Health, School of Medicine*
*Farmington, CT*
*USA*
mead@uchc.edu

# GPUmatrix: Seamlessly harness the power of GPU computing in R

*by César Lobato-Fernández, Juan A. Ferrer-Bonsoms, and Angel Rubio*

**Abstract** GPUs are invaluable for data analysis, particularly in statistics and linear algebra, but integrating them with R has been challenging due to the lack of transparent, easily maintainable packages that don't require significant code alterations. Recognizing this gap, we've developed the GPUmatrix package, now available on CRAN, which emulates the Matrix package's behavior, enabling R to harness the power of GPUs for computations with minimal code adjustments. GPUmatrix supports both single (FP32) and double (FP64) precision data types and includes support for sparse matrices, ensuring broad applicability. Designed for ease of use, it requires only slight modifications to existing code, leveraging the Torch or Tensorflow R packages for GPU operations. We've validated its effectiveness in various statistical and machine learning tasks, including non-negative matrix factorization, logistic regression, and general linear models, and provided a comparative analysis of GPU versus CPU performance, highlighting significant efficiency gains.

## 1 Introduction

R(R Core Team, 2022) is a programming language broadly used in statistics and computational biology. R is part of a collaborative and open project where users can publish packages that extend its base configuration. CRAN (Comprehensive R Archive Network) and Bioconductor are repositories where R packages are stored after passing strict code quality criteria.

Graphics Processing Units (GPUs) perform computations in parallel, making them exceptionally powerful for tasks involving large amounts of data processing. At the hardware level, GPUs consist of many processing cores that can execute thousands of threads simultaneously. These threads are organised into groups called warps (in NVIDIA GPUs) or wavefronts (in AMD GPUs), which typically contain 32 or 64 threads. All threads within a warp operate in lockstep, following the Single Instruction, Multiple Thread (SIMT) model. This means that each thread executes the same instruction at the same time, but on different data elements, enabling massive parallelism and high throughput for suitable tasks.

While the SIMT architecture excels at parallel execution, it presents challenges when threads within a warp need to follow different execution paths due to conditional statements or branching. Matrix operations are particularly well suited to GPUs due to their regular computational patterns and minimal branching. Operations such as matrix multiplication, element-wise arithmetic and linear algebra computations involve applying the same operation to large data sets, which fits perfectly with the SIMT execution model. This makes matrix computations highly efficient on GPUs, leading to significant performance gains in applications such as machine learning, scientific simulation and data analysis.

Statistical and machine learning methods are mostly based on linear algebra operations. Using a GPU for these operations greatly increases the computational power. Since R is a language designed for these types of methods, adapting it to use the GPU would result in a significant computational improvement.

There have been previous attempts to exploit the power of the GPU using R packages. Some of these have been method-specific - such as cuRnet (Bonnici et al., 2018) for graph analysis, qrpca (de Souza et al., 2022) for PCA analysis, rgtsvm (Wang et al., 2017) for support vector machines, **xgboost** (Chen and Guestrin, 2016) for extreme grading boosting, and **gputools** (Buckner et al., 2009) for microarray analysis. **gpuR** (Determan, 2017), on the other hand, provided "GPU-enabled functions in a simple and accessible way". gpuMagic (Wang and Morgan, 2022), allows users to compile R functions in OpenCL and run the code on the GPU. None of these are currently available on CRAN or Bioconductor except gpuMagic and xgboost.

**torch** (Falbel and Luraschi, 2023) and **tensorflow** (Allaire and Tang, 2022) are other GPU-enabled packages for R that target machine learning methods whose workhorse are tensor-like objects, a generalization of the concepts of scalar, vector, and matrix. These packages provide the functionality of the Python packages PyTorch and Tensorflow respectively. Both are available on CRAN and are GPU enabled. Both are maintained by developers from RStudio (RStudio Team, 2020) (now Posit). Both packages represent a rather complex learning process for the standard R user as they focus on machine learning rather than linear algebra and the syntax is reminiscent of its ''Pythonic" origins.

Here we present **GPUmatrix** which, by including methods for most statistical and linear algebra

functions, allows the full functionality of R to be used for GPU computation with almost no learning for the user, making it very easy to adapt to existing programs. GPUmatrix uses either Torch or Tensorflow internally to perform the computations.

The paper is organized as follows: in the first section we give a general description of the package, its objects and features. In the second section we show several statistical applications of the package and their respective performance improvements. The third section contains the results of the comparison of the speed of the operations and functions using either CPUs or GPUs. In the last section we discuss these results.

## 2 Overall Description

GPUmatrix is based on the S4 object structure provided by R, and mimics the behavior of the **Matrix** package (Bates et al., 2023). It implements sparse matrices and **float** (Schmidt, 2017) data types (from Matrix and float packages) as part of the gpu.matrix objects. It includes casting operations i.e conversions from one object type to a different one between gpu.matrix objects, Matrix -including sparse-, float32, and the default matrix objects in R.

For a list of all available methods, see Tables T2, T3 and T4 in the package vignette. These include the standard operations on matrices (sum, difference, inverse, standard matrix product, Hadamard product, etc.), access to matrix elements (entry, rows, columns, submatrices, etc.), matrix factorizations (qr, svd, chol, lu, etc.), algorithms for sorting, computing the fft of a vector, etc. GPUmatrix also includes operations from the **matrixStats** package (Bengtsson, 2022) (rowMedians, rowMaxs, etc.)

Installing GPUmatrix itself is straightforward. It depends on the installation of either torch or tensorflow, which is more involved. In turn, if GPU processing is required, torch and tensorflow depend on specific versions of CUDA (not necessarily identical). In the GPUmatrix vignette we have included a brief explanation on how to install these packages.

GPUmatrix requires either torch or tensorflow to be installed. GPUmatrix can be used if any of them is installed. For this reason, neither of them are strictly required(if the other is available) an we have included no dependencies between GPUmatrix and torch or tensorflow.

### 2.1 Examples of Use

The GPUmatrix package is based on S4 objects in R and we have added a constructor function that similar to the default `matrix()` constructor in R for CPU matrices. The constructor function is `gpu.matrix()` and takes the same parameters as `matrix()`:

```
library(GPUmatrix)

#> Torch tensors allowed

#> Your Torch installation does not have CUDA tensors available. Please check the Torch requirements and installat

#R matrix initialization
m <- matrix(c(1:10)+40,5,2)
#Show CPU matrix
m

#>      [,1] [,2]
#> [1,]   41   46
#> [2,]   42   47
#> [3,]   43   48
#> [4,]   44   49
#> [5,]   45   50

#GPU matrix initialization
Gm <- gpu.matrix(c(1:10)+40,5,2)
#Show GPU matrix
Gm

#> GPUmatrix
#> torch_tensor
```

```
#>  41  46
#>  42  47
#>  43  48
#>  44  49
#>  45  50
#> [ CPUDoubleType{5,2} ]
```

Although the indexing of tensors in both torch and tensorflow is 0-based, the indexing of GPUmatrix objects is 1-based, making it as close as possible to working with native R matrices and more convenient for the user. In the previous example, a normal R CPU matrix called `m` and its GPU counterpart `Gm` are created. Just like regular matrices, the created GPU matrices allow for indexing their elements and assigning values to them. The concatenation operators `rbind()` and `cbind()` work independently of the type of matrices being concatenated, resulting in a `gpu.matrix()`:

```
Gm[c(2,3),1]
```

```
#> GPUmatrix
#> torch_tensor
#>  42
#>  43
#> [ CPUDoubleType{2,1} ]
```

```
Gm[,2]
```

```
#> GPUmatrix
#> torch_tensor
#>  46
#>  47
#>  48
#>  49
#>  50
#> [ CPUDoubleType{5,1} ]
```

```
Gm2 <- cbind(Gm[c(1,2),], Gm[c(3,4),])
Gm2
```

```
#> GPUmatrix
#> torch_tensor
#>  41  46  43  48
#>  42  47  44  49
#> [ CPUDoubleType{2,4} ]
```

```
Gm2[1,3] <- 0
Gm2
```

```
#> GPUmatrix
#> torch_tensor
#>  41  46   0  48
#>  42  47  44  49
#> [ CPUDoubleType{2,4} ]
```

It is also possible to initialize the data with NaN values:

```
Gm3 <- gpu.matrix(nrow = 2,ncol=3)
Gm3[,2]
```

```
#> GPUmatrix
#> torch_tensor
#> nan
#> nan
#> [ CPUDoubleType{2,1} ]
```

```
Gm3[1,2] <- 1
Gm3
```

**Table 1:** Cast options from other packages. If back cast is TRUE, then it is possible to convert a gpu.matrix to this object and vice versa. If is FALSE, it is possible to convert these objects to gpu.matrix but not vice versa.

| MatrixClass | Package | DataTypeDefault | SPARSE | BackCast |
|---|---|---|---|---|
| matrix | base | float64 | FALSE | TRUE |
| data.frame | base | float64 | FALSE | TRUE |
| integer | base | float64 | FALSE | TRUE |
| numeric | base | float64 | FALSE | TRUE |
| dgeMatrix | Matrix | float64 | FALSE | FALSE |
| ddiMatrix | Matrix | float64 | TRUE | FALSE |
| dpoMatrix | Matrix | float64 | FALSE | FALSE |
| dgCMatrix | Matrix | float64 | TRUE | FALSE |
| float32 | float | float32 | FALSE | FALSE |
| torch_tensor | torch | float64 | Depends of tensor type | TRUE |
| tensorflow.tensor | tensorflow | float64 | Depends of tensor type | TRUE |

```
#> GPUmatrix
#> torch_tensor
#> nan  1 nan
#> nan nan nan
#> [ CPUDoubleType{2,3} ]

Gm3[1,3] <- 0
Gm3

#> GPUmatrix
#> torch_tensor
#> nan  1  0
#> nan nan nan
#> [ CPUDoubleType{2,3} ]
```

These examples demonstrate that, contrary to standard R, subsetting a gpu.matrix —even when selecting only one column or row— still results in a gpu.matrix. This behavior is analogous to using `drop=FALSE` in standard R. The default standard matrices in R have limitations. The only allowed numeric data types are int and float64. It neither natively allows the creation or handling of sparse matrices. To make up for this lack of functionality, other R packages hosted in CRAN have been created to manage these types.

### 2.2 Cast from other packages

GPUmatrix allows for compatibility with sparse matrices and different data types such as float32. For this reason, casting operations between different matrix types from multiple packages to GPUmatrix type have been implemented (Table 1).

There are two functions for casting to create a `gpu.matrix`: `as.gpu.matrix()**` and the `gpu.matrix()` constructor itself. Both have the same input parameters for casting: the object to be cast and extra parameters to create a GPUmatrix.

Create 'Gm' from 'm' matrix R-base:

```
m <- matrix(c(1:10)+40,5,2)
Gm <- gpu.matrix(m)
Gm

#> GPUmatrix
#> torch_tensor
#>  41  46
#>  42  47
#>  43  48
#>  44  49
```

```
#>  45  50
#> [ CPUDoubleType{5,2} ]
```

Create 'Gm' from 'M' with Matrix package:

```
library(Matrix)
M <- Matrix(c(1:10)+40,5,2)
Gm <- gpu.matrix(M)
Gm
```

```
#> GPUmatrix
#> torch_tensor
#>  41  46
#>  42  47
#>  43  48
#>  44  49
#>  45  50
#> [ CPUDoubleType{5,2} ]
```

Create 'Gm' from 'mfloat32' with float package:

```
library(float)
mfloat32 <- fl(m)
Gm <- gpu.matrix(mfloat32)
Gm
```

```
#> GPUmatrix
#> torch_tensor
#>  41  46
#>  42  47
#>  43  48
#>  44  49
#>  45  50
#> [ CPUFloatType{5,2} ]
```

Interestingly, GPUmatrix returns a float32 data type matrix if the input is a float matrix. It is also possible to a gpu.matrix create 'Gms' type sparse from 'Ms' type sparse dgCMatrix, dgeMatrix, ddiMatrix or dpoMatrix with Matrix package:

```
Ms <- Matrix(sample(0:1, 10, replace = TRUE), nrow=5, ncol=2, sparse=TRUE)
Ms
```

```
#> 5 x 2 sparse Matrix of class "dgCMatrix"
#>
#> [1,] . .
#> [2,] 1 .
#> [3,] . 1
#> [4,] . .
#> [5,] . 1
```

```
Gms <- gpu.matrix(Ms)
Gms
```

```
#> GPUmatrix
#> torch_tensor
#> [ SparseCPUFloatType{}
#> indices:
#>  1  2  4
#>  0  1  1
#> [ CPULongType{2,3} ]
#> values:
#>  1
#>  1
```

```
#>  1
#> [ CPUFloatType{3} ]
#> size:
#> [5, 2]
#> ]
```

### 2.3 Data type and sparsity

The data types allowed for GPUmatrix are: **float64**, **float32**, **int**, **bool** or **logical**, **complex64** and exclusively in torch **complex32**. We can create a GPU matrix with a specific data type using the `dtype**` parameter of the `gpu.matrix()**` constructor function. It is also possible change the data type of a previously created GPU matrix using the `dtype()**` function. The same applies to GPU sparse matrices, we can create them from the constructor using the `sparse**` parameter, which will return Boolean value of `TRUE/FALSE` depending on whether we want the resulting matrix to be sparse or not. We can also modify the sparsity of an existing GPU matrix with the functions `to\_dense()**`, if we want it to change it from sparse to dense, and `to\_sparse()**`, if we want it to go from dense to sparse.

Creating a float32 matrix:

```
Gm32 <- gpu.matrix(c(1:10)+40,5,2, dtype = "float32")
Gm32
```

```
#> GPUmatrix
#> torch_tensor
#>  41  46
#>  42  47
#>  43  48
#>  44  49
#>  45  50
#> [ CPUFloatType{5,2} ]
```

Creating a non sparse matrix with data type float32 from a sparse matrix type float64:

```
Ms <- Matrix(sample(0:1, 10, replace = TRUE), nrow=5, ncol=2, sparse=TRUE)
Gm32 <- gpu.matrix(Ms, dtype = "float32", sparse = F)
Gm32
```

```
#> GPUmatrix
#> torch_tensor
#>  1  1
#>  1  0
#>  1  0
#>  1  1
#>  0  1
#> [ CPUFloatType{5,2} ]
```

Convert Gm32 in sparse matrix Gms32:

```
Gms32 <- to_sparse(Gm32)
Gms32
```

```
#> GPUmatrix
#> torch_tensor
#> [ SparseCPUFloatType{}
#> indices:
#>  0  1  2  3  0  3  4
#>  0  0  0  0  1  1  1
#> [ CPULongType{2,7} ]
#> values:
#>  1
#>  1
#>  1
```

```
#>  1
#>  1
#>  1
#>  1
#> [ CPUFloatType{7} ]
#> size:
#> [5, 2]
#> ]
```

Convert data type Gms32 into float64:

```
Gms64 <- Gms32
dtype(Gms64) <- "float64"
Gms64
```

```
#> GPUmatrix
#> torch_tensor
#> [ SparseCPUDoubleType{}
#> indices:
#>  0  1  2  3  0  3  4
#>  0  0  0  0  1  1  1
#> [ CPULongType{2,7} ]
#> values:
#>  1
#>  1
#>  1
#>  1
#>  1
#>  1
#>  1
#> [ CPUDoubleType{7} ]
#> size:
#> [5, 2]
#> ]
```

### 2.4 GPUmatrix functions

#### Arithmetic and comparison operators

GPUmatrix supports all of the basic arithmetic operators in R: `+`, `-`, `*`, `\^{}`, `/`, `\%*\%` and `\%\%`. Its usage is the same as for basic R matrices, and it allows compatibility with other matrix objects from the packages mentioned above.

```
(Gm + Gm) == (m + m)
```

```
#>      [,1] [,2]
#> [1,] TRUE TRUE
#> [2,] TRUE TRUE
#> [3,] TRUE TRUE
#> [4,] TRUE TRUE
#> [5,] TRUE TRUE
```

```
(Gm + M) == (mfloat32 + Gm)
```

```
#>      [,1] [,2]
#> [1,] TRUE TRUE
#> [2,] TRUE TRUE
#> [3,] TRUE TRUE
#> [4,] TRUE TRUE
#> [5,] TRUE TRUE
```

```
(M + M) == (mfloat32 + Gm)
```

**Table 2:** Mathematical operators that accept a gpu.matrix as input.

| MathematicalOperators | Usage |
|---|---|
| log | log(Gm) |
| log2 | log2(Gm) |
| log10 | log10(Gm) |
| cos | cos(Gm) |
| cosh | cosh(Gm) |
| acos | acos(Gm) |
| acosh | acosh(Gm) |
| sin | sin(Gm) |
| sinh | sinh(Gm) |
| asin | asin(Gm) |
| asinh | asinh(Gm) |
| tan | tan(Gm) |
| atan | atan(Gm) |
| tanh | tanh(Gm) |
| atanh | atanh(Gm) |
| sqrt | sqrt(Gm) |
| abs | abs(Gm) |
| sign | sign(Gm) |
| ceiling | ceiling(Gm) |
| floor | floor(Gm) |
| cumsum | cumsum(Gm) |
| cumprod | cumprod(Gm) |
| exp | exp(Gm) |
| expm1 | expm1(Gm) |

```
#>      [,1] [,2]
#> [1,] TRUE TRUE
#> [2,] TRUE TRUE
#> [3,] TRUE TRUE
#> [4,] TRUE TRUE
#> [5,] TRUE TRUE
```

As seen in the previous examples, the comparison operators (`==`, `!=`, `\textgreater{}`, `\textless{}`, `\textgreater{=}`, `\textless{=}`) also work following the same dynamic of the arithmetic operators.

### Math operators

Similarly to arithmetic operators, mathematical operators follow the same operation they would perform on regular matrices of R. `Gm` is a `gpu.matrix` variable (Table 2).

### Complex operators

There are certain functions only applicable to numbers of complex type. In R these functions are grouped as complex operators and all of them are available for GPUmatrix matrices with the same functionality as in R base. `Gm` is a `gpu.matrix` variable (Table 3).

### Other functions

In the manual, we can find a number of functions that can be applied to `gpu.matrix` type matrices. Most of these functions are from base R and can be applied to `gpu.matrix` matrices in the same way they would be applied to regular R matrices. There are other functions from other packages like **Matrix** or **matrixStats**, which have been implemented due to their widespread use within the R community, such as `rowVars` or `colMaxs`. The output of these functions, which originally produced R default

**Table 3:** Complex operators that accept a gpu.matrix with complex type as input.

| MathematicalOperators | Usage |
|---|---|
| Re | Re(Gm) |
| Im | Im(Gm) |
| Conj | Conj(Gm) |
| Arg | Arg(Gm) |
| Mod | Mod(Gm) |

matrix type objects, will now return `gpu.matrix` type matrices when the input type of the function is `gpu.matrix`.

```
library(GPUmatrix)
m <- matrix(c(1:10)+40,5,2)
Gm <- gpu.matrix(c(1:10)+40,5,2)

head(tcrossprod(m),2)

#>      [,1] [,2] [,3] [,4] [,5]
#> [1,] 3797 3884 3971 4058 4145
#> [2,] 3884 3973 4062 4151 4240

head(tcrossprod(Gm),2)

#> GPUmatrix
#> torch_tensor
#>  3797  3884  3971  4058  4145
#>  3884  3973  4062  4151  4240
#> [ CPUDoubleType{2,5} ]

Gm <- tail(Gm,3)
rownames(Gm) <- c("a","b","c")
tail(Gm,2)

#> GPUmatrix
#> torch_tensor
#>  44  49
#>  45  50
#> [ CPUDoubleType{2,2} ]
#> rownames: b c

colMaxs(Gm)

#> [1] 45 50
```

There is a wide variety of functions implemented in GPUmatrix, and they are adapted to be used just like regular R matrices.

### 2.5 Using GPUMatrix on CPU

In the GPUmatrix constructor, when using torch, we can specify the location of the matrix, i.e., we can decide to host it on the GPU or in RAM memory to use it with the CPU. As a package, oriented towards algebraic operations in R using the GPU, it is hosted on the GPU by default, but it allows the same functionalities to be used with the CPU. To do this, we use the `device` attribute of the constructor and assign it the value `"cpu"`.

```
#GPUmatrix initialization with CPU option
Gm <- gpu.matrix(c(1:10)+40,5,2,device="cpu")
#Show CPU matrix from GPUmatrix
Gm
```

```
#> GPUmatrix
#> torch_tensor
#>  41  46
#>  42  47
#>  43  48
#>  44  49
#>  45  50
#> [ CPUDoubleType{5,2} ]
```

Notice that instead of CUDADoubleType, now the object is CPUDoubleType.

The standard distribution of R includes a version of basic linear algebra subprograms (BLAS) that is not multithreaded and not fully optimized for present computers. R can be modified to use non-standard BLAS libraries such as OpenBlas, Intel MKL or Accelerate. Switching from Standard R to R using MKL implies changing the default behavior of R and there can be side-effects. For example, some standard packages such as igraph do not work in this case.

The standard BLAS is so slow, that we excluded it from the comparison. We have compared the CUDA-GPU (using GPUmatrix) with base R using Intel MKL (MKL-R).

Torch also runs on the CPU and its backend is MKL. Therefore, the performance between using MKL-R or using the GPUMatrix library on the CPU should be similar. The only differences would be related to the overhead from translating the objects or the different versions of the MKL library.

Interestingly, the standard R matrix operations are indeed slightly slower than using the GPUMatrix package -perhaps owing to a more recent version of the MKL library- (Fig 2), especially in element-wise operations, where MKL-R does not seem to exploit the multithreaded implementation of the Intel MKL BLAS version and Torch does.

In addition, MKL-R does not provide acceleration for float32 -since base R does not include this type of variable-. The multiplication of float32 matrices on MKL-R is, in fact, much slower than multiplying float64 matrices (data not shown). Torch and Tensorflow do include MKL for float32 and there is an improvement in the performance (they are around about twice faster that the float64 counterparts).

## 3 Examples of Statistical Applications

The main advantage of GPUmatrix is its versatility: R code needs only minor changes to adapt it to work in the CPU. We are showing here three statistical applications where its advantages are more apparent.

### 3.1 Non negative factorization of a matrix

The non-negative factorization (NMF) of a matrix is an approximate factorization were an initial matrix $\mathbf{V}$ is approximated by the product of two matrices $\mathbf{W}$ and $\mathbf{H}$ so that,

$$\mathbf{V}_{m\times n} \approx \mathbf{W}_{m\times k}\mathbf{H}_{k\times n}$$

We have implemented our own non-negative matrix factorization (NMF) function using Lee and Seung(Lee and Seung, 1999) multiplicative update rules.

These rules are

$$\mathbf{W}^{n+1}_{[i,j]} \leftarrow \mathbf{W}^{n}_{[i,j]} \frac{\left(\mathbf{V}\left(\mathbf{H}^{n+1}\right)^T\right)_{[i,j]}}{\left(\mathbf{W}^n\mathbf{H}^{n+1}\left(\mathbf{H}^{n+1}\right)^T\right)_{[i,j]}}$$

and

$$\mathbf{H}^{n+1}_{[i,j]} \leftarrow \mathbf{H}^{n}_{[i,j]} \frac{\left(\left(\mathbf{W}^n\right)^T\mathbf{V}\right)_{[i,j]}}{\left(\left(\mathbf{W}^n\right)^T\mathbf{W}^n\mathbf{H}^n\right)_{[i,j]}}$$

to update the $\mathbf{W}$ and $\mathbf{H}$ respectively.

The implemented function is `NMFgpumatrix`. This function operates in the same way with basic R matrices as with GPUmatrix matrices, and it does not require any additional changes beyond initializing the input matrix as a GPUmatrix. Indeed, the input matrices $\mathbf{W}$, $\mathbf{H}$, and $\mathbf{V}$ can be either gpu.matrix or R base matrices interchangeably. Figure 1 shows that using GPUmatrix boosts the performance using both GPU and CPU. This improvement is especially apparent with float32 matrices.

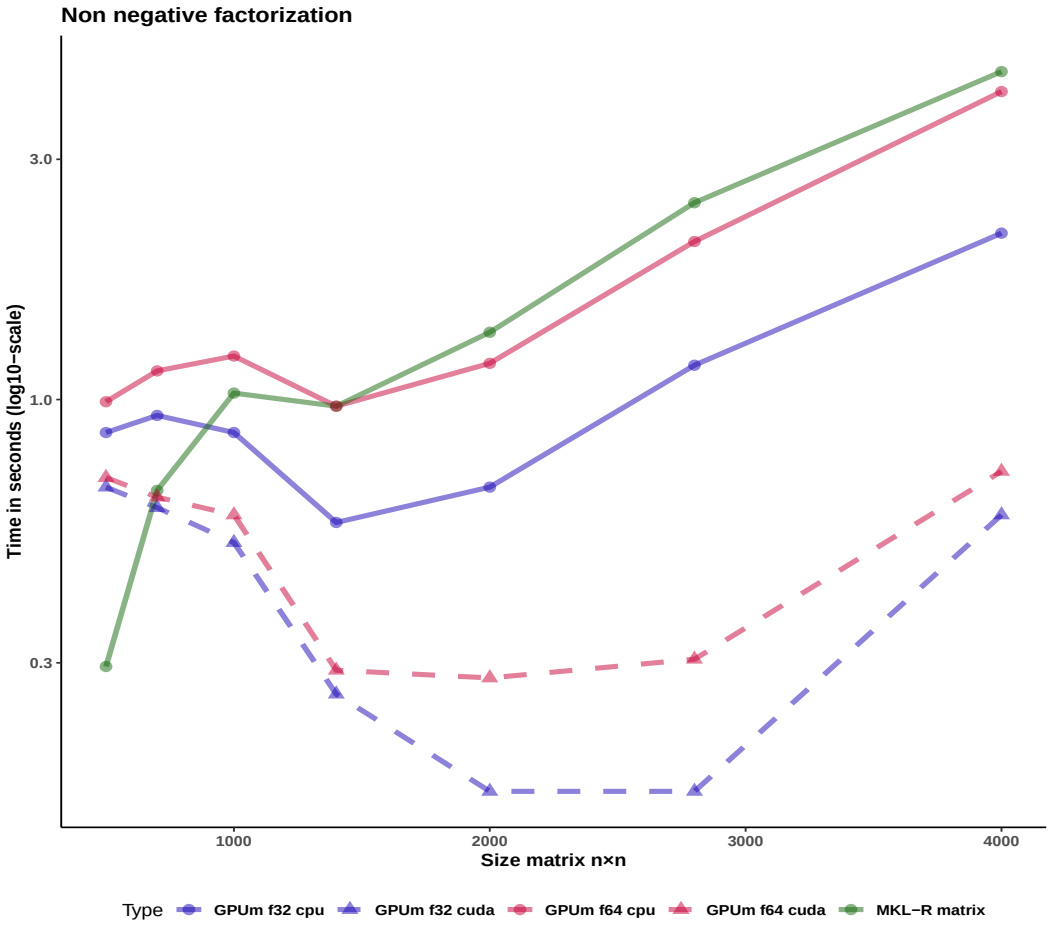



**Figure 1:** Computation time (in seconds) of non-negative factorization for MKL-R (i.e. R with the optimized MKL BLAS library, solid green), solid lines for CPU, dashed lines for GPU with CUDA, pink lines for GPUmatrix with float64, and blue lines for GPUmatrix with float32. Time shown in y-axis is in logarithmic scale. All calculations are performed on square matrices. The x-axis represents the number of rows in the matrices. The internal size of the factorization is 10.

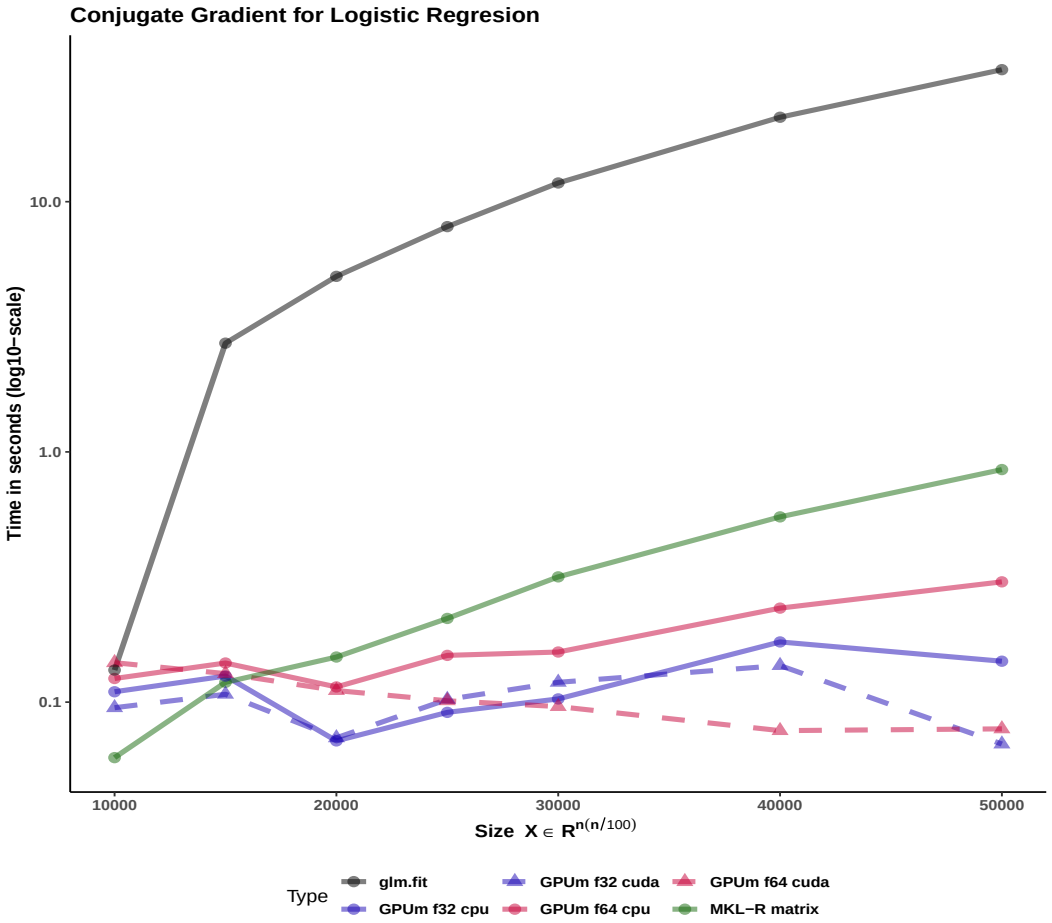



**Figure 2:** Computation time (in seconds) of the logistic regression using the conjugate gradient method for MKL-R (i.e. R with the optimized MKL BLAS library, solid green), solid lines for CPU, dashed lines for GPU with CUDA, pink lines for GPUmatrix with float64, and blue lines for GPUmatrix with float32. Time shown in y-axis is in logarithmic scale. The calculations are performed on random matrices whose size are n x (n/100). Therefore, the leftmost part of the graph shows the computing time for a 10,000 x 100 matrix and the rightmost part a 50,000 x 500 matrix.

### 3.2 Logistic regression of large models

Logistic regression is a widespread statistical analysis technique that is the "first to test" method for classification problems where the outcome is binary. R-base implements it in the `glm` function. However, `glm` can be very slow for big models and, in addition, does not accept sparse coefficient matrices as input. In this example, we have implemented a logistic regression solver that accepts as input both dense or sparse matrices.

The developed function performs the logistic regression using the Conjugate Gradient method (CG) Figure 2. This method has shown to be very effective for logistic regression of big models (Minka, 2003). The code is general enough to accommodate standard R matrices, sparse matrices from the Matrix package and, more interestingly, GPUmatrices from the GPUmatrix package.

We would like to stress several points here. Firstly, the conjugate gradient is an efficient technique that outperforms glm.fit in this case. Secondly, this code runs on matrix, Matrix or GPUmatrix objects without the need of carefully selecting the type of input. Thirdly, using the GPUmatrix accelerates the computation time two-fold if compared to standard R (and more than ten fold if compared to glm.fit function)

We have tested the function also using a sparse input. In this case, the memory requirements are much smaller. However, there are no advantages in execution time (despite the sparsity is 90 %). Torch (and Tensorflow) for R only provides a type of sparse matrices: the "coo" coding where each element is described by its position (i and j) and its value. Matrix (and torch and tensorflow for Python) include other storage models (column compressed format, for example) where the matrix computations can be better optimized. It seems that there is still room for improvement in the sparse matrix algebra in

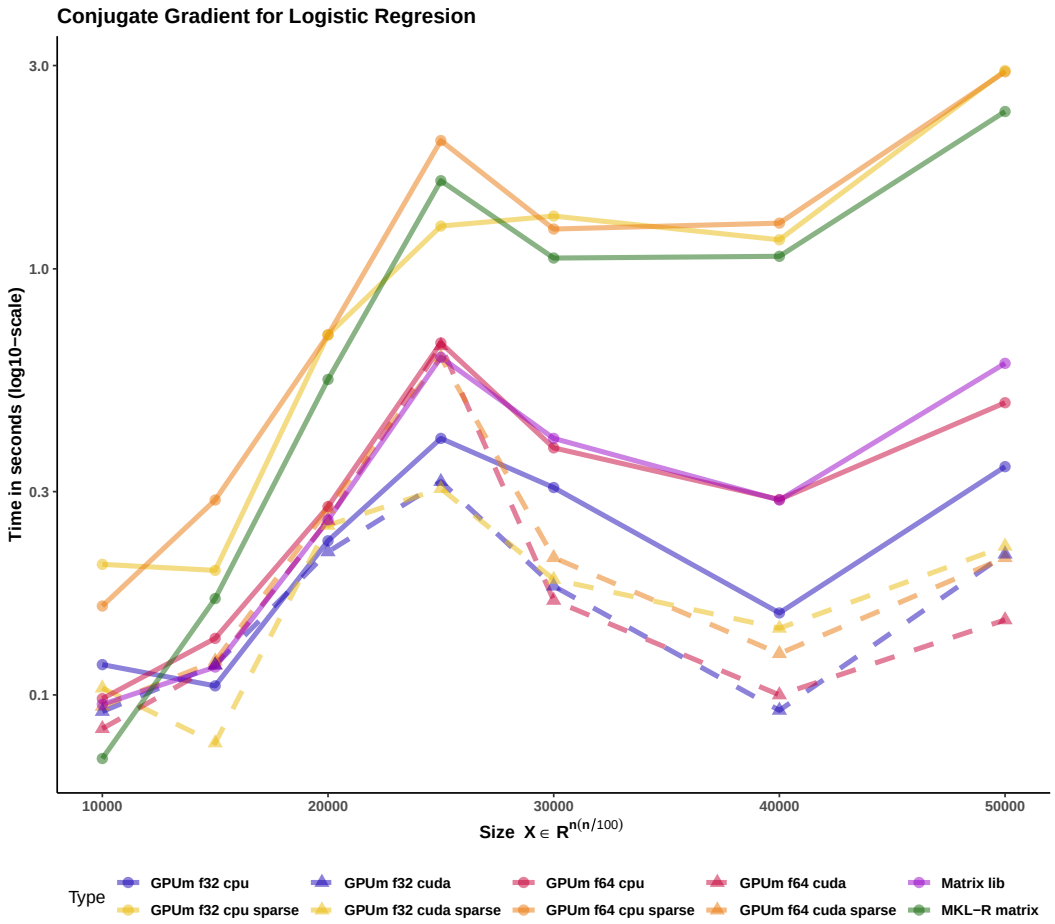



**Figure 3:** Computation time (in seconds) of the logistic regression using the conjugate gradient method in a sparse matrix. Solid green for MKL-R dense case (i.e. the computation is performed without any consideration of the sparsity of the matrix). Solid lines for CPU, dashed lines for GPU with CUDA, pink lines for GPUmatrix dense with float64, blue lines for GPUmatrix dense with float32, yellow lines for GPUmatrix sparse with float32, orange lines for GPUmatrix sparse with float64. Violet line, using Matrix package(that implicitly considers the matrix to be sparse). Time shown in y-axis is in logarithmic scale. The calculations are performed on random matrices whose size are n x (n/100). Therefore, the leftmost part of the graph shows the computing time for a 10,000 x 100 matrix and the rightmost part a 50,000 x 500 matrix.

Torch for R.

Sparse Matrix -that includes the column compressed form- performs extraordinary well in this case Figure 3. Despite Matrix is single-threaded, it is roughly ten times faster than GPUmatrix in sparse matrices.

### 3.3 General Linear Models

One of the most frequently used functions in R is `glm`, that stands for generalized linear models. In turn, `glm` relies on the `glm.fit`. This function uses the iteratively reweighted least squares algorithm to provide the solution of a generalized linear model. glm.fit, subsequently, calls a C function to do the "hard work" for solving the linear models, including the least squares solution of the intermediate linear systems.

Since glm.fit calls a C function is not easy to develop a "plug-in" substitute using GPUmatrix. Specifically, the qr algorithm in torch or tensorflow for R and in the C function (that is similar to base R) substantially differ: in torch the function returns directly the Q and R matrices whereas the C function called by glm.fit returns a matrix and several auxiliary vectors that can be used to reconstruct the Q and R matrices.

One side-effect of the dependency of base glm on a C function is that, even if R is compiled to use an optimized linear algebra library, the function does not exploit that and the execution time is much longer than expected for this task.

In order to ameliorate this problem, there are other packages (**fastlm** and **speedglm**) that mimic the behavior of the base glm. fastlm relies on using multithreaded RcppEigen functions to boost the performance. On the contrary, speedglm is written in plain R and the performance is increased especially if R runs using an optimized BLAS library.

We have used speedglm as starting point for our glm implementation in the GPU since all the code is written in R and includes no calls to external functions. Specifically, the most time-consuming task for large general linear models, is the solution of the intermediate least squares problems. In the GPUmatrix implementation, we used most of the code of speed.glm. The only changes are casts on the matrix of the least squares problem and the corresponding independent term to solve the problem using the GPU. Timing results of the last section show that `GPUglm` outperforms speed.glm for large models.

The following script illustrates how to use the `GPUglm` function in toy examples.

```
# Standard glm

counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
glm.D93 <- glm(counts ~ outcome + treatment, family = poisson())
summary(glm.D93)

#>
#> Call:
#> glm(formula = counts ~ outcome + treatment, family = poisson())
#>
#> Coefficients:
#>              Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  3.045e+00  1.709e-01  17.815   <2e-16 ***
#> outcome2    -4.543e-01  2.022e-01  -2.247   0.0246 *
#> outcome3    -2.930e-01  1.927e-01  -1.520   0.1285
#> treatment2   1.338e-15  2.000e-01   0.000   1.0000
#> treatment3   1.421e-15  2.000e-01   0.000   1.0000
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for poisson family taken to be 1)
#>
#>     Null deviance: 10.5814  on 8  degrees of freedom
#> Residual deviance:  5.1291  on 4  degrees of freedom
#> AIC: 56.761
#>
#> Number of Fisher Scoring iterations: 4
```

```
# Using speedglm
library(speedglm)
sglm.D93 <- speedglm(counts ~ outcome + treatment, family = poisson())
summary(sglm.D93)

#> Generalized Linear Model of class 'speedglm':
#>
#> Call:  speedglm(formula = counts ~ outcome + treatment, family = poisson())
#>
#> Coefficients:
#>  ---------------------------------------------------------------------
#>              Estimate Std. Error    z value  Pr(>|z|)
#> (Intercept)  3.045e+00     0.1709  1.781e+01 5.427e-71 ***
#> outcome2    -4.543e-01     0.2022 -2.247e+00 2.465e-02 *
#> outcome3    -2.930e-01     0.1927 -1.520e+00 1.285e-01
#> treatment2  -5.309e-17     0.2000 -2.655e-16 1.000e+00
#> treatment3  -2.655e-17     0.2000 -1.327e-16 1.000e+00
#>
#> ---------------------------------------------------------------------
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> ---
#> null df: 8; null deviance: 10.58;
#> residuals df: 4; residuals deviance: 5.13;
#> # obs.: 9; # non-zero weighted obs.: 9;
#> AIC: 56.76132; log Likelihood: -23.38066;
#> RSS: 5.2; dispersion: 1; iterations: 4;
#> rank: 5; max tolerance: 2e-14; convergence: TRUE.

# GPU glm
library(GPUmatrix)
gpu.glm.D93 <- GPUglm(counts ~ outcome + treatment, family = poisson())
summary(gpu.glm.D93)

#> Generalized Linear Model of class 'summary.GPUglm':
#>
#> Call:  glm(formula = ..1, family = ..2, method = "glm.fit.GPU")
#>
#> Coefficients:
#>  ---------------------------------------------------------------------
#>              Estimate Std. Error    z value  Pr(>|z|)
#> (Intercept)  3.045e+00     0.1709  1.781e+01 5.427e-71 ***
#> outcome2    -4.543e-01     0.2022 -2.247e+00 2.465e-02 *
#> outcome3    -2.930e-01     0.1927 -1.520e+00 1.285e-01
#> treatment2   2.049e-15     0.2000  1.025e-14 1.000e+00
#> treatment3   2.099e-15     0.2000  1.050e-14 1.000e+00
#>
#> ---------------------------------------------------------------------
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> ---
#> null df: 8; null deviance: 10.58;
#> residuals df: 4; residuals deviance: 5.13;
#> # obs.: 9; # non-zero weighted obs.: 9;
#> AIC: 56.76132; log Likelihood: -23.38066;
#> RSS: 5.2; dispersion: 1; iterations: 4;
#> rank: 5; max tolerance: 2.28e-14; convergence: TRUE.
```

Results are identical in this test. Figure 4 compares the timing of the implementation using speedglm, glm and the implementation of glm in GPUmatrix. GPUmatrix outperforms speedglm with R-MKL. This is especially apparent for float32 matrices.

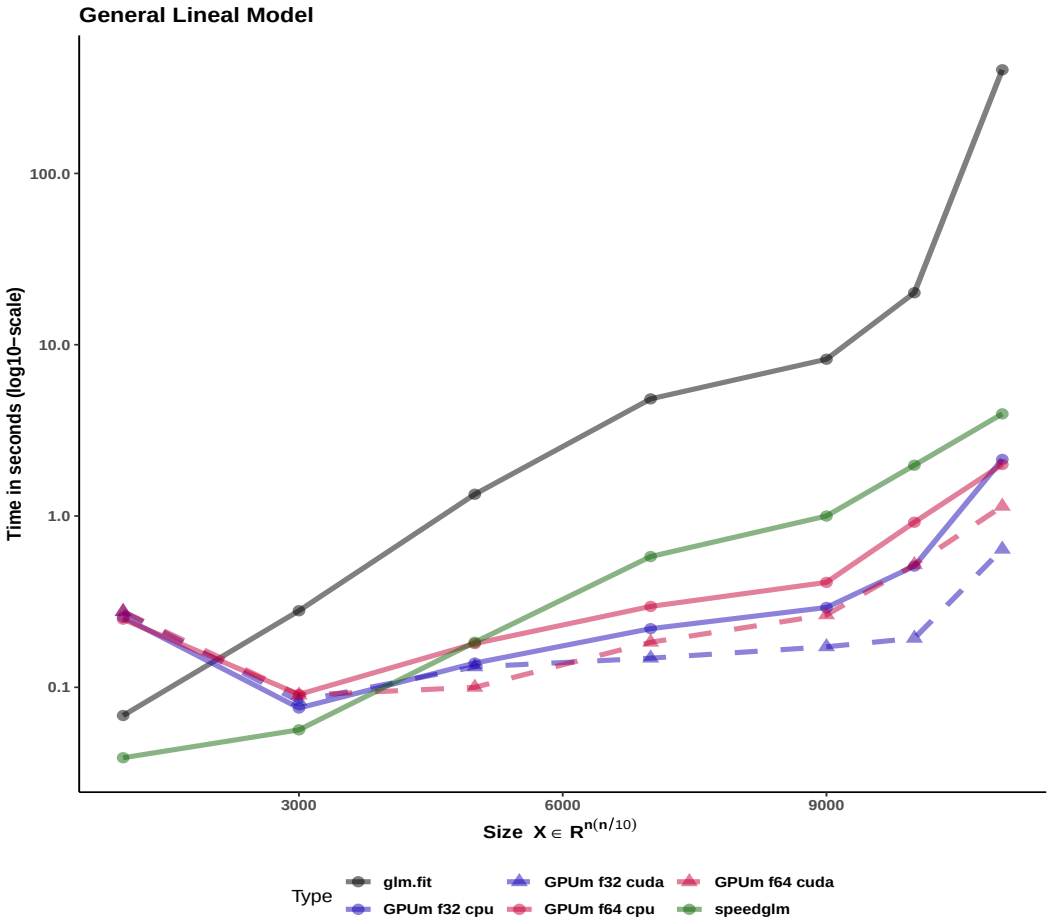



**Figure 4:** Computation time (in seconds) of general linear model using *speedglm* function with MKL-R matrix (i.e. R with the optimized MKL BLAS library, solid green), solid black line for glm function, solid lines for CPU, dashed lines for GPU with CUDA, pink lines for GPUmatrix with float64, and blue lines for GPUmatrix with float32. Time shown in y-axis is in logarithmic scale. The calculations are performed on random matrices whose size are n x (n/10). Therefore, the leftmost part of the graph shows the computing time for a 1,000 x 100 matrix and the rightmost part a 10,000 x 1000 matrix.

## 4 Performance comparison

We have compared the computation time for different matrix functions, different precision types and running the operations either on the GPU or on the CPU (using both GPUmatrix and R with MKL as linear algebra library).

The functions that we tested are * (Hadamard or element-wise product of matrices), exp, (exponential of each element of a matrix), rowMeans (means of the rows of a matrix), \%*\% (standard product of matrices), solve (inverse of a matrix) and svd (singular value decomposition of a matrix).

These functions were tested on square matrices whose row sizes are 500, 700, 1000, 1400, 2000, 2800 and 4000.

Figure 5 compares the different computational architectures, namely, CPU -standard R matrices running on MKL on FP64-, CPU64 -GPUmatrix matrices computed on the CPU with FP64-, CPU32 -similar to the previous using FP32-, GPU64 -GPUmatrix matrices stored and computed on the GPU with FP64- and GPU32 -identical to the previous using FP32-.

It is important to note that the y-axis is in logarithmic scale. For example, the element-wise product of a matrix on GPU32 is around five times faster than the same operation on CPU. The results show that the GPU is particularly effective for element-wise operations (Hadamard product, exponential of a matrix). For these operations, it is easier to fully utilize the huge number of cores of a GPU. R-MKL seems to use a single core to perform element-wise operations. The torch implementation is much faster, but not as much as using the GPU. rowMeans is also faster on the GPU than on the CPU. In this case, the GPUmatrix CPU implementation is on par with the GPU. When the operations become more complex, as in the standard product of matrices and computing the inverse, CPU and GPU (using double precision) are closer to each other. However, GPU32 is still much faster than its CPU32 counterpart. Finally, it is not advisable -in terms of speed- to use the GPU for even more complex operations such as the SVD. In this case, GPU64 is the slowest method. GPU32 hardly stands up the comparison with CPU32.

Regarding sparse matrices, Torch or Tensorflow for R have very limited support for them. In many cases, it is necessary to cast the sparse matrix to a full matrix and perform the operations on the full matrix. However, there are two operations that do not require this casting: element-wise multiplication of matrices and multiplication of a sparse matrix and a full matrix. We created four sparse matrices. The size of two of them (small ones) is 2,000 x 2,000. The fraction of nonzero entries is 0.01 and 0.001. The size of the other two (large ones) is 20,000 x 20,000. The fraction of non-null entries is also 0.01 and 0.001. Figure 6 shows the results for these matrices. The element-wise multiplication is faster using GPUmatrix. On the contrary, the time to multiply a sparse and a full matrices is similar using Matrix and GPUmatrix. As mentioned earlier, the implementation of sparse operations in torch or tensorflow for R is far from perfect, probably, because the only storage mode is “coo” in their respective R packages.

## 5 Discusion

GPUmatrix is an R package that facilitates the use of GPUs for linear algebra operations. It relies heavily on either the torch or tensorflow packages. GPUmatrix will work as long as torch or tensorflow work. Currently, the R implementation of these packages lags behind their Python counterparts. For example, in the case of sparse matrices, only the coo sparse tensor implementation is allowed, although compressed row and column formats are readily available in Python torch. Nevertheless, we have shown that GPUmatrix is a convenient package for easily integrating GPU computations into R programs.

GPUmatrix, like any software that performs a computation on the GPU, has some drawbacks. There is a large overhead associated with moving data from main memory to GPU memory. For simple operations with small objects, the time required to move the object can be greater than the computation itself. We have taken special care to avoid unnecessary movements from CPU to GPU memory and vice versa. In some cases – an operation with a matrix in the CPU and a matrix in the GPU – this overhead is unavoidable. In this case, we favored the GPU memory and the result will be a GPU matrix.

GPUmatrix also introduces some overhead due to casting objects from one type to another. These considerations lead to an important question: when is it worth using the GPU for linear algebra operations? The answer, of course, is that it depends. Here are some factors that affect the answer to this question.

It depends on the size of the operations and the type of operations. GPU is faster (and compensates the burden of moving the data from the CPU to the GPU memory) only when the operations are

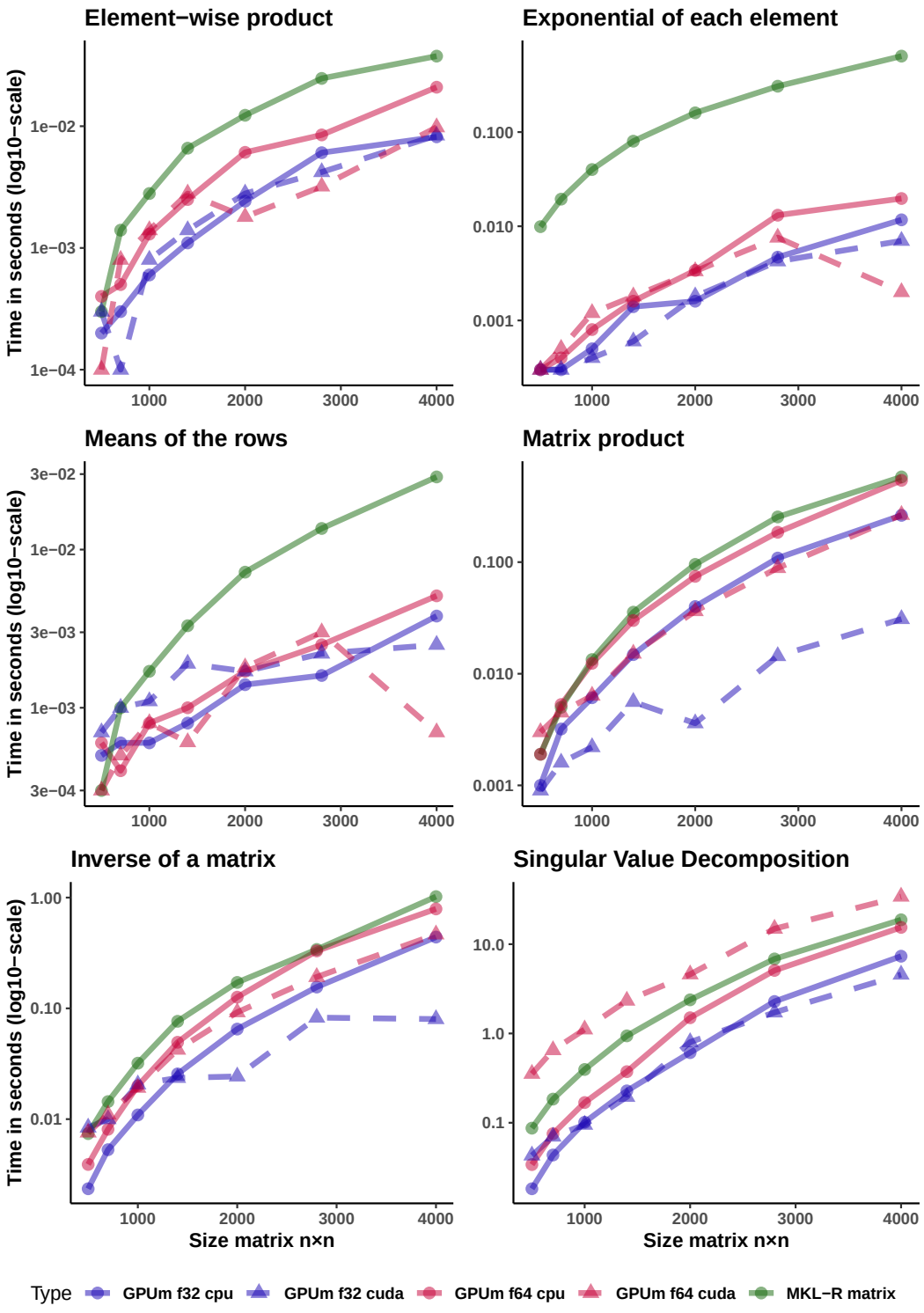



**Figure 5:** Computation time (in seconds) where MKL-R solid green, solid lines for CPU, dashed lines for GPU with CUDA, pink lines for GPUmatrix with float64 and blue lines for GPUmatrix with float32. All calculations are performed on square matrices. The x-axis represents the number of rows in the matrices. The operations are the element-wise or Hadamard product of two matrices, the exponential of each element of a matrix, the mean of the rows of a matrix, the standard matrix product, the inverse of a matrix, and the singular value decomposition of a matrix (SVD).

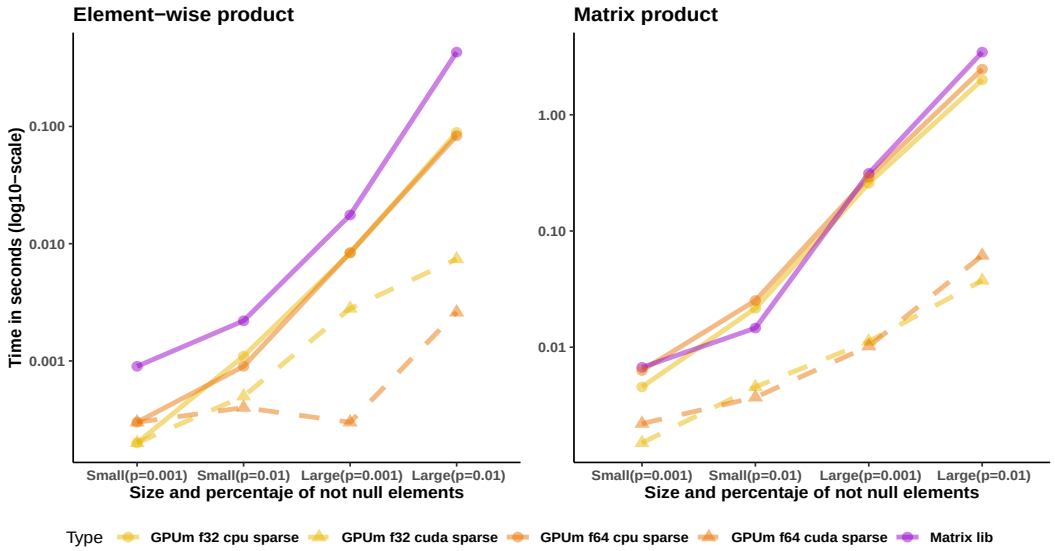



**Figure 6:** Computation time (in seconds) for the Matrix package (solid violet), yellow lines for GPUmatrix with float32, orange lines for GPUmatrix with float64, solid lines for CPU and dashed lines for GPU with CUDA. Time shown in y-axis is in logarithmic scale. The small model is a random square matrix of size 2,000 x 2,000. The proportion of non-zero elements is either 0.001 or 0.01. The large model is a 20,000 x 20,000 matrix with the same proportion of non-zero elements. The element-wise multiplication is performed on the sparse matrix. The right panel shows the time required to multiply these matrices by dense matrices whose sizes are 2,000 x 500 and 20,000 x 500, respectively.

performed on large matrices (each dimension is at least in the hundreds or even the thousands of elements). It also depends on the nature of the operations. Element-wise operations are easily parallelized. However, more complex algorithms such as the SVD factorizations are much harder to implement in parallel, so the improvement from using the GPU is small (if any).

Of course, it also depends on the relative performance of the CPU and GPU. We tested GPUmatrix on several systems (Computer 1: i7-10700 plus RTX 2080Ti, Computer 2: i7-11700 plus RTX 3060 Ti, Computer 3: i9-10940X and RTX A4000). Only results for Computer 1 are shown. These desktops can be considered mid-range to high-end in terms of both GPU and CPU. Computer 3 has an i9 processor with 14 cores. Although the graphics card is slightly better for this computer - unless using FP32 - the CPU is generally faster than the GPU in this case. In Computer 2, the RTX3060Ti is particularly fast with FP32 operations, and this definitely affects its performance.

One of the key points is that GPUs excel at FP32 operations. In the case of a CPU, FP32 operations are about twice as fast as FP64. In the case of GPUs, this gain theoretically skyrockets to 32 times faster (for the NVIDIA 2080Ti and the A4000) or even 64 times faster (for the 3060Ti). In practice, these factors are smaller, closer to 10 or 20. Still, the difference is huge, especially when compared to the CPU. If the operations can be done in FP32, then the GPU is usually the fastest method regardless of the type of operation.

Another crucial point is the linear algebra library installed on the CPU. The standard R library does not take advantage of multiple cores, and all operations run in a single thread. OpenBlas, Intel MKL, or Accelerate (for Mac OS) are popular substitutes to enhance the standard linear algebra library. Using these libraries requires tweaking the R installation, and in some cases there may be unwanted side effects. In addition, although OpenBlas (in Linux) and Accelerate accept FP32 data as input and speed up about two times compared to FP64 data, the Microsoft R implementation of MKL does not include any speedup for FP32. In fact, FP32 data from the float package is forced to use single-threaded operations. As a result, CPU operations for FP32 are even slower than for FP64 and therefore, these results were not included.

Torch is the default backbone for GPUmatrix, although Tensorflow can also be used. An advantage of Torch is that operations can be performed on either the GPU or the CPU. Torch implements a very efficient version of MKL on the CPU. Even if a GPU is not available, using GPUmatrix makes it possible to use MKL (even for FP32 data) without changing standard R. Since the standard R library is unchanged, there are no problems with packages that depend on specific functions of it. Operations on torch tensors are slightly faster than on tensorflow tensors (data not shown). Surprisingly, performing the operations on either CPU64 or CPU32 (i.e., using the GPUmatrix package) is slightly faster than using the CPU with R-MKL optimization.

GPUmatrix stands "on the shoulders of giants". As long as either torch or tensorflow is properly installed and GPU enabled, GPUmatrix will take advantage of it. Since these packages are the workhorses of Rstudio, they will be maintained for the foreseeable future, guaranteeing the functionality of GPUmatrix. GPUmatrix is a first approach to filling the gap of harnessing GPU power in R programs. It is easy to use and allows seamless use of GPU power with little change to the standard code.

## 6 Acknowledgments

This research was funded by Cancer Research UK [C355/A26819] and FC AECC and AIRC under the Accelerator Award Programme, the project PIBA_2020_1_0055 (funded by the Basque Government) and the Synlethal Project (RETOS Investigacion, Spanish Government). Conflict of Interest: none declared.

## 7 Summary

To see more available functions, check the **GPUmatrix** package on CRAN.

## References

J. Allaire and Y. Tang. tensorflow: R interface to 'tensorflow'. 2022. URL https://CRAN.R-project.org/package=tensorflow. R package version 2.11.0. [p158]

D. Bates, M. Maechler, and M. Jagan. Matrix: Sparse and dense matrix classes and methods. 2023. URL https://CRAN.R-project.org/package=Matrix. R package version 1.5-4. [p159]

H. Bengtsson. matrixstats: Functions that apply to rows and columns of matrices (and to vectors). 2022. URL https://CRAN.R-project.org/package=matrixStats. R package version 0.63.0. [p159]

V. Bonnici, F. Busato, S. Aldegheri, M. Akhmedov, L. Cascione, A. A. Carmena, F. Bertoni, N. Bombieri, I. Kwee, and R. Giugno. curnet: an r package for graph traversing on gpu. *BMC Bioinformatics*, 19:356, 2018. ISSN 1471-2105. doi: 10.1186/s12859-018-2310-3. URL https://doi.org/10.1186/s12859-018-2310-3. [p158]

J. Buckner, J. Wilson, M. Seligman, B. Athey, S. Watson, and F. Meng. The gputools package enables GPU computing in R. *Bioinformatics*, 26(1):134–135, 10 2009. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp608. URL https://doi.org/10.1093/bioinformatics/btp608. [p158]

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. pages 785–794, 2016. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785. [p158]

R. S. de Souza, X. Quanfeng, S. Shen, C. Peng, and Z. Mu. qrpca: QR-based Principal Components Analysis. art. ascl:2208.002, Aug. 2022. [p158]

C. Determan. A short introduction to the gpur package. *Vignette*, 2017. [p158]

D. Falbel and J. Luraschi. *torch: Tensors and Neural Networks with 'GPU' Acceleration*, 2023. URL https://CRAN.R-project.org/package=torch. R package version 0.11.0. [p158]

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, Oct 1999. ISSN 1476-4687. doi: 10.1038/44565. URL https://doi.org/10.1038/44565. [p167]

T. P. Minka. A comparison of numerical optimizers for logistic regression. 2003. URL https://tminka.github.io/papers/logreg/minka-logreg.pdf. [p169]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022. URL https://www.R-project.org/. [p158]

RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, PBC., Boston, MA, 2020. URL http://www.rstudio.com/. [p158]

D. Schmidt. *Introducing the float package: 32-Bit Floats for R*, 2017. URL https://cran.r-project.org/package=float. R Vignette. [p159]

J. Wang and M. Morgan. gpumagic: An opencl compiler with the capacity to compile r functions and run the code on gpu. 2022. R package version 1.14.0. [p158]

Z. Wang, T. Chu, L. A. Choate, and C. G. Danko. Rgtsvm: Support vector machines on a gpu in r, 2017. [p158]

*César Lobato-Fernández*
*University of Navarra*
*Department of Biomedical Engineering and Sciences*
*Paseo Mikeletegui,48, 20009, San Sebastian, Gipuzkoa, Spain*
*ORCiD: 0000-0001-9576-7236*
clobatofern@unav.es

*Juan A. Ferrer-Bonsoms*
*University of Navarra*
*Department of Biomedical Engineering and Sciences*
*Paseo Mikeletegui,48, 20009, San Sebastian, Gipuzkoa, Spain*
*ORCiD: 0000-0002-1588-2195*
jafhernandez@unav.es

*Angel Rubio*
*University of Navarra*
*Department of Biomedical Engineering and Sciences*
*Paseo Mikeletegui,48, 20009, San Sebastian, Gipuzkoa, Spain*
*ORCiD: 0000-0002-3274-2450*
arubio@unav.es

# UpAndDownPlots: An R Package for Displaying Absolute and Percentage Changes

*by Antony Unwin*

**Abstract** UpAndDown plots display the ups and downs of sector changes that make up an overall change between two time points. They show percentage changes by height and absolute changes by area. Most alternative displays only show percentage changes. UpAndDown plots can visualise changes in indices, in consumer markets, in stock markets, in elections, showing how the changes for sectors or for individual components contribute to the overall change. Examples in this paper include the UK's Consumer Price Index, Northern Ireland population estimates, and the German car market.

## 1 Introduction

There are many situations when overall percentage changes between two time points are broken down by subgroups. When share indices go up or down, news reports refer to which sectors and shares moved most. Governments produce many official indices, for instance the Consumer Price Index or CPI, and readers want to know what has driven the overall change in the index, which components have changed most. When companies study the consumer markets they supply, such as energy or cars, they are particularly interested in how well their own products have done and how their market shares have moved. When changes are discussed for a country as a whole, then people usually want to know what changes occurred in the individual regions. In all these cases, the percentage changes of the components are generally considered first, although the importance of a particular component, how much it makes up of an index or market, is also highly relevant. A small company which leaps in value, say an IT startup, may be a positive surprise, but it does not move the stock market the way a smaller change for a big company like Microsoft or Apple can. The same goes for fringe products in a consumer market. A specialist chocolate bar may double its sales thanks to a promotional campaign without selling nearly as many additional bars as would match a 1% increase in Mars bar sales. Percentage changes for subgroups are often displayed using lengths, with one bar for each sector and all bars having the same width.

### 1.1 UpAndDown plots

Only displaying percentage changes ignores how important those changes might be. UpAndDown plots use bar heights to represent percentage change and bar widths to display the initial contributions of sectors. Bar areas then represent absolute changes, so that amounts can be compared as well as percentages. Plotting in this way means that the length of the horizontal axis represents the sum of the individual sector contributions at the first time point, rather like a horizontal stacked barchart of no width.

It turns out that area in these examples has a valuable conservation property: if the bar rectangle for a sector is split up horizontally into subsectors and individual bars drawn for these, then the total area of the bars for the subsectors equals the area of the original bar for the whole sector. This means that you can drill down across UpAndDown plots. Displays of multiple levels are consistent, a valuable property when interpreting them.

A new package is required for drawing UpAndDown plots because of the need to draw bars whose width depends on the value of a variable. Flexible as it is, **ggplot2** does not offer this functionality directly. **UpAndDownPlots** is based on **ggplot2**, adding the capability to handle hierarchies as well as tools for ordering and sorting the plots.

### 1.2 Related graphics

UpAndDown plots use both length and area to represent changes. Length is preferable, as Cleveland and McGill (1987) discussed long ago, but area has been used effectively in several displays, especially for different kinds of mosaicplot (Hofmann, 2003) and for treemaps (Shneiderman, 1992). Area is necessary together with length in UpAndDown plots because two different values are displayed.

Two early examples of area-bar charts or sky-line charts can be found in Karsten (1923). One, of average income by occupation, even includes a second level. These examples only have positive

lengths, but Brinton (1939) includes an example in a chapter on area-bar charts of negative values: price distortion by company sales volume. Some recent examples of area-bar charts can be found in MacKay (2009).

Doubledecker plots (Hofmann, 2008), a version of mosaicplots, are used for displaying proportions in subgroups by shading the appropriate height of a bar representing the whole group. All bars are the same height and bar widths show the subgroup size. Doubledecker plots are not suitable for datasets with negative values or values of over 100%. Proportions are highlighted as part of a whole. Usually there are small gaps between the bars to separate the subgroups. UpAndDown plots could be used for these kind of data, but the upper limit of 100% would not be emphasized. On the other hand they would be effective for data sets with low percentage rates such as rare diseases or unemployment rates where the upper limit is not relevant. Treemaps are useful for displaying overall structure but can only show negative values using colour, e.g., as with **treemap** (Tennekes, 2023).

Some demographers have used the term Skyline plots to refer to plots of population size over time, where the size is often constant over periods of time (Pybus et al., 2000). These plots generally have no vertical lines, the heights are never negative, and they may include uncertainty intervals. UpAndDown plots can look similar, but include vertical lines, the heights can be negative, and the horizontal axis represents shares not time.

The data structure discussed so far has assumed that only two fixed time points are considered, an initial base point and a final point. There may be several time points over a longer period and then a display of a succession of changes is wanted. Time series graphics could be used, possibly with line thickness representing amounts. For seasonal data, such as ice-cream, there might be interest in monthly changes, differences to the same month last year or differences to year-to-date last year.

### 1.3 Outline of article

The type of data that can be visualised by UpAndDown plots is discussed first, including what classification hierarchies can arise. The next section presents the mathematics underlying the plots and shows how a particular version of the plot can be used to display changes in brand shares in a market. There is a section on using the package and one on varying UpAndDown plots using different orderings of levels, different sortings within the levels, and colour. There are three detailed applications with annotated code snippets to show how the package can be applied in practice. A summary section concludes the paper.

## 2 What kind of data can be visualised in an UpAndDown plot?

To display changes between two time points, data are needed for both end points for all items. The simplest example is a consumer market in which performance is measured by unit sales. An UpAndDown plot can be drawn if there is a complete list of the products in the market and the unit sales of the products at the initial and end time points are available. All products are treated equally and have the same weight. There are two main advantages of UpAndDown plots. They show both percentage and absolute changes for each product and they allow the user to drill down through classifications to show changes within levels as well as across levels.

More complicated examples involve unequal weightings and multiple grouping variables. Share indices are commonly constructed by weighting share prices by the numbers of issued shares, i.e. it is the company capitalisations that make up the index not the raw share prices. UpAndDown plots display the changes in capitalisations. As long as the numbers of shares do not change (because of a rights issue or a company buyback plan) the percentage change in a company's capitalisation will be the same as the percentage change in share price. A further weighting variant arises with governmental or other indices that are weighted sums of components. The CPI is an example. An individual component's contribution to the percentage change in the index depends on the weighted value at the first time point not just on the weight.

Sometimes a set of components is changed, for instance when the CPI is redefined or when stocks are removed from a financial index and others added or when new products are introduced into a consumer market, for example a new chocolate bar. In all these situations it is not easy to make comparisons at the component level, as corrective adjustments are necessary. UpAndDown plots display changes and the changes must be for comparable data (or for data that has been made comparable). The CPI uses chaining to maintain consistency (ONS, 2014).

### 2.1 Multiple classification levels and nesting

Subgroups and components can be hierarchical (nested), as in the CPI, which has `Sector`, `Subsector`, and `Component` levels, or have no predetermined order. The Northern Ireland Statistics and Research Agency produces population estimates (NISRA, 2019). They look at population over time broken down by four classifications: `age` (four groups), `gender` (two), `LGD2014_name` (Local Government Districts or LGDs, eleven), and `area_name` (District Electoral Areas or DEAs, eighty). Each DEA is a subarea of one LGD, so those two are nested (or form a hierarchy) while the others do not. If both levels of a nesting are to be plotted, then the higher level must come first.

If levels form a complete hierarchy, as with the CPI, then there is only one possible ordering of levels, but many different orderings are possible within the levels. With a partial hierarchy, as with the Northern Ireland data, and assuming the three levels to be plotted were `gender`, `LGD2014_name` and `area_name`, then the ordering of the levels would have to have `gender` either first or last, keeping the nested levels together. If there is no nesting and $p$ classification variables then there are $p!$ orderings of the levels and, again, many different orderings within the levels. In these situations it makes sense to order consistently within the levels. Suppose `age` and then `gender` are plotted for the Northern Ireland data. Plotting male and then female for the three youngest age groups, but female and then male for the oldest age group would just be confusing.

An additional, unusual form of nesting is "double-nesting", where one grouping variable is separately nested in each of two other variables. This can arise with car sales, if models are nested in both market segment and manufacturer. Suppose there are 3 sectors and 5 manufacturers, where each manufacturer offers several models across the sectors. There is a hierarchical relationship between sectors and models (each model is in only one sector) and another between manufacturers and models (each model is produced by only one manufacturer). The model level should then be placed last.

The package's `ud_prep` function includes a function that checks what kind of nesting, if any, exists in the data. It also checks that the ordering of levels specified by the user is permissible, and calculates the relevant statistics. One of the package vignettes discusses nesting and includes plot examples.

### 2.2 Dataset structure

An UpAndDown plot dataset must contain at least two numeric variables representing the item values at the start and end points. There should be at least one classification variable defining the groups. If items are not equally weighted, then there has to be a weight variable. There is no need to provide differences or percentage changes, as the package calculates these from the data as required. The CPIuk dataset includes component index values for August 2017 and August 2018, three classification variables (Sector, Subsector, and Component), and a weighting variable. The AutoSalesX dataset for the German car market includes sales in 2017 and 2018, the three classification variables of Sector, Segment, and Manufacturer, and no weighting variable. Here is an example taking the first few lines of AutoSalesX.

```
#>   Sector Segment Manufacturer sales17 sales18
#> 1    Car Compact   ALFA ROMEO    1171    1008
#> 2    Car Compact         AUDI   49820   45901
#> 3    Car Compact          BMW   86843   82332
#> 4    Car Compact      CITROEN    8390    6308
#> 5    Car Compact        DACIA   31920   32659
#> 6    Car Compact         FIAT   14187    8181
```

## 3 Mathematics of UpAndDown plots

All the bars in a barchart of percentage changes have the same width. In an UpAndDown plot each bar for a sector has a width proportional to the sector's importance at the initial time point. This allows users to drill down consistently to study changes within sectors. The key property is that area is conserved, i.e., that the sum of the areas of subsector bars equals the area of the sector's bar. Consider a dataset of sales of cars and vans. If the cars subsector goes up by 10% and the vans subsector by 20%, how much would the combined sector of cars and vans together change? If nine times more cars than vans were sold last year then it would be 11%. UpAndDown plots display changes for sectors and subsectors appropriately. Regardless of how the sectors are broken down or grouped, the total area remains constant. Other plots commonly in use cannot manage this. Area conservation can be shown as follows.

Assume an index is made up of $m$ components with weight $w_i$ for component $i$ and values $v_{i1}$ and $v_{i2}$ recorded at two time points, $t_1$ and $t_2$.

The index value at time $t_j$ is

$$T_j = \sum_{i=1}^{m} w_i * v_{ij}$$

The overall absolute change over the interval $(t_1, t_2)$ is

$$C = \sum_{i=1}^{m} w_i * (v_{i2} - v_{i1}) = T_2 - T_1$$

and the overall percentage change is

$$pC = 100 * \frac{C}{T_1}$$

The corresponding overall and percentage changes for the individual components are

$$c_i = (v_{i2} - v_{i1})$$

and

$$pc_i = 100 * \frac{c_i}{v_{i1}}$$

### 3.1 Area conservation

A single bar with the overall percentage change, $y = 100 * \frac{T_2 - T_1}{T_1}$, as height and last year's index value as width would have an area of

$$100 * (T_2 - T_1)$$

Individual bars can be drawn for each component $i$ with widths $w_i * v_{i1}$, the components' contributions to the index value at time $t_1$, so that their total width is

$$\sum_{i=1}^{m} w_i * v_{i1} = T_1$$

the same as the width of the single bar for the whole index. If the individual bars are drawn with heights $pc_i$, the percentage changes for the components, then the area of each bar is

$$a_i = (w_i * v_{i1}) * pc_i = 100 * w_i * (v_{i2} - v_{i1})$$

the weighted absolute change for component $i$. The sum of the areas of the $m$ individual bars is

$$\sum_{i=1}^{m} a_i = 100 * \sum_{i=1}^{m} w_i * (v_{i2} - v_{i1}) = 100 * (T_2 - T_1)$$

the same as the area of the bar for the whole index.

### 3.2 Scales

The default vertical scale for an UpAndDown plot depends on the largest positive and negative percentage changes. Sometimes there are small components with very large percentage changes that are not of major importance and the resulting scale masks the details of the rest of the display. One way to get round this is to limit the scale using the vscale option in the `ud_plot` function. This will trim any bars with larger heights at that value. (Using the ggplot2 option `ylim` would exclude components with larger heights.) Any large percentage changes still stand out, but no longer affect the bulk of the display.

### 3.3 Change of baseline and market share change

UpAndDown plots are drawn by default with a baseline of 0 marking no change. In some cases, it is more informative to use an alternative baseline, such as the overall market change. With that choice, the plot emphasizes which components performed better than the market and which performed worse. Area is still conserved if the baseline is set differently to 0, as all bar widths remain as they were and all heights are changed by the same amount.

If the baseline is set at the overall percentage change $y$%, then the area representing the total market change is 0. The sum of the areas representing the components' changes is

$$\sum_{i=1}^{m}(w_i * v_i) * (pc_i - y) = 100 * (T_2 - T_1) - y * \sum_{i=1}^{m}(w_i * v_i) = 0$$

The baseline influences the look of the graphic and also the interpretations of the heights and areas of the bars. Using the default baseline of 0 means that a bar's height represents the percentage change between $t_1$ and $t_2$ while the bar's area represents the absolute change. If the overall percentage change is used as the baseline then a bar's height represents the percentage change relative to the overall change and the bar's area represents the absolute change relative to the overall absolute change. In fact there is more to it than that. The bar's area is proportional to the component's share change.

In consumer markets there can be more interest in changes in market share than in absolute changes. Given sales for brand $i$ of $v_{ij}$ in period $j$, the size of the market is $T_j = \sum_{i=1}^{m} v_{ij}$ and the market share of brand $i$ is

$$s_{ij} = \frac{v_{ij}}{T_j}$$

while the change in market share between the two periods is

$$s_{i2} - s_{i1} = \left(\frac{v_{i2}}{T_2} - \frac{v_{i1}}{T_1}\right) = \frac{v_{i1}}{T_2}\left(\frac{v_{i2}}{v_{i1}} - \frac{T_2}{T_1}\right)$$

If the baseline of each brand's bar is set to be the overall change in sales in the market, $\frac{T_2 - T_1}{T_1}$, then a bar going up shows a performance better than the market and means a gain in market share while a bar going down indicates a performance worse than the market and means a loss in market share. The height of the bar of brand $i$ is the performance difference compared with the market

$$\left(\frac{v_{i2} - v_{i1}}{v_{i1}}\right) - \left(\frac{T_2 - T_1}{T_1}\right) = \left(\frac{v_{i2}}{v_{i1}} - \frac{T_2}{T_1}\right)$$

and the area of the bar is

$$v_{i1} * \left(\frac{v_{i2}}{v_{i1}} - \frac{T_2}{T_1}\right)$$

which is $T_2$ times the change in market share, so the bar areas using the baseline of overall changes in sales represent the market share changes. The height of a rectangle in the UpAndDown plot shows the percentage brand change less the market percentage change, while the area of the rectangle reflects the change in a brand's market share. There is an application to the German car market in §8.

## 4 The package

The **UpAndDownPlots** package is available on CRAN (Unwin, 2024b). There are two main functions, `ud_prep` and `ud_plot`. `ud_prep` prepares the data for plotting. First it checks the input parameters and data. It then calculates the statistics needed for sorting, and sorts the (at most) three levels of classification. There are five sorting options that may be chosen at each level and an option as to whether they should be displayed in reverse or not. `ud_plot` draws the plot. After checking the input parameters, it calculates the cumulative statistics needed for drawing bars with different widths. The layered structure of **ggplot2** is used to draw a barchart layer for each of the levels specified. Two plots are prepared, a horizontal plot to display increases going upwards and decreases going downwards, and a vertical one, so that labels can be added to the bars of one of the levels. Finally, a dataset combining the statistics derived for drawing the plots and the original data is constructed.

Three further functions are included: `ud_colours` to allow the user to specify the colours used in plots; `sort5` to draw a set of plots for one level of changes to compare the outputs of the five sorting methods available; `dgroup` to compare the effects of different grouping variables by drawing a one-level plot for each one. The package help covers the functions in detail and there are several illustrative vignettes.

Three datasets are included: `NIpop` provides Northern Ireland population estimates over seven years from 2011 to 2017 by `age`, `gender`, Local Government District, and District Electoral Area; `CPIuk` provides Consumer Price Index values for the UK from 2017 and 2018 by `Sector`, `Subsector`, and `Component`; `AutoSales` (and the cleaned subset `AutoSalesX`) provide data from the German car market for the years 2017 and 2018 by `Sector`, `Segment`, `ModelSeries`, `Manufacturer`, and `Model`.

### 4.1 Drawing an UpAndDown plot

Drawing a plot with several layers is easy in **ggplot2** (Wickham, 2022), provided the layers are independent of one another. In UpAndDown plots they are not. Each level adds more detail, conditional on the higher levels already specified.

Ordering, sorting, and arranging are key elements in UpAndDown plots. They determine how attractive and informative a plot is. The classifying variables are ordered either by definition (as with a fully nested hierarchy) or to display particular conditional groupings. Classification variable categories are sorted to support comparisons between them. Finally, the graphic layers comprising the plot are arranged to convey the information as effectively as possible.

There are seven steps in preparing and drawing an UpAndDown plot:

1. Order the grouping levels, respecting any nestings that exist.
2. Calculate the statistics at each level that are used for sorting within levels.
3. Sort each level, respecting the natural orders of any grouping variables (e.g., `age`).
4. Calculate the data needed for drawing bars in levels. As bars have different widths, the bases for them are calculated cumulatively from left to right. The values will depend on how the level has been sorted.
5. Decide whether to fill the bars of one level with colour and define an appropriate colour palette if the default does not suit.
6. Prepare individual graphic layers. Given the order within a level and the colour choices across levels, each layer can be drawn.
7. Choose in which order to draw the levels. The default is to draw the levels in grouping order (`drawFrom`="BigToSmall") but the software allows the reverse order ("SmallToBig"). Layers drawn last are more visible. Layers with filled bars may cover information in other layers if not drawn first.

Although drawing more than three levels is not currently offered, as plots become overloaded, it would be possible. Statistic calculations and sortings of bars could be carried out for any number of levels. There are three possible alternative graphics for each individual layer, one with grey-filled bars, one with colour-filled bars and one with transparent bars. These would be prepared in step 6 to provide all options needed to put the layers together in step 7.

## 5 Varying UpAndDown plots

By default, an UpAndDown plot includes a red dotted line to indicate the overall percentage change. Using this as the baseline for the bars gives the plot a different interpretation, covered in §3.3. Other variants can be drawn by reordering classification levels and using colour. Drawing more than one level of subdivision on the same plot requires choosing suitable shading and transparency to keep the different levels perceptually separated. It is usually most effective to draw the lowest level first, but it can depend on which features are to be emphasized. As so often with graphics, the exception proves the rule. In principle, when there are few groups at each level, more levels could be displayed, but this can become confusing and the software is currently restricted to displaying at most three levels.

### 5.1 Orderings of UpAndDown plots

The same data can give rise to many different UpAndDown plots and it is worth choosing carefully from amongst a selection of versions, as different information will be conveyed by each display, sometimes more effectively, sometimes less. Figure 1 shows the Consumer Price Index in the default order provided by the UK Office for National Statistics. Figure 2 shows the index ordered by percentage changes of sectors and by percentage changes of components within sectors.

The individual categories of a level can be ordered in several ways using the R package `UpAndDownPlots`:

- `orig` original, as components arise in the dataset. Consistent orderings are important when comparing plots.
- `base` by initial size (bar width). This orders in terms of the initial importance of the components, emphasizing how the biggest and smallest changed.
- `final` by final size. This orders in terms of the final importance of the components.
- `perc` by ascending percentage change (bar height). Which components went up and which went down? Which components had the biggest relative change?
- `abs` by ascending absolute change (bar area). Which components had the biggest absolute change?

Other orderings could be considered too, using properties of the components, or even just an alphabetic ordering. They can be calculated in advance and used to order the dataset before applying the functions in the package. Setting the sort option to `orig` retains an ordering calculated in advance. This was done for the plot on the right of Figure 6, so that it could be sorted by the size of the changes in market share.

If the components are grouped into higher levels as with the CPI's subsector classification, then the components can be sorted by their subsector and then, within that, by one of the five approaches listed already. It is usually best to use the same sorting for each subsector for consistency of interpretation. A different situation arises when the variables defining the levels do not form a hierarchy. For instance, in the car sales dataset vehicles are split by `Sector` and by `Manufacturer`. A display could have a top level of `Manufacturer` and then split by `Sector` or order the levels the other way round. There is no nested hierarchy and however the top level is ordered, it is best to have the lower level ordered the same way for each top level grouping. Having the lower level consistently ordered for each top level category makes comparisons easier. A possible lower level ordering would be to take whatever ordering you would use if the lower level were the top level.

If inconsistent lower level orderings are desired, then it is best to define the required nested hierarchy by renaming the lower level categories accordingly. For the cars data this would mean using new classifications car_VW, car_Mercedes, van_VW etc if you split on `Sector` first, or VW_car, VW_van, Mercedes_car etc if you split on `Manufacturer` first.

If there is a percentage change of zero then a bar has no height or area, but the width shows its size. If a new component is introduced after the initial time, for instance a new car, then there is no initial weight and an infinitely tall bar. Cases like this must be excluded and a note should be added to the display.

### 5.2 Colour in UpAndDown plots

The default is to fill the bars of the top level with grey and draw transparent unfilled bars with either blue or brown borders on top. The percentage change for the whole system is marked with a red dotted line. Different colours may be used to fill bars for one of the levels drawn. In that case the top level bars are not filled with grey. Colour schemes are helpful in distinguishing sectors and particularly informative when associated with different groups (e.g., for political parties, as in Chapter 26 of Unwin (2024a)). Colours are specified in the function `ud_colours` and the default colour choices can be changed there as required. The help page for `ud_plot` gives the details.

### 5.3 Limitations of UpAndDown plots

UpAndDown plots are designed for comparing changes between two time points. They have not been extended to handle changes over several time points.

Cases with missing data have to excluded. Cases with initial values of 0 cannot be handled as any positive value at the end time point will give an infinite percentage change. Examples might be a new model of car, a new chocolate bar or even a new political party. Cases with very low initial values and high final ones can also lead to very high percentage changes, which then have to be trimmed as described in §3.2. Redefinitions of classification variables (e.g., in the car market when SUVs became common) or reclassifications of items (e.g., deciding a car now belongs to a different group) can cause problems. These are typical issues for any dataset involving data over time and are not restricted to UpAndDown plots.

All three applications in the following sections include large numbers of individual items or components. This means that plots of the full dataset have to be drawn very large or only display data at higher grouping levels. For some applications (e.g., the car market in §8), smaller manufacturers can be grouped together into a group called `Other`.

## 6 Visualising changes in the UK Consumer Price Index

The Consumer Price Index (CPI) summarises how prices change for a basket of a wide range of products and services. It is an important measure of inflation and influences both government policy and public attitudes. The basket is continually reviewed and revised as it is supposed to cover typical costs of living. It is said that Margaret Thatcher, when she was UK Prime Minister, supported the basket being amended yearly to reflect as quickly as possible changes in the public's spending habits. This meant that the public's switching to cheaper alternatives was identified and included more quickly, having a lowering effect on the index.

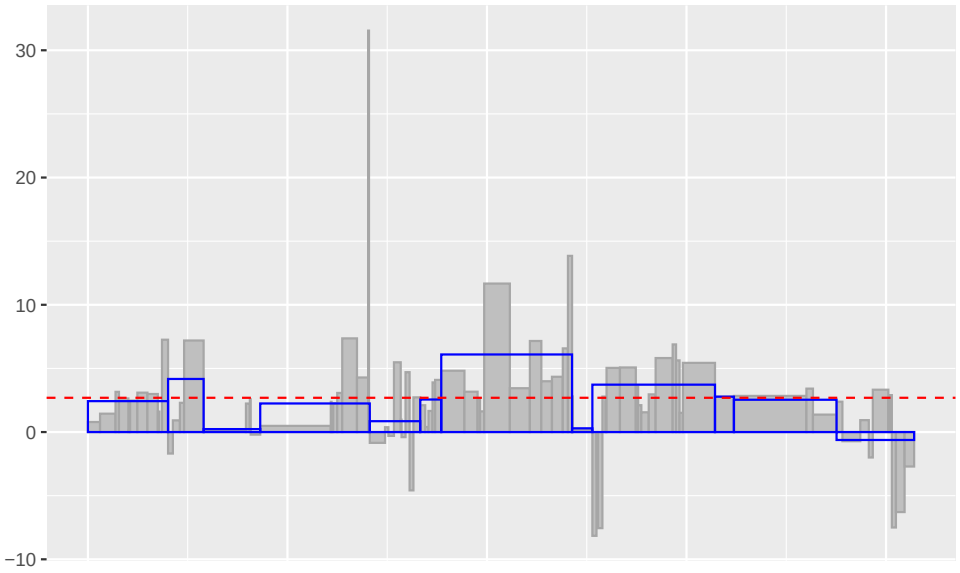



**Figure 1:** UK Consumer Price Index changes August 2017 to August 2018. The red dotted line shows the overall percentage change. The rectangles with blue borders show the changes of the main sectors. Heights are percentage changes, widths are index values in August 2017, so that the areas of the rectangles are each group contributions to the overall change. The grey bars show the percentage and absolute changes for the individual components.

Every country produces its own CPI and they have many different ways of displaying information on how their index changes between two time points. Some use barcharts (e.g., National Bureau of Statistics of China (2024) and US Bureau of Labor Statistics (2023)). France also has stacked bars (Banque de France (2017)). Germany includes a kaleidoscope plot (Statistisches Bundesamt (2024)). The UK includes multiple time series (Office for National Statistics (2024)) as well as different kinds of barchart. Most of these displays just show percentage changes. A few have accompanying displays of how big the sectors comprising the index are. What is needed is a display showing both individual percentage changes and their contributions to the change in the index (i.e., the individual absolute changes). This is what UpAndDown plots do.

In the UK's CPI of 2017 there were 12 main groups with Transport having the biggest weight (156 out of 1000) (ONS, 2019). All of these groups, except Education, were further subdivided into up to 7 subgroups and the subgroups were subdivided yet again into up to 9 individual components. Overall there were 85 individual items with weights ranging from 1 (e.g., solid fuels) to 86 (Restaurants & Cafes). Figure 1 is an UpAndDown plot of the CPI changes over one year plotted horizontally for comparison with the kind of standard barchart that is often used. Horizontal plots are useful for giving an overview of the data before deciding what to concentrate on in detail.

To draw the plot, first load the package with

```
library(UpAndDownPlots)
```

Prepare the data for plotting using the `ud_prep` function. `CPIuk` is the dataset, `weight` is the vector of individual component weightings, `v1` is the initial component value and `v2` the final one. Two levels are to be plotted, `Sector` and `Component`, both are to be left in their original order.

```
yp <- ud_prep(CPIuk, weight="Weight", v1="Aug2017", v2="Aug2018", levs=c("Sector",
"Component"), sortLev=c("orig", "orig"))
```

`ud_plot` takes the data and prepares the plots and data summaries. The parameter `drawFrom` specifies in which order the levels are plotted, in this case the lower `Component` level first and the higher `Sector` level on top.

```
yd <- ud_plot(yp, drawFrom="SmallToBig")
```

A horizontal unlabelled UpAndDown plot has been chosen.

```
yd$uad
```

It can be seen that only one sector declined, the one on the far right that turns out to be Miscellaneous goods and services, that one component (liquid fuels) had a much higher jump in price, over 30%, than any other, and that the components of sectors did not change uniformly. Plots showing only percentage changes would show neither the relative importance of the separate parts of the index nor that liquid fuels are only a small component of the index. They would emphasize neither the

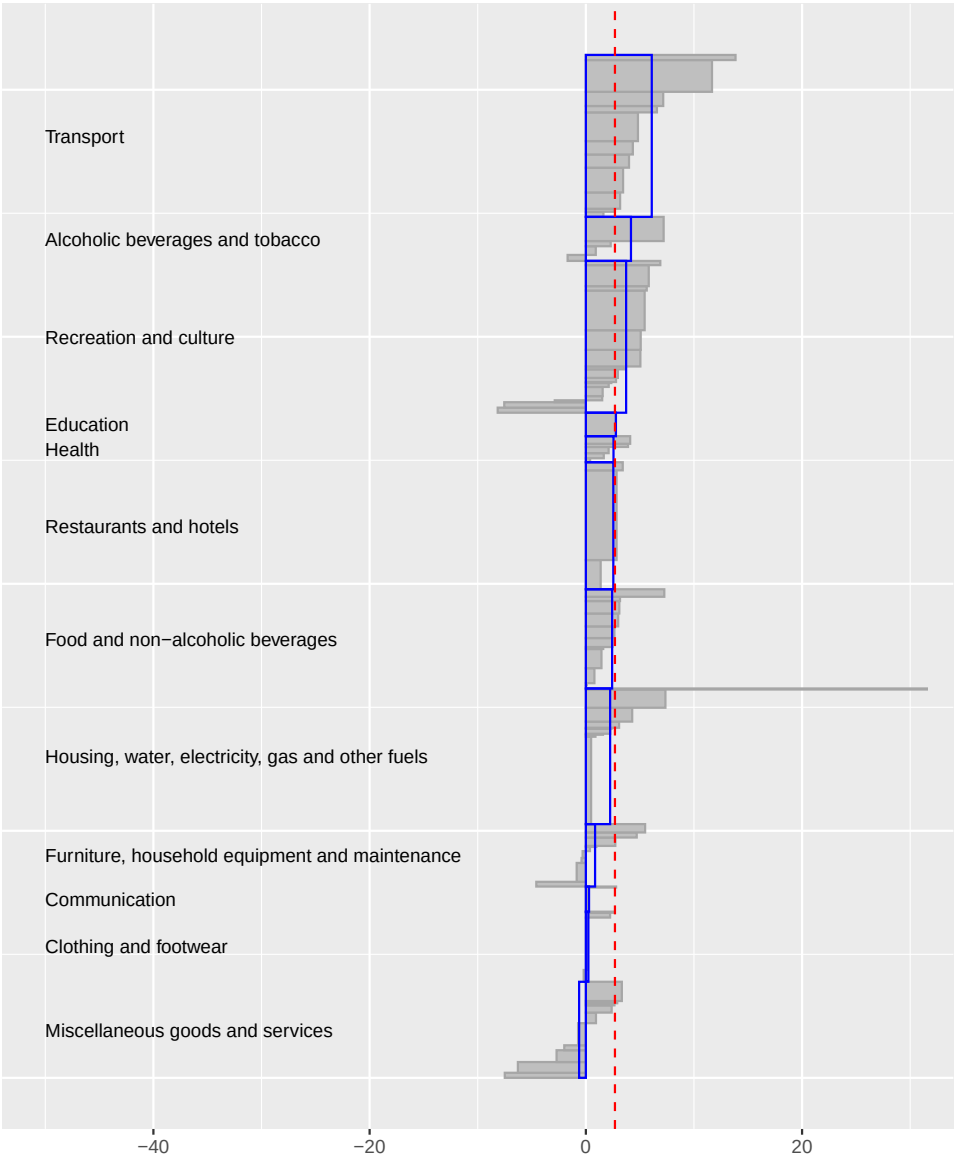



**Figure 2:** CPI changes ordered by percentage changes of sectors and by percentage changes of components within sectors. Two sectors, Transport and Recreation and Culture, made the biggest contributions to change. Prices increased for all components in most sectors.

hierarchical structure of the index nor the relative importance of components within sectors.

There are a number of ways UpAndDown plots can be varied and extended to make them more informative: the plot can be drawn vertically with labels for either sectors or their components; the sectors and their components can be sorted in various ways; colour can be employed to distinguish sectors. Figure 2 shows a version ordered by percentage changes of sectors and by percentage changes of components within the sectors. The plot has labels for the sectors and is drawn vertically to make those labels readable.

The preparatory code with `ud_prep` defines the sorting of the levels using the parameter `sortLev`.

```
yq <- ud_prep(CPIuk, weight="Weight", v1="Aug2017", v2="Aug2018", levs=c("Sector",
"Component"), sortLev=c("perc", "perc"))
```

Labels are added for the sectors using the parameter `labelvar`. Limits have been set for the percentage change axis to add sufficient space for the labels using the parameter `vscale`.

```
yf <- ud_plot(yq, labelvar="Sector", drawFrom="SmallToBig", vscale=c(-30, 30))
```

A vertical labelled UpAndDown plot has been chosen.

```
yf$uadl
```

There are 85 components at the lowest level of the CPI, too many to label effectively unless the plot is drawn very tall. Plots like Figure 2 (and indeed Fig 1) are drawn to give an overview to help

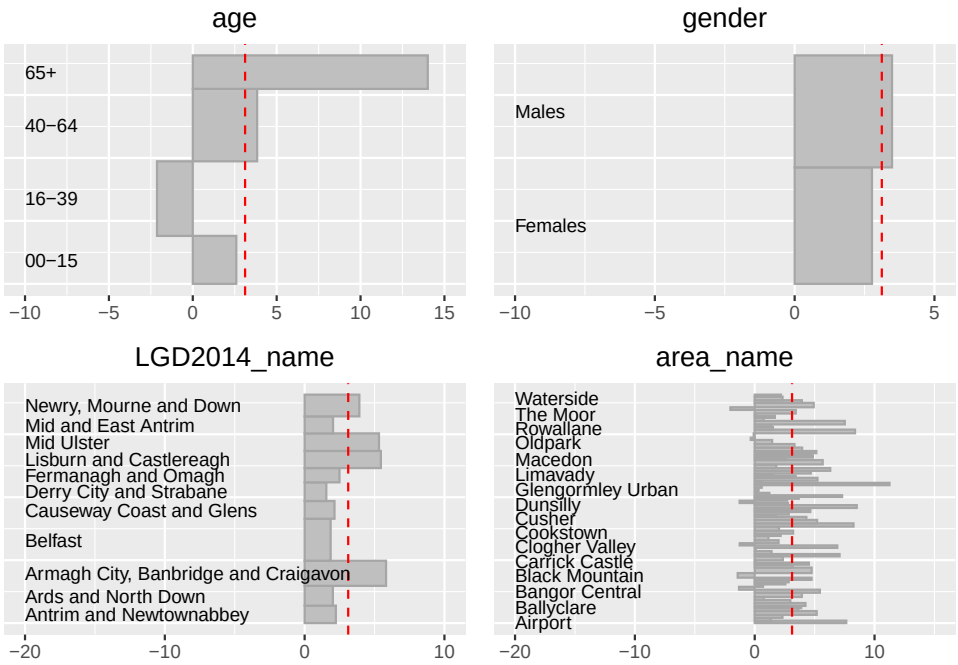



**Figure 3:** Percentage changes in population between 2011 and 2017 by age, gender, district, and area.

decide what final plot or plots to draw. Perhaps only changes at the `Sector` level might be displayed, perhaps changes for the `Component`level of one sector, say Recreation and culture, might be displayed.

## 7 Visualising population changes in Northern Ireland

National Statistics Offices study demographic changes to understand how their country's population is developing: which groups are growing or declining and how areas are changing. The `NIpop` data in the package come from the Northern Ireland Statistics and Research Agency and include population estimates for seven years. This section considers the changes between 2011 and 2017.

Figure 3 shows UpAndDown plots for each of the possible grouping variables. The code uses the `dgroup` function to produce two sets of UpAndDown plots for each of the individual classification variables specified. One set is labelled, the other not. The `grid.arrange` function from **gridExtra**(Auguie, 2017) is used to plot the graphics together in a single display.

Age groups displayed the most variation with a high percentage increase amongst the 65+ group and a decline in the 16-39 group. Male and female changes were similar. All the districts increased in population, while some of the 80 areas actually declined in population.

```
library(gridExtra)
g4 <- dgroup(NIpop, byvars=c("age", "gender", "LGD2014_name", "area_name"),
             v1="y2011", v2="y2017")
grid.arrange(g4$uadgl)
```

There are many alternatives using more than one classification variable. If you plot the Northern Ireland population data changes grouped first by `gender` and then by `age`, there will be two bars in the `gender` layer and eight in the `age` layer. If you plot the changes grouped first by `age` and then by `gender`, there will be four bars in the `age` layer and eight in the `gender` layer. In both plots the eight bars in the lowest layer will be the same, but they will be grouped differently. Table 1 shows the absolute and percentage changes for the data: firstly for `gender` and then `age` by `gender`; secondly for `age` and then `gender` by `age`. Some conclusions can be drawn from the tables by careful inspection. More can be seen with graphics.

Figure 4 is a graphic for age and gender together. It reveals that there was a much bigger percentage increase in numbers of older men than in numbers of older women (albeit from a lower base), that there was little difference between males and females in the youngest group, and that the decline in 16-39 year-olds was greater for females than for males.

```
ag <- ud_prep(NIpop, v1="y2011", v2="y2017", levs=c("age", "gender"),
```

**Table 1:** Absolute and percentage changes in N. Ireland population estimates by age and gender.

| gender | change | % change |
|---|---|---|
| Females | 25590 | 2.77 |
| Males | 30980 | 3.48 |

| age | change | % change |
|---|---|---|
| 00-15 | 9880 | 2.59 |
| 16-39 | -12630 | -2.13 |
| 40-64 | 22080 | 3.84 |
| 65+ | 37240 | 14.01 |

| gender | age | change | % change |
|---|---|---|---|
| Females | 00-15 | 4950 | 2.67 |
| Females | 16-39 | -8790 | -2.95 |
| Females | 40-64 | 12960 | 4.45 |
| Females | 65+ | 16470 | 10.97 |
| Males | 00-15 | 4930 | 2.52 |
| Males | 16-39 | -3840 | -1.31 |
| Males | 40-64 | 9120 | 3.21 |
| Males | 65+ | 20770 | 17.94 |

| age | gender | change | % change |
|---|---|---|---|
| 00-15 | Females | 4950 | 2.67 |
| 00-15 | Males | 4930 | 2.52 |
| 16-39 | Females | -8790 | -2.95 |
| 16-39 | Males | -3840 | -1.31 |
| 40-64 | Females | 12960 | 4.45 |
| 40-64 | Males | 9120 | 3.21 |
| 65+ | Females | 16470 | 10.97 |
| 65+ | Males | 20770 | 17.94 |

```
              sortLev=c("orig", "perc"))
kag <- ud_plot(ag, labelvar="age")
kag$uadl
```

These results may vary by the 11 districts and a plot of changes by district, age, and gender could be drawn. For the purposes of the article, three districts have been chosen: the cities of Belfast and Derry and the district of Newry, Mourne and Down. Figure 5 shows the plot. The two younger groups declined in population for both males and females in Derry and Strabane. In Belfast the number of male 16-39 year-olds actually increased, but the main difference to the other districts is the somewhat lower increase in the male 65+ group and the lack of any increase in the female 65+ group.

```
NIpopW <- NIpop %>% filter(LGD2014_name %in% c("Belfast", "Derry City and Strabane",
                "Newry, Mourne and Down"))
zdag <- ud_prep(NIpopW, v1="y2011", v2="y2017", levs=c("LGD2014_name", "age", "gender"),
                sortLev=c("perc", "orig", "perc"))
zdag2 <- ud_plot(zdag, labelvar="LGD2014_name", drawFrom="SmallToBig")
zdag2$uadl
```

In all the above three displays the levels have been sorted by percentage change, except for the age groups where, for obvious reasons, their original order has been retained.

## 8 Visualising brand share change in German car sales

An example of displaying brand share change in UpAndDown plots is shown in Figure 6 for the German car industry between 2017 and 2018. The **patchwork** package (Pedersen, 2024) is used to plot more than one graphic in a figure.

```
library(patchwork)
```

Only the compact market sector is to be displayed.

```
AutoSalesXcomp <- AutoSalesX %>% filter(Segment=="Compact")
```

The aim is to compare brand shares by `Manufacturer`, so that level is specified.

```
yxp <- ud_prep(AutoSalesXcomp, v1="sales17", v2="sales18", levs=c("Manufacturer"),
sortLev=c("perc"))
```

The `ud_plot` function prepares graphics for changes in sales.

```
yM <- ud_plot(yxp, labelvar="Manufacturer")
```

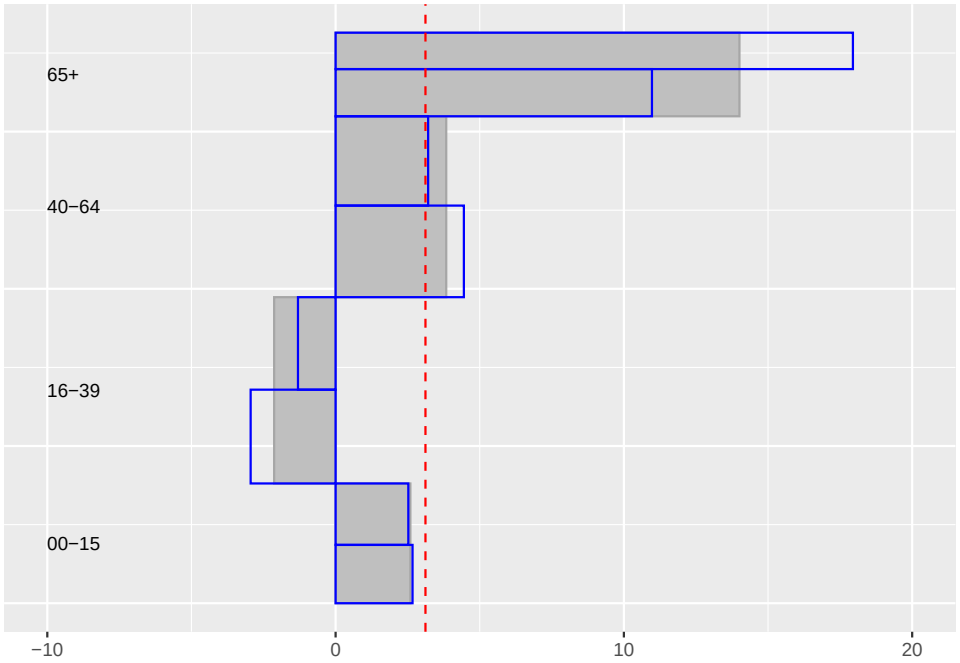



**Figure 4:** Percentage changes in population between 2011 and 2017 by age and then gender (males above, females below).

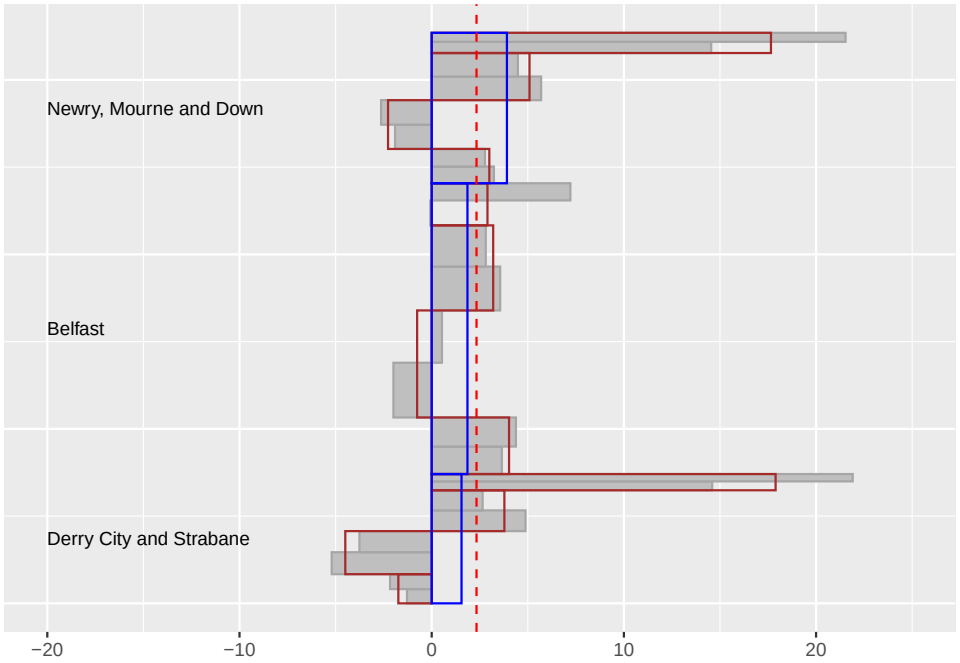



**Figure 5:** Percentage changes in population between 2011 and 2017 for three districts (changes outlined in blue), four age groups (outlined in brown), and gender (filled bars).

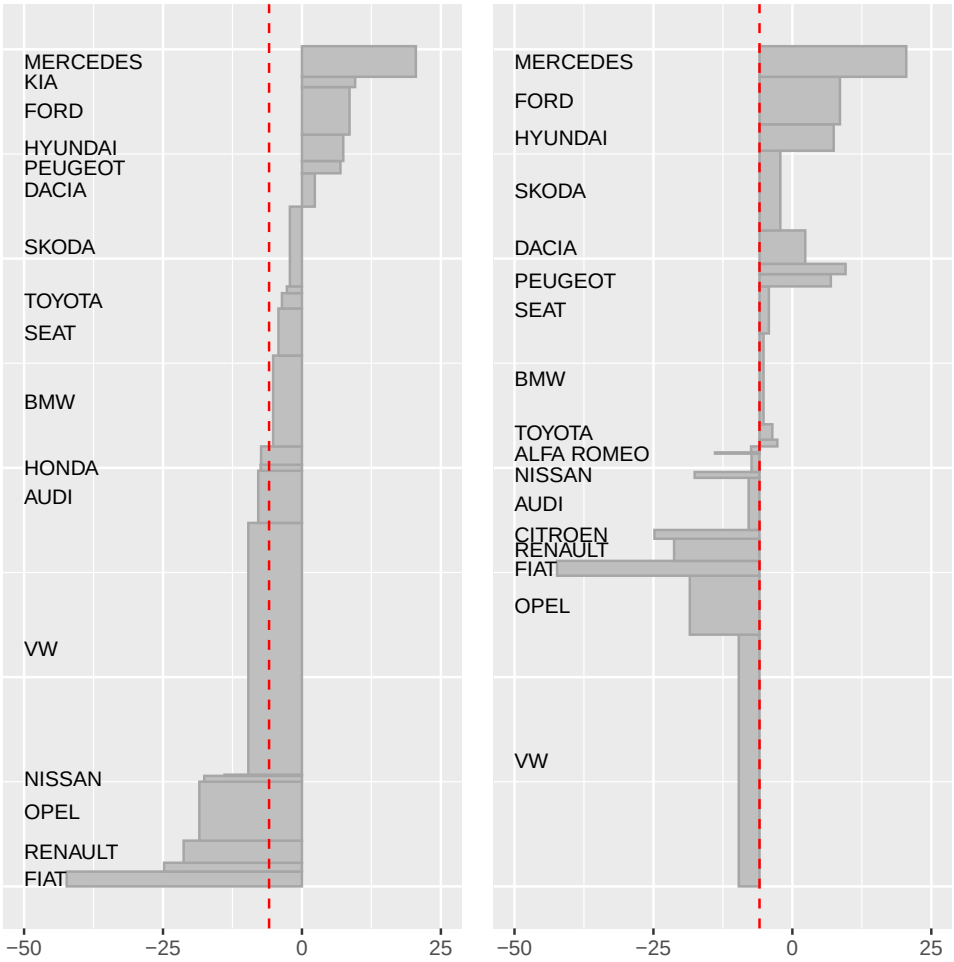



**Figure 6:** Compact car sales changes in Germany by manufacturer between 2017 and 2018. The plot on the left is drawn with a baseline of zero change and ordered by percentage changes. The plot on the right is drawn with a baseline of the overall market change and ordered by changes in market share.

The next code snippet calculates brand shares by manufacturer and the changes in them between the two years. It orders the data by these changes in brand share.

```
AutoSalesXcomp <- AutoSalesXcomp %>% mutate(S17=sum(sales17),    S18=sum(sales18),
p17=100*sales17/S17, p18=100*sales18/S18, msch=p18-p17) AutoSalesXcompS <- AutoSalesXcomp
%>% arrange(msch)
```

The brand share data are prepared in the usual way but using the b parameter to set the baseline value at the overall percentage change in the market.

```
yxs <- ud_prep(AutoSalesXcompS, v1="sales17", v2="sales18", levs=c("Manufacturer"),
sortLev=c("orig")) yS <- ud_plot(yxs, b=yM$TotPerc, labelvar="Manufacturer")
```

The two labelled graphics are drawn side by side using **patchwork**.

```
yM$uadl + yS$uadl
```

Vehicles selling less than 1000 in both years have been reclassified as `Other` and the data have been aggregated by manufacturer. To avoid overlapping, not all manufacturers are labelled. The plot on the left uses a baseline of 0 and shows that the whole segment declined. It has been ordered by bar height, i.e. percentage changes. Of the bigger manufacturers, Mercedes and Ford had year-on-year increases. The plot on the right uses a baseline of the overall market change, a little under −5.9%. It has been ordered by bar area, using the areas proportional to market share changes. The calculations were carried out in advance and the manufacturers sorted accordingly, as can be seen in the code.

Mercedes and Ford gained the most market share and VW and Opel lost the most. Dacia gained in sales and market share, while Skoda lost sales, but still gained more market share than Dacia because they had a bigger starting value and performed better than the market. It would be difficult to compare the bar area sizes in the right plot, so ordering is essential (if those are the comparisons you want to make).

Area conservation holds for share changes as well. If a producer $i$ sells $k_i$ different products, then the sum of their bar areas using total market percentage change as the baseline equals the equivalent area for producer $i$ as a whole.

Graphical displays are always relative. The visual representation of a statistic is proportional to the value of the statistic. Comparisons should only be made between lengths or areas that are on the same scales. The widths of bars in UpAndDown plots are always proportional to initial values (sales, market shares, volumes etc.), while the heights are proportional to percentages representing relative changes, and the areas are proportional to amounts representing absolute changes (if the baseline is 0) or share changes (if the base line is the total percentage change).

## 9 Summary

Displaying percentage and absolute changes between two time points in the same plot is very helpful, especially when individual components are of quite different sizes. Being able to drill down (or aggregate up) through multiple levels thanks to area conservation makes UpAndDown plots a powerful descriptive and exploratory tool. For the UK CPI dataset they showed that one sector actually declined in price and that individual components in other sectors also did. They showed that `Transport` and `Recreation and Culture` were the biggest drivers of inflation. The UpAndDown plots for the example of Northern Ireland population data showed the declines in the numbers of younger people and increases in the numbers of older ones, the increases in population across all districts, the different age patterns for males and females, and the decline in numbers of young people in the Derry Local Government District. The final example of the German car sales data showed that UpAndDown plots can be used to display brand share changes too. The conclusions were that although the overall market declined, two of the biggest manufacturers increased their sales. They also showed that the biggest loss in market share was for a manufacturer with a medium percentage loss of sales, because that manufacturer started from a high initial level of sales.

The display of market share changes in an UpAndDown plot is a surprising and powerful additional feature. In principle it would be possible to draw all three changes (percentage, absolute, and share) in the same plot. Be that as it may, it would not be a good idea. What is a good idea is drawing a separate UpAndDown plot for the market share changes, when this is the kind of change of most interest. The fact that the relevant bar areas also have the property of area conservation underlines the strength of the basic concept of UpAndDown plots.

As graphics become more complex, tools for adjusting and adapting them become more important. UpAndDown plots allow the flexible ordering of classifying levels for non-hierarchical data, multiple sorting methods for the elements of individual levels, and a rearranging of the order of drawing levels, all to make the displays more informative.

Future goals include developing related displays for multiple time points and adding interactivity. Studying an UpAndDown plot would be easier if it were interactive. Querying, changing the order of levels, sorting in different ways, zooming, linking to other graphics, would all be valuable tools (as they would be for every kind of graphic).

### Acknowledgements

Thanks to Bill Venables for coding assistance and to Nick Cox for pointing out two older references.

## References

B. Auguie. *gridExtra: Miscellaneous Functions for "Grid" Graphics*, 2017. URL https://CRAN.R-project.org/package=gridExtra. R package version 2.3. [p188]

Banque de France. Services prices in france and oil prices, 2017. URL https://www.bls.gov/opub/btn/volume-13/a-year-in-review-exploring-consumer-price-trends-in-2023.htm. (accessed 13.11.2024). [p186]

W. C. Brinton. *Graphic Presentation*. Brinton Associates, New York, 1939. URL https://archive.org/details/graphicpresentat00brinrich. [p180]

W. S. Cleveland and R. McGill. Graphical perception: The visual decoding of quantitative information on graphical displays of data. *Journal of the Royal Statistical Society A*, 150(3):192–229, 1987. URL https://doi.org/10.2307/2981473. [p179]

H. Hofmann. Constructing and reading mosaicplots. *Computational Statistics & Data Analysis*, 43(4): 565–580, 2003. URL https://doi.org/10.1016/S0167-9473(02)00293-1. [p179]

H. Hofmann. Mosaic plots and their variants. In C. Chen, W. Haerdle, and A. Unwin, editors, *Handbook of Data Visualization*, pages 617–642. Springer, 2008. URL https://doi.org/10.1007/978-3-540-33037-0_24. [p180]

K. Karsten. *Charts and Graphs: An Introduction to Graphic Methods in the Control and Analysis of Statistics*. Prentice-Hall, Cambridge, 1923. URL https://archive.org/details/chartsgraphsintr0000karl/page/n7/mode/2up. [p179]

D. MacKay. *Sustainable Energy — without the hot air*. UIT, Cambridge, 2009. URL https://www.withouthotair.com. [p180]

National Bureau of Statistics of China. Consumer price index for may 2024, 2024. URL https://www.stats.gov.cn/english/PressRelease/202406/t20240625_1955163.html. (accessed 13.11.2024). [p186]

NISRA. 2017 mid-year population estimates for district electoral areas, 2019. URL https://www.nisra.gov.uk/publications/2017-mid-year-population-estimates-district-electoral-areas. [p181]

Office for National Statistics. Consumer price inflation, uk: September 2024, 2024. URL https://www.ons.gov.uk/economy/inflationandpriceindices/bulletins/consumerpriceinflation/september2024. (accessed 13.11.2024). [p186]

ONS. Consumer price indices technical manual, 2014. URL https://doc.ukdataservice.ac.uk/doc/7022/mrdoc/pdf/7022_technical_manual_2014.pdf. [p180]

ONS. Consumer price inflation tables, 2019. URL https://www.ons.gov.uk/economy/inflationandpriceindices/datasets/consumerpriceinflation/current. (accessed 18.03.2019). [p186]

T. Pedersen. *patchwork: The Composer of Plots*, 2024. URL https://CRAN.R-project.org/package=patchwork. R package version 1.3.0. [p189]

O. Pybus, A. Rambaut, and P. Harvey. An integrated framework for the inference of viral population history from reconstructed genealogies. *Genetics*, 155:1429–1437, 2000. URL https://doi.org/10.1093/genetics/155.3.1429. [p180]

B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11 (1):92–99, 1992. URL https://doi.org/10.1145/102377.115768. [p179]

Statistisches Bundesamt. Price kaleidoscope, 2024. URL https://service.destatis.de/Voronoi/PriceKaleidoscope.svg. (accessed 13.11.2024). [p186]

M. Tennekes. *treemap*, 2023. URL https://CRAN.R-project.org/package=treemap. R package version 2.4-4. [p180]

A. Unwin. *Getting (more out of) Graphics*. Chapman & Hall/CRC, Boca Raton, Florida, 2024a. URL https://doi.org/10.1201/9781003131212. [p185]

A. Unwin. *UpAndDownPlots: Displays Percentage and Absolute Changes*, 2024b. URL https://CRAN.R-project.org/package=UpAndDownPlots. R package version 0.5.0. [p183]

US Bureau of Labor Statistics. BEYOND THE NUMBERS, 2023. URL https://www.bls.gov/opub/btn/volume-13/a-year-in-review-exploring-consumer-price-trends-in-2023.htm. (accessed 13.11.2024). [p186]

H. Wickham. ggplot2, 2022. URL https://ggplot2-book.org. (accessed 01.08.2022). [p184]

*Antony Unwin*
*University of Augsburg*
*Mathematics Institute*
*University of Augsburg*
*86135 Augsburg, Germany*
unwin@math.uni-augsburg.de

# Bioconductor Notes, December 2024

*by Maria Doyle, Bioconductor Community Manager, and Bioconductor Core Developer Team*

## 1 Introduction

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. The project has entered its twentieth year, with funding for core development and infrastructure maintenance secured through 2025 (NIH NHGRI 2U24HG004059). Additional support is provided by NIH NCI, Chan-Zuckerberg Initiative, National Science Foundation, Microsoft, and Amazon. In this news report, we give some updates on core team and project activities.

## 2 Software

Bioconductor 3.20, released in October 2024, is now available. It is compatible with R 4.4 and consists of 2289 software packages, 431 experiment data packages, 928 up-to-date annotation packages, 30 workflows, and 5 books. Books are built regularly from source, ensuring full reproducibility; an example is the community-developed Orchestrating Single-Cell Analysis with Bioconductor.

## 3 Core Team and Infrastructure Updates

Of note:

In GenomicRanges, reference build hg38 is now used in vignettes and examples.

GEOquery now supports retrieval of RNASeq quantification data prepared by NCBI, as well as search.

HDF5Array sports many improvements to coercions from CSC_H5SparseMatrixSeed, H5SparseMatrix, TENxMatrix, or H5ADMatrix to SparseArray:

these should be significantly more efficient, thanks to various tweaks that happened in the SparseArray and Delayed5Array packages.

HDF5Array will now support coercing an object with more than 2^31 nonzero values, and oercion from any of the classes above to a sparseMatrix derivative now fails early if object to coerce has >= 2^31 nonzero values.

All *Seed classes in the package now extend the new OutOfMemoryObject class defined in BiocGenerics (virtual class with no slots).

See the NEWS section in the release announcement for a complete account of changes throughout the ecosystem.

## 4 Community and Impact

### 4.1 Publications and Preprints

In October 2024, two notable preprints were published by Bioconductor contributors:

- Eleven quick tips for writing a Bioconductor package Authors: Charlotte Soneson, Lori Shepherd, Marcel Ramos, Kevin Rue-Albrecht, Johannes Rainer, Hervé Pagès, and Vincent J. Carey (Bioconductor Core Team and collaborators). This preprint provides concise, practical advice for developing high-quality Bioconductor packages.

- Learning and teaching biological data science in the Bioconductor community Authors: Bioconductor Training Committee and others. This preprint explores the Bioconductor community's role in advancing biological data science education and training.

### 4.2 ELIXIR BioHackathon Europe 2024

Bioconductor participated in this collaborative event (November 4–8), held in Barcelona, to advance interoperability and innovation in life sciences. Key contributions included projects focused on EDAM-bio.tools and training infrastructure.

### 4.3 Training Domain Launch

Bioconductor is developing training.bioconductor.org, a centralised website for accessing and organising high-quality Bioconductor training materials. This work is still in progress, with the Training Committee actively discussing ways to restructure existing resources and improve their discoverability. A key focus is supporting community-driven tutorials and emphasising pedagogical quality to better serve learners.

### 4.4 Bioconductor Carpentry Single-Cell Module

Bioconductor now features an official single-cell RNA sequencing (scRNA-seq) module in the Carpentries Incubator. This module, renamed from the Orchestrating Single-Cell Analysis (OSCA) curriculum, provides a structured introduction to scRNA-seq analysis with Bioconductor and is aimed at both learners and educators.

### 4.5 Bioconductor Joins Bluesky

In October 2024, Bioconductor expanded its social media presence by joining Bluesky. This move reflects Bioconductor's commitment to engaging with its global community across diverse platforms. We invite you to follow us to stay updated on the latest news, events, and community highlights.

## 5 Conferences

### 5.1 BioCAsia2024

Held in Sydney, Australia (November 7–8), the Bioconductor Asia meeting fostered collaboration and knowledge sharing among regional and global Bioconductor users and developers. For more details, visit the BioCAsia2024 website.

### 5.2 Galaxy and Bioconductor Community Conference

Announced in September 2024, the first Galaxy and Bioconductor Community Conference (GBCC 2025) is scheduled for June 23–26, 2025, at Cold Spring Harbor Laboratory, New York. The conference aims to showcase the integration of Galaxy and Bioconductor tools, promoting reproducible workflows and community-driven development. For more details, see the blog post.

## 6 Boards and Working Groups Updates

### 6.1 New Board Members

- The Community Advisory Board (CAB) welcomes new members Zahraa Alsafwani, Jasmine Daly, Tobilola Ogunbowale, and Lluis Revilla. We extend our gratitude to

outgoing members Daniela Cassol, Estefania Mancini, Jordana Muwanguzi, and Mike Smith for their service.

- The Technical Advisory Board (TAB) welcomes new members Michael Lawrence and Jacques Serizay. Outgoing members Sean Davis and Michael Love are also warmly thanked for their contributions.

### 6.2 Code of Conduct Committee Elections

In October 2024, Bioconductor sought new members to join its Code of Conduct (CoC) Committee, which plays a vital role in maintaining the integrity and safety of our community. The CoC meets 3–4 times annually to address important issues. Applications are now closed, but you can learn more about the committee and its work on the Code of Conduct page.

If you are interested in becoming involved with any Bioconductor working group please contact the group leader(s).

## 7 Using Bioconductor

Start using Bioconductor by installing the most recent version of R and evaluating the commands

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install()
```

Install additional packages and dependencies, e.g., SingleCellExperiment, with

```
BiocManager::install("SingleCellExperiment")
```

Docker images provides a very effective on-ramp for power users to rapidly obtain access to standardized and scalable computing environments. Key resources include:

- bioconductor.org to install, learn, use, and develop Bioconductor packages.
- A list of available software linking to pages describing each package.
- A question-and-answer style user support site and developer-oriented mailing list.
- A community slack workspace (sign up) for extended technical discussion.
- The F1000Research Bioconductor gateway for peer-reviewed Bioconductor workflows as well as conference contributions.
- The Bioconductor YouTube channel includes recordings of keynote and talks from recent conferences, in addition to video recordings of training courses.
- Our package submission repository for open technical review of new packages.

Upcoming and recently completed events are browsable at our events page.

The Technical and and Community Advisory Boards provide guidance to ensure that the project addresses leading-edge biological problems with advanced technical approaches, and adopts practices (such as a project-wide Code of Conduct) that encourages all to participate. We look forward to welcoming you!

We welcome your feedback on these updates and invite you to connect with us through the Bioconductor Slack workspace or by emailing community@bioconductor.org.

*Maria Doyle, Bioconductor Community Manager*
*University of Limerick*

*Bioconductor Core Developer Team*
*Dana-Farber Cancer Institute, Roswell Park Comprehensive Cancer Center, City University of New York, Fred Hutchinson Cancer Research Center, Mass General Brigham*

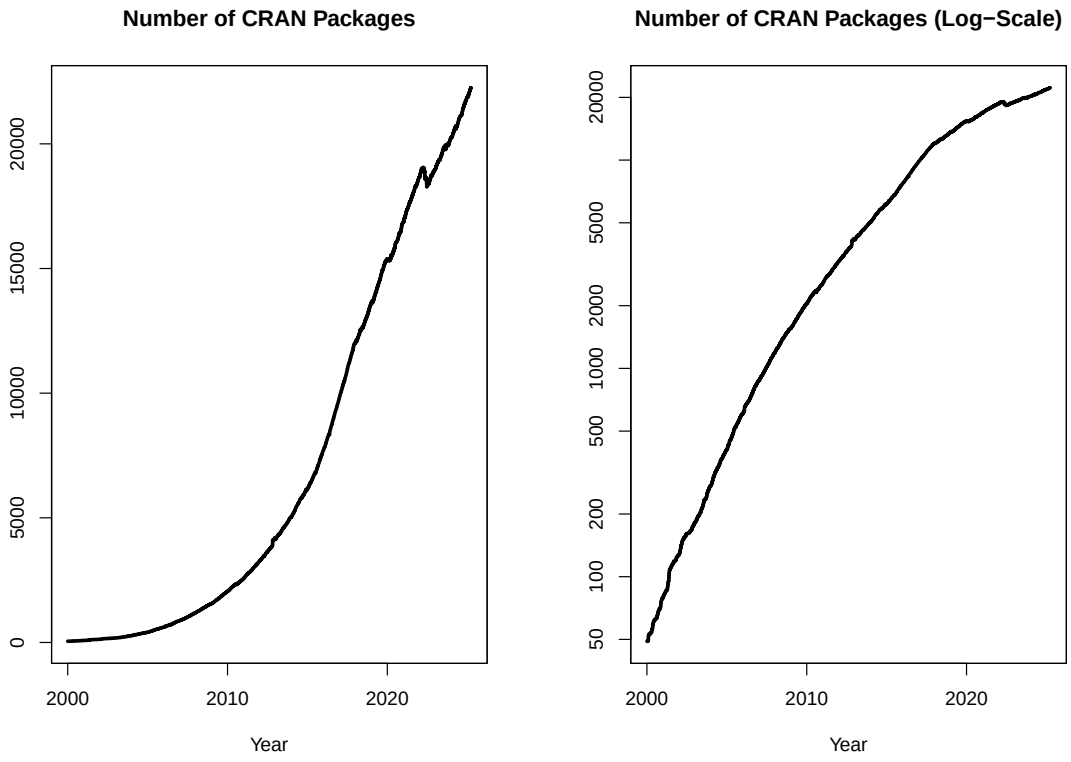
# Changes on CRAN

**2024-10-01 to 2024-12-31**

*by Kurt Hornik, Uwe Ligges, and Achim Zeileis*

## 1 CRAN growth

In the past 3 months, 504 new packages were added to the CRAN package repository. 104 packages were unarchived, 173 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 2024-12-31, the number of active packages was around 21860.

## 2 CRAN package submissions

From oktober 2024 to december 2024 CRAN received 6179 package submissions. For these, 9795 actions took place of which 7408 (76%) were auto processed actions and 2387 (24%) 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 | 2129 | 490 | 1403 | 96 | 0 | 1991 | 668 | 290 |
| manual | 986 | 0 | 8 | 3 | 27 | 1072 | 226 | 58 |

These include the final decisions for the submissions which were

| | archive | publish |
|---|---|---|
| auto | 2056 (33.7%) | 1805 (29.6%) |
| manual | 982 (16.1%) | 1254 (20.6%) |

where we only count those as *auto* processed whose publication or rejection happened automatically in all steps.

## 3 CRAN mirror security

Currently, there are 94 official CRAN mirrors, 73 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:

- Paleontology: Maintained by William Gearty, Lewis A. Jones, Erin Dillon, Pedro Godoy, Harriet Drage, Christopher Dean, Bruna Farina.

Currently there are 48 task views (see `https://CRAN.R-project.org/web/views/`), with median and mean numbers of CRAN packages covered 110 and 121, respectively. Overall, these task views cover 4885 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`

# News from the Forwards Taskforce

*by Heather Turner*

**Abstract** Forwards is an R Foundation taskforce working to widen the participation of under-represented groups in the R project and in related activities, such as the *useR!* conference. This report rounds up activities of the taskforce during 2024.

### 0.1 Accessibility

Di Cook supervised Krisanat Anukarnsakulchularp on an R Consortium funded project to facilitate adding alt text to figures in R Journal articles. Jonathan Godfrey and Heather Turner were advisors on the project. The main outcomes are:

- Some Guidance for Writing alt text for Data Plots
- A prototype Shiny app for authors to add or review alt text for a published article.
- Tests of using LLM services to produce draft alt text for authors to review.

A post on the project is planned for the R Consortium Blog.

Heather Turner, along with Abhishek Ulayil, co-mentored Yinxiang Huang on the Google Summer of Code (GSoC) 2024 project Converting Sweave to R Markdown using the texor package. The motivating application was converting package vignettes from Sweave/PDF to R Markdown/HTML for improved accessibility. This project resulted in the new function `rnw_to_rmd()`, added to **texor** in version 1.4.

Liz Hare was selected for the 2024 rOpenSci Champions program. She wrote a blog post on Resources For Using R With Screen Readers and along with her fellow champion Alican Cagri Gokcek, she organized a multilingual webinar on the same topic. Liz also co-hosted a Turing Way Fireside Chat on How to make things more Accessible in Data Science, considering accessibility in broad sense, not just in terms of disability.

### 0.2 Community engagement

Kevin O'Brien, gwynn gebeyhu and Heather Turner gave virtual talks at the hybrid Ghana R Conference 2024, Kumasi, June 6–7, 2024. Kevin and gwynn have joined the Ghana R Users Community as advisory committee members, as they plan for Ghana R Conference 2025.

gwynn also gave a virtual talk to R-Ladies Gaborone on R in Research.

Kevin attended the inaugural Open Source Community Collider organized by Open Source Science (OSS), which was designed to build connections and drive collaboration across the open source and open science communities.

As part of the *DiveRSE* seminar series, Ella Kaye had a conversation with Malvika Sharan about their experiences in the Research Software Engineering community, and the challenges and successes they've encountered when working to improve equity, diversity, inclusion and accessibility (EDIA).

Ella led the development of new branding for rainbowR, ready for promoting the community more widely. She presented lightning talks on the community at useR! 2024 (with co-lead Hanne Oberman), posit::conf(2024) (video), and RPySOC24, the conference run by the open source community of the National Health Service in the UK. rainbowR also increased their presence on social media with a LinkedIn group and a Bluesky account. Ella has been selected for the Software Sustainability Institute's 2025 Fellowship programme, which will enable her to further establish the rainbowR community in 2025.

### 0.3 R Contribution/on-ramps

Following on from R Project Sprint 2023, the R Contribution Working Group arranged a series of R Dev Days as satellites to R-related conferences, involving many Forwards members (Heather Turner, gwynn gebeyehu, Ella Kaye, Saranjeet Kaur Bhogal, Mike Chirico, Michael Lawrence, Paola Corrales and Di Cook). The R Dev Day format takes advantage of people already travelling to a common venue, so the events could be run at low cost and we found that it was possible for both novice and experienced participants to make meaningful contributions in the shorter time. For more details see the reports on the satellite events to London satRday 2024, useR! 2024, posit::conf(2024), RSECon24, Shiny in Production 2024 and LatinR 2024 (virtual). In summary, attendance at the events ranged from 9 to 44 participants; the number of issues worked on at each event was usually around half the number

of participants, with around half of these resulting in a patch or closing a bug during, or shortly after, the event. The remaining issues were generally works in progress, where participants were able to contribute to addressing the issue. Given the success of these events, we plan to run repeats in 2025 and expand to other regions/communities.

Forwards members also promoted the work of the R Contribution group in conference talks: Heather Turner presented The R Contribution Working Group at useR! 2024 and Contributing to the R Project at posit::conf(2024); Saranjeet Kaur Bhogal gave talks on Enhancing the R Dev Guide at useR! 2024 and Empowering New Contributors: The Evolving Role of the R Development Guide at PyData Global 2024, while Ella Kaye presented C for R Users at useR! 2024. Ella's talk inspired participants to take on a C-based issue at the satellite R Dev Day.

In the first half of 2024, Ella Kaye and Heather Turner facilitated a C Study Group for existing or aspiring contributors wanting to learn or refresh their basic knowledge of C. In contrast to the C Book Club that was run in 2023, the study group used open materials from the C sessions of CS50, Harvard University's Introduction to Computer Science course, supplemented with material from the Deep R Programming book by Marek Gagolewski. Participants of the group have gone on to tackle C-related bugs in R. Ella plans to run the study group again in 2025, see the C Study Group 2025 page on the R Contributors website for further details.

Heather Turner and gwynn gebeyhu were on the steering committee, along with Bettina Grün, for an R Consortium funded project developing the CRAN Cookbook. This online book, written by Jasmine Daly and Beni Altmann, provides a user-friendly guide for solving issues that are common found in packages submitted to CRAN. While this is of benefit to all package maintainers, it should particularly help new package authors make their package CRAN-ready.

Saranjeet Kaur Bhogal visited Heather Turner at Warwick University for a week-long *Book Dash* on the R Development Guide in December 2024. This provided the opportunity to review and merge in long-standing pull requests (PRs) that had been opened during Saranjeet's work on the guide under Google Season of Docs 2022, as well as more recent PRs opened during R Dev Days. We were pleased that other members of the community joined in remotely, see the Forwards blog post for a short report.

### 0.4 Conferences

Several members of Forwards joined in with a *DEI Huddle*, where community members engaged with useR! 2024 organizers on diversity, equality and inclusion matters. The organizers confirmed that diversity scholarships would be available and following the huddle they were able to arrange limited childcare for participants. Heather Turner, along with Abhishek Ulayil, participated in one of the online *Fireside chats* in the run-up to the conference, on the theme of Building foundations for R's future as an accessible and diverse collaboration.

Kevin O'Brien organised a short R track for PyData Global, which took place virtually, December 3-5, 2024. Videos should appear on the PyData YouTube channel in due course. PyData events provide a forum for data scientists using languages including Python, Julia and R. The R track at PyData Global was intended to encourage more involvement of R users in this community.

### 0.5 Social Media

Since Bluesky has gained popularity in recent months, we are now bridging our Mastodon account to Bluesky: R-Forwards.hachyderm.io.ap.brid.gy. We encourage Mastodon and Bluesky users that do not have an account on both platforms to consider bridging with Bridgy Fed to enable greater interaction between R users on both platforms, e.g. following and mentioning.

We also created a Forwards page on LinkedIn enabling LinkedIn users to follow or mention us. LinkedIn is now home to a very vibrant R community and Kevin O'Brien has been helping to signal boost the growing number of R community pages via the R User Community LinkedIn page, the followership of which grew from 16400 to 19000 followers during 2024.

### 0.6 Changes in Membership

#### Previous members

The following members have stepped down: Isabella Gollini, Liz Hare. We thank them for their contributions to the taskforce over several years.

*Heather Turner*

*University of Warwick*
*Coventry, United Kingdom*
https://warwick.ac.uk/heatherturner
*ORCiD: 0000-0002-1256-3375*
heather.turner@r-project.org