Package ‘MuFiCokriging’

February 19, 2015

Type Package
Title Multi-Fidelity Cokriging models
Version 1.2
Date 2012-12-21
Author Loic Le Gratiet
Affiliations Paris VII - CEA, DAM, DIF
Maintainer Loic Le Gratiet <loic.legratiet@gmail.com>
Description This package builds multi-fidelity cokriging models from responses with different levels of fidelity. Important functions: MuFicokm, predict.MuFicokm, summary.MuFicokm.
Depends DiceKriging, methods
Suggests rgenoud
License GPL-3
URL http://www.redice-project.org
Repository CRAN
Date/Publication 2012-12-14 10:33:00
NeedsCompilation no

R topics documented:

MuFiCokriging-package .............................................. 2
CrossValidationMuFicokm .......................................... 3
CrossValidationMuFicokmAll ..................................... 4
ExtractNestDesign ................................................. 6
kmCok ................................................................. 7
kmCok-class ........................................................ 9
MuFicokm .......................................................... 11
NestedDesign ......................................................... 16
NestedDesignBuild ................................................ 18
predict.kmCok ....................................................... 19
predict.MuFicokm .................................................. 20
MuFiCokrigin-package

Multi-Fidelity Cokriging methods for computer experiments

Description

Create multi-fidelity cokriging models using data from codes with multiple levels of fidelity.

Details

Package: MuFiCokrigin
Type: Package
Version: 1.0
Date: 2012-12-21
License: GPL-3

Important functions:

MuFicokm:
Creation of a multi-fidelity cokriging model with unknown or known parameters.

predict.MuFicokm:
Prediction of a multi-fidelity cokriging model at new points (Simple and Universal cokriging).

Author(s)

Loïc Le Gratiet
Maintainer: <loic.legratiet@gmail.com> - Loïc Le Gratiet
Affiliations:
Universite Paris VII Denis-Diderot 75205 Paris Cedex 13
CEA, DAM, DIF, F-91297 Arpajon, France

References

KENNEDY, M.C. & O’HAGAN, A. (2000), Predicting the output from a complex computer code when fast approximations are available. Biometrika 87, 1-13
Cross Validation Procedure for Multi-Fidelity Cokriging models when the observations are removed from the code with the highest level of fidelity

Description

Provide the predictive errors and variances of a cross validation procedure when observations (not necessarily one) are removed only from the code with the highest level of fidelity.

Usage

CrossValidationMuFicokm(model, indice)

Arguments

model: an object of class S3 ("MuFicokm") provided by the function "MuFicokm" corresponding to the multi-fidelity cokriging model.
indice: a vector containing the indices of the observations removed from the highest code level for the cross-validation procedure.

Value

CVar: a vector containing the predictive errors of the cross-validation procedure.
CVar: a vector containing the predictive variances of the cross-validation procedure.
CVCov: a matrix representing the predictive covariance matrix of the cross-validation procedure.

Author(s)

Loic Le Gratiet

References

DUBRULE, O. (1983), Cross Validation in a Unique Neighborhood. Mathematical Geology 15. Mo.6
ZHANG, H. and WANG, Y. (2009), Kriging and cross-validation for massive spatial data. Environmetrics 21, 290-304.
Cross Validation Procedure for Multi-Fidelity Cokriging models when the observations are removed from all code levels.

Description

Provide the predictive errors and variances of the cross validation procedure when observations are removed from all code levels.

Usage

CrossValidationMuFicokmAll(model, indice)
Arguments

model  an object of class S3 ("MuFicokm") provided by the function "MuFicokm" corresponding to the multi-fidelity cokriging model.

indice  a vector containing the indices of the observations removed from the highest code level for the cross-validation procedure.

Details

This function performs all the possible cross-validation procedures. Indeed, due to the nested property of the experimental design sets, we can choose to remove observations only from the highest code level or the two highest code levels and so on.

Value

CVerr  a list of vectors indexed by q containing the predictive errors of the cross-validation procedure when the observations are removed from the q highest code levels.

CVvar  a list of vectors indexed by q containing the predictive variances of the cross-validation procedure when the observations are removed from the q highest code levels.

CVCov  a list indexed by q of the predictive covariance matrices of the cross-validation procedure when the observations are removed from the q highest code levels.

CVerrall  a vector containing the predictive errors of the cross-validation procedure when the observations are removed from all code levels.

CVvarall  a vector containing the predictive variances of the cross-validation procedure when the observations are removed from all code levels.

CVCovall  the predictive covariance matrix of the cross-validation procedure when the observations are removed from all code levels.

Author(s)

Loic Le Gratiet

References


See Also

MuFicokm, CrossValidationMuFicokm

Examples

```r
#--- test functions (see [LE GRATIET, L. 2012])
Funcf <- function(x) {return(0.5*(6*x-2)^2*sin(12*x-4)+sin(10*cos(5*x)))}
Funccc <- function(x) {return((6*x-2)^2*sin(12*x-4)+10*(x-0.5)-5)}
#--- Data
```
```r
Dc <- seq(0,1,0.1)
indDF <- c(1,3,7,11)
DNest <- NestedDesign(Dc, nlevel=2, indices = list(indDF))
zc <- FuncC(DNest$SPK)
zf <- FuncC(ExtractNestDesign(DNest, 2))

#--- Model creation with parameter estimations
mymodel <- Muficokm(
  formula = list(~1, ~1+X1),
  MuFidesign = DNest,
  response = list(zc, zf),
  nlevel = 2,
  covtype = "matern5_2"
)

#--- Cross Validation
indice <- c(1,3)
CVall <- CrossValidationMuficokmAll(mymodel, indice)

#-- predictive errors when we remove the observations from Funcf and Funccc
CVall$CVerrall
#-- predictive variances when we remove the observations from Funcf and Funccc
CVall$CVvarall
#-- predictive covariance matrix when we remove the observations from Funcf and Funccc
CVall$CVcovall
#-- predictive errors when we remove the observations from Funcf
CVall$CVerr[[1]]
#-- predictive variances when we remove the observations from Funcf
CVall$CVvar[[1]]
#-- predictive covariance matrix when we remove the observations from Funcf
CVall$CVcov[[1]]

#--- Leave-One-Out Cross Validation
#-- LOO CV predictive errors
apply(matrix(1:DNest$n), 1, function(x) CrossValidationMuficokmAll(mymodel, x)$CVerrall)
```

---

| ExtractNestDesign | Extraction of the experimental design set of a level from an object of class ("NestedDesign") |
---|---|

**Description**

Extract the experimental design set of level number i from an object of class ("NestedDesign") representing a nested experimental design set.

**Usage**

```r
ExtractNestDesign(NestDes, level)
```

**Arguments**

- **NestDes** an object of class ("NestedDesign").
- **level** an integer indicating the level from which we want to extract the experimental design set.
kmCok

Description

An internal function used to build kriging models included in the multi-fidelity cokriging models.

Usage

```
kmCok( formula = ~1, design, response, formula.rho = ~1, Z = NULL,
covtype = "matern5_2", coef.trend = NULL, coef.cov = NULL,
coef.var = NULL, nugget = NULL, nugget.estim = FALSE,
noise.var = NULL, estim.method="MLE", penalty = NULL,
optim.method = "BFGS", lower = NULL, upper = NULL, parinit = NULL,
control = NULL, gr = TRUE, iso = FALSE, scaling = FALSE, knots = NULL)
```
Arguments

formula.rho an object of class ("formula") specifying the linear trends of the adjustment coefficients. This formula should concern only the input variables, and not the output (response). If there is any, it is automatically dropped. The default is ~1, which defines a constant trend.

Z a vector (or 1-column matrix or data frame) containing the values of the 1-dimensional output given by the function of level k at the design points of level k-1.

formula see km
design see km
response see km
covtype see km
coeff.trend see km
coeff.cov see km
coeff.var see km
estim.method see km
nugget see km
nugget.estim see km
noise.var see km
penalty see km
optim.method see km
lower see km
upper see km
parinit see km
control see km
gr see km
iso see km
scaling see km
knots see km

Value

An object with S4 class "kmCok" (see kmCok-class).

Author(s)

Olivier Roustant, David Ginsbourger, Ecole des Mines de St-Etienne.
Loic Le Gratiet, Universite Paris VII Denis-Diderot
References

KRIGE, D.G. (1951), A statistical approach to some basic mine valuation problems on the witwatersrand, *J. of the Chem., Metal. and Mining Soc. of South Africa*, 52 no. 6, 119-139.


See Also

predict,kmCok-method

kmCok-class

Class "kmCok"

Description

S4 class for cokriging models derived from km-class.

Objects from the Class

Objects can be created by calls of the form `new("kmCok", ...)`.

Slots

AR.p: object of class "integer". The number of regressors for the adjustment coefficient $\rho_{k-1}$ with $k = 2, \ldots, nlevel$. (see "MuFicokm")

AR.formula: object of class "formula". A formula specifying the trend as a linear model for the adjustment coefficient $\rho_{k-1}$ with $k = 2, \ldots, nlevel$. (see "MuFicokm")

AR.Z: object of class "numeric". The vector of response values of level $k - 1$ at design points of level $k$. (see "MuFicokm")

AR.F: object of class "matrix". The experimental matrix corresponding to the evaluation of the linear trend basis functions of $\rho_{k-1}$ at the design of experiments. (see "MuFicokm")

d: object of class "integer" see Class "km" for the other arguments

n: object of class "integer"

X: object of class "matrix"

y: object of class "matrix"
Methods

predict signature(object = "kmCok"): see predict.kmCok-method

Author(s)

Loic Le Gratiet

See Also

km, MuFicokm
_creation of Multi-Fidelity Cokriging models

**Description**

Create multi-fidelity cokriging models when parameters are unknown or known. If parameters are unknown, they are estimated by Maximum Likelihood. "MuFicokm" is based on the function "km" of the package "DiceKriging" and its usage is similar.

**Usage**

```r
MuFicokm( formula, MuFidesign, response, nlevel, formula.rho = ~1, covtype = "matern5_2", coef.trend = NULL, coef.rho = NULL, coef.cov = NULL, coef.var = NULL, nugget = NULL, nugget.estim = FALSE, noise.var = NULL, estim.method = "MLE", penalty = NULL, optim.method = "BFGS", lower = NULL, upper = NULL, parinit = NULL, control = NULL, gr = TRUE, iso = FALSE, scaling = FALSE, knots = NULL)
```

**Arguments**

- **formula**: a list of objects of class "formula" specifying the linear trends of the Gaussian processes $\delta_k(x)$ with $k = 1, \ldots, nlevel$. We use the convention $Z_1(x) = \delta_1(x)$. The length of the list has to be equal to the number of levels. This formula should concern only the input variables, and not the output (response).

- **MuFidesign**: an object of class "MuFiDesign" (see NestedDesign) representing the nested experimental design sets for the different code levels.

- **response**: a list of vectors (or 1-column matrix or data frame) containing the values of the 1-dimensional outputs given by the different code levels. The length of the list has to be equal to the number of levels.

- **nlevel**: the number of levels for the responses.

- **formula.rho**: a list of objects of class "formula" specifying the linear trends of the adjustment coefficients (i.e. it corresponds to $g_{k-1}(x)$, see details). The default is ~1, which defines a constant trend. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

- **covtype**: an optional list of character strings specifying the covariance structure to be used for the Gaussian processes $\delta_k(x)$ with $k = 1, \ldots, nlevel$. We use the convention $Z_1(x) = \delta_1(x))$, to be chosen between "gauss", "matern5_2", "matern3_2", "exp" or "powexp". See a full description of available covariance kernels in covTensorProduct-class in package DiceKriging. Default is "matern5_2". The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

- **coef.trend**: an optional list of vectors containing the values for the trend of the Gaussian processes $\delta_k(x)$ with $k = 1, \ldots, nlevel$. We use the convention $Z_1(x) = \delta_1(x)$. (see below and details)
coef.rho an optional list of vectors containing the values of $\gamma_{k-1}$ for the adjustment coefficients $\rho_{k-1}$ with $k = 2, \ldots, \text{nlevel}$. (see below and details)

coef.cov an optional list of vectors containing the values for the covariance parameters of the Gaussian processes $\delta_k(x)$ with $k = 1, \ldots, \text{nlevel}$. We use the convention $Z_1(x) = \delta_1(x)$. (see below and details)

coef.var an optional list of vectors containing the values for the variance parameters of the Gaussian processes $\delta_k(x)$ with $k = 1, \ldots, \text{nlevel}$. We use the convention $Z_1(x) = \delta_1(x)$ (see details). The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels. For estimation, there are 4 cases: 1. (All unknown) If all are missing, all are estimated. 2. (All known) If all are provided, no estimation is performed. 3. (Known trend) If coef.trend and coef.rho is provided but at least one of coef.cov or coef.var is missing, then BOTH coef.cov and coef.var are estimated. 4. (Unknown trend) If coef.cov and coef.var are provided but coef.trend and coef.rho are missing, then coef.trend and coef.rho are estimated.

nugget an optional list of variance values standing for the homogeneous nugget effect for the Gaussian processes modelling the biases. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

nugget.estim an optional list of booleans indicating whether the nugget effect should be estimated. Note that this option does not concern the case of heterogeneous noisy observations (see noise.var below). If nugget is given, it is used as an initial value. Default is FALSE. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

noise.var for noisy observations: an optional list of vectors containing the noise variance at each observation for each level. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

estim.method a list of character strings specifying the method by which unknown parameters are estimated. Default is "MLE" (Maximum Likelihood). At this stage, a beta version of leave-One-Out estimation (estim.method="LOO") is also implemented for noise-free observations. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

penalty (beta version) an optional list of lists suitable for Penalized Maximum Likelihood Estimation. The list must contain the item fun indicating the penalty function, and the item value equal to the value of the penalty parameter. At this stage the only available fun is "SCAD", and covtype must be "gauss". Default is NULL, corresponding to (un-penalized) Maximum Likelihood Estimation. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

optim.method an optional list of character strings indicating which optimization method is chosen for the likelihood maximization for each level. "BFGS" is the optim quasi-Newton procedure of package stats, with the method "L-BFGS-B". "gen" is the genoud genetic algorithm (using derivatives) from package rgenoud (>= 5.3.3).
The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

lower
(see below)
upper
optional list of vectors containing the bounds of the correlation parameters for optimization for each level. The default values are given by covparametersbounds. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

parinit
a list of n optional vectors containing the initial values for the variables to be optimized over for each level. If no vector is given, an initial point is generated as follows. For method "gen", the initial point is generated uniformly inside the hyper-rectangle domain defined by lower and upper. For method "BFGS", some points (see control below) are generated uniformly in the domain. Then the best point with respect to the likelihood (or penalized likelihood, see penalty) criterion is chosen. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

control
an optional list of lists of control parameters for optimization for each level. To avoid printing information in the command line during optimization progress, indicate trace=FALSE. For method "BFGS", pop.size is the number of candidate initial points generated before optimization starts (see parinit above). Default is 20. For method "gen", one can control pop.size (default : min(20, 4+3*log(nb of variables)), max.generations (5), wait.generations (2) and BFGSburnin (0) of function genoud (see "genoud"). Another tunable parameter is upper.alpha (1e-8) for nugget estimation (see "km1Nugget"). Numbers into brackets are the default values. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

gr
an optional list of booleans indicating whether the analytical gradient should be used. Default is TRUE. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

iso
an optional list of booleans that can be used to force a tensor-product covariance structure (see covTensorProduct-class) to have a range parameter common to all dimensions. Default is FALSE. Not used (at this stage) for the power-exponential type. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

scaling
an optional list of booleans indicating whether a scaling on the covariance structure should be used. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

knots
an optional list of lists of knots for scaling. The j-th element is a vector containing the knots for dimension j. If scaling=TRUE and knots are not specified, then knots are fixed to 0 and 1 in each dimension (which corresponds to affine scaling for the domain [0,1]d). The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

Details

The assumed model is the one presented in the paper [LE GRATIET, L. (2012)]. Let us denote by \( Z_k(x) \) the Gaussian process modelling the code level \( k \). We consider the following autoregressive
model:

\[ Z_k(x) = \rho_{k-1} Z_{k-1}(x) + \delta_k(x) \]

where \( \rho_{k-1} \) is the adjustment coefficient between levels \( k \) and \( k - 1 \) and \( \delta_k(x) \) models the bias between the level \( k \) and the level \( k - 1 \) adjusted. When \( \rho_{k-1} \) depends on \( x \), it has the following form:

\[ \rho_{k-1}(x) = g^T_{k-1}(x) \gamma_{k-1} \]

where the regressors \( g_{k-1}(x) \) are defined thanks to the parameter \( \text{formula.rho} \). Furthermore, the experimental design sets \( D_k \) for each level \( k = 2, \ldots, nlevel \) must be nested.

\[ D_k \subset D_{k-1} \]

**Value**

An object of class S3 MuFicokm.

- **cok**: a list containing objects of class S4 ("km") and ("kmCok").
- **zd**: a list containing the responses of the conditionned Gaussian processes of level \( k = 2, \ldots, nlevel \) at the experimental design set of level \( k - 1 \).
- **response**: a list of vectors containing the known responses at each code level.
- **nlevel**: a numeric representing the number of code levels.
- **Dnest**: an object of class ("NestDesign") representing the nested experimental design sets.
- **nuggets**: a list of numeric representing the nugget effects used to regularize the covariance matrices at each level.

**Author(s)**

Loic Le Gratiet, Universite Paris VII Denis-Diderot - CEA, DAM, DIF

**References**


**See Also**

`predict.MuFicokm`, `summary.MuFicokm`, `NestedDesign`, `CrossValidationMuFicokmAll`
Examples

#--- 3 Dimensional test with 3 levels of response
#- test functions
myfunc1 <- function(x){sin(2*pi*x[1])*0.2*(x[2]+2)^2+cos(4*pi*x[3])^2}
myfunc2 <- function(x){2*myfunc1(x)+x[3]}
myfunc3 <- function(x){3*myfunc2(x)+x[2]+x[1]}
#- Data
#- Nested Experimental design sets
nD1 <- 100
nD2 <- 50
nD3 <- 20
set.seed(1); D1 <- matrix(runif(n=nD1*3, 0,1),ncol=3)
set.seed(2); D2 <- matrix(runif(n=nD2*3, 0,1),ncol=3)
set.seed(3); D3 <- matrix(runif(n=nD3*3, 0,1),ncol=3)
NestDesign <- NestedDesignBuild(design = list(D1,D2,D3))
#- observations
z1 <- myfunc1(NestDesign$PX)
z2 <- myfunc2(ExtractNestDesign(NestDesign,2))
z3 <- myfunc3(ExtractNestDesign(NestDesign,3))
#- Multi-fidelity cokriging creation
mymodel <- MuFicokm(formula = list(~x1~x2~x3),
                     MuFidesign = NestDesign,
                     response = list(z1,z2,z3),
nlevel = 3)
#- Multi-fidelity cokriging prediction
newdata <- matrix(runif(333,0,1),ncol=3)
predictions <- predict(mymodel, newdata, "UK")
z.pred <- predictions$mean
#- Multi-fidelity cokriging cross Validation
set.seed(1); indice <- sample(1:nD3)[1:10]
#- Observations removing from the highest level
resCV.cok <- CrossValidationMuFicokm(mymodel,indice)
#- Observations removing from all levels
resCV.cok.all <- CrossValidationMuFicokmA(mymodel,indice)
#- Multi-fidelity cokriging summary
sum <- summary(mymodel)

#--- 1 Dimensional test with 2 levels of response
#- test functions
Funcf <- function(x){return((6*x-2)^2*sin(12*x-4))}
Funccc <- function(x){return(0.5*Funcf(x)+10*(x-0.5)-5)}
#- Data
Dc <- seq(0.1,0.1)
indDF <- c(1,3,7,11)
DNest <- NestedDesign(Dc, nlevel=2, indices = list(indDF) )
zC <- Func(newest(Dnest$PX))
Df <- ExtractNestDesign(Dnest,2)
zF <- Func(Df)
#--- Multi-fidelity cokriging creation without parameter estimations
mymodel <- MuFicokm(  
  formula = list(~1,-1),
  MuFidesign = Dnest,
  response = list(zC,zF),
  nlevel = 2,
  covtype = list("gauss","matern5_2"),
  coef.trend=list(-5,3),
  coef.rho=list(2),
  coef.var=list(2,2),
  coef.cov=list(0.1,0.2))
predictions <- predict(  
  object = mymodel,
  newdata = seq(0,1,le=100),
  type = "SK")
#--- Multi-fidelity cokriging creation with parameter estimations
mymodel <- MuFicokm(  
  formula = list(~1,-1+X1),
  MuFidesign = Dnest,
  response = list(zC,zF),
  nlevel = 2,
  covtype = "matern5_2")
predictions <- predict(  
  object = mymodel,
  newdata = seq(0,1,le=100),
  type="UK")

---

**NestedDesign**

**Definition of nested experimental design sets for Multi-Fidelity Cokriging models**

**Description**

Build an object of class S3 ("NestDesign") defining nested experimental design sets used to build multi-fidelity Cokriging models.

**Usage**

```r
NestedDesign(x, nlevel, indices = NULL, n = NULL)
```

**Arguments**

- `x` a matrix representing the experimental design set of the code level 1.
- `nlevel` an integer representing the number of code levels.
indices a list of vectors. The \( i \)th element of the list contains the indices of the points in the experimental design set of the level \( i - 1 \) constituting the experimental design set of the level \( i \). If \( \text{indices} = \text{NULL} \) they are randomly sampled according to the number of observations defining in \( n \).

\( n \) a vector containing the number of observations at level \( k = 2, \ldots, n_{\text{level}} \). It is not taking into account if \( \text{indices} \) is different from \( \text{NULL} \).

Details

The procedure does not change the experimental design set of the highest code level. During the procedure, the points of \( D_{k-1} \) the closest to those of \( D_k \) with \( k = 2, \ldots, n_{\text{level}} \) are removed and they are replaced by the points of \( D_k \). Thus, the length of the final \( D_{k-1} \) could be larger than the one of the initial \( D_{k-1} \). (see "MuFicokm")

Value

an object of class ("NestedDesign") representing a nested experimental design set.

Author(s)

Loic Le Gratiet

References


See Also

\texttt{MuFicokm}, \texttt{ExtractNestDesign}, \texttt{NestedDesignBuild}

Examples

```r
#-- Nested Experimental design sets
dimension <- 3
nD1 <- 100
nD2 <- 50
nD3 <- 20
set.seed(1); D1 <- matrix(runif(n=nD1*dimension, 0,1),ncol=dimension)
NestDesign <- NestedDesign(D1, nlevel=3, n = c(nD2,nD3))
```
NestedDesignBuild

Nested experimental design sets building

Description

Procedure to build nested experimental design sets from a list of non-nested experimental design sets.

Usage

NestedDesignBuild(design = NULL)

Arguments

design

a list of matrices representing the experimental design sets of each level.

Value

an object of class ("NestDesign") representing the nested experimental design sets.

Author(s)

Loic Le Gratiet

See Also

MuFicokm, NestedDesign, SubstDesign, CrossValidationMuFicokmAll

Examples

```r
#-- Nested Experimental design sets
dimension <- 3
dD1 <- 100
dD2 <- 50
dD3 <- 20
set.seed(1):D1 <- matrix(runif(n=D1*dimension, 0,1),ncol=dimension)
set.seed(2):D2 <- matrix(runif(n=D2*dimension, 0,1),ncol=dimension)
set.seed(3):D3 <- matrix(runif(n=D3*dimension, 0,1),ncol=dimension)
NestDesign <- NestedDesignBuild(design = list(D1,D2,D3))
#-- Design set at level 1
NestDesign$PX
#-- Extraction of design sets at level 2 and 3
ExtractNestDesign(NestDesign,2)
ExtractNestDesign(NestDesign,3)
```
predict.kmCok

Kriging predictions and confidence intervals used in Multi-Fidelity Cokriging models

Description

An internal function which provides predicted values and conditional variances based on a kmCok model. 95% confidence intervals are given based on Gaussian process assumption. This might be abusive in particular in the case where the number of observations is small.

Usage

```r
## S4 method for signature 'kmCok'
predict(object, newdata, newZ, type,
      se.compute = TRUE, cov.compute = FALSE, checkNames = FALSE, ...)
```

Arguments

- `newZ`: a vector giving the predictions of the level $k = 1, ..., nlevel - 1$ at points `newdata`.
- `object`:
- `newdata`:
- `type`:
- `se.compute`:
- `cov.compute`:
- `checkNames`:
- `...`

Details

- `see km`

Value

- `see km`

Warning

- `see km`

Author(s)

Olivier Roustant, David Ginsbourger, Ecole des Mines de St-Etienne.
Loic Le Gratiet, Universite Paris VII Denis-Diderot
**References**

KRIGE, D.G. (1951), A statistical approach to some basic mine valuation problems on the witwatersrand, *J. of the Chem., Metal. and Mining Soc. of South Africa*, 52 no. 6, 119-139.


**See Also**

kmCok, predict.MuFicokm

---

**predict.MuFicokm**  
*Predictions and confidence intervals of Multi-Kriging Cokriging models at new data*

**Description**

Provide predictive mean, variance and covariance of a multi-fidelity cokriging model. 95% confidence intervals are given based on Gaussian process assumption. This might be abusive in particular in the case where the number of observations is small.

**Usage**

```r
## S3 method for class 'MuFicokm'
predict(object, newdata, type, se.compute = TRUE, cov.compute = FALSE, checkNames = FALSE, ...)
```

**Arguments**

- **object**: an object of class S3 ("MuFicokm") provided by the function "MuFicokm" corresponding to the multi-fidelity cokriging model.
- **newdata**: a vector, matrix or data frame containing the points where to perform predictions.
- **type**: a list of character strings corresponding to the kriging family of the Gaussian processes $\delta_k(x)$ with $k = 1, ..., nlevel$ (we use the convention $\delta_1(x) = Z_1(x)$), to be chosen between simple kriging ("SK"), or universal kriging ("UK"). (see "MuFicokm")
predict.MuFicokm

se.compute a list of optional booleans for each level. If FALSE, only the kriging mean is computed. If TRUE, the kriging variance and confidence intervals are computed too. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

cov.compute a list of optional booleans for each level. If TRUE, the conditional covariance matrix is computed. The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

checkNames a list of optional booleans for each level. If TRUE, a consistency test is performed between the names of newdata and the names of the experimental design (contained in object@x). The length of the list must be either the number of levels or one. If the length is one, the same argument is repeated for all levels.

... no other argument for this method.

Value

mean multi-fidelity co-kriging predictive mean computed at newdata.
sig2 multi-fidelity co-kriging predictive variance computed at newdata.
C multi-fidelity co-kriging predictive conditional covariance matrix. Not computed if cov.compute=FALSE (default).
mux multi-fidelity co-kriging predictive means at each level.
varx multi-fidelity co-kriging predictive variances at each level.
CovMat multi-fidelity co-kriging predictive conditional covariance matrices at each level.

Author(s)

Loic Le Gratiet

References

KENNEDY, M.C. & O’HAGAN, A. (2000), Predicting the output from a complex computer code when fast approximations are available. *Biometrika* 87, 1-13


See Also

*MuFicokm, summary.MuFicokm*
Examples

```r
#--- test functions (see [Le GRATIET, L. 2012])
Funcf <- function(x){return(0.5*(6*x-2)^2*sin(12*x-4)+sin(10*cos(5*x)))}
FuncCc <- function(x){return((6*x-2)^2*sin(12*x-4)+10*(x-0.5)-5)}

#--- Data
Dc <- seq(0,1,0.1)
indDf <- c(1,3,7,11)
DNest <- NestedDesign(Dc, nlevel=2, indices = list(indDf))
zc <- FuncC(DNest$PX)
Df <- ExtractNestDesign(DNest,2)
zf <- Funcf(Df)

#--- Multi-fidelity cokriging creation without parameter estimations
#--- "SK" : Simple CoKriging, i.e. when parameters are known
#--- "UK" : Universal CoKriging, i.e. when parameters are estimated
#--- model creation
mymodelSK <- MuFicokm(formula=list(~1~-1),
MuFidesign = DNest, response=list(zc,zf),
nlevel = 2,
coef.trend=list(0,2),
coef.rho=list(0.5),
coef.var=list(2,2),
coef.cov=list(0.1,0.2))

#--- predictions with "SK"
predictionsSK <- predict(
object = mymodelSK,
newdata = seq(0,1,le=100),
type = "SK")

#--- Multi-fidelity co-kriging building with parameter estimations
#--- model creation
mymodelUK <- MuFicokm(formula=list(~1~-1),
MuFidesign = DNest, response=list(zc,zf),
nlevel = 2)

#--- predictions with "UK"
predictionsUK <- predict(
object = mymodelUK,
newdata = seq(0,1,le=100),
type = "UK")

#--- Multi-fidelity co-kriging building with known and unknown parameters
#--- model creation
mymodelSK_UK <- MuFicokm(formula=list(~1~-1),
MuFidesign = DNest, response=list(zc,zf),
nlevel = 2,
coef.trend=list(-5,0),
coef.rho=list(NULL),
coef.var=list(5,0)
)`
SubstDesign

Imbrication of two experimental design sets

Description

Procedure to nest two experimental design sets.

Usage

SubstDesign( PX2 = NULL, PX1 = NULL )

Arguments

PX2  a matrix representing the first experimental design set. This design set is not changed during the procedure.

PX1  a matrix representing the second experimental design set. This design set is changed in order to include PX2.
Details

The procedure does not change the experimental design set \( PX_2 \) which must have a number of points smaller than the one of \( PX_1 \). During the procedure, the points of \( PX_1 \) the closest to those of \( PX_2 \) are removed and they are replaced by the points of \( PX_2 \). Thus, the length of the final \( PX_1 \) could be larger than the one of the initial \( PX_1 \).

Value

- \( PX \) : a matrix representing the experimental design set \( PX_1 \) which contains \( PX_2 \).
- \( le \) : a numeric representing the number of points of \( PX_2 \).

Author(s)

Loic Le Gratiet

See Also

- MuFicokm, NestedDesignBuild, NestedDesign

Examples

```r
# dimension <- 2
nD1 <- 100
nD2 <- 50
set.seed(1);D1 <- matrix(runif(n=nD1*dimension, 0,1),ncol=dimension)
set.seed(2);D2 <- matrix(runif(n=nD2*dimension, 0,1),ncol=dimension)
subDes <- SubstDesign(PX2 = D2, PX1 = D1)

op <- par(mfrow=c(2,1))
plot(rbind(D1,D2),col=c(rep(1,nD1),rep(2,nD2)),
pch=c(rep(1,nD1),rep(2,nD2)),xlab="x1",ylab="x2")
plot(rbind(subDesPX,X,D2),col=c(rep(1,dim(subDes$PX)[1]),rep(2,nD2)),
pch=c(rep(1,dim(subDes$PX)[1]),rep(2,nD2)),xlab="x1",ylab="x2")
```

---

**summary.MuFicokm**  
*Function summary for Multi-Fidelity Cokriging models*

Description

Provide a summary of a multi-fidelity cokriging model. In particular, it provides the parameter estimations and the results of the cross-validation procedure.

Usage

```r
## S3 method for class 'MuFicokm'
summary(object, CrossValidation = FALSE, ...)
```
Arguments

object an object of class S3 ("MuFicokm") provided by the function MuFicokm corresponding to the multi-fidelity cokriging model.

CrossValidation

a Boolean. If TRUE, a Leave-One-Out cross validation procedure is performed. For the LOO procedure, the responses are removed from all code levels and the trend, adjustment and variance parameters are re-estimated after each removed observation.

... no other argument for this method.

Details

"summary.MuFicokm" return the parameter estimations for each level and the result of the Leave-One-Out Cross-Validation (RMSE=Root Mean Squared Error; Std RMSE=Standardized RMSE; Q2=explained variance).

Value

A list with following items (see "MuFicokm"):

CovNames a list of character strings giving the covariance structures used for the cokriging model. The element i of the list corresponds to the covariance structure of the Gaussian process \( \delta_i(x) \) with \( \delta_1(x) = Z_1(x) \). (see "MuFicokm")

Cov.val a list of vectors giving the values of the hyper-parameters of the cokriging model. The element i of the list corresponds to the hyper-parameters of the Gaussian process \( \delta_i(x) \) with \( \delta_1(x) = Z_1(x) \). (see "MuFicokm")

Var.val a list of numerics giving the values of the variance parameters of the cokriging model. The element i of the list corresponds to the variance of the Gaussian process \( \delta_i(x) \) with \( \delta_1(x) = Z_1(x) \). (see "MuFicokm")

Rho.val a list of vectors giving the values of the trends \( \gamma_i \) of the adjustment parameters \( \rho_i \) of the cokriging model. The element i of the list corresponds to the adjustment parameter between \( Z_i \) and \( \delta_i(x) \). (see "MuFicokm")

Trend.val a list of vectors giving the values of the trend parameters of the Gaussian processes \( \delta_i(x) \) and \( Z_1(x) \).

Author(s)

Loic Le Gratiet

Examples

#--- test functions (see [Le GRATIET, L. 2012])
Funcf <- function(x){return(0.5*(6*x-2)^2*sin(12*x)+sin(10*cos(5*x)))}
Funccc <- function(x){return((6*x-2)^2*sin(12*x-4)+10*(x-0.5)-5)}

#--- Data
Dc <- seq(0,1,0.1)
indDf <- c(1,3,7,11)
DNest <- NestedDesign(Dc, nlevel=2, indices = list(indDf) )
zc <- Func(DNest$PX)
Df <- ExtractNestDesign(DNest, 2)
zf <- Func(Df)

#--- Multi-fidelity cokriging creation without parameter estimations
mymodel <- MuFicokm(
  formula = list(~1, ~1),
  MuFidesign = DNest,
  response = list(zc, zf),
  nlevel = 2)

sum <- summary(object = mymodel, CrossValidation = TRUE)
names(sum)

#--- Saving parameters
#--covariance parameters
sum$Cov.Val
#--variance parameters
sum$Var.Val
#--trend parameters
sum$Trend.Val
#-- adjustment parameters
sum$Rho.Val
Index

CrossValidationMuFicokm, 3, 5
CrossValidationMuFicokmAll, 4, 4, 14, 18

ExtractNestDesign, 6, 17

km, 8–10, 19
kmCok, 7, 20
kmCok-class, 9

MuFicokm, 4, 5, 7, 10, 11, 17, 18, 21, 24
MuFiCokriging (MuFiCokriging-package), 2
MuFiCokriging-package, 2

NestedDesign, 7, 11, 14, 16, 18, 24
NestedDesignBuild, 17, 18, 24

predict (predict.kmCok), 19
predict, kmCok-method (predict.kmCok), 19
predict.kmCok, 19
predict.MuFicokm, 14, 20, 20

SubstDesign, 18, 23
summary.MuFicokm, 14, 21, 24