

Package ‘SISIR’

June 5, 2020

Type Package

Title Sparse Interval Sliced Inverse Regression

Version 0.1-1

Date 2020-06-05

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

Description An interval fusion procedure for functional data in the semiparametric framework of SIR, as described in <doi:10.1007/s11222-018-9806-6>. Standard ridge and sparse SIR are also included in the package.

Depends foreach, doParallel

Imports Matrix, expm, RSpectra, glmnet

License GPL (>= 2)

RoxygenNote 7.1.0

NeedsCompilation no

Author Victor Picheny [aut],
Remi Servien [aut],
Nathalie Vialaneix [aut, cre]

Repository CRAN

Date/Publication 2020-06-05 10:00:02 UTC

R topics documented:

project	2
ridgeRes	3
ridgeSIR	4
SISIR	5
SISIRres	7
sparseRes	8
sparseSIR	8
tune	10

Index	13
--------------	-----------

project

sparse SIR

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>
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

[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)
## Not run:
res_ridge <- ridgeSIR(x, y, H = 10, d = 2)
res_sparse <- sparseSIR(res_ridge, rep(1, ncol(x)))
proj_data <- project(res_sparse)

## End(Not run)

```

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

object	a ridgeRes object
...	not used
x	a ridgeRes 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

<code>x</code>	explanatory variables (numeric matrix or data frame)
<code>y</code>	target variable (numeric vector)
<code>H</code>	number of slices (integer)
<code>d</code>	number of dimensions to be kept
<code>mu2</code>	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. (2016) 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)
## Not run: print(res_ridge)
```

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 ridgeSIR
<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>
 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](#), [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)
## Not run: res_fused <- SISIR(res_ridge, rep(1, ncol(x)))

```

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

object	an object of class <code>ridgeRes</code> as obtained from the function <code>ridgeSIR</code>
inter_len	(numeric) vector with interval lengths
adaptive	should the function returns the list of strong zeros and non strong zeros (logical). Default to <code>FALSE</code>
sel_prop	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
parallel	whether the computation should be performed in parallel or not. Logical. Default is <code>FALSE</code>
ncores	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. (2016) 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))
```

tune

Cross-Validation for ridge SIR

Description

`tune.ridgeSIR` performs a Cross Validation for ridge SIR estimation

Usage

```
tune(x, y, listH, list_mu2, list_d, nfolds, parallel, ncores)
```

```
## S3 method for class 'ridgeSIR'
tune(
  x,
  y,
  listH,
  list_mu2,
  list_d,
  nfolds = 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)
nfolds	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)
## Not run:
res_tune <- tune.ridgeSIR(x, y, listH, list_mu2, list_d,
  nfolds = 10, parallel = TRUE)
```

End(Not run)

Index

`glmnet`, [2](#), [6](#), [9](#)

`predict.glmnet`, [2](#)

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

`print.SISIRres` (`SISIRres`), [7](#)

`print.sparseRes` (`sparseRes`), [8](#)

`project`, [2](#)

`project.sparseRes`, [10](#)

`ridgeRes`, [3](#)

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

`ridgeSIR`, [3](#), [4](#), [5](#), [6](#), [9–11](#)

`SISIR`, [5](#), [5](#), [7](#), [10](#)

`SISIRres`, [7](#), [7](#)

`sparseRes`, [8](#)

`sparseRes-class` (`sparseRes`), [8](#)

`sparseSIR`, [2](#), [5](#), [6](#), [8](#), [8](#)

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

`summary.SISIRres` (`SISIRres`), [7](#)

`summary.sparseRes` (`sparseRes`), [8](#)

`tune`, [10](#)

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