

# Package: SISIR (via r-universe)

August 20, 2024

**Type** Package

**Title** Select Intervals Suited for Functional Regression

**Version** 0.2.3

**Date** 2024-08-16

**Maintainer** Nathalie Vialaneix <nathalie.vialaneix@inrae.fr>

**Description** Interval fusion and selection procedures for regression with functional inputs. Methods include a semiparametric approach based on Sliced Inverse Regression (SIR), as described in <[doi:10.1007/s11222-018-9806-6](https://doi.org/10.1007/s11222-018-9806-6)> (standard ridge and sparse SIR are also included in the package) and a random forest based approach, as described in <[doi:10.1002/sam.11705](https://doi.org/10.1002/sam.11705)>.

**Depends** R (>= 3.5.0), foreach, doParallel, graphics, stats

**URL** <https://forgemia.inra.fr/sfcb/sisir>

**BugReports** <https://forgemia.inra.fr/sfcb/sisir/-/issues>

**Imports** Matrix, expm, RSpectra, glmnet, Boruta, CORElearn, dplyr, mixOmics, purrr, ranger, tidyr, tidyselect, adjclust, magrittr, rlang, ggplot2, aricode, dendextend, reshape2, RColorBrewer

**Suggests** testthat

**License** GPL (>= 2)

**RoxygenNote** 7.3.1

**Encoding** UTF-8

**Repository** CRAN

**NeedsCompilation** no

**Author** Victor Picheny [aut], Remi Servien [aut], Nathalie Vialaneix [aut, cre] (<<https://orcid.org/0000-0003-1156-0639>>)

**Date/Publication** 2024-08-16 12:50:02 UTC

## Contents

project . . . . .	2
ridgeRes . . . . .	3
ridgeSIR . . . . .	4
sfcB . . . . .	5
SFCB-class . . . . .	7
SISIR . . . . .	9
SISIRres . . . . .	11
sparseRes . . . . .	12
sparseSIR . . . . .	12
truffles . . . . .	14
tune.ridgeSIR . . . . .	15

<b>Index</b>	<b>17</b>
--------------	-----------

---

project	<i>sparse SIR</i>
---------	-------------------

---

## Description

project performs the projection on the sparse EDR space (as obtained by the [glmnet](#))

## Usage

```
## S3 method for class 'sparseRes'
project(object)

project(object)
```

## Arguments

object            an object of class sparseRes as obtained from the function [sparseSIR](#)

## Details

The projection is obtained by the function [predict.glmnet](#).

## Value

a matrix of dimension  $n \times d$  with the projection of the observations on the  $d$  dimensions of the sparse EDR space

## Author(s)

Victor Picheny, <[victor.picheny@inrae.fr](mailto:victor.picheny@inrae.fr)>  
 Remi Servien, <[remi.servien@inrae.fr](mailto:remi.servien@inrae.fr)>  
 Nathalie Vialaneix, <[nathalie.vialaneix@inrae.fr](mailto:nathalie.vialaneix@inrae.fr)>

**References**

Picheny, V., Servien, R. and Villa-Vialaneix, N. (2016) Interpretable sparse SIR for digitized functional data. *Statistics and Computing*, **29**(2), 255–267.

**See Also**

[sparseSIR](#)

**Examples**

```
set.seed(1140)
tsteps <- seq(0, 1, length = 200)
nsim <- 100
simulate_bm <- function() return(c(0, cumsum(rnorm(length(tsteps)-1, sd=1))))
x <- t(replicate(nsim, simulate_bm()))
beta <- cbind(sin(tsteps*3*pi/2), sin(tsteps*5*pi/2))
beta[[(tsteps < 0.2) | (tsteps > 0.5)], 1] <- 0
beta[[(tsteps < 0.6) | (tsteps > 0.75)], 2] <- 0
y <- log(abs(x %*% beta[,1]) + 1) + sqrt(abs(x %*% beta[,2]))
y <- y + rnorm(nsim, sd = 0.1)

res_ridge <- ridgeSIR(x, y, H = 10, d = 2)
res_sparse <- sparseSIR(res_ridge, rep(1, ncol(x)))
proj_data <- project(res_sparse)
```

---

ridgeRes

*Print ridgeRes object*


---

**Description**

Print a summary of the result of [ridgeSIR](#) ( `ridgeRes` object)

**Usage**

```
## S3 method for class 'ridgeRes'
summary(object, ...)
```

```
## S3 method for class 'ridgeRes'
print(x, ...)
```

**Arguments**

<code>object</code>	a <code>ridgeRes</code> object
<code>...</code>	not used
<code>x</code>	a <code>ridgeRes</code> object

**Author(s)**

Victor Picheny, <victor.picheny@inrae.fr>  
 Remi Servien, <remi.servien@inrae.fr>  
 Nathalie Vialaneix, <nathalie.vialaneix@inrae.fr>

**See Also**

[ridgeSIR](#)

---

ridgeSIR

*ridge SIR*

---

**Description**

ridgeSIR performs the first step of the method (ridge regularization of SIR)

**Usage**

```
ridgeSIR(x, y, H, d, mu2 = NULL)
```

**Arguments**

x	explanatory variables (numeric matrix or data frame)
y	target variable (numeric vector)
H	number of slices (integer)
d	number of dimensions to be kept
mu2	ridge regularization parameter (numeric, positive)

**Details**

SI-SIR

**Value**

S3 object of class `ridgeRes`: a list consisting of

`EDR` the estimated EDR space (a  $p \times d$  matrix)

`condC` the estimated slice projection on EDR (a  $d \times H$  matrix)

`eigenvalues` the eigenvalues obtained during the generalized eigendecomposition performed by SIR

`parameters` a list of hyper-parameters for the method:

`H` number of slices

`d` dimension of the EDR space

`mu2` regularization parameter for the ridge penalty

`utils` useful outputs for further computations:

Sigma covariance matrix for x  
 slices slice number for all observations  
 invsqrtS value of the inverse square root of the regularized covariance matrix for x

### Author(s)

Victor Picheny, <victor.picheny@inrae.fr>  
 Remi Servien, <remi.servien@inrae.fr>  
 Nathalie Vialaneix, <nathalie.vialaneix@inrae.fr>

### References

Picheny, V., Servien, R. and Villa-Vialaneix, N. (2019) Interpretable sparse SIR for digitized functional data. *Statistics and Computing*, **29**(2), 255–267.

### See Also

[sparseSIR](#), [SISIR](#), [tune.ridgeSIR](#)

### Examples

```

set.seed(1140)
tsteps <- seq(0, 1, length = 50)
simulate_bm <- function() return(c(0, cumsum(rnorm(length(tsteps)-1, sd=1))))
x <- t(replicate(50, simulate_bm()))
beta <- cbind(sin(tsteps*3*pi/2), sin(tsteps*5*pi/2))
y <- log(abs(x %*% beta[,1])) + sqrt(abs(x %*% beta[,2]))
y <- y + rnorm(50, sd = 0.1)
res_ridge <- ridgeSIR(x, y, H = 10, d = 2, mu2 = 10^8)
res_ridge

```

---

sfcf

*sfcf*

---

### Description

sfcf performs interval selection based on random forests

### Usage

```

sfcf(
  X,
  Y,
  group.method = c("adjclust", "cclustofvar"),
  summary.method = c("pls", "basics", "cclustofvar"),
  selection.method = c("none", "boruta", "relief"),
  at = round(0.15 * ncol(X)),

```

```

range.at = NULL,
seed = NULL,
repeats = 5,
keep.time = TRUE,
verbose = TRUE,
parallel = FALSE
)

```

### Arguments

<code>X</code>	input predictors (matrix or data.frame)
<code>Y</code>	target variable (vector whose length is equal to the number of rows in X)
<code>group.method</code>	group method. Default to "adjclust"
<code>summary.method</code>	summary method. Default to "pls"
<code>selection.method</code>	selection method. Default to "none" (no selection performed)
<code>at</code>	number of groups targeted for output results (integer). Not used when <code>range.at</code> is not NULL
<code>range.at</code>	(vector of integer) sequence of the numbers of groups for output results
<code>seed</code>	random seed (integer)
<code>repeats</code>	number of repeats for the final random forest computation
<code>keep.time</code>	keep computational times for each step of the method? (logical; default to TRUE)
<code>verbose</code>	print messages? (logical; default to TRUE)
<code>parallel</code>	not implemented yet

### Value

an object of class "SFCB" with elements:

<code>dendro</code>	a dendrogram corresponding to the method chosen in <code>group.method</code>
<code>groups</code>	a list of length <code>length(range.at)</code> (or of length 1 if <code>range.at == NULL</code> ) that contains the clusterings of input variables for the selected group numbers
<code>summaries</code>	a list of the same length than <code>\$groups</code> that contains the summarized predictors according to the method chosen in <code>summary.methods</code>
<code>selected</code>	a list of the same length than <code>\$groups</code> that contains the names of the variable selected by <code>selection.method</code> if it is not equal to "none"
<code>mse</code>	a data.frame with <code>repeats × length(\$groups)</code> rows that contains Mean Squared Errors of the repeats random forests fitted for each number of groups
<code>importance</code>	a list of the same length than <code>\$groups</code> that contains a data.frame providing variable importances for the variables in selected groups in repeats columns (one for each iteration of the random forest method). When <code>summary.method == "basics"</code> , importance for mean and sd are provided in separated columns, in which case, the number of columns is equal to <code>2repeats</code>
<code>computational.times</code>	a vector with 4 values corresponding to the computational times of (respectively) the group, summary, selection, and RF steps. Only if <code>keep.time == TRUE</code>
<code>call</code>	function call

**Author(s)**

Remi Servien, <remi.servien@inrae.fr>  
Nathalie Vialaneix, <nathalie.vialaneix@inrae.fr>

**References**

Servien, R. and Vialaneix, N. (2024) A random forest approach for interval selection in functional regression. *Statistical Analysis and Data Mining*, **17**(4), e11705. doi:10.1002/sam.11705

**Examples**

```
data(truffles)
out1 <- sfcf(rainfall, truffles, group.method = "adjclust",
            summary.method = "pls", selection.method = "relief")
out2 <- sfcf(rainfall, truffles, group.method = "adjclust",
            summary.method = "basics", selection.method = "none",
            range.at = c(5, 7))
```

---

SFCB-class

*Methods for SFCB objects*

---

**Description**

Print, plot, manipulate or compute quality for outputs of the `sfcf` function (SFCB object)

**Usage**

```
## S3 method for class 'SFCB'
summary(object, ...)

## S3 method for class 'SFCB'
print(x, ...)

## S3 method for class 'SFCB'
plot(
  x,
  ...,
  plot.type = c("dendrogram", "selection", "importance", "quality"),
  sel.type = c("importance", "selection"),
  threshold = "none",
  shape.imp = c("boxplot", "histogram"),
  quality.crit = "mse"
)

extract_at(object, at)

quality(object, ground_truth, threshold = NULL)
```

**Arguments**

<code>object</code>	a SFCB object
<code>...</code>	not used
<code>x</code>	a SFCB object
<code>plot.type</code>	type of the plot. Default to "dendrogram" (see Details)
<code>sel.type</code>	when <code>plot.type == "selection"</code> , criterion on which to base the selection. Default to "importance"
<code>threshold</code>	numeric value. If not NULL, selection of variables to compute qualities is based on a threshold of importance values <code>extract_at</code>
<code>shape.imp</code>	when <code>plot.type == "importance"</code> , type of plot to represent the importance. Default to "boxplot"
<code>quality.crit</code>	character vector (length 1 or 2) indicating one or two quality criteria to display. The values have to be taken in {"mse", "time", "Precision", "Recall", "ARI", "NMI"}. If "time" is chosen, it can not be associated with any other criterion
<code>at</code>	numeric vector. Set of the number of intervals to extract for
<code>ground_truth</code>	numeric vector of ground truth. Target variables to compute qualities correspond to non-zero entries of this vector

**Details**

The plot functions can be used in four different ways to extract information from the SFCB object:

- `plot.type == "dendrogram"` displays the dendrogram obtained at the clustering step of the method. Depending on the cases, the dendrogram comes with additional information on clusters, variable selections and/or importance values;
- `plot.type == "selection"` displays either the evolution of the importance for the simulation with the best (smallest) MSE for each time step in the range of the functional predictor or the evolution of the selected intervals along the whole range of the functional prediction also for the best MSE;
- `plot.type == "importance"` displays a summary of the importance values over the whole range of the functional predictor and for the different experiments. This summary can take the form of a boxplot or of an histogram;
- `plot.type == "quality"` displays one or two quality distribution with respect to the different experiments and different number of intervals.

**Author(s)**

Remi Servien, <remi.servien@inrae.fr>  
 Nathalie Vialaneix, <nathalie.vialaneix@inrae.fr>

**References**

Servien, R. and Vialaneix, N. (2023) A random forest approach for interval selection in functional regression. Preprint.



**See Also**[sfcb](#)**Examples**

```

data(truffles)
out1 <- sfcb(rainfall, truffles, group.method = "adjclust",
            summary.method = "pls", selection.method = "relief")
summary(out1)

plot(out1)
plot(out1, plot.type = "selection")
plot(out1, plot.type = "importance")

out2 <- sfcb(rainfall, truffles, group.method = "adjclust",
            summary.method = "basics", selection.method = "none",
            range.at = c(5, 7))
out3 <- extract_at(out2, at = 6)
summary(out3)

```

SISIR

*Interval Sparse SIR***Description**

SISIR performs an automatic search of relevant intervals

**Usage**

```

SISIR(
  object,
  inter_len = rep(1, nrow(object$EDR)),
  sel_prop = 0.05,
  itermax = Inf,
  minint = 2,
  parallel = TRUE,
  ncores = NULL
)

```

**Arguments**

<code>object</code>	an object of class <code>ridgeRes</code> as obtained from the function <a href="#">ridgeSIR</a>
<code>inter_len</code>	(numeric) vector with interval lengths for the initial state. Default is to set one interval for each variable (all intervals have length 1)
<code>sel_prop</code>	fraction of the coefficients that will be considered as strong zeros and strong non zeros. Default to 0.05

<code>itermax</code>	maximum number of iterations. Default to Inf
<code>minint</code>	minimum number of intervals. Default to 2
<code>parallel</code>	whether the computation should be performed in parallel or not. Logical. Default is FALSE
<code>ncores</code>	number of cores to use if <code>parallel = TRUE</code> . If left to NULL, all available cores minus one are used

### Details

Different quality criteria used to select the best models among a list of models with different interval definitions. Quality criteria are: log-likelihood (`loglik`), cross-validation error as provided by the function `glmnet`, two versions of the AIC (AIC and AIC2) and of the BIC (BIC and BIC2) in which the number of parameters is either the number of non null intervals or the number of non null parameters with respect to the original variables

### Value

S3 object of class SISIR: a list consisting of

`sEDR` the estimated EDR spaces (a list of  $p \times d$  matrices)

`alpha` the estimated shrinkage coefficients (a list of vectors)

`intervals` the interval lengths (a list of vectors)

`quality` a data frame with various qualities for the model. The chosen quality measures are the same than for the function `sparseSIR` plus the number of intervals `nbint`

`init_sel_prop` initial fraction of the coefficients which are considered as strong zeros or strong non zeros

`rSIR` same as the input object

### Author(s)

Victor Picheny, <[victor.picheny@inrae.fr](mailto:victor.picheny@inrae.fr)>

Remi Servien, <[remi.servien@inrae.fr](mailto:remi.servien@inrae.fr)>

Nathalie Vialaneix, <[nathalie.vialaneix@inrae.fr](mailto:nathalie.vialaneix@inrae.fr)>

### References

Picheny, V., Servien, R. and Villa-Vialaneix, N. (2016) Interpretable sparse SIR for digitized functional data. *Statistics and Computing*, **29**(2), 255–267.

### See Also

[ridgeSIR](#), [sparseSIR](#)

**Examples**

```

set.seed(1140)
tsteps <- seq(0, 1, length = 200)
nsim <- 100
simulate_bm <- function() return(c(0, cumsum(rnorm(length(tsteps)-1, sd=1))))
x <- t(replicate(nsim, simulate_bm()))
beta <- cbind(sin(tsteps*3*pi/2), sin(tsteps*5*pi/2))
beta[((tsteps < 0.2) | (tsteps > 0.5)), 1] <- 0
beta[((tsteps < 0.6) | (tsteps > 0.75)), 2] <- 0
y <- log(abs(x %*% beta[,1]) + 1) + sqrt(abs(x %*% beta[,2]))
y <- y + rnorm(nsim, sd = 0.1)
res_ridge <- ridgeSIR(x, y, H = 10, d = 2, mu2 = 10^8)
res_fused <- SISIR(res_ridge, rep(1, ncol(x)), ncores = 2)
res_fused

```

SISIRres

*Print SISIRres object***Description**

Print a summary of the result of [SISIRres](#) (SISIRres object)

**Usage**

```
## S3 method for class 'SISIRres'
summary(object, ...)
```

```
## S3 method for class 'SISIRres'
print(x, ...)
```

**Arguments**

object	a SISIRres object
...	not used
x	a SISIRres object

**Author(s)**

Victor Picheny, <victor.picheny@inrae.fr>  
 Remi Servien, <remi.servien@inrae.fr>  
 Nathalie Vialaneix, <nathalie.vialaneix@inrae.fr>

**See Also**

[SISIR](#)

---

sparseRes	<i>Print sparseRes object</i>
-----------	-------------------------------

---

**Description**

Print a summary of the result of [sparseSIR](#) ( sparseRes object)

**Usage**

```
## S3 method for class 'sparseRes'  
summary(object, ...)
```

```
## S3 method for class 'sparseRes'  
print(x, ...)
```

**Arguments**

object	a sparseRes object
...	not used
x	a sparseRes object

**Author(s)**

Victor Picheny, <victor.picheny@inrae.fr>  
Remi Servien, <remi.servien@inrae.fr>  
Nathalie Vialaneix, <nathalie.vialaneix@inra.fr>

**See Also**

[sparseSIR](#)

---

sparseSIR	<i>sparse SIR</i>
-----------	-------------------

---

**Description**

sparseSIR performs the second step of the method (shrinkage of ridge SIR results)

**Usage**

```
sparseSIR(  
  object,  
  inter_len,  
  adaptive = FALSE,  
  sel_prop = 0.05,  
  parallel = FALSE,  
  ncores = NULL  
)
```

**Arguments**

<code>object</code>	an object of class <code>ridgeRes</code> as obtained from the function <code>ridgeSIR</code>
<code>inter_len</code>	(numeric) vector with interval lengths
<code>adaptive</code>	should the function returns the list of strong zeros and non strong zeros (logical). Default to <code>FALSE</code>
<code>sel_prop</code>	used only when <code>adaptive = TRUE</code> . Fraction of the coefficients that will be considered as strong zeros and strong non zeros. Default to 0.05
<code>parallel</code>	whether the computation should be performed in parallel or not. Logical. Default is <code>FALSE</code>
<code>ncores</code>	number of cores to use if <code>parallel = TRUE</code> . If left to <code>NULL</code> , all available cores minus one are used

**Value**

S3 object of class `sparseRes`: a list consisting of

- `sEDR` the estimated EDR space (a  $p \times d$  matrix)
- `alpha` the estimated shrinkage coefficients (a vector having a length similar to `inter_len`)
- `quality` a vector with various qualities for the model (see Details)
- `adapt_res` if `adaptive = TRUE`, a list of two vectors:
  - `nonzeros` indexes of variables that are strong non zeros
  - `zeros` indexes of variables that are strong zeros
- `parameters` a list of hyper-parameters for the method:
  - `inter_len` lengths of intervals
  - `sel_prop` if `adaptive = TRUE`, fraction of the coefficients which are considered as strong zeros or strong non zeros
- `rSIR` same as the input object
- `fit` a list for LASSO fit with:
  - `glmnet` result of the `glmnet` function
  - `lambda` value of the best Lasso parameter by CV
  - `x` exploratory variable values as passed to fit the model

@details Different quality criteria used to select the best models among a list of models with different interval definitions. Quality criteria are: log-likelihood (`loglik`), cross-validation error as provided by the function `glmnet`, two versions of the AIC (AIC and AIC2) and of the BIC (BIC and BIC2) in which the number of parameters is either the number of non null intervals or the number of non null parameters with respect to the original variables.

**Author(s)**

Victor Picheny, <victor.picheny@inrae.fr>  
 Remi Servien, <remi.servien@inrae.fr>  
 Nathalie Vialaneix, <nathalie.vialaneix@inrae.fr>

## References

Picheny, V., Servien, R., and Villa-Vialaneix, N. (2019) Interpretable sparse SIR for digitized functional data. *Statistics and Computing*, **29**(2), 255–267.

## See Also

[ridgeSIR](#), [project.sparseRes](#), [SISIR](#)

## Examples

```
set.seed(1140)
tsteps <- seq(0, 1, length = 200)
nsim <- 100
simulate_bm <- function() return(c(0, cumsum(rnorm(length(tsteps)-1, sd=1))))
x <- t(replicate(nsim, simulate_bm()))
beta <- cbind(sin(tsteps*3*pi/2), sin(tsteps*5*pi/2))
beta[((tsteps < 0.2) | (tsteps > 0.5)), 1] <- 0
beta[((tsteps < 0.6) | (tsteps > 0.75)), 2] <- 0
y <- log(abs(x %>% beta[,1]) + 1) + sqrt(abs(x %>% beta[,2]))
y <- y + rnorm(nsim, sd = 0.1)
res_ridge <- ridgeSIR(x, y, H = 10, d = 2, mu2 = 10^8)
res_sparse <- sparseSIR(res_ridge, rep(10, 20))
```

---

truffles

*Dataset "Truffles"*

---

## Description

Yearly truffles production and corresponding monthly rainfall information of the Perigord black truffle in the Vaucluse (France) between 1924 and 1949.

## Format

3 datasets are provided:

- `rainfall`: a data frame with 15 columns (months from January Year  $n$  to March Year  $n+1$ ) and 25 rows (production years from 1924/1925 to 1948/1949). Data correspond to cumulated rainfall in mm;
- `truffles`: a vector with 25 values corresponding to the total production (in kg) of truffles in the truffle patch of *T. melanosporum* de Pernes-Les-Fontaines (Vaucluse, France);
- `beta`: 0/1 vector with 15 values indicated the months during which the rainfall has the most important influence on the truffle production, as provided by experts.

## Details

This dataset has been made available by courtesy of the authors of the publication [Baragatti *et al.*, 2019]. Meteorological data have been provided by Meteo France <https://meteofrance.com> (Orange meteorological station) and truffle production data are courtesy of the truffle patch.

## References

Baragatti M., Grollemund P.M., Montpied P., Dupouey J.L., Gravier J., Murat C., Le Tacon F. (2019) Influence of annual climatic variations, climate changes, and sociological factors on the production of the Perigord black truffle (*Tuber melanosporum Vittad.*) from 1903-1904 to 1988-1989 in the Vaucluse (France), *Mycorrhiza*, **29**(2), 113-125.

## Examples

```
data(truffles)
summary(truffles)
plot(1:15, rainfall[1, ], type = "l", xlab = "month", ylab = "rainfall (mm)")
```

---

tune.ridgeSIR	<i>Cross-Validation for ridge SIR</i>
---------------	---------------------------------------

---

## Description

tune.ridgeSIR performs a Cross Validation for ridge SIR estimation

## Usage

```
tune.ridgeSIR(
  x,
  y,
  listH,
  list_mu2,
  list_d,
  n folds = 10,
  parallel = TRUE,
  ncores = NULL
)
```

## Arguments

x	explanatory variables (numeric matrix or data frame)
y	target variable (numeric vector)
listH	list of the number of slices to be tested (numeric vector)
list_mu2	list of ridge regularization parameters to be tested (numeric vector)
list_d	list of the dimensions to be tested (numeric vector)
n folds	number of folds for the cross validation. Default is 10
parallel	whether the computation should be performed in parallel or not. Logical. Default is FALSE
ncores	number of cores to use if parallel = TRUE. If left to NULL, all available cores minus one are used

**Value**

a data frame with tested parameters and corresponding CV error and estimation of R(d)

**Author(s)**

Victor Picheny, <victor.picheny@inrae.fr>  
Remi Servien, <remi.servien@inrae.fr>  
Nathalie Vialaneix, <nathalie.vialaneix@inrae.fr>

**References**

Picheny, V., Servien, R. and Villa-Vialaneix, N. (2016) Interpretable sparse SIR for digitized functional data. *Statistics and Computing*, **29**(2), 255–267.

**See Also**

[ridgeSIR](#)

**Examples**

```
set.seed(1115)
tsteps <- seq(0, 1, length = 200)
nsim <- 100
simulate_bm <- function() return(c(0, cumsum(rnorm(length(tsteps)-1, sd=1))))
x <- t(replicate(nsim, simulate_bm()))
beta <- cbind(sin(tsteps*3*pi/2), sin(tsteps*5*pi/2))
y <- log(abs(x %>% beta[,1])) + sqrt(abs(x %>% beta[,2]))
y <- y + rnorm(nsim, sd = 0.1)
list_mu2 <- 10^(0:10)
listH <- c(5, 10)
list_d <- 1:4
set.seed(1129)

res_tune <- tune.ridgeSIR(x, y, listH, list_mu2, list_d, nfolds = 10,
  parallel = TRUE, ncores = 2)
```



# Index

beta (truffles), [14](#)

extract\_at (SFCB-class), [7](#)

glmnet, [2](#), [10](#), [13](#)

plot.SFCB (SFCB-class), [7](#)

predict.glmnet, [2](#)

print.ridgeRes (ridgeRes), [3](#)

print.SFCB (SFCB-class), [7](#)

print.SISIRres (SISIRres), [11](#)

print.sparseRes (sparseRes), [12](#)

project, [2](#)

project.sparseRes, [14](#)

quality (SFCB-class), [7](#)

rainfall (truffles), [14](#)

ridgeRes, [3](#)

ridgeRes-class (ridgeRes), [3](#)

ridgeSIR, [3](#), [4](#), [4](#), [9](#), [10](#), [13](#), [14](#), [16](#)

sfc, [5](#), [7](#), [9](#)

SFCB-class, [7](#)

SISIR, [5](#), [9](#), [11](#), [14](#)

SISIRres, [11](#), [11](#)

sparseRes, [12](#)

sparseRes-class (sparseRes), [12](#)

sparseSIR, [2](#), [3](#), [5](#), [10](#), [12](#), [12](#)

summary.ridgeRes (ridgeRes), [3](#)

summary.SFCB (SFCB-class), [7](#)

summary.SISIRres (SISIRres), [11](#)

summary.sparseRes (sparseRes), [12](#)

truffles, [14](#)

tune.ridgeSIR, [5](#), [15](#)