Stability Selection and Consensus Clustering in R: The R Package sharp

Barbara Bodinier
Imperial College London

Sabrina Rodrigues
Imperial College London

Maryam Karimi
INSERM

Sarah Filippi
Imperial College London

Julien Chiquet
University Paris-Saclay, AgroParisTech, INRAE

Marc Chadeau-Hyam
Imperial College London

Abstract

The R package **sharp** (Stability-enHanced Approaches using Resampling Procedures) provides an integrated framework for stability-enhanced variable selection, graphical modelling and clustering. In stability selection, a feature selection algorithm is combined with a resampling technique to estimate feature selection probabilities. Features with selection proportions above a threshold are considered stably selected. Similarly, a clustering algorithm is applied on multiple subsamples of items to compute co-membership proportions in consensus clustering. The consensus clusters are obtained by clustering using co-membership proportions as a measure of similarity. We calibrate the hyper-parameters of stability selection (or consensus clustering) jointly by maximising a consensus score calculated under the null hypothesis of equiprobability of selection (or co-membership), which characterises instability. The package offers flexibility in the modelling, includes diagnostic and visualisation tools, and allows for parallelisation.

Keywords: stability selection, consensus clustering, calibration, regularisation, variable selection, graphical modelling, structural equation modelling, R.

1. Introduction

With the emergence of high-resolution data in health, environmental and social sciences, there is a need for efficient statistical methods to extract relevant features and provide interpretable results. These include unsupervised methods for descriptive analyses and supervised methods to investigate associations between predictors (explanatory variables) and outcomes (response variables). We focus here on unsupervised models including (i) clustering, (ii) graphical modelling, and (iii) dimensionality reduction with Principal Component Analysis (PCA), as well as supervised approaches including (i) classification, (ii) regression, (iii) Structural Equation Modelling (SEM), and (iv) Partial Least Squares (PLS) (Chadeau-Hyam, Campanella, Jombart, Bottolo, Portengen, Vineis, Liquet, and Vermeulen 2013).

We use regularisation procedures to induce sparsity in the sets of (i) edges (graphical models, see Meinshausen and Bühlmann 2006; Friedman, Hastie, and Tibshirani 2007; Bodinier, Filippi, Nost, Chiquet, and Chadeau-Hyam 2023a), (ii) variables contributing to a latent

variable (PCA or PLS, Zou, Hastie, and Tibshirani 2006; Shen and Huang 2008; Liquet, de Micheaux, Hejblum, and Thiébaut 2015), (iii) predictors contributing to the definition of the outcome (regression, Tibshirani 1996), or (iv) arrows of a directed acyclic graph corresponding to path coefficients (SEM, Jacobucci, Grimm, and McArdle 2016). The level of sparsity in these regularised models is controlled by a penalty parameter λ that needs to be calibrated. For simplicity, we focus throughout this paper on LASSO regularisation, which is controlled by a single penalty parameter λ. The approach scales to more complex penalties relying on multiple hyper-parameters (e.g. Elastic Net, Zou and Hastie 2005).

To generate reproducible results, penalised models can be complemented by stability approaches. In stability selection, a feature selection model is combined with resampling procedures to compute feature selection proportions, defined as the proportion of models fitted on different samples with the same hyper-parameter(s) in which a given feature is selected (Meinshausen and Bühlmann 2010). Stably selected features are then defined as those with selection proportions above a threshold $\pi \in]0, 1[$. Stability selection results are less likely to be driven by outlying observations and the selection proportion can be interpreted as a measure of feature importance, conditionally on all other important features.

Similarly, consensus clustering enables the identification of stable partitions by applying a clustering algorithm on multiple subsamples of the items (Monti, Tamayo, Mesirov, and Golub 2003). The co-membership proportions calculated over the subsamples are stored in a consensus matrix, which is then used as a measure of similarity. We implemented consensus clustering combined with hierarchical clustering, Partitioning Around Medoids (PAM), K means, Gaussian Mixture Models (GMM), or Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Bodinier, Vuckovic, Rodrigues, Filippi, Chiquet, and Chadeau-Hyam 2023b). For all these algorithms, the number of clusters is determined by a single hyper-parameter. Distance-based clustering algorithms (hierarchical clustering, PAM and DBSCAN) can also be applied on weighted distances calculated using the Clustering Objects on Subsets of Attributes (COSA) algorithm, which introduces a second hyper-parameter λ for the regularisation (Friedman and Meulman 2004; Bodinier *et al.* 2023b).

Both stability selection and consensus clustering remain under-used, partly due to the difficult choice of hyper-parameters. We previously proposed to calibrate the hyper-parameter(s) of stability selection and consensus clustering by maximising a consensus score measuring stability over the subsamples Bodinier *et al.* (2023a,b). Unlike error-based calibration techniques, the consensus score is computed from the sets of results obtained across the resampling iterations and can be used for the calibration of supervised or unsupervised models.

Simulation studies showed that these stability approaches generate a substantial increase in selection or clustering performance compared to a single run of the algorithm, and that our consensus score is relevant for the calibration of hyper-parameters (Bodinier *et al.* 2023a,b). Real-world applications also provided novel and interpretable results in different settings (Elliott, Whitaker, Bodinier, Eales, Riley, Ward, Cooke, Darzi, Chadeau-Hyam, and Elliott 2021; Whitaker, Elliott, Bodinier, Barclay, Ward, Cooke, Donnelly, Chadeau-Hyam, and Elliott 2022; Petrovic, Bodinier, Dagnino, Whitaker, Karimi, Campanella, Haugdahl Nøst, Polidoro, Palli, Krogh, Tumino, Sacerdote, Panico, Lund, Dugué, Giles, Severi, Southey, Vineis, Stringhini, Bochud, Sandanger, Vermeulen, Guida, and Chadeau-Hyam 2022; Dagnino, Bodinier, Guida, Smith-Byrne, Petrovic, Whitaker, Haugdahl Nøst, Agnoli, Palli, Sacerdote, Panico, Tumino, Schulze, Johansson, Keski-Rahkonen, Scalbert, Vineis, Johansson, Sandanger, Vermeulen, and Chadeau-Hyam 2021).

We created the R package **sharp** (Stability-enHanced Approaches using Resampling Procedures) to facilitate and expand the use of stability-based approaches. Our package offers an integrated framework for stability selection and consensus clustering, and an automated calibration of hyper-parameters using a consensus score (Bodinier *et al.* 2023b). It can be used for stability-enhanced (i) clustering, (ii) (multi-block) graphical modelling, (iii) regression, (iv) structural equation modelling, and (vi) dimensionality reduction.

In this paper, we first describe stability selection, consensus clustering and the calibration of hyper-parameters by maximising the consensus score. Second, we outline their implementation in **sharp** and present the architecture of the package. Third, we illustrate the use of the main functions on simulated datasets. We also cover more advanced features of the package including parallelisation and use with external functions, and provide recommendations for fine-tuning of the models. Finally, we apply stability selection and consensus clustering on publicly available gene expression datasets.

2. Statistical framework

2.1. Stability selection

In stability selection, the selection count $C_j(\lambda)$ for feature $j \in \{1, \ldots, N\}$ is the number of models fitted on K different subsamples of the observations with the same hyper-parameter λ that include feature j (Meinshausen and Bühlmann 2010). A feature denotes here a variable (in regression or dimensionality reduction), an edge (in graphical modelling), or an arrow (in structural equation modelling). The feature selection probability is estimated by its selection proportion $\Gamma_j(\lambda)$, calculated as:

$$\Gamma_j(\lambda) = \frac{C_j(\lambda)}{K}$$

Features with a selection proportion $\Gamma_j(\lambda)$ above a threshold $\pi \in]0, 1[$ are considered stably selected. The binary stability selection status $Z_j(\lambda, \pi)$ indicates if feature j is stably selected ($Z_j(\lambda, \pi) = 1$) or not ($Z_j(\lambda, \pi) = 0$):

$$Z_j(\lambda, \pi) = \mathbb{1}_{\Gamma_j(\lambda) \geq \pi}$$

We implemented stability selection for penalised (i) regression using the **glmnet** package, (ii) Structural Equation Modelling (SEM) using **glmnet**, **regsem** or **OpenMx**, (iii) Gaussian Graphical Modelling using **glassoFast**, (iv) Principal Component Analysis using **elasticnet**, and (v) Partial Least Squares using **sgPLS**.

2.2. Consensus clustering

In consensus clustering, the co-membership count $C_j(\lambda, g)$ between the two items in the j^{th} pair of items, where $j \in \{1, \ldots, N\}$, is the number of partitions of g clusters obtained with regularisation parameter λ (if using weighted distances) over the K subsampling iterations where the two items in the pair are (i) both drawn in the subsample, and (ii) assigned to the same cluster (Monti *et al.* 2003). The co-sampling count $H_j \leq K$ is the number of subsamples including both items from the j^{th} pair. The co-membership proportion $\Gamma_j(\lambda, g)$

is then calculated as:

$$\Gamma_j(\lambda, g) = \frac{C_j(\lambda, g)}{H_j}$$

The g consensus clusters are then obtained by applying a distance-based clustering algorithm (e.g. hierarchical clustering) using the co-membership proportions as a measure of similarity. The binary consensus co-membership status $Z_j(\lambda, g)$ indicates if the items in the j^{th} pair are in the same consensus cluster (if $Z_j(\lambda, g) = 1$) or not ($Z_j(\lambda, g) = 0$).

We implemented consensus clustering with (i) hierarchical clustering and K means using the **stats** package, (ii) Partitioning Around Medoids (PAM) using **cluster**, (iii) Density-Based Spatial Clustering of Applications with Noise (DBSCAN) using **dbscan**, and (iv) Gaussian Mixture Models (GMM) using **mclust**. The number of clusters G is a hyper-parameter for all clustering algorithms, except for DBSCAN, where the minimum distance between two items from the same cluster is used instead. Weighted distances calculated using the Clustering Objects on Subsets of Attributes (COSA) algorithm implemented in **rCOSA** can be used for hierarchical clustering, PAM or DBSCAN.

2.3. Calibration using the consensus score

The hyper-parameter(s) λ and π in stability selection, or g (and λ, if weighted) in consensus clustering, are calibrated by maximising a consensus score measuring results consistency over the subsamples (Bodinier *et al.* 2023b). For clarity, these hyper-parameter(s) are denoted by θ in this section, where $\theta = (\lambda, \pi)$ for stability selection and $\theta = g$ or $\theta = (\lambda, g)$ for consensus clustering. To define the consensus score, we first introduce the integers $X_1(\theta)$, $X_0(\theta)$, $N_1(\theta)$ and $N_0(\theta)$ as:

$$X_1(\theta) = \sum_{j=1}^{N} C_j(\theta) \mathbb{1}_{Z_j(\theta)=1}$$

$$X_0(\theta) = \sum_{j=1}^{N} C_j(\theta) \mathbb{1}_{Z_j(\theta)=0}$$

$$N_1(\theta) = \sum_{j=1}^{N} H_j \mathbb{1}_{Z_j(\theta)=1}$$

$$N_0(\theta) = \sum_{j=1}^{N} H_j \mathbb{1}_{Z_j(\theta)=0}$$

where $H_j(\theta) = K, \forall j \in \{1, \ldots, N\}$ in stability selection.

We assume that the counts $C_j(\theta)$, where $j \in \{1, \ldots, N\}$, follow independent binomial distributions, conditionally on the corresponding subsampling count H_j and status $Z_j(\theta)$:

$$C_j(\theta)|H_j, Z(\theta) \sim \mathcal{B}\left(H_j, p_j(\theta)\right)$$

To calculate the consensus score, we make the assumption that the probabilities $p_j(\theta)$ are the same for all stably selected features (or consensus co-members) on the one hand, and for all features that are not stably selected (or items that belong to different consensus clusters) on the other hand, that is:

$$p_j(\theta) = p_{Z_j(\theta)}(\theta)$$

where $p_0(\theta)$ and $p_1(\theta)$ are two unknown probabilities.

As a consequence, $X_0(\theta)$ and $X_1(\theta)$ also follow binomial distributions:

$$X_0(\theta)|H, Z(\theta) \sim \mathcal{B}\left(N_0(\theta), p_0(\theta)\right)$$
$$X_1(\theta)|H, Z(\theta) \sim \mathcal{B}\left(N_1(\theta), p_1(\theta)\right)$$

We consider that stability is characterised by a probability $p_1(\theta)$ that is larger than $p_0(\theta)$. The consensus score $S(\theta)$ is defined as the z statistic from a two-sample z test for the comparison of proportions where the null hypothesis of instability is $p_1(\theta) \leq p_0(\theta)$:

$$S(\theta) = \frac{\hat{p}_1(\theta) - \hat{p}_0(\theta)}{\sqrt{\hat{p}_t(\theta)\left(1 - \hat{p}_t(\theta)\right)\left(\frac{1}{N_1(\theta)} + \frac{1}{N_0(\theta)}\right)}}$$

where $\hat{p}_0(\theta) = \frac{X_0(\theta)}{N_0(\theta)}$, $\hat{p}_1 = \frac{X_1(\theta)}{N_1(\theta)}$, and $\hat{p}_t = \frac{X_0(\theta)+X_1(\theta)}{N_0(\theta)+N_1(\theta)}$.

The consensus score increases with stability and is maximised when proportions $\Gamma_j(\theta)$ are equal to 1 for all features j such that $Z_j(\theta) = 1$ and equal to 0 for all features j such that $Z_j(\theta) = 0$. The hyper-parameter(s) in θ are (jointly) calibrated by maximising the consensus score $S(\theta)$.

3. Implementation

3.1. Pseudocode

We use a grid search algorithm to calibrate the hyper-parameters in stability selection or consensus clustering. The values of hyper-parameter(s) to be tested for calibration need to be defined beforehand. For stability selection, we recommend the grid Λ for the regularisation parameter λ such that resulting model sizes range from 0 to $\min(N, n/2)$. For the threshold π, we use the set Π ranging from 0.01 to 0.99 with increments of 0.01. For consensus clustering, we consider numbers of clusters ranging from 2 to $G = n/4$ by default. The choice of the set Λ for the regularisation parameter λ in weighted distance calculation needs to be tailored for the application, but values ranging from 0.1 to 10 generally provide good performances (Bodinier *et al.* 2023b). The choice of the grids is illustrated in Section 4.8.2.

Given these grids of hyper-parameters, our procedure for stability selection or consensus clustering can be decomposed into four steps. First, we calculate the selection (for stability selection) or co-membership (for consensus clustering) counts $C(\lambda)$ and proportions $\Gamma(\lambda)$ for all values $\lambda \in \Lambda$ (Algorithm 1). To ensure reproducibility of the results, the random number generator is initialised at each subsampling iteration.

When fitting regularised models with multiple regularisation parameters $\lambda \in \Lambda$, computation time can be reduced by using the estimate obtained with Λ_i as a starting point for the gradient descent algorithm with Λ_{i+1}, where Λ is sorted in decreasing order (Simon, Friedman, Hastie, and Tibshirani 2011). Warm start is implemented directly in the R package **glmnet** for penalised regression and in our function `PenalisedGraphical()` in **sharp** which calls functions from **glassoFast** (Bodinier *et al.* 2023a).

This first step is the most computationally expensive as it requires the application of the (selection or clustering) algorithm on K subsamples. Computation time can be reduced using

Algorithm 1 Calculation of the counts $C(\lambda)$ and proportions $\Gamma(\lambda)$ for all $\lambda \in \Lambda$

Inputs: data X, set Λ, subsample proportion τ, number of subsamples K
Outputs: subsampling counts $H(\lambda)$, counts $C(\lambda)$ and proportions $\Gamma(\lambda)$ for all $\lambda \in \Lambda$

Initialise $H(\lambda) = 0_N$, $C(\lambda) = 0_N$ and $\Gamma(\lambda) = 0_N$ for all $\lambda \in \Lambda$
for $k \in \{1, \ldots, K\}$ **do**
  Initialise the random number generator state at k
  Define the subsample X^k with a proportion τ of the observations in X
  Extract the vector H^k of subsampling status
  $H = H + H^k$
  for $\lambda \in \Lambda$ **do**
    Apply the algorithm on X^k with parameter λ
    Extract the status $C^k(\lambda)$
    $C(\lambda) = C(\lambda) + C^k(\lambda)$
  end for
end for

for $j \in \{1, \ldots, N\}$ **do**
  $\Gamma_j(\lambda) = \frac{C_j(\lambda)}{H_j(\lambda)}$
end for

parallelisation over the K independent subsamples. We propose two ways of parallelisation in **sharp**: (i) a built-in procedure using functionalities from the **parallel** package (which relies on forking, only available on Unix systems), or (ii) users have the opportunity to manually run a subset of iterations on different cores and concatenate the results using our `Combine()` function (see Section 4.4).

Second, we define the stability selection status $Z(\lambda, \pi)$ for all $\lambda \in \Lambda$ and $\pi \in \Pi$ (for stability selection, Algorithm 2), or the consensus co-membership status $Z(\lambda, g)$ for all $\lambda \in \Lambda$ and $g \in \{1, \ldots, G\}$ (for consensus clustering, Algorithm 3), based on the proportions calculated in Step 1.

Algorithm 2 Define the stability selection sets $Z(\lambda, \pi)$ for all $\lambda \in \Lambda$ and $\pi \in \Pi$

Inputs: set Λ, set Π, selection proportions $\Gamma(\lambda)$ for all $\lambda \in \Lambda$
Outputs: stability selection sets $Z(\lambda, \pi)$ for all $\lambda \in \Lambda$ and $\pi \in \Pi$

Initialise $Z(\lambda, \pi) = 0_N$ for all $\lambda \in \Lambda$ and $\pi \in \Pi$
for $\lambda \in \Lambda$ **do**
  for $\pi \in \Pi$ **do**
    for $j \in \{1, \ldots, N\}$ **do**
      $Z_j(\lambda, \pi) = \mathbb{1}_{\Gamma_j(\lambda) \geq \pi}$
    end for
  end for
end for

Third, we calculate the consensus score for all visited combinations of hyper-parameters (Algorithm 4). The calibrated hyper-parameters in $\hat{\theta}$ are defined as the ones that yield the

Algorithm 3 Define the consensus co-memberships $Z(\lambda, g)$ for all $\lambda \in \Lambda$ and $g \in \{1, \ldots, G\}$

Inputs: set Λ, number G, co-membership proportions $\Gamma(\lambda)$ for all $\lambda \in \Lambda$
Outputs: consensus co-memberships $Z(\lambda, g)$ for all $\lambda \in \Lambda$ and $g \in \{1, \ldots, G\}$

Initialise $Z(\lambda, g) = 0_N$ for all $\lambda \in \Lambda$ and $g \in \{1, \ldots, G\}$
for $\lambda \in \Lambda$ **do**
  for $g \in \{1, \ldots, G\}$ **do**
    Apply distance-based clustering on distance $(1 - \Gamma(\lambda))$ with g clusters
    for $j \in \{1, \ldots, N\}$ **do**
      Extract consensus co-membership $Z_j(\lambda, g)$ for the j^{th} pair
    end for
  end for
end for

largest value of the consensus score.

Algorithm 4 Calibrate the hyper-parameter(s) in θ

Inputs: set θ, counts $C(\theta)$, status $Z(\theta)$ for all $\theta \in \Theta$
Outputs: calibrated $\hat{\theta}$

Initialise $S_{max} = 0$
for $\theta \in \Theta$ **do**
  $X_1(\theta) = \sum_{j=1}^{N} C_j(\theta) \mathbb{1}_{Z_j(\theta)=1}$
  $X_0(\theta) = \sum_{j=1}^{N} C_j(\theta) \mathbb{1}_{Z_j(\theta)=0}$
  $N_1(\theta) = \sum_{j=1}^{N} H_j \mathbb{1}_{Z_j(\theta)=1}$
  $N_0(\theta) = \sum_{j=1}^{N} H_j \mathbb{1}_{Z_j(\theta)=0}$
  $S(\theta) = \frac{\hat{p}_1(\theta) - \hat{p}_0(\theta)}{\sqrt{\hat{p}_t(\theta)(1-\hat{p}_t(\theta))\left(\frac{1}{N_1(\theta)} + \frac{1}{N_0(\theta)}\right)}}$
  if $S(\theta) \geq S_{max}$ **then**
    $\hat{\theta} = \theta$
  end if
end for

Finally, we extract the stability selection set or consensus clustering obtained in Step 2 (Algorithm 2 or 3) with the calibrated hyper-parameters identified in Step 3 (Algorithm 4).

3.2. Architecture of the package

Stability selection can be conducted using four main functions, including `VariableSelection()` for regression, `BiSelection()` for dimensionality reduction, `StructuralModel()` for structural equation modelling, and `GraphicalModel()` for Gaussian graphical modelling (Figure 1). Consensus clustering is implemented in the `Clustering()` function.

These main functions (Figure 1C) internally call `Resample()` (Figure 1D) for subsampling or bootstrapping (for stability selection only). At each resampling iteration, the selection or clustering algorithm is run via a wrapper function that reads and returns data in a stan-

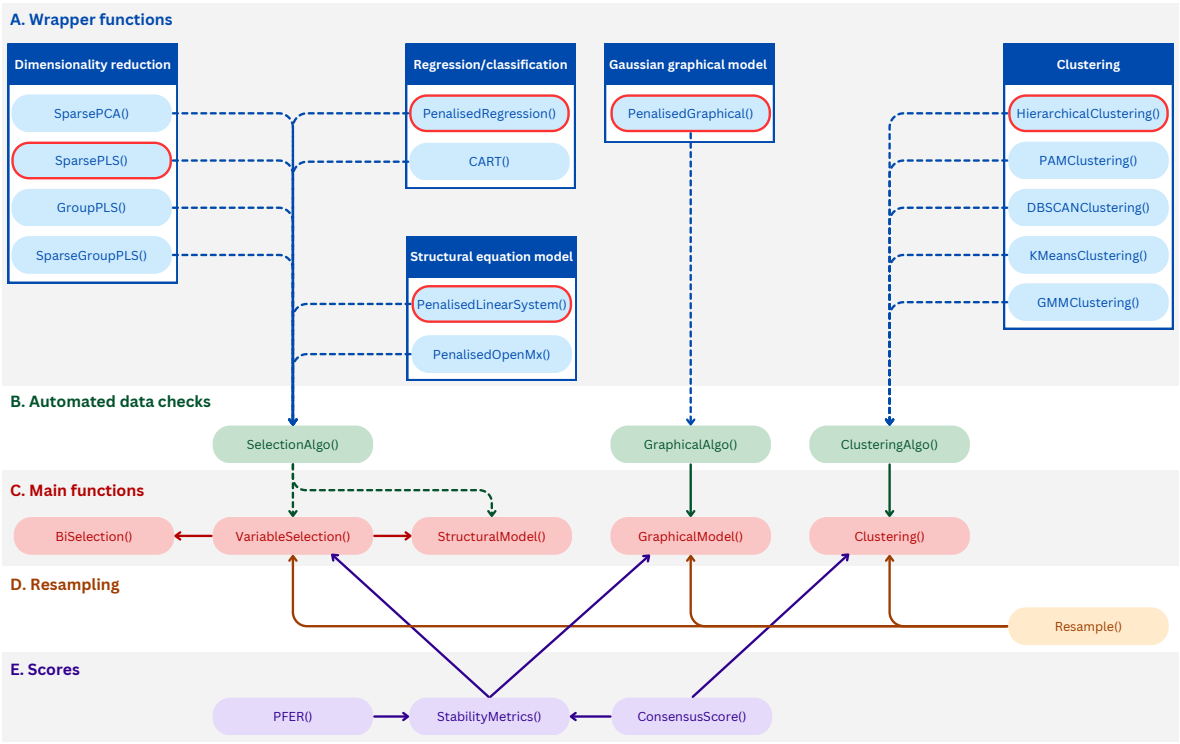

Figure 1: Architecture of the **sharp** package: main functions and their dependencies.

dardised format (Figure 1A). The user can choose the algorithm by specifying the function to use in argument `implementation` of the main function. Wrapper functions are called via intermediate functions (Figure 1B) that perform automated checks, including the exclusion of variables with null standard deviation in a given subsample. The consensus score, along with the upperbound of the expected number of false positives (for stability selection only, see Section 4.8.4) are calculated in separate functions (Figure 1E).

The package also includes a range of functions for results extraction and visualisation. The outputs of the main functions are assigned specific classes and come with S3 methods including `print`, `summary` and `plot`. Other functions are designed to be applied to these outputs, including `CalibrationPlot()` to visualise the consensus scores obtained with different hyper-parameters, `Stable` to extract the stability selection set or consensus clustering membership, `SelectionProportions()` and `ConsensusMatrix()`.

4. Usage

4.1. Requirements

To execute the lines of code presented in this paper, the R packages **sharp**, **corpcor**, **rpart** and **plotrix** need to be installed and loaded.

```
R> library(sharp)
```

```
R> library(corpcor)
R> library(rpart)
R> library(plotrix)
```

4.2. Typical use

We first illustrate the use of the main functions on artificial data generated with the R package **fake** (Figure 2) (Bodinier 2023). The simulation models in **fake** are based on (mixtures of) multivariate Gaussian distributions and allow for the simulation of (i) an outcome expressed as a linear combination of a subset of predictors and normally distributed noise with `SimulateRegression()`, (ii) variables with relationships encoded in a linear structural equation model with `SimulateStructural()`, (iii) data from a Gaussian graphical model with `SimulateGraphical()`, and (iv) clusters of items where attributes have cluster-specific means with `SimulateClustering()` (Bodinier *et al.* 2023a,b) (Figure 2A).

	Variable selection	Structural equation modelling	Gaussian graphical modelling	Clustering

A. Example data	`set.seed(1)`
	`data_reg <- SimulateRegression(`
	`  n = 100,`
	`  pk = 20,`
	`  beta_abs = 1`
	`)`
	`x_reg <- data_reg$xdata`
	`y_reg <- data_reg$ydata`
		`set.seed(1)`
		`data_sem <- SimulateStructural(`
		`  n = 200,`
		`  pk = c(5, 5, 5),`
		`  nu_between = 0.2,`
		`  v_between = 1`
		`)`
		`x_sem <- data_sem$data`
			`set.seed(1)`
			`data_ggm <- SimulateGraphical(`
			`  n = 200,`
			`  pk = 10,`
			`  v_within = 1,`
			`  topology = "scale-free"`
			`)`
			`x_ggm <- data_ggm$data`
				`set.seed(1)`
				`data_clust <- SimulateClustering(`
				`  n = c(5, 10, 20),`
				`  pk = 20,`
				`  ev_xc = 0.3`
				`)`
				`x_clust <- data_clust$data`

B. Stability selection/ consensus clustering	`stab_reg <- VariableSelection(`
	`  xdata = x_reg,`
	`  ydata = y_reg`
	`)`
		`dag <- LayeredDAG(`
		`  layers = c(5, 5, 5)`
		`)`
		`Lambda <- LambdaSequence(`
		`  lmax = 1,`
		`  lmin = 1e-5`
		`)`
		`stab_sem <- StructuralModel(`
		`  xdata = x_sem,`
		`  adjacency = dag,`
		`  Lambda = Lambda`
		`)`
			`stab_ggm <- GraphicalModel(`
			`  xdata = x_ggm`
			`)`
				`stab_clust <- Clustering(`
				`  xdata = x_clust`
				`)`

C. Calibration plot	`CalibrationPlot(stab_reg)`	`CalibrationPlot(stab_sem)`	`CalibrationPlot(stab_ggm)`	`CalibrationPlot(stab_clust)`
D. Results visualisation	`plot(stab_reg)`	`plot(stab_sem)`	`plot(stab_ggm)`	`plot(stab_clust)`

Figure 2: Typical use of functions implemented in **sharp** for stability selection and consensus clustering.

Figure 2B-D compiles code and generated plots illustrating the typical use of the main functions implemented in **sharp**. The parametrisation of the models, including the choice of the number of subsampling iterations K and definition of the grids of hyper-parameters is discussed below (Section 4.8). The consensus score obtained with visited hyper-parameters can

be visualised in the calibration plot (Figure 2C). In Figure 2D, the stability selection set is represented in red in a barplot of selection proportions (for variable selection), as arrows in a directed acyclic graph (for structural equation modelling), or as edges in an undirected graph (for Gaussian graphical modelling). For consensus clustering, the consensus clusters are colour-coded and separated by blue lines in the consensus matrix (Figure 2D).

4.3. Reproducibility

To ensure that two runs with the same parameters generate the same results, the random number generator is initialised using the argument `seed`.

```
R> set.seed(1)
R> simul <- SimulateRegression(
+    n = 50,
+    pk = 10
+  )
R> stab1 <- VariableSelection(
+    xdata = simul$xdata,
+    ydata = simul$ydata,
+    K = 5,
+    seed = 1,
+    verbose = FALSE
+  )
R> stab1_bis <- VariableSelection(
+    xdata = simul$xdata,
+    ydata = simul$ydata,
+    K = 5,
+    seed = 1,
+    verbose = FALSE
+  )
R> all(SelectionProportions(stab1) == SelectionProportions(stab1_bis))

[1] TRUE

R> stab2 <- VariableSelection(
+    xdata = simul$xdata,
+    ydata = simul$ydata,
+    K = 5,
+    seed = 2,
+    verbose = FALSE
+  )
R> all(SelectionProportions(stab1) == SelectionProportions(stab2))

[1] FALSE
```

4.4. Parallelisation

Stability approaches can easily be parallelised over the different subsamples (see section 3.1). We use internal functionalities from the **parallel** package and propose to choose the number of cores for parallelisation via argument `n_cores` (only available on Unix systems).

```
R> set.seed(1)
R> simul <- SimulateRegression(
+    n = 200,
+    pk = 100
+  )
R> stab <- VariableSelection(
+    xdata = simul$xdata,
+    ydata = simul$ydata,
+    K = 100,
+    n_cores = 2,
+    verbose = FALSE
+  )
```

Alternatively, any of the main functions (Figure 1C) can be run multiple times with different `seed`s and results can be merged using `Combine()` (available on all platforms).

```
R> stab1 <- VariableSelection(
+    xdata = simul$xdata,
+    ydata = simul$ydata,
+    K = 10,
+    seed = 1,
+    verbose = FALSE
+  )
R> stab2 <- VariableSelection(
+    xdata = simul$xdata,
+    ydata = simul$ydata,
+    K = 10,
+    seed = 2,
+    verbose = FALSE
+  )
R> stab <- Combine(stab1, stab2)
R> stab$params$K

[1] 20
```

4.5. Flexibility

Additional arguments

Any of the arguments of functions that are called to execute the algorithm within the wrapper functions (Figure 1A) can be specified directly in the main functions (Figure 1C). This is done using the ellipsis (`...`) argument in the functions showed in Figure 1A, B and C.

For example, variables can be forced in a LASSO model (i.e. not penalised) using the argument `penalty.factor` from **glmnet**. Although this is not an explicit argument of `VariableSelection()`, it can directly be used for stability selection. In the code below, we force the first three variables to be included in the model (i.e. we do not penalise their coefficients) using argument `penalty.factor`. The remaining seven variables are penalised (corresponding entries are set to `1` in the input vector). Even with a large penalty parameter, the first three variables have nonzero coefficients:

```
R> set.seed(1)
R> simul <- SimulateRegression(
+    n = 50,
+    pk = 10
+  )
R> stab <- VariableSelection(
+    xdata = simul$xdata,
+    ydata = simul$ydata,
+    Lambda = 1,
+    K = 1,
+    penalty.factor = c(0, 0, 0, rep(1, 7)),
+    verbose = FALSE
+  )
R> print(stab$Beta)

, , 1

      var1       var2      var3 var4 var5 var6 var7      var8 var9
s1 8.84633 -0.3034065 -1.081149    0    0    0    0 -4.846627    0
      var10
s1 2.330282
```

Choice of algorithm

The wrapper functions used by default in each of the main functions are circled in red in Figure 1. The wrapper function to apply at each of the resampling iterations can be specified in argument `implementation` of the main function. For example, classification and regression trees can be used as an alternative to regularised models (Breiman, Friedman, Olshen, and Stone 1984). In **sharp**, the structure of the decision tree is controlled by the number of splits. The features used in any of the splits are considered selected. This is implemented in the wrapper function `CART()`:

```
R> set.seed(1)
R> simul <- SimulateRegression(
+    n = 100,
+    pk = 20
+  )
R> stab <- VariableSelection(
```

```
+    xdata = simul$xdata,
+    ydata = simul$ydata,
+    implementation = CART,
+    verbose = FALSE
+  )
```

Using other algorithms

Selection and clustering algorithms beyond those readily implemented in **sharp** (Figure 1A) can be used by creating new wrapper functions with the same standardised inputs and outputs. Required arguments of wrapper functions for stability selection are `xdata`, `ydata` (for supervised modelling only), `Lambda` and the ellipsis (`...`). Clustering wrapper functions must have the arguments `xdata`, `Lambda` (for weighted distances only), `nc` and the ellipsis (`...`).

To be read correctly within the main functions in **sharp**, the output of the wrapper function must be (i) a named list with `selected` (a binary matrix indicating the selection status) and `beta_full` (matrix of the same size with model coefficients) for `VariableSelection()`, `StructuralModel` or `BiSelection()`, (ii) a three-dimensional array storing adjacency matrices for `GraphicalModel()`, or (iii) a named list with `comembership` (a three-dimensional array storing co-membership matrices) and `weight` (a matrix of attribute weights) for `Clustering()`.

To illustrate this functionality, we write below a function that infers conditional independence relationships using a threshold applied on the shrunk estimate of the partial correlation matrix, as implemented in **corpcor** (Schäfer and Strimmer 2005). Adjacency matrices obtained with different thresholds (provided in argument `Lambda`) are stored in the three-dimensional array `adjacency` and returned by the function.

```
R> library(corpcor)
R> ShrinkageSelection <- function(xdata, Lambda, ...) {
+    mypcor <- corpcor::pcor.shrink(xdata,
+      verbose = FALSE
+    )
+    adjacency <- array(NA, dim = c(nrow(mypcor), ncol(mypcor), nrow(Lambda)))
+    for (k in 1:nrow(Lambda)) {
+      A <- ifelse(abs(mypcor) >= Lambda[k, 1], yes = 1, no = 0)
+      diag(A) <- 0
+      adjacency[, , k] <- A
+    }
+    return(list(adjacency = adjacency))
+  }
```

The function `ShrinkageSelection()` can then be used as `implementation` in `GraphicalModel()`. Internally, this function will be applied on the subsamples and used as the selection algorithm to calculate selection proportions.

```
R> set.seed(1)
R> simul <- SimulateGraphical(
+    n = 100,
```

```
+    pk = 20,
+    nu_within = 0.1
+  )
R> stab <- GraphicalModel(
+    xdata = simul$data,
+    Lambda = matrix(c(0.01, 0.05, 0.1), ncol = 1),
+    implementation = ShrinkageSelection,
+    verbose = FALSE
+  )
```

Other examples with user-defined functions for variable selection are provided in the package documentation.

4.6. Additional tools for regression models

In a regression setting, stability selection aims to identify the predictors that are associated with the outcome but it cannot directly be used to generate predictions as it does not return regression coefficients. To evaluate prediction performances of stably selected variables, we propose to fit a Ridge regression model with all stably selected variables as predictors. The penalty parameter of the Ridge model is calibrated by cross validation as implemented in **glmnet**. From this refitted model, we can (i) extract regression coefficients, and (ii) generate predictions. In particular, this can be used to evaluate the explanatory performances of the model and the conditional contribution of each of the selected features.

To avoid bias and over-fitting, the data is split into non-overlapping training and test sets. In the example below, stability selection is applied on 70% of the observations. The same observations are used to fit the Ridge model with stably selected predictors. Prediction performances of the refitted model are evaluated on the remaining 30% of the observations. These two steps are done internally in `ExplanatoryPerformance()`. The fitted coefficients and Area Under the Curve (AUC) are reported.

```
R> set.seed(1)
R> simul <- SimulateRegression(
+    n = 500,
+    pk = 50,
+    ev_xy = 0.9,
+    family = "binomial"
+  )
R> ids <- Split(
+    data = simul$ydata,
+    family = "binomial",
+    tau = c(0.7, 0.3)
+  )
R> xtrain <- simul$xdata[ids[[1]], , drop = FALSE]
R> ytrain <- simul$ydata[ids[[1]], , drop = FALSE]
R> xtest <- simul$xdata[ids[[2]], , drop = FALSE]
R> ytest <- simul$ydata[ids[[2]], , drop = FALSE]
R> stab <- VariableSelection(
```

```
+    xdata = xtrain,
+    ydata = ytrain,
+    family = "binomial",
+    verbose = FALSE
+  )
R> perf <- ExplanatoryPerformance(
+    xdata = xtrain,
+    ydata = ytrain,
+    new_xdata = xtest,
+    new_ydata = ytest,
+    stability = stab
+  )
R> perf$AUC

[1] 0.8448578
```

Further, we propose to compare the performances of Ridge models where predictors are sequentially added by decreasing selection proportions to (i) quantify the contribution of each variable to the prediction conditionally on more stable variables, and (ii) validate the calibration of stability selection (Bodinier *et al.* 2023a). In Figure 3, we observe no further increase in performance beyond the calibrated model (red points).

```
R> incr <- Incremental(
+    xdata = xtrain,
+    ydata = ytrain,
+    new_xdata = xtest,
+    new_ydata = ytest,
+    stability = stab,
+    n_predictors = ncol(simul$xdata),
+    verbose = FALSE
+  )
R> plot(incr)
```

4.7. Additional tools for simulation studies

The ability of a selection algorithm to recover the correct set of features can be evaluated on simulated data where the true structure of the model is known. This can be done using **sharp** in combination with the simulation package **fake**. The function `SelectionPerformance()` returns several performance metrics by comparing the sets of features that are (i) used to generate the data, and (ii) selected when running the algorithm on the data.

```
R> set.seed(1)
R> simul <- SimulateRegression(
+    n = 100,
+    pk = 20
+  )
```

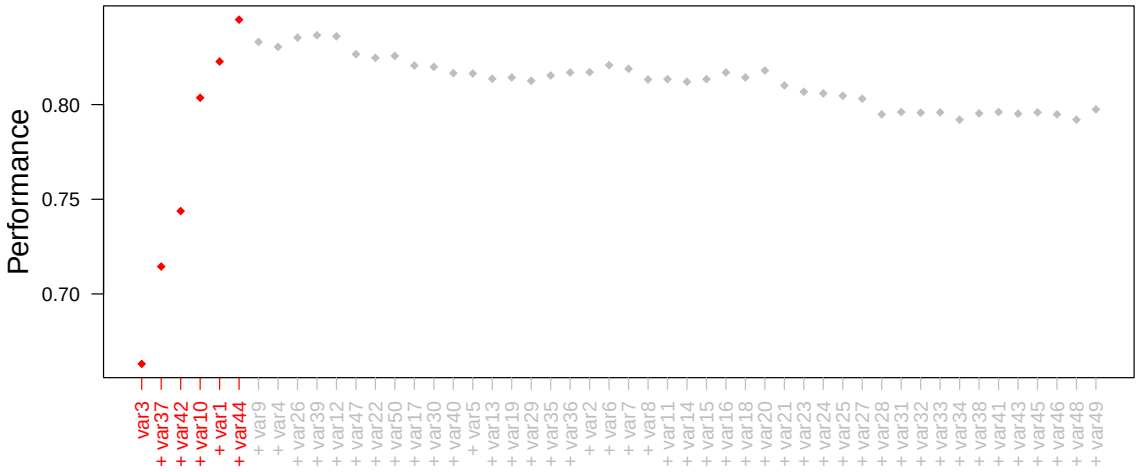

Figure 3: Explanatory performance of logistic models where predictors are incrementally added by order of decreasing selection proportion. The six red points indicate the six models sequentially including stably selected predictors. Grey points correspond to models including one or more predictors that are not stably selected.

```
R> stab <- VariableSelection(
+    xdata = simul$xdata,
+    ydata = simul$ydata,
+    verbose = FALSE
+  )
R> SelectionPerformance(stab, simul)

  TP FN FP TN sensitivity specificity accuracy precision    recall
1  4  2  0 14   0.6666667           1      0.9         1 0.6666667
  F1_score
1      0.8
```

Similarly, the ability of a clustering algorithm to recover the correct clusters can be evaluated by comparing (i) the true grouping structure used in the simulation, and (ii) the clusters detected by the algorithm. This can be done using the function `ClusteringPerformance()`.

```
R> set.seed(1)
R> simul <- SimulateClustering(
+    n = c(30, 30, 30),
+    ev_xc = 0.4
+  )
R> stab <- Clustering(
+    xdata = simul$data,
```

```
+    verbose = FALSE
+  )
R> ClusteringPerformance(stab, simul)

    TP  FN  FP   TN sensitivity specificity  accuracy precision
1 1191 114 117 2583   0.9126437   0.9566667 0.9423221 0.9105505
     recall  F1_score      rand       ari   jaccard
1 0.9126437 0.9115959 0.9423221 0.8687944 0.8375527
```

4.8. Fine-tuning of the models

Number of iterations

The performance of stability approaches depends on the number of resampling iterations that are performed. In a simulation study for the graphical LASSO, optimal performances were reached fairly quickly (around 100 iterations) (Bodinier *et al.* 2023a). More generally, we observe very limited changes in the sets of stably selected features from models using 1,000 iterations or more. Argument `K` defines the number of resampling iterations.

```
R> set.seed(1)
R> simul <- SimulateRegression(
+    n = 100,
+    pk = 20,
+    ev_xy = 0.5
+  )
R> for (K in c(10, 100, 1000)) {
+    print(paste0("K = ", K))
+    stab <- VariableSelection(
+      xdata = simul$xdata,
+      ydata = simul$ydata,
+      K = K,
+      verbose = FALSE
+    )
+    print(SelectionPerformance(stab, simul))
+  }

[1] "K = 10"
  TP FN FP TN sensitivity specificity accuracy precision recall
1  3  3  0 14         0.5           1     0.85         1    0.5
   F1_score
1 0.6666667
[1] "K = 100"
  TP FN FP TN sensitivity specificity accuracy precision    recall
1  4  2  0 14   0.6666667           1      0.9         1 0.6666667
  F1_score
1      0.8
```

```
[1] "K = 1000"
  TP FN FP TN sensitivity specificity accuracy precision    recall
1  4  2  0 14   0.6666667           1      0.9         1 0.6666667
  F1_score
1      0.8
```

Grids of hyper-parameters

The maximisation of the consensus score for calibration of the models is not a convex optimisation problem. To make sure that the global maximum is not missed by the grid search procedure, the sets of hyper-parameter values to explore needs to be chosen carefully. We recommend that the grid contains hyper-parameter values so that visited models range from very small to very large numbers of selected features. For models where the hyper-parameters cannot be directly interpreted in terms of numbers of selected features (e.g. LASSO as implemented in **glmnet**), the calibration plot can help in the definition of the grids. In the example below, we show the calibration plots from three runs with (i) a very restricted grid `Lambda` (A), and (ii) a better defined grid `Lambda` (B). The maximum of the stability score is missed in the first run. Poor calibration of the model can generate weaker selection performances (F_1-score of 0.57 compared to 0.75).

```
R> set.seed(1)
R> par(mfrow = c(1, 2), mar = c(7, 7, 7, 5))
R> simul <- SimulateRegression(
+    n = 200,
+    pk = 50
+  )
R> # Run 1
R> stab <- VariableSelection(
+    xdata = simul$xdata,
+    ydata = simul$ydata,
+    Lambda = LambdaSequence(lmax = 30, lmin = 20),
+    verbose = FALSE
+  )
R> CalibrationPlot(stab)
R> SelectionPerformance(stab, simul)

  TP FN FP TN sensitivity specificity accuracy precision recall
1  4  6  0 40         0.4           1     0.88         1    0.4
   F1_score
1 0.5714286

R> # Run 2
R> stab <- VariableSelection(
+    xdata = simul$xdata,
+    ydata = simul$ydata,
+    Lambda = LambdaSequence(lmax = 30, lmin = 0.5),
```

```
+    verbose = FALSE
+  )
R> CalibrationPlot(stab)
R> SelectionPerformance(stab, simul)

  TP FN FP TN sensitivity specificity accuracy precision recall
1  6  4  0 40         0.6           1     0.92         1    0.6
  F1_score
1     0.75
```

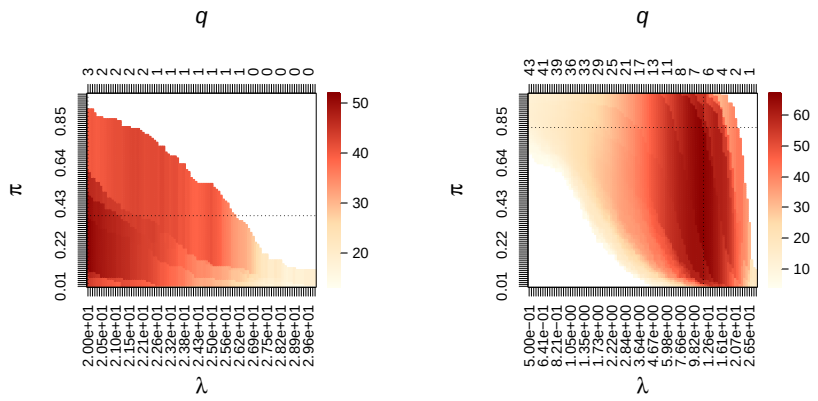

Figure 4: Calibration plots obtained for two runs with different grids of regularisation parameter λ. Each heatmap represents the consensus score (colour-coded) for different pairs of penalty parameter λ (x-axis) and threshold in selection proportion π (y-axis). The average number of selected features q corresponding the penalty parameter is showed on the z-axis.

Subsample size

Model performances may also depend on the parameters of the resampling procedure: the subsample size and the number and composition of subsamples to draw (Bodinier *et al.* 2023a).

For supervised stability selection with binary or categorical outcomes, we ensure that all subsamples contain the same proportions of observations from each category as in the full sample by default. The subsamples can be defined more specifically by writing a function to use in argument `resampling` (see examples for the function `Resample()` in the package documentation).

The subsample proportion `tau` does not generally affect selection performances (Figure 5), as long as values are not too high (values above 0.9 are not recommended to ensure that there is sufficient data perturbation) or too small (to ensure that there is enough data to run the algorithm at each iteration).

```
R> set.seed(1)
R> par(mfrow = c(1, 3), mar = c(7, 7, 7, 5))
R> simul <- SimulateRegression(
```

```
+    n = 200,
+    pk = 50
+  )
R> tau_list <- c(0.2, 0.5, 0.8)
R> for (i in 1:length(tau_list)) {
+    tau <- tau_list[i]
+    print(paste0("tau = ", tau))
+    stab <- VariableSelection(
+      xdata = simul$xdata,
+      ydata = simul$ydata,
+      tau = tau,
+      family = "gaussian",
+      verbose = FALSE
+    )
+    CalibrationPlot(stab)
+    print(SelectionPerformance(stab, simul))
+  }

[1] "tau = 0.2"
  TP FN FP TN sensitivity specificity accuracy precision recall
1  6  4  0 40         0.6           1     0.92         1    0.6
  F1_score
1     0.75
[1] "tau = 0.5"
  TP FN FP TN sensitivity specificity accuracy precision recall
1  6  4  0 40         0.6           1     0.92         1    0.6
  F1_score
1     0.75
[1] "tau = 0.8"
  TP FN FP TN sensitivity specificity accuracy precision recall
1  6  4  0 40         0.6           1     0.92         1    0.6
  F1_score
1     0.75
```

Constrained optimisation

In stability selection, a relationship between the number of selected features, threshold in selection proportion and the Per Family Error Rate (PFER) can be established (Meinshausen and Bühlmann 2010). The PFER is the expectation of type I errors, which can be interpreted in selection models as the expected number of false positives (i.e. features that are selected but should not be). Error control can be incorporated in the calibration by adding a constraint in the optimisation problem limiting the set of visited models to those with an expected PFER below a given threshold (Bodinier *et al.* 2023a). The use of a constraint on the expected number of false positives through the argument `PFER_thr` is recommended for high dimensional graphical models (Bodinier *et al.* 2023a). It ensures that sparser graphs are generated and selected edges are the most reliable ones.

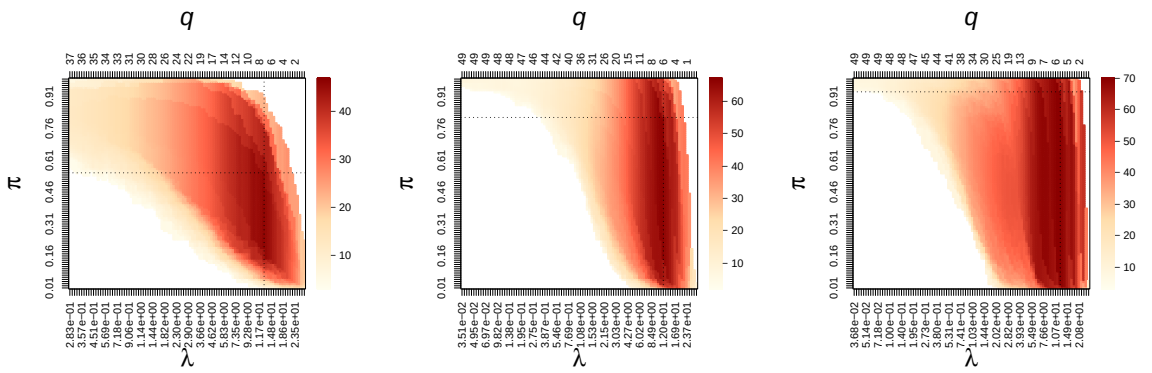

Figure 5: Calibration plots obtained for three runs with different subsample proportions `tau`.

```
R> set.seed(1)
R> simul <- SimulateGraphical(
+    n = 100,
+    pk = 20,
+    topology = "scale-free"
+  )
R> stab <- GraphicalModel(
+    xdata = simul$data,
+    verbose = FALSE
+  )
R> SelectionPerformance(stab, simul)
```

```
  TP FN FP  TN sensitivity specificity  accuracy precision recall
1 19  0  4 167           1   0.9766082 0.9789474  0.826087      1
   F1_score
1 0.9047619
```

```
R> stab <- GraphicalModel(
+    xdata = simul$data,
+    PFER_thr = 20,
+    verbose = FALSE
+  )
R> SelectionPerformance(stab, simul)
```

```
  TP FN FP  TN sensitivity specificity  accuracy precision recall
1 19  0  3 168           1   0.9824561 0.9842105 0.8636364      1
   F1_score
1 0.9268293
```

5. Illustration on real data

5.1. Dysregulated gene expression in lung tumours

In this section, we perform stability selection to identify a set of transcripts with dysregulated levels in lung cancer tumours compared to normal lung tissues. For this, we use publicly available gene expression measurements from 147 lung tumours and 353 normal lung tissues (Lim 2019). We filtered the 1,000 transcripts with the largest standard deviations.

We first split the 500 observations into non-overlapping training (70%) and test (30%) sets. Stability selection with logistic LASSO regression is applied on the training set by specifying the `family` of the model (**glmnet** argument). Visual inspection of the calibration heatmap suggests that the grid of penalty parameter is well defined here (Figure 6). The 19 transcripts with selection proportion above 0.23 are considered stably selected. The logistic model including these stably selected predictors yields an AUC of 0.98 in the test set, which is one of the best performances.

```
R> par(mfrow = c(1, 3), mar = c(7, 7, 7, 5))
R> # Loading example data
R> dat <- readRDS("e-mtab-6699.rds")
R> y <- dat$status
R> x <- dat[, -1]
R> # Data split
R> set.seed(1)
R> ids <- Split(data = y, family = "binomial", tau = c(0.7, 0.3))
R> # Stability selection
R> stab <- VariableSelection(
+    xdata = x[ids[[1]], ],
+    ydata = y[ids[[1]]],
+    family = "binomial",
+    verbose = FALSE
+  )
R> CalibrationPlot(stab)
R> plot(stab, n_predictors = 40)
R> sum(Stable(stab))

[1] 19

R> Argmax(stab)

        lambda   pi
[1,] 0.1127472 0.23

R> # Explanatory performances
R> incr <- Incremental(
+    xdata = x[ids[[1]], ],
+    ydata = y[ids[[1]]],
```

```
+     new_xdata = x[ids[[2]], ],
+     new_ydata = y[ids[[2]]],
+     stability = stab,
+     n_predictors = 40,
+     verbose = FALSE
+   )
R> plot(incr)
R> incr$AUC[[sum(Stable(stab))]]

[1] 0.9702306
```

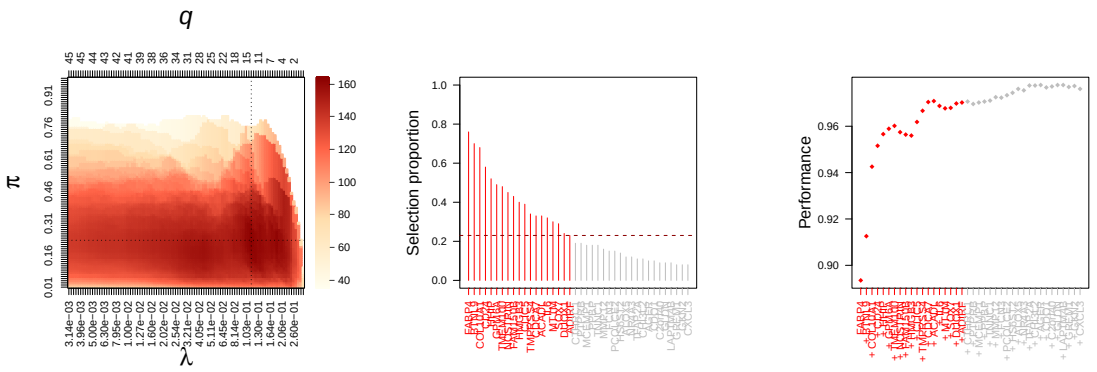

Figure 6: Stability selection to identify dysregulated gene expression in lung tumours compared to normal tissue.

5.2. Single-cell gene expression in multiple cell lines

Consensus clustering is now applied on 250 single-cell RNA-sequencing measurements from five cell lines, available at https://github.com/LuyiTian/sc_mixology. The 200 transcripts with the largest standard deviations are retained for the analysis.

Calibration maximising the consensus score detects 5 clusters, which mostly correspond to the 5 cell lines (Figure 7).

```
R> par(mfrow = c(1, 2), mar = rep(5, 4))
R> # Loading example data
R> dat <- readRDS("single_cell_10x_5cl.rds")
R> y <- dat$cell_line
R> z <- dat$doublet
R> x <- dat[, -c(1, 2)]
R> # Consensus clustering
R> stab <- Clustering(
+     xdata = x,
+     nc = 1:10,
+     verbose = FALSE
```

```
+  )
R> CalibrationPlot(stab)
R> plot(
+    stab,
+    theta_star = y,
+    cex.axis = 0.3
+  )
```

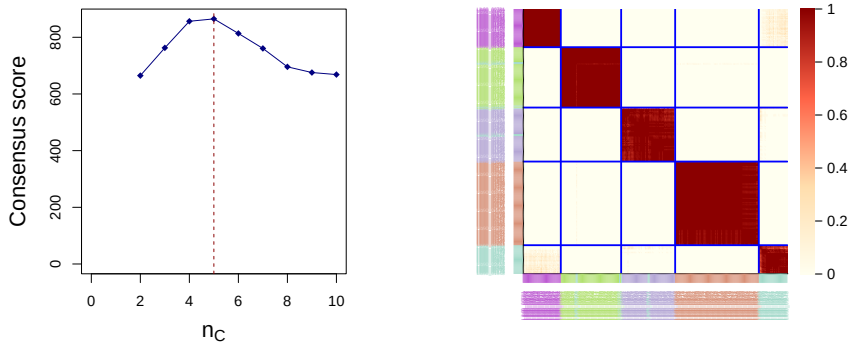

Figure 7: Consensus clustering on single-cell RNA-sequencing measurements from five cell lines.

All 6 observations that are not assigned to the correct cell line are actually doublets and not single cells, which may explain why they are misclassified.

```
R> # Doublets vs single cell
R> table(Clusters(stab), y, z)

, , z = DBL

   y
    A549 H1975 H2228 H838 HCC827
  1    1     0     0    0      0
  2    0     0     0    0      0
  3    0     3     0    0      0
  4    0     3     0    0      0
  5    0     0     0    0      0

, , z = SNG

   y
    A549 H1975 H2228 H838 HCC827
  1   78     0     0    0      0
  2    0    27     0    0      0
```

```
3     0     0    48     0     0
4     0     0     0    54     0
5     0     0     0     0    36
```

6. Conclusions

The functions of the R package **sharp** were implemented to be as flexible as possible, while remaining easy to use. We believe that the homogeneous approach to stability across regression, dimensionality reduction, and graphical modelling constitutes a strength of the proposed implementation. The output and visualisations have been standardised as much as possible across these different modelling techniques to facilitate quality checks and results interpretation. Thanks to the modular architecture of the package, more selection algorithms can be easily incorporated in the future. This opportunity is already available to advanced users who would like to design their own functions.

We implemented novel simulation tools for data with different structures including conditional independence relationships, variables associated with a latent component, and predictor variables contributing to the definition of outcome variables. The generated data can be used to evaluate the performances of graphical, dimensionality reduction and regression models, respectively. We link these topologies to conditional independence structure between variables. In a multivariate Gaussian framework, it can easily be modelled through the nonzero entries of the precision (inverse covariance) matrix. By explicitly modelling the correlation structure between the features, including marginal correlations arising between predictors jointly contributing to the definition of an outcome, we expect that the simulated data are more realistic than previously proposed sequential methods. These simulation models could be used, not only to generate toy datasets, but also to evaluate the performances of other feature selection approaches. This could guide the choice of model for real data applications.

Future versions of the **sharp** package may include additional functions to (i) facilitate the use of more feature selection models, and (ii) complement other modelling techniques by stability approaches.

Computational details

The results in this paper were obtained using R 4.3.0 with the **sharp** 1.4.1 package. The development version of **sharp** is available from https://github.com/barbarabodinier/sharp. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/ or from Bioconductor at https://www.bioconductor.org.

Acknowledgments

We are grateful to Matthew Whitaker, Rin Wada, Khezia Asamoah, Sibo Cheng, David Tang, Fernando Guntoro, Nina de Toro Eadie, as well as students from the cohorts 2019-2020, 2020-2021, 2021-2022 and 2022-2023 of the MSc in Health Data Analytics and Machine Learning at Imperial College for testing and providing feedback on the R package.

B. Bodinier received a PhD Studentship from the MRC Centre for Environment and Health J. Chiquet acknowledges support from ANR-18-CE45-0023 Statistics and Machine Learning for Single Cell Genomics (SingleStatOmics). M. Chadeau-Hyam and B. Bodinier acknowledge support from the H2020-EXPANSE project (Horizon 2020 grant no. 874627) and H2020-Longitools project (Horizon 2020 grant no. 874739).

References

Bodinier B (2023). *fake: Flexible Data Simulation Using the Multivariate Normal Distribution*. R package version 1.4.0, URL https://CRAN.R-project.org/package=fake.

Bodinier B, Filippi S, Nost TH, Chiquet J, Chadeau-Hyam M (2023a). "Automated calibration for stability selection in penalised regression and graphical models." 2106.02521.

Bodinier B, Vuckovic D, Rodrigues S, Filippi S, Chiquet J, Chadeau-Hyam M (2023b). "Automated calibration of consensus weighted distance-based clustering approaches using sharp." 2304.13706, URL https://arxiv.org/abs/2304.13706.

Breiman L, Friedman J, Olshen R, Stone C (1984). *Classification and Regression Trees*. Wadsworth.

Chadeau-Hyam M, Campanella G, Jombart T, Bottolo L, Portengen L, Vineis P, Liquet B, Vermeulen RC (2013). "Deciphering the complex: Methodological overview of statistical models to derive OMICS-based biomarkers." *Environmental and Molecular Mutagenesis*, **54**(7), 542–557. doi:https://doi.org/10.1002/em.21797. https://onlinelibrary.wiley.com/doi/pdf/10.1002/em.21797, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/em.21797.

Dagnino S, Bodinier B, Guida F, Smith-Byrne K, Petrovic D, Whitaker MD, Haugdahl Nøst T, Agnoli C, Palli D, Sacerdote C, Panico S, Tumino R, Schulze MB, Johansson M, Keski-Rahkonen P, Scalbert A, Vineis P, Johansson M, Sandanger TM, Vermeulen RCH, Chadeau-Hyam M (2021). "Prospective Identification of Elevated Circulating CDCP1 in Patients Years before Onset of Lung Cancer." *Cancer Res*, **81**(13), 3738–3748. ISSN 1538-7445 (Electronic); 0008-5472 (Print); 0008-5472 (Linking). doi:10.1158/0008-5472.CAN-20-3454.

Elliott J, Whitaker M, Bodinier B, Eales O, Riley S, Ward H, Cooke G, Darzi A, Chadeau-Hyam M, Elliott P (2021). "Predictive symptoms for COVID-19 in the community: REACT-1 study of over 1 million people." *PLoS medicine*, **18**(9), e1003777.

Friedman J, Hastie T, Tibshirani R (2007). "Sparse inverse covariance estimation with the graphical lasso." *Biostatistics*, **9**(3), 432–441. ISSN 1465-4644. doi:10.1093/biostatistics/kxm045. http://oup.prod.sis.lan/biostatistics/article-pdf/9/3/432/17742149/kxm045.pdf, URL https://doi.org/10.1093/biostatistics/kxm045.

Friedman JH, Meulman JJ (2004). "Clustering objects on subsets of attributes (with discussion)." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **66**(4), 815–849.

Jacobucci R, Grimm KJ, McArdle JJ (2016). "Regularized structural equation modeling." *Structural equation modeling: a multidisciplinary journal*, **23**(4), 555–566.

Lim SB (2019). "A microarray meta-dataset of lung cancer." https://www.ebi.ac.uk/biostudies/arrayexpress/studies/E-MTAB-6699.

Liquet B, de Micheaux PL, Hejblum BP, Thiébaut R (2015). "Group and sparse group partial least square approaches applied in genomics context." *Bioinformatics*, **32**(1), 35–42. ISSN 1367-4803. doi:10.1093/bioinformatics/btv535. https://academic.oup.com/bioinformatics/article-pdf/32/1/35/16919966/btv535.pdf, URL https://doi.org/10.1093/bioinformatics/btv535.

Meinshausen N, Bühlmann P (2006). "High-dimensional graphs and variable selection with the Lasso." *The Annals of Statistics*, **34**(3), 1436 – 1462. doi:10.1214/009053606000000281. URL https://doi.org/10.1214/009053606000000281.

Meinshausen N, Bühlmann P (2010). "Stability selection." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **72**(4), 417–473. doi:https://doi.org/10.1111/j.1467-9868.2010.00740.x. https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2010.00740.x, URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2010.00740.x.

Monti S, Tamayo P, Mesirov J, Golub T (2003). "Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data." *Machine Learning*, **52**(1), 91–118. doi:10.1023/A:1023949509487. URL https://doi.org/10.1023/A:1023949509487.

Petrovic D, Bodinier B, Dagnino S, Whitaker M, Karimi M, Campanella G, Haugdahl Nøst T, Polidoro S, Palli D, Krogh V, Tumino R, Sacerdote C, Panico S, Lund E, Dugué PA, Giles GG, Severi G, Southey M, Vineis P, Stringhini S, Bochud M, Sandanger TM, Vermeulen RCH, Guida F, Chadeau-Hyam M (2022). "Epigenetic mechanisms of lung carcinogenesis involve differentially methylated CpG sites beyond those associated with smoking." *European Journal of Epidemiology*. doi:10.1007/s10654-022-00877-2. URL https://doi.org/10.1007/s10654-022-00877-2.

Schäfer J, Strimmer K (2005). "A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics." *Statistical Applications in Genetics and Molecular Biology*, **4**(1). ISSN 2194-6302.

Shen H, Huang JZ (2008). "Sparse principal component analysis via regularized low rank matrix approximation." *Journal of Multivariate Analysis*, **99**(6), 1015–1034. ISSN 0047-259X. doi:https://doi.org/10.1016/j.jmva.2007.06.007. URL https://www.sciencedirect.com/science/article/pii/S0047259X07000887.

Simon N, Friedman J, Hastie T, Tibshirani R (2011). "Regularization paths for Cox's proportional hazards model via coordinate descent." *Journal of statistical software*, **39**(5), 1.

Tibshirani R (1996). "Regression Shrinkage and Selection Via the Lasso." *Journal of the Royal Statistical Society: Series B (Methodological)*, **58**(1), 267–288. doi:https://doi.org/

10.1111/j.2517-6161.1996.tb02080.x. https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1996.tb02080.x, URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1996.tb02080.x.

Whitaker M, Elliott J, Bodinier B, Barclay WS, Ward H, Cooke G, Donnelly CA, Chadeau-Hyam M, Elliott P (2022). "Variant-specific symptoms of COVID-19 among 1,542,510 people in England." *Nature communications*.

Zou H, Hastie T (2005). "Regularization and variable selection via the elastic net." *Journal of the royal statistical society: series B (statistical methodology)*, **67**(2), 301–320.

Zou H, Hastie T, Tibshirani R (2006). "Sparse Principal Component Analysis." *Journal of Computational and Graphical Statistics*, **15**(2), 265–286. doi:10.1198/106186006X113430. https://doi.org/10.1198/106186006X113430, URL https://doi.org/10.1198/106186006X113430.

Affiliation:

Barbara Bodinier, Marc Chadeau-Hyam
Department of Epidemiology and Biostatistics
School of Public Health
Imperial College London
St Mary's Hospital, Norfolk Place
London, W21PG, United Kingdom
E-mail: barbara.bodinier@gmail.com, m.chadeau@imperial.ac.uk