Package 'constrainedKriging'

February 4, 2025

Version 0.2-11

Type Package

Title Constrained, Covariance-Matching Constrained and Universal Point or Block Kriging

Date 2025-02-04

Description Provides functions for efficient computation of non-linear spatial predictions with local change of support (Hofer, C. and Papritz, A. (2011) ``constrainedKriging: An R-package for customary, constrained and covariance-matching constrained point or block kriging'' <doi:10.1016/j.cageo.2011.02.009>). This package supplies functions for two-dimensional spatial interpolation by constrained (Cressie, N. (1993) ``Aggregation in geostatistical problems'' <doi:10.1007/978-94-011-1739-5_3>), covariance-matching constrained (Aldworth, J. and Cressie, N. (2003) ``Prediction of nonlinear spatial functionals'' <doi:10.1016/S0378-3758(02)00321-X>) and universal (external drift) Kriging for points or blocks of any shape from data with a non-stationary mean function and an isotropic weakly stationary covariance function. The linear spatial interpolation methods, constrained and covariance-matching constrained Kriging, provide approximately unbiased prediction for non-linear target values under change of support. This package extends the range of tools for spatial predictions available in R and provides an alternative to conditional simulation for non-linear spatial prediction problems with local change of support.

License GPL (>= 2)

Depends R (>= 2.9), sp(>= 0.9-60)

Imports graphics, grDevices, methods, parallel, sf(>= 1.0-14), spatialCovariance(>= 0.6-4), stats

Suggests gstat, spdep(>= 0.5-43)

NeedsCompilation yes

Author Christoph Hofer [aut], Andreas Papritz [ctb, cre]

Maintainer Andreas Papritz <papritz@retired.ethz.ch>

Repository CRAN

Date/Publication 2025-02-04 22:40:11 UTC

Contents

`constrainedKriging-package`

Nonlinear Spatial Kriging Predictions under Change of Support

---

Description

The package **constrainedKriging** provides functions for spatial interpolation by *constrained*, *covariance-matching constrained* and *universal* (*external drift*) *Kriging* for points or blocks of any shape in a two-dimensional domain from data with a non-stationary mean function and an isotropic weakly stationary variogram. The linear spatial interpolation methods constrained and covariance-matching constrained Kriging provide approximately unbiased predictions for non-linearly transformed target values under change of support.

The package provides two main user functions, `preCKrige` and `CKrige`, to calculate spatial predictions in two steps:

1. `preCKrige` computes the variance-covariance matrices for sets of prediction points or prediction blocks (polygons).
2. `CKrige` computes from the output of `preCKrige` spatial predictions by one of three Kriging methods mentioned above.

Details

The constrained Kriging predictor introduced by *Cressie (1993)* and the covariance-matching constrained Kriging predictor proposed by *Aldworth and Cressie (2003)* are linear in the data like the universal Kriging predictor. However, the constrained Kriging predictor satisfies in addition to the unbiasedness constraint of universal Kriging a second constraint that matches the variances of the predictions to the variances of the prediction targets (either point values or block means). The covariance-matching constrained Kriging predictor matches for a set of points or blocks both the variances and covariances of predictions and prediction targets and is the extended version of the constrained Kriging predictor. Like constrained Kriging, covariance-matching constrained Kriging is less biased than universal Kriging for predictions of non-linearly transformed functionals of a spatial variable and exactly unbiased if the variable is Gaussian. We summarize the formulae of the three Kriging methods below for predicting block means from point observations, but analogous equations could be given for problems that do not involve change of support.

For a set of m blocks, $B_1, \ldots, B_m$, the covariance-matching constrained Kriging predictor is given by

$$\widehat{\mathbf{Y}}_{\mathrm{CMCK}} = \mathbf{X}_m\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}} + \mathbf{K}'\mathbf{C}'\boldsymbol{\Sigma}^{-1}(\mathbf{Z} - \mathbf{X}\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}}),$$

where $\mathbf{Y} = (Y(B_1), \ldots, Y(B_m))'$ is the set of block means to be predicted, with $Y(B_i)$ the mean value of Y averaged over the block B_i; $\mathbf{Z} = (Z(\mathbf{s}_1), \ldots, Z(\mathbf{s}_n))'$ is the vector with data $Z(\mathbf{s}_i) = Y(\mathbf{s}_i) + \epsilon_i$ where the response $Y(\mathbf{s}_i)$ is possibly contaminated by measurement error ϵ_i; $\mathbf{s} = (x, y)'$ indicates a location in the survey domain; $\mathbf{X} = (\mathbf{x}(\mathbf{s}_1), \ldots, \mathbf{x}(\mathbf{s}_n))'$ is the design matrix of the data and $\mathbf{X}_m = (\mathbf{x}(B_1), \ldots, \mathbf{x}(B_m))'$ the design matrix of the target blocks; $\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}}$ is the vector with the generalised least square estimate of the linear regression coefficients; $\mathbf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_m)$ is a $(n \times m)$-matrix that contains the covariances between the n data points and the m prediction targets; $\boldsymbol{\Sigma}$ is the $(n \times n)$-covariance matrix of the data; and $\mathbf{K}$ is the $(m \times m)$-matrix

$$\mathbf{K} = \mathbf{Q}_1^{-1}\mathbf{P_1},$$

where $\mathbf{P}_1$ is the $(m \times m)$-matrix

$$\mathbf{P}_1 = (\mathrm{Cov}[\mathbf{Y}, \mathbf{Y}'] - \mathrm{Cov}[\mathbf{X}_m\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}}, (\mathbf{X}_m\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}})'])^{\frac{1}{2}}$$

and $\mathbf{Q}_1$ the $(m \times m)$-matrix

$$\mathbf{Q}_1 = (\mathrm{Cov}[\widehat{\mathbf{Y}}_{\mathrm{UK}}, \widehat{\mathbf{Y}}'_{\mathrm{UK}}] - \mathrm{Cov}[\mathbf{X}_m\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}}, (\mathbf{X}_m\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}})'])^{\frac{1}{2}}$$

with

$$\widehat{\mathbf{Y}}_{\mathrm{UK}} = \mathbf{X}_m\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}} + \mathbf{C}'\boldsymbol{\Sigma}^{-1}(\mathbf{Z} - \mathbf{X}\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}}),$$

denoting the universal Kriging predictor of $\mathbf{Y}$.

For $m = 1$ $\widehat{\mathbf{Y}}_{\mathrm{CMCK}}$ reduces to the constrained Kriging predictor

$$\widehat{Y}_{\mathrm{CK}}(B_1) = \mathbf{x}(B_1)'\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}} + K\mathbf{c_1}'\boldsymbol{\Sigma}^{-1}(\mathbf{Z} - \mathbf{X}\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}}),$$

with the scalar

$$K = (\mathrm{Var}[Y(B_1)] - \mathrm{Var}[\mathbf{x}(B_1)'\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}}])^{\frac{1}{2}}/(\mathrm{Var}[\widehat{Y}_{\mathrm{UK}}(B_1)] - \mathrm{Var}[\mathbf{x}(B_1)'\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}}])^{\frac{1}{2}} = (P/Q)^{\frac{1}{2}}.$$

The mean square prediction error (MSEP) of $\widehat{\mathbf{Y}}_{\mathrm{CMCK}}$ is given by

$$\mathrm{MSPE}[\widehat{\mathbf{Y}}_{\mathrm{CMCK}}] = \mathrm{MSPE}[\widehat{\mathbf{Y}}_{\mathrm{UK}}] + (\mathbf{P}_1 - \mathbf{Q}_1)(\mathbf{P}_1 - \mathbf{Q}_1).$$

and of $\widehat{Y}_{\mathrm{CK}}(B_1)$ by

$$\mathrm{MSPE}[\widehat{Y}_{\mathrm{CK}}(B_1)] = \mathrm{MSPE}[\widehat{Y}_{\mathrm{UK}}(B_1)] + (P^{\frac{1}{2}} - Q^{\frac{1}{2}})^2,$$

where the MSEP of universal Kriging is given by

$$\mathrm{MSPE}[\widehat{\mathbf{Y}}_{\mathrm{UK}}] = \mathrm{Cov}[\mathbf{Y}, \mathbf{Y}'] - \mathbf{C}'\boldsymbol{\Sigma}^{-1}\mathbf{C} + (\mathbf{X}'_m - \mathbf{X}'\boldsymbol{\Sigma}^{-1}\mathbf{C})'(\mathbf{X}'\boldsymbol{\Sigma}^{-1}\mathbf{X})^{-1}(\mathbf{X}'_m - \mathbf{X}'\boldsymbol{\Sigma}^{-1}\mathbf{C}).$$

Author(s)

Christoph Hofer, <christoph.hofer@alumni.ethz.ch>

References

Aldworth, J. and Cressie, N. (2003). Prediction of non-linear spatial functionals. *Journal of Statistical Planning and Inference*, **112**, 3–41, doi:10.1016/S03783758(02)00321X.

Cressie, N. (1993). Aggregation in geostatistical problems. In A. Soares, editor, *Geostatistics Troia 92*, **1**, pages 25–36, Dordrecht. Kluwer Academic Publishers, doi:10.1007/9789401117395_3.

Hofer, C. and Papritz, A. (2010). Predicting threshold exceedance by local block means in soil pollution surveys. *Mathematical Geosciences*. **42**, 631–656, doi:10.1007/s1100401092874

Hofer, C. and Papritz, A. (2011). constrainedKriging: an R-package for customary, constrained and covariance-matching constrained point or block Kriging. *Computers & Geosciences*. **37**, 1562–1569, doi:10.1016/j.cageo.2011.02.009

`ck.colors` *Colour Palette*

Description

Create a vector of *n* contiguous colours based on `hsv`.

Usage

```
ck.colors(n)
```

Arguments

`n`	a positive integer scalar with the numbers of colours in the palette.

Value

A character vector with *n* hex colour codes.

Author(s)

Christoph Hofer, <christoph.hofer@alumni.ethz.ch>

Examples

```
ck.colors(3L)
```

CKrige *Constrained, Covariance-matching Constrained and Universal Kriging*

Description

Function computes constrained, covariance-matching constrained and universal (external drift) Kriging predictions for points or blocks (of any shape) in a global neighbourhood and for isotropic covariance models.

Usage

```
CKrige(formula, data, locations, object, ...)

## S4 method for signature 'formula,data.frame,formula,preCKrigePolygons'
CKrige(formula,
  data, locations, object, method = 2, ex.out = FALSE, ncores = 1L,
  fork = !identical(.Platform[["OS.type"]], "windows"))

## S4 method for signature 'formula,data.frame,formula,preCKrigePoints'
CKrige(formula,
  data, locations, object, method = 2, ex.out = FALSE)
```

Arguments

formula	a formula for the linear regression model of the spatial variable in the form response ~ terms of covariates, for ordinary Kriging use the formula response ~ 1.
data	a data frame with the values of the covariates for the observations. The names of the covariates used in formula must match the column names of data.
locations	a one-sided formula that describes the coordinates of the observations (e.g. ~ x+y).
object	either an object of class "preCKrigePolygons" for block Kriging or of class "preCKrigePoints" for point Kriging as generated by the preCKrige function.
...	further arguments to control the spatial interpolation method and the output.
method	a numeric scalar to choose the Kriging approach: method = 1: universal (external drift), method = 2: constrained (default) and method = 3: covariance-matching constrained Kriging.
ex.out	a logical scalar. If ex.out is set equal to TRUE CKrige returns an extended output with additional items, see *Value* for more information, by default ex.out = FALSE.
ncores	a positive integer scalar with the number of CPUs to use for parallel computations.
fork	a logical scalar to control whether parallel computations are done by forking using mclapply (non-windows OSes) or by socket clusters using parLapply (windows OS).

Details

The CKrige function depends on a preCKrige output object that contains the parameters of the isotropic covariance model, the covariates of the prediction targets and the variance-covariance matrices of the prediction targets.

Value

If ex.out = FALSE then the function CKrige returns an object of class "SpatialPointsDataFrame" or "SpatialPolygonsDataFrame". The class of the argument object of CKrige ("preCKrigePoints" or "preCKrigePolygons") determines whether a SpatialPointsDataFrame or SpatialPolygonsDataFrame is returned.

Irrespective of the chosen Kriging method, the data frame in the data slot of the returned object contains the columns:

prediction	a numeric vector with the Kriging predictions by the chosen method.
prediction.se	a numeric vector with the root mean square prediction errors (Kriging standard errors).

For constrained Kriging (argument method = 2) the data frame has 3 additional columns, see constrainedKriging-package:

P1	a numeric vector with $P_1 = (\mathrm{Var}[Y(B_1)] - \mathrm{Var}[\mathbf{x}(B_1)'\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}}])^{0.5}$.
Q1	a numeric vector with $Q_1 = (\mathrm{Var}[\widehat{Y}_{\mathrm{UK}}(B_1)] - \mathrm{Var}[\mathbf{x}(B_1)'\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}}])^{0.5}$.
K	a numeric vector with $K = P_1/Q_1$.

For covariance-matching constrained Kriging (argument method = 3) the data frame has also 3 additional columns, see constrainedKriging-package:

P1.11	a numeric vector with first diagonal elements (which correspond to the target blocks) of the matrices $$\mathbf{P}_1 = (\mathrm{Cov}[\mathbf{Y}, \mathbf{Y}'] - \mathrm{Cov}[\mathbf{X}_m\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}}, (\mathbf{X}_m\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}})'])^{\frac{1}{2}}.$$
Q1.11	a numeric vector with first diagonal elements of the matrices $$\mathbf{Q}_1 = (\mathrm{Cov}[\widehat{\mathbf{Y}}_{\mathrm{UK}}, \widehat{\mathbf{Y}}'_{\mathrm{UK}}] - \mathrm{Cov}[\mathbf{X}_m\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}}, (\mathbf{X}_m\widehat{\boldsymbol{\beta}}_{\mathrm{GLS}})'])^{\frac{1}{2}}.$$
K.11	a numeric vector with first diagonal elements of the matrices of the matrices $$\mathbf{K} = \mathbf{Q}_1^{-1}\mathbf{P_1}.$$

If the argument ex.out = TRUE is used and the argument method is either equal to 1 (universal) or 2 (constrained Kriging) then the function CKrige returns an object either of class "CKrige.exout.polygons" or "CKrige.exout.points", which are lists with the following components:

object	either an object of class "SpatialPointsDataFrame" or "SpatialPolygonsDataFrame" as described above.

`krig.method` a numeric scalar, number of the chosen Kriging `method` (1, 2).

`parameter` a list with 2 components. First component `beta.coef` is the vector with the generalized least squares estimates of the linear regression coefficients, and the second component `cov.beta` contains the covariance matrix of the generalized least squares estimates.

`sk.weights` a matrix with the simple Kriging weights. The ith column contains the simple Kriging weights for the ith prediction target.

`inv.Sigma` a matrix with the inverse covariance matrix $\mathbf{\Sigma}^{-1}$ of the observations.

`residuals` a numeric vector with the generalized least squares residuals of the linear regression.

If the argument `ex.out = TRUE` is used and the argument `method` is either equal to `3` (covariance-matching constrained kriging) then the function `CKrige` returns an object either of class "`CKrige.exout.polygons`" or "`CKrige.exout.points`", which are lists with the following components:

`object` either an object of class "`SpatialPointsDataFrame`" or "`SpatialPolygonsDataFrame`" as described above.

`krig.method` a numeric scalar, number of the chosen Kriging `method` (3).

`CMCK.par` a list of 3 lists with the following components:

- `P1` a list of matrices $\mathbf{P}_1$ for all point or polygon neighbourhood configurations (the first row and columns correspond to the target points or blocks of the configurations).
- `Q1` a list of matrices $\mathbf{Q}_1$ for all point or polygon neighbourhood configurations (the first row and columns correspond to the target points or blocks of the configurations).
- `K` a list of matrices $\mathbf{K}$ for all polygon neighbourhood configurations (the first row and columns correspond to the target points or blocks of the configurations).

`parameter` a list with 2 components. First component `beta.coef` is the vector with the generalized least squares estimates of the linear regression coefficients, and the second component `cov.beta` contains the covariance matrix of the generalized least squares estimates.

`sk.weights` a list with the simple Kriging weights of all point or polygon neighbourhood configurations (the first columns contain the weights of the target points or blocks of the configurations).

`inv.Sigma` a matrix with the inverse covariance matrix $\mathbf{\Sigma}^{-1}$ of the observations.

`residuals` a numeric vector with the generalized least squares residuals of the linear regression.

Note

`print` and `summary` methods are available for `CKrige` output object of the classes "`CKrige.exout.polygons`" and "`CKrige.exout.points`".

Author(s)

Christoph Hofer, <christoph.hofer@alumni.ethz.ch>

References

Cressie, N. (2006) Block Kriging for Lognormal Spatial Processes. *Mathematical Geology*, **38**, 413–443, doi:10.1007/s1100400590228.

Hofer, C. and Papritz, A. (2011). constrainedKriging: an R-package for customary, constrained and covariance-matching constrained point or block Kriging. *Computers & Geosciences*. **37**, 1562–1569, doi:10.1016/j.cageo.2011.02.009.

See Also

preCKrige

Examples

```
### load meuse data
data(meuse, package = "sp")
data(meuse.grid, package = "sp")
data(meuse.blocks)

### convert data frame meuse.grid to SpatialPointsDataFrame
coordinates(meuse.grid) <- ~ x+y

### plot blocks along with observation locations
plot(meuse.blocks)
points(y ~ x, meuse, cex = 0.5)


### example 1: lognormal constrained block Kriging
### cf. Hofer & Papritz, 2011, pp. 1567

### compute the approximated block variance of each block in meuse.blocks
### without any neighbouring blocks (default, required for in universal
### and constrained Kriging) for an exponential covariance function without
### a measurement error, a nugget = 0.15 (micro scale white noise process),
### a partial sill variance = 0.15 and a scale parameter = 192.5
### approximation of block variance by pixels of size 75m x 75m
preCK_block <- preCKrige(newdata = meuse.blocks, model = covmodel(modelname =
    "exponential", mev = 0, nugget = 0.05, variance = 0.15,
    scale = 192.5), pwidth = 75, pheight = 75)

### block prediction by constrained Kriging on the log scale
### extended output required for backtransformation
CK_block <- CKrige(formula = log(zinc) ~ sqrt(dist), data = meuse,
  locations = ~ x+y, object = preCK_block, ex.out = TRUE)

### backtransformation of the CK predictions to original scale
### cf Hofer & Papritz, 2011, eq. (14), p. 1567
### note that Var(\hat{Y}{B}) = Var(Y(B))!
```

```
beta <- CK_block$parameter$beta.coef
M <- meuse.blocks@data$M
var.blockmeans <- unlist(preCK_block@covmat)
CK_block$object@data$Zn <- exp(CK_block$object@data$prediction
  + 0.5*(0.2 + beta[2]^2 * M - var.blockmeans))

### upper limits U of the relative 95%  prediction intervals for CK
### U multiplied by the predictions CK_block$object@data$Zn gives
### the upper limits of the 95% prediction intervals
CK_block$object@data$U <- exp(CK_block$object@data$prediction
    + 1.96 * CK_block$object@data$prediction.se) / CK_block$object@data$Zn

### plot the CK predictions of Zn and the upper limits of relative 95%
### prediction intervals by function spplot of sp package
### function ck.colors(n) creates a rainbow-like color vector
breaks <- seq(0, 1850, by = 185)
spplot(CK_block$object, zcol = "Zn", at = breaks, col.regions = ck.colors(10),
  colorkey = list(labels = list(at = breaks, labels = breaks)),
  main = "CK predictions of Zn block means")

### plot of upper limits of the relative 95% prediction intervals
breaks <- seq(1, 3.2, by = 0.2)
spplot(CK_block$object, zcol = "U", at = breaks, col.regions = ck.colors(11),
  colorkey = list(labels = list(at = breaks, labels = breaks)),
  main = "Upper limits rel. prediction intervals CK predictions")



### example 2: lognormal covariance-matching constrained block Kriging

### define neighbours by using the poly2nb function of spdep package
if(!requireNamespace("spdep", quietly = TRUE)){
  stop("install package spdep to run example")
}
neighbours_block <- spdep::poly2nb(meuse.blocks)
class(neighbours_block)
### neighbours_block should be an object of class "list"
class(neighbours_block) <- "list"

### compute the approximated block variance-covariance matrices of each
### polygon neighbourhood configuration (= target block plus its
### neighbours)
preCMCK_block <- preCKrige(newdata = meuse.blocks, neighbours = neighbours_block,
  model = covmodel("exponential", mev = 0, nugget = 0.05, variance = 0.15,
    scale = 192.5), pwidth = 75, pheight = 75)

### block prediction by covariance-matching constrained Kriging on log
### scale
CMCK_block <- CKrige(formula = log(zinc) ~ sqrt(dist), data = meuse,
  locations = ~ x+y, object = preCMCK_block, method = 3, ex.out = TRUE)

### backtransformation of the CK predictions to the original scale
### cf Hofer & Papritz, 2011, eq. (14), p. 1567
```

```
beta <- CMCK_block$parameter$beta.coef
M <- meuse.blocks@data$M
var.blockmeans <- sapply(preCMCK_block@covmat, function(x) x[1, 1])
CMCK_block$object@data$Zn <- exp(CMCK_block$object@data$prediction
  + 0.5*(0.2 + beta[2]^2 * M - var.blockmeans))

### plot the CMCK predictions of Zn by function spplot of sp package
### function ck.colors(n) creates a rainbow-like color vector
breaks <- seq(0, 1850, by = 185)
spplot(CMCK_block$object, zcol = "Zn", at = breaks, col.regions = ck.colors(10),
  colorkey = list(labels = list(at = breaks, labels = breaks)),
  main = "CMCK predictions of Zn block means")



### example 3: lognormal universal block Kriging

### block prediction by universal Kriging on log scale
UK_block <- CKrige(formula = log(zinc) ~ sqrt(dist), data = meuse,
  locations = ~ x+y, object = preCK_block, method = 1, ex.out = TRUE)

### backtransformation of the CK predictions to the original scale
### cf Hofer & Papritz, 2011, Appendix B, pp. 1568 - 1569
beta <- UK_block$parameter$beta.coef
cov.beta <- UK_block$parameter$cov.beta.coef
M <- meuse.blocks@data$M
var.blockmeans <- unlist(preCK_block@covmat)
SKw <-  UK_block$sk.weights
X <-  model.matrix(~sqrt(dist), meuse)
XB <- model.matrix(~sqrt(dist), meuse.blocks)
c1 <- 0.5 * (0.2 + beta[2]^2*M - var.blockmeans +
  UK_block$object@data$prediction.se^2)
c2 <- rowSums((XB - t(SKw) %*% X) * (XB %*% cov.beta))
UK_block$object@data$Zn <- exp(UK_block$object@data$prediction + c1 - c2)

### upper limits U of the relative 95%  prediction intervals for CK
### U multiplied by the predictions UK_block$object@data$Zn gives
### the upper limits of the 95% prediction intervals
UK_block$object@data$U <- exp(UK_block$object@data$prediction
    + 1.96 * UK_block$object@data$prediction.se) / UK_block$object@data$Zn

### plot the UK predictions of Zn by function spplot of sp package
### function ck.colors(n) creates a rainbow-like color vector
breaks <- seq(0, 1850, by = 185)
spplot(UK_block$object, zcol = "Zn", at = breaks, col.regions = ck.colors(10),
  colorkey = list(labels = list(at = breaks, labels = breaks)),
  main = "UK predictions of Zn block means")

### plot of upper limits of the relative 95% prediction intervals
breaks <- seq(1, 3.2, by = 0.2)
spplot(UK_block$object, zcol = "U", at = breaks, col.regions = ck.colors(11),
  colorkey = list(labels = list(at = breaks, labels = breaks)),
  main = "Upper limits rel. prediction intervals UK predictions")
```

```
### example 4: constrained point Kriging
### generate point CK preCKrige object for locations in meuse.grid
preCK_point <- preCKrige(newdata = meuse.grid, model = covmodel(modelname =
    "exponential", mev = 0, nugget = 0.05, variance = 0.15,
    scale = 192.5))

### point prediction by constrained Kriging on the log scale
### no extended output required for backtransformation
CK_point <- CKrige(formula = log(zinc) ~ sqrt(dist), data = meuse,
  locations = ~ x+y, object = preCK_point)

### backtransformation of the CK predictions to the original scale
### cf. Cressie, 2006, eq. (20), p. 421
### note that Var(\hat{Y}{s_0}) = Var(Y(s_0))!
CK_point@data$Zn <- exp(CK_point@data$prediction)

### convert results object to SpatialGridDataFrame
gridded(CK_point) <- TRUE
fullgrid(CK_point) <- TRUE

### plot the CK predictions of Zn by function spplot of sp package
breaks <- seq(0, 2600, by = 185)
spplot(CK_point, zcol = "Zn", at = breaks, col.regions = ck.colors(20),
  colorkey = list(labels = list(at = breaks, labels = breaks)),
  main = "CK predictions of Zn point values")



### example 5: covariance-matching constrained point Kriging
### define 4 nearest neighbours to each grid location by using the
### knearneigh function of the spdep package
if(!requireNamespace("spdep", quietly = TRUE)){
  stop("install package spdep to run example")
}
neighbours_point <- spdep::knearneigh(meuse.grid, k = 4)
### convert matrix with neighbours indices to list
neighbours_point <- apply(neighbours_point$nn, 1, FUN = function(x) x,
  simplify = FALSE)

### generate point CMCK preCKrige object for locations in meuse.grid
preCMCK_point <- preCKrige(newdata = meuse.grid, neighbours = neighbours_point,
  model = covmodel(modelname = "exponential", mev = 0, nugget = 0.05,
    variance = 0.15, scale = 192.5))

### point prediction by covariance-matching constrained Kriging on the log
### scale
### no extended output required for backtransformation
CMCK_point <- CKrige(formula = log(zinc) ~ sqrt(dist), data = meuse,
  locations = ~ x+y, object = preCMCK_point, method = 3)
```

```
### backtransformation of the CMCK predictions to the original scale
### cf. Cressie, 2006, eq. (20), p. 421
### note that Var(\hat{Y}{s_0}) = Var(Y(s_0))!
CMCK_point@data$Zn <- exp(CMCK_point@data$prediction)

### convert results object to SpatialGridDataFrame
gridded(CMCK_point) <- TRUE
fullgrid(CMCK_point) <- TRUE

### plot the CMCK predictions of Zn by function spplot of sp package
breaks <- seq(0, 2600, by = 185)
spplot(CMCK_point, zcol = "Zn", at = breaks, col.regions = ck.colors(20),
  colorkey = list(labels = list(at = breaks, labels = breaks)),
  main = "CMCK predictions of Zn point values")



### example 6: universal point Kriging
UK_point <- CKrige(formula = log(zinc) ~ sqrt(dist), data = meuse,
  locations = ~ x+y, object = preCK_point, method = 1, ex.out = TRUE)

### backtransformation of the UK predictions to the original scale cf.
### Cressie, 2006, eq.  (20), p.  421 and Hofer & Papritz, 2011, Appendix
### B, p.  1569
cov.beta <- UK_point$parameter$cov.beta.coef
SKw <- UK_point$sk.weights
X <-  model.matrix(~sqrt(dist), meuse)
Xgrid <- model.matrix(~sqrt(dist), meuse.grid)
### universal kriging variances
c1 <- 0.5 * UK_point$object@data$prediction.se^2
### \psi^' x(s_0)
c2 <- rowSums((Xgrid - t(SKw) %*% X) * (Xgrid %*% cov.beta))
UK_point$object@data$Zn <- exp(UK_point$object@data$prediction + c1 - c2)

### convert results object to SpatialGridDataFrame
gridded(UK_point$object) <- TRUE
fullgrid(UK_point$object) <- TRUE

### plot the CMCK predictions of Zn by function spplot of sp package
breaks <- seq(0, 2600, by = 185)
spplot(UK_point$object, zcol = "Zn", at = breaks, col.regions = ck.colors(20),
  colorkey = list(labels = list(at = breaks, labels = breaks)),
  main = "UK predictions of Zn point values")



### example 7: universal block Kriging of mean over whole domain
### cf chapter 4 of vignette of package georob
### (https://CRAN.R-project.org/package=georob)
if(!requireNamespace("gstat", quietly = TRUE)){
  stop("install package gstat to run example")
}
```

```
### load coalash data
data(coalash, package="gstat")

### generate SpatialPointsDataFrame covering entire domain of coalash data
coalash.domain <- rbind(c(0.5,0), c(16.5,0), c(16.5,24), c(0.5,24), c(0.5,0))
coalash.domain <- SpatialPolygonsDataFrame(
  SpatialPolygons(list(Polygons(list(Polygon(coalash.domain)), ID= "domain"))),
  data=data.frame(x=8.5,y=12,row.names="domain"))

### universal block Kriging
preCK_coalash <- preCKrige(newdata = coalash.domain, model = covmodel(modelname =
    "exponential", mev = 1.023, nugget = 0, variance = 0.268,
    scale = 1.907), pwidth = 16, pheight = 24)
UK_coalash <- CKrige(formula = coalash ~ x, data = coalash,
  locations = ~ x+y, object = preCK_coalash, method = 1)
slot( UK_coalash, "data")
```

`covmodel` *Create isotropic covariance model*

Description

Function to generate isotropic covariance models or to add an isotropic covariance model to an existing isotropic model.

Usage

```
covmodel(modelname, mev, nugget,variance, scale, parameter, add.covmodel)

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

Arguments

`modelname`	a character scalar with the name of an isotropic covariance model, see *Details* for a list of implemented models. A call of `covmodel()` without any function argument displays a table with all available models and their parameters, see *Examples*.
`mev`	a numeric scalar, variance of the measurement error.
`nugget`	a numeric scalar, variance of microstructure white noise process with range smaller than the minimal distance between any pair of support data.
`variance`	a numeric scalar, partial sill of the covariance model.
`scale`	a numeric scalar, scale ("range") parameter of the covariance model.
`parameter`	a numeric vector of further covariance parameters, missing for some model like `nugget`, `spherical` or `gauss`, etc, see *Details*. If a model has several extra parameters, say `a`, `b`, ... then they must be given as `c(a, b, ...)`.

add.covmodel	an object of the class covmodel that is added to the covariance model defined by modelname (see examples)
x	a covariance model generated by covmodel
...	further printing arguments

Details

The name and parametrisation of the covariance models originate from the function CovarianceFct of the archived package **RandomFields**, version 2.0.71.

The following isotropic covariance functions are implemented (equations taken from help page of function CovarianceFct of archived package **RandomFields**, version 2.0.71, note that the variance and range parameters are equal to 1 in the following formulae and h is the lag distance.):

- bessel
$$C(h) = 2^a \Gamma(a+1) h^{-a} J_a(h)$$
For a 2-dimensional region, the parameter a must be greater than or equal to 0.
- cauchy
$$C(h) = \left(1 + h^2\right)^{-a}$$
The parameter a must be positive.
- cauchytbm
$$C(h) = (1 + (1 - b/3)h^a)(1 + h^a)^{(-b/a-1)}$$
The parameter a must be in (0,2] and b positive. The model is valid for 3 dimensions. It allows for simulating random fields where fractal dimension and Hurst coefficient can be chosen independently.
- circular
$$C(h) = \left(1 - \frac{2}{\pi}\left(h\sqrt{1-h^2} + \arcsin(h)\right)\right) 1_{[0,1]}(h)$$
This isotropic covariance function is valid only for dimensions less than or equal to 2.
- constant
$$C(h) = 1$$
- cubic
$$C(h) = (1 - 7h^2 + 8.75h^3 - 3.5h^5 + 0.75h^7)1_{[0,1]}(h)$$
This model is valid only for dimensions less than or equal to 3. It is a 2 times differentiable covariance functions with compact support.
- dampedcosine (hole effect model)
$$C(h) = e^{-ah}\cos(h)$$
This model is valid for 2 dimensions iff $a \geq 1$.
- exponential
$$C(h) = e^{-h}$$
This model is a special case of the whittle model (for $a = 0.5$) and the stable model (for $a = 1$).

- gauss

$$C(h) = e^{-h^2}$$

This model is a special case of the stable model (for $a = 2$). See gneiting for an alternative model that does not have the disadvantages of the Gaussian model.

- gencauchy (generalised cauchy)

$$C(h) = (1 + h^a)^{(-b/a)}$$

The parameter a must be in (0,2] and b positive. This model allows for random fields where fractal dimension and Hurst coefficient can be chosen independently.

- gengneiting (generalised gneiting) If $a = 1$ and let $\beta = b + 1$ then

$$C(h) = (1 + \beta h)\,(1 - h)^{\beta} 1_{[0,1]}(h)$$

If $a = 2$ and let $\beta = b + 2$ then

$$C(h) = \left(1 + \beta h + \left(\beta^2 - 1\right) h^2/3\right)(1 - h)^{\beta} 1_{[0,1]}(h)$$

If $a = 3$ and let $\beta = b + 3$ then

$$C(h) = \left(1 + \beta h + \left(2\beta^2 - 3\right)\frac{h^2}{5} + \left(\beta^2 - 4\right)\beta\frac{h^3}{15}\right)(1 - h)^{\beta} 1_{[0,1]}(h)$$

The parameter a is a positive integer; here only the cases $a = 1, 2, 3$ are implemented. For two dimensional regions the parameter b must greater than or equal to $(2 + 2a + 1)/2$.

- gneiting

$$C(h) = \left(1 + 8sh + 25(sh)^2 + 32(sh)^3\right)(1 - sh)^8 1_{[0,1]}(sh)$$

where $s = 0.301187465825$. This covariance function is valid only for dimensions less than or equal to 3. It is a 6 times differentiable covariance functions with compact support. It is an alternative to the gaussian model since its graph is visually hardly distinguishable from the graph of the Gaussian model, but possesses neither the mathematical and nor the numerical disadvantages of the Gaussian model.

- hyperbolic

$$C(h) = c^{-b}(K_b(ac))^{-1}(c^2 + h^2)^{b/2} K_b(a[c^2 + h^2]^{1/2})$$

The parameters are such that
$c \geq 0$, $a > 0$ and $b > 0$, or
$c > 0$, $a > 0$ and $b = 0$, or
$c > 0$, $a \geq 0$, and $b < 0$.
Note that this class is over-parametrised; always one of the three parameters a, c, and scale can be eliminated in the formula.

- lgd1 (local-global distinguisher)

$$C(h) = 1 - \frac{b}{a+b}h^a, h \leq 1 \qquad \text{and} \qquad \frac{a}{a+b}h^{-b}, h > 1$$

Here $b > 0$ and a msut be in $(0, 0.5]$. The random field has for 2-dimensional regions fractal dimension $3 - a/2$ and Hurst coefficient $1 - b/2$ for $b \in (0, 1]$

- `matern`
$$C(h) = 2^{1-a}\Gamma(a)^{-1}(\sqrt{2a}h)^a K_a(\sqrt{2a}h)$$
The parameter a must be positive. This is the model of choice if the smoothness of a random field is to be parametrised: if $a > m$ then the graph is m times differentiable.
- `nugget`
$$C(h) = 1_{[0]}(h)$$
- `penta`
$$C(h) = \left(1 - \frac{22}{3}h^2 + 33h^4 - \frac{77}{2}h^5 + \frac{33}{2}h^7 - \frac{11}{2}h^9 + \frac{5}{6}h^{11}\right) 1_{[0,1]}(h)$$
valid only for dimensions less than or equal to 3. This is a 4 times differentiable covariance functions with compact support.
- `power`
$$C(h) = (1-h)^a 1_{[0,1]}(h)$$
This covariance function is valid for 2 dimensions iff $a \geq 1.5$. For $a = 1$ we get the well-known triangle (or tent) model, which is valid on the real line, only.
- `qexponential`
$$C(h) = (2e^{-h} - ae^{-2x})/(2-a)$$
The parameter a must be in $[0, 1]$.
- `spherical`
$$C(h) = \left(1 - 1.5h + 0.5h^3\right) 1_{[0,1]}(h)$$
This covariance function is valid only for dimensions less than or equal to 3.
- `stable`
$$C(h) = \exp\left(-h^a\right)$$
The parameter a must be in $(0, 2]$. See `exponential` and `gaussian` for special cases.
- `wave`
$$C(h) = \frac{\sin h}{h}, \quad h > 0 \qquad \text{and} \qquad C(0) = 1$$
This isotropic covariance function is valid only for dimensions less than or equal to 3. It is a special case of the `bessel` model (for $a = 0.5$).
- `whittle`
$$C(h) = 2^{1-a}\Gamma(a)^{-1}h^a K_a(h)$$
The parameter a must be positive. This is the model of choice if the smoothness of a random field is to be parametrised: if $a > m$ then the graph is m times differentiable.

The default values of the arguments `mev`, `nugget`, `variance` and `scale` are eq 0.

Value

an object of the class `covmodel` that defines a covariance model.

Author(s)

Christoph Hofer, <christoph.hofer@alumni.ethz.ch>

Examples

```
# table with all available covariance models and their
# parameters
covmodel()

# exponential model without a measurement error and without a nugget,
# partial sill = 10, scale  parameter = 15
covmodel(modelname = "exponential", variance = 10, scale = 15)

# exponential model with a measurement error ( mev = 0.5) and a
# nugget (nugget = 2.1), exponential partial  sill (variance = 10)
# and scale parameter = 15
covmodel(modelname = "exponential", mev  = 0.5, nugget = 2.1,
variance = 10, scale = 15)
```

---

meuse.blocks *Meuse block*

Description

meuse.blocks is an object of the class "SpatialPolygonsDataFrame" that contains the coordinates of 259 artificially defined (mostly square) blocks obtained by intersecting a grid with 150 m mesh width over the the flood plain area of the river Meuse, near the village Stein. The 259 x 2 data frame contains the covariate dist and the attribute M (spatial variance of sqrt(dist)) for each block.

Usage

```
data(meuse.blocks)
```

Format

The object contains the following slots:

data a 259 x 2 data frame contains:

- **dist** mean Euclidean distance of the blocks from the river, normalized to the interval [0;1].
- **M** the (spatial) variance of sqrt(dist) within the blocks, see *Hofer & Papritz (2011)*.

polygons an object of the class SpatialPolygons that contains the coordinates of the 259 blocks, see SpatialPolygons.

plotOrder see SpatialPolygons.

bbox see SpatialPolygons.

proj4string see SpatialPolygons.

Author(s)

Christoph Hofer, <christoph.hofer@alumni.ethz.ch>

References

Hofer, C. and Papritz, A. (2011). constrainedKriging: an R-package for customary, constrained and covariance-matching constrained point or block Kriging. *Computers & Geosciences*. **37**, 1562–1569, doi:10.1016/j.cageo.2011.02.009.

See Also

preCKrige and CKrige

Examples

```
data(meuse.blocks)
summary(meuse.blocks)
### show the shape of the 259 blocks
plot(meuse.blocks)
### plot maps of the covariate dist and attribute M
spplot(meuse.blocks)
```

---

plot.preCKrigePolygons

Plotting a Polygon Neighbourhood Configuration

Description

Plotting method for objects of the class "preCKrige.polygons". The plot shows the polygon neighbourhood configuration for one polygon (block) in a preCKrige.polygons object as well as its representation by the pixels.

Usage

```
## S3 method for class 'preCKrigePolygons'
plot(x, index, ...)
```

Arguments

x	an object of the class "preCKrigePolygons". In general the output object of a preCKrige function call.
index	a numeric scalar with the index of the desired polygon (block) in the list of polygons x@polygons.
...	further plotting parameters.

Value

No return value, called for side effects.

Author(s)

Christoph Hofer, <christoph.hofer@alumni.ethz.ch>

References

Hofer, C. and Papritz, A. (2011). constrainedKriging: an R-package for customary, constrained and covariance-matching constrained point or block Kriging. *Computers & Geosciences*. **37**, 1562–1569, doi:10.1016/j.cageo.2011.02.009.

See Also

preCKrige and preCKrigePolygons.

Examples

```
### load data
data(meuse, package = "sp")
data(meuse.blocks)

### plot blocks
plot(meuse.blocks)

### compute the approximated block variance of each block in
### meuse.blocks without the definition of neighbours blocks (default)
preCK_1  <- preCKrige(newdata = meuse.blocks,
    model = covmodel("exponential", 0.05, 0.15, scale = 192.5),
    pwidth = 75, pheight = 75)

### plot block approximation of block 59
plot(preCK_1, 59)


### define neighbours
if(!requireNamespace("spdep", quietly = TRUE)){
  stop("install package spdep to run example")
}
neighbours <- spdep::poly2nb(meuse.blocks)
class(neighbours)
### neighbours should be an object of the class "list"
class(neighbours) <- "list"
### compute the approximated block variance-covariance matrices of each block in
### meuse.blocks without the defined block neighbours
preCK_2 <- preCKrige(newdata = meuse.blocks, neighbours = neighbours,
    model = covmodel("exponential", 0.05, 0.15, scale = 192.5),
    pwidth = 75, pheight = 75)

### plot block approximation of block 59 and its
### block neighbours
plot(preCK_2, 59)
```

preCKrige *Spatial Variance-Covariance Matrices for Sets of Points and Polygons*

Description

The function preCKrige computes (approximated) spatial variance-covariance matrices for user-defined sets of points or polygons (blocks) of any shape for two-dimensional isotropic random fields. The areas of a set of polygons (polygon neighbourhood configuration) are approximated by pixels and the block-block covariances are approximated by averaging covariances between the pixels used to approximate the polygons.

The object returned by preCKrige is needed by CKrige for computing spatial point or block predictions by constrained, covariance-matching constrained or universal (external drift) Kriging.

Usage

```
preCKrige(newdata, neighbours, model, ...)
## S4 method for signature 'SpatialPoints,ANY,covmodel'
preCKrige(newdata, neighbours, model)
## S4 method for signature 'SpatialPointsDataFrame,ANY,covmodel'
preCKrige(newdata, neighbours, model)
## S4 method for signature 'SpatialPolygons,ANY,covmodel'
preCKrige(newdata, neighbours, model,
  pwidth = 0, pheight = 0, napp = 1, ncores = 1L,
  fork = !identical( .Platform[["OS.type"]], "windows"))
## S4 method for signature 'SpatialPolygonsDataFrame,ANY,covmodel'
preCKrige(newdata, neighbours,
  model, pwidth = 0, pheight = 0, napp = 1, ncores = 1L,
  fork = !identical( .Platform[["OS.type"]], "windows"))
```

Arguments

newdata	either an object of the class "SpatialPointsDataFrame" or "SpatialPoints" that contains the coordinates of the prediction points and optionally additional information (covariates) stored in the data slot of the SpatialPointsDataFrame, or an object of the class "SpatialPolygonsDataFrame" or "SpatialPolygons" with the coordinates of the polygons (blocks) for which predictions are computed and optionally additional information (covariates) stored in the data slot of the SpatialPolygonsDataFrame.
neighbours	a list of length n with integer vectors as components. n is equal to the number of points if newdata is an object of class "SpatialPointsDataFrame" or "SpatialPoints" or equal to number of polygons (blocks) if newdata is an object of class "SpatialPolygonsDataFrame" or "SpatialPolygons". The ith list component defines the neighbours of the ith point or ith polygon (block) in newdata, which form jointly with the ith point or polygon the so-called *point* or *polygon neighbourhood configuration*. If newdata is an object of

class "SpatialPolygonsDataFrame" or "SpatialPolygons" the *i*th list component contains the indices of the neighbouring polygons for the *i*th polygon. If newdata is an object of class "SpatialPoints" or "SpatialPointsDataFrame" the *i*th list component contains the row indices of the neighbouring points in the point coordinate matrix. The *i*th list component is set to integer(0) if the *i*th polygon or *i*th point have no (defined) neighbours. By default, the points or polygons have no neighbours.

See the second example below where the function poly2nb of the package **spdep** is used to build a list of neighbours for target polygons of the data set meuse.blocks.

model — an object of class "covmodel". The object contains the parameters of the isotropic covariance function, generated by the function covmodel.

... — further arguments if newdata is of class "SpatialPolygonsDataFrame" or "SpatialPolygons".

pwidth — a positive numeric scalar, defines the width of the pixels used to approximate the polygon (block) areas.

pheight — a positive numeric scalar, defines the height of the pixels used to approximate the polygon (block) areas.

napp — a positive integer scalar. napp > 1 reduces the block-block variance-covariance approximation error. By default, napp = 1, see *Details*.

ncores — a positive integer scalar with the number of CPUs to use for parallel computations.

fork — a logical scalar to control whether parallel computations are done by forking using mclapply (non-windows OSes) or by socket clusters using parLapply (windows OS).

Details

If the object newdata is of class "SpatialPolygonsDataFrame" or "SpatialPolygons" then preCKrige searches the polygon neighbourhood configuration (defined by neighbours) with the largest bounding box and generates a pixel grid that completely covers the largest bounding box. Subsequently, the covariance matrix of this set of pixels is calculated by the **spatialCovariance** package and the polygon (block) areas of each polygon neighbourhood configuration are approximated by intersecting the polygons with the shifted pixel grid, which yields a pixel representation of the polygon neighbourhood configuration. Finally, the block-block covariances of the polygons are approximated by averaging the covariances of the pixel representation of the polygon neighbourhood configuration.

By default, napp = 1, which means that the approximation of the block-block variance-covariance matrix for each polygon neighbourhood configuration is computed just once. If napp > 1 the approximation of the block-block variance-covariance matrix for one polygon neighbourhood configuration is based on the mean of napp repetitions of the approximation to reduce the approximation error. Each of the napp block-block variance-covariance approximations are based on a new, randomly shifted pixel gird which results each time in a new pixel representation of the polygon neighbourhood configuration. Large values of the argument napp increases the computation time.

There is a plot method plot.preCKrigePolygons for preCKrige output objects of class "preCKrigePolygons" to visually control the polygon (block) area approximation by the pixels.

Value

preCKrige returns an S4 object, either of class "preCKrigePolygons" if newdata is of class "SpatialPolygons" or "SpatialPolygonsDataFrame" or an S4 object of class "preCKrigePoints" if newdata is of class "SpatialPoints" or "SpatialPointsDataFrame".

Notation:

n	number of polygons or points in newdata, $i = 1, \ldots, n$
m_i	size of point or polygon neighbourhood configuration m_i = 1 + number of (defined) neighbours of the ith point or ith polygon
r_{pix}	number of pixel grid rows
c_{pix}	number of pixel grid columns
n_{pix}	number of pixels in pixel grid $n_{\text{pix}} = r_{\text{pix}} \cdot c_{\text{pix}}$

An object of class "preCKrigePoints" contains the following slots:

covmat	a list of length n, the ith list component contains the point-point covariance matrix of the ith prediction point and its neighbours, i.e. of the ith point neighbourhood configuration.
posindex	a list of length n, the ith list component contains a vector with the row indices of the $m_i - 1$ neighbours in the ith point neighbourhood configuration.
model	an object of class "covmodel" with the parameters of the used covariance function.
data	a data frame, which is the data slot of the SpatialPointsDataFrame object. This data frame is used to build the design matrix of the prediction points by the CKrige function. data is empty with dim(data) = (0, 0) if newdata is an object of class "SpatialPoints".
coords	a matrix with dim(coords) = (n, 2) with the coordinates of the prediction points.

An object of class "preCKrigePolygons" contains the following slots:

covmat	a list of length n, the ith list component contains the approximated block-block covariance matrix of the ith polygon and its neighbours, i.e. of the ith polygon neighbourhood configuration.
se.covmat	a list of length n, the ith list component contains a matrix with the standard errors of the approximated block-block covariances of the ith polygon neighbourhood configuration. Values are equal to NaN for argument napp = 1, see *Details*.
pixconfig	a list of lists of length n, the ith list component contains a list with the information about the pixels used for the covariance approximation of the ith polygon neighbourhood configuration. The components of pixconfig are described below.
pixcovmat	a matrix, dim(matrix) = (n_{pix}, n_{pix}), with the covariance matrix of the pixels.
model	an object of class "covmodel" with the parameters of the used covariance function.

`data` a data frame which is the `data` slot of the `SpatialPolygonsDataFrame` object. This data frame is used to build the design matrix of the prediction polygons by the `CKrige` function. `data` is empty with `dim(data)` = (0, 0) if `newdata` is an object of class "`SpatialPolygons`".

`polygons` a `SpatialPolygons` object. A list of length n with the polygons of the `newdata` object.

The ith component of `pixconfig` is a list with the following 10 components:

`pixcenter` a matrix with `dim(pixcenter)` = (n_{pix}, 2) with the coordinates of the pixels centroids for the ith polygon neighbourhood configuration.

`rowwidth` `preCKrige` input argument `pheight`.

`colwidth` `preCKrige` input argument `pwidth`.

`nrows` a numeric scalar with number of rows r_{pix} of the pixel grid.

`ncols` a numeric scalar with number of columns c_{pix} of the pixel grid.

`no.pix.in.poly` a numeric vector of length m_i, each number indicates by how many pixels a polygon of the ith polygon configuration is approximated.

`sa.polygons` a logical vector of length m_i, TRUE means that the ith polygon is treated as a point because its area is smaller than the area of a pixel, and FALSE means that the polygon is approximated by pixels, see *Note* for more details.

`polygon.centroids` a matrix with `dim(polygon.centroids)` = (m_i, 2) with the coordinates of the polygon centroids of the ith polygon neighbourhood configuration.

`posindex` an integer vector of length m_i with indices of the ith polygon and its neighbours as defined by the argument `neighbours`.

`pix.in.poly` is a binary matrix with `dim(pix.in.poly)` = (n_{pix}, m_i). `pix.in.poly[k, j]` = `1` indicates that the centroid of the kth pixel lies in the jth polygon, and `pix.in.poly[k, j]` = `0` indicates that the kth pixel centroid does not lie in the jth polygon.

Note

A polygon (block) is treated as point if the polygon area is smaller than the (defined) pixel area or if all pixel centroids of the generated pixel grid lie outside the polygon (block) area. If a pixel centroid lies inside a polygon that has a smaller area than a pixel, the pixel is allocated to the polygon (block) by which it shares the largest area.

The point-point correlations are calculated via the internal function `CorrelationFct` (this function implements a subset of the covariance models available previously in the function `CovarianceFct` of the archived package **RandomFields**, version 2.0.71) and the point-block covariances are calculated by the C function `PointRectCov` of the package.

Author(s)

Christoph Hofer, <christoph.hofer@alumni.ethz.ch>

References

Hofer, C. and Papritz, A. (2011). constrainedKriging: an R-package for customary, constrained and covariance-matching constrained point or block Kriging. *Computers & Geosciences*. **37**, 1562–1569, doi:10.1016/j.cageo.2011.02.009

See Also

CKrige

Examples

```
### first example
### load data
data(meuse, package = "sp")
data(meuse.blocks)

### plot blocks
plot(meuse.blocks)

### compute the approximated block variance of each block in meuse.blocks
### without any neighbouring blocks (default, required for in universal
### and constrained Kriging) for an exponential covariance function without
### a measurement error, a nugget  = 0.15 (micro scale white noise process),
### a partial sill variance = 0.15 and a scale parameter = 192.5
### approximation of block variance by pixel of size 75m x 75m
preCK_1 <- preCKrige(newdata = meuse.blocks, model = covmodel(modelname =
    "exponential", mev = 0, nugget = 0.05, variance = 0.15,
    scale = 192.5), pwidth = 75, pheight = 75)

### plot block approximation for block 59
plot(preCK_1, 59)

### second example
### define neighbours by using the poly2nb function
### of the spdep package
if(!requireNamespace("spdep", quietly = TRUE)){
  stop("install package spdep to run example")
}
neighbours <- spdep::poly2nb(meuse.blocks)
class(neighbours)
### neighbours should be an object of class "list"
class(neighbours) <- "list"
### compute the approximated block variance-covariance
### matrices of each block in meuse.blocks without the
### defined block neighbours
preCK_2 <- preCKrige(newdata = meuse.blocks, neighbours = neighbours,
  model = covmodel("exponential", nugget = 0.05, variance = 0.15,
    scale = 192.5), pwidth = 75, pheight = 75)

### plot block approximation of block 59 and its
### block neighbours
plot(preCK_2, 59)
```

preCKrigePoints-class *Class "preCKrigePoints"*

Description

Class of objects that are generated by preCKrige if the attribute newdata is of class "SpatialPoints" or "SpatialPointsDataFrame". This class has a show, summary and a CKrige method.

Objects from the Class

Objects can be created by calls of the generic function preCKrige.

Slots

covmat: Object of class "list", see preCKrige, section *Value*.

posindex: Object of class "list", see preCKrige, section *Value*.

model: Object of class "list", see preCKrige, section *Value*.

data: Object of class "data.frame", see preCKrige, section *Value*.

coords: Object of class "matrix", see preCKrige, section *Value*.

Methods

CKrige `signature(formula = "formula", data = "data.frame", locations = "formula", object = "preCKrigePoints", method = "numeric", ex.out = "logical")`

Author(s)

Christoph Hofer, <christoph.hofer@alumni.ethz.ch>

See Also

preCKrige, preCKrigePolygons

Examples

```
showClass("preCKrigePoints")
```

preCKrigePolygons-class *Class preCKrigePolygons*

Description

Class of objects that are generated by preCKrige if the attribute newdata is of the class "SpatialPolygons" or "SpatialPolygonsDataFrame". This class has a show, summary and a CKrige method.

Objects from the Class

Objects can be created by calls of the generic function preCKrige.

Slots

covmat: Object of class "list", see preCKrige, section *Value*.

se.covmat: Object of class "list", see preCKrige, section *Value*.

pixconfig: Object of class "list", see preCKrige, section *Value*.

pixcovmat: Object of class "matrix", see preCKrige, section *Value*.

model: Object of class "list", see preCKrige, section *Value*.

data: Object of class "data.frame", see preCKrige, section *Value*.

polygons: Object of class "list", see preCKrige, section *Value*.

Methods

CKrige signature(formula = "formula", data = "data.frame", locations = "formula", object = "preCKrigePolygons", method = "numeric", ex.out = "logical"): ...

Author(s)

Christoph Hofer, <christoph.hofer@alumni.ethz.ch>

See Also

preCKrige, preCKrigePoints

Examples

```
showClass("preCKrigePolygons")
```

Index