

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, Nøst, 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 Thiébaud 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, Ross Jacobucci 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 least absolute shrinkage and selection operator (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).

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).

However, 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.

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.

Stability selection had previously been implemented for variable selection in regression in the R package **stabs** (Hofner, Boccuto, and Göker 2015). In **stabs**, the amount of regularisation λ is chosen based on thresholds in selection proportion π and in the expected number of falsely selected variables provided by the user (Meinshausen and Bühlmann 2010; Shah and Samworth 2013). In **sharp**, stability selection can also be used for graphical modelling and dimensionality reduction. The choice of hyper-parameters is fully automated by default in **sharp**, but it can also rely on the same constraints in expected numbers of false positives.

The R package **clue** (Hornik 2005) enables the aggregation of multiple clusterings based on optimisation techniques. Consensus clustering using instead co-membership proportions has subsequently been proposed (Monti *et al.* 2003) and implemented in the R packages **ConsensusClusterPlus** (Wilkerson and Hayes 2010) and **M3C** (John, Watson, Russ, Goldmann, Ehrenstein, Pitzalis, Lewis, and Barnes 2020). The choice of the number of clusters is based on a maximisation of the Delta score in **ConsensusClusterPlus**, or on the Relative Cluster Stability Index (RCSI) in **M3C**. In a previous study, we showed better clustering performances using the consensus score used in **sharp** than with the Delta score in both simulated and real datasets (Bodinier *et al.* 2023b). Our consensus score is also very fast to compute, and yields clustering performances that are at least as good as when using the RCSI for a much lower computational cost (Bodinier *et al.* 2023b).

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 $j \in \{1, \dots, N\}$ 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 (Friedman, Hastie, and Tibshirani 2010), (ii) SEM based on a series of regressions fitted using **glmnet** (only possible when all variables are observed) or using **OpenMx** (Neale, Hunter, Pritikin, Zahery, Brick, Kirkpatrick, Estabrook, Bates, Maes, and Boker 2016) (for a more flexible modelling accommodating latent variables), (iii) Gaussian graphical modelling using **glassoFast** (Sustik, Calderhead, and Clavel 2023), (iv) Principal Component Analysis using **elasticnet** (Zou *et al.* 2006), and (v) Partial Least Squares using **sgPLS** (Liquet *et al.* 2015).

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 $j \in \{1, \dots, N\}$, 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 (R Core Team 2023), (ii) PAM using **cluster** (Maechler, Rousseeuw, Struyf, Hubert, and Hornik 2022), (iii) DBSCAN using **dbscan** (Hahsler, Piekenbrock, and Doran 2019), and (iv) GMM using **mclust** (Scrucca, Fop, Murphy, and Raftery 2016). 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 COSA algorithm implemented in **rCOSA** can be used for hierarchical clustering, PAM or DBSCAN (Kampert, Meulman, and Friedman 2017).

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

$$\begin{aligned} X_1(\boldsymbol{\theta}) &= \sum_{j=1}^N C_j(\boldsymbol{\theta}) Z_j(\boldsymbol{\theta}) \\ X_0(\boldsymbol{\theta}) &= \sum_{j=1}^N C_j(\boldsymbol{\theta}) (1 - Z_j(\boldsymbol{\theta})) \\ N_1(\boldsymbol{\theta}) &= \sum_{j=1}^N H_j Z_j(\boldsymbol{\theta}) \\ N_0(\boldsymbol{\theta}) &= \sum_{j=1}^N H_j (1 - Z_j(\boldsymbol{\theta})) \end{aligned}$$

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

We assume that the dependency between the counts $C_i(\boldsymbol{\theta})$ and $C_j(\boldsymbol{\theta})$, where $i \neq j$, can be fully explained by the structure encoded in $\mathbf{Z}(\boldsymbol{\theta})$. This implies that the $C_j(\boldsymbol{\theta})$, where $j \in \{1, \dots, N\}$, follow independent binomial distributions, conditionally on the corresponding subsampling count H_j and status $Z_j(\boldsymbol{\theta})$:

$$C_j(\boldsymbol{\theta}) | H_j, Z_j(\boldsymbol{\theta}) \sim \mathcal{B}(H_j, p_j(\boldsymbol{\theta}))$$

To calculate the consensus score, we make the assumption that the probabilities $p_j(\boldsymbol{\theta})$ 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:

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

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

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

$$\begin{aligned} X_0(\boldsymbol{\theta}) | H, \mathbf{Z}(\boldsymbol{\theta}) &\sim \mathcal{B}(N_0(\boldsymbol{\theta}), p_0(\boldsymbol{\theta})) \\ X_1(\boldsymbol{\theta}) | H, \mathbf{Z}(\boldsymbol{\theta}) &\sim \mathcal{B}(N_1(\boldsymbol{\theta}), p_1(\boldsymbol{\theta})) \end{aligned}$$

We consider that stability is characterised by a probability $p_1(\boldsymbol{\theta})$ that is larger than $p_0(\boldsymbol{\theta})$. The consensus score $S(\boldsymbol{\theta})$ is defined as the z statistic from a two-sample z test for the comparison

of proportions where the null hypothesis of instability is $p_1(\boldsymbol{\theta}) \leq p_0(\boldsymbol{\theta})$:

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

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

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

3. Implementation

3.1. Pseudocode

By default, 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 $\mathbf{\Lambda}$ for the regularisation parameter λ such that resulting model sizes range from 0 to $\min(N, n/2)$, where n is the sample size. For the threshold π , we use the set $\mathbf{\Pi}$ 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 $\mathbf{\Lambda}$ 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 4.9.2.

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 $\mathbf{C}(\boldsymbol{\theta})$ and proportions $\mathbf{\Gamma}(\boldsymbol{\theta})$ for all values $\boldsymbol{\theta} \in \Theta$ (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 $\lambda \in \mathbf{\Lambda}$, 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 $\mathbf{\Lambda}$ 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` (Friedman *et al.* 2010; Sustik *et al.* 2023; 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 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 $\mathbf{Z}(\boldsymbol{\lambda}, \boldsymbol{\pi})$ for all $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ and $\boldsymbol{\pi} \in \mathbf{\Pi}$ (for stability selection, Algorithm 2), or the consensus co-membership status $\mathbf{Z}(\boldsymbol{\lambda}, g)$ for all $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ and

Algorithm 1 Calculation of the counts $\mathbf{C}(\boldsymbol{\theta})$ and proportions $\boldsymbol{\Gamma}(\boldsymbol{\theta})$ for all $\boldsymbol{\theta} \in \Theta$

```

1: Inputs: data  $X$ , set  $\Theta$ , subsample proportion  $\tau$ , number of subsamples  $K$ 
2: Outputs: subsampling counts  $\mathbf{H}$ , counts  $\mathbf{C}(\boldsymbol{\theta})$  and proportions  $\boldsymbol{\Gamma}(\boldsymbol{\theta})$  for all  $\boldsymbol{\theta} \in \Theta$ 
3:
4: Initialise  $\mathbf{H} = \mathbf{0}_N$ ,  $\mathbf{C}(\boldsymbol{\theta}) = \mathbf{0}_N$  and  $\boldsymbol{\Gamma}(\boldsymbol{\theta}) = \mathbf{0}_N$  for all  $\boldsymbol{\theta} \in \Theta$ 
5: for  $k \in \{1, \dots, K\}$  do
6:   Initialise the random number generator state at  $k$ 
7:   Define the subsample  $X^k$  with a proportion  $\tau$  of the observations in  $X$ 
8:   Extract the vector  $H^k$  of subsampling status
9:    $\mathbf{H} = \mathbf{H} + H^k$ 
10:  for  $\boldsymbol{\theta} \in \Theta$  do
11:    Apply the algorithm on  $X^k$  with parameter  $\boldsymbol{\theta}$ 
12:    Extract the status  $\mathbf{C}^k(\boldsymbol{\theta})$ 
13:     $\mathbf{C}(\boldsymbol{\theta}) = \mathbf{C}(\boldsymbol{\theta}) + \mathbf{C}^k(\boldsymbol{\theta})$ 
14:  end for
15: end for
16:
17: for  $j \in \{1, \dots, N\}$  do
18:    $\Gamma_j(\boldsymbol{\theta}) = \frac{C_j(\boldsymbol{\theta})}{H_j(\boldsymbol{\theta})}$ 
19: end for

```

$g \in \{1, \dots, G\}$ (for consensus clustering, Algorithm 3), based on the proportions calculated in Step 1.

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

```

1: Inputs: set  $\Lambda$ , set  $\Pi$ , selection proportions  $\boldsymbol{\Gamma}(\lambda)$  for all  $\lambda \in \Lambda$ 
2: Outputs: stability selection sets  $\mathbf{Z}(\lambda, \pi)$  for all  $\lambda \in \Lambda$  and  $\pi \in \Pi$ 
3:
4: Initialise  $\mathbf{Z}(\lambda, \pi) = \mathbf{0}_N$  for all  $\lambda \in \Lambda$  and  $\pi \in \Pi$ 
5: for  $\lambda \in \Lambda$  do
6:   for  $\pi \in \Pi$  do
7:     for  $j \in \{1, \dots, N\}$  do
8:        $Z_j(\lambda, \pi) = \mathbb{1}_{\Gamma_j(\lambda) \geq \pi}$ 
9:     end for
10:   end for
11: end for

```

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

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).

As an alternative to grid search, it is also possible to use optimisation methods implemented in the R package `nloptr` (Johnson 2007). The lower and upper bounds are defined as above for the construction of the grids of regularisation parameters. The largest regularisation

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

- 1: **Inputs:** set $\mathbf{\Lambda}$, number G , co-membership proportions $\mathbf{\Gamma}(\lambda, g)$ for all $\lambda \in \mathbf{\Lambda}$
 - 2: **Outputs:** consensus co-memberships $\mathbf{Z}(\lambda, g)$ for all $\lambda \in \mathbf{\Lambda}$ and $g \in \{1, \dots, G\}$
 - 3:
 - 4: Initialise $\mathbf{Z}(\lambda, g) = 0_N$ for all $\lambda \in \mathbf{\Lambda}$ and $g \in \{1, \dots, G\}$
 - 5: **for** $\lambda \in \mathbf{\Lambda}$ **do**
 - 6: **for** $g \in \{1, \dots, G\}$ **do**
 - 7: Apply distance-based clustering on distance $(1 - \mathbf{\Gamma}(\lambda, g))$ with g clusters
 - 8: **for** $j \in \{1, \dots, N\}$ **do**
 - 9: Extract consensus co-membership $Z_j(\lambda, g)$ for the j^{th} pair
 - 10: **end for**
 - 11: **end for**
 - 12: **end for**
-

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

- 1: **Inputs:** set θ , counts $\mathbf{C}(\theta)$, status $\mathbf{Z}(\theta)$ for all $\theta \in \theta$
 - 2: **Outputs:** calibrated $\hat{\theta}$
 - 3:
 - 4: Initialise $S_{max} = 0$
 - 5: **for** $\theta \in \theta$ **do**
 - 6: $X_1(\theta) = \sum_{j=1}^N C_j(\theta) Z_j(\theta)$
 - 7: $X_0(\theta) = \sum_{j=1}^N C_j(\theta) (1 - Z_j(\theta))$
 - 8: $N_1(\theta) = \sum_{j=1}^N H_j Z_j(\theta)$
 - 9: $N_0(\theta) = \sum_{j=1}^N H_j (1 - Z_j(\theta))$
 - 10: $S(\theta) = \frac{\hat{p}_1(\theta) - \hat{p}_0(\theta)}{\sqrt{\hat{p}_t(\theta)(1 - \hat{p}_t(\theta)) \left(\frac{1}{N_1(\theta)} + \frac{1}{N_0(\theta)} \right)}}$
 - 11: **if** $S(\theta) \geq S_{max}$ **then**
 - 12: $\hat{\theta} = \theta$
 - 13: **end if**
 - 14: **end for**
-

parameter is used as a starting point. The set of visited hyper-parameters is then determined iteratively by the optimisation algorithm based on the current hyper-parameter value and consensus score. We use stopping criteria based on the number of iterations and absolute change in regularisation parameter (see Section 4.5). The grid search procedure detailed above would give identical results when using the set of visited hyper-parameters. Note that grid search may be faster for regularised approaches as it allows for warm start.

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.

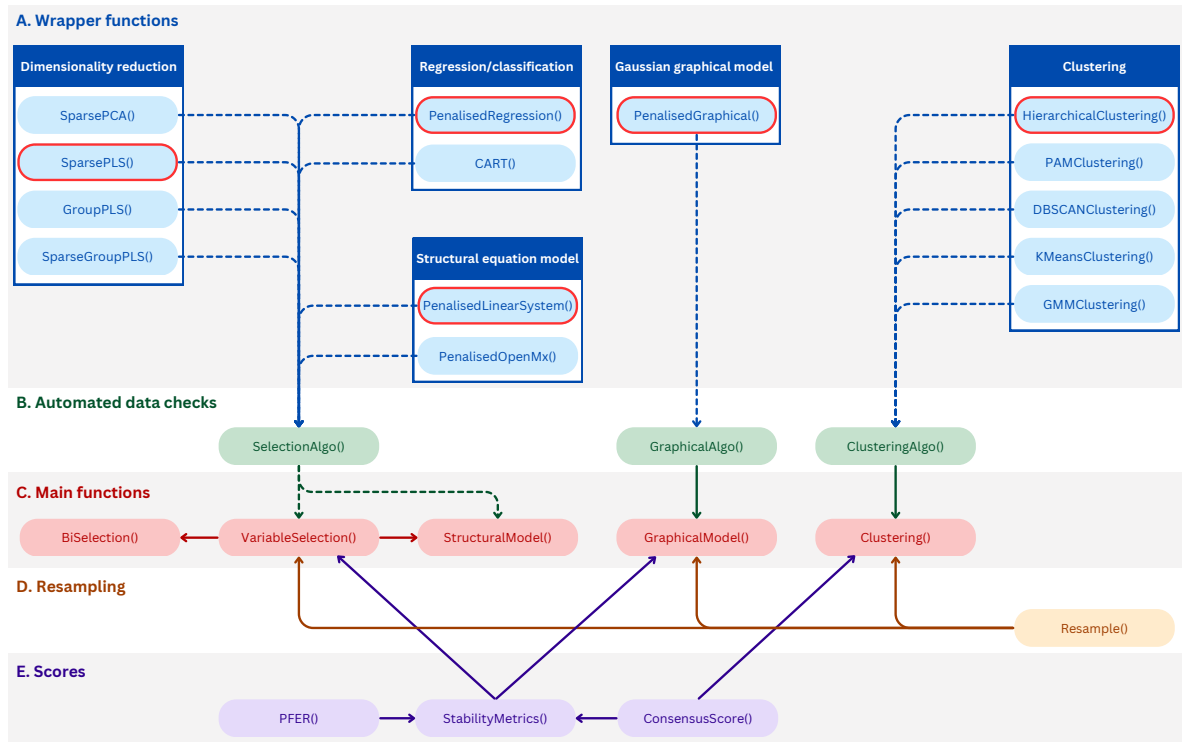


Figure 1: Architecture of the **sharp** package: main functions and their dependencies.

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 standardised 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 4.9.4) 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 **fake**, **sharp**, **corpcor** (Schäfer and Strimmer 2005), **rpart** (Therneau and Atkinson 2023) and **plotrix** (Lemon 2006) need to be installed and loaded.

```
R> library("fake")
R> library("sharp")
R> library("corpcor")
R> library("rpart")
R> library("plotrix")
```

We first use the package **fake** to generate the toy datasets that will be used throughout the paper to illustrate functionalities of the package **sharp**:

```
R> # Regression
R> set.seed(1)
R> data_reg <- SimulateRegression(
+   n = 100,
+   pk = 20,
+   beta_abs = 1
+ )
R> # Classification
R> set.seed(1)
R> data_class <- SimulateRegression(
+   n = 1000,
+   pk = 50,
+   ev_xy = 0.9,
+   family = "binomial"
+ )
R> # Structural equation modelling
R> set.seed(1)
R> data_sem <- SimulateStructural(
+   n = 200,
+   pk = c(5, 5, 5),
+   nu_between = 0.2,
+   v_between = 1
```

```

+ )
R> # Gaussian graphical modelling
R> set.seed(1)
R> data_ggm <- SimulateGraphical(
+   n = 100,
+   pk = 20,
+   topology = "scale-free"
+ )
R> # Clustering
R> set.seed(1)
R> data_clust <- SimulateClustering(
+   n = c(5, 10, 20),
+   pk = 20,
+   ev_xc = 0.3
+ )

```

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).

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 4.9).

The consensus score obtained with visited hyper-parameters can be visualised in the calibration plot (Figure 2C). For stability selection (in the first three columns), the consensus score is colour-coded and represented as a function of the regularisation parameter λ on the x-axis, and of the threshold in selection proportion π on the y-axis. The average number q of selected features across the K models fitted with the corresponding amount of regularisation λ is also reported at the top of the heatmap. The combination of hyper-parameters that yields the most stable model (in darker red) is at the intersection of the horizontal and vertical dashed lines. For consensus clustering, the consensus score is represented as a function of the number of clusters (last column in Figure 2C). The chosen number of clusters is indicated by a vertical dashed line. For both stability selection and consensus clustering, the calibrated hyper-parameters can be obtained with the function `Argmax()` (Figure 2C).

Figure 2D shows the final result, where the stability selection set is represented as red bars 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).

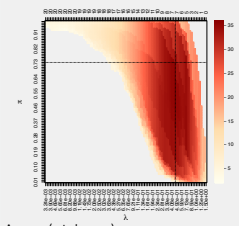
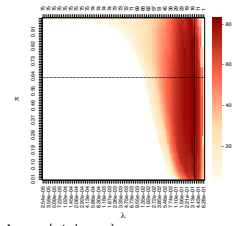
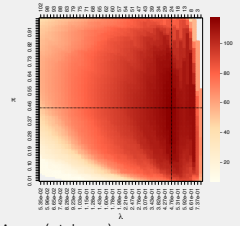
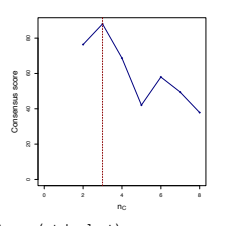
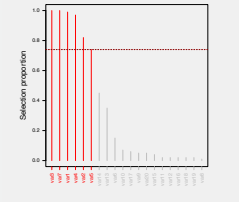
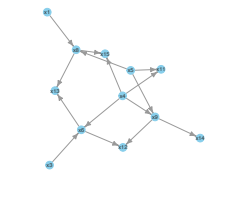
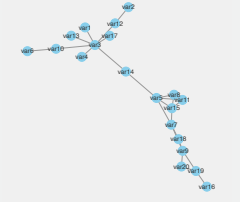
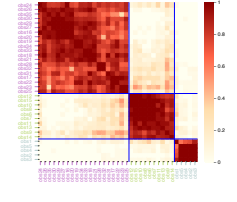
	Variable selection	Structural equation modelling	Gaussian graphical modelling	Clustering
A. Example data	<pre>set.seed(1) data_reg <- SimulateRegression(n = 100, pk = 20, beta_abs = 1) x_reg <- data_reg\$xdata y_reg <- data_reg\$ydata</pre>	<pre>set.seed(1) data_sem <- SimulateStructural(n = 200, pk = c(5, 5, 5), nu_between = 0.2, v_between = 1) x_sem <- data_sem\$data</pre>	<pre>set.seed(1) data_ggm <- SimulateGraphical(n = 100, pk = 20, topology = "scale-free") x_ggm <- data_ggm\$data</pre>	<pre>set.seed(1) data_clust <- SimulateClustering(n = c(5, 10, 20), pk = 20, ev_xc = 0.3) x_clust <- data_clust\$data</pre>
B. Stability selection/ consensus clustering	<pre>stab_reg <- VariableSelection(xdata = x_reg, ydata = y_reg)</pre>	<pre>dag <- LayeredDAG(layers = c(5, 5, 5)) Lambda <- LambdaSequence(lmax = 1, lmin = 1e-5) stab_sem <- StructuralModel(xdata = x_sem, adjacency = dag, Lambda = Lambda)</pre>	<pre>stab_ggm <- GraphicalModel(xdata = x_ggm)</pre>	<pre>stab_clust <- Clustering(xdata = x_clust)</pre>
C. Calibration plot	<pre>CalibrationPlot(stab_reg)</pre>  <pre>Argmax(stab_reg) lambda pi > [1,] 0.4621656 0.74</pre>	<pre>CalibrationPlot(stab_sem)</pre>  <pre>Argmax(stab_sem) lambda pi > [1,] 0.3511192 0.63</pre>	<pre>CalibrationPlot(stab_ggm)</pre>  <pre>Argmax(stab_ggm) lambda pi > [1,] 0.475961 0.45</pre>	<pre>CalibrationPlot(stab_clust)</pre>  <pre>Argmax(stab_clust) nc lambda > [1,] 3 NA</pre>
D. Results visualisation	<pre>plot(stab_reg)</pre> 	<pre>plot(stab_sem)</pre> 	<pre>plot(stab_ggm)</pre> 	<pre>plot(stab_clust)</pre> 

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

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> stab1 <- VariableSelection(
+   xdata = data_reg$xdata,
+   ydata = data_reg$ydata,
+   K = 5,
+   seed = 1,
+   verbose = FALSE
+ )
R> stab1_bis <- VariableSelection(
+   xdata = data_reg$xdata,
+   ydata = data_reg$ydata,
+   K = 5,
+   seed = 1,
+   verbose = FALSE
```

```
+ )
R> all(SelectionProportions(stab1) == SelectionProportions(stab1_bis))

[1] TRUE
```

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 Core Team 2023).

```
R> stab <- VariableSelection(
+   xdata = data_reg$xdata,
+   ydata = data_reg$ydata,
+   K = 100,
+   n_cores = 2,
+   verbose = FALSE
+ )
```

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

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

```
[1] 20
```

4.5. Optimisation

By default, we use a grid search approach maximising the consensus score. Alternatively, optimisation methods implemented in **nloptr** can be used with argument `optimisation="nloptr"`.

In this case, the algorithm `NLOPT_GN_DIRECT_L` is used by default (Gablonsky and Kelley 2001). The optimisation algorithm and stopping criteria can be chosen via the argument `opts` which is passed to functions from `nloptr`.

```
R> stab <- VariableSelection(
+   xdata = data_reg$xdata,
+   ydata = data_reg$ydata,
+   optimisation = "nloptr"
+ )
```

4.6. 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` (Friedman *et al.* 2010). 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 17 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> stab <- VariableSelection(
+   xdata = data_reg$xdata,
+   ydata = data_reg$ydata,
+   Lambda = 1,
+   K = 1,
+   penalty.factor = c(0, 0, 0, rep(1, 17)),
+   verbose = FALSE
+ )
R> print(stab$Beta)
```

```
, , 1
```

	var1	var2	var3	var4	var5	var6	var7	var8	var9	var10	var11
s1	1.5071	1.401676	1.300818	0	0	0	0	0	0	0	0
	var12	var13	var14	var15	var16	var17	var18	var19	var20		
s1	0	0	0	0	0	0	0	0	0		

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 (CART) 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> stab <- VariableSelection(
+   xdata = data_reg$xdata,
+   ydata = data_reg$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> 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> stab <- GraphicalModel(
+   xdata = data_ggm$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.

4.7. 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** (Friedman *et al.* 2010). 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 using the function `Split()`. 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, for a binary outcome), or R squared (for a continuous outcome) are reported.

```
R> set.seed(1)
R> ids <- Split(
+   data = data_class$ydata,
+   family = "binomial",
+   tau = c(0.7, 0.3)
+ )
R> xtrain <- data_class$xdata[ids[[1]], , drop = FALSE]
R> ytrain <- data_class$ydata[ids[[1]], , drop = FALSE]
R> xtest <- data_class$xdata[ids[[2]], , drop = FALSE]
R> ytest <- data_class$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] 0.8809746
```

Further, we use the function `Incremental()` 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(data_class$xdata),
+   verbose = FALSE
+ )
R> plot(incr, ylim = c(0.5, 1))

```

4.8. 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** (Bodinier 2023). 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
+ )
R> stab <- VariableSelection(
+   xdata = simul$xdata,
+   ydata = simul$ydata,

```

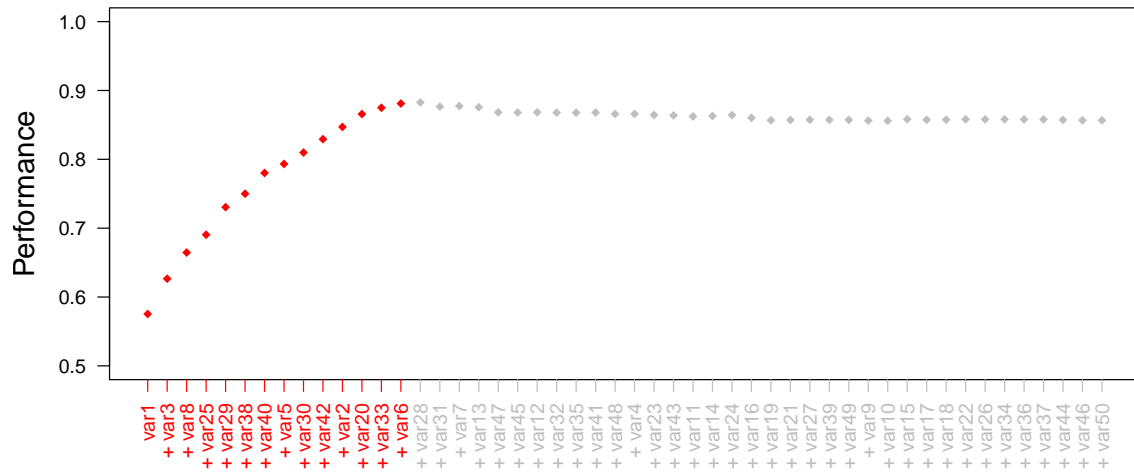


Figure 3: Explanatory performance of logistic models where predictors are incrementally added by order of decreasing selection proportion. The six red points indicate the six models sequentially including stably selected predictors. Grey points correspond to models including one or more predictors that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

```
+   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 1166 139 146 2554  0.8934866   0.9459259 0.928839 0.8887195
      recall F1_score      rand      ari   jaccard
1 0.8934866 0.8910967 0.928839 0.838251 0.8035837

```

4.9. 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> lout <- NULL
R> K_vect <- c(10, 100, 1000)
R> for (K in K_vect) {
+   stab <- VariableSelection(
+     xdata = data_class$xdata,
+     ydata = data_class$ydata,
+     K = K,
+     verbose = FALSE
+   )
+   lout <- c(lout, list(SelectionPerformance(stab, data_class)))
+ }
R> names(lout) <- K_vect
R> print(lout)

```

```

$'10'
      TP FN FP TN sensitivity specificity accuracy precision  recall
1 13  2  0 35  0.8666667           1    0.96           1 0.8666667
      F1_score
1 0.9285714

```

```

$'100'
      TP FN FP TN sensitivity specificity accuracy precision  recall
1 14  1  0 35  0.9333333           1    0.98           1 0.9333333
      F1_score
1 0.9655172

```

```

$'1000'
      TP FN FP TN sensitivity specificity accuracy precision  recall
1 14  1  0 35  0.9333333           1    0.98           1 0.9333333
      F1_score
1 0.9655172

```

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 two runs with (i) a poorly defined grid `Lambda`, and (ii) a well defined grid `Lambda` (Figure 4). The largest stability score obtained with the poorly defined grid (Figure 4, left panel) is much smaller than with the well defined grid (right panel). The global maximum of the stability score is missed in the first run. Visual inspection of the calibration plot using the poorly defined grid suggests that smaller regularisation parameters should be considered as the largest stability score is obtained for the smallest regularisation parameter (as indicated by the vertical dashed line on the left hand side of the heatmap). The calibration plot should show a clear peak of stability if the grid of hyper-parameters is well defined. Poor calibration of the model can generate weaker selection performances (F_1 -score of 0.57 compared to 0.75).

```
R> par(mfrow = c(1, 2), mar = c(7, 7, 7, 5))
R> # Run 1
R> stab <- VariableSelection(
+   xdata = data_class$xdata,
+   ydata = data_class$ydata,
+   Lambda = LambdaSequence(lmax = 0.2, lmin = 0.1),
+   verbose = FALSE
+ )
R> CalibrationPlot(stab)
R> SelectionPerformance(stab, data_class)

  TP FN FP TN sensitivity specificity accuracy precision  recall
1  4 11  0 35  0.2666667           1    0.78           1 0.2666667
  F1_score
1 0.4210526

R> # Run 2
R> stab <- VariableSelection(
+   xdata = data_class$xdata,
+   ydata = data_class$ydata,
+   Lambda = LambdaSequence(lmax = 0.2, lmin = 0.01),
+   verbose = FALSE
+ )
R> CalibrationPlot(stab)
R> SelectionPerformance(stab, data_class)

  TP FN FP TN sensitivity specificity accuracy precision  recall
1 14  1  0 35  0.9333333           1    0.98           1 0.9333333
```

```
F1_score
1 0.9655172
```

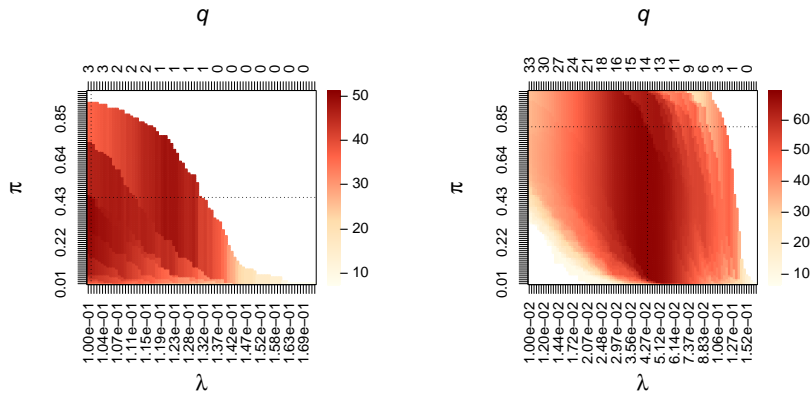


Figure 4: Calibration plots obtained for two runs with different grids of regularisation parameter λ . Each heatmap represents the consensus score (colour-coded) for different pairs of penalty parameter λ (x-axis) and threshold in selection proportion π (y-axis). The average number of selected features 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> par(mfrow = c(1, 3), mar = c(7, 7, 7, 5))
R> lout <- NULL
R> tau_vect <- c(0.2, 0.5, 0.8)
R> for (tau in tau_vect) {
+   stab <- VariableSelection(
+     xdata = data_class$xdata,
+     ydata = data_class$ydata,
+     tau = tau,
+     family = "binomial",
+     verbose = FALSE
```

```

+   )
+   CalibrationPlot(stab)
+   lout <- c(lout, list(SelectionPerformance(stab, data_class)))
+ }
R> names(lout) <- tau_vect
R> print(lout)

```

```

$'0.2'
  TP FN FP TN sensitivity specificity accuracy precision recall
1 14  1  0 35  0.9333333          1    0.98          1 0.9333333
  F1_score
1 0.9655172

```

```

$'0.5'
  TP FN FP TN sensitivity specificity accuracy precision recall
1 14  1  0 35  0.9333333          1    0.98          1 0.9333333
  F1_score
1 0.9655172

```

```

$'0.8'
  TP FN FP TN sensitivity specificity accuracy precision recall
1 14  1  0 35  0.9333333          1    0.98          1 0.9333333
  F1_score
1 0.9655172

```

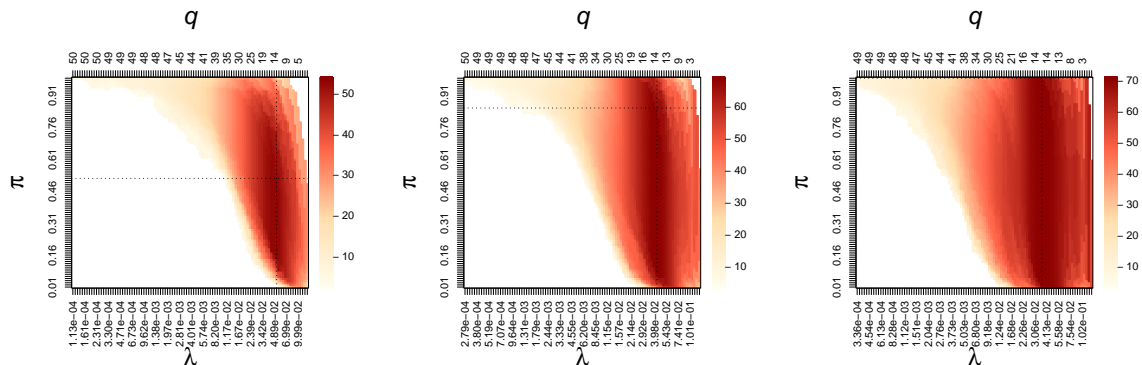


Figure 5: Calibration plots obtained for three runs with different subsample proportions τ .

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.

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

  TP FN FP  TN sensitivity specificity  accuracy precision recall
1 19  0  6 165           1  0.9649123 0.9684211      0.76      1
  F1_score
1 0.8636364
```

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

  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 genes with dysregulated expression levels in patients with Crohn’s disease compared to controls who do not have inflammatory bowel diseases (IBD). For this, we use publicly available RNAseq data for 156 patients and 267 non-IBD controls from the IBD-Character cohort (Halfvarson, Kalla, Adams, Satsangi, and Nowak 2022). We kept the 1,000 transcripts with the largest standard deviations.

We first split the 423 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 as it includes a clear peak in consensus score (Figure 6, left).

The 2 genes with selection proportion above 0.59 are considered stably selected (Figure 6, middle). The logistic model including these stably selected predictors yields an AUC of 0.88 in the test set. We observe a limited increase in AUC when including additional predictors in the model (increase in AUC < 0.02), which suggests that stably selected features carry most of the information needed to predict the outcome (Figure 6, right).

```
R> par(mfrow = c(1, 3), mar = c(7, 7, 7, 5))
R> dat <- readRDS("e-mtab-11349.rds")
R> y <- dat[, 1]
R> x <- dat[, -1]
R> set.seed(1)
R> ids <- Split(data = y, family = "binomial", tau = c(0.7, 0.3))
R> stab <- VariableSelection(
+   xdata = x[ids[[1]], ],
+   ydata = y[ids[[1]]],
+   family = "binomial",
+   verbose = FALSE
+ )
R> CalibrationPlot(stab)
R> plot(stab, n_predictors = 20)
R> sum(Stable(stab))
```

```
[1] 2
```

```
R> Argmax(stab)
```

```
      lambda  pi
[1,] 0.1924526 0.59
```

```
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 = 20,
+   verbose = FALSE
+ )
R> plot(incr, ylim = c(0.5, 0.9))
R> incr$AUC[[sum(Stable(stab))]]
```

```
[1] 0.881534
```

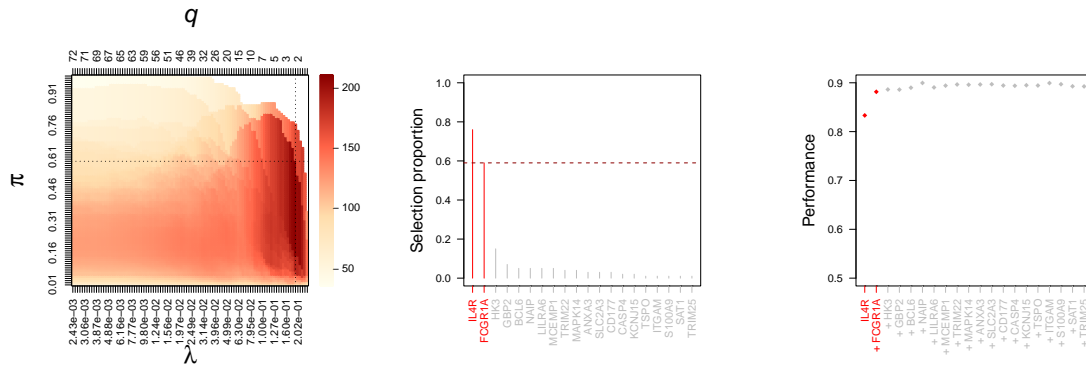



Figure 6: Stability selection to identify dysregulated gene expression in lung tumours compared to normal tissue.

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.

We apply consensus clustering using 2 to 10 clusters. The calibration plot shows that the largest consensus score is obtained with 5 clusters (Figure 7, left). The calibrated consensus matrix suggests very stable results, as co-membership proportions are close to one within clusters and close to zero between clusters (Figure 7, right). In this heatmap, the cell labels are coloured based on the cell line they are derived from. Reassuringly, the detected clusters mostly correspond to the 5 cell lines (Figure 7).

```
R> par(mfrow = c(1, 2), mar = rep(5, 4))
R> dat <- readRDS("single_cell_10x_5cl.rds")
R> y <- dat$cell_line
R> z <- dat$doublet
R> x <- dat[, -c(1, 2)]
R> stab <- Clustering(
+   xdata = x,
+   nc = seq_len(10),
+   verbose = FALSE
+ )
R> CalibrationPlot(stab)
R> plot(
+   stab,
+   theta_star = y,
+   cex.axis = 0.3
+ )
```

Further inspection of the data reveals that the 6 samples that are not assigned to the correct cluster are actually doublets and not single cells, which may explain why they are misclassified.

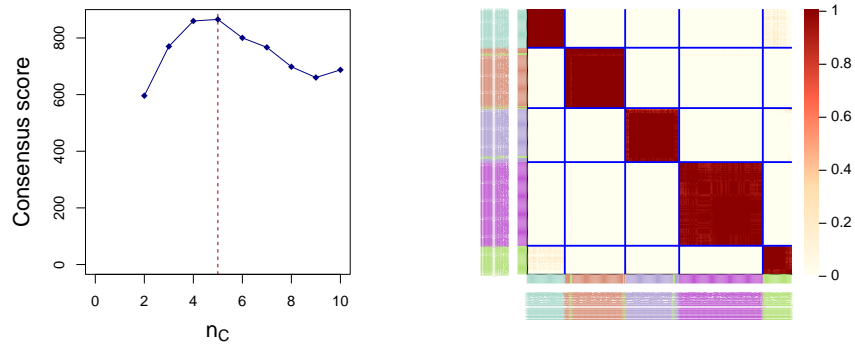


Figure 7: Consensus clustering on single-cell RNA-sequencing measurements from five cell lines.

```
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 R package **sharp** provides an integrated framework using stability in several contexts, including penalised regression, graphical modelling, structural equation modelling and clustering. The automated calibration procedure and visualisation tools provided should facilitate the use of these approaches for the analysis of real-world data. Functions 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. The R package **stablelearner** (Michel Philipp and Strobl 2018) could also provide alternative measures of stability for regression models that could complement the consensus score used in **sharp**.

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** 1.4.6 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, Ruben Colindres Zuehlke, Khezia Asamoah, Sibö 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, Nøst TH, Chiquet J, Chadeau-Hyam M (2023a). “Automated Calibration for Stability Selection in Penalised Regression and Graphical Models.” *Journal of the Royal Statistical Society Series C: Applied Statistics*, **72**(5), 1375–1393. doi: [10.1093/jrssc/qlad058](https://doi.org/10.1093/jrssc/qlad058).
- Bodinier B, Vuckovic D, Rodrigues S, Filippi S, Chiquet J, Chadeau-Hyam M (2023b). “Au-

- tomated Calibration of Consensus Weighted Distance-based Clustering Approaches using sharp.” *Bioinformatics*, **39**(11), btad635. doi:10.1093/bioinformatics/btad635.
- Breiman L, Friedman J, Olshen R, Stone C (1984). *Classification and Regression Trees*. Wadsworth. doi:10.1201/9781315139470.
- 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: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 Research*, **81**(13), 3738–3748. 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), 1–14. doi:10.1371/journal.pmed.1003777.
- Friedman J, Hastie T, Tibshirani R (2007). “Sparse Inverse Covariance Estimation with the Graphical Lasso.” *Biostatistics*, **9**(3), 432–441. doi:10.1093/biostatistics/kxm045.
- Friedman JH, Hastie T, Tibshirani R (2010). “Regularization Paths for Generalized Linear Models via Coordinate Descent.” *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.
- Friedman JH, Meulman JJ (2004). “Clustering Objects on Subsets of Attributes.” *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **66**(4), 815–849. doi:10.1111/j.1467-9868.2004.02059.x.
- Gablonsky JM, Kelley CT (2001). “A locally-biased form of the DIRECT algorithm.” *Journal of Global Optimization*, **21**, 27–37. doi:10.1023/a:1017930332101.
- Hahsler M, Piekenbrock M, Doran D (2019). “**dbscan**: Fast Density-Based Clustering with R.” *Journal of Statistical Software*, **91**(1), 1–30. doi:10.18637/jss.v091.i01.
- Halfvarson J, Kalla R, Adams A, Satsangi J, Nowak J (2022). “Whole blood expression profiling of patients with inflammatory bowel diseases in IBD-Character cohort.” URL <https://www.ebi.ac.uk/biostudies/arrayexpress/studies/E-MTAB-11349>.
- Hofner B, Boccuto L, Göker M (2015). “Controlling False Discoveries in High-dimensional Situations: Boosting with Stability Selection.” *BMC Bioinformatics*, **16**(1), 144. doi:10.1186/s12859-015-0575-3.
- Hornik K (2005). “A CLUE for CLUster Ensembles.” *Journal of Statistical Software*, **14**(12), 1–25. doi:10.18637/jss.v014.i12.

- John CR, Watson D, Russ D, Goldmann K, Ehrenstein M, Pitzalis C, Lewis M, Barnes M (2020). “M3C: Monte Carlo reference-based consensus clustering.” *Scientific reports*, **10**(1), 1816. doi:10.1038/s41598-020-58766-1.
- Johnson SG (2007). “The NLOpt nonlinear-optimization package.” <https://github.com/stevengj/nlopt>.
- Kampert MM, Meulman JJ, Friedman JH (2017). “**rCO**SA: A Software Package for Clustering Objects on Subsets of Attributes.” *Journal of Classification*, **34**(3), 514–547. doi:10.1007/s00357-017-9240-z.
- Lemon J (2006). “**plotrix**: A Package in the Red Light District of R.” *R-News*, **6**(4), 8–12. URL <https://CRAN.R-project.org/package=plotrix>.
- Liquet B, de Micheaux PL, Hejblum BP, Thiébaud R (2015). “Group and Sparse Group Partial Least Square Approaches Applied in Genomics Context.” *Bioinformatics*, **32**(1), 35–42. doi:10.1093/bioinformatics/btv535.
- Maechler M, Rousseeuw P, Struyf A, Hubert M, Hornik K (2022). **cluster**: *Cluster Analysis Basics and Extensions*. URL <https://CRAN.R-project.org/package=cluster>.
- 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.
- Meinshausen N, Bühlmann P (2010). “Stability Selection.” *Journal of the Royal Statistical Society B (Statistical Methodology)*, **72**(4), 417–473. doi:10.1111/j.1467-9868.2010.00740.x.
- Michel Philipp Thomas Rusch KH, Strobl C (2018). “Measuring the Stability of Results From Supervised Statistical Learning.” *Journal of Computational and Graphical Statistics*, **27**(4), 685–700. doi:10.1080/10618600.2018.1473779.
- 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.
- Neale MC, Hunter MD, Pritikin JN, Zahery M, Brick TR, Kirkpatrick RM, Estabrook R, Bates TC, Maes HH, Boker SM (2016). “**OpenMx** 2.0: Extended Structural Equation and Statistical Modeling.” *Psychometrika*, **81**(2), 535–549. doi:10.1007/s11336-014-9435-8.
- 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.
- R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.

- Ross Jacobucci KJG, McArdle JJ (2016). “Regularized Structural Equation Modeling.” *Structural Equation Modeling: A Multidisciplinary Journal*, **23**(4), 555–566. doi:10.1080/10705511.2016.1154793.
- 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). doi:10.2202/1544-6115.1175.
- Scrucca L, Fop M, Murphy TB, Raftery AE (2016). “**mclust** 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models.” *The R Journal*, **8**, 289–317. doi:10.32614/RJ-2016-021.
- Shah RD, Samworth RJ (2013). “Variable Selection With Error Control: Another Look at Stability Selection.” *Journal of the Royal Statistical Society B (Statistical Methodology)*, **75**(1), 55–80. doi:10.1111/j.1467-9868.2011.01034.x.
- Shen H, Huang JZ (2008). “Sparse Principal Component Analysis via Regularized Low Rank Matrix Approximation.” *Journal of Multivariate Analysis*, **99**(6), 1015–1034. doi:10.1016/j.jmva.2007.06.007.
- Simon N, Friedman JH, Hastie T, Tibshirani R (2011). “Regularization Paths for Cox’s Proportional Hazards Model via Coordinate Descent.” *Journal of Statistical Software*, **39**(5), 1–13. doi:10.18637/jss.v039.i05.
- Sustik MA, Calderhead B, Clavel J (2023). **glassoFast**: *Fast Graphical LASSO*. URL <https://CRAN.R-project.org/package=glassoFast>.
- Therneau T, Atkinson B (2023). **rpart**: *Recursive Partitioning and Regression Trees*. URL <https://CRAN.R-project.org/package=rpart>.
- Tibshirani R (1996). “Regression Shrinkage and Selection Via the Lasso.” *Journal of the Royal Statistical Society B (Methodological)*, **58**(1), 267–288. doi:10.1111/j.2517-6161.1996.tb02080.x.
- Whitaker M, Elliott J, Bodinier B, Barclay W, Ward H, Cooke G, Donnelly CA, Chadeau-Hyam M, Elliott P (2022). “Variant-specific symptoms of COVID-19 in a study of 1,542,510 adults in England.” *Nature Communications*, **13**(1), 6856. doi:10.1038/s41467-022-34244-2.
- Wilkerson MD, Hayes DN (2010). “ConsensusClusterPlus: a class discovery tool with confidence assessments and item tracking.” *Bioinformatics*, **26**(12), 1572–1573. doi:10.1093/bioinformatics/btq170.
- 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. doi:10.1111/j.1467-9868.2005.00503.x.
- 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.

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