

# Package ‘slasso’

October 15, 2021

**Type** Package

**Title** S-LASSO Estimator for the Function-on-Function Linear Regression

**Version** 1.0.0

**Description** Implements the smooth LASSO estimator for the function-on-function linear regression model described in Centofanti et al. (2020) <[arXiv:2007.00529](#)>.

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.1.2

**LinkingTo** Rcpp, RcppArmadillo

**Depends** inline

**Imports** Rcpp, RcppArmadillo, fda, fda.usc, matrixcalc, parallel, matrixStats, MASS, plot3D, methods, cxxfunplus

**URL** <https://github.com/unina-sfere/slasso>

**BugReports** <https://github.com/unina-sfere/slasso>

**SystemRequirements** GNU make

**Suggests** knitr, rmarkdown, testthat

**NeedsCompilation** yes

**Author** Fabio Centofanti [cre, aut],  
Antonio Lepore [aut],  
Simone Vantini [aut],  
Matteo Fontana [aut]

**Maintainer** Fabio Centofanti <[fabio.centofanti@unina.it](mailto:fabio.centofanti@unina.it)>

**Repository** CRAN

**Date/Publication** 2021-10-15 07:40:02 UTC

## R topics documented:

slasso-package	2
plot.slasso	3

simulate_data . . . . .	4
slasso.fr . . . . .	5
slasso.fr_cv . . . . .	7

<b>Index</b>	<b>9</b>
--------------	----------

---

slasso-package	<i>Smooth LASSO Estimator for the Function-on-Function Linear Regression Model</i>
----------------	--

---

## Description

Implements the Smooth LASSO Estimator for the Function-on-Function Linear Regression Model described in Centofanti et al. (2020) <arXiv:2007.00529>.

## Details

Package: slasso  
 Type: Package  
 Version: 1.0.0  
 Date: 2021-10-13  
 License: GPL (>= 3)

## Author(s)

Fabio Centofanti, Matteo Fontana, Antonio Lepore, Simone Vantini

## References

Centofanti, F., Fontana, M., Lepore, A., & Vantini, S. (2020). Smooth LASSO Estimator for the Function-on-Function Linear Regression Model. *arXiv preprint arXiv:2007.00529*.

## See Also

[slasso.fr](#), [slasso.fr\\_cv](#)

## Examples

```
library(slasso)
data<-simulate_data("Scenario II",n_obs=150)
X_fd=data$X_fd
Y_fd=data$Y_fd
domain=c(0,1)
n_basis_s<-30
```

```

n_basis_t<-30
breaks_s<-seq(0,1,length.out = (n_basis_s-2))
breaks_t<-seq(0,1,length.out = (n_basis_t-2))
basis_s <- fda::create.bspline.basis(domain,breaks=breaks_s)
basis_t <- fda::create.bspline.basis(domain,breaks=breaks_t)

mod_slasso_cv<-slasso.fr_cv(Y_fd = Y_fd,X_fd=X_fd,basis_s=basis_s,basis_t=basis_t,
lambda_L_vec = 10^seq(0,1,by=1),lambda_s_vec = 10^-9,lambda_t_vec = 10^-7,
B0=NULL,max_iterations=10,K=2,invisible=1,ncores=1)
mod_slasso<-slasso.fr(Y_fd = Y_fd,X_fd=X_fd,basis_s=basis_s,basis_t=basis_t,
lambda_L = 10^0.7,lambda_s =10^-5,lambda_t = 10^-6,B0 =NULL,invisible=1,max_iterations=10)

plot(mod_slasso_cv)
plot(mod_slasso)

```

---

plot.slasso

*Plot the results of the S-LASSO method*


---

## Description

This function provides plots of the S-LASSO coefficient function estimate when applied to the output of `slasso.fr`, whereas provides the cross-validation plots when applied to the output of `slasso.fr_cv`. In the latter case the first plot displays the CV values as a function of `lambda_L`, `lambda_s` and `lambda_t`, and the second plot displays the CV values as a function of `lambda_L` with `lambda_s` and `lambda_t` fixed at their optimal values.

## Usage

```

## S3 method for class 'slasso_cv'
plot(x, ...)

## S3 method for class 'slasso'
plot(x, ...)

```

## Arguments

`x`                    The output of either `slasso.fr_cv` or `slasso.fr`.

`...`                 No additional parameters, called for side effects.

## Value

No return value, called for side effects.

**Examples**

```

library(slasso)
data<-simulate_data("Scenario II",n_obs=150)
X_fd=data$X_fd
Y_fd=data$Y_fd
domain=c(0,1)
n_basis_s<-30
n_basis_t<-30
breaks_s<-seq(0,1,length.out = (n_basis_s-2))
breaks_t<-seq(0,1,length.out = (n_basis_t-2))
basis_s <- fda::create.bspline.basis(domain,breaks=breaks_s)
basis_t <- fda::create.bspline.basis(domain,breaks=breaks_t)
mod_slasso<-slasso.fr(Y_fd = Y_fd,X_fd=X_fd,basis_s=basis_s,basis_t=basis_t,
lambda_L = -1.5,lambda_s =-8,lambda_t = -7,B0 =NULL,invisible=1,max_iterations=10)
plot(mod_slasso)

```

---

simulate_data	<i>Simulate data through the function-on-function linear regression model</i>
---------------	---

---

**Description**

Generate synthetic data as in the simulation study of Centofanti et al. (2020).

**Usage**

```
simulate_data(scenario, n_obs = 3000, type_x = "Bspline")
```

**Arguments**

scenario	A character strings indicating the scenario considered. It could be "Scenario I", "Scenario II", "Scenario III", and "Scenario IV".
n_obs	Number of observations.
type_x	Covariate generating mechanism, either Bspline or Brownian.

**Value**

A list containing the following arguments:

X: Covariate matrix, where the rows correspond to argument values and columns to replications.

Y: Response matrix, where the rows correspond to argument values and columns to replications.

X\_fd: Coavariate functions.

Y\_fd: Response functions.

cclus: True cluster membership vector.

## References

Centofanti, F., Fontana, M., Lepore, A., & Vantini, S. (2020). Smooth LASSO Estimator for the Function-on-Function Linear Regression Model. *arXiv preprint arXiv:2007.00529*.

## Examples

```
library(slasso)
data<-simulate_data("Scenario II",n_obs=150)
```

---

slasso.fr	<i>Smooth LASSO estimator for the function-on-function linear regression model</i>
-----------	--

---

## Description

The smooth LASSO (S-LASSO) method for the function-on-function linear regression model provides interpretable coefficient function estimates that are both locally sparse and smooth (Centofanti et al., 2020).

## Usage

```
slasso.fr(
  Y_fd,
  X_fd,
  basis_s,
  basis_t,
  lambda_L,
  lambda_s,
  lambda_t,
  B0 = NULL,
  ...
)
```

## Arguments

<code>Y_fd</code>	An object of class <code>fd</code> corresponding to the response functions.
<code>X_fd</code>	An object of class <code>fd</code> corresponding to the covariate functions.
<code>basis_s</code>	B-splines basis along the <code>s</code> -direction of class <code>basisfd</code> .
<code>basis_t</code>	B-splines basis along the <code>t</code> -direction of class <code>basisfd</code> .
<code>lambda_L</code>	Regularization parameter of the functional LASSO penalty.
<code>lambda_s</code>	Regularization parameter of the smoothness penalty along the <code>s</code> -direction.
<code>lambda_t</code>	Regularization parameter of the smoothness penalty along the <code>t</code> -direction.
<code>B0</code>	Initial estimator of the basis coefficients matrix of the coefficient function. Should have dimensions in accordance with the basis dimensions of <code>basis_s</code> and <code>basis_t</code> .
<code>...</code>	Other arguments to be passed to the Orthant-Wise Limited-memory Quasi-Newton optimization function. See the <code>lbfgs</code> help page of the package <code>lbfgs</code> .

**Value**

A list containing the following arguments:

- B: The basis coefficients matrix estimate of the coefficient function.
- Beta\_hat\_fd: The coefficient function estimate of class bifd.
- alpha: The intercept function estimate.
- lambda\_s\_L: Regularization parameter of the functional LASSO penalty.
- lambda\_s: Regularization parameter of the smoothness penalty along the s-direction.
- lambda\_t: Regularization parameter of the smoothness penalty along the t-direction.
- Y\_fd: The response functions.
- X\_fd: The covariate functions.
- per\_0: The fraction of domain where the coefficient function is zero.
- type: The output type.

**References**

Centofanti, F., Fontana, M., Lepore, A., & Vantini, S. (2020). Smooth LASSO Estimator for the Function-on-Function Linear Regression Model. *arXiv preprint arXiv:2007.00529*.

**See Also**

[slasso.fr\\_cv](#)

**Examples**

```
library(slasso)
data<-simulate_data("Scenario II",n_obs=150)
X_fd=data$X_fd
Y_fd=data$Y_fd
domain=c(0,1)
n_basis_s<-30
n_basis_t<-30
breaks_s<-seq(0,1,length.out = (n_basis_s-2))
breaks_t<-seq(0,1,length.out = (n_basis_t-2))
basis_s <- fda::create.bspline.basis(domain,breaks=breaks_s)
basis_t <- fda::create.bspline.basis(domain,breaks=breaks_t)
mod_slasso<-slasso.fr(Y_fd = Y_fd,X_fd=X_fd,basis_s=basis_s,basis_t=basis_t,
lambda_L = -1.5,lambda_s =-8,lambda_t = -7,B0 =NULL,invisible=1,max_iterations=10)
```

slasso.fr\_cv

*Cross-validation for the S-LASSO estimator***Description**

K-fold cross-validation procedure to choose the tuning parameters for the S-LASSO estimator (Centofanti et al., 2020).

**Usage**

```
slasso.fr_cv(
  Y_fd,
  X_fd,
  basis_s,
  basis_t,
  K = 10,
  kss_rule_par = 0.5,
  lambda_L_vec = NULL,
  lambda_s_vec = NULL,
  lambda_t_vec = NULL,
  B0 = NULL,
  ncores = 1,
  ...
)
```

**Arguments**

<code>Y_fd</code>	An object of class <code>fd</code> corresponding to the response functions.
<code>X_fd</code>	An object of class <code>fd</code> corresponding to the covariate functions.
<code>basis_s</code>	B-splines basis along the <code>s</code> -direction of class <code>basisfd</code> .
<code>basis_t</code>	B-splines basis along the <code>t</code> -direction of class <code>basisfd</code> .
<code>K</code>	Number of folds. Default is 10.
<code>kss_rule_par</code>	Parameter of the k-standard error rule. If <code>kss_rule_par=0</code> the tuning parameters that minimize the estimated prediction error are chosen. Default is 0.5.
<code>lambda_L_vec</code>	Vector of regularization parameters of the functional LASSO penalty.
<code>lambda_s_vec</code>	Vector of regularization parameters of the smoothness penalty along the <code>s</code> -direction.
<code>lambda_t_vec</code>	Vector of regularization parameters of the smoothness penalty along the <code>t</code> -direction.
<code>B0</code>	Initial estimator of the basis coefficients matrix of the coefficient function. Should have dimensions in accordance with the basis dimensions of <code>basis_s</code> and <code>basis_t</code> .
<code>ncores</code>	If <code>ncores&gt;1</code> , then parallel computing is used, with <code>ncores</code> cores. Default is 1.
<code>...</code>	Other arguments to be passed to the Orthant-Wise Limited-memory Quasi-Newton optimization function. See the <code>lbfgs</code> help page of the package <code>lbfgs</code> .

**Value**

A list containing the following arguments:

- `lambda_opt_vec`: Vector of optimal tuning parameters.
- `CV`: Estimated prediction errors.
- `CV_sd`: Standard errors of the estimated prediction errors.
- `per_0`: The fractions of domain where the coefficient function is zero for all the tuning parameters combinations.
- `comb_list`: The combinations of `lambda_L`, `lambda_s` and `lambda_t` explored.
- `Y_fd`: The response functions.
- `X_fd`: The covariate functions.

**References**

Centofanti, F., Fontana, M., Lepore, A., & Vantini, S. (2020). Smooth LASSO Estimator for the Function-on-Function Linear Regression Model. *arXiv preprint arXiv:2007.00529*.

**See Also**

[slasso.fr](#)

**Examples**

```
library(slasso)
data<-simulate_data("Scenario II",n_obs=150)
X_fd=data$X_fd
Y_fd=data$Y_fd
domain=c(0,1)
n_basis_s<-60
n_basis_t<-60
breaks_s<-seq(0,1,length.out = (n_basis_s-2))
breaks_t<-seq(0,1,length.out = (n_basis_t-2))
basis_s <- fda::create.bspline.basis(domain,breaks=breaks_s)
basis_t <- fda::create.bspline.basis(domain,breaks=breaks_t)
mod_slasso_cv<-slasso.fr_cv(Y_fd = Y_fd,X_fd=X_fd,basis_s=basis_s,basis_t=basis_t,
lambda_L_vec=seq(0,1,by=1),lambda_s_vec=c(-9),lambda_t_vec=-7,B0=NULL,
max_iterations=10,K=2,invisible=1,ncores=1)
```



# Index

`plot.slasso`, [3](#)  
`plot.slasso_cv` (`plot.slasso`), [3](#)  
  
`simulate_data`, [4](#)  
`slasso` (`slasso-package`), [2](#)  
`slasso-package`, [2](#)  
`slasso.fr`, [2](#), [5](#), [8](#)  
`slasso.fr_cv`, [2](#), [6](#), [7](#)