Package 'face'

January 23, 2019

Title Fast Covariance Estimation for Sparse Functional Data

Version 0.1-5

Author Luo Xiao [aut,cre], Cai Li [aut], William Checkley [aut], Ciprian Crainiceanu [aut]

Maintainer Luo Xiao <lxiao5@ncsu.edu>

Description We implement the Fast Covariance Estimation for Sparse Functional Data paper published in Statistics and Computing <doi: 10.1007/s11222-017-9744-8>.

Depends R (>= 3.2.0)

Imports stats, splines, Matrix, matrixcalc, mgcv

License GPL-3

LazyLoad yes

LazyData true

Repository CRAN

NeedsCompilation no

Date/Publication 2019-01-23 06:50:07 UTC

R topics documented:

face-package *face*

Description

Fast Covariance Estimation for Sparse Functional Data

Details

Package:	face
Type:	Package
Version:	0.1-5
Date:	2019-01-19
License:	GPL-3

Author(s)

Luo Xiao, Cai Li, William Checkley and Ciprian Crainiceanu

Maintainer: Luo Xiao <lxiao5@ncsu.edu>

References

Luo Xiao, Cai Li, William Checkley and Ciprian Crainiceanu, Fast covariance estimation for sparse functional data, Stat. Comput., doi: 10.1007/s1122201797448.

cor.face *Extraction of correlation and mean from a* face.sparse *object*

Description

Extraction of correlation and mean from a face.sparse object

Usage

```
cor.face(object,argvals.new,option="raw")
```

Arguments

object	A face.sparse object.
argvals.new	Where to evalulate correlation and mean.
option	Defaults to "raw"; if "smooth", then extract correlation from smoothed covariance function.

Value

argvals.new	Where to evaluate correlation and mean.
option	Defaults to "raw"; if "smooth", then extract correlation from smoothed covariance function.
Cor	estimated correlation matrix at argvals.new
mu	estimated group/population mean at argvals.new

Author(s)

Luo Xiao <lxiao5@ncsu.edu>

References

Luo Xiao, Cai Li, William Checkley and Ciprian Crainiceanu, Fast covariance estimation for sparse functional data, Stat. Comput., doi: 10.1007/s1122201797448.

Examples

```
# See the examples for "face.sparse".
```

---

face-internal *Internal functions for face package*

Description

Internal function.

---

face.sparse *Fast covariance estimation for sparse functional data*

Description

The function is to estimate the mean and covariance function from a cluster of functions/longitudinal observations.

Usage

```
face.sparse(data, newdata = NULL,
             center = TRUE, argvals.new = NULL,
             knots = 7,
             p = 3, m = 2, lambda = NULL, lambda_mean = NULL,
             search.length = 14,
             lower = -3, upper = 10, lower2 = -3, upper2 = 5,
             calculate.scores = FALSE,pve=0.99,two_step=FALSE)
```

Arguments

data	a data frame with three arguments: (1) argvals: observation times; (2) subj: subject indices; (3) y: values of observations. Missing values not allowed.
newdata	of the same strucutre as data; defaults to NULL, then no prediction.
center	logical. If TRUE, then Pspline smoothing of the population mean will be conducted and subtracted from the data before covariance smoothing; if FALSE, then the population mean will be just 0s.
argvals.new	a vector of observation time points to evaluate mean function, covariance function, error variance and etc. If NULL, then 100 equidistant points in the range of data$argvals.
knots	the number of knots for B-spline basis functions to be used; defaults to 7. The resulting number of basis functions is the number of interior knots plus the degree of B-splines.
p	the degrees of B-splines; defaults to 3.
m	the order of differencing penalty; defaults to 2.
lambda	the value of the smoothing parameter for covariance smoothing; defaults to NULL.
lambda_mean	the value of the smoothing parameter for mean smoothing; defaults to NULL.
search.length	the number of equidistant (log scale) smoothing parameters to search; defaults to 14.
lower, upper	bounds for log smoothing parameter for first step of estimation; defaults are -3 and 10, respectively.
lower2, upper2	bounds for log smoothing parameter for second step of estimation; defaults are lower and 5, respectively.
calculate.scores	if TRUE, scores will be calculated.
pve	Defaults 0.99. To select the number of eigenvalues by percentage of variance.
two_step	if TRUE, a two-step estimation procedure will be applied.

Details

This is a generalized version of bivariate P-splines (Eilers and Marx, 2003) for covariance smoothing of sparse functional or longitudinal data. It uses tensor product B-spline basis functions and employes a differencing penalty on the assosciated parameter matrix. The only smoothing parameter in the method is selected by leave-one-subject-out cross validation and is implemented with a fast algorithm.

There are two steps for estimation. During the first step, the objective function to minimize is the penalized least squares on empirical estimates of covariance function. During the second step, the covariance between the empirical estimates (depending on the estimates of covariance function) are accounted and thus a generalized penalized least squares are minimized.

If center is TRUE, then a population mean will be calculated and is smoothed by univariate P-spline smoothing:pspline (Eilers and Marx, 1996). This univariate smoothing uses leave-one-subject-out cross validation to select the smoothing parameter.

The knots are "equally-spaced", the differencing penalty in Eilers and Marx (2003) is used.

If the functional data are observed at the same grid for each function/curve and can be organized into a data matrix, then `fpca.face` in the package `refund` should instead be used. `fpca.face` allows a small percentage (less than 30 percent) of missing data in the data matrix.

Value

`newdata` Input

`y.pred,mu.pred,Chat.diag.pred, var.error.pred`
Predicted/estimated objects at `newdata$argvals`

`Theta` Estimated parameter matrix

`argvals.new` Vector of time points to evaluate population parameters

`mu.new, Chat.new,Cor.new,Cor.raw.new,Chat.raw.diag.new, var.error.new`
Estimated objects at `argvals.new`

`eigenfunctions, eigenvalues`
Estimated eigenfunctions (scaled eigenvector) and eigenvalues at `argvals.new`

`mu.hat,var.error.hat`
Estimated objects at `data$argvals`

`calculate.scores,scores`
if `calculate.scores` is TRUE (default to FALSE) and `newdata` is not NULL, then predicted scores will be calculated.

`...` ...

Author(s)

Luo Xiao <lxiao5@ncsu.edu> and Cai Li <cli9@ncsu.edu>

References

Luo Xiao, Cai Li, William Checkley and Ciprian Crainiceanu, Fast covariance estimation for sparse functional data, Stat. Comput., doi: 10.1007/s1122201797448.

Paul Eilers and Brian Marx, Multivariate calibration with temperature interaction using two-dimensional penalized signal regression, Chemometrics and Intelligent Laboratory Systems 66 (2003), 159-174.

Paul Eilers and Brian Marx, Flexible smoothing with B-splines and penalties, Statist. Sci., 11, 89-121, 1996.

Simon N. Wood, P-splines with derivative based penalties and tensor product smoothing of unevenly distributed data, Stat. Comput., doi: 10.1007/s112220169666x.

See Also

`fpca.face` and `fpca.sc` in `refund`

Examples

```
## Not run:
##########################
#### CD4 data example
##########################

require(refund)
data(cd4)
n <- nrow(cd4)
T <- ncol(cd4)

id <- rep(1:n,each=T)
t <- rep(-18:42,times=n)
y <- as.vector(t(cd4))
sel <- which(is.na(y))


## organize data and apply FACEs
data <- data.frame(y=log(y[-sel]),
argvals = t[-sel],
subj = id[-sel])
data <- data[data$y>4.5,]
fit_face <- face.sparse(data,argvals.new=(-20:40))

data.h <- data
tnew <- fit_face$argvals.new

## scatter plots
Xlab <- "Months since seroconversion"
Ylab <- "log (CD4 count)"
par(mfrow=c(1,1),mar = c(4.5,4.5,3,2))
id <- data.h$subj
uid <- unique(id)
plot(data.h$argvals,data.h$y,
type = "n", ylim = c(4.5,8),
xlab = Xlab, ylab = Ylab,
cex.lab = 1.25,cex.axis=1.25,cex.main = 1.25)

for(i in 1:10){
seq <- which(id==uid[i])
lines(data.h$argvals[seq],data.h$y[seq],lty=1,col="gray",lwd=1,type="l")
#points(data.h$argvals[seq],data.h$y[seq],col=1,lty=1,pch=1)
}

Sample <- seq(10,50,by=10)
for(i in Sample){
seq <- which(id==uid[i])
lines(data.h$argvals[seq],data.h$y[seq],lty=1,col="black",lwd=1,type="l")
}
lines(tnew,fit_face$mu.new,lwd=2,lty=2,col="red")
```

```
## plots of variance/correlation functions

Cov <- fit_face$Chat.new
Cov_diag <- diag(Cov)
Cor <- fit_face$Cor.new

par(mfrow=c(1,2),mar=c(4.5,4.1,3,4.5))


plot(tnew,Cov_diag,type="l",
xlab = Xlab, ylab="",main= "CD4: variance function",
#ylim = c(0.8,1.5),
cex.axis=1.25,cex.lab=1.25,cex.main=1.25,lwd=2)

require(fields)
image.plot(tnew,tnew,Cor,
xlab=Xlab, ylab = Xlab,
main = "CD4: correlation function",
cex.axis=1.25,cex.lab=1.25,cex.main=1.25,
axis.args = list(at = c(0,0.2,0.4,0.6,0.8,1.0)),
legend.shrink=0.75,legend.line=-1.5)


## prediction of several subjects

par(mfrow=c(2,2),mar=c(4.5,4.5,3,2))
Sample <- c(30,40,50,60)
for(i in 1:4){
sel <- which(id==uid[Sample[i]])
dati <- data.h[sel,]

seq <- -20:40
k <- length(seq)
dati_pred <- data.frame(y = rep(NA,nrow(dati) + k ),
argvals = c(rep(NA,nrow(dati)),seq),
subj=rep(dati$subj[1],nrow(dati) + k )
)

dati_pred[1:nrow(dati),] <- dati
yhat2 <- predict(fit_face,dati_pred)

data3 <- dati
Ylim <- range(c(data3$y,yhat2$y.pred))

plot(data3$argvals,data3$y,xlab=Xlab,ylab=Ylab, main = paste("Male ",i,sep=""),
ylim = c(4,8.5),
cex.lab=1.25,cex.axis = 1.25,cex.main = 1.25,pch=1,xlim=c(-20,40))

Ord <- nrow(dati) + 1:k
lines(dati_pred$argvals[Ord],yhat2$y.pred[Ord],col="red",lwd=2)
lines(dati_pred$argvals[Ord],
yhat2$y.pred[Ord] - 1.96*yhat2$se.pred[Ord], col="red",lwd=1,lty=2)
lines(dati_pred$argvals[Ord],
```

```
yhat2$y.pred[Ord] + 1.96*yhat2$se.pred[Ord], col="red",lwd=1,lty=2)

lines(tnew,fit_face$mu.new,lty=3,col="black",lwd=2)
legend("bottomleft",c("mean","prediction"),lty=c(3,1),col=1:2,lwd=2,bty="n")
}

## End(Not run)
```

`predict.face.sparse` *Subject-specific curve prediction from a face.sparse fit*

Description

Predict subject-specific curves based on a fit from "face.sparse".

Usage

```
## S3 method for class 'face.sparse'
predict(object, newdata,...)
```

Arguments

`object`	a fitted object from the R function "face.sparse".
`newdata`	a data frame with three arguments: (1) `argvals`: observation times; (2) `subj`: subject indices; (3) `y`: values of observations. NA values are allowed in "y" but not in the other two.
`...`	further arguments passed to or from other methods.

Details

This function makes prediction based on observed data for each subject. So for each subject, it requires at least one observed data. For the time points prediction is desired but no observation is available, just make the corresponding `data$y` as NA.

Value

`object`	A "face.sparse" fit
`newdata`	Input
`y.pred,mu.pred,Chat.pred, Chat.diag.pred, var.error.pred`	Predicted/estimated objects at the observation time points in `newdata`
`scores`	if `calculate.scores` in `object` is TRUE (typically FALSE), then predicted scores will be calculated.
`...`	...

Author(s)

Luo Xiao <lxiao5@ncsu.edu>

References

Luo Xiao, Cai Li, William Checkley and Ciprian Crainiceanu, Fast covariance estimation for sparse functional data, Stat. Comput., doi: 10.1007/s1122201797448.

Examples

```
#See the examples for "face.sparse".
```

`predict.pspline.face` *Mean prediction from a P-spline smoothing fit*

Description

Predict mean values based on a fit from "pspline".

Usage

```
## S3 method for class 'pspline.face'
predict(object, argvals.new,...)
```

Arguments

`object`	a fitted object from the R function "pspline".
`argvals.new`	a vector of new time points.
`...`	further arguments passed to or from other methods.

Value

Predicted means at `argvals.new`.

Author(s)

Luo Xiao <lxiao5@ncsu.edu>

References

Luo Xiao, Cai Li, William Checkley and Ciprian Crainiceanu, Fast covariance estimation for sparse functional data, Stat. Comput., doi: 10.1007/s1122201797448.

Examples

```
#See the examples for "pspline".
```

pspline *Univariate P-spline smoothing*

Description

Univariate P-spline smoothing with the smoothing parameter selected by leave-one-subject-out cross validation.

Usage

```
pspline(data, argvals.new = NULL, knots = 35,
            p = 3, m = 2, lambda = NULL,
            search.length = 100,
            lower = -20, upper = 20)
```

Arguments

data	a data frame with three arguments: (1) argvals: observation times; (2) subj: subject indices; (3) y: values of observations. Missing values not allowed.
argvals.new	a vector of observations times for prediction; if NULL, then the same as data$argvals.
knots	a vector of interior knots or the number of knots for B-spline basis functions to be used; defaults to 35.
p	the degrees of B-splines; defaults to 3.
m	the order of differencing penalty; defaults to 2.
lambda	the value of the smoothing parameter; defaults to NULL.
search.length	the number of equidistant (log scale) smoothing parameters to search; defaults to 100.
lower, upper	bounds for log smoothing parameter; defaults are -20 and 20.

Details

The function is an implementation of the P-spline smoothing in Eilers and Marx (1996). P-splines uses B-splines as basis functions and employs a differencing penalty on the coefficients. Leave-one-subject-out cross validation is used for selecting the smoothing parameter and a fast algorithm is implemented.

Value

fitted.values	Fitted mean values
B	B-spline design matrix
theta	Estimated coefficients
s	Eigenvalues
knots	Knots
p	The degrees of B-splines

m	The order of differencing penalty
lambda	The value of the smoothing parameter
argvals.new	A vector of observations times
mu.new	Fitted mean values at argvals.new

Author(s)

Luo Xiao <lxiao5@ncsu.edu>

References

Paul Eilers and Brian Marx, Flexible smoothing with B-splines and penalties, Statist. Sci., 11, 89-121, 1996.

Luo Xiao, Cai Li, William Checkley and Ciprian Crainiceanu, Fast covariance estimation for sparse functional data, Stat. Comput., doi: 10.1007/s1122201797448.

See Also

gam in mgcv.

Examples

```
## Not run:
## cd4 data
require(refund)
data(cd4)
n <- nrow(cd4)
T <- ncol(cd4)

id <- rep(1:n,each=T)
t <- rep(-18:42,times=n)
y <- as.vector(t(cd4))
sel <- which(is.na(y))

## organize data
data <- data.frame(y=log(y[-sel]),
argvals = t[-sel],
subj = id[-sel])
data <- data[data$y>4.5,]

## smooth
fit <- pspline(data)

## plot
plot(data$argvals,fit$mu.new,type="p")

## prediction
pred <- predict(fit,quantile(data$argvals,c(0.2,0.6)))
pred

## End(Not run)
```

`select.knots` *Knots selection for P-spline smoothing*

Description

Construct knots from either quantiles of observed time points or equally-spaced time points.

Usage

```
select.knots(t,knots=10,p=3,option="equally-spaced")
```

Arguments

`t`	Observed time points.
`knots`	Number of interior knots.
`p`	Degrees of B-splines to be used.
`option`	Default "equally-spaced": equally-spaced time points in the range of `t`; if "quantile", then quantiles of `t`.

Details

The number of knots in the output will be `knot` plus 2 times `p`; and the B-spline basis matrix constructed from this vector of knots with degrees `p` will be `knots` plus `p`.

Value

A vector of knots

Author(s)

Luo Xiao <lxiao5@ncsu.edu>

Examples

```
t <- rnorm(100)
knots <- select.knots(t)
```

Index