The geoR Package

April 12, 2007

Version 1.6-14

Date 2007/04/11

Title Analysis of geostatistical data

Author Paulo J. Ribeiro Jr <paulojus@est.ufpr.br> and Peter J. Diggle <p.diggle@lancaster.ac.uk>

Maintainer Paulo J. Ribeiro Jr <paulojus@ufpr.br>

Depends R (>= 2.2.0), stats, sp

Suggests MASS,scatterplot3d,RandomFields,methods

Description Geostatistical analysis including traditional, likelihood-based and Bayesian methods.

License GPL Version 2 or later

URL http://www.leg.ufpr.br/geoR

R topics documented:

`boxcox` *The Box-Cox Transformed Normal Distribution*

Description

Functions related with the Box-Cox family of transformations. Density and random generation for the Box-Cox transformed normal distribution with mean equal to `mean` and standard deviation equal to `sd`, *in the normal scale*.

Usage

```
rboxcox(n, lambda, lambda2 = NULL, mean = 0, sd = 1)

dboxcox(x, lambda, lambda2 = NULL, mean = 0, sd = 1)
```

Arguments

`lambda`	numerical value(s) for the transformation parameter λ.
`lambda2`	logical or numerical value(s) of the additional transformation (see DETAILS below). Defaults to `NULL`.
`n`	number of observations to be generated.
`x`	a vector of quantiles (`dboxcox`) or an output of `boxcox.fit` (`print`, `plot`, `lines`).
`mean`	a vector of mean values at the normal scale.
`sd`	a vector of standard deviations at the normal scale.

Details

Denote Y the variable at the original scale and Y' the transformed variable. The Box-Cox transformation is defined by:

$$Y' = \begin{cases} log(Y) \text{ , if } \lambda = 0 \\ \frac{Y^{\lambda}-1}{\lambda} \text{ , otherwise} \end{cases}$$

.

An additional shifting parameter λ_2 can be included in which case the transformation is given by:

$$Y' = \begin{cases} log(Y+\lambda_2) \text{ , } \lambda = 0 \\ \frac{(Y+\lambda_2)^{\lambda}-1}{\lambda} \text{ , otherwise} \end{cases}$$

.

The function `rboxcox` samples Y' from the normal distribution using the function `rnorm` and backtransform the values according to the equations above to obtain values of Y. If necessary the back-transformation truncates the values such that $Y' \geq \frac{1}{\lambda}$ results in $Y = 0$ in the original scale. Increasing the value of the mean and/or reducing the variance might help to avoid truncation.

Value

The functions returns the following results:

`rboxcox`	a vector of random deviates.
`dboxcox`	a vector of densities.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Box, G.E.P. and Cox, D.R.(1964) An analysis of transformations. JRSS B **26**:211–246.

See Also

The parameter estimation function `boxcox.fit`, the function `boxcox` in the package **MASS** and the function `boxcox` in the package **car**.

Examples

```
## Simulating data
simul <- rboxcox(100, lambda=0.5, mean=10, sd=2)
##
## Comparing models with different lambdas,
## zero  means and unit variances
curve(dboxcox(x, lambda=-1), 0, 8)
for(lambda in seq(-.5, 1.5, by=0.5))
  curve(dboxcox(x, lambda), 0, 8, add = TRUE)
```

InvChisquare *The (Scaled) Inverse Chi-Squared Distribution*

Description

Density and random generation for the scaled inverse chi-squared (χ^2_{ScI}) distribution with `df` degrees of freedom and optional non-centrality parameter `scale`.

Usage

```
dinvchisq(x, df, scale, log = FALSE)
rinvchisq(n, df, scale = 1/df)
```

Arguments

`x`	vector of quantiles.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`df`	degrees of freedom.
`scale`	scale parameter.
`log`	logical; if TRUE, densities d are given as log(d).

Details

The inverse chi-squared distribution with `df`$= n$ degrees of freedom has density

$$f(x) = \frac{1}{2^{n/2}\Gamma(n/2)}(1/x)^{n/2+1}e^{-1/(2x)}$$

for $x > 0$. The mean and variance are $\frac{1}{(n-2)}$ and $\frac{2}{(n-4)(n-2)^2}$.

The non-central chi-squared distribution with `df`$= n$ degrees of freedom and non-centrality parameter `scale` $= S^2$ has density

$$f(x) = \frac{n/2^{n/2}}{\Gamma(n/2)} S^n (1/x)^{n/2+1} e^{-(nS^2)/(2x)}$$

, for $x \geq 0$. The first is a particular case of the latter for $\lambda = n/2$.

Value

`dinvchisq` gives the density and `rinvchisq` generates random deviates.

See Also

`rchisq` for the chi-squared distribution which is the basis for this function.

Examples

```
set.seed(1234); rinvchisq(5, df=2)
set.seed(1234); 1/rchisq(5, df=2)

set.seed(1234); rinvchisq(5, df=2, scale=5)
set.seed(1234); 5*2/rchisq(5, df=2)

## inverse Chi-squared is a particular case
x <- 1:10
all.equal(dinvchisq(x, df=2), dinvchisq(x, df=2, scale=1/2))
```

Ksat *Saturated Hydraulic Conductivity*

Description

The data consists of 32 measurements of the saturated hydraulic conductivity of a soil.

Usage

```
data(Ksat)
```

Format

The object `Ksat` is a list of the class `geodata` with the following elements:

`coords` a matrix with the coordinates of the soil samples.

`data` measurements of the saturated hidraulic conductivity.

`borders` a data-frame with the coordinates of a polygon defining the borders of the area.

Source

Data provided by Dr. Décio Cruciani, ESALQ/USP, Brasil.

Examples

```
data(Ksat)
summary(Ksat)
plot(Ksat, border=borders)
```

SIC *Spatial Interpolation Comparison data*

Description

Data from the SIC-97 project: Spatial Interpolation Comparison.

Usage

```
data(SIC)
```

Format

Four objects of the class "geodata": sic.all, sic.100, sic.367, sic.some. Each is a list with two components:

coords the coordinates of the data locations. The distance are given in kilometers.

data rainfall values. The unit is milimeters.

altitude elevation values. The unit is milimeters.

Additionally an matrix sic.borders with the borders of the country.

Source

Data from the project *Spatial Interpolation Comparison 97*; see ftp://ftp.geog.uwo.ca/SIC97/

References

Christensen, O.F., Diggle, P.J. and Ribeiro Jr, P.J. (2001) Analysing positive-valued spatial data: the transformed Gaussian model. In Monestiez, P., Allard, D. and Froidevaux (eds), GeoENV III - Geostatistics for environmental applications. Quantitative Geology and Geostatistics, Kluwer Series, 11, 287–298.

Examples

```
data(SIC)
points(sic.100, borders=sic.borders)
points(sic.all, borders=sic.borders)
```

`as.geodata` *Converts an Object to the Class "geodata"*

Description

The default method converts a matrix or a data-frame to an object of the `class` `"geodata"`. Objects of the class `"geodata"` are lists with two obligatory components: `coords` and `data`. Optional components are allowed and a typical example is a vector or matrix with covariate(s) values.

Usage

```
as.geodata(obj, ...)

## Default S3 method:
as.geodata(obj, coords.col = 1:2, data.col = 3, data.names = NULL,
                   covar.col = NULL, covar.names = "obj.names",
                   units.m.col = NULL, realisations = NULL,
                   na.action = c("ifany", "ifdata", "ifcovar", "none"),
                   rep.data.action, rep.covar.action, rep.units.action,
                   ...)

## S3 method for class 'geodata':
as.data.frame(x, ..., borders = TRUE)

## S3 method for class 'geodata.frame':
as.geodata(obj, ...)

## S3 method for class 'SpatialPointsDataFrame':
as.geodata(obj, data.col = 1, ...)

is.geodata(x)
```

Arguments

`obj`	a matrix or data-frame where each line corresponds to one spatial location. It should contain values of 2D coordinates, data and, optionally, covariate(s) value(s) at the locations. A method for `SpatialPointsDataFrame` is also provided. It can also take an output of the function `grf`, see DETAILS below.
`coords.col`	a vector with the column numbers corresponding to the spatial coordinates.
`data.col`	a scalar or vector with column number(s) corresponding to the data.
`data.names`	optional. A string or vector of strings with names for the data columns. Only valid if there is more than one column of data. By default, takes the names from the original object.

`covar.col` optional. A scalar or numeric vector with the column number(s) corresponding to the covariate(s). Alternativelly can be a character vector with the names of the covariates.

`covar.names` optional. A string or vector of strings with the name(s) of the covariates. By default take the names from the original object.

`units.m.col` optional. A scalar with the column number corresponding to the offset variable. Alternativelly can be a character vector with the name of the offset. This option is particularly relevant when using the package **geoRglm**. All values must be greater then zero.

`realisations` optional. A vector indicating the realisation number or a number indicating a column in `obj` with the realisation indicator variable. See `DETAILS` below.

`na.action` string defining action to be taken in the presence of `NA`'s. The default option `"ifany"` excludes all points for which there are `NA`'s in the data or covariates. The option `"ifdata"` excludes points for which there are `NA`'s in the data. The default option `"ifcovar"` excludes all points for which there are `NA`'s in the covariates. The option `"none"` do not exclude points.

`rep.data.action` a string or a function. Defines action to be taken when there is more than one data at the same location. The default option `"none"` keeps the repeated locations, if any. The option `"first"` retains only the first data recorded at each location. Alternativelly a function can be passed and it will be used. For instance if `mean` is provided, the function will compute and return the average of the data at coincident locations. The non-default options will eliminate the repeated locations.

`rep.covar.action` idem to `rep.data.locations`, to be applied to the covariates, if any. Defaults to the same option set for `rep.data.locations`.

`x` an object which is tested for the class `geodata`.

`rep.units.action` a string or a function. Defines action to be taken on the element `units.m`, if present when there is more than one data at the same location. The default option is the same value set for `rep.data.action`.

`borders` logical. If TRUE the element borders in the `geodata` object is set as an attribute of the data-frame.

`...` values to be passed for the methods.

Details

Objects of the class `"geodata"` contain data for geostatistical analysis using the package **geoR**. Storing data in this format facilitates the usage of the functions in **geoR**. However, conversion of objects to this class is not obligatory to carry out the analysis.

`NA`'s are not allowed in the coordinates. By default the respective rows will not be included in the output.

Realisations

Tipically geostatistical data correspond to a unique realisation of the spatial process. However,

sometimes different "realisations" are possible. For instance, if data are collected in the same area at different points in time and independence between time points is assumed, each time can be considered a different "replicate" or "realisation" of the same process. The argument `realisations` takes a vector indication the replication number and can be passed to other **geoR** functions as, for instance, `likfit`.

The data format is similar to the usual `geodata` format in **geoR**. Suppose there are realisations (times) $1, \ldots, J$ and for each realisations $n_1, ..., n_j$ observations are available. The coordinates for different realisations should be combined in a single $n \times 2$ object, where $n = n_1 + \ldots + n_J$. Similarly for the data vector and covariates (if any).

grf objects
If an object of the class `grf` is provided the functions just extracts the elements `coords` and `data` of this object.

Value

An object of the `class` `"geodata"` which is a list with two obligatory components (coords and data) and other optional components:

`coords`	an $n \times 2$ matrix where n is the number of spatial locations.
`data`	a vector of length n, for the univariate case or, an $n \times v$ matrix or data-frame for the multivariate case, where v is the number of variables.
`covariates`	a vector of length n or an $n \times p$ matrix with covariate(s) values, where p is the number of covariates. Only returned if covariates are provided.
`realisations`	a vector on size n with the replication number. Only returned if argument realisations is provided.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`read.geodata` for reading data from an *ASCII* file and `list` for general information on lists.

Examples

```
## Not run:
## converting the data-set "topo" from the package MASS (VR's bundle)
## to the geodata format:
require(MASS)
data(topo)
topo
topogeo <- as.geodata(topo)
```

```
names(topogeo)
topogeo
## End(Not run)
```

`boxcox.fit` *Parameter Estimation for the Box-Cox Transformation*

Description

Parameter estimation and plotting of the results for the Box-Cox transformed normal distribution.

Usage

```
boxcox.fit(object, xmat, lambda, lambda2 = NULL, add.to.data = 0, ...)

## S3 method for class 'boxcox.fit':
print(x, ...)

## S3 method for class 'boxcox.fit':
plot(x, hist = TRUE, data = eval(x$call$object), ...)

## S3 method for class 'boxcox.fit':
lines(x, data = eval(x$call$object), ...)
```

Arguments

`object`	a vector with the data.
`xmat`	a matrix with covariates values. Defaults to `rep(1, length(y))`.
`lambda`	numerical value(s) for the transformation parameter λ. Used as the initial value in the function for parameter estimation. If not provided default values are assumed. If multiple values are passed the one with highest likelihood is used as initial value.
`lambda2`	logical or numerical value(s) of the additional transformation (see DETAILS below). Defaults to `NULL`. If `TRUE` this parameter is also estimated and the initial value is set to the absolute value of the minimum data. A numerical value is provided it is used as the initial value. Multiple values are allowed as for `lambda`.
`add.to.data`	a constant value to be added to the data.
`x`	a list, typically an output of the function `boxcox.fit`.
`hist`	logical indicating whether histograms should to be plotted.
`data`	data values.
`...`	extra parameters to be passed to the minimization function `optim` (`boxcox.fit`), `hist` (`plot`) or `curve` (`lines`).

Value

The functions returns the following results:

`boxcox.fit` a list with estimated parameters and results on the numerical minimization.

`print.boxcox.fit`
print estimated parameters. No values returned.

`plot.boxcox.fit`
plots histogram of the data (optional) and the model. No values returned. This function is only valid if covariates are not included in `boxcox.fit`.

`lines.boxcox.fit`
adds a line with the fitted model to the current plot. No values returned. This function is only valid if covariates are not included in `boxcox.fit`.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Box, G.E.P. and Cox, D.R.(1964) An analysis of transformations. JRSS B **26**:211–246.

See Also

`rboxcox` and `dboxcox` for the expression and more on the Box-Cox transformation, the minimization function `optim`, the function `boxcox` in the package **MASS** and the function `boxcox` in the package **car**.

Examples

```
set.seed(384)
## Simulating data
simul <- rboxcox(100, lambda=0.5, mean=10, sd=2)
## Finding the ML estimates
ml <- boxcox.fit(simul)
ml
## Ploting histogram and fitted model
plot(ml)
##
## Comparing models with different lambdas,
## zero  means and unit variances
curve(dboxcox(x, lambda=-1), 0, 8)
for(lambda in seq(-.5, 1.5, by=0.5))
  curve(dboxcox(x, lambda), 0, 8, add = TRUE)
##
## Another example, now estimating lambda2
##
simul <- rboxcox(100, lambda=0.5, mean=10, sd=2)
ml <- boxcox.fit(simul, lambda2 = TRUE)
ml
```

```
plot(ml)
##
## An example with a regression model
##
if(require(MASS)){
  data(trees)
  boxcox.fit(object = trees[,3], xmat = trees[,1:2])
  }
```

`boxcox.geodata` *Box-Cox transformation for geodata objects*

Description

Method for Box-Cox transformation for objects of the class `geodata` assuming the data are independent. Computes and optionally plots profile log-likelihoods for the parameter of the Box-Cox simple power transformation y^lambda.

Usage

```
boxcox.geodata(object, trend = "cte", ...)
```

Arguments

`object`	an object of the class geodata. See `as.geodata`.
`trend`	specifies the mean part of the model. See `trend.spatial` for further details. Defaults to `"cte"`.
`...`	arguments to be passed for the function `boxcox`.

Details

This is just a wrapper for the function `boxcox` facilitating its usage with `geodata` objects.

Notice this assume independent observations which is typically not the case for `geodata` objects.

Value

A list of the `lambda` vector and the computed profile log-likelihood vector, invisibly if the result is plotted.

See Also

`boxcox`, `boxcox.fit` for parameter estimation results for independent data and `likfit` for parameter estimation within the geostatistical model.

Examples

```
if(require(MASS)){
data(wolfcamp)
boxcox(wolfcamp)

data(ca20)
boxcox(ca20, trend = ~altitude)
}
```

`ca20` *Calcium content in soil samples*

Description

This data set contains the calcium content measured in soil samples taken from the 0-20cm layer at 178 locations within a certain study area divided in three sub-areas. The elevation at each location was also recorded.

The first region is tipically flooded during the rain season and not used as an experimental area. The calcium levels would represent the natural content in the region. The second region has received fertilizers a while ago and is tipically occupied by rice fields. The third region has recieved fertilizers recently and is frequently used as an experimental area.

Usage

```
data(ca20)
```

Format

The object `ca20` belongs to the class `geodata` and is a list with the following elements:

`coords` a matrix with the coordinates of the soil samples.

`data` calcium content measured in $mmol_c/dm^3$.

`covariate` a data-frame with the covariates

- **`altitude`** a vector with the elevation of each sampling location, in meters (m).
- **`area`** a factor indicating the sub area to which the locations belongs.

`borders` a matrix with the coordinates defining the borders of the area.

`reg1` a matrix with the coordinates of the limits of the sub-area 1.

`reg1` a matrix with the coordinates of the limits of the sub-area 2.

`reg1` a matrix with the coordinates of the limits of the sub-area 3.

Source

The data was collected by researchers from PESAGRO and EMBRAPA-Solos, Rio de Janeiro, Brasil and provided by Dra. Maria Cristina Neves de Oliveira.

Capeche, C.L.; Macedo, J.R.; Manzatto, H.R.H.; Silva, E.F. (1997) Caracterização pedológica da fazenda Angra - PESAGRO/RIO - Estação experimental de Campos (RJ). (compact disc). In: Congresso BRASILEIRO de Ciência do Solo. 26., Informação, globalização, uso do solo; Rio de Janeiro, 1997. trabalhos. Rio de Janeiro: Embrapa/SBCS.

References

Oliveira, M.C.N. (2003) *Métodos de estimação de parâmetros em modelos geoestatísticos com diferentes estruturas de covariâncias: uma aplicação ao teor de cálcio no solo.* Tese de Doutorado, ESALQ/USP/Brasil.

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

`camg` *Calcium and magnesium content in soil samples*

Description

This data set contains the calcium content measured in soil samples taken from the 0-20cm layer at 178 locations within a certain study area divided in three sub-areas. The elevation at each location was also recorded.

The first region is tipically flooded during the rain season and not used as an experimental area. The calcium levels would represent the natural content in the region. The second region has received fertilizers a while ago and is tipically occupied by rice fields. The third region has recieved fertilizers recently and is frequently used as an experimental area.

Usage

```
data(camg)
```

Format

A data frame with 178 observations on the following 10 variables.

east east-west coordinates, in meters.

north north-south coordinates, in meters.

elevation elevation, in meters

region a factor where numbers indicate different sub-regions within the area

ca020 calcium content in the 0-20cm soil layer, measured in $mmol_c/dm^3$.

mg020 calcium content in the 0-20cm soil layer, measured in $mmol_c/dm^3$.

ctc020 calcium content in the 0-20cm soil layer.

ca2040 calcium content in the 20-40cm soil layer, measured in $mmol_c/dm^3$.

mg2040 calcium content in the 20-40cm soil layer, measured in $mmol_c/dm^3$.

ctc2040 calcium content in the 20-40cm soil layer.

Details

More details about this data-set, including coordinates of the region and sub-region borders can be found in the data object `ca20`.

Source

The data was collected by researchers from PESAGRO and EMBRAPA-Solos, Rio de Janeiro, Brasil and provided by Dra. Maria Cristina Neves de Oliveira.

Capeche, C.L.; Macedo, J.R.; Manzatto, H.R.H.; Silva, E.F. (1997) Caracterização pedológica da fazenda Angra - PESAGRO/RIO - Estação experimental de Campos (RJ). (compact disc). In: Congresso BRASILEIRO de Ciência do Solo. 26., Informação, globalização, uso do solo; Rio de Janeiro, 1997. trabalhos. Rio de Janeiro: Embrapa/SBCS.

Examples

```
data(camg)
plot(camg[-(1:2),])
mg20 <- as.geodata(camg, data.col=6)
plot(mg20)
points(mg20)
```

---

cite.geoR *Citing Package geoR in Publications*

Description

How to cite the package **geoR** in publications.

Usage

```
cite.geoR()
```

Details

Execute function `cite.geoR()` for information on how to cite geoR in publications.

Examples

```
cite.geoR()
```

`coords.aniso` *Geometric Anisotropy Correction*

Description

Transforms or back-transforms a set of coordinates according to the geometric anisotropy parameters.

Usage

```
coords.aniso(coords, aniso.pars, reverse = FALSE)
```

Arguments

`coords`	an $n \times 2$ matrix with the coordinates to be transformed.
`aniso.pars`	a vector with two elements, ψ_A and ψ_R, the *anisotropy angle* and the *anisotropy ratio*, respectively. Notice that the parameters must be provided in this order. See section `DETAILS` below for more information on anisotropy parameters.
`reverse`	logical. Defaults to `FALSE`. If `TRUE` the reverse transformation is performed.

Details

Geometric anisotropy is defined by two parameters:

Anisotropy angle defined here as the azimuth angle of the direction with greater spatial continuity, i.e. the angle between the *y-axis* and the direction with the maximum range.

Anisotropy ratio defined here as the ratio between the ranges of the directions with greater and smaller continuity, i.e. the ratio between maximum and minimum ranges. Therefore, its value is always greater or equal to one.

If `reverse = FALSE` (the default) the coordinates are transformed from the *anisotropic space* to the *isotropic space*. The transformation consists in multiplying the original coordinates by a rotation matrix R and a shrinking matrix T, as follows:

$$X_m = XRT,$$

where X_m is a matrix with the modified coordinates (isotropic space) , X is a matrix with original coordinates (anisotropic space), R rotates coordinates according to the anisotropy angle ψ_A and T shrinks the coordinates according to the anisotropy ratio ψ_R.

If `reverse = TRUE`, the back-transformation is performed, i.e. transforming the coordinates from the *isotropic space* to the *anisotropic space* by computing:

$$X = X_m(RT)^{-1}$$

Value

An $n \times 2$ matrix with the transformed coordinates.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

Examples

```
op <- par(no.readonly = TRUE)
par(mfrow=c(3,2))
par(mar=c(2.5,0,0,0))
par(mgp=c(2,.5,0))
par(pty="s")
## Defining a set of coordinates
coords <- expand.grid(seq(-1, 1, l=3), seq(-1, 1, l=5))
plot(c(-1.5, 1.5), c(-1.5, 1.5), xlab="", ylab="", type="n")
text(coords[,1], coords[,2], 1:nrow(coords))
## Transforming coordinates according to some anisotropy parameters
coordsA <- coords.aniso(coords, aniso.pars=c(0, 2))
plot(c(-1.5, 1.5), c(-1.5, 1.5), xlab="", ylab="", type="n")
text(coordsA[,1], coordsA[,2], 1:nrow(coords))
##
coordsB <- coords.aniso(coords, aniso.pars=c(pi/2, 2))
plot(c(-1.5, 1.5), c(-1.5, 1.5), xlab="", ylab="", type="n")
text(coordsB[,1], coordsB[,2], 1:nrow(coords))
##
coordsC <- coords.aniso(coords, aniso.pars=c(pi/4, 2))
plot(c(-1.5, 1.5), c(-1.5, 1.5), xlab="", ylab="", type="n")
text(coordsC[,1], coordsC[,2], 1:nrow(coords))
##
coordsD <- coords.aniso(coords, aniso.pars=c(3*pi/4, 2))
plot(c(-1.5, 1.5), c(-1.5, 1.5), xlab="", ylab="", type="n")
text(coordsD[,1], coordsD[,2], 1:nrow(coords))
##
coordsE <- coords.aniso(coords, aniso.pars=c(0, 5))
plot(c(-1.5, 1.5), c(-1.5, 1.5), xlab="", ylab="", type="n")
text(coordsE[,1], coordsE[,2], 1:nrow(coords))
##
par(op)
```

---

coords2coords *Operations on Coordinates*

Description

Functions for shifting, zooming and envolving rectangle of a set of coordinates.

Usage

```
coords2coords(coords, xlim, ylim, xlim.ori, ylim.ori)

zoom.coords(x, ...)

## Default S3 method:
zoom.coords(x, xzoom, yzoom, xlim.ori, ylim.ori, xoff=0, yoff=0, ...)

## S3 method for class 'geodata':
zoom.coords(x, ...)

rect.coords(coords, xzoom = 1, yzoom=xzoom, add.to.plot=TRUE,
            quiet = FALSE, ...)
```

Arguments

coords, x	two column matrix or data-frame with coordinates.
xlim	range of the new x-coordinates.
ylim	range of the new y-coordinates.
xlim.ori	optional. Range of the original x-coordinates, by default the range of the original x-coordinates.
ylim.ori	optional. Range of the original y-coordinates, by default the range of the original y-coordinates.
xzoom	scalar, expanding factor in the x-direction.
yzoom	scalar, expanding factor in the y-direction.
xoff	scalar, shift in the x-direction.
yoff	scalar, shift in the y-direction.
add.to.plot	logical, if TRUE the retangle is added to the current plot.
quiet	logical, none is returned.
...	further arguments to be passed to rect.

Value

coords2coords and zoom.coords	return an object of the same type as given in the argument coords with the transformed coordinates.
rect.coords	returns a matrix with the 4 coordinates of the rectangle defined by the coordinates.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

See Also

subarea, rect

Examples

```
foo <- matrix(c(4,6,6,4,2,2,4,4), nc=2)
foo1 <- zoom.coords(foo, 2)
foo1
foo2 <- coords2coords(foo, c(6,10), c(6,10))
foo2
plot(1:10, 1:10, type="n")
polygon(foo)
polygon(foo1, lty=2)
polygon(foo2, lwd=2)
arrows(foo[,1], foo[,2],foo1[,1],foo1[,2], lty=2)
arrows(foo[,1], foo[,2],foo2[,1],foo2[,2])
legend(1,10, c("foo", "foo1 (zoom.coords)", "foo2 (coords2coords)"), lty=c(1,2,1), lwd=c(1,1

## "zooming" part of The Gambia map
data(gambia)
gb <- gambia.borders/1000
gd <- gambia[,1:2]/1000
plot(gb, ty="l", asp=1, xlab="W-E (kilometres)", ylab="N-S (kilometres)")
points(gd, pch=19, cex=0.5)
r1b <- gb[gb[,1] < 420,]
rc1 <- rect.coords(r1b, lty=2)

r1bn <- zoom.coords(r1b, 1.8, xoff=90, yoff=-90)
rc2 <- rect.coords(r1bn, xz=1.05)
segments(rc1[c(1,3),1],rc1[c(1,3),2],rc2[c(1,3),1],rc2[c(1,3),2], lty=3)

lines(r1bn)
r1d <- gd[gd[,1] < 420,]
r1dn <- zoom.coords(r1d, 1.7, xlim.o=range(r1b[,1],na.rm=TRUE), ylim.o=range(r1b[,2],na.rm=T
points(r1dn, pch=19, cex=0.5)
text(450,1340, "Western Region", cex=1.5)
```

---

cov.spatial *Computes Value of the Covariance Function*

Description

Computes the covariances for pairs variables, given the separation distance of their locations. Options for different correlation functions are available. The results can be seen as a change of metric, from the *Euclidean distances* to *covariances*.

Usage

```
cov.spatial(obj, cov.model= "matern",
            cov.pars=stop("no cov.pars argument provided"),
            kappa = 0.5)
```

Arguments

`obj`	a numeric object (vector or matrix), typically with values of distances between pairs of spatial locations.
`cov.model`	string indicating the type of the correlation function. Available choices are: "matern", "exponential", "gaussian", "spherical", "circular", "cubic", "wave", "power", "powered.exponential", "cauchy", "gencauchy", "gneiting", "gneiting.matern", "pure.nugget". See section `DETAILS` for available options and expressions of the correlation functions.
`cov.pars`	a vector with 2 elements or an $ns \times 2$ matrix with the covariance parameters. The first element (if a vector) or first column (if a matrix) corresponds to the variance parameter σ^2. The second element or column corresponds to the range parameter ϕ of the correlation function. If a matrix is provided, each row corresponds to the parameters of one *spatial structure* (see DETAILS below).
`kappa`	numerical value for the additional smoothness parameter of the correlation function. Only required by the following correlation functions: `"matern"`, `"powered.exponential"`, `"cauchy"`, `"gencauchy"` and `"gneiting.matern"`.

Details

Covariance functions return the value of the covariance $C(h)$ between a pair variables located at points separated by the distance h. The covariance function can be written as a product of a variance parameter σ^2 times a positive definite *correlation function* $\rho(h)$:

$$C(h) = \sigma^2 \rho(h).$$

The expressions of the covariance functions available in **geoR** are given below. We recommend the *LaTeX* (and/or the corresponding *.dvi*, *.pdf* or *.ps*) version of this document for better visualization of the formulas.

Denote ϕ the basic parameter of the correlation function and name it the *range parameter*. Some of the correlation functions will have an extra parameter κ, the *smoothness parameter*. $K_\kappa(x)$ denotes the modified Bessel function of the third kind of order κ. See documentation of the function `besselK` for further details. In the equations below the functions are valid for $\phi > 0$ and $\kappa > 0$, unless stated otherwise.

cauchy

$$\rho(h) = [1 + (\frac{h}{\phi})^2]^{-\kappa}$$

gencauchy (generalised Cauchy)

$$\rho(h) = [1 + (\frac{h}{\phi})_2^\kappa]^{-\kappa_1/\kappa_2}, \kappa_1 > 0, 0 < \kappa_2 \leq 2$$

circular
Let $\theta = \min(\frac{h}{\phi}, 1)$ and

$$g(h) = 2\frac{(\theta\sqrt{1-\theta^2} + \sin^{-1}\sqrt{\theta})}{\pi}.$$

Then, the circular model is given by:

$$\rho(h) = \begin{cases} 1 - g(h) \text{ , if } h < \phi \\ 0 \text{ , otherwise} \end{cases}$$

cubic

$$\rho(h) = \begin{cases} 7(\frac{h}{\phi})^2 - 8.75(\frac{h}{\phi})^3 + 3.5(\frac{h}{\phi})^5 - 0.75(\frac{h}{\phi})^7 \text{ , if } h < \phi \\ 0 \text{ , otherwise.} \end{cases}$$

gaussian

$$\rho(h) = \exp[(-\frac{h}{\phi})^2]$$

exponential

$$\rho(h) = \exp(-\frac{h}{\phi})$$

matern

$$\rho(h) = \frac{1}{2^{\kappa-1}\Gamma(\kappa)}(\frac{h}{\phi})^{\kappa}K_{\kappa}(\frac{h}{\phi})$$

spherical

$$\rho(h) = \begin{cases} 1 - 1.5\frac{h}{\phi} + 0.5(\frac{h}{\phi})^3 \text{ , if } h < \phi \\ 0 \text{ , otherwise} \end{cases}$$

power (and linear)
The parameters of the this model σ^2 and ϕ can not be interpreted as *partial sill* and *range* as for the other models. This model implies an unlimited dispersion and, therefore, has no sill and corresponds to a process which is only intrinsically stationary. The variogram function is given by:

$$\gamma(h) = \sigma^2 h^{\phi} \text{ , } 0 < \phi < 2, \sigma^2 > 0$$

Since the corresponding process is not second order stationary the covariance and correlation functions are not defined. For internal calculations the *geoR* functions uses the fact the this model possesses locally stationary representations with covariance functions of the form:

$$C_(h) = \sigma^2(A - h^{\phi}),$$

where A is a suitable constant as given in Chilès & Delfiner (pag. 511, eq. 7.35).

The *linear* model corresponds a particular case with $\phi = 1$.

powered.exponential (or stable)

$$\rho(h) = \exp[-(\frac{h}{\phi})^{\kappa}] \, , 0 < \kappa \leq 2$$

gneiting

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

where $s = 0.301187465825$. For further details see documentation of the function `CovarianceFct` in the package `RandomFields` from where we extract the following :
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.

gneiting.matern
Let $\alpha = \phi\kappa_2$, $\rho_m(\cdot)$ denotes the Matérn model and $\rho_g(\cdot)$ the Gneiting model. Then the Gneiting-Matérn is given by

$$\rho(h) = \rho_g(h|\phi = \alpha)\, \rho_m(h|\phi = \phi, \kappa = \kappa_1)$$

wave

$$\rho(h) = \frac{\phi}{h} \sin(\frac{h}{\phi})$$

pure.nugget

$$\rho(h) = k$$

where k is a constant value. This model corresponds to no spatial correlation.

Nested models Models with several structures usually called *nested models* in the geostatistical literature are also allowed. In this case the argument `cov.pars` takes a matrix and `cov.model` and `lambda` can either have length equal to the number of rows of this matrix or length 1. For the latter cov.model and/or lambda are recycled, i.e. the same value is used for all structures.

Value

The function returns values of the covariances corresponding to the given distances. The type of output is the same as the type of the object provided in the argument `obj`, typically a vector, matrix or array.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

For a review on correlation functions:
Schlather, M. (1999) *An introduction to positive definite functions and to unconditional simulation of random fields*. Technical report ST 99-10, Dept. of Maths and Statistics, Lancaster University.

Chilès, J.P. and Delfiner, P. (1999) **Geostatistics: Modelling Spatial Uncertainty**, Wiley.

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

matern for computation of the Matérn model, besselK for computation of the Bessel function and varcov.spatial for computations related to the covariance matrix.

Examples

```
#
# Variogram models with the same "practical" range:
#
v.f <- function(x, ...){1-cov.spatial(x, ...)}
#
curve(v.f(x, cov.pars=c(1, .2)), from = 0, to = 1,
      xlab = "distance", ylab = expression(gamma(h)),
      main = "variograms with equivalent \"practical range\"")
curve(v.f(x, cov.pars = c(1, .6), cov.model = "sph"), 0, 1,
      add = TRUE, lty = 2)
curve(v.f(x, cov.pars = c(1, .6/sqrt(3)), cov.model = "gau"),
      0, 1, add = TRUE, lwd = 2)
legend(0.5,.3, c("exponential", "spherical", "gaussian"),
       lty=c(1,2,1), lwd=c(1,1,2))
#
# Matern models with equivalent "practical range"
# and varying smoothness parameter
#
curve(v.f(x, cov.pars = c(1, 0.25), kappa = 0.5),from = 0, to = 1,
      xlab = "distance", ylab = expression(gamma(h)), lty = 2,
      main = "models with equivalent \"practical\" range")
curve(v.f(x, cov.pars = c(1, 0.188), kappa = 1),from = 0, to = 1,
      add = TRUE)
curve(v.f(x, cov.pars = c(1, 0.14), kappa = 2),from = 0, to = 1,
      add = TRUE, lwd=2, lty=2)
curve(v.f(x, cov.pars = c(1, 0.117), kappa = 2),from = 0, to = 1,
      add = TRUE, lwd=2)
legend(0.4,.4, c(expression(paste(kappa == 0.5, "  and  ",
       phi == 0.250)),
       expression(paste(kappa == 1, "  and  ", phi == 0.188)),
       expression(paste(kappa == 2, "  and  ", phi == 0.140)),
       expression(paste(kappa == 3, "  and  ", phi == 0.117))),
       lty=c(2,1,2,1), lwd=c(1,1,2,2))
# plotting a nested variogram model
curve(v.f(x, cov.pars = rbind(c(.4, .2), c(.6,.3)),
          cov.model = c("sph","exp")), 0, 1, ylab='nested model')
```

dup.coords *Locates duplicated coordinates*

Description

This funtions takes an object with 2-D coordinates and returns the positions of the duplicated coordinates. Also sets a method for `duplicated`

Usage

```
dup.coords(x, ...)
## Default S3 method:
dup.coords(x, ...)
## S3 method for class 'geodata':
dup.coords(x, incomparables, ...)
## S3 method for class 'geodata':
duplicated(x, incomparables, ...)
```

Arguments

`x`	a two column numeric matrix or data frame.
`incomparables`	unused. Just for compatibility with the generic function `duplicated`.
`...`	unused.

Value

Function and methods returns `NULL` if there are no duplicates locations.

Otherwise, the default method returns a list where each component is a vector with the positions or the rownames, if available, of the duplicates coordinates.

The method for `geodata` returns a data-frame with rownames equals to the positions of the duplicated coordinates, the first column is a factor indicating duplicates and the remaning are output of `as.data.frame.geodata`.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

See Also

`as.geodata`, `duplicated`.

Examples

```
## simulating data
foo <- grf(30, cov.p=c(1, .3))
## "forcing" some duplicated locations
foo$coords[4,] <- foo$coords[14,] <- foo$coords[24,] <- foo$coords[2,]
foo$coords[17,] <- foo$coords[23,] <- foo$coords[8,]
## output of the method for geodata
dup.coords(foo)
## which is the same as a method for duplicated()
duplicated(foo)
## the default method:
dup.coords(foo$coords)
```

`elevation` *Surface Elevations*

Description

Surface elevation data taken from Davis (1972). An onject of the class `geodata` with elevation values at 52 locations.

Usage

```
data(elevation)
```

Format

An object of the class `geodata` which is a list with the following elements:

`coords` x-y coordinates (multiples of 50 feet).

`data` elevations (multiples of 10 feet).

Source

Davis, J.C. (1973) *Statistics and Data Analysis in Geology.* Wiley.

Examples

```
data(elevation)
summary(elevation)
plot(elevation)
```

eyefit *Interactive Variogram Estimation*

Description

Function to fit an empirical variogram "by eye" using an interactive Tcl-Tk interface.

Usage

```
eyefit(vario, silent = FALSE)
```

Arguments

vario	An empirical variogram object as returned by the function variog.
silent	logical indicating wheather or not the fitted variogram must be returned.

Value

Returns a list of list with the model parameters for each of the saved fit(s).

Author(s)

Andreas Kiefer ⟨andreas@inf.ufpr.br⟩
Paulo Justiniano Rineiro Junior ⟨paulojus@est.ufpr.br⟩.

See Also

variofit for least squares variogram fit, likfit for likelihood based parameter estimation and krige.bayes to obtain the posterior distribution for the model parameters.

gambia *Gambia Malaria Data*

Description

Malaria prevalence in children recorded at villages in The Gambia, Africa.

Usage

```
data(gambia)
```

Format

Two objects are made available:

1. `gambia`
   A data frame with 2035 observations on the following 8 variables.

   x x-coordinate of the village (UTM).

   y y-coordinate of the village (UTM).

   pos presence (1) or absence (0) of malaria in a blood sample taken from the child.

   age age of the child, in days

   netuse indicator variable denoting whether (1) or not (0) the child regularly sleeps under a bed-net.

   treated indicator variable denoting whether (1) or not (0) the bed-net is treated (coded 0 if netuse=0).

   green satellite-derived measure of the green-ness of vegetation in the immediate vicinity of the village (arbitrary units).

   phc indicator variable denoting the presence (1) or absence (0) of a health center in the village.

2. `gambia.borders`
   A data frame with 2 variables:

   x x-coordinate of the country borders.

   y y-coordinate of the country borders.

References

Thomson, M., Connor, S., D Alessandro, U., Rowlingson, B., Diggle, P., Cresswell, M. & Greenwood, B. (1999). Predicting malaria infection in Gambian children from satellite data and bednet use surveys: the importance of spatial correlation in the interpretation of results. *American Journal of Tropical Medicine and Hygiene* 61: 2–8.

Diggle, P., Moyeed, R., Rowlingson, B. & Thomson, M. (2002). Childhood malaria in The Gambia: a case-study in model-based geostatistics, *Applied Statistics*.

Examples

```
data(gambia)
plot(gambia.borders, type="l", asp=1)
points(gambia[,1:2], pch=19)
# a built-in function for a zoomed map
gambia.map()
# Building a "village-level" data frame
ind <- paste("x",gambia[,1], "y", gambia[,2], sep="")
village <- gambia[!duplicated(ind),c(1:2,7:8)]
village$prev <- as.vector(tapply(gambia$pos, ind, mean))
plot(village$green, village$prev)
```

geoR-defunct *Defunct Functions in the Package geoR*

Description

The functions listed here are no longer part of the package **geoR** as they are not needed (any more).

Usage

```
geoRdefunct()
```

Details

The following functions are now defunct:

olsfit	functionality incorporated by `variofit`. From `geoR_1.0-6`.
wlsfit	functionality incorporated by `variofit`. From `geoR_1.0-6`.
likfit.old	functionality incorporated by `likfit`. From `geoR_1.0-6`. The associated functions were also made defunct: `\likfit.nospatial`, `loglik.spatial`, `proflik.nug`, `proflik.phi`, `proflik.ftau`.
distdiag	functionally is redundant with `dist`.

See Also

`variofit`

geoR-internal *geoR internal functions*

Description

These are functions internally called by other functions in the package **geoR** and not meant to be called by the user.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

`globalvar` *function to do ...*

Description

A concise (1-5 lines) description of what the function does.

Usage

```
globalvar(geodata, locations, coords = geodata$coords, krige)
```

Arguments

`geodata`	Describe `geodata` here
`locations`	
`coords`	Describe `coords` here
`krige`	a list defining the model components and the type of kriging. It can take an output to a call to `krige.control` or a list with elements as for the arguments in `krige.control`. More details in the documention for

Details

If necessary, more details than the description above

Value

Describe the value returned If it is a LIST, use

`comp1`	Description of 'comp1'
`comp2`	Description of 'comp2'

...

Author(s)

who you are

References

Isaaks, E.S and Srivastava, R.M. (1989) An Introduction to Applied Geostatistics, pag. 508, eq. 20.7. Oxford University Press.

See Also

`krige.conv` for the kriging algorithm.

Examples

```
data(s100)
```

`grf` *Simulation of Gaussian Random Fields*

Description

`grf()` generates (unconditional) simulations of Gaussian random fields for given covariance parameters. `geoR2RF` converts model specification used by **geoR** to the correponding one in **RandomFields**.

Usage

```
grf(n, grid = "irreg", nx, ny, xlims = c(0, 1), ylims = c(0, 1),
    nsim = 1, cov.model = "matern",
    cov.pars = stop("missing covariance parameters sigmasq and phi"),
    kappa = 0.5, nugget = 0, lambda = 1, aniso.pars,
    mean = 0, method, RF=TRUE, messages)

geoR2RF(cov.model, cov.pars, nugget = 0, kappa, aniso.pars)
```

Arguments

`n`	number of points (spatial locations) in each simulations.
`grid`	optional. An $n \times 2$ matrix with coordinates of the simulated data.
`nx`	optional. Number of points in the X direction.
`ny`	optional. Number of points in the Y direction.
`xlims`	optional. Limits of the area in the X direction. Defaults to $[0, 1]$.
`ylims`	optional. Limits of the area in the Y direction. Defaults to $[0, 1]$.
`nsim`	Number of simulations. Defaults to 1.
`cov.model`	correlation function. See `cov.spatial` for further details. Defaults to the *exponential* model.
`cov.pars`	a vector with 2 elements or an $n \times 2$ matrix with values of the covariance parameters σ^2 (partial sill) and ϕ (range parameter). If a vector, the elements are the values of σ^2 and ϕ, respectively. If a matrix, corresponding to a model with several structures, the values of σ^2 are in the first column and the values of ϕ are in the second.
`kappa`	additional smoothness parameter required only for the following correlation functions: `"matern"`, `"powered.exponential"`, `"cauchy"` and `"gneiting.matern"`. More details on the documentation for the function `cov.spatial`.
`nugget`	the value of the nugget effect parameter τ^2.
`lambda`	value of the Box-Cox transformation parameter. The value $\lambda = 1$ corresponds to no transformation, the default. For any other value of λ Gaussian data is simulated and then transformed.

aniso.pars	geometric anisotropy parameters. By default an isotropic field is assumed and this argument is ignored. If a vector with 2 values is provided, with values for the anisotropy angle ψ_A (in radians) and anisotropy ratio ψ_A, the coordinates are transformed, the simulation is performed on the isotropic (transformed) space and then the coordinates are back-transformed such that the resulting field is anisotropic. Coordinates transformation is performed by the function coords.aniso.
mean	a numerical vector of size 1 or the sema dimention of the data to be simulated.
method	simulation method with options for "cholesky", "svd", "eigen", "RF". Defaults to the *Cholesky* decomposition. See section DETAILS below.
RF	logical, with defaults to TRUE, indicating whether the algorithm should try to use the function GaussRF from the package **RandomFields** in case of method is missing and the number of points is greater than 500.
messages	logical, indicating whether or not status messages are printed on the screen (or output device) while the function is running. Defaults to TRUE.

Details

For the methods "cholesky", "svd" and "eigen" the simulation consists of multiplying a vector of standardized normal deviates by a square root of the covariance matrix. The square root of a matrix is not uniquely defined. These three methods differs in the way they compute the square root of the (positive definite) covariance matrix.

The previously available method method = "circular.embedding" is no longer available in **geoR**. For simulations in a fine grid and/or with a large number of points use the package **RandomFields**.

The option "RF" calls internally the function GaussRF from the package **RandomFields**.

Value

grf returns a list with the components:

coords	an $n \times 2$ matrix with the coordinates of the simulated data.
data	a vector (if nsim = 1) or a matrix with the simulated values. For the latter each column corresponds to one simulation.
cov.model	a string with the name of the correlation function.
nugget	the value of the nugget parameter.
cov.pars	a vector with the values of σ^2 and ϕ, respectively.
kappa	value of the parameter κ.
lambda	value of the Box-Cox transformation parameter λ.
aniso.pars	a vector with values of the anisotropy parameters, if provided in the function call.
method	a string with the name of the simulation method used.
sim.dim	a string "1d" or "2d" indicating the spatial dimension of the simulation.
.Random.seed	the random seed by the time the function was called.

`messages`	messages produced by the function describing the simulation.
`call`	the function call.
`model`	**RandomFields** name of the correlation model
`param`	**RandomFields** parameter vector

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Wood, A.T.A. and Chan, G. (1994) Simulation of stationary Gaussian process in $[0,1]^d$. *Journal of Computatinal and Graphical Statistics*, **3**, 409–432.

Schlather, M. (1999) *Introduction to positive definite functions and to unconditional simulation of random fields*. Tech. Report ST–99–10, Dept Maths and Stats, Lancaster University.

Schlather, M. **RandomFields**: *Simulation and Analysis of Random Fields*. R package. http://www.cu.lu/~schlathe.

Schlather, M. (2001) *Simulation and Analysis of Random Fields*. R-News **1** (2), p. 18-20.

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`plot.grf` and `image.grf` for graphical output, `coords.aniso` for anisotropy coordinates transformation and, `chol`, `svd` and `eigen` for methods of matrix decomposition and `GaussRF` function in the package **RandomFields**.

Examples

```
sim1 <- grf(100, cov.pars = c(1, .25))
# a display of simulated locations and values
points(sim1)
# empirical and theoretical variograms
plot(sim1)
#
# a "smallish" simulation
sim2 <- grf(441, grid = "reg", cov.pars = c(1, .25))
image(sim2)
##
## 1-D simulations using the same seed and different noise/signal ratios
##
set.seed(234)
sim11 <- grf(100, ny=1, cov.pars=c(1, 0.25), nug=0)
set.seed(234)
sim12 <- grf(100, ny=1, cov.pars=c(0.75, 0.25), nug=0.25)
set.seed(234)
sim13 <- grf(100, ny=1, cov.pars=c(0.5, 0.25), nug=0.5)
##
```

```
par.ori <- par(no.readonly = TRUE)
par(mfrow=c(3,1), mar=c(3,3,.5,.5))
yl <- range(c(sim11$data, sim12$data, sim13$data))
image(sim11, type="l", ylim=yl)
image(sim12, type="l", ylim=yl)
image(sim13, type="l", ylim=yl)
par(par.ori)
```

`head` *Head observations in a regional confined aquifer*

Description

Measurements of potentiometric head at 29 locations in a regional confined sandstone aquifer. Extract from Kitanidis' book.

Usage

```
data(head)
```

Format

An object of the class `geodata` which is a list with the elements:

coords coordinates of the data location.

data the data vector with head measurements (feet).

Source

Kitanidis, P.K. Introduction to geostatistics - applications in hidrology (1997). Cambridge University Press.

Examples

```
data(head)
summary(head)
plot(head)
```

`hist.krige.bayes` *Plots Sample from Posterior Distributions*

Description

Plots histograms and/or density estimation with samples from the posterior distribution of the model parameters

Usage

```
## S3 method for class 'krige.bayes':
hist(x, pars, density.est = TRUE, histogram = TRUE, ...)
```

Arguments

`x`	an object of the class `krige.bayes`, with an output of the funtions `krige.bayes`.
`pars`	a vector with the names of one or more of the model parameters. Defaults to all model parameters. Setting to -1 excludes the intercept.
`density.est`	logical indication whether a line with the density estimation should be added to the plot.
`histogram`	logical indicating whether the histogram is included in the plot.
`...`	further arguments for the plotting functions and or for the density estimation.

Value

Produces a plot in the currently graphics device.
Returns a `invisible` list with the components:

`histogram`	with the output of the function `hist` for each parameter
`density.estimation`	with the output of the function `density` for each parameter

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

See Also

`krige.bayes`, `hist`, `density`.

Examples

```
## See documentation for krige.bayes()
```

`hoef` *Data for spatial analysis of experiments*

Description

The `hoef` data frame has 25 rows and 5 columns.
The data consists of a uniformity trial for which *artificial* treatment effects were assign to the plots.

Usage

```
data(hoef)
```

Format

This data frame contains the following columns:

x1 x-coordinate of the plot.

x2 y-coordinate of the plot.

dat the *artificial* data.

trat the treatment number.

ut the data from the uniformity trial, without the treatment effect.

Details

The treatment effects assign to the plots are:

- Treatment 1: $\tau_1 = 0$
- Treatment 2: $\tau_2 = -3$
- Treatment 3: $\tau_3 = -5$
- Treatment 4: $\tau_4 = 6$
- Treatment 5: $\tau_5 = 6$

Source

Ver Hoef, J.M. & Cressie, N. (1993) Spatial statistics: analysis of field experiments. In Scheiner S.M. and Gurevitch, J. (Eds) *Design and Analysis of Ecological Experiments*. Chapman and Hall.

Examples

```
data(hoef)
hoef.geo <- as.geodata(hoef, covar.col=4)
summary(hoef)
summary(hoef.geo)
points(hoef.geo, cex.min=2, cex.max=2, pt.div="quintiles")
```

`image.grf` *Image, Contour or Perspective Plot of Simulated Gaussian Random Field*

Description

Methods for image, contour or perspective plot of a realisation of a Gaussian random field, simulated using the function `grf`.

Usage

```
## S3 method for class 'grf':
image(x, sim.number = 1, x.leg, y.leg, ...)
## S3 method for class 'grf':
contour(x, sim.number = 1, filled = FALSE, ...)
## S3 method for class 'grf':
persp(x, sim.number = 1, ...)
```

Arguments

`x`	an object of the class `grf`, typically an output of the function `grf`.
`sim.number`	simulation number. Indicates the number of the simulation top be plotted. Only valid if the object contains more than one simulation. Defaults to 1.
`x.leg, y.leg`	limits for the legend in the horizontal and vertical directions.
`filled`	logical. If `FALSE` the function `contour` is used otherwise `filled.contour`. Defaults to `FALSE`.
`...`	further arguments to be passed to the functions `image`, `contour` or `persp`.

Value

An image or perspective plot is produced on the current graphics device. No values are returned.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information about the package **geoR** can be found at:
`http://www.est.ufpr.br/geoR`.

See Also

`grf` for simulation of Gaussian random fields, `image` and `persp` for the generic plotting functions.

Examples

```
# generating 4 simulations of a Gaussian random field
sim <- grf(441, grid="reg", cov.pars=c(1, .25), nsim=4)
op <- par(no.readonly = TRUE)
par(mfrow=c(2,2), mar=c(3,3,1,1), mgp = c(2,1,0))
for (i in 1:4)
  image(sim, sim.n=i)
par(op)
```

---

image.krige.bayes *Plots Results of the Predictive Distribution*

Description

This function produces an image or perspective plot of a selected element of the predictive distribution returned by the function krige.bayes.

Usage

```
## S3 method for class 'krige.bayes':
image(x, locations, borders,
                  values.to.plot=c("mean", "variance",
                            "mean.simulations", "variance.simulations",
                            "quantiles", "probabilities", "simulation"),
                  number.col, coords.data, x.leg, y.leg, messages, ...)
## S3 method for class 'krige.bayes':
contour(x, locations, borders,
                  values.to.plot = c("mean", "variance",
                       "mean.simulations", "variance.simulations",
                       "quantiles", "probabilities", "simulation"),
                  filled=FALSE, number.col, coords.data,
                  x.leg, y.leg, messages, ...)
## S3 method for class 'krige.bayes':
persp(x, locations, borders,
                  values.to.plot=c("mean", "variance",
                       "mean.simulations", "variance.simulations",
                       "quantiles", "probabilities", "simulation"),
                  number.col, messages, ...)
```

Arguments

x	an object of the class krige.bayes, typically an output of the function krige.bayes.
locations	an $n \times 2$ matrix with the coordinates of the prediction locations, which should define a regular grid in order to be plotted by image or persp. By default does not need to be provided and evaluates the attribute "prediction.locations" from the input object.

`borders`	an $n \times 2$ matrix with the coordinates defining the borders of a region inside the grid defined by `locations`. Elements in the argument `values` are assigned to locations internal to the borders and `NA`'s to the external ones.
`values.to.plot`	select the element of the predictive distribution to be plotted. See DETAILS below.
`filled`	logical. If `FALSE` the function `contour` is used otherwise `filled.contour`. Defaults to `FALSE`.
`number.col`	Specifies the number of the column to be plotted. Only used if previous argument is set to one of `"quantiles"`, `"probabilities"` or `"simulation"`.
`coords.data`	optional. If an $n \times 2$ matrix with the data coordinates is provided, points indicating the data locations are included in the plot.
`x.leg, y.leg`	limits for the legend in the horizontal and vertical directions.
`messages`	logical, if TRUE status messages are printed while running the function.
`...`	extra arguments to be passed to the plotting function `image` or `persp`.

Details

The function `krige.bayes` returns summaries and other results about the predictive distributions. The argument `values.to.plot` specifies which result will be plotted. It can be passed to the function in two different forms:

- a vector with the object containing the values to be plotted, or
- one of the following options: `"moments.mean"`, `"moments.variance"`, `"mean.simulations"`, `"variance.simulations"`, `"quantiles"`, `"probability"` or `"simulation"`.

For the last three options, if the results are stored in matrices, a column number must be provided using the argument `number.col`.

The documentation for the function `krige.bayes` provides further details about these options.

Value

An `image` or `persp` plot is produced on the current graphics device. No values are returned.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
`http://www.est.ufpr.br/geoR`.

See Also

`krige.bayes` for Bayesian Kriging computations and, `image` and `persp` for the generic plotting functions.

Examples

```
#See examples in the documentation for the function krige.bayes().
```

`image.kriging` *Image or Perspective Plot with Kriging Results*

Description

Plots image or perspective plots with results of the kriging calculations.

Usage

```
## S3 method for class 'kriging':
image(x, locations, borders, values = x$predict,
              coords.data, x.leg, y.leg, ...)
## S3 method for class 'kriging':
contour(x, locations, borders, values = x$predict,
              coords.data, filled=FALSE, ...)
## S3 method for class 'kriging':
persp(x, locations, borders, values = x$predict, ...)
```

Arguments

`x`	an object of the class `kriging`, typically with the output of the functions `krige.conv` or `ksline`.
`locations`	an $n \times 2$ matrix with the coordinates of the prediction locations, which should define a regular grid in order to be plotted by `image` or `persp`. By default does not need to be provided and evaluates the attribute "prediction.locations" from the input object.
`borders`	an $n \times 2$ matrix with the coordinates defining the borders of a region inside the grid defined by `locations`. Elements in the argument `values` are assigned to locations internal to the borders and `NA`'s to the external ones.
`values`	a vector with values to be plotted. Defaults to `obj$predict`.
`coords.data`	optional. If an $n \times 2$ matrix with the data coordinates is provided, points indicating the data locations are included in the plot.
`x.leg, y.leg`	limits for the legend in the horizontal and vertical directions.
`filled`	logical. If `FALSE` the function `contour` is used otherwise `filled.contour`. Defaults to `FALSE`.
`...`	further arguments to be passed to the functions `image`, `contour`, `filled.contour`, `persp` or `legend.krige`. For instance, the argument `zlim` can be used to set the the minimum and maximum 'z' values for which colors should be plotted. See documentation for those function for possible arguments.

Details

`plot1d` and `prepare.graph.kriging` are auxiliary functions called by the others.

Value

An image or perspective plot is produced o the current graphics device. No values are returned.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`krige.conv` and `ksline` for kriging calculations. Documentation for `image`, `contour`, `filled.contour` and `persp` contain basic information on the plotting functions.

Examples

```
data(s100)
loci <- expand.grid(seq(0,1,l=31), seq(0,1,l=31))
kc <- krige.conv(s100, loc=loci,
                 krige=krige.control(cov.pars=c(1, .25)))
image(kc)
contour(kc)
image(kc)
contour(kc, add=TRUE, nlev=21)
persp(kc, theta=20, phi=20)
contour(kc, filled=TRUE)
contour(kc, filled=TRUE, color=terrain.colors)
contour(kc, filled=TRUE, col=gray(seq(1,0,l=21)))
# adding data locations
image(kc, coords.data=s100$coords)
contour(kc,filled=TRUE,coords.data=s100$coords,color=terrain.colors)
#
# now dealing with borders
#
bor <- matrix(c(.4,.1,.3,.9,.7,.9,.7,.3,.2,.5,.8),
              ncol=2)
# plotting just inside borders
image(kc, borders=bor)
contour(kc, borders=bor)
image(kc, borders=bor)
contour(kc, borders=bor, add=TRUE)
contour(kc, borders=bor, filled=TRUE, color=terrain.colors)
# kriging just inside borders
```

```
kc1 <- krige.conv(s100, loc=loci,
                  krige=krige.control(cov.pars=c(1, .25)),
                  borders=bor)
image(kc1)
contour(kc1)
# avoiding the borders
image(kc1, borders=NULL)
contour(kc1, borders=NULL)

op <- par(no.readonly = TRUE)
par(mfrow=c(1,2), mar=c(3,3,0,0), mgp=c(1.5, .8,0))
image(kc)
image(kc, val=sqrt(kc$krige.var))

# different ways to add the legends and pass arguments:
image(kc, ylim=c(-0.2, 1), x.leg=c(0,1), y.leg=c(-0.2, -0.1))
image(kc, val=kc$krige.var, ylim=c(-0.2, 1))
legend.krige(y.leg=c(-0.2,-0.1), x.leg=c(0,1), val=sqrt(kc$krige.var))

image(kc, ylim=c(-0.2, 1), x.leg=c(0,1), y.leg=c(-0.2, -0.1), cex=1.5)
image(kc, ylim=c(-0.2, 1), x.leg=c(0,1), y.leg=c(-0.2, -0.1), offset.leg=0.5)

image(kc, xlim=c(0, 1.2))
legend.krige(x.leg=c(1.05,1.1), y.leg=c(0,1),kc$pred, vert=TRUE)
image(kc, xlim=c(0, 1.2))
legend.krige(x.leg=c(1.05,1.1), y.leg=c(0,1),kc$pred, vert=TRUE, off=1.5, cex=1.5)

par(op)
```

`isaaks` *Data from Isaaks and Srisvastava's book*

Description

Toy example used in the book *An Introduction to Geostatistics* to illustrate the effects of different models and parameters in the kriging results when predicting at a given point.

Usage

```
data(isaaks)
```

Format

An object of the class `geodata` which is a list with the elements:

coords coordinates of the data location.

data the data vector.

x0 coordinate of the prediction point.

Source

Isaaks, E.H. & Srisvastava, R.M. (1989) An Introduction to Applied Geostatistics. Oxford University Press.

Examples

```
data(isaaks)
isaaks
summary(isaaks)
plot(isaaks$coords, asp=1, type="n")
text(isaaks$coords, as.character(isaaks$data))
points(isaaks$x0, pch="?", cex=2, col=2)
```

---

krige.bayes *Bayesian Analysis for Gaussian Geostatistical Models*

Description

The function `krige.bayes` performs Bayesian analysis of geostatistical data allowing specifications of different levels of uncertainty in the model parameters.
It returns results on the posterior distributions for the model parameters and on the predictive distributions for prediction locations (if provided).

Usage

```
krige.bayes(geodata, coords = geodata$coords, data = geodata$data,
            locations = "no", borders, model, prior, output)

model.control(trend.d = "cte", trend.l = "cte", cov.model = "matern",
              kappa = 0.5, aniso.pars = NULL, lambda = 1)

prior.control(beta.prior = c("flat", "normal", "fixed"),
              beta = NULL, beta.var.std = NULL,
              sigmasq.prior = c("reciprocal", "uniform",
                                "sc.inv.chisq", "fixed"),
              sigmasq = NULL, df.sigmasq = NULL,
              phi.prior = c("uniform", "exponential","fixed",
                            "squared.reciprocal", "reciprocal"),
              phi = NULL, phi.discrete = NULL,
              tausq.rel.prior = c("fixed", "uniform", "reciprocal"),
              tausq.rel, tausq.rel.discrete = NULL)

post2prior(obj)
```

Arguments

geodata	a list containing elements coords and data as described next. Typically an object of the class "geodata" - a **geoR** data-set. If not provided the arguments coords and data must be provided instead.
coords	an $n \times 2$ matrix where each row has the 2-D coordinates of the n data locations. By default it takes the component coords of the argument geodata, if provided.
data	a vector with *n* data values. By default it takes the component data of the argument geodata, if provided.
locations	an $N \times 2$ matrix or data-frame with the 2-D coordinates of the N prediction locations, or a list for which the first two components are used. Input is internally checked by the function check.locations. Defaults to "no" in which case the function returns only results on the posterior distributions of the model parameters.
borders	optional. If missing, by default reads the element borders from the geodata object, if present. Setting to NULL preents this behavior. If a two column matrix defining a polygon is provided the prediction is performed only at locations inside this polygon.
model	a list defining the fixed components of the model. It can take an output to a call to model.control or a list with elements as for the arguments in model.control. Default values are assumed for arguments not provided. See section DETAILS below.
prior	a list with the specification of priors for the model parameters. It can take an output to a call to prior.control or a list with elements as for the arguments in prior.control. Default values are assumed for arguments not provided. See section DETAILS below.
output	a list specifying output options. It can take an output to a call to output.control or a list with elements as for the arguments in output.control. Default values are assumed for arguments not provided. See documentation for output.control for further details.
trend.d	specifies the trend (covariates) values at the data locations. See documentation of trend.spatial for further details. Defaults to "cte".
trend.l	specifies the trend (covariates) at the prediction locations. Must be of the same type as defined for trend.d. Only used if prediction locations are provided in the argument locations.
cov.model	string indicating the name of the model for the correlation function. Further details in the documentation for cov.spatial.
kappa	additional smoothness parameter. Only used if the correlation function is one of: "matern", "powered.exponential", "cauchy" or "gneiting.matern". In the current implementation this parameter is always regarded as fixed during the Bayesian analysis.
aniso.pars	fixed parameters for geometric anisotropy correction. If aniso.pars = FALSE no correction is made, otherwise a two elements vector with values for the anisotropy parameters must be provided. Anisotropy correction consists of a

transformation of the data and prediction coordinates performed by the function coords.aniso.

lambda — numerical value of the Box-Cox transformation parameter. The value $\lambda = 1$ corresponds to no transformation. The Box-Cox parameter λ is always regarded as fixed and data transformation is performed before the analysis. Prediction results are back-transformed and returned is the same scale as for the original data. For $\lambda = 0$ the log-transformation is performed. If $\lambda < 0$ the mean predictor doesn't make sense (the resulting distribution has no expectation).

beta.prior — prior distribution for the mean (vector) parameter β. The options are "flat" (default), "normal" or "fixed" (known mean).

beta — mean hyperparameter for the distribution of the mean (vector) parameter β. Only used if beta.prior = "normal" or beta.prior = "fixed". For the later beta defines the value of the known mean.

beta.var.std — standardised (co)variance hyperparameter(s) for the prior for the mean (vector) parameter β. The (co)variance matrix forβ is given by the multiplication of this matrix by σ^2. Only used if beta.prior = "normal".

sigmasq.prior — specifies the prior for the parameter σ^2. If "reciprocal" (the default), the prior $\frac{1}{\sigma^2}$ is used. Otherwise the parameter is regarded as fixed.

sigmasq — fixed value of the sill parameter σ^2. Only used if sigmasq.prior = FALSE.

df.sigmasq — numerical. Number of degrees of freedom for the prior for the parameter σ^2. Only used if sigmasq.prior = "sc.inv.chisq".

phi.prior — prior distribution for the range parameter ϕ. Options are: "uniform", "exponential", "reciprocal" , "squared.reciprocal" and "fixed". Alternativelly, a user defined discrete distribution can be specified. In this case the argument takes a vector of numerical values of probabilities with corresponding support points provided in the argument phi.discrete.
If "fixed" the argument ϕ must be provided and is regarded as fixed when performing predictions.
For the exponential prior the argument phi must provide the value for of hyperparameter ν which corresponds to the expected value for this distribution.

phi — fixed value of the range parameter ϕ. Only needed if phi.prior = "fixed" or if phi.prior = "exponential".

phi.discrete — support points of the discrete prior for the range parameter ϕ. The default is a sequence of 51 values between 0 and 2 times the maximum distance between the data locations.

tausq.rel.prior — specifies a prior distribution for the relative nugget parameter $\frac{\tau^2}{\sigma^2}$. If tausq.rel.prior = "fixed" the relative nugget is considered known (fixed) with value given by the argument tausq.rel. If tausq.rel.prior = "uniform" a discrete uniform prior is used with support points given by the argument tausq.rel.discrete. Alternativelly, a user defined discrete distribution can be specified. In this case the argument takes the a vector of probabilities of a discrete distribution and the support points should be provided in the argument tausq.rel.discrete.

tausq.rel — fixed value for the relative nugget parameter. Only used if tausq.rel.prior = "fixed".

`tausq.rel.discrete`
: support points of the discrete prior for the relative nugget parameter $\frac{\tau^2}{\sigma^2}$.

`obj`
: an object of the class `krige.bayes` or `posterior.krige.bayes` with the output of a call to `krige.bayes`. The function `post2prior` takes the posterior distribution computed by one call to `krige.bayes` and prepares it to be used a a prior in a subsequent call. Notice that in this case the function `post2prior` is used instead of `prior.control`.

Details

`krige.bayes` is a generic function for Bayesian geostatistical analysis of (transformed) Gaussian where predictions take into account the parameter uncertainty.

It can be set to run conventional kriging methods which use known parameters or *plug-in* estimates. However, the functions `krige.conv` and `ksline` are preferable for prediction with fixed parameters.

PRIOR SPECIFICATION

The basis of the Bayesian algorithm is the discretisation of the prior distribution for the parameters ϕ and $\tau^2_{rel} = \frac{\tau^2}{\sigma^2}$. The Tech. Report (see `References` below) provides details on the results used in the current implementation.
The expressions of the implemented priors for the parameter ϕ are:

"uniform": $p(\phi) \propto 1$.

"exponential": $p(\phi) = \frac{1}{\nu}\exp(-\frac{1}{\nu} * \phi)$.

"reciprocal": $p(\phi) \propto \frac{1}{\phi}$.

"squared.reciprocal": $p(\phi) \propto \frac{1}{\phi^2}$.

"fixed": fixed known or estimated value of ϕ.

The expressions of the implemented priors for the parameter τ^2_{rel} are:

"fixed": fixed known or estimated value of τ^2_{rel}. Defaults to zero.

"uniform": $p(\tau^2_{rel}) \propto 1$.

"reciprocal": $p(\tau^2_{rel}) \propto \frac{1}{\tau^2_{rel}}$.

Apart from those a *user defined* prior can be specifyed by entering a vector of probabilities for a discrete distribution with suport points given by the argument `phi.discrete` and/or `tausq.rel.discrete`.

CONTROL FUNCTIONS

The function call includes auxiliary control functions which allows the user to specify and/or change the specification of model components (using `model.control`), prior distributions (using `prior.control`) and output options (using `output.control`). Default options are available in most of the cases.

Value

An object with class "krige.bayes" and "kriging". The attribute prediction.locations containing the name of the object with the coordinates of the prediction locations (argument locations) is assigned to the object. Returns a list with the following components:

posterior	results on on the posterior distribution of the model parameters. A list with the following possible components:
beta	summary information on the posterior distribution of the mean parameter β.
sigmasq	summary information on the posterior distribution of the variance parameter σ^2 (partial sill).
phi	summary information on the posterior distribution of the correlation parameter ϕ (range parameter) .
tausq.rel	summary information on the posterior distribution of the relative nugget variance parameter τ^2_{rel}.
joint.phi.tausq.rel	information on discrete the joint distribution of these parameters.
sample	a data.frame with a sample from the posterior distribution. Each column corresponds to one of the basic model parameters.
predictive	results on the predictive distribution at the prediction locations, if provided. A list with the following possible components:
mean	expected values.
variance	expected variance.
distribution	type of posterior distribution.
mean.simulations	mean of the simulations at each locations.
variance.simulations	variance of the simulations at each locations.
quantiles.simulations	quantiles computed from the the simulations.
probabilities.simulations	probabilities computed from the simulations.
simulations	simulations from the predictive distribution.
prior	a list with information on the prior distribution and hyper-parameters of the model parameters ($\beta, \sigma^2, \phi, \tau^2_{rel}$).
model	model specification as defined by model.control.
.Random.seed	system random seed before running the function. Allows reproduction of results. If the .Random.seed is set to this value and the function is run again, it will produce exactly the same results.
max.dist	maximum distance found between two data locations.
call	the function call.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Diggle, P.J. & Ribeiro Jr, P.J. (2002) Bayesian inference in Gaussian model-based geostatistics. Geographical and Environmental Modelling, Vol. 6, No. 2, 129-146.

The technical details about the implementation of `krige.bayes` can be found at:
Ribeiro, P.J. Jr. and Diggle, P.J. (1999) *Bayesian inference in Gaussian model-based geostatistics*. Tech. Report ST-99-08, Dept Maths and Stats, Lancaster University.
Available at: http://www.est.ufpr.br/geoR/geoRdoc/bayeskrige.pdf

Further information about **geoR** can be found at:
http://www.est.ufpr.br/geoR.

For a extended list of examples of the usage see http://www.est.ufpr.br/geoR/tutorials/examples.krige.bayes.R and/or the **geoR** tutorials page at http://www.est.ufpr.br/geoR/tutorials.

See Also

`lines.variomodel.krige.bayes`, `plot.krige.bayes` for outputs related to the parameters in the model, `image.krige.bayes` and `persp.krige.bayes` for graphical output of prediction results. `krige.conv` and `ksline` for conventional kriging methods.

Examples

```
## Not run:
# generating a simulated data-set
ex.data <- grf(50, cov.pars=c(10, .25))
#
# defining the grid of prediction locations:
ex.grid <- as.matrix(expand.grid(seq(0,1,l=21), seq(0,1,l=21)))
#
# computing posterior and predictive distributions
# (warning: the next command can be time demanding)
ex.bayes <- krige.bayes(ex.data, loc=ex.grid, prior =
                 prior.control(phi.discrete=seq(0, 2, l=21)))
#
# Prior and posterior for the parameter phi
plot(ex.bayes, type="h", tausq.rel = FALSE, col=c("red", "blue"))
#
# Plot histograms with samples from the posterior
par(mfrow=c(3,1))
hist(ex.bayes)
par(mfrow=c(1,1))

# Plotting empirical variograms and some Bayesian estimates:
# Empirical variogram
plot(variog(ex.data, max.dist = 1), ylim=c(0, 15))
# Since ex.data is a simulated data we can plot the line with the "true" model
```

```
lines(ex.data)
# adding lines with summaries of the posterior of the binned variogram
lines(ex.bayes, summ = mean, lwd=1, lty=2)
lines(ex.bayes, summ = median, lwd=2, lty=2)
# adding line with summary of the posterior of the parameters
lines(ex.bayes, summary = "mode", post = "parameters")

# Plotting again the empirical variogram
plot(variog(ex.data, max.dist=1), ylim=c(0, 15))
# and adding lines with median and quantiles estimates
my.summary <- function(x){quantile(x, prob = c(0.05, 0.5, 0.95))}
lines(ex.bayes, summ = my.summary, ty="l", lty=c(2,1,2), col=1)

# Plotting some prediction results
op <- par(no.readonly = TRUE)
par(mfrow=c(2,2))
par(mar=c(3,3,1,1))
par(mgp = c(2,1,0))
image(ex.bayes, main="predicted values")
image(ex.bayes, val="variance", main="prediction variance")
image(ex.bayes, val= "simulation", number.col=1,
      main="a simulation from the \npredictive distribution")
image(ex.bayes, val= "simulation", number.col=2,
      main="another simulation from \nthe predictive distribution")
#
par(op)
## End(Not run)
##
## For a extended list of exemples of the usage of krige.bayes()
## see http://www.est.ufpr.br/geoR/geoRdoc/examples.krige.bayes.R
##
```

`krige.conv` *Spatial Prediction – Conventional Kriging*

Description

This function performs spatial prediction for fixed covariance parameters using global neighbourhood.

Options available implement the following types of kriging: *SK* (simple kriging), *OK* (ordinary kriging), *KTE* (external trend kriging) and *UK* (universal kriging).

Usage

```
krige.conv(geodata, coords=geodata$coords, data=geodata$data,
           locations, borders, krige, output)

krige.control(type.krige = "ok", trend.d = "cte", trend.l = "cte",
```

```
obj.model = NULL, beta, cov.model, cov.pars, kappa,
nugget, micro.scale = 0, dist.epsilon = 1e-10,
aniso.pars, lambda)
```

Arguments

geodata	a list containing elements coords and data as described next. Typically an object of the class "geodata" - a **geoR** data-set. If not provided the arguments coords and data must be provided instead.
coords	an $n \times 2$ matrix or data-frame with the 2-D coordinates of the n data locations. By default it takes the component coords of the argument geodata, if provided.
data	a vector with *n* data values. By default it takes the component data of the argument geodata, if provided.
locations	an $N \times 2$ matrix or data-frame with the 2-D coordinates of the N prediction locations, or a list for which the first two components are used. Input is internally checked by the function check.locations.
borders	optional. By default reads the element borders from the geodata object, if present. Setting to NULL prevents this behavior. If a two column matrix defining a polygon is provided the prediction is performed only at locations inside this polygon.
krige	a list defining the model components and the type of kriging. It can take an output to a call to krige.control or a list with elements as for the arguments in krige.control. Default values are assumed for arguments or list elements not provided. See the description of arguments in 'krige.control' below.
output	a list specifying output options. It can take an output to a call to output.control or a list with elements as for the arguments in output.control. Default values are assumed for arguments not provided. See documentation for output.control for further details.
type.krige	type of kriging to be performed. Options are "SK", "OK" corresponding to simple or ordinary kriging. Kriging with external trend and universal kriging can be defined setting type.krige = "OK" and specifying the trend model using the arguments trend.d and trend.l.
trend.d	specifies the trend (covariate) values at the data locations. See documentation of trend.spatial for further details. Defaults to "cte".
trend.l	specifies the trend (covariate) values at prediction locations. It must be of the same type as for trend.d. Only used if prediction locations are provided in the argument locations.
obj.model	a list with the model parameters. Typically an output of likfit or variofit.
beta	numerical value of the mean (vector) parameter. Only used if type.krige="SK".
cov.model	string indicating the name of the model for the correlation function. Further details can be found in the documentation of the function cov.spatial.
cov.pars	a 2 elements vector with values of the covariance parameters σ^2 (partial sill) and ϕ (range parameter), respectively.

`kappa` additional smoothness parameter required by the following correlation functions: `"matern"`, `"powered.exponential"`, `"cauchy"` and `"gneiting.matern"`.

`nugget` the value of the nugget variance parameter τ^2. Defaults to zero.

`micro.scale` micro-scale variance. If different from zero, the nugget variance is divided into 2 terms: *micro-scale variance* and *measurement error*. This affect the precision of the predictions. Often in practice, these two variance components are indistinguishable but the distinction can be made here if justifiable. See the section `DETAILS` in the documentation of `output.control`.

`dist.epsilon` a numeric value. Locations which are separated by a distance less than this value are considered co-located.

`aniso.pars` parameters for geometric anisotropy correction. If `aniso.pars = FALSE` no correction is made, otherwise a two elements vector with values for the anisotropy parameters must be provided. Anisotropy correction consists of a transformation of the data and prediction coordinates performed by the function `coords.aniso`.

`lambda` numeric value of the Box-Cox transformation parameter. The value $\lambda = 1$ corresponds to no transformation and $\lambda = 0$ corresponds to the log-transformation. Prediction results are back-transformed and returned is the same scale as for the original data.

Details

According to the arguments provided, one of the following different types of kriging: *SK*, *OK*, *UK* or *KTE* is performed. Defaults correspond to ordinary kriging.

Value

An object of the `class` `kriging`. The attribute `prediction.locations` containing the name of the object with the coordinates of the prediction locations (argument `locations`) is assigned to the object. Returns a list with the following components:

`predict` a vector with predicted values.

`krige.var` a vector with predicted variances.

`beta.est` estimates of the β, parameter implicit in kriging procedure. Not included if `type.krige = "SK"`.

`simulations` an $ni \times n.sim$ matrix where ni is the number of prediction locations. Each column corresponds to a conditional simulation of the predictive distribution. Only returned if `n.sim > 0`.

`message` messages about the type of prediction performed.

`call` the function call.

Other results can be included depending on the options passed to `output.control`.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

output.control sets output options, image.kriging and persp.kriging for graphical output of the results, krige.bayes for Bayesian prediction and ksline for a different implementation of kriging allowing for moving neighborhood. For model fitting see likfit or variofit.

Examples

```
## Not run:
data(s100)
# Defining a prediction grid
loci <- expand.grid(seq(0,1,l=21), seq(0,1,l=21))
# predicting by ordinary kriging
kc <- krige.conv(s100, loc=loci,
                 krige=krige.control(cov.pars=c(1, .25)))
# mapping point estimates and variances
par.ori <- par(no.readonly = TRUE)
par(mfrow=c(1,2), mar=c(3.5,3.5,1,0), mgp=c(1.5,.5,0))
image(kc, main="kriging estimates")
image(kc, val=sqrt(kc$krige.var), main="kriging std. errors")
# Now setting the output to simulate from the predictive
# (obtaining conditional simulations),
# and to compute quantile and probability estimators
s.out <- output.control(n.predictive = 1000, quant=0.9, thres=2)
set.seed(123)
kc <- krige.conv(s100, loc = loci,
         krige = krige.control(cov.pars = c(1,.25)),
         output = s.out)
par(mfrow=c(2,2))
image(kc, val=kc$sim[,1], main="a cond. simul.")
image(kc, val=kc$sim[,1], main="another cond. simul.")
image(kc, val=(1 - kc$prob), main="Map of P(Y > 2)")
image(kc, val=kc$quant, main="Map of y s.t. P(Y < y) = 0.9")
par(par.ori)
## End(Not run)
```

`krweights` *Computes kriging weights*

Description

Computes the weights assign for each data point in simple and ordinary krigring

Usage

```
krweights(coords, locations, krige)
```

Arguments

coords	matrix with data coordinates
locations	matrix with coordinates of the prediciton points
krige	kriging parameters. See krige.control in krige.conv

Value

A matrix of weights

Examples

```
## Figure 8.4 in Webster and Oliver (2001), see help(wo)
data(wo)
attach(wo)
par(mfrow=c(2,2))
plot(c(-10,130), c(-10,130), ty="n", asp=1)
points(rbind(coords, x1))
KC1 <- krige.control(cov.pars=c(0.382,90.53))
w1 <- krweights(wo$coords, loc=x1, krige=KC1)
text(coords[,1], 5+coords[,2], round(w1, dig=3))
##
plot(c(-10,130), c(-10,130), ty="n", asp=1)
points(rbind(coords, x1))
KC2 <- krige.control(cov.pars=c(0.282,90.53), nug=0.1)
w2 <- krweights(wo$coords, loc=x1, krige=KC2)
text(coords[,1], 5+coords[,2], round(w2, dig=3))
##
plot(c(-10,130), c(-10,130), ty="n", asp=1)
points(rbind(coords, x1))
KC3 <- krige.control(cov.pars=c(0.082,90.53), nug=0.3)
w3 <- krweights(wo$coords, loc=x1, krige=KC3)
text(coords[,1], 5+coords[,2], round(w3, dig=3))
##
plot(c(-10,130), c(-10,130), ty="n", asp=1)
points(rbind(coords, x1))
KC4 <- krige.control(cov.pars=c(0,90.53), nug=0.382, micro=0.382)
w4 <- krweights(wo$coords, loc=x1, krige=KC4)
text(coords[,1], 5+coords[,2], round(w4, dig=3))
##
## SK vs OK
##
plot(c(-10,130), c(-10,130), ty="n", asp=1)
points(rbind(coords, x1))
KC5 <- krige.control(cov.pars=c(0.382,50))
w5 <- krweights(wo$coords, loc=x1, krige=KC5)
KC6 <- krige.control(type="sk", beta=2, cov.pars=c(0.382,50))
w6 <- krweights(wo$coords, loc=x1, krige=KC6)
```

```
text(coords[,1],  5+coords[,2], round(w5, dig=3))
text(coords[,1], -5+coords[,2], round(w6, dig=3))
##
plot(c(-10,130), c(-10,130), ty="n", asp=1)
points(rbind(coords, x1))
KC7 <- krige.control(cov.pars=c(0.382,0))
w7 <- krweights(wo$coords, loc=x1, krige=KC7)
KC8 <- krige.control(type="sk", beta=2, cov.pars=c(0.382,0))
w8 <- krweights(wo$coords, loc=x1, krige=KC8)
text(coords[,1],  5+coords[,2], round(w7, dig=3))
text(coords[,1], -5+coords[,2], round(w8, dig=3))
```

`ksline` *Spatial Prediction – Conventional Kriging*

Description

This function performs spatial prediction for given covariance parameters. Options implement the following kriging types: *SK* (simple kriging), *OK* (ordinary kriging), *KTE* (external trend kriging) and *UK* (universal kriging).

The function `krige.conv` should be preferred, unless moving neighborhood is to be used.

Usage

```
ksline(geodata, coords = geodata$coords, data = geodata$data,
       locations, borders = NULL,
       cov.model = "matern",
       cov.pars=stop("covariance parameters (sigmasq and phi) needed"),
       kappa = 0.5, nugget = 0, micro.scale = 0,
       lambda = 1, m0 = "ok", nwin = "full",
       n.samples.backtransform = 500, trend = 1, d = 2,
       ktedata = NULL, ktelocations = NULL, aniso.pars = NULL,
       signal = FALSE, dist.epsilon = 1e-10, messages)
```

Arguments

`geodata`	a list containing elements `coords` and `data` as described next. Typically an object of the `class` `"geodata"` - a **geoR** data-set. If not provided the arguments `coords` and `data` must be provided instead.
`coords`	an $n \times 2$ matrix where each row has the 2-D coordinates of the n data locations. By default it takes the component `coords` of the argument `geodata`, if provided.
`data`	a vector with n data values. By default it takes the component `data` of the argument `geodata`, if provided.
`locations`	an $N \times 2$ matrix or data-frame with the 2-D coordinates of the N prediction locations, or a list for which the first two components are used. Input is internally checked by the function `check.locations`.

`borders` optional. If a two column matrix defining a polygon is provided the prediction is performed only at locations inside this polygon.

`cov.pars` a vector with 2 elements or an $n \times 2$ matrix with the covariance parameters σ^2 (partial sill) and ϕ (range parameter). If a vector, the elements are the values of σ^2 and ϕ, respectively. If a matrix, corresponding to a model with several structures, the values of σ^2 are in the first column and the values of ϕ are in the second.

`nugget` the value of the nugget variance parameter τ^2. Defaults to zero.

`micro.scale` micro-scale variance. If different from zero, the nugget variance is divided into 2 terms: *micro-scale variance* and *measurement error*. This might affect the precision of the predictions. In practice, these two variance components are usually indistinguishable but the distinction can be made here if justifiable.

`cov.model` string indicating the name of the model for the correlation function. Further details in the documentation for `cov.spatial`. Defaults are equivalent to the *exponential* model.

`kappa` additional smoothness parameter required by the following correlation functions: `"matern"`, `"powered.exponential"`, `"cauchy"` and `"gneiting.matern"`.

`lambda` numeric value of the Box-Cox transformation parameter. The value $\lambda = 1$ corresponds to no transformation and $\lambda = 0$ corresponds to the log-transformation. Prediction results are back-transformed and returned is the same scale as for the original data.

`m0` The default value `"ok"` indicates that ordinary kriging will be performed. Other options are `"kt"` for kriging with a trend model (universal kriging) and `"kte"` for kriging with external trend (covariates). If a numeric value is provided it is assumed to be the value of a know mean and simple kriging is then performed. If `"av"` the arithmetic mean of the data is assumed to be the know mean for simple kriging algorithm.

`nwin` If `"full"` *global neighborhood* is used i.e., all data values are used in the prediction of every prediction location. An integer number defines the *moving neighborhood* algorithm. The number provided is used as the number closest neighbors to be used for the prediction at each location. Defaults to `"full"`.

`n.samples.backtransform` number of samples used in the back-transformation. When transformations are used (specified by an argument `lambda`), back-transformations are usually performed by sampling from the predictive distribution and then back-transforming the sampled values. The exceptions are for $\lambda = 0$ (log-transformation) and $\lambda = 1$ (no transformation).

`trend` only required if `m0 = "kt"` (universal kriging). Possible values are 1 or 2, corresponding to a first or second degree polynomial trend on the coordinates, respectively.

`d` spatial dimension, 1 defines a prediction on a line, 2 on a plane (the default).

`ktedata` only required if `m0 = "kte"`. A vector or matrix with the values of the external trend (covariates) at the data locations.

`ktelocations` only required if `m0 = "kte"`. A vector or matrix with the values of the external trend (covariates) at the prediction locations.

aniso.pars	parameters for geometric anisotropy correction. If aniso.pars = FALSE no correction is made, otherwise a two elements vector with values for the anisotropy parameters must be provided. Anisotropy correction consists of a transformation of the data and prediction coordinates performed by the function coords.aniso.
signal	logical. If TRUE the signal is predicted, otherwise the variable is predicted. If no transformation is performed the expectations are the same in both cases and the difference is only for values of the kriging variance, if the value of the nugget is different from zero.
dist.epsilon	a numeric value. Points which are separated by a distance less than this value are considered co-located.
messages	logical. Indicates whether or not status messages are printed on the screen (or other output device) while the function is running.

Value

An object of the class kriging which is a list with the following components:

predict	the predicted values.
krige.var	the kriging variances.
dif	the difference between the predicted value and the global mean. Represents the contribution to the neighboring data to the prediction at each point.
summary	values of the arithmetic and weighted mean of the data and standard deviations. The weighted mean corresponds to the estimated value of the global mean.
ktrend	the matrix with trend if m0 = "kt" (universal kriging).
ktetrend	the matrix with trend if m0 = "kte" (external trend kriging).
beta	the value of the mean which is implicitly estimated for m0 = "ok", "kte" or "kt".
wofmean	weight of mean. The predicted value is an weighted average between the global mean and the values at the neighboring locations. The value returned is the weight of the mean.
locations	the coordinates of the prediction locations.
message	status messages returned by the algorithm.
call	the function call.

Note

This is a preliminary and inefficient function implementing kriging methods. For predictions using global neighborhood the function krige.conv should be used instead.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

krige.conv for a more efficient implementation of conventional kriging methods,
krige.bayes for Bayesian prediction.

Examples

```
data(s100)
loci <- expand.grid(seq(0,1,l=31), seq(0,1,l=31))
kc <- ksline(s100, loc=loci, cov.pars=c(1, .25))
par(mfrow=c(1,2))
image(kc, main="kriging estimates")
image(kc, val=sqrt(kc$krige.var), main="kriging std. errors")
```

---

legend.krige *Add a legend to a image with kriging results*

Description

This function allows adds a legend to an image plot generated by image.kriging or image.krige.bayes. It can be called internally by these functions or directly by the user.

Usage

```
legend.krige(x.leg, y.leg, values, scale.vals,
             vertical = FALSE, offset.leg = 1, ...)
```

Arguments

x.leg	limits for the legend in the x direction.
y.leg	limits for the legend in the y direction.
values	values plotted in the image.
scale.vals	optional. Values to appear in the legend. If not provided the function pretty is used to define the values.
vertical	If TRUE the legend is drawn in the vertical direction. Defaults to FALSE.
offset.leg	numeric value controlling the distance between the legend text and the legend box.
...	further arguments to be passed to the function text.

Value

A legend is added to the current plot. No values are returned.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

image.kriging, image.krige.bayes.

Examples

```
# See examples in the documentation for image.kriging
```

---

likfit *Likelihood Based Parameter Estimation for Gaussian Random Fields*

Description

Maximum likelihood (ML) or *restricted maximum likelihood* (REML) parameter estimation for (transformed) Gaussian random fields.

Usage

```
likfit(geodata, coords = geodata$coords, data = geodata$data,
       trend = "cte", ini.cov.pars, fix.nugget = FALSE, nugget = 0,
       fix.kappa = TRUE, kappa = 0.5, fix.lambda = TRUE, lambda = 1,
       fix.psiA = TRUE, psiA = 0, fix.psiR = TRUE, psiR = 1,
       cov.model = "matern", realisations,
       method.lik = "ML", components = FALSE,
       nospatial = TRUE, limits = pars.limits(),
       print.pars = FALSE, messages, ...)

## S3 method for class 'likGRF':
fitted(object, spatial = TRUE, ...)

## S3 method for class 'likGRF':
resid(object, spatial = FALSE, ...)
```

Arguments

`geodata`	a list containing elements `coords` and `data` as described next. Typically an object of the class `"geodata"`. If not provided the arguments `coords` and `data` must be provided instead.
`coords`	an $n \times 2$ matrix where each row has the 2-D coordinates of the n data locations. By default it takes the component `coords` of the argument `geodata`, if provided.
`data`	a vector with n data values. By default it takes the component `data` of the argument `geodata`, if provided.
`trend`	specifies the mean part of the model. See documentation of `trend.spatial` for further details. Defaults to `"cte"`.
`ini.cov.pars`	initial values for the covariance parameters: σ^2 (partial sill) and ϕ (range parameter). Typically a vector with two components. However a matrix can be used to provide several initial values. See `DETAILS` below.
`fix.nugget`	logical, indicating whether the parameter τ^2 (nugget variance) should be regarded as fixed (`fix.nugget = TRUE`) or should be estimated (`fix.nugget = FALSE`). Defaults to `FALSE`.
`nugget`	value of the nugget parameter. Regarded as a fixed value if `fix.nugget = TRUE` otherwise as the initial value for the minimisation algorithm. Defaults to zero.
`fix.kappa`	logical, indicating whether the extra parameter κ should be regarded as fixed (`fix.kappa = TRUE`) or should be estimated (`fix.kappa = FALSE`). Defaults to `TRUE`.
`kappa`	value of the extra parameter κ. Regarded as a fixed value if `fix.kappa = TRUE` otherwise as the initial value for the minimisation algorithm. Defaults to 0.5. This parameter is valid only if the covariance function is one of: `"matern"`, `"powered.exponential"`, `"cauchy"` or `"gneiting.matern"`. For more details on covariance functions see documentation for `cov.spatial`.
`fix.lambda`	logical, indicating whether the Box-Cox transformation parameter λ should be regarded as fixed (`fix.lambda = TRUE`) or should be be estimated (`fix.lambda = FALSE`). Defaults to `TRUE`.
`lambda`	value of the Box-Cox transformation parameter λ. Regarded as a fixed value if `fix.lambda = TRUE` otherwise as the initial value for the minimisation algorithm. Defaults to 1. Two particular cases are $\lambda = 1$ indicating no transformation and $\lambda = 0$ indicating log-transformation.
`fix.psiA`	logical, indicating whether the anisotropy angle parameter ψ_R should be regarded as fixed (`fix.psiA = TRUE`) or should be estimated (`fix.psiA = FALSE`). Defaults to `TRUE`.
`psiA`	value (in radians) for the anisotropy angle parameter ψ_A. Regarded as a fixed value if `fix.psiA = TRUE` otherwise as the initial value for the minimisation algorithm. Defaults to 0. See `coords.aniso` for further details on anisotropy correction.
`fix.psiR`	logical, indicating whether the anisotropy ratio parameter ψ_R should be regarded as fixed (`fix.psiR = TRUE`) or should be estimated (`fix.psiR = FALSE`). Defaults to `TRUE`.

psiR	value, always greater than 1, for the anisotropy ratio parameter ψ_R. Regarded as a fixed value if fix.psiR = TRUE otherwise as the initial value for the minimisation algorithm. Defaults to 1. See coords.aniso for further details on anisotropy correction.
cov.model	a string specifying the model for the correlation function. For further details see documentation for cov.spatial. Defaults are equivalent to the *exponential* model.
realisations	optional. A vector indicating the number of replication for each datum. For further information see DETAILS below and documentation for as.geodata.
method.lik	options are "ML" for maximum likelihood and "REML" for restricted maximum likelihood. Defaults to "ML".
components	an $n \times 3$ data-frame with fitted values for the three model components: trend, spatial and residuals. See the section DETAILS below for the model specification.
nospatial	logical. If TRUE parameter estimates for the model without spatial component are included in the output.
limits	values defining lower and upper limits for the model parameters used in the numerical minimisation. The auxiliary function pars.limits is called to set the limits. See also **Limits** in DETAILS below.
print.pars	logical. If TRUE the parameters and the value of the negative log-likelihood (up to a constant) are printed each time the function to be minimised is called.
messages	logical. Indicates whether status messages should be printed on the screen (or output device) while the function is running.
...	additional parameters to be passed to the minimisation function. Typically arguments of the type control() which controls the behavior of the minimisation algorithm. For further details see documentation for the minimisation function optim.
object	an object with output of the function likfit.
spatial	logical, determines whether the spatial component of the model in included in the output. The geostatistical model components are: *trend*, *spatial* and *residulas*. See DETAILS.

Details

This function estimate the parameters of the Gaussian random field model, specified as:

$$Y(x) = \mu(x) + S(x) + e$$

where

- x defines a spatial location. Typically Euclidean coordinates on a plane.
- Y is the variable been observed.
- $\mu(x) = X\beta$ is the mean component of the model (trend).
- $S(x)$ is a stationary Gaussian process with variance σ^2 (partial sill) and a correlation function parametrized in its simplest form by ϕ (the range parameter). Possible extra parameters for the correlation function are the smoothness parameter κ and the anisotropy parameters ϕ_R and ϕ_A (anisotropy ratio and angle, respectively).

- e is the error term with variance parameter τ^2 (nugget variance).

The additional parameter λ allows for the Box-Cox transformation of the response variable. If used (i.e. if $\lambda \neq 1$) $Y(x)$ above is replaced by $g(Y(x))$ such that

$$g(Y(x)) = \frac{Y^{\lambda}(x) - 1}{\lambda}.$$

Two particular cases are $\lambda = 1$ which indicates no transformation and $\lambda = 0$ indicating the log-transformation.

Numerical minimization

In general parameter estimation is performed numerically using the R function `optim` to minimise the negative log-likelihood computed by the function `negloglik.GRF`. If the nugget, anisotropy (ψ_A, ψ_R), smoothness (κ) and transformation (λ) parameters are held fixed then the numerical minimisation can be reduced to one-dimension and the function `optimize` is used instead of `optim`. In this case initial values are irrelevant.

Limits

Lower and upper limits for parameter values can be individually specified using the function `link{pars.limits}`. For example, including the following in the function call:
`limits = pars.limits(phi=c(0, 10), lambda=c(-2.5, 2.5)),`
will change the limits for the parameters ϕ and λ. Default values are used if the argument `limits` is not provided.

There are internal reparametrisation depending on the options for parameters to be estimated. For instance for the common situation when `fix.nugget=FALSE` the minimisation is performed in a reduced parameter space using $\tau^2_{rel} = \frac{\tau^2}{\sigma^2}$. In this case values of σ^2 and β are then given by analytical expressions which are function of the two parameters remaining parameters and limits for these two parameters will be ignored.

Since parameter values are found by numerical optimization using the function `optim`, in given circunstances the algorithm may not converge to correct parameter values when called with default options and the user may need to pass extra options for the optimizer. For instance the function `optim` takes a `control` argument. The user should try different initial values and if the parameters have different orders of magnitude may need to use options to scale the parameters. Some possible workarounds in case of problems include:

- rescale you data values (dividing by a constant, say)
- rescale your coordinates (subtracting values and/or dividing by constants)
- Use the mechanism to pass `control()` options for the optimiser internally

Transformation If the `fix.lambda = FALSE` and `nospatial = FALSE` the Box-Cox parameter for the model without the spatial component is obtained numerically, with log-likelihood computed by the function `boxcox.ns`.

Multiple initial values can be specified providing a n matrix for the argument `ini.cov.pars` and/or providing a vector for the values of the remaining model parameters. In this case the log-likelihood is computed for all combinations of the model parameters. The parameter set which maximises the value of the log-likelihood is then used to start the minimisation algorithm.

Alternatively the argument `ini.cov.pars` can take an object of the class `eyefit` or `variomodel`. This allows the usage of an output of the functions `eyefit`, `variofit` or `likfit` be used as initial value.

The argument **realisations** allows replicated data to be used. For instance, data collected at different times at least partially the same locations can be pooled together in the parameter estimation if independence is assumed between time points. The argument `realisations` takes a vector indicating the replication number (e.g. the times). The log-likelihoods are computed for each replication and added together.
Notice that this assumes independence among the replications.

Value

An object of the classes `"likGRF"` and `"variomodel"`.
The function `summary.likGRF` is used to print a summary of the fitted model.
The object is a list with the following components:

`cov.model`	a string with the name of the correlation function.
`nugget`	value of the nugget parameter τ^2. This is an estimate if `fix.nugget = FALSE` otherwise, a fixed value.
`cov.pars`	a vector with the estimates of the parameters σ^2 and ϕ, respectively.
`kappa`	value of the smoothness parameter. Valid only if the correlation function is one of: `"matern"`, `"powered.exponential"`, `"cauchy"` or `"gneiting.matern"`.
`beta`	estimate of mean parameter β. This can be a scalar or vector depending on the trend (covariates) specified in the model.
`beta.var`	estimated variance (or covariance matrix) for the mean parameter β.
`lambda`	values of the Box-Cox transformation parameter. A fixed value if `fix.lambda = TRUE` otherwise the estimate value.
`aniso.pars`	fixed values or estimates of the anisotropy parameters, according to the function call.
`method.lik`	estimation method used, `"ML"` (maximum likelihood) or `"REML"` (restricted maximum likelihood).
`loglik`	the value of the maximized likelihood.
`npars`	number of estimated parameters.
`AIC`	value of the Akaike Information Criteria, $AIC = -2ln(L) + 2p$ where L is the maximised likelihood and p is the number of parameters in the model.
`BIC`	value of the Bayesian Information Criteria, $BIC = -2ln(L) + plog(n)$, where n is the number of data, L, p as for AIC above.
`parameters.summary`	a data-frame with all model parameters, their status (estimated or fixed) and values.
`info.minimisation`	results returned by the minimisation function.
`max.dist`	maximum distance between 2 data points. This information relevant for other functions which use outputs from `likfit`.
`trend`	the trend (covariates) matrix X.
`log.jacobian`	numerical value of the logarithm of the Jacobian of the transformation.
`nospatial`	estimates for the model without the spatial component.
`call`	the function call.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

summary.likGRF for summary of the results, plot.variogram, lines.variogram and lines.variomodel for graphical output, proflik for computing profile likelihoods, variofit and for other estimation methods, and optim for the numerical minimisation function.

Examples

```
## Not run:
data(s100)
ml <- likfit(s100, ini=c(0.5, 0.5), fix.nug = TRUE)
ml
summary(ml)
reml <- likfit(s100, ini=c(0.5, 0.5), fix.nug = TRUE, met = "REML")
summary(reml)
plot(variog(s100))
lines(ml)
lines(reml, lty = 2)
## End(Not run)
```

---

likfitBGCCM *Fits the bivariate Gaussian common component geostatistical model*

Description

Computes maximum likelihood estimates of the bivariate Gaussian common component geostatistical model.

Usage

```
likfitBGCCM(geodata1, geodata2, ini.sigmasq, ini.phi,
            cov0.model="matern", cov1.model="matern", cov2.model="matern",
            kappa0=0.5, kappa1=0.5, kappa2=0.5,
            fc.min = c("optim", "nlminb"), ...)
```

Arguments

geodata1	an object of the class geodata with the data of the first variable.
geodata2	an object of the class geodata with the data of the first variable.
ini.sigmasq	optional, a vector with initial values for the correlation range parameters. If not provided default values are used.
ini.phi	optional, a vector with initial values for the correlation range parameters. If not provided default values are used.
cov0.model, cov1.model, cov2.model	covariance model for each of the processes. See cov.spatial for details.
kappa0, kappa1, kappa2	extra parameter for some covariance models.
fc.min	a string indication which function should be used to minimise the negative of the log-likelihood.
...	further arguments to be passed to optim or nlminb.

Value

A list with model fitting information to which the class BGCCM is assigned.

mu	a 2 elements vector with estimated variance values.
sigmasq	a 4 elements vector with estimated mean values.
phi	a 3 elements vector with estimated correlation parameters values.
loglik	a scalar. Maximised value of the log-likelihood.
optim	results returned by optim or nlminb.
...	and other information related to the model fitting.

Warning

This is a new function and still in draft format and pretty much untested.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

See Also

optim, nlminb, varcovBGCCM, as.geodata, likfit.

Examples

```
# see http://www.est.ufpr.br/geoR/tutorials/CCM.R
```

`lines.variomodel.krige.bayes` *Adds a Bayesian Estimate of the Variogram to a Plot*

Description

Adds a Bayesian estimate of the variogram model to a plot typically with an empirical variogram. The estimate is a chosen summary (mean, mode or mean) of the posterior distribution returned by the function `krige.bayes`.

Usage

```
## S3 method for class 'krige.bayes':
lines.variomodel(x, summary.posterior, max.dist, uvec,
                 posterior = c("variogram", "parameters"),  ...)
```

Arguments

`x`	an object of the class `krige.bayes`, typically an output of the function `krige.bayes`.
`summary.posterior`	specify which summary of the posterior distribution should be used as the parameter estimate. Options are `"mean"`, `"median"` or `"mode"`. See `DETAILS` below.
`max.dist`	numerical, the maximum distance for the x-axis.
`uvec`	a numerical vector with support points to compute the variogram values. Only used if `posterior = "variogram"`. Defaults to `seq(0, max.dist, length = 51)`.
`posterior`	indicates whether the the variogram line is based on the posterior of the variogram function (default) or the posterior of the model parameters.
`...`	arguments passed to the functions `lines` or `curve`.

Details

The function `krige.bayes` returns samples from the posterior distribution of the parameters $(\sigma^2, \phi, \tau^2_{rel})$.

This function allows for two basic options to draw a line with a summary of the variogram function.

1. "variogram": for each sample of the parameters the variogram function is computed at the support points defined in the argument `uvec`. Then a function provided by the user in the argument `summary.posterior` is used to compute a summary of the values obtained at each support point.
2. "parameters": in this case summaries of the posterior distribution of the model parameters as "plugged-in" in the variogram function. One of the options `"mode"` (default) ,`"median"` or `"mean"` can be provided in the argument `summary.posterior`. The option `mode`, uses the mode of (ϕ, τ^2_{rel}) and the mode of of σ^2 conditional on the modes of the former parameters. For the options `mean` and `median` these summaries are computed from the samples of the posterior.

Value

A line with the estimated variogram plot is added to the plot in the current graphics device. No values are returned.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`lines.variomodel`, `krige.bayes` and `lines`.

Examples

```
#See examples in the documentation of the function krige.bayes().
```

---

lines.variogram *Line with a Empirical Variogram*

Description

A sample (empirical) variogram computed using the function `variog` is added to the current plot.

Usage

```
## S3 method for class 'variogram':
lines(x, max.dist, type = "o",  scaled = FALSE,
         pts.range.cex, ...)
```

Arguments

`x`	an object of the class `"variogram"`, typically an output from the function `variog`.
`max.dist`	maximum distance for the x-axis. By default takes the maximum distance for which the sample variogram was computed.
`type`	type of line for the empirical variogram. The default is `"o"` (dots and lines). See documentation for `lines` for further details.
`scaled`	logical. If `TRUE` the variogram values are divided by the sample variance. This allows comparison between variograms of different variables.

`pts.range.cex`
: optional. A two elements vector with maximum and minimum values for the caracter expansion factor `cex`. If provided the point sizes in binned variogram are proportional to the number of pairs of points used to compute each bin.

`...`
: other arguments to be passed to the function `lines`.

Value

A line with the empirical variogram is added to the plot in the current graphics device. No values are returned.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`variog`, `lines.variogram`, `lines.variomodel`, `variog.model.env`, `variog.mc.env`, `plot.grf`, `lines`.

---

lines.variogram.envelope

Adds Envelopes Lines to a Variogram Plot

Description

Variogram envelopes computed by `variog.model.env` or `variog.mc.env` are added to the current variogram plot.

Usage

```
## S3 method for class 'variogram.envelope':
lines(x, lty = 3, ...)
```

Arguments

`x`
: an object of the class `"variogram.envelope"`, typically an output of the functions `variog.model.env` or `variog.mc.env`.

`lty`
: line type. Defaults to 3.

`...`
: arguments to be passed to the function `lines`.

Value

Lines defining the variogram envelope are added to the plotin the current graphics device.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

variog for variogram computation, variog.model.env and variog.mc.env for computation of variogram envelopes, and lines for the generic function.

Examples

```
if(is.R()) data(s100)
s100.vario <- variog(s100, max.dist = 1)
s100.ml <- likfit(s100, ini=c(.5, .5))
s100.mod.env <- variog.model.env(s100, obj.variog = s100.vario,
   model = s100.ml)
s100.mc.env <- variog.mc.env(s100, obj.variog = s100.vario)
plot(s100.vario)
lines(s100.mod.env)
lines(s100.mc.env, lwd=2)
```

---

lines.variomodel *Adds a Line with a Variogram Model to a Variogram Plot*

Description

This function adds a line with a variogram model specifyed by the user to a current variogram plot. The variogram is specifyed either by passing a list with values for the variogram elements or using each argument in the function.

Usage

```
## S3 method for class 'variomodel':
lines(x, ...)
## Default S3 method:
lines.variomodel(x, cov.model, cov.pars, nugget, kappa,
                         max.dist, scaled = FALSE, ...)
```

Arguments

x	a list with the values for the following components: cov.model, cov.pars, nugget, kappa , max.dist; or a numeric vector with x-axis values for the variogram. This argument is not required if the other arguments in the function are provided, otherwise is compulsory. If a list is provided the arguments which match the list elements are ignored.
cov.model	a string with the type of the variogram function. See documentation of cov.spatial for further details.
cov.pars	a vector or matrix with the values for the partial sill (σ^2) and range (ϕ) parameters.
nugget	a scalar with the value of the nugget (τ^2) parameter.
kappa	a scalar with the value of the smoothness (κ) parameters. Only required if cov.model is one of the following: "matern", "powered.exponential", "cauchy" and "gneiting.matern"
max.dist	maximum distance (x-axis) to compute and draw the line representing the variogram model. If a list is provided in x the default is the distance given by x$max.dist. If a vector is provided in x it takes max(x).
scaled	logical. If TRUE the total sill in the plot is equals to 1.
...	arguments to be passed to the function curve.

Details

Adds a line with a variogram model to a plot. In conjuction with plot.variogram can be used for instance to compare sample variograms against fitted models returned by variofit and/or likfit.

Value

A line with a variogram model is added to a plot on the current graphics device. No values are returned.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

lines.variomodel.krige.bayes, lines.variomodel.grf, lines.variomodel.variofit, lines.variomodel.likGRF, plot.variogram, lines.variogram, variofit, likfit, curve.

Examples

```
data(s100)
# computing and ploting empirical variogram
vario <- variog(s100, max.dist = 1)
plot(vario)
# estimating parameters by weighted least squares
vario.wls <- variofit(vario, ini = c(1, .3), fix.nugget = TRUE)
# adding fitted model to the plot
lines(vario.wls)
#
# Ploting different variogram models
plot(0:1, 0:1, type="n")
lines.variomodel(cov.model = "exp", cov.pars = c(.7, .25), nug = 0.3, max.dist = 1)
# an alternative way to do this is:
my.model <- list(cov.model = "exp", cov.pars = c(.7, .25), nugget = 0.3,
max.dist = 1)
lines.variomodel(my.model, lwd = 2)
# now adding another model
lines.variomodel(cov.m = "mat", cov.p = c(.7, .25), nug = 0.3,
                 max.dist = 1, kappa = 1, lty = 2)
# adding the so-called "nested" models
# two exponential structures
lines.variomodel(seq(0,1,l=101), cov.model="exp",
                 cov.pars=rbind(c(0.6,0.15),c(0.4,0.25)), nug=0, col=2)
## exponential and spherical structures
lines.variomodel(seq(0,1,l=101), cov.model=c("exp", "sph"),
                 cov.pars=rbind(c(0.6,0.15), c(0.4,0.75)), nug=0, col=3)
```

---

lines.variomodel.grf

Lines with True Variogram for Simulated Data

Description

This functions adds to the graphics device a line with the theoretical (true) variogram used when generating simulations with the function `grf`.

Usage

```
## S3 method for class 'grf':
lines.variomodel(x, max.dist = max(dist(x$coords)),
length = 100, lwd = 2, ...)
```

Arguments

`x`	an object from the class `grf` typically an output of the function `grf`.
`max.dist`	maximum distance to compute and plot the true variogram. Defaults to the maximum distance between two data locations.

`length`	number of points used to compute and draw the variogram line.
`lwd`	width of the line.
`...`	further arguments to be passed to the function `curve`.

Value

A line with the true variogram model is added to the current plot on the graphics device. No values are returned.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`lines.variomodel`, `grf`, `plot.grf`, `curve`.

Examples

```
sim <- grf(100, cov.pars=c(1, .25)) # simulates data
plot(variog(sim, max.dist=1))       # plot empirical variogram
```

---

lines.variomodel.likGRF

Adds a Variogram Line to a Variogram Plot

Description

This function adds a fitted variogram based on the estimates of the model parameters returned by the function `likfit` to a current variogram plot.

Usage

```
## S3 method for class 'likGRF':
lines.variomodel(x, max.dist, scaled = FALSE, ...)
```

Arguments

x	an object of the class likGRF which is a list containing information about the fitted model parameters, typically an output of the function likfit.
max.dist	maximum distance (x-axis) to compute and draw the line representing the variogram model. The default is the distance given by obj$max.dist.
scaled	logical. If TRUE the total sill in the plot is equals to 1.
...	arguments to be passed to the function curve.

Details

Adds variogram model(s) to a plot. In conjuction with plot.variogram can be used to compare sample variograms against fitted models returned by likfit.

Value

A line with a variogram model is added to a plot on the current graphics device. No values are returned.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

lines.variomodel, lines.variomodel.variofit, plot.variogram, lines.variogram, variofit, likfit, curve.

Examples

```
data(s100)
# compute and plot empirical variogram
vario <- variog(s100, max.dist = 1)
plot(vario)
# estimate parameters
vario.ml <- likfit(s100, ini = c(1, .3), fix.nugget = TRUE)
# adds fitted model to the plot
lines(vario.ml)
```

`lines.variomodel.variofit`
Adds a Line with a Fitted Variogram Model to a Variogram Plot

Description

This function adds a line with the variogram model fitted by the function `variofit` to a current variogram plot.

Usage

```
## S3 method for class 'variofit':
lines.variomodel(x, max.dist, scaled = FALSE, ...)
```

Arguments

`x`	an object of the class `variofit` which is a list containing information about the fitted model parameters, typically an output of the function `variofit`.
`max.dist`	maximum distance (x-axis) to compute and draw the line representing the variogram model. The default is the distance given by `x$max.dist`.
`scaled`	logical. If `TRUE` the total sill in the plot is set to 1.
`...`	arguments to be passed to the function `curve`.

Details

Adds fitted variogram model to a plot. In conjuction with `plot.variogram` can be used to compare empirical variograms against fitted models returned by `variofit`.

Value

A line with a variogram model is added to a plot on the current graphics device. No values are returned.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`lines.variomodel`, `lines.variomodel.likGRF`, `plot.variogram`, `lines.variogram`, `variofit`, `likfit`, `curve`.

Examples

```
data(s100)
# compute and plot empirical variogram
vario <- variog(s100, max.dist = 1)
plot(vario)
# estimate parameters
vario.wls <- variofit(vario, ini = c(1, .3), fix.nugget = TRUE)
# adds fitted model to the plot
lines(vario.wls)
```

---

locations.inside *Select prediction locations inside borders*

Description

Selects the prediction locations located inside a polygon defining borders of a region where prediction is aimed. Typically internally called by **geoR** functions `krige.bayes`, `krige.conv`, `ksline`.

Usage

```
locations.inside(locations, borders, as.is = TRUE, ...)
```

Arguments

`locations`	a two columns matrix or dqata frame with coordinates of the prediction locations.
`borders`	a two column matrix or data-frame with coordinates of a polygon defining the borders of the region.
`as.is`	logical defining if the returned object of of the same type (list, data-frame or matrix) as the provided in `locations`. If `FALSE` the function returns a matrix.
`...`	arguments to be passed to the internal function `.geoR_pip` and currently not used.

Value

A two columns matrix, data-frame or a list with 2 elements with coordinates of points inside the borders.

See Also

`overlay`,`coordinates`, `SpatialPoints`.

Examples

```
data(parana)
gr <- pred_grid(parana$borders, by=20)
plot(gr, asp=1, pch="+")
polygon(parana$borders)
gr.in <- locations.inside(gr, parana$borders)
points(gr.in, col=2, pch="+")
```

---

loglik.GRF *Log-Likelihood for a Gaussian Random Field*

Description

This function computes the value of the log-likelihood for a Gaussian random field.

Usage

```
loglik.GRF(geodata, coords = geodata$coords, data = geodata$data,
           obj.model = NULL, cov.model = "exp", cov.pars, nugget = 0,
           kappa = 0.5, lambda = 1, psiR = 1, psiA = 0,
           trend = "cte", method.lik = "ML", compute.dists = TRUE,
           realisations = NULL)
```

Arguments

`geodata`	a list containing elements `coords` and `data` as described next. Typically an object of the class `"geodata"` - a **geoR** data-set. If not provided the arguments `coords` and `data` must be provided instead.
`coords`	an $n \times 2$ matrix, each row containing Euclidean coordinates of the n data locations. By default it takes the element `coords` of the argument `geodata`.
`data`	a vector with data values. By default it takes the element `data` of the argument `geodata`.
`obj.model`	a object of the class `variomodel` with a fitted model. Tipically an output of `likfit` or `variofit`.
`cov.model`	a string specifying the model for the correlation function. For further details see documentation for `cov.spatial`.
`cov.pars`	a vector with 2 elements with values of the covariance parameters σ^2 (partial sill) and ϕ (range parameter).
`nugget`	value of the nugget parameter. Defaults to 0.
`kappa`	value of the smoothness parameter. Defaults to 0.5.
`lambda`	value of the Box-Cox tranformation parameter. Defaults to 1.
`psiR`	value of the anisotropy ratio parameter. Defaults to 1, corresponding to isotropy.
`psiA`	value (in radians) of the anisotropy rotation parameter. Defaults to zero.

`trend` specifies the mean part of the model. The options are: `"cte"` (constant mean), `"1st"` (a first order polynomial on the coordinates), `"2nd"` (a second order polynomial on the coordinates), or a formula of the type `~X` where `X` is a matrix with the covariates (external trend). Defaults to `"cte"`.

`method.lik` options are `"ML"` for likelihood and `"REML"` for restricted likelihood. Defaults to `"ML"`.

`compute.dists` for internal use with other function. Don't change the default unless you know what you are doing.

`realisations` optional. A vector indicating replication number for each data. For more details see `as.geodata`.

Details

The expression log-likelihood is:

$$l(\theta) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(y - F\beta)'\Sigma^{-1}(y - F\beta),$$

where n is the size of the data vector y, β is the mean (vector) parameter with dimention p, Σ is the covariance matrix and F is the matrix with the values of the covariates (a vector of 1's if the mean is constant.

The expression restricted log-likelihood is:

$$rl(\theta) = -\frac{n-p}{2}\log(2\pi) + \frac{1}{2}\log|F'F| - \frac{1}{2}\log|\Sigma| - \frac{1}{2}\log|F'\Sigma F| - \frac{1}{2}(y - F\beta)'\Sigma^{-1}(y - F\beta).$$

Value

The numerical value of the log-likelihood.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`likfit` for likelihood-based parameter estimation.

Examples

```
data(s100)
loglik.GRF(s100, cov.pars=c(0.8, .25), nugget=0.2)
loglik.GRF(s100, cov.pars=c(0.8, .25), nugget=0.2, met="RML")

## Computing the likelihood of a model fitted by ML
s100.ml <- likfit(s100, ini=c(1, .5))
s100.ml
s100.ml$loglik
loglik.GRF(s100, obj=s100.ml)

## Computing the likelihood of a variogram fitted model
s100.v <- variog(s100, max.dist=1)
s100.vf <- variofit(s100.v, ini=c(1, .5))
s100.vf
loglik.GRF(s100, obj=s100.vf)
```

---

matern *Computer Values of the Matern Correlation Function*

Description

This function computes values of the Matérn correlation function for given distances and parameters.

Usage

```
matern(u, phi, kappa)
```

Arguments

u	a vector, matrix or array with values of the distances between pairs of data locations.
phi	value of the range parameter ϕ.
kappa	value of the smoothness parameter κ.

Details

The Matérn model is defined as:

$$\rho(u;\phi,\kappa) = \{2^{\kappa-1}\Gamma(\kappa)\}^{-1}(u/\phi)^{\kappa}K_{\kappa}(u/\phi)$$

where ϕ and κ are parameters and $K_{\kappa}(\cdot)$ denotes the modified Bessel function of the third kind of order κ. The family is valid for $\phi > 0$ and $\kappa > 0$.

Value

A vector matrix or array, according to the argument u, with the values of the Matérn correlation function for the given distances.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

See Also

`cov.spatial` for the correlation functions implemented in **geoR**, and `besselK` for computation of the Bessel functions.

Examples

```
#
# Models with fixed range and varying smoothness parameter
#
curve(matern(x, phi= 0.25, kappa = 0.5),from = 0, to = 1,
      xlab = "distance", ylab = expression(rho(h)), lty = 2,
      main=expression(paste("varying  ", kappa, "  and fixed  ", phi)))
curve(matern(x, phi= 0.25, kappa = 1),from = 0, to = 1, add = TRUE)
curve(matern(x, phi= 0.25, kappa = 2),from = 0, to = 1, add = TRUE,
      lwd = 2, lty=2)
curve(matern(x, phi= 0.25, kappa = 3),from = 0, to = 1, add = TRUE,
      lwd = 2)
legend(0.6,1, c(expression(kappa == 0.5), expression(kappa == 1),
    expression(kappa == 2), expression(kappa == 3)),
    lty=c(2,1,2,1), lwd=c(1,1,2,2))
#
# Correlations with equivalent "practical range"
# and varying smoothness parameter
#
curve(matern(x, phi = 0.25, kappa = 0.5),from = 0, to = 1,
      xlab = "distance", ylab = expression(gamma(h)), lty = 2,
      main = "models with equivalent \"practical\" range")
curve(matern(x, phi = 0.188, kappa = 1),from = 0, to = 1, add = TRUE)
curve(matern(x, phi = 0.14, kappa = 2),from = 0, to = 1,
      add = TRUE, lwd=2, lty=2)
curve(matern(x, phi = 0.117, kappa = 2),from = 0, to = 1,
      add = TRUE, lwd=2)
legend(0.4,1, c(expression(paste(kappa == 0.5, "  and  ",
       phi == 0.250)),
       expression(paste(kappa == 1, "  and  ", phi == 0.188)),
       expression(paste(kappa == 2, "  and  ", phi == 0.140)),
       expression(paste(kappa == 3, "  and  ", phi == 0.117))),
       lty=c(2,1,2,1), lwd=c(1,1,2,2))
```

---

names.geodata *Lists names of the key elements of a geodata object*

Description

Produces a list with the names of the main elements of `geodata` object: coords, data, units.m, covariate and realisation. Can be useful to list names before using `{subset.geodata}`.

Usage

```
## S3 method for class 'geodata':
names(x)
```

Arguments

`x`	an object of the class `geodata`.

Value

A list with

`coords`	names of the coordinates in the geodata object.
`data`	name(s) of the data elements in the geodata object.
`units.m`	returns the string `units.m`.
`covariates`	return the covariate(s) name(s) if present in the `geodata object`
`realisations`	returns the string `units.m` if present in the `geodata object`.
`other`	other elements in the `geodata` object.

See Also

`names`, `subset.geodata`, `as.geodata`.

Examples

```
data(ca20)
names(ca20)
```

---

nearloc *Near location to a point*

Description

For a given set of points and locations identified by 2D coordinates this function finds the nearest location of each point

Usage

```
nearloc(points, locations, positions = FALSE)
```

Arguments

points	a matrix, data-frame or list with the 2D coordinates of a set of points for which you want to find the nearest location.
locations	a matrix, data-frame or list with the 2D coordinates of a set of locations.
positions	logical defining what to be returned. If TRUE the function returns the positions of the locations, otherwise the coordinates of the locations. Defaults to FALSE.

Value

If positions = FALSE the function returns a matrix, data-frame or list of the same type and size as the object provided in the argument points with the coordinates of the nearest locations.

If positions = FALSE the function returns a vector with the position of the nearest points in the locations object.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

See Also

loccoords

Examples

```
set.seed(276)
gr <- expand.grid(seq(0,1, l=11), seq(0,1, l=11))
plot(gr, asp=1)
pts <- matrix(runif(10), nc=2)
points(pts, pch=19)
near <- nearloc(points=pts, locations=gr)
points(near, pch=19, col=2)
rownames(near)
nearloc(points=pts, locations=gr, pos=TRUE)
```

.nlmP *Adapts nlm for Constraints in the Parameter Values*

Description

This function adapts the R function nlm to allow for constraints (upper and/or lower bounds) in the values of the parameters.

Usage

```
.nlmP(objfunc, params, lower=rep(-Inf, length(params)),
      upper=rep(+Inf, length(params)), ...)
```

Arguments

`objfunc`	the function to be minimized.
`params`	starting values for the parameters.
`lower`	lower bounds for the variables. Defaults to $-Inf$.
`upper`	upper bounds for the variables. Defaults to $-Inf$.
`...`	further arguments to be passed to the function `nlm`.

Details

Constraints on the parameter values are internally imposed by using exponential, logarithmic, and logit transformation of the parameter values.

Value

The output is the same as for the function `nlm`.

Author(s)

Patrick E. Brown ⟨p.brown@lancaster.ac.uk⟩.
Adapted and included in **geoR** by
Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`nlm`, `optim`.

`output.control` *Defines output options for prediction functions*

Description

Auxiliary function defining output options for `krige.bayes` and `krige.conv`.

Usage

```
output.control(n.posterior, n.predictive, moments, n.back.moments,
               simulations.predictive, mean.var, quantile,
               threshold, sim.means, sim.vars, signal, messages)
```

Arguments

n.posterior	number of samples to be taken from the posterior distribution. Defaults to 1000.
n.predictive	number of samples to be taken from the predictive distribution. Default equals to n.posterior.
moments	logical. Indicates whether the moments of the predictive distribution are returned. If lambda = 1 there is no transformation/back-transformation. If lambda = 0 or lambda = 0.5 the moments are back-transformed by analytical expressions. For other cases the back-transformation is done by simulation. Defaults to TRUE.
n.back.moments	number of sample to back-transform moments by simulation. Defaults to 1000.
simulations.predictive	logical. Defines whether to draw simulations from the predictive distribution. Only considered if prediction locations are provided in the argument locations of the main functions. Defaults to FALSE but changed to TRUE if an integer greater then zero is provided in the argument n.predictive and/or simulations are required in order to compute quantities required by other arguments such as threshold, quantiles and some values of the transformation parameter.
mean.var	logical (optional). Indicates whether mean and variances of the simulations of the predictive distributions are computed and returned.
quantile	a (optional) numeric vector. If provided indicates whether quantiles of the simulations from the predictive distribution are computed and returned. If a vector with numbers in the interval $[0, 1]$ is provided, the output includes the object quantiles, which contains values of corresponding estimated quantiles. For example, if quantile = c(0.25, 0.50, 0.75) the function returns the quartiles of the distributions at each of the prediction locations. If quantile = TRUE default values equals to c(0.025, 0.5, 0.975), are assumed. A measure of uncertainty of the predictions, analogous to the kriging standard error, can be computed by $(quantile0.975 - quantile0.025)/4$. Only used if prediction locations are provided in the argument locations.
threshold	a (optional) numerical vector. If one or more values are provided, object named probabilities is included in the output. This object contains, for each prediction location, the probability that the variable is less than or equal than the threshold provided by the user. Defaults to FALSE.
sim.means	logical (optional). Indicates whether mean of each of the conditional simulations of the predictive distribution should be computed and returned. Defaults to TRUE if simulations from the predictive are required.
sim.vars	logical (optional). Indicates whether variance of each of the conditional simulations of the predictive distribution should be computed and returned. Defaults to FALSE.
signal	logical indicating whether the signal or the variable is to be predicted. Defaults to NULL and changed internally in the functions which call output.control. See DETAILS below.
messages	logical. Indicates whether or not status messages are printed on the output device while the function is running. Defaults to TRUE.

Details

SIGNAL

This function is typically called by the **geoR**'s prediction functions `krige.bayes` and `krige.conv` defining the output to be returned by these functions.

The underlying model

$$Y(x) = \mu + S(x) + \epsilon$$

assumes that observations $Y(x)$ are noisy versions of a *signal* $S(x)$ and $Var(\epsilon) = \tau^2$ is the nugget variance.

If $\tau^2 = 0$ the Y and S are indistiguishable.

If $\tau^2 > 0$ and regarded as measurement error the option `signal` defines whether the S (`signal = TRUE`) or the variable Y (`signal = FALSE`) is to be predicted.
For the latter the predictions will "honor" the data, i.e. at data locations predictions will coincide with the data.
For unsampled locations, when there is no transformation of the data, the predicted values will be the same regardless whether `signal = TRUE` or `FALSE` but the predictions variances will differ.

By default `krige.bayes` sets `signal = TRUE` and `krige.conv` sets `signal = FALSE`.

The function `krige.conv` has an argument `micro.scale`. If $micro.scale > 0$ the error term is divided as $\epsilon = \epsilon_{ms} + \epsilon_{me}$ and the nugget variance is divided into two terms: *micro-scale variance* and *measurement error*.
If `signal = TRUE` the term ϵ_{ms} is regarded as part of the signal and consequently the *micro-scale variance* is added to the prediction variance.
If `signal = FALSE` the total error variance τ^2 is added to the prediction variance.

Value

A list with processed arguments to be passed to the main function.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

See Also

The prediction functions `krige.bayes` and `krige.conv`.

---

parana — *Rainfall Data from Parana State, Brasil*

Description

This data-set was used by Diggle and Ribeiro (2001) to illustrate the methods discussed in the paper. The data reported analysis was carried out using the package **geoR**.

The data refers to average rainfall over different years for the period May-June (dry-season). It was collected at 143 recording stations throughout Paraná State, Brasil.

Usage

```
data(parana)
```

Format

The object `parana` of the class `geodata`, which is a list containing the following components:

`coords` a matrix with the coordinates of the recording stations.

`data` a vector with the average recorded rainfall for the May-June period.

`borders` a matrix with the coordinates defining the borders of Paraná state.

`loci.paper` a matrix with the coordinates of the four prediction locations discussed in the paper.

Source

The data were collected at several recording stations at Paraná State, Brasil, belonging to the following companies: COPEL, IAPAR, DNAEE, SUREHMA and INEMET.

The data base was organized by Laura Regina Bernardes Kiihl (IAPAR, Instituto Agronômico do Paraná, Londrina, Brasil) and the fraction of the data included in this data-set was provided by Jacinta Loudovico Zamboti (Universidade Estadual de Londrina, Brasil). The coordinates of the borders of Paraná State were provided by João Henrique Caviglione (IAPAR).

References

Diggle, P.J. & Ribeiro Jr, P.J. (2002) Bayesian inference in Gaussian model-based geostatistics. Geographical and Environmental Modelling, Vol. 6, No. 2, 129-146.

Examples

```
data(parana)
summary(parana)
plot(parana, bor=borders)
```

`pars.limits` *Set limits for the parameter values*

Description

The functions `likfit` and `limits{variofit}` in the package **geoR**

Usage

```
pars.limits(phi = c(lower = 0, upper = +Inf),
            sigmasq = c(lower = 0, upper = +Inf),
            nugget.rel = c(lower = 0, upper = +Inf),
            kappa = c(lower = 0, upper = +Inf),
            kappa2 = c(lower = 0, upper = +Inf),
            lambda = c(lower = -3, upper = 3),
            psiR = c(lower = 1, upper = +Inf),
            psiA = c(lower = 0, upper = 2 * pi),
            tausq.rel = nugget.rel)
```

Arguments

`phi`	a two elements vector with limits for the parameter phi. Defaults to [0, +Inf]
`sigmasq`	idem for sigmasq. Defaults to [0, +Inf]
`nugget.rel`	idem for nugget.rel. Defaults to [0, +Inf]
`kappa,kappa2`	idem. Defaults to [0, +Inf]
`lambda`	idem for lambda. Defaults to [-3, +3]. Only used in `likfit`.
`psiR`	idem for psiR. Defaults to [1, +Inf]. Only used in `likfit`.
`psiA`	idem for psiA. Defaults to [0, 2 pi]. Only used in `likfit`.
`tausq.rel`	idem for tausq.rel. Defaults to [0, +Inf]

Details

Lower and upper limits for parameter values can be individually specified. For example, including the following in the function call in `likfit` or `variofit`:
`limits = pars.limits(phi=c(0, 10), lambda=c(-2.5, 2.5)),`
will change the limits for the parameters ϕ and λ. Default values are used if the argument `limits` is not provided.

Value

A list of a 2 elements vector with limits for each parameters

See Also

Examples

```
pars.limits(phi=c(0,10))
pars.limits(phi=c(0,10), sigmasq=c(0, 100))
```

`plot.geodata` *Exploratory Geostatistical Plots*

Description

This function produces a 2×2 display with the following plots: the first indicates the spatial locations assign different colors to data in different quartiles, the next two shows data against the *X* and *Y* coordinates and the last is an histogram of the data values or optionally, a 3-D plot with spatial locations and associated data values.

Usage

```
## S3 method for class 'geodata':
plot(x, coords=x$coords, data = x$data,
             borders, trend="cte", lambda = 1, col.data = 1,
             weights.divide = "units.m", lowess = FALSE, scatter3d = FALSE,
             qt.col, ...)
```

Arguments

`x`	a list containing elements `coords` and `data` described next. Typically an object of the class `"geodata"` - a **geoR** data-set. If not provided the arguments `coords` and `data` must be provided instead.
`coords`	an $n \times 2$ matrix containing in each row Euclidean coordinates of the *n* data locations. By default it takes the element `coords` of the argument `geodata`.
`data`	a vector with data values. By default it takes the element `data` of the argument `geodata`.
`borders`	If an $n \times 2$ matrix or data-frame with the borders of the area is provided, the borders are included in the first plot. By default it searches for a element named "borders" in the geodata object.
`trend`	specifies the mean part of the model. The options are: `"cte"` (constant mean - default option), `"1st"` (a first order polynomial on the coordinates), `"2nd"` (a second order polynomial on the coordinates), or a formula of the type `~X` where `X` is a matrix with the covariates (external trend). If provided the trend is "removed" using the function `lm` and the residuals are plotted.
`lambda`	value of the Box-Cox transformation parameter. Two particular cases are $\lambda = 1$ which corresponds to no transformation and $\lambda = 0$ corresponding to the log-transformation.
`col.data`	indicates the column number for the data to be plotted. Only valid if more than one data-set is available i.e., if the argument `data` is a matrix.
`weights.divide`	if a vector of weights with the same length as the data is provided each data is divided by the corresponding element in this vector. Defaults divides the data by the element `units.m` in the data object, if present, otherwise no action is taken and original data is used. The usage of `units.m` is common for data objects to be analysed using the package **geoRglm**.

lowess	logical. Indicates whether the function lowess should be used in the plots of the data against the coordinates.
scatter3d	logical. If TRUE the last plot is produced by scatterplot3d showing a 3d plot with data locations and corresponding values.
qt.col	colors for the quartiles in the first plot. If missing defaults to blue, green, yellow and red.
...	further arguments to be passed to the function hist or scatterplot3d.

Value

A plot is produced on the graphics device. No values are returned.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

points.geodata, scatterplot3d.

Examples

```
require(geoR)
data(s100)
plot(s100)
plot(s100, scatter3d=TRUE)
plot(s100, qt.col=1)

data(ca20)
plot(ca20, bor=borders)           # original data
plot(ca20, trend=~altitude+area) # residuals from an external trend
plot(ca20, trend='1st')           # residuals from a polynomial trend

data(SIC)
plot(sic.100, bor=sic.borders)            # original data
plot(sic.100, bor=sic.borders, lambda=0) # logarithm of the data
```

`plot.grf` *Plots Variograms for Simulated Data*

Description

This function plots variograms for simulated geostatistical data generated by the function `grf`.

Usage

```
## S3 method for class 'grf':
plot(x, model.line = TRUE, plot.locations = FALSE, ...)
```

Arguments

`x`	an object of the class `grf`, typically an output of the function `grf`.
`model.line`	logical. If `TRUE` the true variogram model is added to the plot with the sample variogram(s).
`plot.locations`	logical. If `TRUE` a plot with data locations is also shown.
`...`	further arguments to be passed to the functions `variog` and `plot`.

Value

A plot with the empirical variogram(s) is produced on the output device. No values are returned.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
`http://www.est.ufpr.br/geoR`.

See Also

`grf` for simulation of Gaussian random fields, `plot.variogram` for plotting empirical variogram, `variog` for computation of empirical variograms and `plot` for the generic plotting function.

Examples

```
op <- par(no.readonly = TRUE)
par(mfrow=c(2,1))
sim1 <- grf(100, cov.pars=c(10, .25))
# generates simulated data
plot(sim1, plot.locations = TRUE)
#
# plots the locations and the sample true variogram model
#
par(mfrow=c(1,1))
sim2 <- grf(100, cov.pars=c(10, .25), nsim=10)
# generates 10 simulated data
plot(sim1)
# plots sample variograms for all simulations with the true model
par(op)
```

---

plot.krige.bayes *Plots Prior and/or Posterior Distributions*

Description

Produces plots the priors and posteriors distribuitions for the paramters `phi` and `tausq.rel` based on results returned by `krige.bayes`.

Usage

```
## S3 method for class 'krige.bayes':
plot(x, phi.dist = TRUE, tausq.rel.dist = TRUE, add = FALSE,
                 type=c("bars", "h", "l", "b", "o", "p"), thin, ...)
```

Arguments

`x`	an object of the class `krige.bayes`, with an output of the funtions `krige.bayes`.
`phi.dist`	logical indicating whether or not plot the distributions for this parameter.
`tausq.rel.dist`	logical indicating whether or not plot the distributions for this parameter.
`add`	logical. If `TRUE` plots is added to current one.
`type`	indicates the type of plot. Option `"bars"` uses the function `barplot` and the others uses `matplot`.
`thin`	a numerical vector defining the thining for values of the parameters `phi` and `tausq.rel` respectively. This improves visualisation when there are many values in the discrete distribution of the parameters.
`...`	further arguments for the plotting function.

Value

For `plot.krige.bayes` a plot is produced or added to the current graphics device. No values are returned.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

See Also

`krige.bayes`, `barplot`, `matplot`.

Examples

```
## See documentation for krige.bayes
```

---

plot.proflik *Plots Profile Likelihoods*

Description

This function produces plots of the profile likelihoods computed by the function `proflik`.

Usage

```
## S3 method for class 'proflik':
plot(x, pages = c("user", "one", "two"), uni.only, bi.only,
             type.bi = c("contour", "persp"), conf.int = c(0.90, 0.95),
             yaxis.lims = c("conf.int", "as.computed"),
             by.col = TRUE, log.scale = FALSE, use.splines = TRUE,
             par.mar.persp = c(0, 0, 0, 0), ask = FALSE, ...)
```

Arguments

`x`	an object of the class `proflik`, typically an output of the function `proflik`.
`pages`	specify how the plots will be arranged in the graphics device. The default option, `"user"`, uses the current graphical parameters. The option `"one"` places all the profiles in the same page and the option `"two"` places the univariate profiles in one page and the bivariate profiles in a second page.
`uni.only`	only 1-D profiles are plotted even if the object contains results about the 2-D profiles.
`bi.only`	only 2-D profile are plotted even if the object contains results about the 1-D profiles.
`type.bi`	Type of plot for the 2-D profiles. Options are `"contour"` for contour plot (the default) and `"persp"` for perspective plot.

`conf.int`	a vector with numbers in the interval $[0, 1]$ specifying levels of the (approximated) confidence intervals. Defaults corresponds to the levels 90% and 95%.
`yaxis.lims`	defines the lower limits for the y-axis in the 1-D plots. If `"conf.int"` the limit is determined by the level of the confidence interval (the default) otherwise will be determined by the smallest computed value.
`by.col`	logical, If `TRUE` the plots are arranged by columns in a multiple graphics device.
`log.scale`	plots the x-axis in the logarithmic scale. Defaults to `FALSE`.
`use.splines`	logical. If `TRUE` (the default) the function `spline` is used to interpolate between the points computed by `proflik`.
`par.mar.persp`	graphical parameters to be used with `persp` plots. For more details see `par`.
`ask`	logical. Defines whether or not the user is prompted before each plot is produced.
`...`	additional arguments to be passed to the functions `plot`, `contour` and/or `persp`.

Value

Produces plots with the profile likelihoods on the current graphics device. No values are returned.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
`http://www.est.ufpr.br/geoR`.

See Also

`proflik` for computation of the profile likelihoods. For the generic plotting functions see `plot`, `contour`, `persp`. See `spline` for interpolation.

Examples

```
# see examples in the documentation for the function proflik()
```

`plot.variog4` *Plot Directional Variograms*

Description

This function plot directional variograms computed by the function `variog4`. The omnidirectional variogram can be also included in the plot.

Usage

```
## S3 method for class 'variog4':
plot(x, omnidirectional=FALSE, same.plot=TRUE, legend = TRUE, ...)
```

Arguments

`x`	an object of the class `variog4`, typically an output of the function `variog4`.
`omnidirectional`	logical. Indicates whether the omnidirectional variogram is included in the plot.
`same.plot`	logical. Indicates whether the directional variograms are plotted in the same or separated plots.
`legend`	logical indicating whether legends are automatically included in the plots.
`...`	further arguments to be passed to the function `plot`. Typical arguments are `col`, `lty`, `lwd`. For `same.plot = TRUE` the arguments are passed to the function `matplot` which is used to produce the plot.

Value

A plot is produced on the output device. No values returned.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information about the **geoR** package can be found at:
http://www.est.ufpr.br/geoR.

See Also

`variog4` for variogram calculations and `matplot` for multiple lines plotting.

Examples

```
if(is.R()) data(s100)
s100.v4 <- variog4(s100, max.dist=1)
# Plotting variograms for the four directions
plot(s100.v4)
# changing plot options
plot(s100.v4, lwd=2)
plot(s100.v4, lty=1, col=c("darkorange", "darkblue", "darkgreen","darkviolet"))
plot(s100.v4, lty=1, lwd=2)
# including the omnidirectional variogram
plot(s100.v4, omni=TRUE)
# variograms on different plots
plot(s100.v4, omni=TRUE, same=FALSE)
```

---

plot.variogram *Plot Empirical Variogram*

Description

Plots sample (empirical) variogram computed using the function variog.

Usage

```
## S3 method for class 'variogram':
plot(x, max.dist, vario.col = "all", scaled = FALSE,
                var.lines = FALSE, envelope.obj = NULL,
                pts.range.cex, bin.cloud = FALSE,  ...)
```

Arguments

x	an object of the class "variogram", typically an output of the function variog.
max.dist	maximum distance for the x-axis. The default is the maximum distance for which the sample variogram was computed.
vario.col	only used if obj has information on more than one empirical variogram. The default "all" indicates that variograms of all variables should be plotted. Alternatively a numerical vector can be used to select variables.
scaled	If TRUE the variogram values are divided by the sample variance. This allows comparison of variograms of variables measured in different scales.
var.lines	If TRUE a horizontal line is drawn at the value of the variance of the data (if scaled = F) or at 1 (if scaled = T).
envelope.obj	adds a variogram envelope computed by the function variog.model.env or variog.mc.env.
pts.range.cex	optional. A two elements vector with maximum and minimum values for the caracter expansion factor cex. If provided the point sizes in binned variogram are proportional to the number of pairs of points used to compute each bin.

`bin.cloud` logical. If `TRUE` and the sample variogram was computed with the option `keep.cloud = TRUE`, box-plots of values at each bin are plotted instead of the empirical variograms.

`...` other arguments to be passed to the function `plot` or `matplot`

Details

This function plots empirical variograms. Toghether with `lines.variogram` can be used to compare sample variograms of different variables and to compare variogram models against the empirical variogram.

It uses the function `matplot` when plotting variograms for more them one variable.

Value

Produces a plot with the sample variogram on the current graphics device. No values are returned.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`variog` for variogram calculations, `lines.variogram` and `lines.variomodel` for adding lines to the current plot, `variog.model.env` and `variog.mc.env` for variogram envelops computation, `matplot` for multiple lines plot and `plot` for generic plot function.

Examples

```
op <- par(no.readonly = TRUE)
sim <- grf(100, cov.pars=c(1, .2)) # simulates data
vario <- variog(sim, max.dist=1)   # computes sample variogram
par(mfrow=c(2,2))
plot(vario)                      # the sample variogram
plot(vario, scaled = TRUE)       # the scaled sample variogram
plot(vario, max.dist = 1)        # limiting the maximum distance
plot(vario, pts.range = c(1,3)) # points sizes proportional to number of pairs
par(op)
```

`plot.xvalid` *Plot Cross-Validation Results*

Description

This function produces ten plots with the results produced by the cross-validation function `xvalid`.

Usage

```
## S3 method for class 'xvalid':
plot(x, coords, borders = NULL, ask = TRUE,
            error = TRUE, std.error = TRUE, data.predicted = TRUE,
            pp = TRUE, map = TRUE, histogram = TRUE,
            error.predicted = TRUE, error.data = TRUE, ...)
```

Arguments

`x`	an object of the class `"xvalid"`, typically an output from the function `xvalid`.
`coords`	an $n \times 2$ object containing coordinates of the (cross-)validation locations.
`borders`	optional. Takes a two column matrix or data-frame with coordinates of the borders. If provided the borders are included in the errors maps.
`ask`	logical. Defines whether or not the user is prompted before each plot is produced.
`error`	logical. Defines whether the plots for the errors ($error = data - predicted$) will be produced.
`std.error`	logical. Defines whether the plots for the standardised errors will be produced.
`data.predicted`	logical defining whether a plot of data versus predicted should be displayed. Defaults to `TRUE`.
`pp`	logical defining whether a *pp* plot should be displayed. Defaults to `TRUE`.
`map`	logical defining whether a map of the errors should be displayed. Defaults to `TRUE`.
`histogram`	logical defining whether a histogram of the errors should be displayed. Defaults to `TRUE`.
`error.predicted`	logical defining whether a plot of errors versus predicted should be displayed. Defaults to `TRUE`.
`error.data`	logical defining whether a plot of errors versus data should be displayed. Defaults to `TRUE`.
`...`	other arguments to be passed to the function `plot`.

Details

The number of plots to be produced will depend on the input options. If the graphics device is set to just one plot (something equivalent to '`par(mfcol=c(1,1))`') after each graphic being displayed the user will be prompt to press *<return>* to see the next graphic.

Alternativaly the user can set the graphical parameter to have several plots in one page. With default options for the arguments the maximum number of plots (10) is produced and setting '`par(mfcol=c(5,2))`') will display them in the same page.

The "errors" for the plots are defined as

$$error = data - predicted$$

and the plots uses the color blue to indicate positive errors and red to indicate negative erros.

Value

No value returned. Plots are produced on the current graphics device.

See Also

`xvalid` for the cross-validation computations.

Examples

```
data(s100)
wls <- variofit(variog(s100, max.dist = 1), ini = c(.5, .5), fix.n = TRUE)
xvl <- xvalid(s100, model = wls)
#
op <- par(no.readonly = TRUE)
par(mfcol = c(3,2))
par(mar = c(3,3,0,1))
par(mgp = c(2,1,0))
plot(xvl, error = FALSE, ask = FALSE)
plot(xvl, std.err = FALSE, ask = FALSE)
par(op)
```

---

points.geodata *Plots Spatial Locations and Data Values*

Description

This function produces a plot with points indicating the data locations. Arguments can control the points sizes, patterns and colors. These can be set to be proportional to data values, ranks or quantiles. Alternatively, points can be added to the current plot.

Usage

```
## S3 method for class 'geodata':
points(x, coords=x$coords, data=x$data, data.col = 1, borders,
               pt.divide=c("data.proportional","rank.proportional",
                           "quintiles", "quartiles", "deciles", "equal"),
               lambda = 1, trend = "cte", abs.residuals = FALSE,
               weights.divide = "units.m", cex.min, cex.max, cex.var,
               pch.seq, col.seq, add.to.plot = FALSE,
               x.leg, y.leg, dig.leg = 2,
               round.quantiles = FALSE, graph.pars = FALSE,
               permute = FALSE, ...)
```

Arguments

`x`	a list containing elements `coords` and `data` described next. Typically an object of the class `"geodata"` - a **geoR** data-set. If not provided the arguments `coords` and `data` must be provided instead.
`coords`	an $n \times 2$ matrix containing coordinates of the n data locations in each row. Defaults to `geodata$coords`.
`data`	a vector or matrix with data values. If a matrix is provided each column is regarded as one variable or realization. Defaults to `geodata$data`.
`data.col`	the number of the data column. Only used if `data` is a matrix with columns corresponding to different variables or simulations.
`borders`	If an $n \times 2$ matrix or data-frame with the coordinates of the borders of the regions is provided, the borders are added to the plot. By default it searches for a element named "borders" in the geodata object.
`pt.divide`	defines the division of the points in categories. See `DETAILS` below for the available options. Defaults to `pt.divide = "data.proportional"`.
`trend`	specifies the mean part of the model. The options are: `"cte"` (constant mean - default option), `"1st"` (a first order polynomial on the coordinates), `"2nd"` (a second order polynomial on the coordinates), or a formula of the type `~X` where `X` is a matrix with the covariates (external trend). If provided the trend is "removed" using the function `lm` and the residuals are plotted.
`abs.residuals`	logical. If `TRUE` and the value passed to the argument `trend` is different from `"cte"` the point sizes are proportional to absolute values of the residuals.
`lambda`	value of the Box-Cox transformation parameter. Two particular cases are $\lambda = 1$ which corresponds to no transformation and $\lambda = 0$ corresponding to the log-transformation.
`weights.divide`	if a vector of weights with the same length as the data is provided each data is divided by the corresponding element in this vector. Defaults divides the data by the element `units.m` in the data object, if present, otherwise no action is taken and original data is used. The usage oc `units.m` is common for data objects to be analysed using the package **geoRglm**.

cex.min	minimum value for the graphical parameter cex. This value defines the size of the point corresponding the minimum of the data. Defaults to 0.5.
cex.max	maximum value for the graphical parameter cex. This value defines the size of the point corresponding the maximum of the data. If pt.divide = "equal" it is used to set the value for the graphical parameter cex. Defaults to 1.5.
cex.var	a numeric vector with the values of a variable defining the size of the points. Particularly useful for displaying 2 variables at once.
pch.seq	number(s) defining the graphical parameter pch.
col.seq	number(s) defining the colors in the graphical parameter col.
add.to.plot	logical. If TRUE the points are added to the current plot otherwise a display is open. Defaults to FALSE.
x.leg, y.leg	x and y location of the legend.
dig.leg	integer indicating the precision to be used in the legend values.
round.quantiles	logical. Defines whether or not the values of the quantiles should be rounded. Defaults to FALSE.
graph.pars	logical. If TRUE the graphics parameters used to produce the plots are returned. Defaults to FALSE.
permute	logical indication whether the data values should be randomly re-alocatted to the coordinates. See DETAILS below.
...	further arguments to be passed to the function plot, if add.to.plot = FALSE; or to the function points, if add.to.plot = TRUE.

Details

The points can be devided in categories and have different sizes and/or colours according to the argument pt.divide. The options are:

"data.proportional" sizes proportional to the data values.

"rank.proportional" sizes proportional to the rank of the data.

"quintiles" five different sizes according to the quintiles of the data.

"quartiles" four different sizes according to the quartiles of the data.

"deciles" ten different sizes according to the deciles of the data.

"equal" all points with the same size.

a scalar defines a number of quantiles, the number provided defines the number of different points sizes and colors.

a numerical vector with quantiles and length > 1 the values in the vector will be used by the function cut as break points to divide the data in classes.

For cases where points have different sizes the arguments cex.min and cex.max set the minimum and the maximum point sizes. Additionally, pch.seq can set different patterns for the points and col.seq can be used to define colors. For example, different colors can be used for quartiles, quintiles and deciles while a sequence of gray tones (or a color sequence) can be used for point sizes proportional to the data or their ranks. For more details see the section EXAMPLES.

The argument `cex.var` allows for displaying 2 variables at once. In this case one variable defines the backgroung colour of the points and the other defines the points size.

The argument `permute` if set to `TRUE` randomly realocates the data in the coordinates. This may be used to contrast the spatial pattern of original data against another situation where there is no spatial dependence (when setting `permute = TRUE`). If a `trend` is provided the residuals (and not the original data) are permuted.

Value

A plot is created or points are added to the current graphics device.
By default no value is returned. However, if `graph.pars = TRUE` a list with graphical parameters used to produce the plot is returned. According to the input options, the list has some or all of the following components:

`quantiles`	the values of the quantiles used to divide the data.
`cex`	the values of the graphics expansion parameter `cex`.
`col`	the values of the graphics color parameter `col`.
`pch`	the values of the graphics pattern parameter `pch`.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`plot.geodata` for another display of the data and `points` and `plot` for information on the generic R functions. The documentation of `par` provides details on graphical parameters. For color schemes in R see `gray` and `rainbow`.

Examples

```
data(s100)
op <- par(no.readonly = TRUE)
par(mfrow=c(2,2), mar=c(3,3,1,1), mgp = c(2,1,0))
points(s100, xlab="Coord X", ylab="Coord Y")
points(s100, xlab="Coord X", ylab="Coord Y", pt.divide="rank.prop")
points(s100, xlab="Coord X", ylab="Coord Y", cex.max=1.7,
               col=gray(seq(1, 0.1, l=100)), pt.divide="equal")
points(s100, pt.divide="quintile", xlab="Coord X", ylab="Coord Y")
par(op)

data(ca20)
points(ca20, pt.div='quartile', x.leg=4900, y.leg=5850, bor=borders)
```

```
par(mfrow=c(1,2), mar=c(3,3,1,1), mgp = c(2,1,0))
points(s100, main="Original data")
points(s100, permute=TRUE, main="Permuting locations")

## Now an example using 2 variable, 1 defining the
## gray scale and the other the points size
data(camg)
points.geodata(coords=camg[,1:2], data=camg[,3], col="gray",
               cex.var=camg[,5])
points.geodata(coords=camg[,1:2], data=camg[,3], col="gray",
               cex.var=camg[,5], pt.div="quint")
```

`polygrid` *Coordinates of Points Inside a Polygon*

Description

This function builds a rectangular grid and extracts points which are inside of an internal polygonal region.

Usage

```
polygrid(xgrid, ygrid, borders, vec.inout = FALSE, ...)
```

Arguments

`xgrid`	grid values in the *x*-direction.
`ygrid`	grid values in the *y*-direction.
`borders`	a matrix with polygon coordinates defining the borders of the region.
`vec.inout`	logical. If `TRUE` a logical vector is included in the output indicating whether each point of the grid is inside the polygon. Defaults to `FALSE`.
`...`	currently not used (kept for back compatibility).

Details

The function works as follows: First it creates a grid using the R function `expand.grid` and then it uses the geoR' internal function `.geoR_inout()` which wraps usage of `SpatialPoints` and `overlay` from the package **sp** to extract the points of the grid which are inside the polygon.

Within the package **geoR** this function is typically used to select points in a non-rectangular region to perform spatial prediction using `krige.bayes`, `krige.conv` or `ksline`. It is also useful to produce image or perspective plots of the prediction results.

Value

A list with components:

`xypoly`	an $n \times 2$ matrix with the coordinates of the points inside the polygon.
`vec.inout`	logical, a vector indicating whether each point of the rectangular grid is inside the polygon. Only returned if `vec.inout = TRUE`.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`expand.grid`, `overlay`, `SpatialPoints`.

Examples

```
poly <- matrix(c(.2, .8, .7, .1, .2, .1, .2, .7, .7, .1), ncol=2)
plot(0:1, 0:1, type="n")
lines(poly)
poly.in <- polygrid(seq(0,1,l=11), seq(0,1,l=11), poly, vec=TRUE)
points(poly.in$xy)
```

---

pred_grid *Generates a 2D Prediction Grid*

Description

This function facilitates the generation of a 2D prediction grid for geostatistical kriging.

Usage

```
pred_grid(coords, y.coords = NULL, ..., y.by = NULL,
          y.length.out = NULL, y.along.with = NULL)
```

Arguments

`coords`	a list, matrix or data-frame with xy-coordinates of prediction points or a vector with x-coordinates.
`y.coords`	a vector with y-coordinates. Needed if argument `coords` provides only x-coordinates.
`...`	arguments `by` or `length.out` to be passed to the function `rep`. These arguments are used for the x-coordinates and are default optioons for y-coordinates.
`y.by`	Optional. `by` argument for `rep` to be used with the y-coordinates.
`y.length.out`	Optional. `length.out` argument for `rep` to be used with the y-coordinates.
`y.along.with`	Optional. `along.with` argument for `rep` to be used with the y-coordinates.

Value

An two column data-frame which is on output of `expand.grid`.

See Also

See `seq` and `expand.grid` which are used internally and `locations.inside` and `polygrid` to select points inside a border.

Examples

```
pred_grid(c(0,1), c(0,1), by=0.25) ## create a grid in a unit square
data(ca20)
loc0 <- pred_grid(ca20$borders, by=20)
points(ca20, borders=borders)
points(loc0, pch="+")
points(locations.inside(loc0, ca20$border), pch="+", col=2)
```

---

predict.BGCCM *Prediction for the bivariate Gaussian common component geostatistical model*

Description

Performs prediction for the bivariate Gaussian common component geostatistical model

Usage

```
## S3 method for class 'BGCCM':
predict(object, locations, borders,
              variable.to.predict = 1, ...)
```

Arguments

`object`	on object of the classBGCCMfit, which is an output of `likfitBGCCM`.
`locations`	an $N \times 2$ matrix or data-frame with the 2-D coordinates of the N prediction locations, or a list for which the first two components are used. Input is internally checked by the function `check.locations`.
`borders`	optional. If missing, by default reads the element `borders` of the `geodata` object of the variable to be predicted. Ignored if set to `NULL`. If a two column matrix defining a polygon is provided the prediction is performed only at locations inside this polygon.
`variable.to.predict`	scalar with options for values or 2 indicating which variable is to be predicted.
`...`	not yet used.

Value

A list of the class `BGCCMpred` with components:

`predicted`	predicted values.
`krige.var`	prediction variances.

Warning

This is a new function and still in draft format and pretty much untested.

Author(s)

who you are

See Also

`likfitBGCCM`

Examples

```
# see http://www.est.ufpr.br/geoR/tutorials/CCM.R
```

---

print.BGCCM *Prints an summary of of the output from likfitBGCCM.*

Description

Prints a short version of an object of the class `BGCCM`.

Usage

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

Arguments

`x`	an object of the class `BGCCM`.
`...`	arguments to be passed to `format`.

See Also

`format` for options to format the output.

`proflik` *Computes Profile Likelihoods*

Description

Computes profile likelihoods for model parameters previously estimated using the function `likfit`.

Usage

```
proflik(obj.likfit, geodata, coords = geodata$coords,
        data = geodata$data, sill.values, range.values,
        nugget.values, nugget.rel.values, lambda.values,
        sillrange.values = TRUE, sillnugget.values = TRUE,
        rangenugget.values = TRUE, sillnugget.rel.values = FALSE,
        rangenugget.rel.values = FALSE, silllambda.values = FALSE,
        rangelambda.values = TRUE,  nuggetlambda.values = FALSE,
        nugget.rellambda.values = FALSE,
        uni.only = TRUE, bi.only = FALSE, messages, ...)
```

Arguments

`obj.likfit`	an object of the class `likfit`, typically an output of the function `likfit`.
`geodata`	a list containing elements `coords` and `data` described next. Typically an object of the class `"geodata"` - a **geoR** data-set. If not provided the arguments `coords` and `data` must be provided instead.
`coords`	an $n \times 2$ matrix containing in each row Euclidean coordinates of the n data locations. By default it takes the element `coords` of the argument `geodata`.
`data`	a vector with data values. By default it takes the element `data` of the argument `geodata`.
`sill.values`	set of values of the partial sill parameter σ^2 for which the profile likelihood will be computed.
`range.values`	set of values of the range parameter ϕ for which the profile likelihood will be computed.
`nugget.values`	set of values of the nugget parameter τ^2 for which the profile likelihood will be computed. Only used if the model was fitted using the function `likfit` with the option `fix.nugget = FALSE`.
`nugget.rel.values`	set of values of the relative nugget parameter τ^2_R for which the profile likelihood will be computed. Only used if the model was fitted using the function `likfit` with the option `fix.nugget = FALSE`.
`lambda.values`	set of values of the Box-Cox transformation parameter λ for which the profile likelihood will be computed. Only to be used if the model was fitted using the function `likfit` with the option `fix.lambda = FALSE`.

`sillrange.values`
: logical indicating whether or not the 2-D profile likelihood should be computed. Only valid if `uni.only = FALSE`.

`sillnugget.values`
: as above.

`rangenugget.values`
: as above.

`sillnugget.rel.values`
: as above.

`rangenugget.rel.values`
: as above.

`silllambda.values`
: as above.

`rangelambda.values`
: as above.

`nuggetlambda.values`
: as above.

`nugget.rellambda.values`
: as above.

`uni.only`
: as above.

`bi.only`
: as above.

`messages`
: logical. Indicates whether status messages should be printed on the screen (i.e. current output device) while the function is running.

`...`
: additional parameters to be passed to the minimization function.

Details

The functions `.proflik.*` are auxiliary functions used to compute the profile likelihoods. These functions are internally called by the minimization functions when estimating the model parameters.

Value

An object of the class `"proflik"` which is a list. Each element contains values of a parameter (or a pair of parameters for 2-D profiles) and the corresponding value of the profile likelihood. The components of the output will vary according to the input options.

Note

1. Profile likelihoods for Gaussian Random Fields are usually uni-modal. Unusual or jagged shapes can be due to the lack of the convergence in the numerical minimization for particular values of the parameter(s). If this is the case it might be necessary to pass `control` arguments to the minimization functions using the argument `....` It's also advisable to try the different options for the `minimisation.function` argument. See documentation of the functions `optim` and/or `nlm` for further details.
2. 2-D profiles can be computed by setting the argument `uni.only = FALSE`. However, before computing 2-D profiles be sure they are really necessary. Their computation can be time demanding since it is performed on a grid determined by the cross-product of the values defining the 1-D profiles.

3. There is no "default strategy" to find reasonable values for the x-axis. They must be found in a "try-and-error" exercise. It's recommended to use short sequences in the initial attempts. The EXAMPLE section below illustrates this.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

plot.proflik for graphical output, likfit for the parameter estimation, optim and nlm for further details about the minimization functions.

Examples

```
op <- par(no.readonly=TRUE)
data(s100)
ml <- likfit(s100, ini=c(.5, .5), fix.nug=TRUE)
## a first atempt to find reasonable values for the x-axis:
prof <- proflik(ml, s100, sill.values=seq(0.5, 1.5, l=4),
                range.val=seq(0.1, .5, l=4))
par(mfrow=c(1,2))
plot(prof)
## a nicer setting
## Not run:
prof <- proflik(ml, s100, sill.values=seq(0.45, 2, l=11),
                range.val=seq(0.1, .55, l=11))
plot(prof)
## to include 2-D profiles use:
## (commented because this is time demanding)
#prof <- proflik(ml, s100, sill.values=seq(0.45, 2, l=11),
#                range.val=seq(0.1, .55, l=11), uni.only=FALSE)
#par(mfrow=c(2,2))
#plot(prof, nlevels=16)
## End(Not run)
par(op)
```

---

read.geodata *Reads and Converts Data to geoR Format*

Description

Reads data from a *ASCII* file and converts it to an object of the class geodata, the standard data format for the **geoR** package.

Usage

```
read.geodata(file, header = FALSE, coords.col = 1:2, data.col = 3,
             data.names = NULL, covar.col = NULL, covar.names = "header",
             units.m.col = NULL, realisations = NULL,
             na.action = c("ifany", "ifdata", "ifcovar", "none"),
             rep.data.action, rep.covar.action, rep.units.action, ...)
```

Arguments

`file`	a string with the name of the *ASCII* file.
`header`	logical. Indicates whether the variables names should be read from the first line of the input file.
`coords.col`	a vector with the numbers of the columns containing the coordinates.
`data.col`	a scalar or vector with the number of the column(s) containing the data.
`data.names`	a string or vector of strings with names for the data columns. Only valid if there is more than one column of data. By default the names in the original object are used.
`covar.col`	optional. A scalar or vector with the number of the column(s) with the values of the covariate(s).
`covar.names`	optional. A vector with the names of the the covariates. By default the names in the original object are used.
`units.m.col`	optional. A scalar with the column number corresponding to the offset variable. Alternativelly can be a character vector with the name of the offset. This option is particularly relevant when using the package **geoRglm**.
`realisations`	optional. A vector indicating the replication number. For more details see documentation for `as.geodata`.
`na.action`	a string. Defines action to be taken in the presence of `NA`'s. For more details see documentation for `as.geodata`.
`rep.data.action`	a string or a function. Defines action to be taken when there is more than one data at the same location. For more details see documentation for `as.geodata`.
`rep.covar.action`	a string or a function. Defines action to be taken when there is more than one covariate at the same location. For more details see documentation for `as.geodata`.
`rep.units.action`	a string or a function. Defines action to be taken on the element `units.m`, if present when there is more than one data at the same location. The default option is the same value set for `rep.data.action`.
`...`	further arguments to be passed to the function `read.table`.

Details

The function `read.table` is used to read the data from the *ASCII* file and then `as.geodata` is used to convert to an object of the `class` `geodata`.

Value

An object of the class geodata. See documentation for the function as.geodata for further details.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

as.geodata to convert existing R objects, read.table, the basic R function used to read *ASCII* files, and list for detailed information about lists.

---

s100 and s121 — *Simulated Data-Sets which Illustrate the Usage of the Package geoR*

Description

These two simulated data sets are the ones used in the Technical Report which describes the package **geoR** (see reference below). These data-sets are used in several examples throughout the package documentation.

Usage

```
data(s100)

data(s121)
```

Format

Two objects of the class geodata. Both are lists with the following components:

coords the coordinates of data locations.

data the simulated data. Notice that for s121 this a 121×10 matrix with 10 simulations.

cov.model the correlation model.

nugget the values of the nugget parameter.

cov.pars the covariance parameters.

kappa the value of the parameter *kappa*.

lambda the value of the parameter *lambda*.

References

Ribeiro Jr, P.J. and Diggle, P.J. (1999) geoS: A geostatistical library for S-PLUS. *Technical report ST-99-09, Dept of Maths and Stats, Lancaster University*.

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

Examples

```
data(s100)
plot(s100)
data(s121)
plot(s121, type="l")
```

`s256i` *Simulated Data-Set which Illustrate the Usage of krige.bayes*

Description

This is the simulated data-set used in the Technical Report which describes the implementation of the function **krige.bayes** (see reference below).

Usage

```
data(s256i)
```

Format

Two objects of the `class` `geodata`. Both are lists with the following components:

`coords` the coordinates of data locations.

`data` the simulated data.

References

Ribeiro Jr, P.J. and Diggle, P.J. (1999) Bayesian inference in Gaussian model-based geostatistics. *Technical report ST-99-08, Dept of Maths and Stats, Lancaster University*.

Further information about the **geoR** package can be found at:
http://www.est.ufpr.br/geoR.

Examples

```
data(s256i)
points(s256i, pt.div="quintiles", cex.min=1, cex.max=1)
```

`sample.geodata` *Sampling from geodata objects*

Description

This functions facilitates extracting samples from geodata objects.

Usage

```
sample.geodata(x, size, replace = FALSE, prob = NULL, coef.logCox,
               external)
```

Arguments

`x`	an object of the class `geodata`.
`size`	non-negative integer giving the number of items to choose.
`replace`	Should sampling be with replacement?
`prob`	A vector of probability weights for obtaining the elements of the data points being sampled.
`coef.logCox`	optional. A scalar with the coeficient for the log-Cox process. See DETAILS below.
`external`	numeric values of a random field to be used in the log-Cox inhomogeneous poisson process.

Details

If `prob=NULL` and the argument `coef.logCox`, is provided, sampling follows a log-Cox proccess, i.e. the probability of each point being sampled is proportional to:

$$exp(bY(x))$$

with b given by the value passed to the argument `coef.logCox` and $Y(x)$ taking values passed to the argument `external` or, if this is missing, the element `data` of the `geodata` object. Therefore, the latter generates a preferential sampling.

Value

a list which is an object of the class `geodata`.

See Also

`as.geodata`, `sample`.

Examples

```
par(mfrow=c(1,2))
S1 <- grf(2500,  grid="reg", cov.pars=c(1, .23))
image(S1, col=gray(seq(0.9,0.1,l=100)))
y1 <- sample.geodata(S1, 80)
points(y1$coords, pch=19)
## Now a preferential sampling
y2 <- sample.geodata(S1, 80, coef=1.3)
## which is equivalent to
## y2 <- sample.geodata(S1, 80, prob=exp(1.3*S1$data))
points(y2$coords, pch=19, col=2)
## and now a clustered (but not preferential)
S2 <- grf(2500,  grid="reg", cov.pars=c(1, .23))
y3 <- sample.geodata(S1, 80, prob=exp(1.3*S2$data))
## which is equivalent to
## points(y3$coords, pch=19, col=4)
image(S2, col=gray(seq(0.9,0.1,l=100)))
points(y3$coords, pch=19, col=4)
```

---

sample.posterior *Samples from the posterior distribution*

Description

Sample quadruples $(\beta, \sigma^2, \phi, \tau^2_{rel})$ from the posterior distribution returned by `krige.bayes`.

Usage

```
sample.posterior(n, kb.obj)
```

Arguments

`n`	number of samples
`kb.obj`	on object with an output of `krige.bayes`.

Value

A $n \times 4$ data-frame with samples from the posterior distribution of the model parameters.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

krige.bayes and sample.posterior.

`sample.prior` *Sample the prior distribution*

Description

Sample quadruples $(\beta, \sigma^2, \phi, \tau^2_{rel})$ from the prior distribution of parameters specifying a Gaussian random field. Typically the prior is specified in the same manner as when calling `krige.bayes`.

Usage

```
sample.prior(n, kb.obj=NULL, prior=prior.control())
```

Arguments

`n`	number of samples
`kb.obj`	on object with an output of `krige.bayes`.
`prior`	an call to `prior.control`. Unnecessary if `kb.obj` is provided.

Value

A $p + 3 \times 4$ data-frame with a sample of the prior distribution of model parameters, where p is the length of the mean parameter β.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

krige.bayes and sample.posterior.

Examples

```
sample.prior(50, prior=prior.control(beta.prior = "normal", beta = .5, beta.var.std=0.1, sig
```

set.coords.lims *Sets Limits to Scale Plots*

Description

This is an function typically called by functions in the package **geoR** to set limits for the axis when plotting spatial data.

Usage

```
set.coords.lims(coords, xlim, ylim)
```

Arguments

coords	an $n \times 2$ matrix with coordinates.
xlim, ylim	the ranges to be encompassed by the x and y axes.

Value

A 2×2 matrix with limits for the axis.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

soil *Soil chemistry properties data set*

Description

Several soil chemistry properties measured on a 10x25 grid of points.

Usage

```
data(soil)
```

Format

A data frame with 250 observations on the following 22 variables.

Linha x-coordinate

Coluna y-coordinate

Cota elevation

AGrossa a numeric vector

Silte a numeric vector

Argila a numeric vector

pHAgua a numeric vector

pHKCl a numeric vector

Ca a numeric vector, calcium content

Mg a numeric vector, magnesium content

K a numeric vector, potassio content

Al a numeric vector, aluminium content

H a numeric vector, hidrogen content

C a numeric vector, carbon content

N a numeric vector, nitrogenm content

CTC a numeric vector, catium exchange capability

S a numeric vector

V a numeric vector

M a numeric vector

NC a numeric vector

CEC a numeric vector

CN a numeric vector

Source

Bassoi and Moraes

References

Bassoi thesis...

Examples

```
data(soil)
ctc <- as.geodata(soil, data.col=16)
plot(ctc)
```

`statistics.predictive` *Summary statistics from predictive distributions*

Description

Computes summaries based on simulations of predictive distribution returned by `krige.bayes` and `krige.conv`.

Usage

```
statistics.predictive(simuls, mean.var = TRUE, quantile, threshold,
                      sim.means, sim.vars)
```

Arguments

`simuls`	object with simulations from the predictive distribution
`mean.var`	Logical. Indicates whether or not to compute mean and variances of the simulations at each location.
`quantile`	defines quantile estimator. See documentation for `output.control` .
`threshold`	defines probability estimator. See documentation for `output.control`.
`sim.means`	Logical. Indicates whether or not to compute the mean of of the conditional simulations.
`sim.vars`	Logical. Indicates whether or not to compute the variances of the conditional simulations.

Value

A list with one ore more of the following components.

`mean`	mean at each prediction location.
`variance`	variance at each prediction location.
`quantiles`	quantiles, at each prediction location.
`probabilities`	probabilities, at each prediction location, of been below the provided threshold.
`sim.means`	vector with means of each conditional simulation.
`sim.vars`	vector with variances of each conditional simulation.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

subarea *Selects a Subarea from a Geodata Object*

Description

Usage

```
subarea(geodata, xlim, ylim, ...)
```

Arguments

geodata	an object of the class geodata as defined in as.geodata.
xlim	optional, a vector with selected range of the x-coordinates.
ylim	optional, a vector with selected range of the y-coordinates.
...	further arguments to be passed to zoom.coords.

Details

The function copies the original geodata object and selects values of $coords, $data, $borders, $covariate and $units.m which lies within the selected sub-area. Remaining components of the geodata objects are untouched.

If xlim and/or ylim are not provided the function prompts the user to click 2 points defining an retangle defining the subarea on a existing plot.

Value

Returns an geodata object, subsetting of the original one provided.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

See Also

zoom.coords, locator

Examples

```
foo <- matrix(c(4,6,6,4,2,2,4,4), nc=2)
foo1 <- zoom.coords(foo, 2)
foo1
foo2 <- coords2coords(foo, c(6,10), c(6,10))
foo2
plot(1:10, 1:10, type="n")
```

```
polygon(foo)
polygon(foo1, lty=2)
polygon(foo2, lwd=2)
arrows(foo[,1], foo[,2],foo1[,1],foo1[,2], lty=2)
arrows(foo[,1], foo[,2],foo2[,1],foo2[,2])
legend(1,10, c("foo", "foo1 (zoom.coords)", "foo2 (coords2coords)"), lty=c(1,2,1), lwd=c(1,1

## "zooming" part of The Gambia map
data(gambia)
gb <- gambia.borders/1000
gd <- gambia[,1:2]/1000
plot(gb, ty="l", asp=1, xlab="W-E (kilometres)", ylab="N-S (kilometres)")
points(gd, pch=19, cex=0.5)
r1b <- gb[gb[,1] < 420,]
rc1 <- rect.coords(r1b, lty=2)

r1bn <- zoom.coords(r1b, 1.8, xoff=90, yoff=-90)
rc2 <- rect.coords(r1bn, xz=1.05)
segments(rc1[c(1,3),1],rc1[c(1,3),2],rc2[c(1,3),1],rc2[c(1,3),2], lty=3)

lines(r1bn)
r1d <- gd[gd[,1] < 420,]
r1dn <- zoom.coords(r1d, 1.7, xlim.o=range(r1b[,1],na.rm=TRUE), ylim.o=range(r1b[,2],na.rm=T
points(r1dn, pch=19, cex=0.5)
text(450,1340, "Western Region", cex=1.5)

if(require(geoRglm)){
data(rongelap)
points(rongelap, bor=borders)
## zooming the western area
rongwest <- subarea(rongelap, xlim=c(-6300, -4800))
points(rongwest, bor=borders)
## now zooming in the same plot
points(rongelap, bor=borders)
rongwest.z <- zoom.coords(rongwest, xzoom=3, xoff=2000, yoff=3000)
points(rongwest.z, bor=borders, add=TRUE)
rect.coords(rongwest$sub, quiet=TRUE)
rect.coords(rongwest.z$sub, quiet=TRUE)
}
```

`subset.geodata` *Method for subsetting geodata objects*

Description

Subsets a object of the class `geodata` by transforming it to a data-frame, using `subset` and back transforming to a `geodata` object.

Usage

```
## S3 method for class 'geodata':
subset(x, ..., other = TRUE)
```

Arguments

x	an object of the class geodata.
...	arguments to be passed to subset.data.frame.
other	logical. If TRUE non-standard geodata elements of the original geodata object are copied to the resulting object.

Value

A list which is an object of the class geodata.

See Also

subset for the generic function and methods and as.geodata for more information on geodata objects.

Examples

```
data(ca20)
subset(ca20, data > 70)
subset(ca20, area == 1)
```

---

summary.geodata *Summaries for geodata object*

Description

Sumarises each of the main elements of an object of the class geodata.

Usage

```
## S3 method for class 'geodata':
summary(object, lambda =1, add.to.data = 0,
                by.realisations=TRUE, ...)
```

Arguments

object	an object of the class geodata.
lambda	value of the Box-Cox transformation parameter. Two particular cases are $\lambda = 1$ which corresponds to no transformation and $\lambda = 0$ corresponding to the log-transformation.
add.to.data	scalar, Constant value to be added to the data values. Only used if a value different from 1 is passed to the argument lambda.

`by.realisations`
: logical. Indicates whether the summary must be performed separatly for each realisation, if the `geodata` object contains the element `realisations`. Defaults to `TRUE`.

`...`
: further arguments to be passed to the function `summary.default`.

Value

A list with components

`coords.summary`
: a matrix with minimum and maximum values for the coordinates.

`distances.summary`
: minimum and maximum distances between pairs of points.

`borders.summary`
: a matrix with minimum and maximum values for the coordinates. Only returned if there is an element `borders` in the `geodata` object.

`data.summary`
: summary statistics (min, max, quartiles and mean) for the data.

`units.m.summary`
: summary statistics (min, max, quartiles and mean) for the offset variable. Only returned if there is an element `units.m` in the `geodata` object.

`covariate.summary`
: summary statistics (min, max, quartiles and mean) for the covariate(s). Only returned if there is an element `covariate` in the `geodata` object.

`others`
: names of other elements if present in the `geodata` object.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`summary`, `as.geodata`.

Examples

```
data(s100)
summary(s100)

data(ca20)
summary(ca20)
```

summary.likGRF *Summarizes Parameter Estimation Results for Gaussian Random Fields*

Description

Summarizes results returned by the function likfit.
Functions are *methods* for summary and print for the classes likGRF and summary.likGRF.

Usage

```
## S3 method for class 'likGRF':
summary(object, ...)
## S3 method for class 'likGRF':
print(x, digits = max(3, getOption("digits") - 3), ...)
## S3 method for class 'summary.likGRF':
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object	an object of class *likGRF*, typically a result of a call to likfit.
x	an object of class *likGRF* or class *summary.likGRF*, typically resulting from a call to likfit.
digits	the number of significant digits to use when printing.
...	extra arguments for print.

Details

A detailed summary of a object of the class likGRF is produced by by summary.likGRF and printed by print.summary.likGRF. This includes model specification with values of fixed and estimated parameters. A simplified summary of the parameter estimation is printed by print.likGRF.

Value

print.likGRF prints the parameter estimates and the value of the maximized likelihood.
summary.likGRF returns a list with main results of a call to likfit.
print.summary.likGRF prints these results on the screen (or other output device) in a "nice" format.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

likfit, print, summary.

Examples

```
# See examples for the function likfit()
```

---

summary.variofit *Summarize Results of Variogram Estimation*

Description

This function prints a summary of the parameter estimation results given by variofit.

Usage

```
## S3 method for class 'variofit':
summary(object, ...)
```

Arguments

object	an object of the class "variomodel" typically an output of variofit.
...	other arguments to be passed to the function print or summary.

Value

Prints a summary of the estimation results on the screen or other output device.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

The functions variofit for variogram based estimation. For likelihood based parameter estimation see likfit.

Examples

```
data(s100)
s100.vario <- variog(s100, max.dist=1)
wls <- variofit(s100.vario, ini=c(.5, .5), fix.nugget = TRUE)
wls
summary(wls)
```

`tce` *TCE concentrations in groundwater in a vertical cross section*

Description

Measurements at 56 locations of concentration of trichloroethylene (TCE) in groundwater on a transect in a fine-sand superficial aquifer. Extract from Kitanidis' book.

Usage

```
data(tce)
```

Format

An object of the class `geodata` which is a list with the elements:

coords coordinates of the data location (feet).

data the data vector with measurements of the TCE concentration (ppb).

Source

Kitanidis, P.K. Introduction to geostatistics - applications in hidrology (1997). Cambridge University Press.

Examples

```
data(tce)
summary(tce)
summary(tce, lambda=0)
plot(tce)
points(tce)
points(tce, lambda=0)
```

`trend.spatial` *Builds the Trend Matrix*

Description

Builds the *trend* matrix in accordance to a specification of the mean provided by the user.

Usage

```
trend.spatial(trend, geodata, add.to.trend)
```

Arguments

`trend`	specifies the mean part of the model. See `DETAILS` below.
`geodata`	optional. An object of the class `geodata` as described in `as.geodata`.
`add.to.trend`	optional. Specifies aditional terms to the mean part of the model. See details below.

Details

The implicity model assumes that there is an underlying process with mean $\mu(x)$, where $x = (x_1, x_2)$ denotes the coordinates of a spatial location. The argument `trend` defines the form of the mean and the following options are allowed:

`"cte"` the mean is assumed to be constant over the region, in which case $\mu(x) = \mu$. This is the default option.

`"1st"` the mean is assumed to be a first order polynomial on the coordinates:

$$\mu(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

`"2nd"` the mean is assumed to be a second order polynomial on the coordinates:

$$\mu(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 (x_1)^2 + \beta_4 (x_2)^2 + \beta_5 x_1 * x_2$$

`~ model` a model specification. See `formula` for further details on how to specify a model in R using formulas. Notice that the model term before the ~ is not necessary. Typically used to include covariates (external trend) in the model.

Denote by x_1 and x_2 the spatial coordinates. The following specifications are equivalent:

- `trend = "1st"` and `trend = ~ x1 + x2`
- `trend = "2nd"` and `trend = ~ x1 + x2 + I(x1^2) + I(x2^2) + I(x1*x2)`

Search path for covariates
Typically, functions in the package **geoR** which calls `trend.spatial` will have the arguments `geodata`, `coords` and `data`.

When the trend is specifed as `trend = ~ model` the terms included in the model will be searched for in the following path sequence (in this order):

1. in the users/session Global environment
2. in the session search path
3. as elements of the list `geodata`
4. as columns in a data-frame `geodata$covariates`
5. as columns in a data-frame `geodata$data`
6. in remainder of the session search path

The argument `add.to.trend` adds terms to what is specified in the argument `trend`. This seems redundant but allow specifications of the type: `trend="2nd", add.trend=~other.covariates`.

Value

An object of the class `trend.spatial` which is an $n \times p$ *trend* matrix, where n is the number of spatial locations and p is the number of mean parameters in the model.

Note

This is an auxiliary function typically called by other **geoR** functions.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

The section `DETAILS` in the documentation for `likfit` for more about the underlying model.

Examples

```
data(SIC)
# a first order polynomial trend
trend.spatial("1st", sic.100)[1:5,]
# a second order polynomial trend
trend.spatial("2nd", sic.100)[1:5,]
# a trend with a covariate
trend.spatial(~altitude, sic.100)[1:5,]
# a first degree trend plus a covariate
trend.spatial(~coords+altitude, sic.100)[1:5,]
# with produces the same as
trend.spatial("1st", sic.100, add=~altitude)[1:5,]
# and yet another exemple
trend.spatial("2nd", sic.100, add=~altitude)[1:5,]
```

varcov.spatial *Computes Covariance Matrix and Related Results*

Description

This function builds the covariance matrix for a set of spatial locations, given the covariance parameters. According to the input options other results related to the covariance matrix (such as decompositions, determinants, inverse. etc) can also be returned.

Usage

```
varcov.spatial(coords = NULL, dists.lowertri = NULL,
               cov.model = "matern", kappa = 0.5, nugget = 0,
               cov.pars = stop("no cov.pars argument"),
               inv = FALSE, det = FALSE,
               func.inv = c("cholesky", "eigen", "svd", "solve"),
               scaled = FALSE,  only.decomposition = FALSE,
               sqrt.inv = FALSE, try.another.decomposition = TRUE,
               only.inv.lower.diag = FALSE, ...)
```

Arguments

coords	an $n \times 2$ matrix with the coordinates of the data locations. If not provided the argument dists.lowertri should be provided instead.
dists.lowertri	a vector with the lower triangle of the matrix of distances between pairs of data points. If not provided the argument coords should be provided instead.
cov.model	a string indicating the type of the correlation function. More details in the documentation for cov.spatial. Defaults are equivalent to the *exponential* model.
kappa	values of the additional smoothness parameter, only required by the following correlation functions: "matern", "powered.exponential", "cauchy" and "gneiting.matern".
nugget	the value of the nugget parameter τ^2.
cov.pars	a vector with 2 elements or an $ns \times 2$ matrix with the covariance parameters. The first element (if a vector) or first column (if a matrix) corresponds to the variance parameter σ^2. second element or column corresponds to the correlation function parameter ϕ. If a matrix is provided each row corresponds to the parameters of one *spatial structure*. Models with several structures are also called *nested models* in the geostatistical literature.
inv	if TRUE the inverse of covariance matrix is returned. Defaults to FALSE.
det	if TRUE the logarithmic of the square root of the determinant of the covariance matrix is returned. Defaults to FALSE.

`func.inv` algorithm used for the decomposition and inversion of the covariance matrix. Options are `"chol"` for Cholesky decomposition, `"svd"` for singular value decomposition and `"eigen"` for eigenvalues/eigenvectors decomposition. Defaults to `"chol"`.

`scaled` logical indicating whether the covariance matrix should be scaled. If `TRUE` the partial sill parameter σ^2 is set to 1. Defaults to `FALSE`.

`only.decomposition` logical. If `TRUE` only the square root of the covariance matrix is returned. Defaults to `FALSE`.

`sqrt.inv` if `TRUE` the square root of the inverse of covariance matrix is returned. Defaults to `FALSE`.

`try.another.decomposition` logical. If `TRUE` and the argument `func.inv` is one of `"cholesky"`, `"svd"` or `"solve"`, the matrix decomposition or inversion is tested and, if it fails, the argument `func.inv` is re-set to `"eigen"`.

`only.inv.lower.diag` logical. If `TRUE` only the lower triangle and the diagonal of the inverse of the covariance matrix are returned. Defaults to `FALSE`.

`...` for naw, only for internal usage.

Details

The elements of the covariance matrix are computed by the function `cov.spatial`. Typically this is an auxiliary function called by other functions in the **geoR** package.

Value

The result is always list. The components will vary according to the input options. The possible components are:

`varcov` the covariance matrix.

`sqrt.varcov` a square root of the covariance matrix.

`lower.inverse` the lower triangle of the inverse of covariance matrix.

`diag.inverse` the diagonal of the inverse of covariance matrix.

`inverse` the inverse of covariance matrix.

`sqrt.inverse` a square root of the inverse of covariance matrix.

`log.det.to.half` the logarithmic of the square root of the determinant of the covariance matrix.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

cov.spatial for more information on the correlation functions; chol, solve, svd and eigen for matrix inversion and/or decomposition.

---

varcovBGCCM *Covariance matrix for the bivariate Gaussian common component geostatistical model*

Description

Covariance matrix for the bivariate Gaussian common component geostatistical model or its inverse, and optionally the determinant of the matrix.

Usage

```
varcovBGCCM(dists.obj, cov0.pars, cov1.pars, cov2.pars,
            cov0.model = "matern", cov1.model = "matern",
            cov2.model = "matern", kappa0 = 0.5, kappa1 = 0.5,
            kappa2 = 0.5, scaled = TRUE, inv = FALSE, det = FALSE)
```

Arguments

dists.obj	Describe dists.obj here
cov0.pars	Describe cov0.pars here
cov1.pars	Describe cov1.pars here
cov2.pars	Describe cov2.pars here
cov0.model	Describe cov0.model here
cov1.model	Describe cov1.model here
cov2.model	Describe cov2.model here
kappa0	Describe kappa0 here
kappa1	Describe kappa1 here
kappa2	Describe kappa2 here
scaled	Describe scaled here
inv	logical. If TRUE the inverse of the covariance matrix is returned instead.
det	logical. Optional, if TRUE the logarithm of the detarminant of the covariance matrix is returned as an attribute.

Value

A matrix which is the covariance matrix for the bivariate Gaussian common component geostatistical model or its inverse if `inv=TRUE`. If `det=T` the logarithm of the determinant of the matrix is also returned as an attribute named `logdetS`.

Warning

This is a new function and still in draft format and pretty much untested.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

See Also

`cov.spatial`, `varcov.spatial`

Examples

```
# see http://www.est.ufpr.br/geoR/tutorials/CCM.R
```

---

variofit *Variogram Based Parameter Estimation*

Description

Estimate covariance parameters by fitting a parametric model to a empirical variogram. Variograms models can be fitted by using weighted or ordinary least squares.

Usage

```
variofit(vario, ini.cov.pars, cov.model = "matern",
         fix.nugget = FALSE, nugget = 0,
         fix.kappa = TRUE, kappa = 0.5,
         simul.number = NULL, max.dist = vario$max.dist,
         weights, minimisation.function,
         limits = pars.limits(), messages, ...)
```

Arguments

`vario`	an object of the class `"variogram"`, typically an output of the function `variog`. The object is a list with information about the empirical variogram.
`ini.cov.pars`	initial values for the covariance parameters: σ^2 (partial sill) and ϕ (range parameter). See `DETAILS` below.

`cov.model` a string with the name of the correlation function. For further details see documentation for `cov.spatial`. For the linear model use `cov.model = "linear"`. Defaults are equivalent to the *exponential* model.

`fix.nugget` logical, indicating whether the parameter τ^2 (nugget variance) should be regarded as fixed (`fix.nugget = TRUE`) or should be estimated (`fix.nugget = FALSE`). Defaults to `FALSE`.

`nugget` value for the nugget parameter. Regarded as a fixed values if `fix.nugget = TRUE` or as a initial value for the minimization algorithm if `fix.nugget = FALSE`. Defaults to zero.

`fix.kappa` logical, indicating whether the parameter κ should be regarded as fixed or be estimated. Defaults to `TRUE`.

`kappa` value of the smoothness parameter. Regarded as a fixed values if `fix.kappa = TRUE` or as a initial value for the minimization algorithm if `fix.kappa = FALSE`. Only required if one of the following correlation functions is used: `"matern"`, `"powered.exponential"`, `"cauchy"` and `"gneiting.matern"`. Defaults to 0.5.

`simul.number` number of simulation. To be used when the object passed to the argument `vario` has empirical variograms for more than one data-set (or simulation). Indicates to which one the model will be fitted.

`max.dist` maximum distance considered when fitting the variogram. Defaults to `vario$max.dist`.

`weights` type weights used in the loss function. See `DETAILS` below.

`limits` values defining lower and upper limits for the model parameters used in the numerical minimisation. Only valid if `minimisation.function = "optim"`. The auxiliary function `pars.limits` is called to set the limits.

`minimisation.function` minimization function used to estimate the parameters. Options are `"optim"`, `"nlm"`. If `weights = "equal"` the option `"nls"` is also valid and det as default. Otherwise defaults to `"optim"`.

`messages` logical. Indicates whether or not status messages are printed on the screen (or other output device) while the function is running.

`...` further parameters to be passed to the minimization function. Typically arguments of the type `control()` which controls the behavior of the minimization algorithm. See documentation for the selected minimization function for further details.

Details

Numerical minimization

The parameter values are found by numerical optimization using one of the functions: `optim`, `nlm` and `nls`. In given circunstances the algorithm may not converge to correct parameter values when called with default options and the user may need to pass extra options for the optimizers. For instance the function `optim` takes a `control` argument. The user should try different initial values and if the parameters have different orders of magnitude may need to use options to scale the parameters. Some possible workarounds in case of problems include:

- rescale you data values (dividing by a constant, say)

- rescale your coordinates (subtracting values and/or dividing by constants)
- Use the mechanism to pass `control()` options for the optimiser internally

Initial values

The algorithms for minimization functions require initial values of the parameters.

A unique initial value is used if a vector is provided in the argument `ini.cov.pars`. The elements are initial values for σ^2 and ϕ, respectively. This vector is concatenated with the value of the argument `nugget` if `fix.nugget = FALSE` and `kappa` if `fix.kappa = TRUE`.

Specification of multiple initial values is also possible. If this is the case, the function searches for the one which minimizes the loss function and uses this as the initial value for the minimization algorithm. Multiple initial values are specified by providing a matrix in the argument `ini.cov.pars` and/or, vectors in the arguments `nugget` and `kappa` (if included in the estimation). If `ini.cov.pars` is a matrix, the first column has values of σ^2 and the second has values of ϕ.

Alternatively the argument `ini.cov.pars` can take an object of the class `eyefit` or `variomodel`. This allows the usage of an output of the functions `eyefit`, `variofit` or `likfit` be used as initial value.

If `minimisation.function = "nls"` only the values of ϕ and κ (if this is included in the estimation) are used. Values for the remaning are not need by the algorithm.

If `cov.model = "linear"` only the value of σ^2 is used. Values for the remaning are not need by this algorithm.

If `cov.model = "pure.nugget"` no initial values are needed since no minimisation function is used.

Weights

The different options for the argument `weights` are used to define the loss function to be minimised. The available options are as follows.

`"npairs"` indicates that the weights are given by the number of pairs in each bin. This is the default option unless `variog$output.type == "cloud"`. The loss function is:

$$LOSS(\theta) = \sum_k n_k[(\hat{\gamma}_k) - \gamma_k(\theta)]^2$$

`"cressie"` weights as suggested by Cressie (1985).

$$LOSS(\theta) = \sum_k n_k[\frac{\hat{\gamma}_k - \gamma_k(\theta)}{\gamma_k(\theta)}]^2$$

`"equal"` equal values for the weights. For this case the estimation corresponds to the ordinary least squares variogram fitting. This is the default option if `variog$output.type == "cloud"`.

$$LOSS(\theta) = \sum_k [(\hat{\gamma}_k) - \gamma_k(\theta)]^2$$

Where θ is the vector with the variogram parameters and for each k^{th}-bin n_k is the number of pairs, $(\hat{\gamma}_k)$ is the value of the empirical variogram and $\gamma_k(\theta)$ is the value of the theoretical variogram.

See also Cressie (1993) and Barry, Crowder and Diggle (1997) for further discussions on methods to estimate the variogram parameters.

Value

An object of the class "variomodel" and "variofit" which is list with the following components:

nugget	value of the nugget parameter. An estimated value if fix.nugget = FALSE or a fixed value if fix.nugget = TRUE.
cov.pars	a two elements vector with estimated values of the covariance parameters σ^2 and ϕ, respectively.
cov.model	a string with the name of the correlation function.
kappa	fixed value of the smoothness parameter.
value	minimized value of the loss function.
max.dist	maximum distance considered in the variogram fitting.
minimisation.function	minimization function used.
weights	a string indicating the type of weights used for the variogram fitting.
method	a string indicating the type of variogram fitting method (OLS or WLS).
fix.kappa	logical indicating whether the parameter κ was fixed.
fix.nugget	logical indicating whether the nugget parameter was fixed.
lambda	transformation parameters inherith from the object provided in the argument vario.
message	status messages returned by the function.
call	the function call.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Barry, J.T., Crowder, M.J. and Diggle, P.J. (1997) Parametric estimation of the variogram. *Tech. Report, Dept Maths & Stats, Lancaster University*.

Cressie, N.A.C (1985) *Mathematical Geology*. **17**, 563-586.

Cressie, N.A.C (1993) *Statistics for Spatial Data*. New York: Wiley.

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

cov.spatial for a detailed description of the available correlation (variogram) functions, likfit for maximum and restricted maximum likelihood estimation, lines.variomodel for graphical output of the fitted model. For details on the minimization functions see optim, nlm and nls.

Examples

```
data(s100)
vario100 <- variog(s100, max.dist=1)
ini.vals <- expand.grid(seq(0,1,l=5), seq(0,1,l=5))
ols <- variofit(vario100, ini=ini.vals, fix.nug=TRUE, wei="equal")
summary(ols)
wls <- variofit(vario100, ini=ini.vals, fix.nug=TRUE)
summary(wls)
plot(vario100)
lines(wls)
lines(ols, lty=2)
```

---

variog *Compute Empirical Variograms*

Description

Computes sample (empirical) variograms with options for the *classical* or *robust* estimators. Output can be returned as a `binned variogram`, a `variogram cloud` or a `smoothed variogram`. Data transformation (Box-Cox) is allowed. "Trends" can be specified and are fitted by ordinary least squares in which case the variograms are computed using the residuals.

Usage

```
variog(geodata, coords = geodata$coords, data = geodata$data,
       uvec = "default", breaks = "default",
       trend = "cte", lambda = 1,
       option = c("bin", "cloud", "smooth"),
       estimator.type = c("classical", "modulus"),
       nugget.tolerance, max.dist, pairs.min = 2,
       bin.cloud = FALSE, direction = "omnidirectional", tolerance = pi/8,
       unit.angle = c("radians","degrees"), angles = FALSE, messages, ...)
```

Arguments

`geodata`	a list containing element `coords` as described next. Typically an object of the class `"geodata"` - a **geoR** data-set. If not provided the arguments `coords` must be provided instead.
`coords`	an $n \times 2$ matrix containing coordinates of the n data locations in each row. Defaults to `geodata$coords`, if provided.
`data`	a vector or matrix with data values. If a matrix is provided, each column is regarded as one variable or realization. Defaults to `geodata$data`, if provided.

`uvec` a vector with values used to define the variogram binning. Only used when `option = "bin"`. See `DETAILS` below for more details on how to specify the bins.

`breaks` a vector with values to define the variogram binning. Only used when `option = "bin"`. See `DETAILS` below for more details on how to specify the bins.

`trend` specifies the mean part of the model. See documentation of `trend.spatial` for further details. Defaults to `"cte"`.

`lambda` values of the Box-Cox transformation parameter. Defaults to 1 (no transformation). If another value is provided the variogram is computed after transforming the data. A case of particular interest is $\lambda = 0$ which corresponds to log-transformation.

`option` defines the output type: the options `"bin"` returns values of binned variogram, `"cloud"` returns the variogram cloud and `"smooth"` returns the kernel smoothed variogram. Defaults to `"bin"`.

`estimator.type`
`"classical"` computes the classical method of moments estimator. `"modulus"` returns the variogram estimator suggested by Hawkins and Cressie (see Cressie, 1993, pg 75). Defaults to `"classical"`.

`nugget.tolerance`
a numeric value. Points which are separated by a distance less than this value are considered co-located. Defaults to zero.

`max.dist` a numerical value defining the maximum distance for the variogram. Pairs of locations separated for distance larger than this value are ignored for the variogram calculation. If not provided defaults takes the maximum distance among all pairs of data locations.

`pairs.min` a integer number defining the minimum numbers of pairs for the bins. For `option = "bin"`, bins with number of pairs smaller than this value are ignored. Defaults to `NULL`.

`bin.cloud` logical. If `TRUE` and `option = "bin"` the cloud values for each class are included in the output. Defaults to `FALSE`.

`direction` a numerical value for the directional (azimuth) angle. This used to specify directional variograms. Default defines the omnidirectional variogram. The value must be in the interval $[0, \pi]$ radians ($[0, 180]$ degrees).

`tolerance` numerical value for the tolerance angle, when computing directional variograms. The value must be in the interval $[0, \pi/2]$ radians ($[0, 90]$ degrees). Defaults to $\pi/8$.

`unit.angle` defines the unit for the specification of angles in the two previous arguments. Options are `"radians"` and `"degrees"`, with default to `"radians"`.

`angles` Logical with default to `FALSE`. If `TRUE` the function also returns the angles between the pairs of points (unimplemented).

`messages` logical. Indicates whether status messages should be printed on the screen (or output device) while the function is running.

`...` arguments to be passed to the function `ksmooth`, if `option = "smooth"`.

Details

Variograms are widely used in geostatistical analysis for exploratory purposes, to estimate covariance parameters and/or to compare theoretical and fitted models against sample variograms.

Estimators

The two estimators currently implemented are:

- *classical* (method of moments) estimator:

$$\gamma(h) = \frac{1}{2N_h}\sum_{i=1}^{N_h}[Y(x_{i+h}) - Y(x_i)]^2$$

- Hawkins and Cressie's *modulus* estimator

$$\gamma(h) = \frac{[\frac{1}{N_h}\sum_{i=1}^{N_h}|Y(x_{i+h}) - Y(x_i)|^{\frac{1}{2}}]^4}{0.914 + \frac{0.988}{N_h}}$$

Defining the bins

The defaults

If arguments `breaks` and `uvec` are not provided, the binning is defined as follows:

1. read the argument `max.dist`. If not provided it is set to the maximum distance between the pairs of points.
2. the center of the bins are initially defined by the sequence `u = seq(0, max.dist, l = 13)`.
3. the interval spanned by each bin is given by the mid-points between the centers of the bins.

If an vector is passed to the argument `breaks` its elements are taken as the limits of the bins (classes of distance) and the argument `uvec` is ignored.

Variations on the default

The default definition of the bins is different for some particular cases.

1. if there are coincident data locations the bins follows the default above but one more bin is added at the origin (distance zero) for the collocated points.
2. if the argument `nugget.tolerance` is provided the separation distance between all pairs in the interval $[0, nugget.tolerance]$ are set to zero. The first bin distance is set to zero (`u[1] = 0`). The remaining bins follows the default.
3. if a scalar is provided to the argument `uvec` the default number of bins is defined by this number.
4. if a vector is provided to the argument `uvec`, its elements are taken as central points of the bins.

Value

An object of the `class` `variogram` which is a list with the following components:

`u` a vector with distances.

`v`	a vector with estimated variogram values at distances given in `u`.
`n`	number of pairs in each bin, if `option = "bin"`.
`sd`	standard deviation of the values in each bin.
`bins.lim`	limits defining the interval spanned by each bin.
`ind.bin`	a logical vector indicating whether the number of pairs in each bin is greater or equal to the value in the argument `pairs.min`.
`var.mark`	variance of the data.
`beta.ols`	parameters of the mean part of the model fitted by ordinary least squares.
`output.type`	echoes the `option` argument.
`max.dist`	maximum distance between pairs allowed in the variogram calculations.
`estimator.type`	echoes the type of estimator used.
`n.data`	number of data.
`lambda`	value of the transformation parameter.
`trend`	trend specification.
`nugget.tolerance`	value of the nugget tolerance argument.
`direction`	direction for which the variogram was computed.
`tolerance`	tolerance angle for directional variogram.
`uvec`	lags provided in the function call.
`call`	the function call.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Cressie, N.A.C (1993) *Statistics for Spatial Data*. New York: Wiley.

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`variog4` for more on computation of directional variograms, `variog.model.env` and `variog.mc.env` for variogram envelopes, `variofit` for variogram based parameter estimation and `plot.variogram` for graphical output.

Examples

```
# Loading data:
data(s100)
#
# computing variograms:
#
# binned variogram
vario.b <- variog(s100, max.dist=1)
# variogram cloud
vario.c <- variog(s100, max.dist=1, op="cloud")
#binned variogram and stores the cloud
vario.bc <- variog(s100, max.dist=1, bin.cloud=TRUE)
# smoothed variogram
vario.s <- variog(s100, max.dist=1, op="sm", band=0.2)
#
#
# plotting the variograms:
par(mfrow=c(2,2))
plot(vario.b, main="binned variogram")
plot(vario.c, main="variogram cloud")
plot(vario.bc, bin.cloud=TRUE, main="clouds for binned variogram")
plot(vario.s, main="smoothed variogram")

# computing a directional variogram
vario.0 <- variog(s100, max.dist=1, dir=0, tol=pi/8)
plot(vario.b, type="l", lty=2)
lines(vario.0)
legend(0, 1.2, legend=c("omnidirectional", expression(0 * degree)), lty=c(2,1))
```

---

variog.mc.env *Envelops for Empirical Variograms Based on Permutation*

Description

Computes envelops for empirical variograms by permutation of the data values on the spatial locations.

Usage

```
variog.mc.env(geodata, coords = geodata$coords, data = geodata$data,
              obj.variog, nsim = 99, save.sim = FALSE, messages)
```

Arguments

`geodata`	a list containing elements `coords` and `data` as described next. Typically an object of the class `"geodata"` - a **geoR** data-set. If not provided the arguments `coords` and `data` must be provided instead.
`coords`	an $n \times 2$ matrix, each row containing Euclidean coordinates of the n data locations. By default it takes the element `coords` of the argument `geodata`.

data	a vector with the data values. By default it takes the element data of the argument geodata.
obj.variog	an object of the class "variogram", typically an output of the function variog.
nsim	number of simulations used to compute the envelope. Defaults to 99.
save.sim	logical. Indicates whether or not the simulated data are included in the output. Defaults to FALSE.
messages	logical. If TRUE, the default, status messages are printed while the function is running.

Details

The envelops are obtained by permutation. For each simulations data values are randomly allocated to the spatial locations. The empirical variogram is computed for each simulation using the same lags as for the variogram originally computed for the data. The envelops are computed by taking, at each lag, the maximum and minimum values of the variograms for the simulated data.

Value

An object of the class "variogram.envelope" which is a list with the following components:

u	a vector with distances.
v.lower	a vector with the minimum variogram values at each distance in u.
v.upper	a vector with the maximum variogram values at each distance in u.
simulations	a matrix with simulated data. Only returned if save.sim = TRUE.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

variog.model.env for envelops computed by from a model specification, variog for variogram calculations, plot.variogram and variog.mc.env for graphical output.

Examples

```
if(is.R()) data(s100)
s100.vario <- variog(s100, max.dist=1)
s100.env <- variog.mc.env(s100, obj.var = s100.vario)
plot(s100.vario, envelope = s100.env)
```

`variog.model.env` *Envelops for Empirical Variograms Based on Model Parameters*

Description

Computes envelopes for a empirical variogram by simulating data for given model parameters.

Computes bootstrap paremeter estimates

Usage

```
variog.model.env(geodata, coords = geodata$coords, obj.variog,
                 model.pars, nsim = 99, save.sim = FALSE, messages)

boot.variofit(geodata, coords = geodata$coords, obj.variog,
              model.pars, nsim = 99, trace = FALSE, messages)
```

Arguments

`geodata`	a list containing element `coords` as described next. Typically an object of the class `"geodata"` - a **geoR** data-set. If not provided the argument `coords` must be provided instead.
`coords`	an $n \times 2$ matrix, each row containing Euclidean coordinates of the n data locations. By default it takes the element `coords` of the argument `geodata`.
`obj.variog`	an object of the class `"variogram"`, typically an output of the function `variog`.
`model.pars`	a list with model specification and parameter values. The input is typically an object of the class `variomodel` which is an output of `likfit`, `variofit`. The required components of the list are: • `beta`, the mean parameter. Defaults to zero. • `cov.model`, the covariance model. Defaults to "exponential". • `cov.pars`, the covariance parameters σ^2 and ϕ. • `kappa`, the extra covariance parameters for some of the covariance models. Defaults to 0.5. • `nugget`, the error component variance. Defaults to zero. • `estimator.type`, the type of variogram estimator. Options for `"classical"` and `"robust"`. Defaults to `obj.variog$estimator`.
`nsim`	number of simulations used to compute the envelopes. Defaults to 99.
`save.sim`	logical. Indicates whether or not the simulated data are included in the output. Defaults to `FALSE`.
`trace`	logical. If `TRUE` the fitted values for the bootstrap parameter estimation are printend while the function is running.
`messages`	logical. If `TRUE`, the default, status messages are printed while the function is running.

Details

The envelopes are computed assuming a (transformed) Gaussian random field model. Simulated values are generated at the data locations, given the model parameters. The empirical variogram is computed for each simulation using the same lags as for the original variogram of the data. The envelopes are computed by taking, at each lag, the maximum and minimum values of the variograms for the simulated data.

Value

An object of the class "variogram.envelope" which is a list with the components:

u	a vector with distances.
v.lower	a vector with the minimum variogram values at each distance in u.
v.upper	a vector with the maximum variogram values at each distance in u.
simulations	a matrix with the simulated data. Only returned if save.sim = TRUE.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

variog.mc.env for envelops computed by using data permutation, variog for variogram calculations, plot.variogram and variog.mc.env for graphical output. The functions likfit, variofit are used to estimate the model parameters.

Examples

```
if(is.R()) data(s100)
s100.ml <- likfit(s100, ini = c(0.5, 0.5), fix.nugget = TRUE)
s100.vario <- variog(s100, max.dist = 1)
s100.env <- variog.model.env(s100, obj.v = s100.vario,
                             model.pars = s100.ml)
plot(s100.vario, env = s100.env)
```

variog4 *Computes Directional Variograms*

Description

Computes directional variograms for 4 directions provided by the user.

Usage

```
variog4(geodata, coords = geodata$coords, data = geodata$data,
        uvec = "default", breaks = "default", trend = "cte", lambda = 1,
        option = c("bin", "cloud", "smooth"),
        estimator.type = c("classical", "modulus"),
        nugget.tolerance, max.dist, pairs.min = 2,
        bin.cloud = FALSE, direction = c(0, pi/4, pi/2, 3*pi/4), tolerance = pi/8,
        unit.angle = c("radians", "degrees"), messages, ...)
```

Arguments

geodata	a list containing element coords as described next. Typically an object of the class "geodata" - a **geoR** data-set. If not provided the arguments coords must be provided instead.
coords	an $n \times 2$ matrix containing coordinates of the n data locations in each row. Defaults to geodata$coords, if provided.
data	a vector or matrix with data values. If a matrix is provided, each column is regarded as one variable or realization. Defaults to geodata$data, if provided.
uvec	a vector with values to define the variogram binning. For further details see documentation for variog.
breaks	a vector with values to define the variogram binning. For further details see documentation for variog.
trend	specifies the mean part of the model. The options are: "cte" (constant mean), "1st" (a first order polynomial on the coordinates), "2nd" (a second order polynomial on the coordinates), or a formula of the type ~X where X is a matrix with the covariates (external trend). Defaults to "cte".
lambda	values of the Box-Cox transformation parameter. Defaults to 1 (no transformation). If another value is provided the variogram is computed after transforming the data. A case of particular interest is $\lambda = 0$ which corresponds to log-transformation.
option	defines the output type: the options "bin" returns values of binned variogram, "cloud" returns the variogram cloud and "smooth" returns the kernel smoothed variogram. Defaults to "bin".
estimator.type	"classical" computes the classical method of moments estimator. "modulus" returns the variogram estimator suggested by Hawkins and Cressie (see Cressie, 1993, pg 75). Defaults to "classical".

`nugget.tolerance` a numeric value. Points which are separated by a distance less than this value are considered co-located. Defaults to zero.

`max.dist` a numerical value defining the maximum distance for the variogram. Pairs of locations separated for distance larger than this value are ignored for the variogram calculation. Defaults to the maximum distance among the pairs of data locations.

`pairs.min` a integer number defining the minimum numbers of pairs for the bins. For `option = "bin"`, bins with number of pairs smaller than this value are ignored. Defaults to `NULL`.

`bin.cloud` logical. If `TRUE` and `option = "bin"` the cloud values for each class are included in the output. Defaults to `FALSE`.

`direction` a vector with values of 4 angles, indicating the directions for which the variograms will be computed. Default corresponds to `c(0, 45, 90, 135)` (degrees).

`tolerance` numerical value for the tolerance angle, when computing directional variograms. The value must be in the interval $[0, 90]$ degrees. Defaults to $\pi/8$.

`unit.angle` defines the unit for the specification of angles in the two previous arguments. Options are `"degrees"` and `"radians"`.

`messages` logical. Indicates whether status messages should be printed on the screen (or output device) while the function is running.

`...` arguments to be passed to the function `ksmooth`, if `option = "smooth"`.

Value

The output is an object of the class `variog4`, a list with five components. The first four elements are estimated variograms for the directions provided and the last is the omnidirectional variogram. Each individual component is an object of the class `variogram`, an output of the function `variog`.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

`variog` for variogram calculations and `plot.variog4` for plotting results

Examples

```
data(s100)
var4 <- variog4(s100, max.dist=1)
plot(var4)
```

---

wo *Kriging example data from Webster and Oliver*

Description

Data used in Chapter 8, page 156 of Webster and Oliver (2001) to illustrate properties of the kriging predictor.

Usage

```
data(wo)
```

Format

An object of the class `geodata` which is a list with the elements:

coords coordinates of the data location.

data the data vector.

x1 coordinate of the centrally located prediction point.

x2 coordinate of the off-centre prediction point.

Source

Webster, R. and Oliver, M.A. (2001). Geostatistics for Environmental Scientists. Wiley.

Examples

```
data(wo)
attach(wo)
par(mfrow=c(1,2))
plot(c(-10,130), c(-10,130), ty="n", asp=1)
points(rbind(coords, x1))
text(coords[,1], 5+coords[,2], format(data))
text(x1[1]+5, x1[2]+5, "?", col=2)
plot(c(-10,130), c(-10,130), ty="n", asp=1)
points(rbind(coords, x2))
text(coords[,1], 5+coords[,2], format(data))
text(x2[1]+5, x2[2]+5, "?", col=2)
```

`wolfcamp` *Wolfcamp Aquifer Data*

Description

Piezometric head measurements taken at the Wolfcamp Aquifer, Texas, USA. See Cressie (1993, p.212–214) for description of the scientific problem and the data. Original data were converted to SI units: coordinates are given in kilometers and pressure heads to meters.

Usage

```
data(wolfcamp)
```

Format

An object of the `class` `"geodata"`, which is list with two components:

`coords` the coordinates of the data locations. The distance are given in kilometers.

`data` values of the piezometric head. The unit is heads to meters.

Source

Harper, W.V and Furr, J.M. (1986) Geostatistical analysis of potentiometric data in the Wolfcamp Aquifer of the Palo Duro Basin, Texas. *Technical Report BMI/ONWI-587, Bettelle Memorial Institute, Columbus, OH*.

References

Cressie, N.A.C (1993) *Statistics for Spatial Data*. New York: Wiley.

Papritz, A. and Moyeed, R. (2001) Parameter uncertainty in spatial prediction: checking its importance by cross-validating the Wolfcamp and Rongelap data sets. *GeoENV 2000: Geostatistical for Environmental Applications. Ed. P. Monestiez, D. Allard, R. Froidevaux*. Kluwer.

Examples

```
data(wolfcamp)
summary(wolfcamp)
plot(wolfcamp)
```

`wrappers` *Wrappers for the C functions used in geoR*

Description

These functions are *wrappers* for some (but not all) the C functions included in the **geoR** package. Typically the C code is directly called from the **geoR** functions but these functions allows independent calls.

Usage

```
diffpairs(coords, data)
loccoords(coords, locations)
.diagquadraticformXAX(X, lowerA, diagA)
.bilinearformXAY(X, lowerA, diagA, Y)
.corr.diaglowertri(coords, cov.model, phi, kappa)
.Ccor.spatial(x, phi, kappa, cov.model)
```

Arguments

`coords`	an $n \times 2$ matrix with the data coordinates.
`data`	an vector with the data values.
`locations`	an $N \times 2$ matrix with the coordinates of the prediction locations.
`lowerA`	a vector with the diagonal terms of the symmetric matrix A.
`diagA`	a vector with the diagonal terms of the symmetric matrix A.
`X`	a matrix with conforming dimensions.
`Y`	a matrix with conforming dimensions.
`cov.model`	covariance model, see `cov.spatial` for options and more details.
`phi`	numerical value of the correlation function parameter phi.
`kappa`	numerical value of the correlation function parameter kappa.
`x`	a vector of distances.

Value

The outputs for the different functions are:

`diffpairs`	returns a list with elements `dist` - the distance between pairs of points, and `diff` - the difference between the values of the attributes.
`loccoords`	returns a $n \times N$ matrix with distances between data points and prediction locations.
`diagquadraticformXAX`	returns a vector with the diagonal term of the quadratic form $X'AX$.
`bilinearformXAY`	returns a vector or a matrix with the terms of the quadratic form $X'AY$.

`corr.diaglowertri`
: returns the lower triangle of the correlation matrix, including the diagonal.

`Ccor.spatial` returns a vector of values of spatial correlations.

Author(s)

Paulo Justiniano Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

---

xvalid *Cross-validation using kriging*

Description

This is a function to perform model validation Options include *leaving-one-out* cross-validation where. each data location is removed from the data set and the variable at this location is predicted using the remaining locations, for as given model. This can be done for all or some of the locations. Alternativelly, other validation locations which are not the same as the original data locations can be used.

Usage

```
xvalid(geodata, coords = geodata$coords, data = geodata$data,
       model, reestimate = FALSE, variog.obj = NULL,
       output.reestimate = FALSE, locations.xvalid = "all",
       data.xvalid = NULL, messages, ...)
```

Arguments

`geodata` a list containing element `coords` as described next. Typically an object of the class `"geodata"` - a **geoR** data-set. If not provided the arguments `coords` must be provided instead.

`coords` an $n \times 2$ matrix containing coordinates of the n data locations in each row. Defaults to `geodata$coords`, if provided.

`data` a vector or matrix with data values. If a matrix is provided, each column is regarded as one variable or realization. Defaults to `geodata$data`, if provided.

`model` an object containing information on a fitted model. Typically an output of `likfit`, `variofit`. If an object of the class `eyefit` is passed it takes the first model specified in the object.

`reestimate` logical. Indicates whether or not the model parameters should be re-estimated for each point removed from the data-set.

`variog.obj` on object with the empirical variogram, typically an output of the function `variog`. Only used if `reestimate = TRUE` and the object passed to the argument `model` is the result of a variogram based estimation, i.e. if the model was fitted by `variofit`.

`output.reestimate`
logical. Only valid if `reestimate = TRUE`. Specifies whether the re-estimated parameters are returned.

`locations.xvalid`
there are three possible specifications for this argument: `"all"` indicates the *leaving-on-out* method is used. The second possibility is to use only a sub-set of the data for cross-validation. For this case a vector should be provided with numbers indicating at which locations the cross-validation should be performed. The third option is to perform validation on a different set of data. For this a matrix with the coordinates of the validation points should be provided and the the argument `locations.data` should also be provided.

`data.xvalid` data values at the validation locations. Only used if the validation locations are not the same or a subset of the original data coordinates.

`messages` logical. Indicates whether status messages should be printed on the screen (or output device) while the function is running.

`...` further arguments to the minimization functions used by `likfit`, `variofit`.

Details

The cross-validation uses the function `krige.conv` to predict at each location.

For models fitted by `variofit` the parameters κ, ψ_A, ψ_R and λ are always regarded as fixed.

See documentation of the function `likfit` for more details on the model and its parameters.

Value

An object of the `class` `"xvalid"` which is a list with the following components:

`data` the original data.

`predicted` the values predicted by cross-validation.

`krige.var` the cross-validation prediction variance.

`error` difference `data - predicted`.

`std.error` the errors divided by the square root of the prediction variances.

`prob` the cumulative probability at original value under a normal distribution with parameters given by the cross-validation results.

A method for `summary` returns summary statistics for the errors and standard errors.

If `reestimate = TRUE` and `output = TRUE` additional columns are added to the data-frame. Each column will contain the values of the re-estimated parameters.

Author(s)

Paulo J. Ribeiro Jr. ⟨Paulo.Ribeiro@est.ufpr.br⟩,
Peter J. Diggle ⟨p.diggle@lancaster.ac.uk⟩.

References

Further information on the package **geoR** can be found at:
http://www.est.ufpr.br/geoR.

See Also

plot.xvalid for plotting of the results, likfit, variofit for parameter estimation and krige.conv for the kriging method used for predictions.

Examples

```
data(s100)
#
# Maximum likelihood estimation
#
s100.ml <- likfit(s100, ini = c(.5, .5), fix.nug = TRUE)
#
# Weighted least squares estimation
#
s100.var <- variog(s100, max.dist = 1)
s100.wls <- variofit(s100.var, ini = c(.5, .5), fix.nug = TRUE)
#
# Now, performing cross-validation without reestimating the model
#
s100.xv.ml <- xvalid(s100, model = s100.ml)
s100.xv.wls <- xvalid(s100, model = s100.wls)
##
## Plotting results
##
par.ori <- par(no.readonly = TRUE)
##
par(mfcol=c(5,2), mar=c(2.3,2.3,.5,.5), mgp=c(1.3, .6, 0))
plot(s100.xv.ml)
par(mfcol=c(5,2))
plot(s100.xv.wls)
##
par(par.ori)
#
```

Index