Package 'fpca'

July 2, 2014

Version 0.2-1

Date 2011-02-15

Title Restricted MLE for Functional Principal Components Analysis

Author Jie Peng <jie@wald.ucdavis.edu> , Debashis Paul <debashis@wald.ucdavis.edu>

Maintainer Jie Peng <jie@wald.ucdavis.edu>

Depends sm,splines

Description A geometric approach to MLE for functional principal components

License GPL (>= 2)

URL anson.ucdavis.edu/~jie/software.html

Repository CRAN

Date/Publication 2011-02-24 09:00:27

NeedsCompilation no

R topics documented:

easy — *An example with M=5 (basis functions) and r=3 (non-zero eigenvalues)*

Description

A simulated dataset as an example which corresponds to the "easy" case in the paper

Usage

```
data(easy)
```

Format

easy is a list with six components (in the given order):

data data matrix with three columns: column 1–ID, column 2–measurement, column 3–time.

eigenfunctions true eigenfunctions: generated from cubic Bsplines with M=5 equally spaced knots.

eigenvalues true eigenvalues: first–1, second–0.66, third–0.52, others–zero.

number_of_basis true number of basis functions: M=5.

dimension true dimension of the process: r=3.

error_sd true error standard deviation: 0.25.

Details

mean curve of the process is zero; principal component scores and errors are all i.i.d N(0,1); there are 200 subjects, and each has 2~10 measurements uniformly distributed on [0,1]; in total there are 1227 measurements

fpca — *The fpca package: summary information*

Description

The package implements the restricted maximum likelihood estimation through a Newton-Raphson procedure described in Peng and Paul (2009) to estimate functional principal components from (sparsely and irregularly observed) longitudinal data.

Details

This is version 0.2-1 updated in Feb, 2011. Two new functions, fpca.score, fpca.pred, are included. Missing values are not allowed. Subjects with only one measurement will be automatically excluded. The main function is 'fpca.mle'. Simulated data sets can be called by 'data(easy)' and 'data(prac)'. Type 'help(easy)' and 'help(prac)' to see details. Packages 'sm' and 'splines' are used by this package. The code for EM (as initial estimate) is provided by Professor G. James in USC (with slight modifications).

Author(s)

J. Peng, D. Paul

References

Peng, J. and Paul, D. (2009). A geometric approach to maximum likelihood estimation of the functional principal components from sparse longitudinal data. Journal of Computational and Graphical Statistics. December 1, 2009, 18(4): 995-1015

James, G. M., Hastie, T. J. and Sugar, C. A. (2000) Principal component models for sparse functional data. Biometrika, 87, 587-602.

Yao, F., Mueller, H.-G. and Wang, J.-L. (2005) Functional data analysis for sparse longitudinal data. Journal of the American Statistical Association 100, 577-590

See Also

fpca.mle for model fitting, fpca.score for fpc scores, fpca.pred for prediction

---

`fpca.mle` *Restricted MLE of Functional Principal Components*

Description

A function to obtain restricted MLE of functional principal components by Newton-Raphson algorithm for (sparsely and irregularly observed) longitudinal data (Peng and Paul, 2009). Subjects with only one measurement will be automatically excluded by 'fpca.mle'. Acknowledgements: The code for EM (as initial estimate) is provided by Professor G. James in USC (with slight modifications).

Usage

```
fpca.mle(data.m, M.set,r.set, ini.method="EM", basis.method="bs", sl.v=rep(0.5,10), max.step=50,
grid.l=seq(0,1,0.01), grids=seq(0,1,0.002))
```

Arguments

`data.m`	Matrix with three columns. Each row corresponds to one measurement for one subject. Column One: subject ID (numeric or string); Column 2: measurement (numeric); Column 3: corresponding measurement time (numeric); Missing values are not allowed.
`M.set`	numeric vector with integer values (>=4). Its elements M denote the number of basis functions used in the model for representing the eigenfunctions.
`r.set`	numeric vector with integer values (>=1). Its elements r denote the dimension of the process (number of nonzero eigenvalues) used in the model.
`ini.method`	string. It specifies the initial method for Newton. Its value is either "EM" (default): EM algorithm by James et al. 2002; or "loc": the local linear method by Yao et al. 2005.

`basis.method` string. It specifies the basis functions. Its value is either "bs" (default): cubic B-splines with equally spaced knots; or "ns": truncated basis cubic splines with equally spaced knots. Given basis.method, each combination of M and r specifies a model.

`sl.v` numeric vector. An ordinary Newton step has length 1 which could be too large in the initial few steps. This vector specifies the step length for the first K steps, where K is the length of sl.v. If K >= max.step (see below), then sl.v will be truncated at max.step. If K < max.step, then the steps after the K-th step will have length 1. The default value of sl.v sets the first 10 steps of Newton to be of length 0.5.

`max.step` integer. It is the maximum number of iterations Newton will run. Newton will be terminated if max.step number of iterations has been achieved even if it has not converged.

`grid.l` numeric vector ranging from 0 to 1. This specifies the grid used by the local linear method (when "loc" is the initial method). Note that, due to the "sm" package used for fitting "loc", this grid can not be too dense; the default is grid.l=seq(0,1,0.01).

`grids` numeric vector ranging from 0 to 1. This specifies the grid used by EM (when EM is the initial method) and Newton. Note that, for both grid.l and grids, the denser the grid is, the more computation is needed.

Details

'fpca.mle' uses the Newton-Raphson algorithm on a `Stiefel manifold` to obtain the restricted maximum likelihood estimator of the functional principal components from longitudinal data. It also performs model selection over (M and r) based on an approximate leave-one-curve-out cross-validation score.

Value

A list with eight components

`selected_model` table. the selected M (number of basis functions) and r (dimension of the process).

`eigenfunctions` numeric matrix. The estimated eigenfunctions under the selected model, evaluated at "grid" (see below). dimension: r.select by grid_length

`eigenvalues` numeric vector. The estimated eigenvalues under the selected model.

`error_var` numeric value. The estimated error variance under the selected model.

`fitted_mean` numeric vector. The estimated mean curve (by local linear fitting) evaluated at "grid".

`grid` numeric vector ranging from L1 and L2. This is the grid of time points rescaled to fit the actual time domain of the data, where L1 is the earliest time point, and L2 is the latest time point in the data.

`cv_scores` numeric matrix. Approximate cv score for each combination of M and r.

`converge` numeric matrix. Indicates the convergence of Newton for each combination of M and r. If an entry is less than 1e-3, it indicates that Newton converged under the corresponding model; otherwise it has not converged.

Author(s)

J. Peng, D. Paul

References

Peng, J. and Paul, D. (2009). A geometric approach to maximum likelihood estimation of the functional principal components from sparse longitudinal data. Journal of Computational and Graphical Statistics. December 1, 2009, 18(4): 995-1015

James, G. M., Hastie, T. J. and Sugar, C. A. (2000) Principal component models for sparse functional data. Biometrika, 87, 587-602.

Yao, F., Mueller, H.-G. and Wang, J.-L. (2005) Functional data analysis for sparse longitudinal data. Journal of the American Statistical Association 100, 577-590

See Also

fpca.score for fpc scores, fpca.pred for prediction

Examples

```
############################## example I: "easy" case

##load data
data(easy)

##sample trajectory
 plot(easy$data[,3],easy$data[,2],xlab="grid",ylab="",type='p')

  for(i in 1:500){
   cur<-easy$data[easy$data[,1]==i,]
   points(cur[,3],cur[,2],type="l")

  }

## candidate models for fitting
  M.set<-c(4,5,6)
  r.set<-c(2,3,4)

##parameters for fpca.mle
  ini.method="EM"
  basis.method="bs"
  sl.v=rep(0.5,10)
  max.step=50
  grid.l=seq(0,1,0.01)
  grids=seq(0,1,0.002)


 ##fit candidate models by fpca.mle
 result<-fpca.mle(easy$data, M.set,r.set,ini.method, basis.method,sl.v,max.step,grid.l,grids)
 summary(result)

 ##rescaled grid
```

```
grids.new<-result$grid

##model selection result: the true model M=5, r=3 is selected with the smallest CV score among all converged mode

M<-result$selected_model[1]
r<-result$selected_model[2]


##compare estimated eigenvalues with the truth
evalest<-result$eigenvalues     ## estimated
easy$eigenvalues    ## true

##compare estimated error variance with the truth
sig2est<-result$error_var        ## estimated
easy$error_sd^2     ## true

##plot: compare estimated eigenfunctions with the truth
eigenfest<-result$eigenfunctions
eigenf<-easy$eigenfunctions  ##true
par(mfrow=c(2,2))
for(i in 1:r){
plot(grids.new,eigenfest[i,],ylim=range(eigenfest),xlab="time",ylab=paste("eigenfunction",i))
points(grids, eigenf[i,],col=5,type="o")
}

##plot: compare estimated mean curve with the truth

muest<-result$fitted_mean
plot(grids.new,muest,xlab="time",ylab="mean curve",ylim=range(result$fitted_mean))
points(grids,numeric(length(grids)),col=5)
par(mfrow=c(1,1))

##look at the CV scores and convergence for each model: note that model (M=5, r=4) does not converge.
result$cv_scores   ##CV
result$converge   ##convergence


## derive fpc scores and look at the predicted curve
#fpc scores
fpcs<-fpca.score(easy$data,grids.new,muest,evalest,eigenfest,sig2est,r)
#get predicted trajectories on a fine grid: the same grid for which mean and eigenfunctions are evaluated
pred<-fpca.pred(fpcs, muest,eigenfest)

#get predicted trajectories on the observed measurement points
N<-length(grids.new)

par(mfrow=c(3,3))
for (i in 1:9){
id<-i                                   ##for curve i
t.c<-easy$data[easy$data[,1]==id,3]    ##measurement points
t.proj<-ceiling(N*t.c)                  ##measurement points projected on the grid
y.c<-easy$data[easy$data[,1]==id,2]    ##obs
y.pred.proj<-pred[t.proj,id]            ##predicted obs on the measurement points
```

```
#plots
plot(t.c,y.c,ylim=range(pred[,id]),xlab="time",ylab="obs", main=paste("predicted trajectory of curve", id))
points(grids.new,pred[,id],col=3,type='l')
##points(t.c,y.pred.proj,col=2, pch=2)    ##predicted measurements at observed measurement times
}
par(mfrow=c(1,1))



 ############################### example II: "practical" case

 ##load data
 ## Not run:
 data(prac)

 ##sample trajectory
 plot(prac$data[,3],prac$data[,2],xlab="grid",ylab="",type='p')

  for(i in 1:500){
   cur<-prac$data[prac$data[,1]==i,]
   points(cur[,3],cur[,2],type="l")

  }


  ## candidate models for fitting
  M.set<-c(5,10,15,20)
  r.set<-5

  ##parameters for fpca.mle
  ini.method="EM"
  basis.method="bs"
  sl.v=rep(0.5,10)
  max.step=50
  grid.l=seq(0,1,0.01)
  grids=seq(0,1,0.002)


 ##fit candidate models by fpca.mle
 result<-fpca.mle(prac$data, M.set,r.set,ini.method, basis.method,sl.v,max.step,grid.l,grids)
 summary(result)

 ##rescaled grid
 grids.new<-result$grid

 ##model selection result: the true model M=5, r=3 is selected with the smallest CV score among all converged mode
 M<-result$selected_model[1]
 r<-result$selected_model[2]

 ##compare estimated eigenvalues with the truth
 result$eigenvalues     ## estimated
```

```
prac$eigenvalues    ## true

##compare estimated error variance with the truth
result$error_var        ## estimated
prac$error_sd^2     ## true

##plot: compare estimated eigenfunctions with the truth
eigenf<-prac$eigenfunctions  ##true
par(mfrow=c(2,3))
for(i in 1:r){
plot(grids.new,result$eigenfunctions[i,],ylim=range(result$eigenfunctions),xlab="time",ylab=paste("eigenfunct
points(grids, eigenf[i,],col=5,type="o")
}

##plot: compare estimated mean curve with the truth
plot(grids.new,result$fitted_mean,xlab="time",ylab="mean curve",ylim=range(result$fitted_mean))
points(grids,numeric(length(grids)),col=5)
par(mfrow=c(1,1))

##look at the CV scores and convergence for each model: note that model (M=5, r=4) does not converge.
result$cv_scores   ##CV
result$converge   ##convergence

## End(Not run)
```

`fpca.pred` *Predicted trajectories*

Description

A function to predict trajectory for each subject

Usage

```
fpca.pred(fpcs, muhat,eigenfuncs)
```

Arguments

`fpcs`	Functional principal component scores. An estimate is returned by fpca.score
`muhat`	Mean curve evaluated on a grid. An estimate is returned by fpca.mle.
`eigenfuncs`	Eigenfunctions evaluated on the same grids as in 'muhat'. An estimate is returned by fpca.mle.

Details

'fpca.pred' gives predicted trajectories (evaluated on a fine grid).

Value

A matrix where each column corresponds to the predicted trajectory of a subject.

Author(s)

J. Peng, D. Paul

References

Peng, J. and Paul, D. (2009). A geometric approach to maximum likelihood estimation of the functional principal components from sparse longitudinal data. Journal of Computational and Graphical Statistics. December 1, 2009, 18(4): 995-1015

James, G. M., Hastie, T. J. and Sugar, C. A. (2000) Principal component models for sparse functional data. Biometrika, 87, 587-602.

Yao, F., Mueller, H.-G. and Wang, J.-L. (2005) Functional data analysis for sparse longitudinal data. Journal of the American Statistical Association 100, 577-590

See Also

fpca.mle for model fitting, fpca.score for fpc scores

`fpca.score` *Functional principal component scores*

Description

A function to estimate the functional principal component scores by the best linear unbiased predictors (Yao et al. 2005).

Usage

```
fpca.score(data.m,grids.u,muhat,eigenvals,eigenfuncs,sig2hat,K)
```

Arguments

`data.m`	Matrix with three columns. Each row corresponds to one measurement for one subject. Column One: subject ID (numeric or string); Column 2: measurement (numeric); Column 3: corresponding measurement time (numeric); Missing values are not allowed. Same format as the data input for fpca.mle.
`grids.u`	Grid of time points used in evaluating the mean and eigenfunctions (on the original scale). Same as 'grid' returned by fpca.mle.
`muhat`	Mean evaluated on the same grids as in grids.u. An estimate is returned by fpca.mle.
`eigenvals`	Eigenvalues. An estimate is returned by fpca.mle.
`eigenfuncs`	Eigenfunctions evaluated on the same grids as in grids.u. An estimate is returned by fpca.mle.
`sig2hat`	Noise variance. An estimate is returned by fpca.mle.
`K`	Number of eigenfunctions used to derive the fpc scores.

Details

'fpca.score' uses best linear unbiased predictors (BLUP) to estimate the functional principal component scores for each subject

Value

An n by K matrix containing the first K functional principal component scores for each subject.

Author(s)

J. Peng, D. Paul

References

Peng, J. and Paul, D. (2009). A geometric approach to maximum likelihood estimation of the functional principal components from sparse longitudinal data. Journal of Computational and Graphical Statistics. December 1, 2009, 18(4): 995-1015

James, G. M., Hastie, T. J. and Sugar, C. A. (2000) Principal component models for sparse functional data. Biometrika, 87, 587-602.

Yao, F., Mueller, H.-G. and Wang, J.-L. (2005) Functional data analysis for sparse longitudinal data. Journal of the American Statistical Association 100, 577-590

See Also

fpca.mle for model fitting, fpca.pred for prediction

---

prac *An example with M=10 (basis functions) and r=5 (non-zero eigenvalues)*

Description

A simulated dataset as an example which corresponds to the "practical" case in the paper

Usage

```
data(prac)
```

Format

prac is a list with six components (in the given order):

data data matrix with three columns: column 1–ID, column 2–measurement, column 3–time.

eigenfunctions true eigenfunctions: generated from cubic Bsplines with M=10 equally spaced knots.

eigenvalues true eigenvalues: first–1, second–0.66, third–0.52, fourth–0.44, fifth–0.38, others–zero.

number_of_basis true number of basis functions: M=10.

dimension true dimension of the process: r=5.

error_sd true error standard deviation: 0.25.

Details

mean curve of the process is zero; principal component scores and errors are all i.i.d N(0,1); there are 500 subjects, and each has 2~10 measurements uniformly distributed on [0,1]; in total there are 3018 measurements

Index