Package ‘fanovaGraph’

October 13, 2022

Type Package
Title Building Kriging Models from FANOVA Graphs
Version 1.5
Date 2020-10-01
Author Jana Fruth, Thomas Muehlenstaedt, Olivier Roustant, Malte Jastrow, Sonja Kuhnt
Maintainer Sonja Kuhnt <sonja.kuhnt@fh-dortmund.de>
Description Estimation and plotting of a function's FANOVA graph to identify the interaction structure and fitting, prediction and simulation of a Kriging model modified by the identified structure. The interactive function plotManipulate() can only run on the 'RStudio IDE' with 'RStudio' package 'manipulate' loaded. 'RStudio' is freely available (<https://rstudio.com/>), and includes package 'manipulate'. The equivalent function plotTk() bases on CRAN Repository packages only. For further information on the method see Fruth, J., Roustant, O., Kuhnt, S. (2014) <doi:10.1016/j.jspi.2013.11.007>.
License GPL-3
LazyLoad yes
Depends sensitivity, igraph, DiceKriging (>= 1.4)
Suggests manipulate, testthat
NeedsCompilation no
Repository CRAN
Date/Publication 2020-10-07 12:10:02 UTC

R topics documented:

fanovaGraph-package ........................................ 2
estimateGraph ............................................. 3
i2Index .................................................... 5
kmAdditive ................................................ 6
kmPredictWrapper ........................................ 8
L ............................................................... 9
plot.graphlist ............................................. 9
fanovaGraph-package

plotDeltaJumps ................................................................. 11
plotGraphChange, plotTk, plotManipulate .................................. 12
predictAdditive .................................................................. 13
simAdditive ..................................................................... 15
threshold ......................................................................... 16
thresholdIdentification .......................................................... 18
totalIndex ......................................................................... 19

Index 21

fanovaGraph-package Building Kriging models with FANOVA graphs

Description

Estimates and plots the FANOVA graph of a function to identify its interaction structure and fits a kriging model modified by the identified structure

Details

Important functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>estimateGraph</td>
<td>Estimate indices for the graph, create graph structure</td>
</tr>
<tr>
<td>threshold</td>
<td>Set indices below a threshold to zero</td>
</tr>
<tr>
<td>plot.graphlist</td>
<td>Plot a given graph structure</td>
</tr>
<tr>
<td>plotDeltaJumps</td>
<td>Provide plots for the choice of the threshold</td>
</tr>
<tr>
<td>kmAdditive</td>
<td>Kriging model estimation with block-additive kernel</td>
</tr>
<tr>
<td>predictAdditive</td>
<td>Prediction function from Kriging model with block-additive kernel</td>
</tr>
<tr>
<td>simAdditive</td>
<td>Simulation from Kriging model with block-additive kernel</td>
</tr>
</tbody>
</table>

Author(s)

J. Fruth, T. Muehlenstaedt, O. Roustant, M. Jastrow

References


**See Also**

`DiceKriging`, `sensitivity.igraph`

**Examples**

```r
#demo(ExampleIshigami)
#demo(Example6D)
#demo(Estimation)
#demo(Threshold)
```

---

**estimateGraph**

*FANOVA graph estimation.*

**Description**

Estimates the structure of the FANOV A graph by estimating the total interaction indices for the graph edges (a particular case of superset importance introduced by Liu and Owen, 2006), the main effect indices for the graph vertices and the overall variance for normalizing the indices and finding the clique structure of the estimated graph.

**Usage**

```r
estimateGraph(f.mat, d, q = NULL, q.arg = NULL, n.tot = NULL, method = "LiuOwen", n.lo = NULL, n.mc = NULL, n.fast = 500, L = NULL, M = 6, n.pf = NULL, n.main = 1000, confint = TRUE, print.loop.index = FALSE, ...)
```

**Arguments**

- `f.mat` vectorized function for which the FANOV A graph shall be estimated
- `d` integer, number of input factors (vertices)
- `q` a vector of character strings of quantile functions corresponding to the factors distributions, it can be a single character string meaning same distribution for all, if not specified "qunif" is taken
- `q.arg` a list of lists of quantile functions parameters of the distributions in q, it can be a single list meaning same parameters for all, if not specified the default values of the respective distributions are taken
- `n.tot` optional integer, total number of function evaluations, instead of n.tot method related parameters (n.lo, n.mc, L or n.sobol) can be provided
- `method` character string specifying the estimation method of the total interaction indices, to be chosen between "LiuOwen", "FixFast", "RBD" and "PickFreeze", defaults to "LiuOwen", see references for further details
estimateGraph

n.lo optional integer, only if method="LiuOwen", number of Monte Carlo simulations in method of Liu and Owen
n.mc optional integer, only if method="FixFast", number of Monte Carlo simulations for the expectation in fixing method using FAST
n.fast optional integer, only if method="FixFast", number of design points for FAST algorithm, defaults to 500
L optional integer, only if method="RBD", parameter L in RBD-FAST method
M optional integer, only if method="RBD", parameter M in RBD-FAST method
n.pf optional integer, only if method="PickFreeze", number of Monte Carlo simulations in pick-and-freeze method
n.main integer, number of Monte Carlo Simulations for computing main effect indices
confint optional Boolean, if TRUE, standard error and 95% confidence intervals of the indices are computed additionally for method="LiuOwen", defaults to TRUE
print.loop.index optional Boolean, if TRUE, loop indices are printed

... additional arguments to be passed to the function f.mat

Value

an object of class graphlist containing the graph structure which includes

d number of input factors
tii matrix containing the unscaled total interaction indices and if confint = TRUE their standard error and lower and upper confidence limits
i1 matrix containing the unscaled main effect indices
V overall variance
tii.scaled matrix containing the scaled total interaction indices
cliques list of cliques

Author(s)

J. Fruth, T. Muehlenstaedt

References


### Examples

```r
# Ishigami function, true analytical values: D12 = D23 = 0, D13 =~ 3.374
q.arg = list(list(min=-pi, max=pi), list(min=-pi, max=pi), list(min=-pi, max=pi))
estimateGraph(f.mat=ishigami.fun, d=3, q.arg=q.arg, n.tot=10000, method="LiuOwen")
estimateGraph(f.mat=ishigami.fun, d=3, q.arg=q.arg, n.tot=10000, method="FixFast")
estimateGraph(f.mat=ishigami.fun, d=3, q.arg=q.arg, n.tot=10000, method="RBD")
estimateGraph(f.mat=ishigami.fun, d=3, q.arg=q.arg, n.tot=10000, method="PickFreeze")
```

---

### i2Index

**Estimation of pure second order indices**

### Description

Estimation of the unscaled pure second order Sobol indices.

### Usage

```r
i2Index(f.mat, d, q = NULL, q.arg = NULL, n.i2, ...)
```

### Arguments

- **f.mat**: vectorized function of which indices shall be estimated
- **d**: integer, number of input factors (vertices)
- **q**: a vector of character strings of quantile functions corresponding to the factors distributions, it can be a single character string meaning same distribution for all, if not specified "qunif" is taken
- **q.arg**: a list of lists of quantile functions parameters of the distributions in q, it can be a single list meaning same parameters for all, if not specified the default values of the respective distributions are taken
- **n.i2**: number of Monte Carlo evaluations
- **...**: additional arguments to be passed to the function f.mat

### Value

A vector containing the unscaled pure second order indices

### Author(s)

J. Fruth
kmAdditive

Constrained MLE Optimization

Description
Constrained MLE optimization for kernels defined by cliques using constrOptim

Usage
kmAdditive(x, y, n.initial.tries = 50, limits = NULL, eps.R = 1e-08, cl, covtype = "gauss", eps.Var = 1e-06, max.it = 1000, iso = FALSE)

Arguments
- x: a design matrix of input variables, number of columns should be number of variables
- y: a vector of output variables of the same length as the columns of x
- n.initial.tries: number of random initial parameters for optimization, defaults to 50
- limits: a list with items lower, upper containing boundaries for the covariance parameter vector theta, if NULL suitable bounds are computed from the range of x
- eps.R: small positive number indicating the nugget effect added to the covariance matrix diagonal, defaults to eps.R = 1e-08
- cl: list of cliques, can be obtained by function threshold
- covtype: an optional character string specifying the covariance structure to be used, to be chosen between "gauss", "matern5_2", "matern3_2", "exp" or "powexp" (see DiceKriging), defaults to "gauss"
- eps.Var: small positive number providing the limits for the alpha parameters in order to guarantee strict inequalities (0+eps.Var <= alpha <= 1-esp.Var), defaults to eps.Var = 1e-06
- max.it: maximum number of iterations for optimization, defaults to max.it=1000
- iso: boolean vector indicating for each clique if it is isotropic (TRUE) or anisotropic (FALSE), defaults to iso = FALSE (all cliques anisotropic)

Examples
i2Index(f.mat=ishigami.fun, d=3, q.arg=list(min=-pi,max=pi), n.i2=10000)

See Also
estimateGraph totalIndex

References
kmAdditive

Value
list of estimated parameter 'alpha' and 'theta' corresponding to the clique structure in 'cl'

Author(s)
T. Muehlenstadt, O. Roustant, J. Fruth

References

See Also
predictAdditive

Examples
### example for ishigami function with cliques {1,3} and {2}
d <- 3
x <- matrix(runif(100*d,-pi,pi),nc=d)
y <- ishigami.fun(x)

c1 <- list(c(2), c(1,3))

# constrained ML optimation with kernel defined by the cliques
parameter <- kmAdditive(x, y, cl = c1)

# prediction with the new model
xpred <- matrix(runif(500 * d,-pi,pi), ncol = d)
ypred <- predictAdditive(xpred, x, y, parameter, cl=c1)
yexact <- ishigami.fun(xpred)

# rmse
sqrt(mean((ypred[,1]- yexact)^2))

# scatterplot
par(mfrow=c(1,1))
plot(yexact, ypred[,1], asp = 1)
abline(0, 1)

### compare to one single clique {1,2,3}
c1 <- list(c(1,2,3))

# constrained ML optimation with kernel defined by the cliques
parameter <- kmAdditive(x, y, cl = c1)

# prediction with the new model
ypred <- predictAdditive(xpred, x, y, parameter, cl=c1)

# rmse
sqrt(mean((ypred$mean- yexact)^2))
# scatterplot
par(mfrow=c(1,1))
plot(yexact, ypred$mean, asp = 1)
abline(0, 1)

### isotropic cliques

c1 <- list(c(2), c(1, 3))
parameter <- kmAdditive(x, y, cl = c1, iso=c(FALSE, TRUE))
ypred <- predictAdditive(xpred, x, y, parameter, cl=c1, iso=c(FALSE, TRUE))
sqrt(mean((ypred$mean - yexact)^2))

# the same since first clique has length 1
parameter <- kmAdditive(x, y, cl = c1, iso=c(TRUE, TRUE))
ypred <- predictAdditive(xpred, x, y, parameter, cl=c1, iso=c(TRUE, TRUE))
sqrt(mean((ypred$mean - yexact)^2))

---

kmPredictWrapper  
Wrapper for the Kriging model prediction

Description

Wrapper for the Kriging model prediction function `predict.km` from package `DiceKriging` to simplify the use of Kriging prediction functions as arguments for functions like `estimateGraph` or `fast99`.

Usage

```
kmPredictWrapper(newdata, km.object)
```

Arguments

- `newdata`: a vector, matrix or data frame containing the points where to perform predictions
- `km.object`: an object of class `km`

Value

kriging mean computed at `newdata`

Author(s)

J. Fruth, O. Roustant

See Also

- `estimateGraph`
Examples

```r
## graph estimation of a kriging prediction of the ishigami function
set.seed(1)
x <- matrix(runif(150,-pi,pi),100,3)
y <- ishigami.fun(x)
KM <- km(~1, design = data.frame(x), response = y)

g <- estimateGraph(f.mat = kmPredictWrapper, d = 3, n.tot = 10000, q.arg =
list(min = -pi, max = pi), method = "LiuOwen", km.object = KM)
print(g$tii)
```

---

### L

#### LHS Dataset

**Description**

6-dimensional Latin Hypercube Sampling Dataset

**Usage**

```r
data(L)
```

**Format**

The format is: num [1:100, 1:6] -0.7105 -0.7739 -0.5017 0.6158 0.0245 ...  

**Examples**

```r
data(L)
## str(L) ; pairs(L) ... 
```

---

### plot.graphlist

**Plot Graph via Package igraph**

**Description**

Plot FANOVA graphs using functions from package igraph.

**Usage**

```r
## S3 method for class 'graphlist'
plot(x, names = NULL, i2 = NULL, layout = NULL, plot.i1=TRUE, max.thickness=15, 
circle.diameter=40, ...) 
```
Arguments

- **x**: an object of class `graphlist` as obtained from `estimateGraph`
- **names**: optional character string, names of vertices, defaults to 1:d
- **i2**: optional vector of second order interaction indices (thickness of inner edges)
- **plot.i1**: optional boolean, if TRUE main effects are drawn in the graph by vertices thicknesses, should be set to FALSE when only total interaction indices are of interest
- **layout**: optional layout for the graph as in `igraph`, default is `layout.fruchterman.reingold`
- **max.thickness**: optional value for the maximal line thickness, defaults to 20
- **circle.diameter**: optional value for the circle diameter, defaults to 40
- **...**: additional arguments, passed to `plot`

Author(s)

J. Fruth, O. Roustant, S. Hess, S. Neumaerker

References


See Also

`plotGraphChange`

Examples

```r
op <- par(no.readonly=TRUE)
g1 <- estimateGraph(f.mat=ishigami.fun, d=3, q.arg=list(min=-pi,max=pi), n.tot=10000)
plot(g1)
plot(g1, names=c("A","B","C"))
plot(g1, names=c("A","B","C"), plot.i1 = FALSE)

# include pure second order indices
g2 <- estimateGraph(f.mat=function(x) x[,1]*x[,2]*x[,3]*x[,2]*x[,3], d=3,
  q.arg=list(min=-1,max=1), n.tot=10000)
plot(g2)
plot(g2, i2 = c(0.001, 0.001, 0.05))

# equal layouts and different edge thicknesses and circle diameters
g3 <- estimateGraph(f.mat=function(x) x[,1]*x[,2]*x[,3]*x[,4]*x[,5], d=5,
  q.arg=list(min=-1,max=1), n.tot=10000)
g4 <- estimateGraph(f.mat=function(x) x[,1]*x[,2]*x[,3]*x[,4]*x[,5], d=5,
  q.arg=list(min=-1,max=1), n.tot=10000)
```
plotDeltaJumps <- graph.full(n = 5, directed = FALSE)
layout <- layout.circle(graphClassIgraph)

plot(g3, max.thickness= 10, circle.diameter= 30, layout=layout)
plot(g4, max.thickness= 30, circle.diameter= 50, layout=layout)

##

#### Delta Jump Plot

**Description**

Threshold discission plot. plotDeltaJumps plots the threshold steps (the values of delta at which the graph changes) equidistant against the number of cliques and the values of delta on the real axis. The indices are assumed to be scaled for the threshold cuts.

**Usage**

```r
plotDeltaJumps(graphlist, interval = c(0, 1), mean.clique.size = FALSE)
```

**Arguments**

- `graphlist`: an object of class `graphlist` as obtained from `estimateGraph`
- `interval`: an optional vector of size 2, range for the values of delta to be shown in the plot, defaults to c(0,1)
- `mean.clique.size`: logical, if TRUE (default) an additional line is drawn representing the mean of the number of vertices in the cliques

**Details**

The plots shall give help in the choice for the threshold. In the first plot a small number of cliques might be preferable in order to have less parameters to estimate. If several values result in the same number of cliques the ones with higher mean clique size are possibly preferable.

In the second plot a high jump indicates a point of big distance between two successive edge indices and thus a clear change in the graph structure. The intervals with notable jumps are highlighted in green, the higher the jump the darker the colour. Those highlighted intervals together with a small number of cliques are probably good choices for the threshold.

Use `plotGraphChange` to visualize the graph structure for possible threshold values.

**Author(s)**

J. Fruth, O. Roustant

**See Also**

`thresholdIdentification`, `plotGraphChange`
Examples

tii <- matrix(c(0.0018, 0.0265, 0.0017, 0.0277, 0.0018, 0.028, 0.0013, 0.0212, 0.002, 0.0372, 0.0024, 0.0022, 0.0157, 0.003))
g <- list(d = 6,
        tii = tii,
        il = matrix(c(0.0901, 0.1288, 0.0683, 0.0979, 0.0882, 0.1572),
                    V = 0.8,
        tii.scaled = tii/0.8,
        cliques = list(1:6))

### Delta Jump Plot (jump between 0.0038 and 0.0196)
plotDeltaJumps(g)

Description

Graphs are plotted depending on a change on delta, the threshold for edges to appear in the graph, to enable a visual decision for delta by graph behavior.

Usage

plotGraphChange(graphlist, fix.layout = TRUE, delta.layout = 0.01)
plotTk(graphlist, delta.layout=0.01)
plotManipulate(graphlist, delta.layout=0.01)

Arguments

- `graphlist`: an object of class `graphlist` as obtained from `estimateGraph`
- `fix.layout`: logical, if TRUE (default) the position of the vertices is fixed for all plots such that the positions are optimal for delta = delta.layout
- `delta.layout`: optional value between 0 and 1, see fix.layout, defaults to 0.01

Note

plotGraphChange shows the changing of the graph step by step by changing plots as in demo, plotTk is an interactive version using tcltk, plotManipulate is an interactive version using manipulate

Author(s)

J. Fruth, O. Roustant
predictAdditive

See Also

plotDeltaJumps, plot.graphlist

Examples

# see demo(Threshold)

---

**predictAdditive**

**Prediction Function with Modified Kernel**

**Description**

Standard kriging prediction function for the modified correlation functions.

**Usage**

```r
predictAdditive(newdata, x, y, parameter, covtype = "gauss", eps.R = 1e-08, cl, iso = FALSE, se.compute=FALSE)
```

**Arguments**

- `newdata`: matrix containing the points where to perform predictions
- `x`: matrix of input data
- `y`: vector of output data
- `parameter`: (by `kmAdditive` estimated) kriging parameters, list of size of `cl` containing for each clique a list of parameters alpha (single value) and theta (numeric vector of values)
- `covtype`: an optional character string specifying the covariance structure to be used, to be chosen between "gauss", "matern5_2", "matern3_2", "exp" or "powexp" (see DiceKriging), defaults to "gauss"
- `eps.R`: small positive number indicating the nugget effect added to the covariance matrix diagonal, defaults to `eps.R = 1e-08`
- `cl`: list of cliques
- `iso`: boolean vector indicating for each clique if it is isotropic (TRUE) or anisotropic (FALSE), defaults to `iso = FALSE` (all cliques anisotropic)
- `se.compute`: optional boolean. If FALSE, only the kriging mean is computed. If TRUE, the kriging variance (actually, the corresponding standard deviation) is computed, too

**Value**

- `mean`: kriging mean computed at `newdata`
- `sd`: kriging standard deviation computed at `newdata`. Only computed if `se.compute=TRUE`. 
Author(s)
T. Muehlenstaedt, O. Roustant, J. Fruth

References

See Also
kmAdditive

Examples
```r
### example for ishigami function with cliques \{1,3\} and \{2\}
d <- 3
x <- matrix(runif(100*d,-pi,pi),nc=d)
y <- ishigami.fun(x)

cl <- list(c(2), c(1,3))

# constrained ML optimisation with kernel defined by the cliques
parameter <- kmAdditive(x, y, cl = cl)

# prediction with the new model
xpred <- matrix(runif(500 * d,-pi,pi), ncol = d)
ypred <- predictAdditive(xpred, x, y, parameter, cl=cl)
yexact <- ishigami.fun(xpred)

# rmse
sqrt(mean((ypred[,1]- yexact)^2))

# scatterplot
par(mfrow=c(1,1))
plot(yexact, ypred[,1], asp = 1)
abline(0, 1)

### compare to one single clique \{1,2,3\}
c1 <- list(c(1,2,3))

# constrained ML optimisation with kernel defined by the cliques
parameter <- kmAdditive(x, y, cl = c1)

# prediction with the new model
ypred <- predictAdditive(xpred, x, y, parameter, cl=cl)

# rmse
sqrt(mean((ypred$mean- yexact)^2))

# scatterplot
par(mfrow=c(1,1))
plot(yexact, ypred$mean, asp = 1)
```

### isotropic cliques

cl <- list(c(2), c(1, 3))
parameter <- kmAdditive(x, y, cl = cl, iso = c(FALSE, TRUE))
ypred <- predictAdditive(xpred, x, y, parameter, cl = cl, iso = c(FALSE, TRUE))
sqrt(mean((ypred$mean - yexact)^2))

# the same since first clique has length 1
parameter <- kmAdditive(x, y, cl = cl, iso = c(TRUE, TRUE))
ypred <- predictAdditive(xpred, x, y, parameter, cl = cl, iso = c(TRUE, TRUE))
sqrt(mean((ypred$mean - yexact)^2))

---

**simAdditive**

*Simulate GP values from block-additive kernel*

**Description**

Simulate Gaussian process values from a given block-additive kernel

**Usage**

```r
simAdditive(newdata, mu, parameter, covtype, cl, iso = FALSE, eps.R = 1e-08)
```

**Arguments**

- **newdata**: matrix containing the points where to perform simulations
- **mu**: trend parameter
- **parameter**: list of size of `cl` containing for each clique a list of parameters alpha (single value) and theta (numeric vector of values)
- **covtype**: character string specifying the covariance structure to be used, to be chosen between "gauss", "matern5_2", "matern3_2", "exp" or "powexp" (seeDiceKriging)
- **cl**: list of cliques
- **iso**: boolean vector indicating for each clique if it is isotropic (TRUE) or anisotropic (FALSE), defaults to iso = FALSE (all cliques anisotropic)
- **eps.R**: small positive number indicating the nugget effect added to the covariance matrix diagonal, defaults to eps.R = 1e-08

**Value**

A vector containing the simulated values

**Author(s)**

J. Fruth
threshold

References


See Also

`kmAdditive`, `simulate`

Examples

```r
## 2 dimensional simulation
x1 <- x2 <- seq(-1, 1, , 20)
x <- expand.grid(x1, x2)
covtype <- "matern3_2"
mu <- 0

op <- par(no.readonly=TRUE); par(mfrow=c(1, 2), mar=c(1, 1, 1, 1))
# non-additive simulation
parameter <- list(list(alpha=1, theta=c(0.8, 0.8)))
cl <- list(1:2)
set.seed(1)
y <- simAdditive(x, mu, parameter, covtype, cl)
persp(x1, x2, matrix(y, 20), theta=40, col="lightblue", zlab="y")

# additive simulation
parameter <- list(list(alpha=0.5, theta=0.8),
                 list(alpha=0.5, theta=0.8))
cl <- list(1, 2)
set.seed(1)
y <- simAdditive(x, mu, parameter, covtype, cl)
persp(x1, x2, matrix(y, 20), theta=40, col="lightblue", zlab="y")

par(op)
```

---

### threshold

**Threshold indices**

**Description**

All indices below a threshold are set to be zero.

**Usage**

```r
threshold(graphlist, delta, scaled = TRUE, robust = FALSE)
```
Arguments

graphlist an object of class graphlist as obtained from estimateGraph

delta numeric threshold, between 0 and 1 if scaled = TRUE

scaled optional boolean, if TRUE, indices are normalized by the overall variance before for threshold cut, defaults to TRUE

robust optional boolean, if TRUE, upper confidence intervals limits are used for the threshold cut instead of indices themselves, confidence intervals must be provided in graphlist, defaults to FALSE

Value

an object of class graphlist where the indices are thresholded the clique structure is updated respectively, see estimateGraph for a detailed description

Warning

The threshold cut is by default performed on scaled indices. For a cut on the original unscaled indices set scaled = FALSE.

Author(s)

J. Fruth, T. Muehlenstaedt, O. Roustant

References


Examples

# Kriging model prediction
x <- matrix(runif(100*3,-pi,pi),100,3)
KM <- km(~1, design = data.frame(x), response = ishigami.fun(x))
krigingMean <- function(Xnew) predict(object = KM, newdata = Xnew, type = "UK", se.compute = FALSE, checkNames = FALSE)$mean

# full graph estimation
g <- estimateGraph(krigingMean, d=3, n.tot=10000, q.arg=list(min=-pi, max=pi))
print(g[c(2,6)])

# threshold graph
g.cut <- threshold(g, delta = 0.1)
print(g.cut[c(2,6)])
thresholdIdentification

*Function to identify a suitable threshold for an estimateGraph object*

**Description**

From an estimateGraph object and a corresponding data set, candidate threshold values are compared on the prediction performance of the corresponding additive Kriging model. The candidate thresholds are chosen by the biggest jumps in plotDeltaJumps together with 0 (the full model) and 1 (the complete additive model). For each of them the Kriging model with corresponding kernel is estimated and the leave-one-out crossvalidations on the original data sets are compared on scatterplots and RMSE-values.

**Usage**

thresholdIdentification(g, x, y, n.cand = 3, covtype = "matern5_2", KM = NULL)

**Arguments**

- **g**: object of class graphlist as obtained from estimateGraph
- **x**: design matrix of input variables corresponding to g, number of columns should be number of variables
- **y**: vector of output variables of the same length as the columns of x
- **n.cand**: integer, the n.cand biggest jumps are chosen as candidate threshold values. The default value is 3
- **covtype**: optional character string specifying the covariance structure to be used. The default is "matern5_2"
- **KM**: optional object of class km, the full kriging model corresponding to g. With default value NULL, this kriging model is computed by the function itself

**Value**

a list including

- **delta**: vector of threshold candidates
- **models**: list of full model and models with applied thresholds
- **y.cv**: list of vectors containing crossvalidation predictions for each model
- **RMSE**: vector of residual mean squared errors for each model

**Author(s)**

J. Fruth, M. Jastrow

**See Also**

plotDeltaJumps, plotGraphChange
Examples

```r
########### simple 3-dimensional example with one interaction

### data (usually existing)
x <- matrix(seq(0,1,,20), 20, 3)
x <- apply(x,2,sample)
y <- 2*(x[,1]-0.5) * (x[,2]-0.5) + 0.1*sin(10*x[,3])

### FANVOA graph (usually estimated from a meta model over the data)
g <- list(d=3,
tii = matrix(c(0.0140, 0.0008, 0.0002)),
V = 0.0222,
tii.scaled = matrix(c(0.6976, 0.0432, 0.0113))
)
class(g) <- "graphlist"

### plot complete graph
plot(g, plot.i1=FALSE)

### Compare candidate thresholds on prediction performance
set.seed(1)
comparison <- thresholdIdentification(g, x, y, n.cand = 1)
```

---

**totalIndex**

*Estimation of main index indices*

**Description**

Estimation of the unscaled total Sobol index of all main indices by method Liu & Owen (superset importance of main indices).

**Usage**

```r
totalIndex(f.mat, d, q = NULL, q.arg = NULL, n.mc, ...)
```

**Arguments**

- `f.mat` vectorized function of which indices shall be estimated
- `d` integer, number of input factors (vertices)
- `q` a vector of character strings of quantile functions corresponding to the factors distributions, it can be a single character string meaning same distribution for all, if not specified "qunif" is taken
- `q.arg` a list of lists of quantile functions parameters of the distributions in q, it can be a single list meaning same parameters for all, if not specified the default values of the respective distributions are taken
- `n.mc` number of Monte Carlo evaluations
- `...` additional arguments to be passed to the function `f.mat`
Value
A vector containing the unscaled total Sobol indices

Author(s)
J. Fruth

References

See Also
estimateGraph

Examples
```r
totalIndex(f.mat=ishigami.fun, d=3, q.arg=list(min=-pi,max=pi), n.mc=10000)
totalIndex(f.mat=sobol.fun, d=8, q.arg=list(min=0,max=1), n.mc=10000)
```
Index

* datasets
  L, 9

* package
  fanovaGraph-package, 2

constrOptim, 6

DiceKriging, 3, 6, 8

estimateGraph, 2, 3, 6, 8, 10–12, 17, 20

fanovaGraph (fanovaGraph-package), 2
fanovaGraph-package, 2

fast99, 8

i2Index, 5

igraph, 3, 9, 10

kmAdditive, 2, 6, 13, 14, 16

kmPredictWrapper, 8

L, 9

plot, 10
plot (plot.graphlist), 9
plot.graphlist, 2, 9, 13
plotDeltaJumps, 2, 11, 13, 18
plotGraphChange, 10, 11, 18
plotGraphChange (plotGraphChange, plotTk, plotManipulate), 12
plotGraphChange, plotTk, plotManipulate, 12
plotManipulate (plotGraphChange, plotTk, plotManipulate), 12
plotTk (plotGraphChange, plotTk, plotManipulate), 12

predict.km, 8

predictAdditive, 2, 7, 13

print.graphlist (estimateGraph), 3

sensitivity, 3

simAdditive, 2, 15

simulate, 16

tcltk, 12

threshold, 2, 6, 16

thresholdIdentification, 11, 18

totalIndex, 6, 19