Package 'rlibkriging'

February 13, 2023

Type Package

Title Kriging Models using the 'libKriging' Library

Version 0.7-4.3

Date 2023-02-13

Maintainer Yann Richet <yann.richet@irsn.fr>

Description Interface to 'libKriging' 'C++' library <https://github.com/libKriging> that should provide most standard Kriging / Gaussian process regression features
(like in 'DiceKriging', 'kergp' or 'RobustGaSP' packages).
'libKriging' relies on Armadillo linear algebra library (Apache 2 license) by Conrad Sanderson, and 'lbfgsb_cpp' is a 'C++' wrapper by Colin Fang around 'lbfgsb' library (BSD-3 license) by Ciyou Zhu, Richard Byrd, Jorge Nocedal and Jose Luis Morales used for hyperparameters optimization.

License Apache License (>= 2)

Encoding UTF-8

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 4.2)

Imports Rcpp (>= 0.12.11), methods, DiceKriging

Suggests testthat, RobustGaSP, utils, DiceDesign, foreach

SystemRequirements GNU make, cmake (>= 3.2.0), gcc, gfortran, C++17

URL https://github.com/libKriging

RoxygenNote 7.2.1

NeedsCompilation yes

Author Pascal Havé [aut],
Yann Richet [aut, cre] (<https://orcid.org/0000-0002-5677-8458>),
Yves Deville [aut],
Conrad Sanderson [ctb],
Colin Fang [ctb],
Ciyou Zhu [ctb],
Richard Byrd [ctb],
Jorge Nocedal [ctb],
Jose Luis Morales [ctb]

Repository CRAN

Date/Publication 2023-02-13 14:00:02 UTC

R topics documented:

`as.km` *Coerce an Object into a* `km` *Object*

Description

Coerce an object into an object with S4 class `"km"` from the **DiceKriging** package.

Usage

```
as.km(x, ...)
```

Arguments

`x`	Object to be coerced.
`...`	Further arguments for methods.

Details

Such a coercion is typically used to compare the performance of the methods implemented in the current **rlibkriging** package to those which are available in the **DiceKriging** package.

Value

An object with S4 class `"km"`.

`as.km.Kriging` *Coerce a* `Kriging` *object into the* `"km"` *class of the* **DiceKriging** *package.*

Description

Coerce a `Kriging` object into the `"km"` class of the **DiceKriging** package.

Usage

```
## S3 method for class 'Kriging'
as.km(x, .call = NULL, ...)
```

Arguments

`x`	An object with S3 class `"Kriging"`.
`.call`	Force the `call` slot to be filled in the returned `km` object.
`...`	Not used.

Value

An object of having the S4 class `"KM"` which extends the `"km"` class of the **DiceKriging** package and contains an extra `Kriging` slot.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)

k <- Kriging(y, X, "matern3_2")
print(k)

k_km <- as.km(k)
print(k_km)
```

`as.km.NoiseKriging` *Coerce a* `NoiseKriging` *object into the* `"km"` *class of the* **DiceKriging** *package.*

Description

Coerce a `NoiseKriging` object into the `"km"` class of the **DiceKriging** package.

Usage

```
## S3 method for class 'NoiseKriging'
as.km(x, .call = NULL, ...)
```

Arguments

`x`	An object with S3 class `"NoiseKriging"`.
`.call`	Force the `call` slot to be filled in the returned `km` object.
`...`	Not used.

Value

An object of having the S4 class `"KM"` which extends the `"km"` class of the **DiceKriging** package and contains an extra `NoiseKriging` slot.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + X/10 * rnorm(nrow(X)) # add noise dep. on X
## fit and print
k <- NoiseKriging(y, noise=(X/10)^2, X, kernel = "matern3_2")
print(k)

k_km <- as.km(k)
print(k_km)
```

`as.km.NuggetKriging` *Coerce a* `NuggetKriging` *object into the* `"km"` *class of the* **DiceKriging** *package.*

Description

Coerce a `NuggetKriging` object into the `"km"` class of the **DiceKriging** package.

Usage

```
## S3 method for class 'NuggetKriging'
as.km(x, .call = NULL, ...)
```

Arguments

`x`	An object with S3 class `"NuggetKriging"`.
`.call`	Force the `call` slot to be filled in the returned `km` object.
`...`	Not used.

Value

An object of having the S4 class `"KM"` which extends the `"km"` class of the **DiceKriging** package and contains an extra `NuggetKriging` slot.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + 0.1 * rnorm(nrow(X))

k <- NuggetKriging(y, X, "matern3_2")
print(k)

k_km <- as.km(k)
print(k_km)
```

`as.list.Kriging` *Coerce a* `Kriging` *Object into a List*

Description

Coerce a `Kriging` Object into a List

Usage

```
## S3 method for class 'Kriging'
as.list(x, ...)
```

Arguments

`x`	An object with class "`Kriging`".
`...`	Ignored

Value

A list with its elements copying the content of the `Kriging` object fields: `kernel`, `optim`, `objective`, `theta` (vector of ranges), `sigma2` (variance), `X`, `centerX`, `scaleX`, `y`, `centerY`, `scaleY`, `regmodel`, `F`, `T`, `M`, `z`, `beta`.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x ) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)

k <- Kriging(y, X, kernel = "matern3_2")

l <- as.list(k)
cat(paste0(names(l), " =" , l, collapse = "\n"))
```

`as.list.NoiseKriging` *Coerce a* `NoiseKriging` *Object into a List*

Description

Coerce a `NoiseKriging` Object into a List

Usage

```
## S3 method for class 'NoiseKriging'
as.list(x, ...)
```

Arguments

`x`	An object with class "`NoiseKriging`".
`...`	Ignored

Value

A list with its elements copying the content of the `NoiseKriging` object fields: `kernel`, `optim`, `objective`, `theta` (vector of ranges), `sigma2` (variance), `X`, `centerX`, `scaleX`, `y`, `centerY`, `scaleY`, `regmodel`, `F`, `T`, `M`, `z`, `beta`.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + X/10 * rnorm(nrow(X)) # add noise dep. on X

k <- NoiseKriging(y, noise=(X/10)^2, X, kernel = "matern3_2")

l <- as.list(k)
cat(paste0(names(l), " =" , l, collapse = "\n"))
```

`as.list.NuggetKriging` *Coerce a* `NuggetKriging` *Object into a List*

Description

Coerce a `NuggetKriging` Object into a List

Usage

```
## S3 method for class 'NuggetKriging'
as.list(x, ...)
```

Arguments

`x`	An object with class "`NuggetKriging`".
`...`	Ignored

Value

A list with its elements copying the content of the `NuggetKriging` object fields: `kernel`, `optim`, `objective`, `theta` (vector of ranges), `sigma2` (variance), `X`, `centerX`, `scaleX`, `y`, `centerY`, `scaleY`, `regmodel`, `F`, `T`, `M`, `z`, `beta`.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + 0.1 * rnorm(nrow(X))

k <- NuggetKriging(y, X, kernel = "matern3_2")

l <- as.list(k)
cat(paste0(names(l), " =" , l, collapse = "\n"))
```

copy *Duplicate object.*

Description

Duplicate a model given in `object`.

Usage

```
copy(object, ...)
```

Arguments

`object`	An object representing a fitted model.
`...`	Ignored.

Value

The copied object.

copy.Kriging *Duplicate a Kriging Model*

Description

Duplicate a Kriging Model

Usage

```
## S3 method for class 'Kriging'
copy(object, ...)
```

Arguments

`object`	An S3 Kriging object.
`...`	Not used.

Value

The copy of object.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)

k <- Kriging(y, X, kernel = "matern3_2", objective="LMP")
print(k)

print(copy(k))
```

---

copy.NoiseKriging *Duplicate a NoiseKriging Model*

Description

Duplicate a NoiseKriging Model

Usage

```
## S3 method for class 'NoiseKriging'
copy(object, ...)
```

Arguments

object	An S3 NoiseKriging object.
...	Not used.

Value

The copy of object.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + X/10 * rnorm(nrow(X))

k <- NoiseKriging(y, (X/10)^2, X, kernel = "matern3_2", objective="LL")
print(k)

print(copy(k))
```

copy.NuggetKriging *Duplicate a NuggetKriging Model*

Description

Duplicate a NuggetKriging Model

Usage

```
## S3 method for class 'NuggetKriging'
copy(object, ...)
```

Arguments

object	An S3 NuggetKriging object.
...	Not used.

Value

The copy of object.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + 0.1 * rnorm(nrow(X))

k <- NuggetKriging(y, X, kernel = "matern3_2", objective="LMP")
print(k)

print(copy(k))
```

fit *Fit model on data.*

Description

Fit a model given in object.

Usage

```
fit(object, ...)
```

Arguments

object	An object representing a fitted model.
...	Further arguments of function

Value

No return value. Kriging object argument is modified.

---

fit.Kriging *Fit* Kriging *object on given data.*

Description

The hyper-parameters (variance and vector of correlation ranges) are estimated thanks to the optimization of a criterion given by objective, using the method given in optim.

Usage

```
## S3 method for class 'Kriging'
fit(
  object,
  y,
  X,
  regmodel = c("constant", "linear", "interactive"),
  normalize = FALSE,
  optim = c("BFGS", "Newton", "none"),
  objective = c("LL", "LOO", "LMP"),
  parameters = NULL,
  ...
)
```

Arguments

object	S3 Kriging object.
y	Numeric vector of response values.
X	Numeric matrix of input design.
regmodel	Universal Kriging linear trend.
normalize	Logical. If TRUE both the input matrix X and the response y in normalized to take values in the interval $[0, 1]$.
optim	Character giving the Optimization method used to fit hyper-parameters. Possible values are: "BFGS", "Newton" and "none", the later simply keeping the values given in parameters. The method "BFGS" uses the gradient of the objective. The method "Newton" uses both the gradient and the Hessian of the objective.

objective	Character giving the objective function to optimize. Possible values are: "LL" for the Log-Likelihood, "LOO" for the Leave-One-Out sum of squares and "LMP" for the Log-Marginal Posterior.
parameters	Initial values for the hyper-parameters. When provided this must be named list with elements "sigma2" and "theta" containing the initial value(s) for the variance and for the range parameters. If theta is a matrix with more than one row, each row is used as a starting point for optimization.
...	Ignored.

Value

No return value. Kriging object argument is modified.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)
points(X, y, col = "blue", pch = 16)

k <- Kriging("matern3_2")
print(k)

fit(k,y,X)
print(k)
```

---

fit.NoiseKriging *Fit* NoiseKriging *object on given data.*

Description

The hyper-parameters (variance and vector of correlation ranges) are estimated thanks to the optimization of a criterion given by objective, using the method given in optim.

Usage

```
## S3 method for class 'NoiseKriging'
fit(
  object,
  y,
  noise,
  X,
```

```
  regmodel = c("constant", "linear", "interactive"),
  normalize = FALSE,
  optim = c("BFGS", "none"),
  objective = c("LL"),
  parameters = NULL,
  ...
)
```

Arguments

object	S3 NoiseKriging object.
y	Numeric vector of response values.
noise	Numeric vector of response variances.
X	Numeric matrix of input design.
regmodel	Universal NoiseKriging linear trend.
normalize	Logical. If TRUE both the input matrix X and the response y in normalized to take values in the interval $[0, 1]$.
optim	Character giving the Optimization method used to fit hyper-parameters. Possible values are: "BFGS" and "none", the later simply keeping the values given in parameters. The method "BFGS" uses the gradient of the objective.
objective	Character giving the objective function to optimize. Possible values are: "LL" for the Log-Likelihood.
parameters	Initial values for the hyper-parameters. When provided this must be named list with elements "sigma2" and "theta" containing the initial value(s) for the variance and for the range parameters. If theta is a matrix with more than one row, each row is used as a starting point for optimization.
...	Ignored.

Value

No return value. NoiseKriging object argument is modified.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + X/10 * rnorm(nrow(X)) # add noise dep. on X
points(X, y, col = "blue", pch = 16)

k <- NoiseKriging("matern3_2")
print(k)
```

```
fit(k,y,noise=(X/10)^2,X)
print(k)
```

`fit.NuggetKriging` *Fit* `NuggetKriging` *object on given data.*

Description

The hyper-parameters (variance and vector of correlation ranges) are estimated thanks to the optimization of a criterion given by `objective`, using the method given in `optim`.

Usage

```
## S3 method for class 'NuggetKriging'
fit(
  object,
  y,
  X,
  regmodel = c("constant", "linear", "interactive"),
  normalize = FALSE,
  optim = c("BFGS", "none"),
  objective = c("LL", "LMP"),
  parameters = NULL,
  ...
)
```

Arguments

`object`	S3 NuggetKriging object.
`y`	Numeric vector of response values.
`X`	Numeric matrix of input design.
`regmodel`	Universal NuggetKriging linear trend.
`normalize`	Logical. If `TRUE` both the input matrix `X` and the response `y` in normalized to take values in the interval $[0, 1]$.
`optim`	Character giving the Optimization method used to fit hyper-parameters. Possible values are: `"BFGS"` and `"none"`, the later simply keeping the values given in `parameters`. The method `"BFGS"` uses the gradient of the objective.
`objective`	Character giving the objective function to optimize. Possible values are: `"LL"` for the Log-Likelihood and `"LMP"` for the Log-Marginal Posterior.
`parameters`	Initial values for the hyper-parameters. When provided this must be named list with some elements `"sigma2"`, `"theta"`, `"nugget"` containing the initial value(s) for the variance, range and nugget parameters. If `theta` is a matrix with more than one row, each row is used as a starting point for optimization.
`...`	Ignored.

Value

No return value. NuggetKriging object argument is modified.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + 0.1 * rnorm(nrow(X))
points(X, y, col = "blue", pch = 16)

k <- NuggetKriging("matern3_2")
print(k)

fit(k,y,X)
print(k)
```

KM *Create an* KM *Object*

Description

Create an object of S4 class "KM" similar to a km object in the **DiceKriging** package.

Usage

```
KM(
  formula = ~1,
  design,
  response,
  covtype = c("matern5_2", "gauss", "matern3_2", "exp"),
  coef.trend = NULL,
  coef.cov = NULL,
  coef.var = NULL,
  nugget = NULL,
  nugget.estim = FALSE,
  noise.var = NULL,
  estim.method = c("MLE", "LOO"),
  penalty = NULL,
  optim.method = "BFGS",
  lower = NULL,
  upper = NULL,
  parinit = NULL,
```

```
  multistart = 1,
  control = NULL,
  gr = TRUE,
  iso = FALSE,
  scaling = FALSE,
  knots = NULL,
  kernel = NULL,
  ...
)
```

Arguments

formula	R formula object to setup the linear trend in Universal Kriging. Supports ~ 1, ~. and ~ .^2.
design	Data frame. The design of experiments.
response	Vector of output values.
covtype	Covariance structure. For now all the kernels are tensor product kernels.
coef.trend	Optional value for a fixed vector of trend coefficients. If given, no optimization is done.
coef.cov	Optional value for a fixed correlation range value. If given, no optimization is done.
coef.var	Optional value for a fixed variance. If given, no optimization is done.
nugget, nugget.estim, noise.var	Not implemented yet.
estim.method	Estimation criterion. "MLE" for Maximum-Likelihood or "LOO" for Leave-One-Out cross-validation.
penalty	Not implemented yet.
optim.method	Optimization algorithm used in the optimization of the objective given in estim.method. Supports "BFGS".
lower, upper	Not implemented yet.
parinit	Initial values for the correlation ranges which will be optimized using optim.method.
multistart, control, gr, iso	Not implemented yet.
scaling, knots, kernel,	Not implemented yet.
...	Ignored.

Details

The class "KM" extends the "km" class of the **DiceKriging** package, hence has all slots of "km". It also has an extra slot "Kriging" slot which contains a copy of the original object.

Value

A KM object. See **Details**.

Author(s)

Yann Richet <yann.richet@irsn.fr>

See Also

km in the **DiceKriging** package for more details on the slots.

Examples

```
# a 16-points factorial design, and the corresponding response
d <- 2; n <- 16
design.fact <- as.matrix(expand.grid(x1 = seq(0, 1, length = 4),
                                     x2 = seq(0, 1, length = 4)))
y <- apply(design.fact, 1, DiceKriging::branin)

# Using `km` from DiceKriging and a similar `KM` object
# kriging model 1 : matern5_2 covariance structure, no trend, no nugget effect
km1 <- DiceKriging::km(design = design.fact, response = y, covtype = "gauss",
                       parinit = c(.5, 1), control = list(trace = FALSE))
KM1 <- KM(design = design.fact, response = y, covtype = "gauss",
          parinit = c(.5, 1))
```

KM-class *S4 class for Kriging Models Extending the "km" Class*

Description

This class is intended to be used either by using its own dedicated S4 methods or by using the S4 methods inherited from the "km" class of the **libKriging** package.

Slots

`d,n,X,y,p,F` Number of (numeric) inputs, number of observations, design matrix, response vector, number of trend variables, trend matrix.

`trend.formula,trend.coef` Formula used for the trend, vector $\hat{\beta}$ of estimated (or fixed) trend coefficients with length p.

`covariance` A S4 object with class "covTensorProduct" representing a covariance kernel.

`noise.flag,noise.var` Logical flag and numeric value for an optional noise term.

`known.param` A character code indicating what parameters are known.

`lower,upper` Bounds on the correlation range parameters.

`method,penalty,optim.method,control,gr,parinit` Objects defining the estimation criterion, the optimization.

`T,M,z` Auxiliary variables (matrices and vectors) that can be used in several computations.

`case` The possible concentration (a.k.a. profiling) of the likelihood.

`param.estim` Logical. Is an estimation used?

`Kriging` A copy of the `Kriging` object used to create the current `KM` object.

Author(s)

Yann Richet <yann.richet@irsn.fr>

See Also

km-class in the **DiceKriging** package. The creator KM.

`Kriging` *Create an object with S3 class* `"Kriging"` *using the* **libKriging** *library.*

Description

The hyper-parameters (variance and vector of correlation ranges) are estimated thanks to the optimization of a criterion given by `objective`, using the method given in `optim`.

Usage

```
Kriging(
  y = NULL,
  X = NULL,
  kernel = NULL,
  regmodel = c("constant", "linear", "interactive"),
  normalize = FALSE,
  optim = c("BFGS", "Newton", "none"),
  objective = c("LL", "LOO", "LMP"),
  parameters = NULL
)
```

Arguments

`y`	Numeric vector of response values.
`X`	Numeric matrix of input design.
`kernel`	Character defining the covariance model: `"exp"`, `"gauss"`, `"matern3_2"`, `"matern5_2"`.
`regmodel`	Universal Kriging linear trend.
`normalize`	Logical. If `TRUE` both the input matrix `X` and the response `y` in normalized to take values in the interval $[0, 1]$.
`optim`	Character giving the Optimization method used to fit hyper-parameters. Possible values are: `"BFGS"`, `"Newton"` and `"none"`, the later simply keeping the values given in `parameters`. The method `"BFGS"` uses the gradient of the objective. The method `"Newton"` uses both the gradient and the Hessian of the objective.
`objective`	Character giving the objective function to optimize. Possible values are: `"LL"` for the Log-Likelihood, `"LOO"` for the Leave-One-Out sum of squares and `"LMP"` for the Log-Marginal Posterior.

parameters — Initial values for the hyper-parameters. When provided this must be named list with elements "sigma2" and "theta" containing the initial value(s) for the variance and for the range parameters. If theta is a matrix with more than one row, each row is used as a starting point for optimization.

Value

An object with S3 class "Kriging". Should be used with its predict, simulate, update methods.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)
## fit and print
k <- Kriging(y, X, kernel = "matern3_2")
print(k)

x <- as.matrix(seq(from = 0, to = 1, length.out = 101))
p <- predict(k, x = x, stdev = TRUE, cov = FALSE)

plot(f)
points(X, y)
lines(x, p$mean, col = "blue")
polygon(c(x, rev(x)), c(p$mean - 2 * p$stdev, rev(p$mean + 2 * p$stdev)),
border = NA, col = rgb(0, 0, 1, 0.2))

s <- simulate(k, nsim = 10, seed = 123, x = x)

matlines(x, s, col = rgb(0, 0, 1, 0.2), type = "l", lty = 1)
```

---

leaveOneOut *Compute Leave-One-Out*

Description

Compute the leave-One-Out error of a model given in object.

Usage

```
leaveOneOut(object, ...)
```

Arguments

object	An object representing a fitted model.
...	Ignored.

Value

The Leave-One-Out sum of squares.

---

leaveOneOut.Kriging *Get leaveOneOut of Kriging Model*

Description

Get leaveOneOut of Kriging Model

Usage

```
## S3 method for class 'Kriging'
leaveOneOut(object, ...)
```

Arguments

object	An S3 Kriging object.
...	Not used.

Value

The leaveOneOut computed for fitted $\boldsymbol{theta}$.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)

k <- Kriging(y, X, kernel = "matern3_2", objective="LOO")
print(k)

leaveOneOut(k)
```

leaveOneOutFun *Leave-One-Out function*

Description

Compute the leave-One-Out error of a model given in `object`, at a different value of the parameters.

Usage

```
leaveOneOutFun(object, ...)
```

Arguments

`object`	An object representing a fitted model.
`...`	Further arguments of function (eg. range).

Value

The Leave-One-Out sum of squares.

leaveOneOutFun.Kriging *Compute Leave-One-Out (LOO) error for an object with S3 class "Kriging" representing a kriging model.*

Description

The returned value is the sum of squares $\sum_{i=1}^{n}[y_i - \hat{y}_{i,(-i)}]^2$ where $\hat{y}_{i,(-i)}$ is the prediction of y_i based on the the observations y_j with $j \neq i$.

Usage

```
## S3 method for class 'Kriging'
leaveOneOutFun(object, theta, grad = FALSE, ...)
```

Arguments

`object`	A `Kriging` object.
`theta`	A numeric vector of range parameters at which the LOO will be evaluated.
`grad`	Logical. Should the gradient (w.r.t. `theta`) be returned?
`...`	Not used.

Value

The leave-One-Out value computed for the given vector $\boldsymbol{\theta}$ of correlation ranges.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)

k <- Kriging(y, X, kernel = "matern3_2", objective = "LOO", optim="BFGS")
print(k)

loo <-  function(theta) leaveOneOutFun(k, theta)$leaveOneOut
t <-  seq(from = 0.001, to = 2, length.out = 101)
plot(t, loo(t), type = "l")
abline(v = k$theta(), col = "blue")
```

---

logLikelihood *Compute Log-Likelihood*

Description

Compute the log-Likelihood of a model given in `object`.

Usage

```
logLikelihood(object, ...)
```

Arguments

`object`	An object representing a fitted model.
`...`	Ignored.

Value

The log-likelihood.

logLikelihood.Kriging *Get Log-Likelihood of Kriging Model*

Description

Get Log-Likelihood of Kriging Model

Usage

```
## S3 method for class 'Kriging'
logLikelihood(object, ...)
```

Arguments

object	An S3 Kriging object.
...	Not used.

Value

The log-Likelihood computed for fitted $\boldsymbol{theta}$.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)

k <- Kriging(y, X, kernel = "matern3_2", objective="LL")
print(k)

logLikelihood(k)
```

logLikelihood.NoiseKriging *Get logLikelihood of NoiseKriging Model*

Description

Get logLikelihood of NoiseKriging Model

Usage

```
## S3 method for class 'NoiseKriging'
logLikelihood(object, ...)
```

Arguments

object	An S3 NoiseKriging object.
...	Not used.

Value

The logLikelihood computed for fitted $theta_sigma2$.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + X/10 * rnorm(nrow(X))

k <- NoiseKriging(y, (X/10)^2, X, kernel = "matern3_2", objective="LL")
print(k)

logLikelihood(k)
```

---

logLikelihood.NuggetKriging

Get logLikelihood of NuggetKriging Model

Description

Get logLikelihood of NuggetKriging Model

Usage

```
## S3 method for class 'NuggetKriging'
logLikelihood(object, ...)
```

Arguments

object	An S3 NuggetKriging object.
...	Not used.

Value

The logLikelihood computed for fitted $theta_alpha$.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + 0.1 * rnorm(nrow(X))

k <- NuggetKriging(y, X, kernel = "matern3_2", objective="LL")
print(k)

logLikelihood(k)
```

---

logLikelihoodFun *Log-Likelihood function*

Description

Compute the log-Likelihood of a model given in `object`, at a different value of the parameters.

Usage

```
logLikelihoodFun(object, ...)
```

Arguments

`object`	An object representing a fitted model.
`...`	Further arguments of function (eg. range).

Value

The log-likelihood.

logLikelihoodFun.Kriging *Compute Log-Likelihood of Kriging Model*

Description

Compute Log-Likelihood of Kriging Model

Usage

```
## S3 method for class 'Kriging'
logLikelihoodFun(object, theta, grad = FALSE, hess = FALSE, ...)
```

Arguments

object	An S3 Kriging object.
theta	A numeric vector of (positive) range parameters at which the log-likelihood will be evaluated.
grad	Logical. Should the function return the gradient?
hess	Logical. Should the function return Hessian?
...	Not used.

Value

The log-Likelihood computed for given $\boldsymbol{theta}$.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)

k <- Kriging(y, X, kernel = "matern3_2")
print(k)

ll <- function(theta) logLikelihoodFun(k, theta)$logLikelihood

t <- seq(from = 0.001, to = 2, length.out = 101)
plot(t, ll(t), type = 'l')
abline(v = k$theta(), col = "blue")
```

logLikelihoodFun.NoiseKriging *Compute Log-Likelihood of NoiseKriging Model*

Description

Compute Log-Likelihood of NoiseKriging Model

Usage

```
## S3 method for class 'NoiseKriging'
logLikelihoodFun(object, theta_sigma2, grad = FALSE, ...)
```

Arguments

object	An S3 NoiseKriging object.
theta_sigma2	A numeric vector of (positive) range parameters and variance at which the log-likelihood will be evaluated.
grad	Logical. Should the function return the gradient?
...	Not used.

Value

The log-Likelihood computed for given $theta_sigma2$.

Author(s)

Yann Richet <yann.richet@irsn.fr>

logLikelihoodFun.NuggetKriging *Compute Log-Likelihood of NuggetKriging Model*

Description

Compute Log-Likelihood of NuggetKriging Model

Usage

```
## S3 method for class 'NuggetKriging'
logLikelihoodFun(object, theta_alpha, grad = FALSE, ...)
```

Arguments

object	An S3 NuggetKriging object.
theta_alpha	A numeric vector of (positive) range parameters and variance over variance plus nugget at which the log-likelihood will be evaluated.
grad	Logical. Should the function return the gradient?
...	Not used.

Value

The log-Likelihood computed for given $theta_alpha$.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + 0.1 * rnorm(nrow(X))

k <- NuggetKriging(y, X, kernel = "matern3_2")
print(k)

theta0 = k$theta()
ll_alpha <- function(alpha) logLikelihoodFun(k,cbind(theta0,alpha))$logLikelihood
a <- seq(from = 0.9, to = 1.0, length.out = 101)
plot(a, Vectorize(ll_alpha)(a), type = "l",xlim=c(0.9,1))
abline(v = k$sigma2()/(k$sigma2()+k$nugget()), col = "blue")

alpha0 = k$sigma2()/(k$sigma2()+k$nugget())
ll_theta <- function(theta) logLikelihoodFun(k,cbind(theta,alpha0))$logLikelihood
t <- seq(from = 0.001, to = 2, length.out = 101)
plot(t, Vectorize(ll_theta)(t), type = 'l')
abline(v = k$theta(), col = "blue")

ll <- function(theta_alpha) logLikelihoodFun(k,theta_alpha)$logLikelihood
a <- seq(from = 0.9, to = 1.0, length.out = 31)
t <- seq(from = 0.001, to = 2, length.out = 101)
contour(t,a,matrix(ncol=length(a),ll(expand.grid(t,a))),xlab="theta",ylab="sigma2/(sigma2+nugget)")
points(k$theta(),k$sigma2()/(k$sigma2()+k$nugget()),col='blue')
```

logMargPost *Compute log-Marginal Posterior*

Description

Compute the log-Marginal Posterior of a model given in `object`.

Usage

```
logMargPost(object, ...)
```

Arguments

`object`	An object representing a fitted model.
`...`	Ignored.

Value

The log-marginal posterior.

logMargPost.Kriging *Get logMargPost of Kriging Model*

Description

Get logMargPost of Kriging Model

Usage

```
## S3 method for class 'Kriging'
logMargPost(object, ...)
```

Arguments

`object`	An S3 Kriging object.
`...`	Not used.

Value

The logMargPost computed for fitted $\boldsymbol{theta}$.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)

k <- Kriging(y, X, kernel = "matern3_2", objective="LMP")
print(k)

logMargPost(k)
```

`logMargPost.NuggetKriging` *Get logMargPost of NuggetKriging Model*

Description

Get logMargPost of NuggetKriging Model

Usage

```
## S3 method for class 'NuggetKriging'
logMargPost(object, ...)
```

Arguments

`object`	An S3 NuggetKriging object.
`...`	Not used.

Value

The logMargPost computed for fitted $theta_alpha$.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + 0.1 * rnorm(nrow(X))

k <- NuggetKriging(y, X, kernel = "matern3_2", objective="LMP")
print(k)

logMargPost(k)
```

logMargPostFun *log-Marginal Posterior function*

Description

Compute the log-Marginal Posterior of a model given in `object`, at a different value of the parameters.

Usage

```
logMargPostFun(object, ...)
```

Arguments

`object`	An object representing a fitted model.
`...`	Further arguments of function (eg. range).

Value

The log-marginal posterior.

logMargPostFun.Kriging *Compute the log-marginal posterior of a kriging model, using the prior XXXY.*

Description

Compute the log-marginal posterior of a kriging model, using the prior XXXY.

Usage

```
## S3 method for class 'Kriging'
logMargPostFun(object, theta, grad = FALSE, ...)
```

Arguments

`object`	S3 Kriging object.
`theta`	Numeric vector of correlation range parameters at which the function is to be evaluated.
`grad`	Logical. Should the function return the gradient (w.r.t theta)?
`...`	Not used.

Value

The value of the log-marginal posterior computed for the given vector theta.

Author(s)

Yann Richet <yann.richet@irsn.fr>

References

XXXY A reference describing the model (prior, ...)

See Also

rgasp in the RobustGaSP package.

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)

k <- Kriging(y, X, "matern3_2", objective="LMP")
print(k)

lmp <- function(theta) logMargPostFun(k, theta)$logMargPost

t <- seq(from = 0.01, to = 2, length.out = 101)
plot(t, lmp(t), type = "l")
abline(v = k$theta(), col = "blue")
```

---

logMargPostFun.NuggetKriging

Compute the log-marginal posterior of a kriging model, using the prior XXXY.

Description

Compute the log-marginal posterior of a kriging model, using the prior XXXY.

Usage

```
## S3 method for class 'NuggetKriging'
logMargPostFun(object, theta_alpha, grad = FALSE, ...)
```

Arguments

object	S3 NuggetKriging object.
theta_alpha	Numeric vector of correlation range and variance over variance plus nugget parameters at which the function is to be evaluated.
grad	Logical. Should the function return the gradient (w.r.t theta_alpha)?
...	Not used.

Value

The value of the log-marginal posterior computed for the given vector $theta_alpha$.

Author(s)

Yann Richet <yann.richet@irsn.fr>

References

XXXY A reference describing the model (prior, ...)

See Also

rgasp in the RobustGaSP package.

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + 0.1 * rnorm(nrow(X))

k <- NuggetKriging(y, X, "matern3_2", objective="LMP")
print(k)

theta0 = k$theta()
lmp_alpha <- function(alpha) k$logMargPostFun(cbind(theta0,alpha))$logMargPost
a <- seq(from = 0.9, to = 1.0, length.out = 101)
plot(a, Vectorize(lmp_alpha)(a), type = "l",xlim=c(0.9,1))
abline(v = k$sigma2()/(k$sigma2()+k$nugget()), col = "blue")

alpha0 = k$sigma2()/(k$sigma2()+k$nugget())
lmp_theta <- function(theta) k$logMargPostFun(cbind(theta,alpha0))$logMargPost
t <- seq(from = 0.001, to = 2, length.out = 101)
plot(t, Vectorize(lmp_theta)(t), type = 'l')
abline(v = k$theta(), col = "blue")

lmp <- function(theta_alpha) k$logMargPostFun(theta_alpha)$logMargPost
t <- seq(from = 0.4, to = 0.6, length.out = 51)
a <- seq(from = 0.9, to = 1, length.out = 51)
contour(t,a,matrix(ncol=length(t),lmp(expand.grid(t,a))),
 nlevels=50,xlab="theta",ylab="sigma2/(sigma2+nugget)")
points(k$theta(),k$sigma2()/(k$sigma2()+k$nugget()),col='blue')
```

NoiseKM *Create an* NoiseKM *Object*

Description

Create an object of S4 class "NoiseKM" similar to a km object in the **DiceKriging** package.

Usage

```
NoiseKM(
  formula = ~1,
  design,
  response,
  covtype = c("matern5_2", "gauss", "matern3_2", "exp"),
  coef.trend = NULL,
  coef.cov = NULL,
  coef.var = NULL,
  nugget = NULL,
  nugget.estim = FALSE,
  noise.var,
  estim.method = c("MLE", "LOO"),
  penalty = NULL,
  optim.method = "BFGS",
  lower = NULL,
  upper = NULL,
  parinit = NULL,
  multistart = 1,
  control = NULL,
  gr = TRUE,
  iso = FALSE,
  scaling = FALSE,
  knots = NULL,
  kernel = NULL,
  ...
)
```

Arguments

formula	R formula object to setup the linear trend in Universal NoiseKriging. Supports ~ 1, ~. and ~ . ^2.
design	Data frame. The design of experiments.
response	Vector of output values.
covtype	Covariance structure. For now all the kernels are tensor product kernels.
coef.trend	Optional value for a fixed vector of trend coefficients. If given, no optimization is done.

`coef.cov`	Optional value for a fixed correlation range value. If given, no optimization is done.
`coef.var`	Optional value for a fixed variance. If given, no optimization is done.
`nugget, nugget.estim`	Not implemented.
`noise.var`	Vector of output values variance.
`estim.method`	Estimation criterion. `"MLE"` for Maximum-Likelihood or `"LOO"` for Leave-One-Out cross-validation.
`penalty`	Not implemented yet.
`optim.method`	Optimization algorithm used in the optimization of the objective given in `estim.method`. Supports `"BFGS"`.
`lower, upper`	Not implemented yet.
`parinit`	Initial values for the correlation ranges which will be optimized using `optim.method`.
`multistart, control, gr, iso`	Not implemented yet.
`scaling, knots, kernel,`	Not implemented yet.
`...`	Ignored.

Details

The class `"NoiseKM"` extends the `"km"` class of the **DiceKriging** package, hence has all slots of `"km"`. It also has an extra slot `"NoiseKriging"` slot which contains a copy of the original object.

Value

A NoiseKM object. See **Details**.

Author(s)

Yann Richet <yann.richet@irsn.fr>

See Also

`km` in the **DiceKriging** package for more details on the slots.

Examples

```
# a 16-points factorial design, and the corresponding response
d <- 2; n <- 16
design.fact <- as.matrix(expand.grid(x1 = seq(0, 1, length = 4),
                                     x2 = seq(0, 1, length = 4)))
y <- apply(design.fact, 1, DiceKriging::branin) + rnorm(nrow(design.fact))

# Using `km` from DiceKriging and a similar `NoiseKM` object
# kriging model 1 : matern5_2 covariance structure, no trend, no nugget effect
km1 <- DiceKriging::km(design = design.fact, response = y, covtype = "gauss",
```

```
                        noise.var=rep(1,nrow(design.fact)),
                        parinit = c(.5, 1), control = list(trace = FALSE))
KM1 <- NoiseKM(design = design.fact, response = y, covtype = "gauss",
          noise=rep(1,nrow(design.fact)), parinit = c(.5, 1))
```

NoiseKM-class *S4 class for NoiseKriging Models Extending the* "km" *Class*

Description

This class is intended to be used either by using its own dedicated S4 methods or by using the S4 methods inherited from the "km" class of the **libKriging** package.

Slots

`d,n,X,y,p,F` Number of (numeric) inputs, number of observations, design matrix, response vector, number of trend variables, trend matrix.

`trend.formula,trend.coef` Formula used for the trend, vector $\hat{\beta}$ of estimated (or fixed) trend coefficients with length p.

`covariance` A S4 object with class "covTensorProduct" representing a covariance kernel.

`noise.flag,noise.var` Logical flag and numeric value for an optional noise term.

`known.param` A character code indicating what parameters are known.

`lower,upper` Bounds on the correlation range parameters.

`method,penalty,optim.method,control,gr,parinit` Objects defining the estimation criterion, the optimization.

`T,M,z` Auxiliary variables (matrices and vectors) that can be used in several computations.

`case` The possible concentration (a.k.a. profiling) of the likelihood.

`param.estim` Logical. Is an estimation used?

`NoiseKriging` A copy of the `NoiseKriging` object used to create the current `NoiseKM` object.

Author(s)

Yann Richet <yann.richet@irsn.fr>

See Also

`km-class` in the **DiceKriging** package. The creator `NoiseKM`.

`NoiseKriging` *Create an object with S3 class* `"NoiseKriging"` *using the* **libKriging** *library.*

Description

The hyper-parameters (variance and vector of correlation ranges) are estimated thanks to the optimization of a criterion given by `objective`, using the method given in `optim`.

Usage

```
NoiseKriging(
  y = NULL,
  noise = NULL,
  X = NULL,
  kernel = NULL,
  regmodel = c("constant", "linear", "interactive"),
  normalize = FALSE,
  optim = c("BFGS", "none"),
  objective = c("LL"),
  parameters = NULL
)
```

Arguments

`y`	Numeric vector of response values.
`noise`	Numeric vector of response variances.
`X`	Numeric matrix of input design.
`kernel`	Character defining the covariance model: `"exp"`, `"gauss"`, `"matern3_2"`, `"matern5_2"`.
`regmodel`	Universal NoiseKriging linear trend.
`normalize`	Logical. If `TRUE` both the input matrix `X` and the response `y` in normalized to take values in the interval $[0, 1]$.
`optim`	Character giving the Optimization method used to fit hyper-parameters. Possible values are: `"BFGS"` and `"none"`, the later simply keeping the values given in `parameters`. The method `"BFGS"` uses the gradient of the objective.
`objective`	Character giving the objective function to optimize. Possible values are: `"LL"` for the Log-Likelihood.
`parameters`	Initial values for the hyper-parameters. When provided this must be named list with elements `"sigma2"` and `"theta"` containing the initial value(s) for the variance and for the range parameters. If `theta` is a matrix with more than one row, each row is used as a starting point for optimization.

Value

An object with S3 class `"NoiseKriging"`. Should be used with its `predict`, `simulate`, `update` methods.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + X/10 * rnorm(nrow(X)) # add noise dep. on X
## fit and print
k <- NoiseKriging(y, noise=(X/10)^2, X, kernel = "matern3_2")
print(k)

x <- as.matrix(seq(from = 0, to = 1, length.out = 101))
p <- predict(k,x = x, stdev = TRUE, cov = FALSE)

plot(f)
points(X, y)
lines(x, p$mean, col = "blue")
polygon(c(x, rev(x)), c(p$mean - 2 * p$stdev, rev(p$mean + 2 * p$stdev)),
border = NA, col = rgb(0, 0, 1, 0.2))

s <- simulate(k, nsim = 10, seed = 123, x = x)

matlines(x, s, col = rgb(0, 0, 1, 0.2), type = "l", lty = 1)
```

NuggetKM *Create an* NuggetKM *Object*

Description

Create an object of S4 class "NuggetKM" similar to a km object in the **DiceKriging** package.

Usage

```
NuggetKM(
  formula = ~1,
  design,
  response,
  covtype = c("matern5_2", "gauss", "matern3_2", "exp"),
  coef.trend = NULL,
  coef.cov = NULL,
  coef.var = NULL,
  nugget = NULL,
  nugget.estim = TRUE,
  noise.var = NULL,
  estim.method = c("MLE", "LOO"),
  penalty = NULL,
```

```
  optim.method = "BFGS",
  lower = NULL,
  upper = NULL,
  parinit = NULL,
  multistart = 1,
  control = NULL,
  gr = TRUE,
  iso = FALSE,
  scaling = FALSE,
  knots = NULL,
  kernel = NULL,
  ...
)
```

Arguments

`formula`	R formula object to setup the linear trend in Universal NuggetKriging. Supports ~ 1, ~. and ~ . ^2.
`design`	Data frame. The design of experiments.
`response`	Vector of output values.
`covtype`	Covariance structure. For now all the kernels are tensor product kernels.
`coef.trend`	Optional value for a fixed vector of trend coefficients. If given, no optimization is done.
`coef.cov`	Optional value for a fixed correlation range value. If given, no optimization is done.
`coef.var`	Optional value for a fixed variance. If given, no optimization is done.
`nugget.estim, nugget`	Should nugget be estimated? (defaults TRUE) or given values.
`noise.var`	Not implemented.
`estim.method`	Estimation criterion. `"MLE"` for Maximum-Likelihood or `"LOO"` for Leave-One-Out cross-validation.
`penalty`	Not implemented yet.
`optim.method`	Optimization algorithm used in the optimization of the objective given in `estim.method`. Supports `"BFGS"`.
`lower, upper`	Not implemented yet.
`parinit`	Initial values for the correlation ranges which will be optimized using `optim.method`.
`multistart, control, gr, iso`	Not implemented yet.
`scaling, knots, kernel,`	Not implemented yet.
`...`	Ignored.

Details

The class `"NuggetKM"` extends the `"km"` class of the **DiceKriging** package, hence has all slots of `"km"`. It also has an extra slot `"NuggetKriging"` slot which contains a copy of the original object.

Value

A NuggetKM object. See **Details**.

Author(s)

Yann Richet <yann.richet@irsn.fr>

See Also

km in the **DiceKriging** package for more details on the slots.

Examples

```
# a 16-points factorial design, and the corresponding response
d <- 2; n <- 16
design.fact <- as.matrix(expand.grid(x1 = seq(0, 1, length = 4),
                                     x2 = seq(0, 1, length = 4)))
y <- apply(design.fact, 1, DiceKriging::branin) + rnorm(nrow(design.fact))

# Using `km` from DiceKriging and a similar `NuggetKM` object
# kriging model 1 : matern5_2 covariance structure, no trend, no nugget effect
km1 <- DiceKriging::km(design = design.fact, response = y, covtype = "gauss",
                       nugget.estim=TRUE,
                       parinit = c(.5, 1), control = list(trace = FALSE))
KM1 <- NuggetKM(design = design.fact, response = y, covtype = "gauss",
          parinit = c(.5, 1))
```

NuggetKM-class *S4 class for NuggetKriging Models Extending the "km" Class*

Description

This class is intended to be used either by using its own dedicated S4 methods or by using the S4 methods inherited from the "km" class of the **libKriging** package.

Slots

`d,n,X,y,p,F` Number of (numeric) inputs, number of observations, design matrix, response vector, number of trend variables, trend matrix.

`trend.formula,trend.coef` Formula used for the trend, vector $\hat{\beta}$ of estimated (or fixed) trend coefficients with length p.

`covariance` A S4 object with class "covTensorProduct" representing a covariance kernel.

`noise.flag,noise.var` Logical flag and numeric value for an optional noise term.

`known.param` A character code indicating what parameters are known.

`lower,upper` Bounds on the correlation range parameters.

`method,penalty,optim.method,control,gr,parinit` Objects defining the estimation criterion, the optimization.

`T,M,z` Auxiliary variables (matrices and vectors) that can be used in several computations.

`case` The possible concentration (a.k.a. profiling) of the likelihood.

`param.estim` Logical. Is an estimation used?

`NuggetKriging` A copy of the `NuggetKriging` object used to create the current `NuggetKM` object.

Author(s)

Yann Richet <yann.richet@irsn.fr>

See Also

`km-class` in the **DiceKriging** package. The creator `NuggetKM`.

`NuggetKriging` *Create an object with S3 class* "NuggetKriging" *using the* **libKriging** *library.*

Description

The hyper-parameters (variance and vector of correlation ranges) are estimated thanks to the optimization of a criterion given by `objective`, using the method given in `optim`.

Usage

```
NuggetKriging(
  y = NULL,
  X = NULL,
  kernel = NULL,
  regmodel = c("constant", "linear", "interactive"),
  normalize = FALSE,
  optim = c("BFGS", "none"),
  objective = c("LL", "LMP"),
  parameters = NULL
)
```

Arguments

`y`	Numeric vector of response values.
`X`	Numeric matrix of input design.
`kernel`	Character defining the covariance model: `"exp"`, `"gauss"`, `"matern3_2"`, `"matern5_2"`.
`regmodel`	Universal NuggetKriging linear trend.
`normalize`	Logical. If `TRUE` both the input matrix `X` and the response `y` in normalized to take values in the interval $[0, 1]$.

optim	Character giving the Optimization method used to fit hyper-parameters. Possible values are: "BFGS" and "none", the later simply keeping the values given in parameters. The method "BFGS" uses the gradient of the objective.
objective	Character giving the objective function to optimize. Possible values are: "LL" for the Log-Likelihood and "LMP" for the Log-Marginal Posterior.
parameters	Initial values for the hyper-parameters. When provided this must be named list with some elements "sigma2", "theta", "nugget" containing the initial value(s) for the variance, range and nugget parameters. If theta is a matrix with more than one row, each row is used as a starting point for optimization.

Value

An object with S3 class "NuggetKriging". Should be used with its predict, simulate, update methods.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + 0.1 * rnorm(nrow(X))
## fit and print
k <- NuggetKriging(y, X, kernel = "matern3_2")
print(k)

x <- sort(c(X,as.matrix(seq(from = 0, to = 1, length.out = 101))))
p <- predict(k, x = x, stdev = TRUE, cov = FALSE)

plot(f)
points(X, y)
lines(x, p$mean, col = "blue")
polygon(c(x, rev(x)), c(p$mean - 2 * p$stdev, rev(p$mean + 2 * p$stdev)),
border = NA, col = rgb(0, 0, 1, 0.2))

s <- simulate(k, nsim = 10, seed = 123, x = x)

matlines(x, s, col = rgb(0, 0, 1, 0.2), type = "l", lty = 1)
```

---

predict,KM-method *Prediction Method for a* KM *Object*

Description

Compute predictions for the response at new given input points. These conditional mean, the conditional standard deviation and confidence limits at the 95% level. Optionnally the conditional covariance can be returned as well.

Usage

```
## S4 method for signature 'KM'
predict(
  object,
  newdata,
  type = "UK",
  se.compute = TRUE,
  cov.compute = FALSE,
  light.return = TRUE,
  bias.correct = FALSE,
  checkNames = FALSE,
  ...
)
```

Arguments

object	KM object.
newdata	Matrix of "new" input points where to perform prediction.
type	character giving the kriging type. For now only "UK" is possible.
se.compute	Logical. Should the standard error be computed?
cov.compute	Logical. Should the covariance matrix between newdata points be computed?
light.return	Logical. If TRUE, no auxiliary results will be returned (such as the Cholesky root of the correlation matrix).
bias.correct	Logical. If TRUE the UK variance and covariance are .
checkNames	Logical to check the consistency of the column names between the design stored in object@X and the new one given newdata.
...	Ignored.

Details

Without a dedicated predict method for the class "KM", this method would have been inherited from the "km" class. The dedicated method is expected to run faster. A comparison can be made by coercing a KM object to a km object with as.km before calling predict.

Value

A named list. The elements are the conditional mean and standard deviation (mean and sd), the predicted trend (trend) and the confidence limits (lower95 and upper95). Optionnally, the conditional covariance matrix is returned in cov.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
## a 16-points factorial design, and the corresponding response
d <- 2; n <- 16
design.fact <- expand.grid(x1 = seq(0, 1, length = 4), x2 = seq(0, 1, length = 4))
y <- apply(design.fact, 1, DiceKriging::branin)

## library(DiceKriging)
## kriging model 1 : matern5_2 covariance structure, no trend, no nugget
## m1 <- km(design = design.fact, response = y, covtype = "gauss",
##          parinit = c(.5, 1), control = list(trace = FALSE))
KM1 <- KM(design = design.fact, response = y, covtype = "gauss",
               parinit = c(.5, 1))
Pred <- predict(KM1, newdata = matrix(.5,ncol = 2), type = "UK",
                checkNames = FALSE, light.return = TRUE)
```

---

predict,NoiseKM-method

Prediction Method for a NoiseKM *Object*

Description

Compute predictions for the response at new given input points. These conditional mean, the conditional standard deviation and confidence limits at the 95% level. Optionnally the conditional covariance can be returned as well.

Usage

```
## S4 method for signature 'NoiseKM'
predict(
  object,
  newdata,
  type = "UK",
  se.compute = TRUE,
  cov.compute = FALSE,
  light.return = TRUE,
  bias.correct = FALSE,
  checkNames = FALSE,
  ...
)
```

Arguments

object	NoiseKM object.
newdata	Matrix of "new" input points where to perform prediction.
type	character giving the kriging type. For now only "UK" is possible.
se.compute	Logical. Should the standard error be computed?

cov.compute	Logical. Should the covariance matrix between newdata points be computed?
light.return	Logical. If TRUE, no auxiliary results will be returned (such as the Cholesky root of the correlation matrix).
bias.correct	Logical. If TRUE the UK variance and covariance are .
checkNames	Logical to check the consistency of the column names between the design stored in object@X and the new one given newdata.
...	Ignored.

Details

Without a dedicated predict method for the class "NoiseKM", this method would have been inherited from the "km" class. The dedicated method is expected to run faster. A comparison can be made by coercing a NoiseKM object to a km object with as.km before calling predict.

Value

A named list. The elements are the conditional mean and standard deviation (mean and sd), the predicted trend (trend) and the confidence limits (lower95 and upper95). Optionnally, the conditional covariance matrix is returned in cov.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
## a 16-points factorial design, and the corresponding response
d <- 2; n <- 16
design.fact <- expand.grid(x1 = seq(0, 1, length = 4), x2 = seq(0, 1, length = 4))
y <- apply(design.fact, 1, DiceKriging::branin) + rnorm(nrow(design.fact))

## library(DiceKriging)
## kriging model 1 : matern5_2 covariance structure, no trend, no nugget
## m1 <- km(design = design.fact, response = y, covtype = "gauss",
##          noise.var=rep(1,nrow(design.fact)),
##          parinit = c(.5, 1), control = list(trace = FALSE))
KM1 <- NoiseKM(design = design.fact, response = y, covtype = "gauss",
               noise=rep(1,nrow(design.fact)),
               parinit = c(.5, 1))
Pred <- predict(KM1, newdata = matrix(.5,ncol = 2), type = "UK",
                checkNames = FALSE, light.return = TRUE)
```

predict,NuggetKM-method

Prediction Method for a NuggetKM *Object*

Description

Compute predictions for the response at new given input points. These conditional mean, the conditional standard deviation and confidence limits at the 95% level. Optionnally the conditional covariance can be returned as well.

Usage

```
## S4 method for signature 'NuggetKM'
predict(
  object,
  newdata,
  type = "UK",
  se.compute = TRUE,
  cov.compute = FALSE,
  light.return = TRUE,
  bias.correct = FALSE,
  checkNames = FALSE,
  ...
)
```

Arguments

object	NuggetKM object.
newdata	Matrix of "new" input points where to perform prediction.
type	character giving the kriging type. For now only "UK" is possible.
se.compute	Logical. Should the standard error be computed?
cov.compute	Logical. Should the covariance matrix between newdata points be computed?
light.return	Logical. If TRUE, no auxiliary results will be returned (such as the Cholesky root of the correlation matrix).
bias.correct	Logical. If TRUE the UK variance and covariance are .
checkNames	Logical to check the consistency of the column names between the design stored in object@X and the new one given newdata.
...	Ignored.

Details

Without a dedicated predict method for the class "NuggetKM", this method would have been inherited from the "km" class. The dedicated method is expected to run faster. A comparison can be made by coercing a NuggetKM object to a km object with as.km before calling predict.

Value

A named list. The elements are the conditional mean and standard deviation (`mean` and `sd`), the predicted trend (`trend`) and the confidence limits (`lower95` and `upper95`). Optionnally, the conditional covariance matrix is returned in `cov`.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
## a 16-points factorial design, and the corresponding response
d <- 2; n <- 16
design.fact <- expand.grid(x1 = seq(0, 1, length = 4), x2 = seq(0, 1, length = 4))
y <- apply(design.fact, 1, DiceKriging::branin) + rnorm(nrow(design.fact))

## library(DiceKriging)
## kriging model 1 : matern5_2 covariance structure, no trend, no nugget
## m1 <- km(design = design.fact, response = y, covtype = "gauss",
##          nugget.estim=TRUE,
##          parinit = c(.5, 1), control = list(trace = FALSE))
KM1 <- NuggetKM(design = design.fact, response = y, covtype = "gauss",
                parinit = c(.5, 1))
Pred <- predict(KM1, newdata = matrix(.5,ncol = 2), type = "UK",
                checkNames = FALSE, light.return = TRUE)
```

`predict.Kriging` *Predict from a* `Kriging` *object.*

Description

Given "new" input points, the method compute the expectation, variance and (optionnally) the covariance of the corresponding stochastic process, conditional on the values at the input points used when fitting the model.

Usage

```
## S3 method for class 'Kriging'
predict(object, x, stdev = TRUE, cov = FALSE, deriv = FALSE, ...)
```

Arguments

`object`	S3 Kriging object.
`x`	Input points where the prediction must be computed.
`stdev`	`Logical`. If `TRUE` the standard deviation is returned.
`cov`	`Logical`. If `TRUE` the covariance matrix of the predictions is returned.
`deriv`	`Logical`. If `TRUE` the derivatives of mean and sd of the predictions are returned.
`...`	Ignored.

Value

A list containing the element mean and possibly stdev and cov.

Note

The names of the formal arguments differ from those of the predict methods for the S4 classes "km" and "KM". The formal x corresponds to newdata, stdev corresponds to se.compute and cov to cov.compute. These names are chosen **Python** and **Octave** interfaces to **libKriging**.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)
points(X, y, col = "blue", pch = 16)

k <- Kriging(y, X, "matern3_2")

x <-seq(from = 0, to = 1, length.out = 101)
p <- predict(k, x)

lines(x, p$mean, col = "blue")
polygon(c(x, rev(x)), c(p$mean - 2 * p$stdev, rev(p$mean + 2 * p$stdev)),
 border = NA, col = rgb(0, 0, 1, 0.2))
```

---

predict.NoiseKriging *Predict from a* NoiseKriging *object.*

Description

Given "new" input points, the method compute the expectation, variance and (optionnally) the covariance of the corresponding stochastic process, conditional on the values at the input points used when fitting the model.

Usage

```
## S3 method for class 'NoiseKriging'
predict(object, x, stdev = TRUE, cov = FALSE, deriv = FALSE, ...)
```

Arguments

object	S3 NoiseKriging object.
x	Input points where the prediction must be computed.
stdev	Logical. If TRUE the standard deviation is returned.
cov	Logical. If TRUE the covariance matrix of the predictions is returned.
deriv	Logical. If TRUE the derivatives of mean and sd of the predictions are returned.
...	Ignored.

Value

A list containing the element mean and possibly stdev and cov.

Note

The names of the formal arguments differ from those of the predict methods for the S4 classes "km" and "KM". The formal x corresponds to newdata, stdev corresponds to se.compute and cov to cov.compute. These names are chosen **Python** and **Octave** interfaces to **libKriging**.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + X/10 * rnorm(nrow(X))
points(X, y, col = "blue", pch = 16)

k <- NoiseKriging(y, (X/10)^2, X, "matern3_2")

x <-seq(from = 0, to = 1, length.out = 101)
p <- predict(k, x)

lines(x, p$mean, col = "blue")
polygon(c(x, rev(x)), c(p$mean - 2 * p$stdev, rev(p$mean + 2 * p$stdev)),
 border = NA, col = rgb(0, 0, 1, 0.2))
```

predict.NuggetKriging *Predict from a* NuggetKriging *object.*

Description

Given "new" input points, the method compute the expectation, variance and (optionnally) the covariance of the corresponding stochastic process, conditional on the values at the input points used when fitting the model.

Usage

```
## S3 method for class 'NuggetKriging'
predict(object, x, stdev = TRUE, cov = FALSE, deriv = FALSE, ...)
```

Arguments

object	S3 NuggetKriging object.
x	Input points where the prediction must be computed.
stdev	Logical. If TRUE the standard deviation is returned.
cov	Logical. If TRUE the covariance matrix of the predictions is returned.
deriv	Logical. If TRUE the derivatives of mean and sd of the predictions are returned.
...	Ignored.

Value

A list containing the element mean and possibly stdev and cov.

Note

The names of the formal arguments differ from those of the predict methods for the S4 classes "km" and "KM". The formal x corresponds to newdata, stdev corresponds to se.compute and cov to cov.compute. These names are chosen **Python** and **Octave** interfaces to **libKriging**.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + 0.1 * rnorm(nrow(X))
points(X, y, col = "blue", pch = 16)

k <- NuggetKriging(y, X, "matern3_2")
```

```
## include design points to see interpolation
x <- sort(c(X,seq(from = 0, to = 1, length.out = 101)))
p <- predict(k, x)

lines(x, p$mean, col = "blue")
polygon(c(x, rev(x)), c(p$mean - 2 * p$stdev, rev(p$mean + 2 * p$stdev)),
 border = NA, col = rgb(0, 0, 1, 0.2))
```

`print.Kriging` *Print the content of a* `Kriging` *object.*

Description

Print the content of a `Kriging` object.

Usage

```
## S3 method for class 'Kriging'
print(x, ...)
```

Arguments

`x`	A (S3) `Kriging` Object.
`...`	Ignored.

Value

String of printed object.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)

k <- Kriging(y, X, "matern3_2")

print(k)
## same thing
k
```

`print.NoiseKriging` *Print the content of a* `NoiseKriging` *object.*

Description

Print the content of a `NoiseKriging` object.

Usage

```
## S3 method for class 'NoiseKriging'
print(x, ...)
```

Arguments

`x`	A (S3) `NoiseKriging` Object.
`...`	Ignored.

Value

String of printed object.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + X/10 * rnorm(nrow(X)) # add noise dep. on X

k <- NoiseKriging(y, noise=(X/10)^2, X, kernel = "matern3_2")

print(k)
## same thing
k
```

print.NuggetKriging *Print the content of a* NuggetKriging *object.*

Description

Print the content of a NuggetKriging object.

Usage

```
## S3 method for class 'NuggetKriging'
print(x, ...)
```

Arguments

x	A (S3) NuggetKriging Object.
...	Ignored.

Value

String of printed object.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + 0.1 * rnorm(nrow(X))

k <- NuggetKriging(y, X, "matern3_2")

print(k)
## same thing
k
```

simulate,KM-method *Simulation from a* KM *Object*

Description

The simulate method is used to simulate paths from the kriging model described in object.

Usage

```
## S4 method for signature 'KM'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  newdata,
  cond = TRUE,
  nugget.sim = 0,
  checkNames = FALSE,
  ...
)
```

Arguments

object	A KM object.
nsim	Integer: number of response vectors to simulate.
seed	Random seed.
newdata	Numeric matrix with it rows giving the points where the simulation is to be performed.
cond	Logical telling wether the simulation is conditional or not. Only TRUE is accepted for now.
nugget.sim	Numeric. A postive nugget effect used to avoid numerical instability.
checkNames	Check consistency between the design data X within object and newdata. The default is FALSE. XXXY Not used!!!
...	Ignored.

Details

Without a dedicated simulate method for the class "KM", this method would have been inherited from the "km" class. The dedicated method is expected to run faster. A comparison can be made by coercing a KM object to a km object with as.km before calling simulate.

Value

A numeric matrix with nrow(newdata) rows and nsim columns containing as its columns the simulated paths at the input points given in newdata.

XXX method simulate KM

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <-  function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(5))
y <- f(X)
points(X, y, col = 'blue')
k <- KM(design = X, response = y, covtype = "gauss")
x <- seq(from = 0, to = 1, length.out = 101)
s_x <- simulate(k, nsim = 3, newdata = x)
lines(x, s_x[ , 1], col = 'blue')
lines(x, s_x[ , 2], col = 'blue')
lines(x, s_x[ , 3], col = 'blue')
```

---

simulate,NoiseKM-method

Simulation from a NoiseKM *Object*

Description

The simulate method is used to simulate paths from the kriging model described in object.

Usage

```
## S4 method for signature 'NoiseKM'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  newdata,
  cond = TRUE,
  nugget.sim = 0,
  checkNames = FALSE,
  ...
)
```

Arguments

object	A NoiseKM object.
nsim	Integer: number of response vectors to simulate.
seed	Random seed.

newdata	Numeric matrix with it rows giving the points where the simulation is to be performed.
cond	Logical telling wether the simulation is conditional or not. Only TRUE is accepted for now.
nugget.sim	Numeric. A postive nugget effect used to avoid numerical instability.
checkNames	Check consistency between the design data X within object and newdata. The default is FALSE. XXXY Not used!!!
...	Ignored.

Details

Without a dedicated simulate method for the class "NoiseKM", this method would have been inherited from the "km" class. The dedicated method is expected to run faster. A comparison can be made by coercing a NoiseKM object to a km object with as.km before calling simulate.

Value

A numeric matrix with nrow(newdata) rows and nsim columns containing as its columns the simulated paths at the input points given in newdata.

XXX method simulate NoiseKM

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <-  function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(5))
y <- f(X) + 0.01*rnorm(nrow(X))
points(X, y, col = 'blue')
k <- NoiseKM(design = X, response = y, covtype = "gauss", noise=rep(0.01^2,nrow(X)))
x <- seq(from = 0, to = 1, length.out = 101)
s_x <- simulate(k, nsim = 3, newdata = x)
lines(x, s_x[ , 1], col = 'blue')
lines(x, s_x[ , 2], col = 'blue')
lines(x, s_x[ , 3], col = 'blue')
```

simulate,NuggetKM-method

Simulation from a NuggetKM *Object*

Description

The simulate method is used to simulate paths from the kriging model described in object.

Usage

```
## S4 method for signature 'NuggetKM'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  newdata,
  cond = TRUE,
  nugget.sim = 0,
  checkNames = FALSE,
  ...
)
```

Arguments

object	A NuggetKM object.
nsim	Integer: number of response vectors to simulate.
seed	Random seed.
newdata	Numeric matrix with it rows giving the points where the simulation is to be performed.
cond	Logical telling wether the simulation is conditional or not. Only TRUE is accepted for now.
nugget.sim	Numeric. A postive nugget effect used to avoid numerical instability.
checkNames	Check consistency between the design data X within object and newdata. The default is FALSE. XXXY Not used!!!
...	Ignored.

Details

Without a dedicated simulate method for the class "NuggetKM", this method would have been inherited from the "km" class. The dedicated method is expected to run faster. A comparison can be made by coercing a NuggetKM object to a km object with as.km before calling simulate.

Value

A numeric matrix with nrow(newdata) rows and nsim columns containing as its columns the simulated paths at the input points given in newdata.

XXX method simulate NuggetKM

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <-  function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(5))
y <- f(X) + 0.01*rnorm(nrow(X))
points(X, y, col = 'blue')
k <- NuggetKM(design = X, response = y, covtype = "gauss")
x <- seq(from = 0, to = 1, length.out = 101)
s_x <- simulate(k, nsim = 3, newdata = x)
lines(x, s_x[ , 1], col = 'blue')
lines(x, s_x[ , 2], col = 'blue')
lines(x, s_x[ , 3], col = 'blue')
```

---

simulate.Kriging *Simulation from a* Kriging *model object.*

Description

This method draws paths of the stochastic process at new input points conditional on the values at the input points used in the fit.

Usage

```
## S3 method for class 'Kriging'
simulate(object, nsim = 1, seed = 123, x, ...)
```

Arguments

object	S3 Kriging object.
nsim	Number of simulations to perform.
seed	Random seed used.
x	Points in model input space where to simulate.
...	Ignored.

Value

a matrix with length(x) rows and nsim columns containing the simulated paths at the inputs points given in x.

Note

The names of the formal arguments differ from those of the `simulate` methods for the S4 classes `"km"` and `"KM"`. The formal `x` corresponds to `newdata`. These names are chosen **Python** and **Octave** interfaces to **libKriging**.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)
points(X, y, col = "blue")

k <- Kriging(y, X, kernel = "matern3_2")

x <- seq(from = 0, to = 1, length.out = 101)
s <- simulate(k, nsim = 3, x = x)

lines(x, s[ , 1], col = "blue")
lines(x, s[ , 2], col = "blue")
lines(x, s[ , 3], col = "blue")
```

`simulate.NoiseKriging` *Simulation from a* `NoiseKriging` *model object.*

Description

This method draws paths of the stochastic process at new input points conditional on the values at the input points used in the fit.

Usage

```
## S3 method for class 'NoiseKriging'
simulate(object, nsim = 1, seed = 123, x, ...)
```

Arguments

`object`	S3 NoiseKriging object.
`nsim`	Number of simulations to perform.
`seed`	Random seed used.
`x`	Points in model input space where to simulate.
`...`	Ignored.

Value

a matrix with `length(x)` rows and `nsim` columns containing the simulated paths at the inputs points given in `x`.

Note

The names of the formal arguments differ from those of the `simulate` methods for the S4 classes `"km"` and `"KM"`. The formal `x` corresponds to `newdata`. These names are chosen **Python** and **Octave** interfaces to **libKriging**.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + X/10 * rnorm(nrow(X))
points(X, y, col = "blue")

k <- NoiseKriging(y, (X/10)^2, X, kernel = "matern3_2")

x <- seq(from = 0, to = 1, length.out = 101)
s <- simulate(k, nsim = 3, x = x)

lines(x, s[ , 1], col = "blue")
lines(x, s[ , 2], col = "blue")
lines(x, s[ , 3], col = "blue")
```

---

simulate.NuggetKriging

Simulation from a `NuggetKriging` *model object.*

Description

This method draws paths of the stochastic process at new input points conditional on the values at the input points used in the fit.

Usage

```
## S3 method for class 'NuggetKriging'
simulate(object, nsim = 1, seed = 123, x, ...)
```

Arguments

object	S3 NuggetKriging object.
nsim	Number of simulations to perform.
seed	Random seed used.
x	Points in model input space where to simulate.
...	Ignored.

Value

a matrix with length(x) rows and nsim columns containing the simulated paths at the inputs points given in x.

Note

The names of the formal arguments differ from those of the simulate methods for the S4 classes "km" and "KM". The formal x corresponds to newdata. These names are chosen **Python** and **Octave** interfaces to **libKriging**.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + 0.1  *rnorm(nrow(X))
points(X, y, col = "blue")

k <- NuggetKriging(y, X, kernel = "matern3_2")

x <- seq(from = 0, to = 1, length.out = 101)
s <- simulate(k, nsim = 3, x = x)

lines(x, s[ , 1], col = "blue")
lines(x, s[ , 2], col = "blue")
lines(x, s[ , 3], col = "blue")
```

---

update,KM-method *Update a* KM *Object with New Points*

Description

The update method is used when new observations are added to a fitted kriging model. Rather than fitting the model from scratch with the updated observations added, the results of the fit as stored in object are used to achieve some savings.

Usage

```
## S4 method for signature 'KM'
update(
  object,
  newX,
  newy,
  newX.alreadyExist = FALSE,
  cov.reestim = TRUE,
  trend.reestim = cov.reestim,
  nugget.reestim = FALSE,
  newnoise.var = NULL,
  kmcontrol = NULL,
  newF = NULL,
  ...
)
```

Arguments

object	A KM object.
newX	A numeric matrix containing the new design points. It must have object@d columns in correspondence with those of the design matrix used to fit the model which is stored as object@X.
newy	A numeric vector of new response values, in correspondence with the rows of newX.
newX.alreadyExist	Logical. If TRUE, newX can contain some input points that are already in object@X.
cov.reestim	Logical. If TRUE, the vector theta of correlation ranges will be re-estimated using the new observations as well as the observations already used when fitting object. Only TRUE can be used for now.
trend.reestim	Logical. If TRUE the vector beta of trend coefficients will be re-estimated using all the observations. Only TRUE can be used for now.
nugget.reestim	Logical. If TRUE the nugget effect will be re-estimated using all the observations. Only FALSE can be used for now.
newnoise.var	Optional variance of an additional noise on the new response.
kmcontrol	A list of options to tune the fit. Not available yet.
newF	New trend matrix. XXXY?
...	Ignored.

Details

Without a dedicated update method for the class "KM", this would have been inherited from the class "km". The dedicated method is expected to run faster. A comparison can be made by coercing a KM object to a km object with as.km before calling update.

Value

The updated KM object.

Author(s)

Yann Richet <yann.richet@irsn.fr>

See Also

as.km to coerce a KM object to the class "km".

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(5))
y <- f(X)
points(X, y, col = "blue")
KMobj <- KM(design = X, response = y,covtype = "gauss")
x <-  seq(from = 0, to = 1, length.out = 101)
p_x <- predict(KMobj, x)
lines(x, p_x$mean, col = "blue")
lines(x, p_x$lower95, col = "blue")
lines(x, p_x$upper95, col = "blue")
newX <- as.matrix(runif(3))
newy <- f(newX)
points(newX, newy, col = "red")

## replace the object by its udated version
KMobj <- update(KMobj, newX=newX, newy=newy)

x <- seq(from = 0, to = 1, length.out = 101)
p2_x <- predict(KMobj, x)
lines(x, p2_x$mean, col = "red")
lines(x, p2_x$lower95, col = "red")
lines(x, p2_x$upper95, col = "red")
```

---

update,NoiseKM-method *Update a* NoiseKM *Object with New Points*

Description

The update method is used when new observations are added to a fitted kriging model. Rather than fitting the model from scratch with the updated observations added, the results of the fit as stored in object are used to achieve some savings.

Usage

```
## S4 method for signature 'NoiseKM'
update(
  object,
  newX,
  newy,
  newnoise.var,
  newX.alreadyExist = FALSE,
  cov.reestim = TRUE,
  trend.reestim = cov.reestim,
  nugget.reestim = FALSE,
  kmcontrol = NULL,
  newF = NULL,
  ...
)
```

Arguments

object	A NoiseKM object.
newX	A numeric matrix containing the new design points. It must have object@d columns in correspondence with those of the design matrix used to fit the model which is stored as object@X.
newy	A numeric vector of new response values, in correspondence with the rows of newX.
newnoise.var	Variance of an additional noise on the new response.
newX.alreadyExist	Logical. If TRUE, newX can contain some input points that are already in object@X.
cov.reestim	Logical. If TRUE, the vector theta of correlation ranges will be re-estimated using the new observations as well as the observations already used when fitting object. Only TRUE can be used for now.
trend.reestim	Logical. If TRUE the vector beta of trend coefficients will be re-estimated using all the observations. Only TRUE can be used for now.
nugget.reestim	Logical. If TRUE the nugget effect will be re-estimated using all the observations. Only FALSE can be used for now.
kmcontrol	A list of options to tune the fit. Not available yet.
newF	New trend matrix. XXXY?
...	Ignored.

Details

Without a dedicated update method for the class "NoiseKM", this would have been inherited from the class "km". The dedicated method is expected to run faster. A comparison can be made by coercing a NoiseKM object to a km object with as.km before calling update.

Value

The updated NoiseKM object.

Author(s)

Yann Richet <yann.richet@irsn.fr>

See Also

as.km to coerce a NoiseKM object to the class "km".

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(5))
y <- f(X) + 0.01*rnorm(nrow(X))
points(X, y, col = "blue")
KMobj <- NoiseKM(design = X, response = y, noise=rep(0.01^2,5), covtype = "gauss")
x <-  seq(from = 0, to = 1, length.out = 101)
p_x <- predict(KMobj, x)
lines(x, p_x$mean, col = "blue")
lines(x, p_x$lower95, col = "blue")
lines(x, p_x$upper95, col = "blue")
newX <- as.matrix(runif(3))
newy <- f(newX) + 0.01*rnorm(nrow(newX))
points(newX, newy, col = "red")

## replace the object by its udated version
KMobj <- update(KMobj, newX=newX, newy=newy, newnoise.var=rep(0.01^2,3))

x <- seq(from = 0, to = 1, length.out = 101)
p2_x <- predict(KMobj, x)
lines(x, p2_x$mean, col = "red")
lines(x, p2_x$lower95, col = "red")
lines(x, p2_x$upper95, col = "red")
```

---

update,NuggetKM-method

Update a NuggetKM *Object with New Points*

Description

The update method is used when new observations are added to a fitted kriging model. Rather than fitting the model from scratch with the updated observations added, the results of the fit as stored in object are used to achieve some savings.

Usage

```
## S4 method for signature 'NuggetKM'
update(
  object,
  newX,
  newy,
  newX.alreadyExist = FALSE,
  cov.reestim = TRUE,
  trend.reestim = cov.reestim,
  nugget.reestim = FALSE,
  newnoise.var = NULL,
  kmcontrol = NULL,
  newF = NULL,
  ...
)
```

Arguments

object	A NuggetKM object.
newX	A numeric matrix containing the new design points. It must have object@d columns in correspondence with those of the design matrix used to fit the model which is stored as object@X.
newy	A numeric vector of new response values, in correspondence with the rows of newX.
newX.alreadyExist	Logical. If TRUE, newX can contain some input points that are already in object@X.
cov.reestim	Logical. If TRUE, the vector theta of correlation ranges will be re-estimated using the new observations as well as the observations already used when fitting object. Only TRUE can be used for now.
trend.reestim	Logical. If TRUE the vector beta of trend coefficients will be re-estimated using all the observations. Only TRUE can be used for now.
nugget.reestim	Logical. If TRUE the nugget effect will be re-estimated using all the observations. Only FALSE can be used for now.
newnoise.var	Optional variance of an additional noise on the new response.
kmcontrol	A list of options to tune the fit. Not available yet.
newF	New trend matrix. XXXY?
...	Ignored.

Details

Without a dedicated update method for the class "NuggetKM", this would have been inherited from the class "km". The dedicated method is expected to run faster. A comparison can be made by coercing a NuggetKM object to a km object with as.km before calling update.

Value

The updated NuggetKM object.

Author(s)

Yann Richet <yann.richet@irsn.fr>

See Also

as.km to coerce a NuggetKM object to the class "km".

Examples

```
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(5))
y <- f(X) + 0.01*rnorm(nrow(X))
points(X, y, col = "blue")
KMobj <- NuggetKM(design = X, response = y,covtype = "gauss")
x <-  seq(from = 0, to = 1, length.out = 101)
p_x <- predict(KMobj, x)
lines(x, p_x$mean, col = "blue")
lines(x, p_x$lower95, col = "blue")
lines(x, p_x$upper95, col = "blue")
newX <- as.matrix(runif(3))
newy <- f(newX) + 0.01*rnorm(nrow(newX))
points(newX, newy, col = "red")

## replace the object by its udated version
KMobj <- update(KMobj, newX=newX, newy=newy)

x <- seq(from = 0, to = 1, length.out = 101)
p2_x <- predict(KMobj, x)
lines(x, p2_x$mean, col = "red")
lines(x, p2_x$lower95, col = "red")
lines(x, p2_x$upper95, col = "red")
```

---

update.Kriging *Update a* Kriging *model object with new points*

Description

Update a Kriging model object with new points

Usage

```
## S3 method for class 'Kriging'
update(object, newy, newX, ...)
```

Arguments

object	S3 Kriging object.
newy	Numeric vector of new responses (output).
newX	Numeric matrix of new input points.
...	Ignored.

Value

No return value. Kriging object argument is modified.

Caution

The method *does not return the updated object*, but instead changes the content of `object`. This behaviour is quite unusual in R and differs from the behaviour of the methods `update.km` in **DiceKriging** and `update,KM-method`.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1- 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x)*x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X)
points(X, y, col = "blue")

k <- Kriging(y, X, "matern3_2")

x <- seq(from = 0, to = 1, length.out = 101)
p <- predict(k, x)
lines(x, p$mean, col = "blue")
polygon(c(x, rev(x)), c(p$mean - 2 * p$stdev, rev(p$mean + 2 * p$stdev)),
 border = NA, col = rgb(0, 0, 1, 0.2))

newX <- as.matrix(runif(3))
newy <- f(newX)
points(newX, newy, col = "red")

## change the content of the object 'k'
update(k, newy, newX)

x <- seq(from = 0, to = 1, length.out = 101)
p2 <- predict(k, x)
lines(x, p2$mean, col = "red")
polygon(c(x, rev(x)), c(p2$mean - 2 * p2$stdev, rev(p2$mean + 2 * p2$stdev)),
 border = NA, col = rgb(1, 0, 0, 0.2))
```

update.NoiseKriging *Update a* NoiseKriging *model object with new points*

Description

Update a NoiseKriging model object with new points

Usage

```
## S3 method for class 'NoiseKriging'
update(object, newy, newnoise, newX, ...)
```

Arguments

object	S3 NoiseKriging object.
newy	Numeric vector of new responses (output).
newnoise	Numeric vector of new noise variances (output).
newX	Numeric matrix of new input points.
...	Ignored.

Value

No return value. NoiseKriging object argument is modified.

Caution

The method *does not return the updated object*, but instead changes the content of object. This behaviour is quite unusual in R and differs from the behaviour of the methods update.km in **DiceKriging** and update,KM-method.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1- 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x)*x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + X/10 * rnorm(nrow(X))
points(X, y, col = "blue")

k <- NoiseKriging(y, (X/10)^2, X, "matern3_2")

x <- seq(from = 0, to = 1, length.out = 101)
p <- predict(k, x)
```

```
lines(x, p$mean, col = "blue")
polygon(c(x, rev(x)), c(p$mean - 2 * p$stdev, rev(p$mean + 2 * p$stdev)),
 border = NA, col = rgb(0, 0, 1, 0.2))

newX <- as.matrix(runif(3))
newy <- f(newX) + 0.1 * rnorm(nrow(newX))
points(newX, newy, col = "red")

## change the content of the object 'k'
update(k, newy, rep(0.1^2,3), newX)

x <- seq(from = 0, to = 1, length.out = 101)
p2 <- predict(k, x)
lines(x, p2$mean, col = "red")
polygon(c(x, rev(x)), c(p2$mean - 2 * p2$stdev, rev(p2$mean + 2 * p2$stdev)),
 border = NA, col = rgb(1, 0, 0, 0.2))
```

`update.NuggetKriging` *Update a* `NuggetKriging` *model object with new points*

Description

Update a `NuggetKriging` model object with new points

Usage

```
## S3 method for class 'NuggetKriging'
update(object, newy, newX, ...)
```

Arguments

`object`	S3 NuggetKriging object.
`newy`	Numeric vector of new responses (output).
`newX`	Numeric matrix of new input points.
`...`	Ignored.

Value

No return value. NuggetKriging object argument is modified.

Caution

The method *does not return the updated object*, but instead changes the content of `object`. This behaviour is quite unusual in R and differs from the behaviour of the methods `update.km` in **DiceKriging** and `update,KM-method`.

Author(s)

Yann Richet <yann.richet@irsn.fr>

Examples

```
f <- function(x) 1- 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x)*x^5 + 0.7)
plot(f)
set.seed(123)
X <- as.matrix(runif(10))
y <- f(X) + 0.1 * rnorm(nrow(X))
points(X, y, col = "blue")

k <- NuggetKriging(y, X, "matern3_2")

## include design points to see interpolation
x <- sort(c(X,seq(from = 0, to = 1, length.out = 101)))
p <- predict(k, x)
lines(x, p$mean, col = "blue")
polygon(c(x, rev(x)), c(p$mean - 2 * p$stdev, rev(p$mean + 2 * p$stdev)),
 border = NA, col = rgb(0, 0, 1, 0.2))

newX <- as.matrix(runif(3))
newy <- f(newX) + 0.1 * rnorm(nrow(newX))
points(newX, newy, col = "red")

## change the content of the object 'k'
update(k, newy, newX)

## include design points to see interpolation
x <- sort(c(X,newX,seq(from = 0, to = 1, length.out = 101)))
p2 <- predict(k, x)
lines(x, p2$mean, col = "red")
polygon(c(x, rev(x)), c(p2$mean - 2 * p2$stdev, rev(p2$mean + 2 * p2$stdev)),
 border = NA, col = rgb(1, 0, 0, 0.2))
```

Index