Package ‘stabs’

July 19, 2017

Title Stability Selection with Error Control
Version 0.6-3
Date 2017-07-19
Description Resampling procedures to assess the stability of selected variables with additional finite sample error control for high-dimensional variable selection procedures such as Lasso or boosting. Both, standard stability selection (Meinshausen & Buhlmann, 2010, <doi:10.1111/j.1467-9868.2010.00740.x>) and complementary pairs stability selection with improved error bounds (Shah & Samworth, 2013, <doi:10.1111/j.1467-9868.2011.01034.x>) are implemented. The package can be combined with arbitrary user specified variable selection approaches.

VignetteBuilder knitr
Depends R (>= 2.14.0), methods, stats, parallel
Imports graphics, grDevices, utils
Suggests glmnet, lars, mboost (> 2.3-0), gamboostLSS (>= 1.2-0), QUIC, TH.data, hdi, testthat, knitr, rmarkdown, igraph, huge
LazyData yes
License GPL-2
URL https://github.com/hofnerb/stabs
NeedsCompilation no
Author Benjamin Hofner [aut, cre], Torsten Hothorn [aut]
Maintainer Benjamin Hofner <benjamin.hofner@pei.de>
Repository CRAN
Date/Publication 2017-07-19 21:09:23 UTC

R topics documented:

  check_folds ................................................................. 2
  Fitting Functions ........................................................... 3
check_folds

Check if folds result from subsampling with $p = 0.5$.

**Description**

(Internal) function that checks if folds result from subsampling with $p = 0.5$ and adds complementary pairs if needed.

**Usage**

```r
check_folds(folds, B, n, sampling.type)
```

**Arguments**

- `folds`: a weight matrix that represents the subsamples.
- `B`: number of subsampling replicates.
- `n`: the number of observations; needed for internal checks.
- `sampling.type`: sampling type to be used.

**Details**

This is an internal function used to check if `folds` are specified correctly. For details (e.g. on arguments) see `stabsel`.

**Value**

A matrix containing the `folds`, possibly after adding the complementary pairs.

**References**


**See Also**

For details see `stabsel`. 
Fitting Functions

Description
Functions that fit a model until \( q \) variables are selected and that returns the indices (and names) of the selected variables.

Usage

## package lars:
```r
lars.lasso(x, y, q, ...)
lars.stepwise(x, y, q, ...)
```

## package glmnet:
```r
glmnet.lasso(x, y, q, type = c("conservative", "anticonservative"), ...)
glmnet.lasso_maxCoef(x, y, q, ...)
```  

Arguments

- \( x \) a matrix containing the predictors or an object of class "mboost".
- \( y \) a vector or matrix containing the outcome.
- \( q \) number of (unique) selected variables (or groups of variables depending on the model) that are selected on each subsample.
- \( \text{type} \) a character vector specifying if the number of selected variables per subsample is \( \leq q \) (\( \text{type} = \text{"conservative"} \)) or \( \geq q \) (\( \text{type} = \text{"anticonservative"} \)). The conservative version ensures that the PFER is controlled.
- \( ... \) additional arguments passed to the underlying fitting function. See the example on `glmnet.lasso_maxCoef` in `stabsel` for the specification of additional arguments via `stabsel`.

Details

All fitting functions are named after the package and the type of model that is fitted: `package_name.model`, e.g., `glmnet.lasso` stands for a lasso model that is fitted using the package `glmnet`.

`glmnet.lasso_maxCoef` fits a lasso model with a given penalty parameter and returns the \( q \) largest coefficients. If one wants to use `glmnet.lasso_maxCoef`, one must specify the penalty parameter `lambda` (via the `...` argument) or in `stabsel` via `args.fitfun(lambda = )`. Note that usually, the penalty parameter cannot be specified but is chosen such that \( q \) variables are selected. For an example on how to use `glmnet.lasso_maxCoef` see `stabsel`.

Value

A named list with elements

- `selected` logical. A vector that indicates which variable was selected.
See Also

stabsel for stability selection itself, and quic.graphical_model for stability selection for graphical models.

Examples

```r
if (require("TH.data")) {
  ## make data set available
  data("bodyfat", package = "TH.data")
} else {
  ## simulate some data if TH.data not available.
  ## Note that results are non-sense with this data.
  bodyfat <- matrix(rnorm(720), nrow = 72, ncol = 10)
}

if (require("lars")) {
  ## selected variables
  lars.lasso(bodyfat[, -2], bodyfat[,2], q = 3)$selected
  lars.stepwise(bodyfat[, -2], bodyfat[,2], q = 3)$selected
}

if (require("glmnet")) {
  glmnet.lasso(bodyfat[, -2], bodyfat[,2], q = 3)$selected
  ## selection path
  glmnet.lasso(bodyfat[, -2], bodyfat[,2], q = 3)$path

  ## Using the anticonservative glmnet.lasso (see args.fitfun):
  stab(glmnet <- stabsel(x = bodyfat[, -2], y = bodyfat[,2],
             fitfun = glmnet.lasso,
             args.fitfun = list(type = "anticonservative"),
             cutoff = 0.75, PFER = 1)
}
```

---

## Method to Extract Parameters

<table>
<thead>
<tr>
<th>Method to Extract Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>parameters(object)</code></td>
</tr>
</tbody>
</table>

**Description**

Extract stability selection parameters, i.e., tuning parameters, from a stabsel object.

**Usage**

```r
parameters(object)
```

## Parameters

- `path` logical. A matrix that indicates which variable was selected in which step. Each row represents one variable, the columns represent the steps.
plot.stabsel

## S3 method for class 'stabsel'

stabsel_parameters(p, ...)

### Arguments

- **object**: an object of class "stabsel" or "stabsel_parameters".
- **p**: an object of class "stabsel".
- **...**: additional arguments, currently not used.

### Value

An object of class stabsel_parameters with a special print method. See there for details.

### See Also

stabsel to run stability selection and stabsel_parameters for details on the parameters.

---

### Description

Display results of stability selection.

### Usage

```r
## S3 method for class 'stabsel'
plot(x, main = deparse(x$call), type = c("maxsel", "paths"),
     xlab = NULL, ylab = NULL, col = NULL, ymargin = 10, np = sum(x$max > 0),
     labels = NULL, ...)
```

```r
## S3 method for class 'stabsel'
print(x, decreasing = FALSE, print.all = TRUE, ...)
```

### Arguments

- **x**: object of class stabsel.
- **main**: main title for the plot.
- **type**: plot type: either stability paths ("paths") or a plot of the maximum selection frequency ("maxsel").
- **xlab, ylab**: labels for the x- and y-axis of the plot. Per default, sensible labels are used depending on the type of the plot.
- **col**: a vector of colors; Typically, one can specify a single color or one color for each variable. Per default, colors depend on the maximal selection frequency of the variable and range from grey to red.
y.margin (temporarily) specifies the y margin of of the plot in lines (see argument "mar" of function par). This only affects the right margin for type = "paths" and the left margin for type = "maxsel". Explicit user specified margins are kept and are not overwritten.

np number of variables to plot for the maximum selection frequency plot (type = "maxsel"); the first np variables with highest selection frequency are plotted.

labels variable labels for the plot; one label per variable / effect must be specified. Per default, the names of x$max$ are used.

decreasing logical. Should the selection frequencies be printed in descending order (TRUE) or in ascending order (FALSE)?

print.all logical. Should all selection frequencies be displayed or only those that are greater than zero?

... additional arguments to plot and print functions.

Details

This function implements the stability selection procedure by Meinshausen and Buehlmann (2010) and the improved error bounds by Shah and Samworth (2013).

Two of the three arguments cutoff, q and PFER must be specified. The per-family error rate (PFER), i.e., the expected number of false positives $E(V)$, where $V$ is the number of false positives, is bounded by the argument PFER.

As controlling the PFER is more conservative as controlling the family-wise error rate (FWER), the procedure also controls the FWER, i.e., the probability of selecting at least one non-influential variable (or model component) is less than PFER.

Value

An object of class stabsel with a special print method. The object has the following elements:

- phat selection probabilities.
- selected elements with maximal selection probability greater cutoff.
- max maximum of selection probabilities.
- cutoff cutoff used.
- q average number of selected variables used.
- PFER per-family error rate.
- sampling.type the sampling type used for stability selection.
- assumption the assumptions made on the selection probabilities.
- call the call.

References


**See Also**

`stabsel`

**Examples**

```r
if (require("TH.data")) {
  ## make data set available
  data("bodyfat", package = "TH.data")
} else {
  ## simulate some data if TH.data not available.
  ## Note that results are non-sense with this data.
  bodyfat <- matrix(rnorm(720), nrow = 72, ncol = 10)
}

## set seed
set.seed(1234)

#############################################################
# using stability selection with Lasso methods:
if (require("lars")) {
  (stab.lasso <- stabsel(x = bodyfat[, -2], y = bodyfat[,2],
                        fitfun = lars.lasso, cutoff = 0.75,
                        PFER = 1))
  par(mfrow = c(2, 1))
  plot(stab.lasso, ymargin = 6)
  opar <- par(mai = par("mai") * c(1, 1, 1, 2.7))
  plot(stab.lasso, type = "paths")
}
```

**Description**

Stability selection fit function for sparse inverse covariance using package QUIC.

**Usage**

```r
quic.graphical_model(x, y, q, ...)
getLamPath(max, min, len, log = FALSE)
```
Arguments

- **x**: data matrix
- **y**: data matrix
- **q**: number of variables
- **max**: maximum value for regularization (lambda)
- **min**: min value for lambda
- **len**: length of path
- **log**: log spacing

Details

This is a wrapper for **QUIC** to be used in stability selection. Pass it as the fit function to **stabsel**. If you supply your own graphical model fitter, please set the class to "graphical_model". **getLamPath** can be used to create a regularization path.

Value

A named list with elements

- **selected**: logical. A vector that indicates which variable was selected.
- **path**: logical. A matrix that indicates which variable was selected in which step. Each row represents one variable, the columns represent the steps.

Author(s)

Code contributed by Richard Beare.

See Also

**stabsel** for stability selection itself, and **fitfun** for other fitting functions.

Examples

```r
### Do not test the following code per default on CRAN as it takes some time to run:
if (require("huge") & require("QUIC")) {
  set.seed(10010)
  dat.hubs <- huge.generator(n = 1000, d = 40, graph = "hub")
  s.hubs <- stabsel(x = dat.hubs$data, fitfun = quic.graphical_model,
                    cutoff = 0.75, PFER = 10)
}
```
**run_stabsel**

*Run Stability Selection*

---

### Description

(Internal) function that is used to run stability selection (i.e. to apply the fit-function to the subsamples. This function is not intended to be directly called.

### Usage

```r
run_stabsel(fitter, args.fitter, n, p, cutoff, q, PFER, folds, B, assumption, sampling.type, papply, verbose, FWER, eval, names, mc.preschedule = FALSE, ...)
```

### Arguments

- **fitter**: a function to fit the model on subsamples. See argument `fitfun` of `stabsel` for details.
- **args.fitter**: a named list containing additional arguments that are passed to `fitter`. See argument `args.fitfun stabsel` for details.
- **n**: the number of observations; needed for internal checks.
- **p**: number of possible predictors (including intercept if applicable).
- **cutoff**: cutoff between 0.5 and 1.
- **q**: number of (unique) selected variables (or groups of variables depending on the model) that are selected on each subsample.
- **PFER**: upper bound for the per-family error rate.
- **folds**: a weight matrix that represents the subsamples.
- **B**: number of subsampling replicates.
- **assumption**: distributional assumption.
- **sampling.type**: sampling type to be used.
- **papply**: (parallel) apply function.
- **verbose**: logical (default: TRUE) that determines whether warnings should be issued.
- **FWER**: deprecated. Only for compatibility with older versions, use PFER instead.
- **eval**: logical. Determines whether stability selection is evaluated.
- **names**: variable names that are used to label the results.
- **mc.preschedule**: preschedule tasks?
- **...**: additional arguments to be passed to next function.
Details

This is an internal function that fits the actual models to the subsamples, i.e., this is the work horse that runs stability selection. Usually, one should use stabsel, which internally calls run_stabsel. run_stabsel can be used by expert users to implement stability selection methods for new model types.

For details (e.g. on arguments) see stabsel.

Value

An object of class stabsel with the following elements:

- pihat: selection probabilities.
- selected: elements with maximal selection probability greater cutoff.
- max: maximum of selection probabilities.
- cutoff: cutoff used.
- q: average number of selected variables used.
- PFER: per-family error rate.
- p: the number of effects subject to selection.
- sampling.type: the sampling type used for stability selection.
- assumption: the assumptions made on the selection probabilities.

References


See Also

For details see stabsel.

---

**selected**

**Method to Extract Selected Variables**

**Description**

Extract selected variables from a stabsel object.

**Usage**

```r
selected(object, ...)  
```

## S3 method for class 'stabsel'

```r
selected(object, ...)  
```
stabsel

Arguments

object an object of class "stabsel".
... additional arguments passed to specific selected methods.

Details

The ids of variables selected during the stability selection process can be extracted using `selected()`.

---

stabsel Stability Selection

Description

Selection of influential variables or model components with error control.

Usage

```r
## generic stability selection function
stabsel(x, ...)

## a method to fit models with stability selection
## S3 method for class 'matrix'
stabsel(x, y, fitfun = lars.lasso,
    args.fitfun = list(), cutoff, q, PFER,
    folds = subsample(rep(1, nrow(x)), B = B),
    B = ifelse(sampling.type == "MB", 100, 50),
    assumption = c("unimodal", "r-concave", "none"),
    sampling.type = c("SS", "MB"),
    papply = mclapply, mc.preschedule = FALSE,
    verbose = TRUE, FWER, eval = TRUE, ...)

## essentially a wrapper for data.frames (see details)
## S3 method for class 'data.frame'
stabsel(x, y, intercept = FALSE, ...)
```

Arguments

x a matrix or a data.frame containing the predictors.
y a vector or matrix containing the outcome.
intercept logical. If x is a data.frame, this argument determines if the resulting model matrix should contain a separate intercept or not.
fitfun a function that takes the arguments x, y as above, and additionally the number of variables to include in each model q. The function then needs to fit the model and to return a logical vector that indicates which variable was selected (among the q selected variables).
stabsel

args.fitfun a named list containing additional arguments that are passed to the fitting function; see also argument args in do.call.
cutoff cutoff between 0.5 and 1. Preferably a value between 0.6 and 0.9 should be used.
q number of (unique) selected variables (or groups of variables depending on the model) that are selected on each subsample.
PFER upper bound for the per-family error rate. This specifies the amount of falsely selected base-learners, which is tolerated. See details.
folds a weight matrix with number of rows equal to the number of observations, see subsample. Usually one should not change the default here as subsampling with a fraction of $1/2$ is needed for the error bounds to hold. One usage scenario where specifying the folds by hand might be the case when one has dependent data (e.g. clusters) and thus wants to draw clusters (i.e., multiple rows together) not individuals.

assumption Defines the type of assumptions on the distributions of the selection probabilities and simultaneous selection probabilities. Only applicable for sampling.type = "SS". For sampling.type = "MB" we always use code"none".
sampling.type use sampling scheme of of Shah & Samworth (2013), i.e., with complementary pairs (sampling.type = "SS"), or the original sampling scheme of Meinshausen & Buehlmann (2010).
B number of subsampling replicates. Per default, we use 50 complementary pairs for the error bounds of Shah & Samworth (2013) and 100 for the error bound derived in Meinshausen & Buehlmann (2010). As we use $B$ complementray pairs in the former case this leads to $2B$ subsamples.
papply (parallel) apply function, defaults to mclapply. Alternatively, parlapply can be used. In the latter case, usually more setup is needed (see example of cvrisk for some details).
mc.preschedule preschedule tasks if papply = mclapply (default: mc.preschedule = FALSE)? For details see mclapply.
verbose logical (default: TRUE) that determines whether warnings should be issued.
FWER deprecated. Only for compatibility with older versions, use PFER instead.
eval logical. Determines whether stability selection is evaluated (eval = TRUE; default) or if only the parameter combination is returned.
...

Details

This function implements the stability selection procedure by Meinshausen and Buehlmann (2010) and the improved error bounds by Shah and Samworth (2013). For details see also Hofner et al. (2014). The error bounds are implemented in the function stabsel_parameters. Two of the three arguments cutoff, q and PFER must be specified. The per-family error rate (PFER), i.e., the expected number of false positives $E(V)$, where $V$ is the number of false positives, is bounded by the argument PFER.
As controlling the PFER is more conservative as controlling the family-wise error rate (FWER), the procedure also controls the FWER, i.e., the probability of selecting at least one non-influential variable (or model component) is less than PFER.

Predefined fitfun functions exist but more can be easily implemented. Note that stepwise regression methods are usually not advised as they tend to be relatively unstable. See example below.

The function stabsel for data frames is essentially just a wrapper to the matrix function with the same arguments. The only difference is that in a pre-processing step, the data set is converted to a model matrix using the function model.matrix. The additional argument intercept determines if an explicit intercept should be added to the model matrix. This is often not necessary but depends on the fitfun.

Value

An object of class stabsel with a special print method. The object has the following elements:

- **phat**: selection probabilities.
- **selected**: elements with maximal selection probability greater cutoff.
- **max**: maximum of selection probabilities.
- **cutoff**: cutoff used.
- **q**: average number of selected variables used.
- **PFER**: (realized) upper bound for the per-family error rate.
- **specifiedPFER**: specified upper bound for the per-family error rate.
- **p**: the number of effects subject to selection.
- **B**: the number of subsamples.
- **sampling.type**: the sampling type used for stability selection.
- **assumption**: the assumptions made on the selection probabilities.
- **call**: the call.

References


See Also

* stabsel_parameters* for the computation of error bounds, *stabsel.stabsel* for the fast recomputation of parameters of a fitted stabsel object, *fitfun* for available fitting functions and *plot.stabsel* for available plot functions
Examples

```r
if (require("TH.data")) {
  ## make data set available
  data("bodyfat", package = "TH.data")
} else {
  ## simulate some data if TH.data not available.
  ## Note that results are non-sense with this data.
  bodyfat <- matrix(rnorm(720), nrow = 72, ncol = 10)
}

## set seed
set.seed(1234)

### using stability selection with Lasso methods:

if (require("lars")) {
  (stab.lasso <- stabsel(x = bodyfat[, -2], y = bodyfat[,2],
                        fitfun = lars.lasso, cutoff = 0.75,
                        PFER = 1))
  (stab.stepwise <- stabsel(x = bodyfat[, -2], y = bodyfat[,2],
                           fitfun = lars.stepwise, cutoff = 0.75,
                           PFER = 1))
  par(mfrow = c(2, 1))
  plot(stab.lasso, main = "Lasso")
  plot(stab.stepwise, main = "Stepwise Selection")
  ## --> stepwise selection seems to be quite unstable even in this low
  ## dimensional example!
}

## set seed (again to make results comparable)
set.seed(1234)
if (require("glmnet")) {
  (stab.glmnet <- stabsel(x = bodyfat[, -2], y = bodyfat[,2],
                          fitfun = glmnet.lasso, cutoff = 0.75,
                          PFER = 1))
  par(mfrow = c(2, 1))
  plot(stab.glmnet, main = "Lasso (glmnet)")
  if (exists("stab.lasso"))
    plot(stab.lasso, main = "Lasso (lars)")
}

## Select variables with maximum coefficients based on lasso estimate

set.seed(1234) # reset seed
if (require("glmnet")) {
  ## use cross-validated lambda
  lambda.min <- cv.glmnet(x = as.matrix(bodyfat[, -2]), y = bodyfat[,2])$lambda.min
  (stab.maxCoef <- stabsel(x = bodyfat[, -2], y = bodyfat[,2],
                           fitfun = glmnet.lasso_maxCoef,
```

---

The above code demonstrates the use of the `stabsel` function from the `stabsel` package in R for stability selection with Lasso methods. It includes examples of using both `lars` and `glmnet` packages for Lasso regression and selecting variables with maximum coefficients based on the lasso estimate.
# specify additional parameters to fitfun
args.fitfun = list(lambda = lambda.min,
cutoff = 0.75, PFER = 1))

## WARNING: Using a fixed penalty (lambda) is usually not permitted and
## not sensible. See ?fitfun for details.

## now compare standard lasso with "maximal parameter estimates" from lasso
par(mfrow = c(2,1))
plot(stab.maxCoeff, main = "Lasso (glmnet; Maximum Coefficients")
plot(stab.glmnet, main = "Lasso (glmnet")
## --> very different results.
}

#########################################################################
### using stability selection directly on computed boosting models
### from mboost

if (require("mboost")) {
  ### low-dimensional example
  mod <- glmboost(DEXfat ~ ., data = bodyfat)
  
  ## compute cutoff ahead of running stabsel to see if it is a sensible
  ## parameter choice.
  ## p = ncol(bodyfat) - 1 (= Outcome) + 1 (= Intercept)
  stabsel_parameters(q = 3, PFER = 1, p = ncol(bodyfat) - 1 + 1,
    sampling.type = "MB")
  
  ## the same:
  stabsel(mod, q = 3, PFER = 1, sampling.type = "MB", eval = FALSE)

  ### Do not test the following code per default on CRAN as it takes some time to run:
  ### now run stability selection
  (sbody <- stabsel(mod, q = 3, PFER = 1, sampling.type = "MB")
  opar <- par(mai = par("mai") * c(1, 1, 1, 2.7))
  plot(sbody)
  par(opar)
  plot(sbody, type = "maxsel", ymargin = 6)
}

---

stabsel.stabsel  Change Parameters of Stability Selection

Description

Method to change the parameters cutoff, PFER and assumption of stability selection that can be altered without the need to re-run the subsampling process.
Usage

## S3 method for class 'stabsel'
stabsel(x, cutoff, PFER, assumption = x$assumption, ...)

Arguments

- `x`: an object that results from a call to `stabsel`.
- `cutoff`: cutoff between 0.5 and 1. Preferably a value between 0.6 and 0.9 should be used.
- `PFER`: upper bound for the per-family error rate. This specifies the amount of falsely selected base-learners, which is tolerated. See details.
- `assumption`: Defines the type of assumptions on the distributions of the selection probabilities and simultaneous selection probabilities. Only applicable for `sampling.type = "SS"`. For `sampling.type = "MB"` we always use code 'none'.
- `...`: additional arguments that are currently ignored.

Details

This function allows to alter the parameters `cutoff`, `PFER` and `assumption` of a fitted stability selection result. All other parameters are re-used from the original stability selection results. The missing parameter is computed and the selected variables are updated accordingly.

Value

An object of class `stabsel`. For details see there.

See Also

`stabsel` for the generic function, `stabsel_parameters` for the computation of error bounds, `fitfun` for available fitting functions and `plot.stabsel` for available plot functions

Examples

```r
if (require("TH.data")) {
  ## make data set available
  data("bodyfat", package = "TH.data")
} else {
  ## simulate some data if TH.data not available.
  ## Note that results are non-sense with this data.
  bodyfat <- matrix(rnorm(720), nrow = 72, ncol = 10)
}

## set seed
set.seed(1234)

# using stability selection with Lasso methods:
if (require("lars")) {
  ```
stabsel_parameters

(stab.lasso <- stabsel(x = bodyfat[, -2], y = bodyfat[, 2],
    fitfun = lars.lasso, cutoff = 0.75,
    PFER = 1))

par(mfrow = c(2, 1))
plot(stab.lasso)

## now change the PFER and the assumption:
(stab.lasso_cf0.93_rconc <- stabsel(stab.lasso, cutoff = 0.93,
    assumption = "r-concave"))
plot(stab.lasso_cf0.93_rconc)
## the cutoff did change and hence the PFER and the selected
## variables

stabsel_parameters

Compute Error Bounds for Stability Selection

Description

Compute the missing parameter from the two given parameters in order to assess suitability of the parameter constellation

Usage

stabsel_parameters(p, ...)

## Default S3 method:
stabsel_parameters(p, cutoff, q, PFER,
    B = ifelse(sampling.type == "MB", 100, 50),
    assumption = c("unimodal", "r-concave", "none"),
    sampling.type = c("SS", "MB"),
    verbose = FALSE, FWER, ...)

## S3 method for class 'stabsel_parameters'
print(x, heading = TRUE, ...)

Arguments

p number of possible predictors (including intercept if applicable).
cutoff cutoff between 0.5 and 1. Preferably a value between 0.6 and 0.9 should be used.
q number of (unique) selected variables (or groups of variables depending on the model) that are selected on each subsample.
PFER upper bound for the per-family error rate. This specifies the amount of falsely selected base-learners, which is tolerated. See details.
The `stabsel_parameters` function implements the error bounds for stability selection by Meinshausen and Buehlmann (2010) and the improved error bounds by Shah and Samworth (2013). For details see also Hofner et al. (2014).

Two of the three arguments `cutoff`, `q`, and `PFER` must be specified. The per-family error rate (PFER), i.e., the expected number of false positives $E(V)$, where $V$ is the number of false positives, is bounded by the argument `PFER`.

For more details see also `stabsel`.

The function accepts the following arguments:

- **B**
  - number of subsampling replicates. Per default, we use 50 complementary pairs for the error bounds of Shah & Samworth (2013) and 100 for the error bound derived in Meinshausen & Buehlmann (2010). As we use $B$ complementray pairs in the former case this leads to $2B$ subsamples.

- **assumption**
  - Defines the type of assumptions on the distributions of the selection probabilities and simultaneous selection probabilities. Only applicable for `sampling.type = "SS"`. For `sampling.type = "MB"` we always use code "none".

- **sampling.type**
  - use sampling scheme of of Shah & Samworth (2013), i.e., with complementarty pairs (`sampling.type = "SS"`), or the original sampling scheme of Meinshausen & Buehlmann (2010).

- **verbose**
  - logical (default: `TRUE`) that determines whether warnings should be issued.

- **FWER**
  - deprecated. Only for compatibility with older versions, use PFER instead.

- **x**
  - an object of class "stabsel_parameters".

- **heading**
  - logical. Specifies if a heading line should be printed.

- **...**
  - additional arguments to be passed to next function.

### Details

This function implements the error bounds for stability selection by Meinshausen and Buehlmann (2010) and the improved error bounds by Shah and Samworth (2013). For details see also Hofner et al. (2014).

Two of the three arguments `cutoff`, `q`, and `PFER` must be specified. The per-family error rate (PFER), i.e., the expected number of false positives $E(V)$, where $V$ is the number of false positives, is bounded by the argument `PFER`.

For more details see also `stabsel`.

### Value

An object of class `stabsel_parameters` with a special `print` method. The object has the following elements:

- **cutoff**
  - cutoff used.

- **q**
  - average number of selected variables used.

- **PFER**
  - (realized) upper bound for the per-family error rate.

- **specifiedPFER**
  - specified upper bound for the per-family error rate.

- **p**
  - the number of effects subject to selection.

- **B**
  - the number of subsamples.

- **sampling.type**
  - the sampling type used for stability selection.

- **assumption**
  - the assumptions made on the selection probabilities.
References


See Also

For more details see also `stabsel`.

```
subsample <- function(weights, B = 100, strata = NULL) {
  # Set up weight matrix for subsampling with sample proportion 1/2 to be used with stabsel.
  # If strata is defined sampling is performed in each stratum separately thus preserving the distribution of the strata variable in each fold.
  
  # Arguments
  # weights: a numeric vector of weights for the model to be cross-validated.
  # B: number of folds, per default 25 for bootstrap and subsampling and 10 for kfold.
  # strata: a factor of the same length as weights for stratification.

  # Details
  # The function `subsample` can be used to build an appropriate weight matrix to be used with stabsel.
  # See there for more details.

  subsample <- if (is.null(strata)) {
    rep(1, length(weights))
  } else {
    replicate(B, rep(1, length(weights)), simplify = FALSE)
  }

  return(subsample)
}
```

Examples

```r
# just a low-dimensional example
subsample(weights = rep(1, 10), B = 50)
```
Index

*Topic helper
  check_folds, 2
  run_stabsel, 9
  stabsel_parameters, 17

*Topic methods
  parameters, 4
  selected, 10

*Topic models
  Fitting Functions, 3
  quic.graphical_model, 7

*Topic nonlinear
  Fitting Functions, 3
  quic.graphical_model, 7

*Topic nonparametric
  Fitting Functions, 3
  plot.stabsel, 5
  quic.graphical_model, 7
  stabsel, 11
  stabsel.stabsel, 15
  subsample, 19

check_folds, 2
cvrisk, 12

data.frame, 11, 13
do.call, 12

fitfun, 8, 13, 16
  fitfun(Fitting Functions), 3
  fitfuns, 13
  fitfuns(Fitting Functions), 3
  Fitting Functions, 3

getLamPath (quic.graphical_model), 7
glmnet.lasso(Fitting Functions), 3
glmnet.lasso_maxCoef(Fitting Functions), 3

lars.lasso(Fitting Functions), 3
lars.stepwise(Fitting Functions), 3

matrix, 11, 13
mclapply, 12
model.matrix, 13
par, 6
parameters, 4
plot(plot.stabsel), 5
plot.stabsel, 5, 13, 16
print.stabsel(plot.stabsel), 5
print.stabsel_parameters
  (stabsel_parameters), 17
QUIC, 8
quic.graphical_model, 4, 7

run_stabsel, 9
selected, 10
stabsel, 2–5, 7–10, 11, 16, 18, 19
stabsel.stabsel, 13, 15
stabsel_parameters, 5, 12, 13, 16, 17
stabsel_parameters.stabsel
  (parameters), 4
subsample, 12, 19