Package ‘PFLR’

August 22, 2024

Type Package

Title Estimating Penalized Functional Linear Regression

Version 1.1.0

Description Implementation of commonly used penalized functional linear regression models, including the Smooth and Locally Sparse (SLoS) method by Lin et al. (2016) <doi:10.1080/10618600.2016.1195273>, Nested Group bridge Regression (NGR) method by Guan et al. (2020) <doi:10.1080/10618600.2020.1713797>, Functional Linear Regression That's interpretable (FLIRTI) by James et al. (2009) <doi:10.1214/08-AOS641>, and the Penalized B-spline regression method.

License GPL-2

LazyData true

Encoding UTF-8

Imports fda, MASS, flare, psych, glmnet, stats, utils

RoxygenNote 7.3.1

NeedsCompilation no

Depends R (>= 3.5.0)

Author Tianyu Guan [aut],
Haolun Shi [aut, cre, cph],
Rob Cameron [aut],
Zhenhua Lin [aut]

Maintainer Haolun Shi <shl2003@connect.hku.hk>

Repository CRAN

Date/Publication 2024-08-22 06:30:09 UTC

Contents

`FLiRTI` *FLiRTI Regression Model*

Description

Calculates functional regression that's interpretable using the FLiRTI method.

Usage

```
FLiRTI(
  Y,
  X,
  d,
  cons,
  domain,
  extra = list(Mf = 6:30, lambda = seq(5e-04, 100, length.out = 50))
)
```

Arguments

`Y`	Vector of length n, centred response.
`X`	Matrix of n x p, covariate matrix, should be dense.
`d`	Integer, degree of the B-spline basis functions.
`cons`	Divide subinterval into how many small ones.
`domain`	The range over which the function X(t) is evaluated and the coefficient function β(t) is expanded by the B-spline basis functions.
`extra`	List containing parameters which have default values: • Mf: Mf+1 is the number of knots for the B-spline basis functions that expand β(t), default is 6:30. • lambda: Tuning parameter, default is seq(0.0005,100,length.out = 50).

Value

beta: Estimated β(t) at discrete points.

extra: List containing other values which may be of use:

- X: Matrix of n x p used for model.
- Y: Vector of length n used for model.
- domain: The range over which the function X(t) was evaluated and the coefficient function β(t) was expanded by the B-spline basis functions.
- delta: Estimated cutoff point.
- OptM: Optimal number of B-spline knots selected by BIC.
- Optlambda: Optimal shrinkage parameter selected by BIC.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 200
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)
lambda = seq(0.0005,0.01,length.out = 10)
Mf = 6:13
extra=list(Mf=Mf,lambda=lambda)

for(itersim in 1:nsim)
{
 dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=1)
 Y[,itersim]  = dat$Y
  X[,,itersim] = dat$X
}


fltyfit = FLiRTI(Y=Y[1:n,1],(X[1:n,,1]),d=3,cons=4,domain=domain,extra=extra)
```

`ngr` *Nested Group Bridge Regression Model*

Description

Calculates a functional regression model using a nested group bridge approach.

Usage

```
ngr(
  Y,
  X,
  M,
  d,
  domain,
  extra = list(alphaPS = 10^(-10:0), kappa = 10^(-(9:7)), tau = exp(seq(-50, -15, len =
    20)), gamma = 0.5, niter = 100)
)
```

Arguments

`Y`	Vector of length n.
`X`	Matrix of n x p, covariate matrix, should be dense.
`M`	Integer, t1,..., tM are M equally spaced knots.
`d`	Integer, the degree of B-Splines.
`domain`	The range over which the function X(t) is evaluated and the coefficient function β(t) is expanded by the B-spline basis functions.
`extra`	List containing other parameters which have defaults:

- alphaPs: Smoothing parameter for the Penalized B-splines method, default is 10^(-10:0).
- kappa: Tuning parameter for roughness penalty, default is 10^(-(9:7)).
- tau: Tuning parameter for the group bridge penalty, default is exp(seq(-50,-15,len = 20)).
- gamma: Real number, default is 0.5.
- niter: Integer, maximum number of iterations, default is 100.

Value

beta: Estimated β(t) at discrete points.

extra: List containing other values which may be of use:

- b: Estimated b-hat.
- delta: Estimated cutoff point.
- Ymean: Estimated y-hat.
- Xmean: Estimated x-hat.
- Optkappa: Optimal roughness penalty selected.
- Opttau: Optimal group bridge penalty selected.
- M: Integer representing the number of knots used in the model calculation.
- d: Integer, degree of B-Splines used.
- domain: The range over which the function X(t) was evaluated and the coefficient function β(t) was expanded by the B-spline basis functions.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 1
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)

M = 20
d = 3
norder   = d+1
nknots   = M+1
tobs = seq(domain[1],domain[2],length.out = p)
knots    = seq(domain[1],domain[2],length.out = nknots)
nbasis   = nknots + norder - 2
basis    = create.bspline.basis(knots,nbasis,norder)
basismat = eval.basis(tobs, basis)
h = (domain[2]-domain[1])/M
cef = c(1, rep(c(4,2), (M-2)/2), 4, 1)

V = eval.penalty(basis,int2Lfd(2))
alphaPS = 10^(-(10:3))
kappa   = 10^(-(8:7))
tau     = exp(seq(-35,-28,len=20))
gamma   = 0.5


for(itersim in 1:nsim)
{
dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=betaind)
Y[,itersim]  = dat$Y
X[,,itersim] = dat$X
}

ngrfit = ngr(Y=Y[1:n,1],X=(X[1:n,,1]),M,d,domain,extra= list(alphaPS=alphaPS, kappa=kappa, tau=tau))
```

---

ngr.data.generator.bsplines

Generating random curves from B-Splines

Description

Generating random curves from B-Splines n,nknots,norder,p,domain=c(0,1),snr,betaind

Usage

```
ngr.data.generator.bsplines(
  n,
  nknots,
  norder,
  p,
  domain = c(0, 1),
  snr,
  betaind
)
```

Arguments

`n`	Number of curves
`nknots`	Number of knots
`norder`	Degree
`p`	Number of time points
`domain`	Domain of time
`snr`	Signal to noise ratio
`betaind`	Numeric index for function

Value

X: The generated X matrix of curve sampled at each timepoint

Y: The generated dependent variable

`PenS` *Penalized B-splines Regression Model*

Description

Calculates a functional regression model using the penalized B-splines method.

Usage

```
PenS(Y, X, alpha, M, d, domain)
```

Arguments

`Y`	Vector of length n.
`X`	Matrix of n x p, covariate matrix, should be dense.
`alpha`	Vector.
`M`	Integer, t1,..., tM are M equally spaced knots.
`d`	Integer, the degree of B-Splines.
`domain`	The range over which the function X(t) is evaluated and the coefficient function β(t) is expanded by the B-spline basis functions.

Value

beta: Estimated β(t) at discrete points.

extra: List containing other values which may be of use:

- b: Estimated B-spline coefficients.
- Ymean: Mean of the Y values.
- Xmean: Mean of all X values.
- Optalpha: Optimal alpha value chosen.
- M: Integer representing the number of knots used in the model calculation.
- d: Integer, degree of B-Splines used.
- domain: The range over which the function X(t) was evaluated and the coefficient function β(t) was expanded by the B-spline basis functions.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 1
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)

M = 20
d = 3
alpha = 10^(-(10:3))


for(itersim in 1:nsim)
{
dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=betaind)
Y[,itersim]  = dat$Y
X[,,itersim] = dat$X
}

psfit = PenS(Y=Y[1:n,1],X=(X[1:n,,1]), alpha=alpha, M=M, d=d, domain=domain)
```

`plot.flirti` *Plot Method for flirti Objects*

Description

Plots coefficient function of objects of class "flirti".

Usage

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

Arguments

x	An object of class "flirti".
...	Other parameters to be passed through to plotting functions.

Value

A line graph of the beta values versus time.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 200
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)
lambda = seq(0.0005,0.01,length.out = 10)
Mf = 6:13
extra=list(Mf=Mf,lambda=lambda)

for(itersim in 1:nsim)
{
 dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=1)
 Y[,itersim]  = dat$Y
  X[,,itersim] = dat$X
}


fltyfit = FLiRTI(Y=Y[1:n,1],(X[1:n,,1]),d=3,cons=4,domain=domain,extra=extra)

plot(fltyfit)
```

---

plot.ngr *Plot Method for ngr Objects*

Description

Plots coefficient function for objects of the class "ngr".

Usage

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

Arguments

`x`	An object of class "ngr".
`...`	Other parameters to be passed through to plotting functions.

Value

A line graph of the beta values over time.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 1
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)

M = 20
d = 3
norder   = d+1
nknots   = M+1
tobs = seq(domain[1],domain[2],length.out = p)
knots    = seq(domain[1],domain[2],length.out = nknots)
nbasis   = nknots + norder - 2
basis    = create.bspline.basis(knots,nbasis,norder)
basismat = eval.basis(tobs, basis)
h = (domain[2]-domain[1])/M
cef = c(1, rep(c(4,2), (M-2)/2), 4, 1)

V = eval.penalty(basis,int2Lfd(2))
alphaPS = 10^(-(10:3))
kappa   = 10^(-(8:7))
tau     = exp(seq(-35,-28,len=20))
gamma   = 0.5


for(itersim in 1:nsim)
{
dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=betaind)
Y[,itersim]  = dat$Y
X[,,itersim] = dat$X
}
```

```
ngrfit = ngr(Y=Y[1:n,1],X=(X[1:n,,1]),M,d,domain,extra= list(alphaPS=alphaPS, kappa=kappa, tau=tau))

plot(ngrfit)
```

---

plot.ps *Plot Method for Penalized B-splines Objects*

Description

Plots coefficient function of objects of class "ps".

Usage

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

Arguments

x	An object of class "ps".
...	Other parameters to be passed through to plotting functions.

Value

A line graph of the beta values versus time.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 1
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)

M = 20
d = 3
alpha = 10^(-(10:3))


for(itersim in 1:nsim)
{
dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=betaind)
Y[,itersim]  = dat$Y
X[,,itersim] = dat$X
}
```

```
psfit = PenS(Y=Y[1:n,1],X=(X[1:n,,1]), alpha=alpha, M=M, d=d, domain=domain)

plot(psfit)
```

`plot.slos` *Plot Method for SLoS Objects*

Description

Plots coefficient function for objects of class "slos".

Usage

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

Arguments

`x`	An object of class "slos".
`...`	Other parameters to be passed through to plotting functions.

Value

A line graph of the beta values versus time.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 1
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)

M = 20
d = 3
norder   = d+1
nknots   = M+1
knots    = seq(domain[1],domain[2],length.out = nknots)
nbasis   = nknots + norder - 2
basis    = create.bspline.basis(knots,nbasis,norder)
V = eval.penalty(basis,int2Lfd(2))

extra=list(lambda=exp(seq(-18,-12, length.out = 10)),gamma=10^(-8:-6))
```

```
for(itersim in 1:nsim)
{
 dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=betaind)
  Y[,itersim]  = dat$Y
  X[,,itersim] = dat$X
}

slosfit = SLoS(Y=Y[1:n,1],(X[1:n,,1]),M=M,d=d,domain=domain,extra=extra)
plot(slosfit)
```

---

predict.flirti *Predict Method for flirti Objects*

Description

Predicted values based on objects of the class "flirti".

Usage

```
## S3 method for class 'flirti'
predict(object, Xnew, ...)
```

Arguments

object	An object of class "flirti".
Xnew	New covariate matrix for prediction, should be dense, centred.
...	Not applicable

Value

Predicted values.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 200
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)
lambda = seq(0.0005,0.01,length.out = 10)
Mf = 6:13
extra=list(Mf=Mf,lambda=lambda)
```

```
for(itersim in 1:nsim)
{
  dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=1)
 Y[,itersim]  = dat$Y
  X[,,itersim] = dat$X
}


fltyfit = FLiRTI(Y=Y[1:n,1],(X[1:n,,1]),d=3,cons=4,domain=domain,extra=extra)

predict(fltyfit,(X[1:n,,1]))
```

`predict.ngr` *Predict Method for ngr Objects*

Description

Predicted values based on "ngr" class objects.

Usage

```
## S3 method for class 'ngr'
predict(object, Xnew, ...)
```

Arguments

`object`	An object of class "ngr".
`Xnew`	New covariate matrix for prediction, should be dense, centred.
`...`	Not applicable

Value

Estimated Y hat value.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 1
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)
```

```
M = 20
d = 3
norder   = d+1
nknots   = M+1
tobs = seq(domain[1],domain[2],length.out = p)
knots    = seq(domain[1],domain[2],length.out = nknots)
nbasis   = nknots + norder - 2
basis    = create.bspline.basis(knots,nbasis,norder)
basismat = eval.basis(tobs, basis)
h = (domain[2]-domain[1])/M
cef = c(1, rep(c(4,2), (M-2)/2), 4, 1)

V = eval.penalty(basis,int2Lfd(2))
alphaPS = 10^(-(10:3))
kappa   = 10^(-(8:7))
tau     = exp(seq(-35,-28,len=20))
gamma   = 0.5


for(itersim in 1:nsim)
{
dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=betaind)
Y[,itersim]  = dat$Y
X[,,itersim] = dat$X
}

ngrfit = ngr(Y=Y[1:n,1],X=(X[1:n,,1]),M,d,domain,extra= list(alphaPS=alphaPS, kappa=kappa, tau=tau))
predict(ngrfit,X[1:n,,1])
```

---

predict.ps — *Predict Method for Penalized B-splines objects*

Description

Predicted values based on objects of class "ps".

Usage

```
## S3 method for class 'ps'
predict(object, Xnew, ...)
```

Arguments

object	An object of class "ps".
Xnew	New covariate matrix for prediction, should be dense, centred.
...	Not applicable

Value

Predicted values.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 1
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)

M = 20
d = 3
alpha = 10^(-(10:3))


for(itersim in 1:nsim)
{
dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=betaind)
Y[,itersim]  = dat$Y
X[,,itersim] = dat$X
}

psfit = PenS(Y=Y[1:n,1],X=(X[1:n,,1]), alpha=alpha, M=M, d=d, domain=domain)

predict(psfit,X[1:n,,1])
```

`predict.slos` *Predict Method for SLoS objects*

Description

Predicted values based on objects of class "slos".

Usage

```
## S3 method for class 'slos'
predict(object, Xnew, ...)
```

Arguments

`object`	An object of class "slos".
`Xnew`	New covariate matrix for prediction, should be dense, centred.
`...`	Not applicable

Value

Predicted values.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 1
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)

M = 20
d = 3
norder   = d+1
nknots   = M+1
knots    = seq(domain[1],domain[2],length.out = nknots)
nbasis   = nknots + norder - 2
basis    = create.bspline.basis(knots,nbasis,norder)
V = eval.penalty(basis,int2Lfd(2))

extra=list(lambda=exp(seq(-18,-12, length.out = 10)),gamma=10^(-8:-6))

for(itersim in 1:nsim)
{
 dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=betaind)
  Y[,itersim]  = dat$Y
  X[,,itersim] = dat$X
}

slosfit = SLoS(Y=Y[1:n,1],(X[1:n,,1]),M=M,d=d,domain=domain,extra=extra)

predict(slosfit,(X[1:n,,1]))
```

SLoS *SLoS regression Model*

Description

Calculates functional regression using the Smooth and Locally Sparse (SLoS) method.

Usage

```
SLoS(
  Y,
```

```
  X,
  M,
  d,
  domain,
  extra = list(Maxiter = 100, lambda = exp(seq(-20, -12, length.out = 10)), gamma =
    10^(-9:0), absTol = 10^(-10), Cutoff = 10^(-6))
)
```

Arguments

`Y`	Vector, length n, centred response.
`X`	Matrix of n x p, covariate matrix, should be dense, centred.
`M`	Integer, t1,..., tM are M equally spaced knots.
`d`	Integer, the degree of B-Splines.
`domain`	The range over which the function X(t) is evaluated and the coefficient function β(t) is expanded by the B-spline basis functions.
`extra`	List of parameters which have default values: • Maxiter: Maximum number of iterations for convergence of beta, default is 100. • lambda: Positive number, tuning parameter for fSCAD penalty, default is exp(seq(-20,-12, length.out = 10)). • gamma: Positive number, tuning parameter for the roughness penalty, default is 10^(-9:0). • absTol: Number, if max(norm(bHat)) is smaller than absTol, we stop another iteration, default is 10^(-10). • Cutoff: Number, if bHat is smaller than Cutoff, set it to zero to avoid being numerically unstable, default is 10^(-6).

Value

beta: Estimated β(t) at discrete points.

extra: List containing other values which may be of use:

- X: Matrix of n x p used for model.
- Y: Vector of length n used for model.
- M: Integer representing the number of knots used in the model calculation.
- d: Integer, degree of B-Splines used.
- domain: The range over which the function X(t) was evaluated and the coefficient function β(t) was expanded by the B-spline basis functions.
- b: Estimated b values.
- delta: Estimated cutoff point.
- Optgamma: Optimal smoothing parameter selected by BIC.
- Optlambda: Optimal shrinkage parameter selected by BIC.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 1
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)

M = 20
d = 3
norder   = d+1
nknots   = M+1
knots    = seq(domain[1],domain[2],length.out = nknots)
nbasis   = nknots + norder - 2
basis    = fda::create.bspline.basis(knots,nbasis,norder)
V = eval.penalty(basis,int2Lfd(2))

extra=list(lambda=exp(seq(-18,-12, length.out = 10)),gamma=10^(-8:-6))

for(itersim in 1:nsim)
{
 dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=betaind)
  Y[,itersim]  = dat$Y
  X[,,itersim] = dat$X
}

slosfit = SLoS(Y=Y[1:n,1],(X[1:n,,1]),M=M,d=d,domain=domain,extra=extra)
```

`summary.flirti` *Summary Method for flirti Objects*

Description

Summarizes the values of an object of class "flirti".

Usage

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

Arguments

`object`	An object of class "flirti".
`...`	Not applicable

Value

Prints a 5 number summary of the beta values, delta, OptM, and Optlambda

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 200
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)
lambda = seq(0.0005,0.01,length.out = 10)
Mf = 6:13
extra=list(Mf=Mf,lambda=lambda)

for(itersim in 1:nsim)
{
 dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=1)
 Y[,itersim]  = dat$Y
  X[,,itersim] = dat$X
}


fltyfit = FLiRTI(Y=Y[1:n,1],(X[1:n,,1]),d=3,cons=4,domain=domain,extra=extra)

summary(fltyfit)
```

---

summary.ngr — *Summary Method for ngr Objects*

Description

Summarizes objects of class "ngr".

Usage

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

Arguments

`object`	An object of class "ngr".
`...`	Not applicable

Value

Prints the 5 number summaries of beta and b values. Prints delta, Optkappa, and Opttau values.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 1
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)

M = 20
d = 3
norder   = d+1
nknots   = M+1
tobs = seq(domain[1],domain[2],length.out = p)
knots    = seq(domain[1],domain[2],length.out = nknots)
nbasis   = nknots + norder - 2
basis    = create.bspline.basis(knots,nbasis,norder)
basismat = eval.basis(tobs, basis)
h = (domain[2]-domain[1])/M
cef = c(1, rep(c(4,2), (M-2)/2), 4, 1)

V = eval.penalty(basis,int2Lfd(2))
alphaPS = 10^(-(10:3))
kappa   = 10^(-(8:7))
tau     = exp(seq(-35,-28,len=20))
gamma   = 0.5


for(itersim in 1:nsim)
{
dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=1)
Y[,itersim]  = dat$Y
X[,,itersim] = dat$X
}

ngrfit = ngr(Y=Y[1:n,1],X=(X[1:n,,1]),M,d,domain,extra= list(alphaPS=alphaPS, kappa=kappa, tau=tau))

summary(ngrfit)
```

---

summary.ps *Summary Method for Penalized B-splines Objects*

Description

Summarizes the values of an object of class "ps".

Usage

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

Arguments

`object`	An object of class "ps".
`...`	Not applicable

Value

Prints a 5 number summary of the beta values and coefficient values, and the optimal alpha.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 1
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)

M = 20
d = 3
alpha = 10^(-(10:3))


for(itersim in 1:nsim)
{
dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=betaind)
Y[,itersim]  = dat$Y
X[,,itersim] = dat$X
}

psfit = PenS(Y=Y[1:n,1],X=(X[1:n,,1]), alpha=alpha, M=M, d=d, domain=domain)

summary(psfit)
```

summary.slos *Summary Method for SLoS Objects*

Description

Summarizes values of an object of class "slos".

Usage

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

Arguments

object	An object of class "slos".
...	Not applicable

Value

Prints five number summary of beta values, delta, Optgamma, and Optlambda.

Examples

```
library(fda)
betaind = 1
snr  = 2
nsim = 1
n    = 50
p    = 21
Y = array(NA,c(n,nsim))
X = array(NA,c(n,p,nsim))
domain = c(0,1)

M = 20
d = 3
norder   = d+1
nknots   = M+1
knots    = seq(domain[1],domain[2],length.out = nknots)
nbasis   = nknots + norder - 2
basis    = create.bspline.basis(knots,nbasis,norder)
V = eval.penalty(basis,int2Lfd(2))

extra=list(lambda=exp(seq(-18,-12, length.out = 10)),gamma=10^(-8:-6))

for(itersim in 1:nsim)
{
 dat = ngr.data.generator.bsplines(n=n,nknots=64,norder=4,p=p,domain=domain,snr=snr,betaind=betaind)
  Y[,itersim]  = dat$Y
  X[,,itersim] = dat$X
```

```
}

slosfit = SLoS(Y=Y[1:n,1],(X[1:n,,1]),M=M,d=d,domain=domain,extra=extra)

summary(slosfit)
```

---

truck *Truck emissions data*

Description

The particulate matter emissions data, taken from the Coordinating Research Councils E55/E59 research project (Clark et al. 2007). In the project, trucks were placed on the chassis dynamometer bed to mimic inertia and particulate matter was measured by an emission analyzer on standard test cycles. The engine acceleration of diesel trucks was also recorded.

Usage

```
truck
```

Format

A data frame with 108 rows and 91 columns:

Y Emmission

X1-X90 Acceleration at each second

Source

Clark, N., Gautam, M., Wayne, W., Lyons, D., Thompson, G., and Zielinska, B. (2007), "Heavy-Duty Vehicle Chassis Dynamometer Testing for Emissions Inventory, Air Quality Modeling, Source Apportionment and Air Toxics Emissions Inventory: E55/59 All Phases," Coordinating Research Council, Alpharetta

Index