Package 'GpOutput2D'

November 28, 2020

Type Package

Title What the Package Does (Title Case)

Version 0.1.0

Author Tran Vi-vi Elodie PERRIN <tran-vivi-elodie.perrin@outlook.fr>

Maintainer Tran Vi-vi Elodie PERRIN <tran-vivi-elodie.perrin@outlook.fr>

Description Package for two-dimensional functional data analysis (for instance : data set of images or maps). Functions for performing Functional Principal Component Analysis (FPCA) on such data have been developped. Furthermore, the package contains methods for Computer Experiments by using kriging methods : metamodelling of models with two-dimensional functional output.

Imports lhs,DiceDesign,stats,graphics,doParallel,fields,pracma,foreach

Depends waveslim, orthogonalsplinebasis,DiceKriging,kergp

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

R topics documented:

`GpOutput2D-package`	*Memodelling for model with two-dimensional functional output by using Functional Principal Component Analysis and kriging methods*

Description

Package for two-dimensional functional data analysis (for instance : data set of images or maps). Functions for performing Functional Principal Component Analysis (FPCA) on such data have been developped. Furthermore, the package contains methods for Computer Experiments by using kriging methods : metamodelling of models with two-dimensional functional output.

Details

Package :	GpOutput2D
Type :	Package
Version :	0.4
Date :	2020-11-15
License :	GPL (=2.0)
LazyLoad :	yes

Note

Important functions or methods :

- `Fpca2d` : Functional Principal Component Analysis (FPCA) for two-dimensional data (images, maps,...).
- `km_Fpca2d` : Gaussian Process Model on principal components of `Fpca2d`, by using `DiceKriging` package.
- `gp_Fpca2d` : Gaussian Process Model on principal components of `Fpca2d`, by using `kergp` package.
- `predict` : Prediction of the objective function at new points using `km_Fpca2d` or `gp_Fpca2d` model (Simple and Universal Kriging).

Author(s)

Tran Vi-vi Élodie PERRIN (tran-vivi-elodie.perrin@outlook.fr)

References

test

Campbell2D *Example of function called Campbell2D*

Description

Example of function called Campbell2D

Usage

```
Campbell2D(X, z1, z2)
```

Arguments

X	a matrix with eight columns, which correspond to the function input.
z1, z2	vectors with the spatial coordinates.

Value

a three-dimensional array with the output maps. The first two dimensions correspond to the maps dimension. The third dimension corresponds to the number of maps (or simulations).

Examples

```
# inputs of the Campbell2D function
x1<-rep(-1,8); x2<-rep(5,8); x3<-c(5,3,1,-1,5,3,1,-1)
X <- rbind(x1,x2,x3)

# spatial domain of the Campbell2D output
nz<-64 # root of the size of the spatial domain
z<-seq(-90,90,length=nz)

# Campbell2D function
Y = Campbell2D(X,z,z)
```

coef.OrthoNormalBsplines2D *Get control points of two-dimensional B-splines basis*

Description

Get control points of two-dimensional B-splines basis

Usage

```
## S3 method for class 'OrthoNormalBsplines2D'
coef(object, y, ...)
```

Arguments

object	an object of class OrthoNormalBsplines2D with functions of the B-splines basis.
y	two-dimensional data to approximate on B-splines basis.
...	other arguments.

Value

a matrix with control points on two-dimensional B-splines basis.

See Also

OrthoNormalBsplines2D Inverse2D.OrthoNormalBsplines2D Inverse2D

---

error.predict *RMSE and Q2 computation*

Description

RMSE and Q2 computation

Usage

```
error.predict(
  y,
  py,
  fpca,
  sd.epsilon = 0,
  scores.epsilon = 0,
  rtx.scores = FALSE
)
```

Arguments

y	an array with real two dimensional functional outputs.
py	Predictions getting with km_Fpca2d. The two first dimensions correspond to data dimensions, which are denoted M and N. The third one is the size of the data set.
fpca	an object of class Fpca2d.
sd.epsilon	a value with y standard deviation threshold. Q2 values associated to standard deviations less or equal to sd.epsilon are NA. The default is 0. This is to avoid infinite values of Q2.
scores.epsilon	a value with y scores standard deviation threshold. Q2 values less or equal to scores.epsilon are NA. The default is 0. This is to avoid infinite values of Q2.
rtx.scores	a logical value. If TRUE, rmse and Q2 of the predicted scores are returned. The default is FALSE

Value

- if rtx.scores=FALSE, a list with the following items is returned :
  - `rmse` : rmse of `y` prediction.
  - `Q2` : rmse of `y` prediction.
- if rtx.scores=TRUE, a list with the following items is returned :
  - `scores` :
    - `rmse` : rmse of `y` score prediction.
    - `Q2` : Q2 of `y` score prediction.
  - `y` :the same returned list as for rtx.scores=FALSE

`Fpca2d` *Two-dimensional Functional Principal Component Analysis (FPCA).*

Description

A wrapper function to perform FPCA on two-dimensional data (images, maps, etc.), given a implementation method. For instance, the valid methods are two methods based on projection method (here, "Wavelets" and "Bsplines").

Usage

```
Fpca2d(method = c("Bsplines", "Wavelets"), ...)
```

Arguments

`method`	a character string which specifies the method used to implement FPCA.
`...`	further parameters for the basis decomposition : see `Fpca2d.Wavelets`, for using wavelets, and `Fpca2d.Bsplines`, for using B-splines.

Value

an object of class "Fpca2d" which is a list with the following items :

- `sdev` : the standard deviations of the principal components (i.e., the square roots of the eigenvalues of the covariance operator).
- `EigFct` : a three dimensional array with the eigenfunctions. The two first dimensions correspond to data (images, maps, etc.) dimensions. The third one is the maximal number of principal component used.
- `x` : if `retx` is `TRUE` (see `prcomp`). Data scores (coordinates in the eigen basis) are returned.
- `center`, `scale` : a logical value which indicates if data coefficients are respectively centered or scaled.

Author(s)

Tran Vi-vi Elodie PERRIN

See Also

`Fpca2d.Wavelets`

Examples

```
################################
### two-dimensional data set ###
################################
n<-200 # size of the learning sample
nz<-64; z <- seq(-90,90,length=nz) # spatial domain

### inputs of Campbell2D ###
library(lhs)
library(DiceDesign)

x <- maximinLHS(n=n,k=8)
X <-maximinSA_LHS(x)$design
X<-X*6 -1

# Campbell2D
Y <- Campbell2D(X,z,z)

#############
### FPCA ###
#############


### using wavelet basis ###
fpca_w<- Fpca2d(Y,method="Wavelets",
                wf="d4", J=1, # wavelet parameters
                ncoeff=1200, rank.=5) # FPCA configuration

plot(fpca_w,type="eigenfunctions")
plot(fpca_w)

## Not run:
### using B-splines basis ###

# knots for B-splines basis
K<-35
z.knots <- seq(-90,90,length=K)

fpca_Bs<- Fpca2d(Y,method="Bsplines",
                z1=z,z2=z,z1.knots=z.knots,z2.knots=z.knots, # wavelet parameters
                ncoeff=1225, rank.=2) # FPCA configuration

plot(fpca_Bs,type="eigenfunctions")
plot(fpca_Bs)

## End(Not run)
```

`Fpca2d.Bsplines` *Two dimensional functional principal component analysis (FPCA) by using B-splines basis*

Description

Two dimensional functional principal component analysis (FPCA) by using B-splines basis

Usage

```
Fpca2d.Bsplines(
  x,
  z1,
  z2,
  z1.knots,
  z2.knots,
  norder = 2,
  ortho = "Redd",
  expand_knots = FALSE,
  p = 1,
  ncoeff = NULL,
  center = TRUE,
  scale. = FALSE,
  ...
)
```

Arguments

`x`	a three dimensional array, which contains two-dimensional data (images, maps, etc.). The two first dimensions correspond to data dimensions, which are denoted M and N. The third one is the size of the data set.
`z1, z2`	two vectors of argument values at which the B-spline basis functions are to be evaluated.
`z1.knots, z2.knots`	The full set of knots used to define the basis functions.
`norder`	Order of the spline fit (degree= order-1). The default is 2.
`ortho`	a character string which specify the orthogonalization method. If it is "Redd", the `orthogonalsplinebasis` package is used. If it is "GS", the Gram-Schmidt is used. The default is "Redd".
`expand_knots`	a boolean. If it is TRUE, knots are expanded for appropriate number of knots in bsplines (see `expand.knots`). The default is FALSE.
`p`	a value which fixes the total mean proportion of energy (or mean spatial variance). The number of coefficients (`ncoeff`) used for PCA is calibrated according to its value. The default is 1. If a value is given in `ncoeff`, `p` is not used.
`ncoeff`	a value which fixes the number of coefficients used for PCA. The default is `NULL`. If it is `NULL`, the number of coefficients is calibrated by using the parameter `p`.
`center`	a logical value indicating whether the coefficients should be shifted to be zero centered. The default is `TRUE`.
`scale.`	a logical value indicating whether the coefficients should be scaled to have unit variance before the analysis takes place. The default is `FALSE`.
`...`	arguments of `prcomp`, which can fix characteristics of PCA.

Value

an object of class "Fpca2d" which is a list with the following items :

- `sdev` : the standard deviations of the principal components (i.e., the square roots of the eigenvalues of the covariance operator).

- `EigFct` : a three dimensional array with the eigenfunctions. The two first dimensions correspond to data (images, maps, etc.) dimensions. The third one is the maximal number of principal component used.
- `x` : if `retx` is `TRUE` (see `prcomp`). Scores (coordinates in the eigen basis) of the data are returned.
- `center`, `scale` : a logical value which indicates if data coefficients are respectively centered or scaled.

See Also

`Fpca2d`

`Fpca2d.Wavelets` *Two dimensional functional principal component analysis (FPCA) by using wavelet basis*

Description

performs FPCA on two-dimensional functional data (data set of images, maps, ...). The implementation of FPCA is based on wavelet basis.

Usage

```
Fpca2d.Wavelets(
  x,
  wf,
  J,
  boundary = "periodic",
  p = 1,
  ncoeff = NULL,
  center = TRUE,
  scale. = FALSE,
  ...
)
```

Arguments

`x`	a three dimensional array, which contains two-dimensional data (images, maps, etc.). The two first dimensions correspond to data dimensions, which are denoted M and N. The third one is the size of the data set.
`wf`	a character string which specifies the wavelet filter (see `dwt.2d`).
`J`	depth of the wavelet decomposition, must be a number less than or equal to log(min(M,N),2).
`boundary`	a character string which specifies how boundaries are treated. Only "periodic" is currently implemented (see `dwt.2d`).
`p`	a value which fixes the total mean proportion of energy (or mean spatial variance). The number of coefficients (`ncoeff`) used for PCA is calibrated according to its value. The default is 1. If a value is given in `ncoeff`, `p` is not used.
`ncoeff`	a value which fixes the number of coefficients used for PCA. The default is `NULL`. If it is `NULL`, the number of coefficients is calibrated by using the parameter `p`.

center	a logical value indicating whether the coefficients should be shifted to be zero centered. The default is TRUE.
scale.	a logical value indicating whether the coefficients should be scaled to have unit variance before the analysis takes place. The default is FALSE.
...	arguments of prcomp, which can fix characteristics of PCA.

Value

an object of class "Fpca2d" which is a list with the following items :

- sdev : the standard deviations of the principal components (i.e., the square roots of the eigenvalues of the covariance operator).
- EigFct : a three dimensional array with the eigenfunctions. The two first dimensions correspond to data (images, maps, etc.) dimensions. The third one is the maximal number of principal component used.
- x : if retx is TRUE (see prcomp). Scores (coordinates in the eigen basis) of the data are returned.
- center, scale : a logical value which indicates if data coefficients are respectively centered or scaled.

See Also

Fpca2d

---

gp_Fpca2d *Gaussian Process Model on principal components of* Fpca2d*, by using* kergp *package*

Description

the function gp of the kergp package is use to fit kriging models on each principal component modeled on Fpca2d (for more details, see gp).

Usage

```
gp_Fpca2d(formula = ~1, design, response, cov, estim = TRUE, ...)
```

Arguments

formula	an object of class "formula" (or a list of "formula" which the length is equal to the number of modeled principal component) specifying the linear trend of the kriging model (see lm) on each principal component. This formula should concern only the input variables (design), and not the output (response). The default is ~1, which defines a constant trend on each principal component.
design	a data frame representing the design of experiments. The ith row contains the values of the d input variables corresponding to the ith evaluation.
response	n object of class Fpca2d which contains eigen decomposition of the model/function ouput.
cov	a covariance kernel object or call (or a list of covariance kernel objects or call)

estim	Logical. If TRUE, the model parameters are estimated by Maximum Likelihood. The initial values can then be specified using the parCovIni and varNoiseIni arguments of mle,covAll-method passed though dots. If FALSE, a simple Generalized Least Squares estimation will be used, see gls,covAll-method. Then the value of varNoise must be given and passed through dots in case noise is TRUE.
...	other inputs of gp.

Value

a list of object of class gp for each modeled principal component.

See Also

gp kergp

Examples

```
################################
### two-dimensional data set ###
################################
n<-200 # size of the learning sample
nz<-64; z <- seq(-90,90,length=nz) # spatial domain

### inputs of Campbell2D ###
library(lhs)
library(DiceDesign)

x <- maximinLHS(n=n,k=8)
X <-maximinSA_LHS(x)$design
X<-X*6 -1

# Campbell2D
Y <- Campbell2D(X,z,z)

# change X on data.frame
colnames(X)<-paste("x",1:8,sep="")

############
### FPCA ###
############

### by using wavelet basis ###
fpca_w<- Fpca2d(Y,method="Wavelets",
                wf="d4", J=1, # wavelet parameters
                ncoeff=1200, rank.=2) # FPCA configuration

####################
###              ###
### Kriging model ###
###              ###
####################

#----------------------------------#
#----------------------------------#
#   Example by using wavelet basis   #
#----------------------------------#
```

```
#-----------------------------------#

#-------------------------------------------#
#  Same kernel for all principal components  #
#-------------------------------------------#

## kernel
myCov <- covTS(inputs = colnames(X),
kernel = "k1Matern5_2",
dep = c(range = "input"),
value = c(range = 0.4))

myGp<- gp_Fpca2d(design=X, response=fpca_w, cov=myCov,estim=FALSE)

#-------------------------------------------------------------#
#  Different kernel and formula for each principal component  #
#-------------------------------------------------------------#

## Not run:
## kernel of firt principal component
myCov1<-myCov

## kernel of second principal component
myCov2 <- covTS(inputs = colnames(X),
kernel = "k1Matern3_2",
dep = c(range = "input"),
value = c(range = 0.4))

## List of both kernels
myCovList <- list(myCov1,myCov2)

## Gp model
myGp2<- gp_Fpca2d(formula=list(~1,~x1+x2+x3+x4+x5+x6+x7+x8),
           design=X, response=fpca_w, cov=myCovList,estim=FALSE)

## End(Not run)
##################
### Prediction ###
##################

NewX<-matrix(runif(5*8,min=-1,max=5),ncol=8) # newdata
RealY <- Campbell2D(NewX,z,z)# real maps

# change NewX on data.frame
colnames(NewX)<-colnames(X)

#-----------------------------------#
#-----------------------------------#
#   Example by using wavelet basis   #
#-----------------------------------#
#-----------------------------------#
pw.UK <- predict(myGp,NewX,"UK")

################################
### Prediction RMSE and Q2  ###
################################
```

```
#-----------------------------------#
#-----------------------------------#
#   Example by using wavelet basis   #
#-----------------------------------#
#-----------------------------------#
err.pw.UK <-error.predict(RealY,pw.UK,fpca_w,rtx.scores=TRUE)

### scores ###
print(err.pw.UK$scores$rmse)
print(err.pw.UK$scores$Q2)

### images/maps ###
library(fields)
image.plot(err.pw.UK$y$rmse, main="RMSE")
image.plot(err.pw.UK$y$Q2, main="Q2")
```

`Inverse2D` *Inverse transform of wavelet or orthonormal B-splines basis.*

Description

embeds the coefficients of "Wavelet2D" or "OrthonormalBsplines2D" basis onto two-dimensional domain.

Usage

```
Inverse2D(object, ...)
```

Arguments

`object`	an object of class `Wavelet2D` or `OrthoNormalBsplines2D`.
`...`	if `object` is an object of class `OrthoNormalBsplines2D`, the matrix of control points must be given (see `Inverse2D.OrthoNormalBsplines2D`).

Value

a three dimensional array. The first two dimensions correspond to maps dimensions. The third dimension corresponds to the size of the data set.

Author(s)

Tran Vi-vi Elodie PERRIN

See Also

`Inverse2D.OrthoNormalBsplines2D` `Inverse2D.Wavelet2D`

Inverse2D.OrthoNormalBsplines2D *Inverse transform of control points of two-dimensional B-splines basis*

Description

Inverse transform of control points of two-dimensional B-splines basis

Usage

```
## S3 method for class 'OrthoNormalBsplines2D'
Inverse2D(object, coeff, ...)
```

Arguments

object	an object of class OrthoNormalBsplines2D with functions of the B-splines basis.
coeff	matrix with control points on two-dimensional B-splines basis.
...	other arguments.

Value

an array with two-dimensional approximated data.

See Also

coef.OrthoNormalBsplines2D OrthoNormalBsplines2D Inverse2D

Inverse2D.Wavelet2D *Multiple two-dimensional inverse wavelet transform*

Description

performs inverse wavelet transform of each set of wavelet coefficients from a data set.

Usage

```
## S3 method for class 'Wavelet2D'
Inverse2D(object, ...)
```

Arguments

object	an object of class Wavelet2D.
...	other arguments

Value

a three dimensional array. The first two dimensions correspond to maps dimensions. The third dimension corresponds to the size of the data set

Author(s)

Tran Vi-vi Elodie PERRIN

See Also

Wavelet2D

---

km_Fpca2d *Gaussian Process Model on principal components of* Fpca2d*, by using* DiceKriging *package.*

Description

the function km of the DiceKriging package is use to fit kriging models on each principal component modeled on Fpca2d (for more details, see km).

Usage

```
km_Fpca2d(
  formula = ~1,
  design,
  response,
  parallel = FALSE,
  covtype = "matern5_2",
  coef.trend = NULL,
  coef.cov = NULL,
  coef.var = NULL,
  nugget = NULL,
  noise.var = NULL,
  lower = NULL,
  upper = NULL,
  parinit = NULL,
  multistart = 1,
  kernel = NULL,
  ...
)
```

Arguments

formula	an object of class "formula" (or a list of "formula" which the length is equal to the number of modeled principal components) specifying the linear trend of the kriging model (see lm) on each principal component. This formula should concern only the input variables (design), and not the output (response). The default is ~1, which defines a constant trend on each principal component.
design	a data frame representing the design of experiments. The ith row contains the values of the d input variables corresponding to the ith evaluation.
response	an object of class Fpca2d which contains eigen decomposition of the model/function ouput.
parallel	a logical value specifying if parallelization is done on principal components. The default is FALSE.

covtype	optional character string or vector of character strings specifying the covariance structure to be used on each modeled principal component (see km for possible inputs of covtype). If a vector, the length should be equal to the number of modeled principal components.
coef.trend, coef.cov, coef.var	optional vectors or matrices containing the values for the trend, covariance and variance parameters. If matrices, the number of rows should be equal to the number of modeled principal components. For details, see km).
nugget	an optional variance value or vector standing for the homogeneous nugget effect. If vector, the length should be equal to the number of modeled principal components.
noise.var	an optional vector or matrix containing the noise variance at each observation on each modeled principal component.
lower, upper	optional vectors or matrices containing the bounds of the correlation parameters of each principal component for optimization. For details, see km). If matrices, the number of rows should be equal to the number of modeled principal components.
parinit	an optional vector or matrix containing the initial values for the variables to be optimized over. For details, see km).
multistart	an optional integer indicating the number of initial points from which running the BFGS optimizer. (see km).
kernel	an optional function or list of functions containing a new covariance structure for each principal component. At this stage, the parameters must be provided as well, and are not estimated.
...	other parameters of km function from DiceKriging. (see DiceKriging)

Value

a list of object of class km for each modeled principal component.

Author(s)

Tran Vi-vi Elodie PERRIN

See Also

km DiceKriging

Examples

```
################################
###   Learning Sample        ###
################################
n<-200 # size of the learning sample
nz<-64; z <- seq(-90,90,length=nz) # spatial domain

### inputs of Campbell2D ###
library(lhs)
library(DiceDesign)

x <- maximinLHS(n=n,k=8)
X <-maximinSA_LHS(x)$design
```

```
X<-X*6 -1

# Campbell2D
Y <- Campbell2D(X,z,z)

# change X on data.frame
colnames(X)<-paste("x",1:8,sep="")

############################
###   Test Sample        ###
############################

NewX<-matrix(runif(5*8,min=-1,max=5),ncol=8) # newdata
RealY <- Campbell2D(NewX,z,z)# real maps

# change NewX on data.frame
colnames(NewX)<-colnames(X)


######################################
#
#   Example by using wavelets
#
######################################

############
### FPCA ###
############

fpca_w<- Fpca2d(Y,method="Wavelets",
                wf="d4", J=1, # wavelet parameters
                ncoeff=1200, rank.=1) # FPCA configuration

####################
### Kriging model ###
####################

mw <- km_Fpca2d(design=X,response=fpca_w,control=list(trace=FALSE))

#---------------------------------------------------------
# (same example) To fix different kernel and formula
#                for each principal component
#---------------------------------------------------------
## Not run:
mw <- km_Fpca2d(formula=list(~.,~1)),
                design=X,response=fpca_w,
                covtype = c("matern5_2","matern3_2"),
                control=list(trace=FALSE))

## End(Not run)

#---------------------------------------------------------
# (same example) how to use the multistart argument of km
#---------------------------------------------------------
## Not run:
nCores <- 2
require(doParallel)
```

```
cl <-  makeCluster(nCores)
registerDoParallel(cl)
mw <- km_Fpca2d(design=X,response=fpca_w,multistart=4,control=list(trace=FALSE))
stopCluster(cl)

## End(Not run)

######################
###   Prediction   ###
######################

pw.UK <- predict(mw,NewX,"UK")

####################
### RMSE and Q2  ###
####################

err.pw.UK <-error.predict(RealY,pw.UK,fpca_w,rtx.scores=TRUE)

### scores ###
print(err.pw.UK$scores$rmse)
print(err.pw.UK$scores$Q2)

### images/maps ###
library(fields)
image.plot(err.pw.UK$y$rmse, main="RMSE")
image.plot(err.pw.UK$y$Q2, main="Q2")


#####################################
#
#   Example by using B-splines
#
#####################################

## Not run:
############
### FPCA ###
############

### using B-splines basis ###

# knots for B-splines basis
K<-35
z.knots <- seq(-90,90,length=K)

fpca_Bs<- Fpca2d(Y,method="Bsplines",
                z1=z,z2=z,z1.knots=z.knots,z2.knots=z.knots, ortho="GS",
                expand_knots=TRUE,# B-splines parameters
                ncoeff=1225, rank.=5) # FPCA configuration


####################
### Kriging model ###
####################

mB <- km_Fpca2d(design=X,response=fpca_Bs,control=list(trace=FALSE))
```

```
##################
### Prediction ###
##################

pB.UK <- predict(mB,NewX,"UK")

########################
###   RMSE and Q2    ###
########################

err.pB.UK <-error.predict(RealY,pB.UK,fpca_Bs,rtx.scores=TRUE)

### scores ###
print(err.pB.UK$scores$rmse)
print(err.pB.UK$scores$Q2)

### images/maps ###
library(fields)
image.plot(err.pB.UK$y$rmse, main="RMSE")
image.plot(err.pB.UK$y$Q2, main="Q2")

## End(Not run)
```

`OrthoNormalBsplines2D` *Creating a two-dimensional othonormal B-splines basis*

Description

Creating a two-dimensional othonormal B-splines basis

Usage

```
OrthoNormalBsplines2D(
  x,
  y,
  x.knots,
  y.knots,
  order = 2,
  ortho = "Redd",
  expand_knots = TRUE,
  ...
)
```

Arguments

`x, y`	two vectors of argument values at which the B-spline basis functions are to be evaluated.
`x.knots, y.knots`	The full set of knots used to define the basis functions.
`order`	Order of the spline fit (degree= order-1). The default is 2.

ortho	a character string which specify the orthogonalization method. If it is "Redd", the orthogonalsplinebasis package is used. If it is "GS", the Gram-Schmidt is used. The default is "Redd".
expand_knots	a boolean. If it is TRUE, knots are expanded for appropriate number of knots in bsplines (see expand.knots). The default is TRUE.
...	other arguments (see SplineBasis).

Value

an array with two dimensional orthonormal B-splines. The two first dimensions correspond to the data dimensions. The last two dimensions correspond to the basis dimensions.

See Also

coef.OrthoNormalBsplines2D Inverse2D.OrthoNormalBsplines2D Inverse2D

Examples

```
####################################
#
#  Create two-dimensional data set
#
####################################

# inputs of the Campbell2D function
x1<-rep(-1,8); x2<-rep(5,8); x3<-c(5,3,1,-1,5,3,1,-1)
X <- rbind(x1,x2,x3)

# spatial domain of the Campbell2D output
nz<-64 # root of the size of the spatial domain
z<-seq(-90,90,length=nz)

# Campbell2D function
Y = Campbell2D(X,z,z)

#########################################
#
#  Create two-dimensional B-splines basis
#
#########################################

# knots for B-splines basis
K<-35
z.knots <- seq(-90,90,length=K)

# Generating a two-dimensional othonormal B-splines basis
OPhi <-OrthoNormalBsplines2D(z,z,z.knots,z.knots,ortho="GS")

########################
#
#  Get control points
#
########################

coeff_Y<-coef(OPhi,Y)
```

```
######################
#
#  Approximation of Y
#
########################

hatY <-Inverse2D(OPhi,coeff_Y)
```

`plot.Fpca2d` *Plot for Fpca2d object*

Description

plots characteristic of Fpca2d decomposition : eigenfunctions, coefficients total mean proportion of energy, coefficients truncation etc.

Usage

```
## S3 method for class 'Fpca2d'
plot(
  x,
  type = c("inertia", "energy", "MeanPoe", "SelectedCoeff"),
  PC = 1:2,
  p = seq(0.5, 1, by = 0.05),
  z1 = NULL,
  z2 = NULL,
  ...
)
```

Arguments

`x`	an object of class "Fpca2d".
`type`	a character vector, which the characteristics of `x` are plotted. The default is c("inertia","energy","MeanPoe","SelectedCoeff"). It means that the `barplot` of proportion of inertia of each principal component (`type="inertia"`), the one of the number of coefficients estimated with PCA according to the total mean proportion of energy (`type="energy"`), the image of the coefficients mean proportion of energy (`type="MeanPoe"`), and the image which indicates which coefficients are estimated by PCA or by empirical mean (`type="SelectedCoeff"`). These graphics can be separetely plotted. The eigenfunctions can be plotted be specifying (`type="eigenfunctions"`).
`PC`	a vector with the numbers of which principal components are plotted. The default is c(1,2).
`p`	a vector with the proportion of energy which are plotted. The default is seq(0.5,1,by=0.05).
`z1, z2`	spatial coordinates.
`...`	plot parameters (see `plot`, `image` or `image.plot`).

`predict.gp_Fpca2d` *Predict values and confidence intervals at newdata for a km_Fpca2d object*

Description

Predict values and confidence intervals at newdata for a km_Fpca2d object

Usage

```
## S3 method for class 'gp_Fpca2d'
predict(object, newdata, type, se.compute = TRUE, ...)
```

Arguments

`object`	an object of class `gp_Fpca2d`.
`newdata`	a vector, matrix or data frame containing the points where to perform predictions.
`type`	a character string corresponding to the kriging family, to be chosen between simple kriging ("SK"), or universal kriging ("UK").
`se.compute`	an optional boolean. If FALSE, only the kriging mean is computed. If TRUE, the kriging variance (actually, the corresponding standard deviation) and confidence intervals are computed too.
`...`	see `predict.km`.

Value

a list with the following items :

- `scores_predict` : prediction of scores from two-dimensional FPCA (see `Fpca2d` and `predict.km`).
- `mean` : an array with three dimensions which contains image (or map) kriging means. The two first dimensions corresponds to the output data dimensions. The third correspond to the number of predictions.
- `sd` : it is return only if se.compute = TRUE. If TRUE, an array which contains image (or map) prediction standard deviation.
- `lower95,upper95` : bounds of the 95 Not computed if se.compute=FALSE.

See Also

`predict.km`

`predict.km_Fpca2d` *Predict values and confidence intervals at newdata for a km_Fpca2d object*

Description

Predict values and confidence intervals at newdata for a km_Fpca2d object

Usage

```
## S3 method for class 'km_Fpca2d'
predict(object, newdata, type, se.compute = TRUE, ...)
```

Arguments

`object`	an object of class `km_Fpca2d`.
`newdata`	a vector, matrix or data frame containing the points where to perform predictions.
`type`	a character string corresponding to the kriging family, to be chosen between simple kriging ("SK"), or universal kriging ("UK").
`se.compute`	an optional boolean. If FALSE, only the kriging mean is computed. If TRUE, the kriging variance (actually, the corresponding standard deviation) and confidence intervals are computed too.
`...`	see `predict.km`.

Value

a list with the following items :

- `scores_predict` : prediction of scores from two-dimensional FPCA (see `Fpca2d` and `predict.km`).
- `mean` : an array with three dimensions which contains image (or map) kriging means. The two first dimensions corresponds to the output data dimensions. The third correspond to the number of predictions.
- `sd` : it is return only if se.compute = TRUE. If TRUE, an array which contains image (or map) prediction standard deviation.
- `lower95,upper95` : bounds of the 95 Not computed if se.compute=FALSE.

See Also

`predict.km`

`Wavelet2D` *Multiple two dimensional wavelet transform*

Description

performs two dimensional wavelet transforms on all images/maps from a data set.

Usage

```
Wavelet2D(x, wf, J = 1, boundary = "periodic")
```

Arguments

`x`	a matrix (for one map) or a three dimensional array (for several maps). Two first dimensions correspond to maps dimensions. The third one is the number of maps.
`wf`	name of the wavelet filter to use in the decomposition.
`J`	depth of the decomposition, must be a number less than or equal to log(minM,N,2), with M and N are respectively the number of rows and columns of each map.
`boundary`	a character string which specified the method used for side effect. Only "periodic" is currently implemented.

Value

an object of class "Wavelet2D", which corresponds to a matrix (for one map) or an array (for several maps) of same dimension as `x`, with the wavelet coefficients.

Author(s)

Tran Vi-vi Elodie PERRIN

See Also

`Inverse2D.Wavelet2D` `Inverse2D`

Examples

```
# inputs of the Campbell2D function
x1<-rep(-1,8); x2<-rep(5,8); x3<-c(5,3,1,-1,5,3,1,-1)
X <- rbind(x1,x2,x3)

# spatial domain of the Campbell2D output
nz<-64
z<-seq(-90,90,length=nz)

# Campbell2D function
Y = Campbell2D(X,z,z)

# Wavelet transform
w <-Wavelet2D(Y,wf="d4",J=2)

# Inverse wavelet transform
```

```
hatY <- Inverse2D(w)
```

Index