Package 'geospt'

June 21, 2025

Type Package

Title Geostatistical Analysis and Design of Optimal Spatial Sampling Networks

Version 1.0-6

Date 2025-06-21

Maintainer Ali Santacruz <amsantac@unal.edu.co>

Depends R (>= 3.5.0), gstat, genalg, MASS, sp, minqa

Imports fields, gsl, plyr, TeachingDemos, sgeostat, grDevices, stats, methods, graphics, utils

Description Estimation of the variogram through trimmed mean, radial basis functions (optimization, prediction and cross-validation), summary statistics from cross-validation, pocket plot, and design of optimal sampling networks through sequential and simultaneous points methods.

License GPL (>= 2)

BugReports https://github.com/amsantac/geospt/issues

URL https://github.com/amsantac/geospt

Encoding UTF-8

LazyLoad yes

NeedsCompilation no

Author Carlos Melo [aut] (ORCID: <https://orcid.org/0000-0002-5598-1913>), Ali Santacruz [aut, cre], Oscar Melo [aut] (ORCID: <https://orcid.org/0000-0002-0296-4511>)

Repository CRAN

Date/Publication 2025-06-21 05:00:02 UTC

Contents

`geospt-package` *Geostatistical Analysis and Design of Optimal Spatial Sampling Networks*

Description

A set of functions for: estimation of the variogram through trimmed mean, radial basis functions (optimization, prediction and cross-validation), summary statistics from cross-validation, pocket plot, and design of optimal sampling networks through sequential and simultaneous points methods

Details

Package:	geospt
Type:	Package
Version:	1.0-6
Date:	2025-06-21
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

Carlos Melo <cmelo@udistrital.edu.co>, Ali Santacruz, Oscar Melo <oomelom@unal.edu.co

Maintainer: Ali Santacruz <amsantac@unal.edu.co>

See Also

rbf, est.variograms, seqPtsOptNet, simPtsOptNet

---

ariari *Ariari Map.*

Description

Map Basin Map. Spatial reference system: UTM 18S

Usage

```
data(ariari)
```

Format

The format is: Formal class 'SpatialPolygonsDataFrame' [package "sp"]

Examples

```
data(ariari)
pts <- spsample(ariari, n=25000, type="regular")
plot(pts)
```

---

ariprec *Data from climatic stations of the Ariari River (Meta-Colombia Department)*

Description

Data from climatic stations of the Ariari River (Meta-Colombia Department) associated with the rainfall variable

Usage

```
data(ariprec)
```

Format

A data frame with 18 observations on the following 6 variables:

`Obs` a numeric vector; observation number

`Nombre` a character vector; station name

`x` a numeric vector; x-coordinate

`y` a numeric vector; y-coordinate

`ELEV` a numeric vector; Elevation above sea level

`PRECI_TOT` a numeric vector; the target variable

Examples

```
data(ariprec)
summary(ariprec)
```

`bestnet` *Generate a SpatialPoints object corresponding to the best result obtained in an optimized network*

Description

Generate a SpatialPoints object with the *x* and *y* coordinates corresponding to the best result obtained in an optimized network. The parameter to be passed to this function must be the result of `seqPtsOptNet` or `simPtsOptNet`

Usage

```
bestnet(optimnet)
```

Arguments

`optimnet`	object of class rbga resulting from `seqPtsOptNet` or `simPtsOptNet`

Value

a SpatialPoints object

See Also

See function `rbga` in the `genalg` package; for examples see `seqPtsOptNet` and `simPtsOptNet`

bp.with.outlier.label *geospt internal function*

Description

geospt internal function

Note

This function is not meant to be called by users directly

COSha10 *Soil organic carbon database at a sampling depth of 0-10 cm*

Description

Soil organic carbon database of samples taken in several soil and land cover types at La Libertad Research Center at a sampling depth of 0-10 cm

Usage

```
data(COSha10)
```

Format

A data frame with 122 observations on the following 10 variables:

`ID` ID of each sampling site

`x` *x*-coordinate of each site. Spatial reference system: UTM 18N

`y` *y*-coordinate of each site. Spatial reference system: UTM 18N

`DA10` measured soil bulk density (g cm^{-3})

`CO10` measured soil carbon concentration (%)

`COB1r` land cover at each sampling site in 2007. See details below

`S_UDS` soil type at each sampling site. See details below

`COSha10` calculated total soil carbon stock (t ha^{-1}). See details below

`Cor4DAidep` total soil carbon stock (t ha^{-1}) corrected by soil compaction factors

`CorT` corrected total soil carbon stock with Box-Cox transformation applied

Details

A total of 150 samples for a 0-10 cm depth was collected and analyzed for soil bulk density and organic carbon concentration in 2007 at La Libertad Research Center in Villavicencio, Colombia. The samples were taken in soils under different land cover types: rice crops (`Az`), citrus crops (`Ci`), forest plantations (`Cpf`), annual crops (`Ctv`), grasses (`P`), and oil palm crops (`Pl`). In the soil type names, the first two letters correspond to the short name of the soil series, the lower-case letters indicate the slope class, and the number denotes the type of soil drainage.

Total soil carbon stock $COSha$ was calculated as follows (Guo & Gifford, 2002):

$$COSha = DA * CO * d$$

where DA is soil bulk density (g cm^{-3}), CO is soil organic carbon concentration (%) and d is sampling depth (cm).

Given that the data did not fit a normal distribution, a Box-Cox transformation was applied (Box & Cox, 1964). Some samples were discarded for the design of sampling networks. The complete database and description can be found in Santacruz (2010) and in Santacruz et al., (2014).

Source

Santacruz, A. 2010. *Design of optimal spatial sampling networks for the monitoring of soil organic carbon at La Libertad Research Center through the application of genetic algorithms*. M.Sc. Thesis. National University of Colombia, Bogota. 162 p. (In Spanish)

References

Santacruz, A., Rubiano, Y., Melo, C., 2014. *Evolutionary optimization of spatial sampling networks designed for the monitoring of soil carbon*. In: Hartemink, A., McSweeney, K. (Eds.). *Soil Carbon. Series: Progress in Soil Science*. (pp. 77-84). Springer. [link]

Santacruz, A., 2011. *Evolutionary optimization of spatial sampling networks. An application of genetic algorithms and geostatistics for the monitoring of soil organic carbon*. Editorial Academica Espanola. 183 p. ISBN: 978-3-8454-9815-7 (In Spanish)

Guo, L., Gifford, R., 2002. *Soil carbon stocks and land use change: a meta analysis*. Global Change Biology 8, 345-360.

Box, G., Cox, D., 1964. *An analysis of transformations*. Journal of the Royal Statistical Society. Series B (Methodological) 26 (2), 211-252.

See Also

`COSha10map`

Examples

```
data(COSha10)
str(COSha10)
```

COSha10map *Map of total soil carbon stock (t/ha) at 0-10 cm depth*

Description

Map of total soil carbon stock (t ha^{-1}) at 0-10 cm depth at La Libertad Research Center. The map was obtained through ordinary kriging interpolation. Spatial reference system: UTM 18N

Usage

```
data(COSha10map)
```

Format

The format is: Formal class 'SpatialPixelsDataFrame' [package "sp"]

Source

Santacruz, A., 2010. *Design of optimal spatial sampling networks for the monitoring of soil organic carbon at La Libertad Research Center through the application of genetic algorithms*. M.Sc. Thesis. National University of Colombia, Bogota. 162 p. (In Spanish)

References

Santacruz, A., Rubiano, Y., Melo, C., 2014. *Evolutionary optimization of spatial sampling networks designed for the monitoring of soil carbon*. In: Hartemink, A., McSweeney, K. (Eds.). *Soil Carbon. Series: Progress in Soil Science*. (pp. 77-84). Springer. [link]

Santacruz, A., 2011. *Evolutionary optimization of spatial sampling networks. An application of genetic algorithms and geostatistics for the monitoring of soil organic carbon*. Editorial Academica Espanola. 183 p. ISBN: 978-3-8454-9815-7 (In Spanish)

See Also

COSha10

Examples

```
data(COSha10map)
data(lalib)
summary(COSha10map)
l1 = list("sp.polygons", lalib)
spplot(COSha10map, "var1.pred", main="Soil carbon stock (t/ha) at 0-10 cm depth",
    col.regions=bpy.colors(100), scales = list(draw =TRUE), xlab ="East (m)",
    ylab = "North (m)", sp.layout=list(l1))
```

COSha30 *Soil organic carbon database at a sampling depth of 0-30 cm*

Description

Soil organic carbon database of samples taken in several soil and land cover types at La Libertad Research Center at a sampling depth of 0-30 cm

Usage

```
data(COSha30)
```

Format

A data frame with 118 observations on the following 10 variables:

`ID` ID of each sampling site

`x` x-coordinate of each site. Spatial reference system: UTM 18N

`y` y-coordinate of each site. Spatial reference system: UTM 18N

`DA30` measured soil bulk density (g cm^{-3})

`CO30` measured soil carbon concentration (%)

`COB1r` land cover at each sampling site in 2007. See details below

`S_UDS` soil type at each sampling site. See details below

`COSha30` calculated total soil carbon stock (t ha^{-1}). See details below

`Cor4DAidep` total soil carbon stock (t ha^{-1}) corrected by soil compaction factors

`CorT` corrected total soil carbon stock with Box-Cox transformation applied

Details

A total of 150 samples for a 0-30 cm depth was collected and analyzed for soil bulk density and organic carbon concentration in 2007 at La Libertad Research Center in Villavicencio, Colombia. The samples were taken in soils under different land cover types: rice crops (`Az`), citrus crops (`Ci`), forest plantations (`Cpf`), annual crops (`Ctv`), grasses (`P`), and oil palm crops (`Pl`). In the soil type names, the first two letters correspond to the short name of the soil series, the lower-case letters indicate the slope class, and the number denotes the type of soil drainage.

Total soil carbon stock $COSha$ was calculated as follows (Guo & Gifford, 2002):

$$COSha = DA * CO * d$$

where DA is soil bulk density (g cm^{-3}), CO is soil organic carbon concentration (%) and d is sampling depth (cm).

Given that the data did not fit a normal distribution, a Box-Cox transformation was applied (Box & Cox, 1964). Some samples were discarded for the design of sampling networks. The complete database and description can be found in Santacruz (2010) and in Santacruz et al., (2014).

Source

Santacruz, A. 2010. *Design of optimal spatial sampling networks for the monitoring of soil organic carbon at La Libertad Research Center through the application of genetic algorithms*. M.Sc. Thesis. National University of Colombia, Bogota. 162 p. (In Spanish)

References

Santacruz, A., Rubiano, Y., Melo, C., 2014. *Evolutionary optimization of spatial sampling networks designed for the monitoring of soil carbon*. In: Hartemink, A., McSweeney, K. (Eds.). *Soil Carbon. Series: Progress in Soil Science*. (pp. 77-84). Springer. [link]

Santacruz, A., 2011. *Evolutionary optimization of spatial sampling networks. An application of genetic algorithms and geostatistics for the monitoring of soil organic carbon*. Editorial Academica Espanola. 183 p. ISBN: 978-3-8454-9815-7 (In Spanish)

Guo, L., Gifford, R., 2002. *Soil carbon stocks and land use change: a meta analysis*. Global Change Biology 8, 345-360.

Box, G., Cox, D., 1964. *An analysis of transformations*. Journal of the Royal Statistical Society. Series B (Methodological) 26 (2), 211-252.

See Also

COSha30map

Examples

```
data(COSha30)
str(COSha30)
```

COSha30map — *Map of total soil carbon stock (t/ha) at 0-30 cm depth*

Description

Map of total soil carbon stock (t ha^{-1}) at 0-30 cm depth at La Libertad Research Center. The map was obtained through ordinary kriging interpolation. Spatial reference system: UTM 18N

Usage

```
data(COSha30map)
```

Format

The format is: Formal class 'SpatialPixelsDataFrame' [package "sp"]

Source

Santacruz, A., 2010. *Design of optimal spatial sampling networks for the monitoring of soil organic carbon at La Libertad Research Center through the application of genetic algorithms*. M.Sc. Thesis. National University of Colombia, Bogota. 162 p. (In Spanish)

References

Santacruz, A., Rubiano, Y., Melo, C., 2014. *Evolutionary optimization of spatial sampling networks designed for the monitoring of soil carbon*. In: Hartemink, A., McSweeney, K. (Eds.). *Soil Carbon. Series: Progress in Soil Science*. (pp. 77-84). Springer. [link]

Santacruz, A., 2011. *Evolutionary optimization of spatial sampling networks. An application of genetic algorithms and geostatistics for the monitoring of soil organic carbon*. Editorial Academica Espanola. 183 p. ISBN: 978-3-8454-9815-7 (In Spanish)

See Also

COSha30

Examples

```
data(COSha30map)
data(lalib)
summary(COSha30map)
l1 = list("sp.polygons", lalib)
spplot(COSha30map, "var1.pred", main="Soil carbon stock (t/ha) at 0-30 cm depth",
    col.regions=bpy.colors(100), scales = list(draw =TRUE), xlab ="East (m)",
    ylab = "North (m)", sp.layout=list(l1))
```

`criteria.cv` *Cross-validation summaries*

Description

Generate a data frame of statistical values associated with cross-validation

Usage

```
criteria.cv(m.cv)
```

Arguments

`m.cv`	data frame containing: the coordinates of data, prediction columns, prediction variance of cross-validation data points, observed values, residuals, zscore (residual divided by kriging standard error), and fold. If the `rbf.tcv` function is used, the prediction variance and zscore (residual divided by standard error) will have NA's

Value

data frame containing: mean prediction errors (MPE), average kriging standard error (AKSE), root-mean-square prediction errors (RMSPE), mean standardized prediction errors (MSPE), root-mean-square standardized prediction errors (RMSSPE), mean absolute percentage prediction errors (MAPPE), coefficient of correlation of the prediction errors (CCPE), coefficient of determination (R2) and squared coefficient of correlation of the prediction errors (pseudoR2)

Examples

```
library(gstat)
data(meuse)
coordinates(meuse) <- ~x+y
m <- vgm(.59, "Sph", 874, .04)

# leave-one-out cross validation:
out <- krige.cv(log(zinc)~1, meuse, m, nmax = 40)
criteria.cv(out)

# multiquadratic function
data(preci)
coordinates(preci) <- ~x+y

# predefined eta
tab <- rbf.tcv(prec~x+y,preci,eta=1.488733, rho=0, n.neigh=9, func="M")
criteria.cv(tab)
```

`est.variograms` *Variogram Estimator*

Description

Calculate empirical `variogram` estimates. An object of class variogram contains empirical variogram estimates which are generated from a point object and a pair object. A variogram object is stored as a data frame containing seven columns: `lags`, `bins`, `classic`, `robust`,`med`, `trim` and `n`. The length of each vector is equal to the number of lags in the pair object used to create the variogram object, say l. The `lags` vector contains the lag numbers for each lag, beginning with one (1) and going to the number of lags (l). The `bins` vector contains the spatial midpoint of each lag. The `classic`, `robust`, `med` and `trimmed.mean` vectors contain: the classical, robust, median, and trimmed mean, respectively, which are given, respectively, by (see Cressie, 1993, p. 75)

classical

$$\gamma_c(h) = \frac{1}{n} \sum_{(i,j) \in N(h)} (z(x_i) - z(x_j))^2$$

robust,

$$\gamma_m(h) = \frac{(\frac{1}{n} \sum_{(i,j) \in N(h)} (\sqrt{|z(x_i) - z(x_j)|}))^4}{0.457 + \frac{0.494}{n}}$$

median

$$\gamma_{me}(h) = \frac{(median_{(i,j) \in N(h)} (\sqrt{|z(x_i) - z(x_j)|}))^4}{0.457 + \frac{0.494}{|N(h)|}}$$

and trimmed mean

$$\gamma_{tm}(h) = \frac{(trimmed.mean_{(i,j) \in N(h)} (\sqrt{|z(x_i) - z(x_j)|}))^4}{0.457 + \frac{0.494}{|N(h)|}}$$

The n vector contains the number $|N(h)|$ of pairs of points in each lag $N(h)$.

Usage

```
est.variograms(point.obj, pair.obj, a1, a2, trim)
```

Arguments

point.obj	a point object generated by point()
pair.obj	a pair object generated by pair()
a1	a variable to calculate semivariogram for
a2	an optional variable name, if entered cross variograms will be created between a1 and a2
trim	percent of trimmed mean

Value

A variogram object:

lags	vector of lag identifiers
bins	vector of midpoints of each lag
classic	vector of classic variogram estimates for each lag
robust	vector of robust variogram estimates for each lag
med	vector of median variogram estimates for each lag
trimmed.mean	vector of trimmed mean variogram estimates for each lag
n	vector of the number of pairs in each lag

Note

Based on the est.variogram function of the sgeostat package

References

Bardossy, A., 2001. *Introduction to Geostatistics*. University of Stuttgart.

Cressie, N.A.C., 1993. *Statistics for Spatial Data*. Wiley.

Majure, J., Gebhardt, A., 2009. sgeostat: An Object-oriented Framework for Geostatistical Modeling in S+. R package version 1.0-23.

Roustant O., Dupuy, D., Helbert, C., 2007. *Robust Estimation of the Variogram in Computer Experiments*. Ecole des Mines, Departement 3MI, 158 Cours Fauriel, 42023 Saint-Etienne, France

See Also

point, pair

Examples

```
library(sgeostat, pos=which(search()=="package:gstat")+1)
data(maas)
maas.point <- point(maas)
maas.pair <- pair(maas.point, num.lags=24, maxdist=2000)
maas.v <- est.variograms(maas.point,maas.pair,'zinc',trim=0.1)
maas.v
```

`extractFormula` *geospt internal function*

Description

geospt internal function

Note

This function is not meant to be called by users directly

`graph.idw` *Graph that describes the behavior of the optimized* p *smoothing parameter.*

Description

Function for plotting the RMSPE for several values of the p smoothing parameter with the same dataset. A curve is fitted to the points, and then the optimal p that provides the smallest RMSPE is determined from the curve, by the `optimize` function from the `stats` package.

Usage

```
graph.idw(formula, data, locations, np, p.dmax, P.T=NULL, nmax=Inf, nmin=0, pleg,
    progress=F, iter, ...)
```

Arguments

`formula`	formula that defines the dependent variable as a linear model of independent variables; suppose the dependent variable has name z, for a *idw* detrended use $z\sim1$
`data`	SpatialPointsDataFrame: should contain the dependent variable, independent variables, and coordinates.
`locations`	object of class *Spatial*, or (deprecated) formula defines the spatial data locations (coordinates) such as $\sim x+y$
`np`	number of points, where the idw is calculated

p.dmax	maximum value of the range of the *p* parameter that will be evaluated by the optimize function
P.T	logical. Print Table (TRUE) or not (FALSE). Default P.T=NULL.
nmax	maximum number of nearest observations that should be used for a *idw* prediction, where nearest is defined in terms of the spatial locations. By default, all observations are used
nmin	minimum number of nearest observations that should be used for a *idw* prediction, where nearest is defined in terms of the spatial locations. see krige
pleg	the x and y co-ordinates to be used to position the legend. They can be specified by keyword or in any way which is accepted by xy.coords, by default pleg="topright.
progress	logical. Use TRUE to see the percentage of progress of the process and FALSE otherwise). Default progress=FALSE.
iter	The maximum allowed number of function evaluations.
...	further parameters to be passed to the minimization functions optimize or bobyqa, typically arguments of the type control() which control the behavior of the minimization algorithm. See documentation about the selected minimization function for further details.

Value

Returns a graph that describes the behavior of the optimized *p* parameter associated with the RMSPE, and a table of values associated with the graph including optimal smoothing *p* parameter, which generates the lowest RMSPE.

References

Johnston, K., Ver, J., Krivoruchko, K., Lucas, N. 2001. *Using ArcGIS Geostatistical Analysis*. ESRI.

Examples

```
## Not run:
data(ariari)
data(ariprec)
# p optimization
gp <- graph.idw(PRECI_TOT~ 1, ~x+y, data=ariprec, np=50, p.dmax=4, nmax=15,
    nmin=15,pleg = "center", progress=T)
gp
gp$p

library(sp)
library(fields)
plot(ariari)
gridAri <- spsample(ariari,20000,"regular")
plot(gridAri)

idw.p <- idw(PRECI_TOT~ 1, ~ x+y, ariprec, gridAri, nmax=15, nmin=15, idp=2)
```

```
pal2 <- colorRampPalette(c("snow3","royalblue1", "blue4"))

# Inverse Distance Interpolations Precipitation Weighted (P = 2)
p1 <- spplot(idw.p[1], col.regions=pal2(100), cuts =60, scales = list(draw =T),
    xlab ="East (m)", ylab = "North (m)",
       main = "", auto.key = F)

split.screen( rbind(c(0, 1,0,1), c(1,1,0,1)))
split.screen(c(1,2), screen=1)-> ind
screen( ind[1])
p1
screen( ind[2])
image.plot(legend.only=TRUE, legend.width=0.5, col=pal2(100),
    smallplot=c(0.6,0.68, 0.5,0.75),
    zlim=c(min(idw.p$var1.pred),max(idw.p$var1.pred)),
    axis.args = list(cex.axis = 0.7))
close.screen( all=TRUE)


## End(Not run)
```

`graph.rbf` *Graph that describes the behavior of the optimized* eta *and* rho *parameters, associated with a radial basis function*

Description

Function for plotting the RMSPE for several values of the smoothing parameter eta with the same dataset. A curve is fitted to the points, and then the optimal eta that provides the smallest RMSPE is determined from the curve, by the `optimize` function from the `stats` package.

Usage

```
graph.rbf(formula, data, eta.opt, rho.opt, n.neigh, func, np, x0, eta.dmax,
rho.dmax, P.T, iter, ...)
```

Arguments

`formula`	formula that defines the dependent variable as a linear model of independent variables; suppose the dependent variable has name *z*, for a *rbf* detrended use *z~1*; for a *rbf* with trend, suppose *z* is linearly dependent on *x* and *y*, use the formula *z~x+y* (linear trend).
`data`	SpatialPointsDataFrame: should contain the dependent variable, independent variables, and coordinates.
`eta.opt`	logical, indicating whether the parameter *eta* should be regarded as fixed (eta.opt = FALSE) or should be estimated (eta.opt = TRUE)
`rho.opt`	logical, indicating whether the parameter *rho* should be regarded as fixed (rho.opt = FALSE) or should be estimated (rho.opt = TRUE)

n.neigh	number of nearest observations that should be used for a *rbf* prediction, where nearest is defined in terms of the spatial locations
func	function to be optimized. The following radial basis function model types are currently available: gaussian "GAU", exponential "EXPON", trigonometric "TRI", thin plate spline "TPS", completely regularized spline "CRS", spline with tension "ST", inverse multiquadratic "IM", and multiquadratic "M", are currently available
np	number of points, where the radial basis function is calculated
x0	starting point for searching the optimum. Defaults to c(0.5, 0.5), *eta* and *rho* respectively. Use this statement only if eta and rho are equal to TRUE.
eta.dmax	maximum value of the range of the *eta* parameter that will be evaluated by the optimize function
rho.dmax	maximum value of the range of the *rho* parameter that will be evaluated by the optimize function
P.T	logical. Print Table (TRUE) or not (FALSE). Default P.T=NULL.
iter	The maximum allowed number of function evaluations.
...	further parameters to be passed to the minimization functions optimize or bobyqa, typically arguments of the type control() which control the behavior of the minimization algorithm. See documentation about the selected minimization function for further details.

Value

Returns a graph that describes the behavior of the optimized *eta* or *rho* parameter, and a table of values associated with the graph including optimal smoothing *eta* or *rho* parameters. If both *eta* and *rho* are FALSE simultaneously, then the function returns a list with; the best value obtained from the combinations smoothing *eta* and *rho* parameters and a lattice plot of class "trellis" with RMSPE pixel values associated with combinations of *eta* and *rho* parameters. Finally if both *eta* and *rho* are TRUE, the function will return a list with the best combination of values of the smoothing *eta* or *rho* parameters and the RMSPE associated with these.

References

Johnston, K., Ver, J., Krivoruchko, K., Lucas, N. 2001. *Using ArcGIS Geostatistical Analysis*. ESRI.

Examples

```
data(preci)
coordinates(preci)<-~x+y
# optimizing eta
graph.rbf(prec~1, preci, eta.opt=TRUE, rho.opt=FALSE, n.neigh=9, func="TPS",
    np=40, eta.dmax=0.2, P.T=TRUE)
## Not run:
# optimizing rho
graph.rbf(prec~x+y, preci, eta.opt=FALSE, rho.opt=TRUE, n.neigh=9, func="M",
    np=20, rho.dmax=2, P.T=TRUE)
# optimizing eta and rho
```

```
tps.lo <- graph.rbf(prec~1, preci, eta.opt=TRUE, rho.opt=TRUE, n.neigh=9, func="TPS",
    eta.dmax=2, rho.dmax=2, x0=c(0.1,0.1), iter=40)
tps.lo$Opt  # best combination of eta and rho obtained
# other optimization options
opt.u <- uobyqa(c(0.1,0.1), rbf.cv1, control = list(maxfun=40), formula=prec~1, data=preci,
                n.neigh=9, func="TPS")
opt.n <- newuoa(c(0.1,0.1), rbf.cv1, control = list(maxfun=40), formula=prec~1, data=preci,
                n.neigh=9, func="TPS")
# lattice of RMSPE values associated with a range of eta and rho, without optimization
tps.l <- graph.rbf(prec~1, preci, eta.opt=FALSE, rho.opt=FALSE, n.neigh=9, func="TPS",
    np=10, eta.dmax=2, rho.dmax=2)
tps.l$opt.table  # best combination of eta and rho obtained from lattice
tps.l$spplot     # lattice of RMSPE

## End(Not run)
```

`idw.cv` *idw cross validation leave-one-out*

Description

Generate the RMSPE value, which is given by the idw function with p smoothing parameter.

Usage

```
idw.cv(formula, locations, data, nmax = Inf, nmin = 0, p = 2, progress=FALSE, ...)
```

Arguments

`formula`	formula that defines the dependent variable as a linear model of independent variables; suppose the dependent variable has name *z*, for a *idw* detrended use *z~1*
`data`	SpatialPointsDataFrame: should contain the dependent variable, independent variables, and coordinates.
`locations`	object of class *Spatial*, or (deprecated) formula defines the spatial data locations (coordinates) such as ~*x*+*y*
`nmax`	number of nearest observations that should be used for a *idw* prediction, where nearest is defined in terms of the spatial locations. By default, all observations are used.
`nmin`	if the number of nearest observations within distance maxdist is less than nmin, a missing value will be generated; see maxdist.
`p`	value of smoothing parameter; we recommend using the parameter found by minimizing the root-mean-square prediction errors using cross-validation. Default is 2.
`progress`	logical. Use TRUE to see the percentage of progress of the process and FALSE otherwise). Default progress=FALSE.
`...`	Other arguments passed to *idw*

Value

returns the RMSPE value

See Also

gstat::idw()

Examples

```
data(preci)
idw.cv(prec~1, ~x+y, preci, nmax=9, nmin=9, p=2, progress=TRUE)
```

---

lalib *Map of boundary enclosing La Libertad Research Center*

Description

Map of boundary enclosing La Libertad Research Center

Usage

```
data(lalib)
```

Format

The format is: Formal class 'SpatialPolygonsDataFrame' [package "sp"]

Details

Map of boundary enclosing La Libertad Research Center. Spatial reference system: UTM 18N

Source

Santacruz, A. 2010. *Design of optimal spatial sampling networks for the monitoring of soil organic carbon at La Libertad Research Center through the application of genetic algorithms*. M.Sc. Thesis. National University of Colombia, Bogota. 162 p. (In Spanish)

References

Santacruz, A., Rubiano, Y., Melo, C., 2014. *Evolutionary optimization of spatial sampling networks designed for the monitoring of soil carbon*. In: Hartemink, A., McSweeney, K. (Eds.). *Soil Carbon. Series: Progress in Soil Science*. (pp. 77-84). Springer. [link]

Santacruz, A., 2011. *Evolutionary optimization of spatial sampling networks. An application of genetic algorithms and geostatistics for the monitoring of soil organic carbon*. Editorial Academica Espanola. 183 p. ISBN: 978-3-8454-9815-7 (In Spanish)

Examples

```
data(lalib)
summary(lalib)
plot(lalib)
```

`network.design` *Generating* `AKSE` *associated with a conditioned network design*

Description

Generates a sampling network for a given sampling distance or type (configuration), and calculates the average kriging standard error (`AKSE`) associated with the spatial configuration for a given predefined variogram

Usage

```
network.design(formula, vgm.model, xmin, xmax, ymin, ymax, npoint.x, npoint.y,
npoints, boundary=NULL, type, ...)
```

Arguments

`formula`	formula that defines the dependent variable as a linear model of independent variables; suppose the dependent variable has name z, for ordinary and simple kriging use the formula `z~1`; for simple kriging also define beta (see below); for universal kriging, suppose `z` is linearly dependent on `x` and `y`, use the formula `z~x+y`
`vgm.model`	variogram model of dependent variable (or its residuals), defined by a call to vgm or fit.variogram
`npoint.x`	number of points to generate on the `x`-axis
`npoint.y`	number of points to generate on the `y`-axis
`npoints`	(approximate) sample size inside (shapefile) border
`xmin`	minimum `x`-coordinate of the study area.
`ymin`	minimum `y`-coordinate of the study area.
`xmax`	maximum `x`-coordinate of the study area.
`ymax`	maximum `y`-coordinate of the study area.
`boundary`	SpatialPolygons or SpatialPolygonsDataFrame object
`type`	character; "random" for completely spatial random; "regular" for regular (systematically aligned) sampling; "stratified" for stratified random (one single random location in each "cell"); "nonaligned" for nonaligned systematic sampling (nx random y coordinates, ny random x coordinates); "hexagonal" for sampling on a hexagonal lattice; "clustered" for clustered sampling; "Fibonacci" for Fibonacci sampling on the sphere (see references). By default type ="regular".
`...`	further arguments will be passed of the `krige` and `spsample` functions.

Value

returns the AKSE value associated with the spatial distribution of points and the kriging method used.

References

Fibonacci sampling: Alvaro Gonzalez, 2010. *Measurement of Areas on a Sphere Using Fibonacci and Latitude-Longitude Lattices*. Mathematical Geosciences 42(1), p. 49-64

See Also

krige, krige.cv, spsample, point.in.polygon, sample

Examples

```
## Not run:
### regular grid 10x10
vgmok <- vgm(112.33, "Sph", 4.3441,0)
vgmsk <- vgm(74.703, "Sph", 3.573,0)
vgmuk <- vgm(53.064, "Sph", 2.8858,0)
vgmuk2 <- vgm(19.201, "Sph", 1.5823,0)
# network: ordinary kriging (without boundary)
net1.ok <- network.design(z~1,vgmok, xmin=0,xmax=10, ymin=0, ymax=10, npoint.x=10,
                          npoint.y=10, nmax=6)
net2.ok <- network.design(z~1,vgmok, xmin=0,xmax=10, ymin=0, ymax=10, npoint.x=20,
                          npoint.y=20, nmax=6)
# it's worth noting that the variograms are different in each kriging
# network: simple kriging (without boundary)
net1.sk <- network.design(z~1,vgmsk, xmin=0,xmax=10, ymin=0, ymax=10, npoint.x=10,
                          npoint.y=10, nmax=6, beta=2)
net2.sk <- network.design(z~1,vgmsk, xmin=0,xmax=10, ymin=0, ymax=10, npoint.x=20,
                          npoint.y=20, nmax=6, beta=2)
# network: universal kriging, first order trend (without boundary)
net1.uk <- network.design(z~x + y,vgmuk, xmin=0,xmax=10, ymin=0, ymax=10,
                          npoint.x=10, npoint.y=10, nmax=8)
# network: universal kriging, second order trend (without boundary)
net2.uk <- network.design(z~x + y + x*y + I(x^2)+I(y^2),vgmuk2, xmin=0,xmax=10, ymin=0,
                          ymax=10, npoint.x=20, npoint.y=20, nmax=8)
# Creating the grid with the prediction and plotting points
library(geoR)
data(ca20)
Sr1 <- Polygon(ca20$borders)
Srs1 = Polygons(list(Sr1), "s1")
Polygon = SpatialPolygons(list(Srs1))
vgmok.ca <- vgm(112.33, "Sph", 244.9,0)
vgmsk.ca <- vgm(100, "Sph", 150.2,0)
vgmuk.ca <- vgm(85.57, "Sph", 110.5,0)
vgmuk2.ca <- vgm(62.14, "Sph", 89.7,0)
# network: ordinary kriging (with boundary)
netb1.ok<- network.design(z~1, vgmok.ca, npoints=50, boundary=Polygon, nmax=6)
netb2.ok<- network.design(z~1, vgmok.ca, npoints=100, boundary=Polygon, nmax=6)
# network: simple kriging (with boundary)
netb1.sk <- network.design(z~1, vgmsk.ca, npoints=50, boundary=Polygon, nmax=6, beta=2)
```

```
netb2.sk <- network.design(z~1, vgmsk.ca, npoints=100, boundary=Polygon, nmax=6, beta=2)
# network: universal kriging, first order trend (with boundary)
netb1.uk <- network.design(z~x + y, vgmuk.ca, npoints=50, boundary=Polygon, nmax=8)
# network: universal kriging, second order trend (with boundary)
netb2.uk <- network.design(z~x + y + x*y + I(x^2)+I(y^2), vgmuk2.ca, npoints=100,
                           boundary=Polygon, nmax=8)

## End(Not run)
```

`pocket.plot` *graphs the probability or standardized variance in the directions north-south or east-west*

Description

The pocket-plot (so named because of its use in detecting pockets of non-stationarity) is a technique necessary to identify a localized area that is atypical with respect to the stationarity model. It is built to exploit the spatial nature of the data through the coordinates of rows and columns (east "X" and north "Y", respectively).

Usage

```
pocket.plot(data, graph, X, Y, Z, Iden=F, ...)
```

Arguments

`data`	data frame should contain the dependent variable and coordinates X and Y, data must be gridded
`graph`	type of graph associated with the probability or standardized variance plot pocket in the directions north-south or east-west; Probabilities PocketPlot by rows, ie horizontal "south-north" (PPR), Probabilities PocketPlot by columns, ie vertical "east-west" (PPC), PocketPlot of variance by rows, ie horizontal "south-north" (PVR) and PocketPlot of variance by columns, ie vertical "east-west" (PVC)
`X`	defined by the spatial coordinates
`Y`	defined by the spatial coordinates
`Z`	regionalized variable with which you construct the statistics associated with the probability or standardized variance, these are plotted in the so-called pocket plot
`Iden`	logical. The users can identify the points by themselves, TRUE or FALSE. FALSE by default is used.
`...`	arguments to be passed to ...

Details

For identifying outliers, this function uses a modification of the boxplot.with.outlier.label function, available at https://www.r-statistics.com/2011/01/how-to-label-all-the-outliers-in-a-boxplot/

Value

returns (or plots) the pocket plot

References

Cressie, N.A.C. 1993. *Statistics for Spatial Data*. Wiley.

Gomez, M., Hazen, K. 1970. *Evaluating sulfur and ash distribution in coal seems by statistical response surface regression analysis*. U.S. Bureau of Mines Report RI 7377.

Examples

```
# Core measurements (in % coal ash) at reoriented locations.
# Units on the vertical axis are % coal ash.

# These data was found in mining samples originally reported by
# Gomez and Hazen (1970), and later used by Cressie (1993).

# These data are available in the sp and gstat packages

library(gstat)
data(coalash)
plot(coalash[,1:2], type="n", xlab="x", ylab="y")
text(coalash$x,coalash$y,coalash$coalash,cex=0.6)

# Pocket plot in the north-south direction.
# Units on the vertical axis are root (% coal ash)

# Plot generated with the function pocket.plot
# Clearly rows 2, 6, and 8 are atypical

# This serves as verification that these rows are potentially problematic

# Analysis of local stationarity in probabilities of the coal in south-north direction

pocket.plot(coalash, "PPR", coalash$x, coalash$y, coalash$coalash, FALSE)

# Analysis of local stationarity in variance of the coal in south-north direction

pocket.plot(coalash, "PVR", coalash$x, coalash$y, coalash$coalash, FALSE)

# Analysis of local stationarity in probabilities of the coal in east-west direction

pocket.plot(coalash, "PPC", coalash$x, coalash$y, coalash$coalash, FALSE)

# Analysis of local stationarity in variance of the coal in east-west direction

pocket.plot(coalash, "PVC", coalash$x, coalash$y, coalash$coalash, FALSE)
```

preci *Empirical data related to rainfall*

Description

Empirically generated data in 10 arbitrary locations associated with the rainfall variable

Usage

```
data(preci)
```

Format

A data frame with 10 observations on the following 4 variables:

`Obs` a numeric vector; observation number

`x` a numeric vector; x-coordinate; unknown reference

`y` a numeric vector; y-coordinate; unknown reference

`prec` a numeric vector; the target variable

Examples

```
data(preci)
summary(preci)
```

rbf *gaussian, exponential, trigonometric, thin plate spline, inverse multiquadratic, and multiquadratic radial basis function prediction*

Description

Function for gaussian (GAU), exponential (EXPON), trigonometric (TRI), thin plate spline (TPS), completely regularized spline (CRS), spline with tension (ST), inverse multiquadratic (IM), and multiquadratic (M) radial basis function (*rbf*), where *rbf* is in a local neighbourhood

Usage

```
rbf(formula, data, eta, rho, newdata, n.neigh, func)
```

Arguments

formula	formula that defines the dependent variable as a linear model of independent variables; suppose the dependent variable has name z, for a *rbf* detrended use z~1, for a *rbf* with trend, suppose z is linearly dependent on x and y, use the formula z~x+y (linear trend).
data	SpatialPointsDataFrame: should contain the dependent variable, independent variables, and coordinates.
eta	the optimal smoothing parameter, we recommend using the parameter found by minimizing the root-mean-square prediction errors using cross-validation
rho	the optimal parameter robustness, we recommend using the parameter found by minimizing the root-mean-square prediction errors using cross-validation. eta and rho parameters can be optimized simultaneously, through the bobyqa function from nloptr or minqa packages
newdata	data frame or spatial object with prediction/simulation locations; should contain attribute columns with the independent variables (if present) and (if locations is a formula) the coordinates with names, as defined in locations where you want to generate new predictions
n.neigh	number of nearest observations that should be used for a *rbf* prediction, where nearest is defined in terms of the spatial locations
func	radial basis function model type, e.g. "GAU", "EXPON", "TRI", "TPS", "CRS", "ST", "IM" and "M", are currently available

Details

rbf function generates individual predictions from gaussian (GAU), exponential (EXPON), trigonometric (TRI) thin plate spline (TPS), completely regularized spline (CRS), spline with tension (ST), inverse multiquadratic (IM), and multiquadratic (M) functions

Value

Attributes columns contain coordinates, predictions, and the variance column contains NA's

Examples

```
data(preci)
coordinates(preci) <- ~x+y

# prediction case: one point
point <- data.frame(3,4)
names(point) <- c("x","y")
coordinates(point) <- ~x+y
rbf(prec~x+y, preci, eta=0.1460814, rho=0, newdata=point,n.neigh=10, func="TPS")

# prediction case: a grid of points
puntos<-expand.grid(x=seq(min(preci$x),max(preci$x),0.05), y=seq(min(preci$y),
max(preci$y),0.05))
coordinates(puntos) <- ~x+y
pred.rbf <- rbf(prec~x+y, preci, eta=0.1460814, rho=0, newdata=puntos, n.neigh=10, func="TPS")
```

```
coordinates(pred.rbf) = c("x", "y")
gridded(pred.rbf) <- TRUE

# show prediction map
spplot(pred.rbf["var1.pred"], cuts=40, col.regions=bpy.colors(100),
main = "rainfall map TPS", key.space=list(space="right", cex=0.8))
```

---

rbf.cv *rbf cross validation leave-one-out*

Description

Generate the RMSPE value, which is given by the radial basis function with smoothing parameter eta and robustness parameter rho.

Usage

```
rbf.cv(formula, data, eta, rho, n.neigh, func)
```

Arguments

formula	formula that defines the dependent variable as a linear model of independent variables; suppose the dependent variable has name z, for a *rbf* detrended use z~1, for a *rbf* with trend, suppose z is linearly dependent on x and y, use the formula z~x+y (linear trend).
data	SpatialPointsDataFrame: should contain the dependent variable, independent variables, and coordinates.
eta	the optimal smoothing parameter; we recommend using the parameter found by minimizing the root-mean-square prediction errors using cross-validation
rho	value of optimal robustness parameter; we recommend using the parameter found by minimizing the root-mean-square prediction errors using cross-validation. *eta* and *rho* parameters can be optimized simultaneously, through the bobyqa function from nloptr or minqa packages
n.neigh	number of nearest observations that should be used for a *rbf* prediction, where nearest is defined in terms of the spatial locations
func	radial basis function model type, e.g. "GAU", "EXPON", "TRI", "TPS", "CRS", "ST", "IM" and "M", are currently available

Value

returns the RMSPE value

See Also

rbf

Examples

```
data(preci)
coordinates(preci)<-~x+y
rbf.cv(prec~1, preci, eta=0.2589, rho=0, n.neigh=9, func="M")
```

`rbf.cv1` *Generates a RMSPE value, result of cross validation leave-one-out*

Description

Generate the RMSPE value, which is given by the radial basis function with smoothing parameter *eta* and robustness parameter *rho*.

Usage

```
rbf.cv1(param, formula, data, n.neigh, func)
```

Arguments

`param`	vector starting points (*eta* and *rho* respectively) for searching the *RMSPE* optimum.
`formula`	formula that defines the dependent variable as a linear model of independent variables; suppose the dependent variable has name *z*, for a *rbf* detrended use *z~1*, for a *rbf* with trend, suppose *z* is linearly dependent on *x* and *y*, use the formula *z~x+y* (linear trend).
`data`	SpatialPointsDataFrame: should contain the dependent variable, independent variables, and coordinates.
`n.neigh`	number of nearest observations that should be used for a *rbf* prediction, where nearest is defined in terms of the spatial locations
`func`	radial basis function model type, e.g. "GAU", "EXPON", "TRI", "TPS", "CRS", "ST", "IM" and "M", are currently available

Value

returns the RMSPE value

See Also

`rbf`

Examples

```
## Not run:
data(preci)
coordinates(preci) <- ~x+y
bobyqa(c(0.5, 0.5), rbf.cv1, lower=c(1e-05,0), upper=c(2,2), formula=prec~x+y, data=preci,
    n.neigh=9, func="TRI")
# obtained with the optimal values previously estimated
rbf.cv1(c(0.2126191,0.1454171), prec~x+y, preci, n.neigh=9, func="TRI")

## End(Not run)
```

RBF.phi *radial basis function evaluation*

Description

generate the value associated with radial basis functions; gaussian GAU), exponential (EXPON), trigonometric (TRI), thin plate spline (TPS), completely regularized spline (CRS), spline with tension (ST), inverse multiquadratic (IM), and multiquadratic (M)

Usage

```
RBF.phi(distance, eta, func)
```

Arguments

distance	corresponds to the Euclidean distance between two points in space
eta	the optimal smoothing parameter is found by minimizing the root-mean-square prediction errors using cross-validation
func	radial basis function model type, e.g. "GAU", "EXPON", "TRI", "TPS", "CRS", "ST", "IM" and "M", are currently available

Value

value obtained from the radial basis function generated with a distance, a *eta* smoothing parameter, and a function "GAU", "EXPON", "TRI", "TPS", "CRS", "ST", "IM" or "M"

Examples

```
data(preci)
d1 <- dist(rbind(preci[1,],preci[2,]))
RBF.phi(distance=d1, eta=0.5, func="TPS")
```

`rbf.tcv` *table of rbf cross validation, leave-one-out*

Description

Generates a table with the results of the evaluation of radial basis functions (*rbf*): gaussian (GAU), exponential (EXPON), trigonometric (TRI), thin plate spline (TPS), completely regularized spline (CRS), spline with tension (ST), inverse multiquadratic (IM), and multiquadratic (M) from the leave-one-out cross validation method.

Usage

```
rbf.tcv(formula, data, eta, rho, n.neigh, func)
```

Arguments

`formula`	formula that defines the dependent variable as a linear model of independent variables; suppose the dependent variable has name *z*, for a *rbf* detrended use *z~1*, for a *rbf* with trend, suppose *z* is linearly dependent on *x* and *y*, use the formula *z~x+y* (linear trend).
`data`	SpatialPointsDataFrame: should contain the dependent variable, independent variables, and coordinates.
`eta`	the optimal smoothing parameter; we recommend using the parameter found by minimizing the root-mean-square prediction errors using cross-validation
`rho`	value of optimal parameter robustness; we recommend using the parameter found by minimizing the root-mean-square prediction errors using cross-validation. *eta* and *rho* parameters can be optimized simultaneously, through the `bobyqa` function from `nloptr` or `minqa` packages
`n.neigh`	number of nearest observations that should be used for a *rbf* prediction, where nearest is defined in terms of the spatial locations
`func`	radial basis function model type, e.g. "GAU", "EXPON", "TRI", "TPS", "CRS", "ST", "MI" and "M", are currently available

Details

Leave-one-out cross validation (LOOCV) visits a data point, predicts the value at that location by leaving out the observed value, and proceeds with the next data point. The observed value is left out because rbf would otherwise predict the same value

Value

data frame contain the data coordinates, prediction columns, observed values, residuals, the prediction variance, zscore (residual divided by standard error) which left with NA's, and the fold column which is associated to cross-validation count. Prediction columns and residuals are obtained from cross-validation data points

See Also

rbf

Examples

```
data(preci)
coordinates(preci)<-~x+y
rbf.tcv(prec~x+y, preci, eta=0.1460814, rho=0, n.neigh=9, func="TPS")
```

---

samplePts *sample* n *point locations in (or on) a spatial object*

Description

sample location points within a square area, a grid, a polygon, or on a spatial line, using regular or random sampling methods. The function spsample from the package sp is used iteratively to find exactly *n* sample locations

Usage

```
samplePts(x, n, type, ...)
```

Arguments

x	Spatial object; see the sp package for details
n	exact sample size
type	character; "random" for completely spatial random; "regular" for regular (systematically aligned) sampling; "stratified" for stratified random (one single random location in each "cell"); "nonaligned" for nonaligned systematic sampling (nx random y coordinates, ny random x coordinates); "hexagonal" for sampling on a hexagonal lattice; "clustered" for clustered sampling; "Fibonacci" for Fibonacci sampling on the sphere. See the sp package for details
...	other arguments to be passed to spsample

Value

an object of class SpatialPoints-class

See Also

See spsample in the sp package

Examples

```
data(lalib)
hexPts <- samplePts(lalib, 5, "hexagonal")
plot(lalib, xlim=c(bbox(lalib)[1], bbox(lalib)[3]), ylim=c(bbox(lalib)[2],
    bbox(lalib)[4]))
plot(hexPts, add=TRUE)
## Not run:
randomPts <- samplePts(lalib, 5, "random")
plot(lalib, xlim=c(bbox(lalib)[1], bbox(lalib)[3]), ylim=c(bbox(lalib)[2],
    bbox(lalib)[4]))
plot(randomPts, add=TRUE)

## End(Not run)
```

`seqPtsOptNet` *Design of optimal sampling networks through the sequential points method*

Description

Search for the optimum location of one additional point to be added to an initial network, minimizing the average standard error of kriging through a genetic algorithm. It takes, as input for the optimization, the minimum and maximum values of the coordinates that enclose the study area. This function uses previous samples information to direct additional sampling. The location of the new point is searched randomly.

Usage

```
seqPtsOptNet(formula, loc=NULL, data, fitmodel, BLUE=FALSE, n=1, prevSeqs=NULL,
popSize, generations, xmin, ymin, xmax, ymax, plotMap=FALSE, spMap=NULL, ...)
```

Arguments

`formula`	formula that defines the interpolation method to be used. In this parameter, a dependent variable is defined as a linear model of independent variables. Suppose the dependent variable has name `z`, for ordinary and simple kriging use the formula `z~1`; for simple kriging also define `beta`; for universal kriging, suppose `z` is linearly dependent on `x` and `y`, use the formula `z~x+y`. See the `gstat` package for details
`loc`	object of class Spatial, or (deprecated) formula that defines the spatial data locations (coordinates) such as `~x+y`; see the `gstat` package for details
`data`	data frame containing the dependent variable, independent variables, and coordinates; see the `gstat` package for details
`fitmodel`	variogram model of dependent variable (or its residuals), defined by a call to `vgm` or `fit.variogram`; see the `gstat` package for details

BLUE	logical; if TRUE return the BLUE trend estimates only, if FALSE return the BLUP predictions (kriging); see predict.gstat in the gstat package for details
n	by default, set to 1 for the sequential points method
prevSeqs	if NULL, the function finds the first optimum sequential point. Otherwise, an object of class SpatialPoints containing optimum points previously found must be provided
popSize	population size; see the genalg package for details
generations	number of iterations. Usually, hundreds or thousands are required. See the genalg package for details
xmin	minimum *x*-coordinate of the study area
ymin	minimum *y*-coordinate of the study area
xmax	maximum *x*-coordinate of the study area
ymax	maximum *y*-coordinate of the study area
plotMap	logical; if TRUE, the optimized spatial locations of additional points are plotted
spMap	an object of class Spatial; it must be provided if plotMap is set to TRUE
...	other arguments to be passed to gstat or rbga

Value

an object of class rbga containing the population and the evaluation of the objective function for each chromosome in the last generation, the best and mean evaluation value in each generation, and additional information

References

Santacruz, A., Rubiano, Y., Melo, C., 2014. *Evolutionary optimization of spatial sampling networks designed for the monitoring of soil carbon*. In: Hartemink, A., McSweeney, K. (Eds.). *Soil Carbon. Series: Progress in Soil Science*. (pp. 77-84). Springer. [link]

Santacruz, A., 2011. *Evolutionary optimization of spatial sampling networks. An application of genetic algorithms and geostatistics for the monitoring of soil organic carbon*. Editorial Academica Espanola. 183 p. ISBN: 978-3-8454-9815-7 (In Spanish)

Delmelle, E., 2005. *Optimization of second-phase spatial sampling using auxiliary information*. Ph.D. Thesis, Dept. Geography, State University of New York, Buffalo, NY.

See Also

See rbga in the genalg package and krige in the gstat package

Examples

```
## Not run:
## Load data
data(COSha10)
data(COSha10map)
data(lalib)
```

```
## Calculate the sample variogram for data, generate the variogram model and
## fit ranges and/or sills from the variogram model to the sample variogram
ve <- variogram(CorT~ 1, loc=~x+y, data=COSha10, width = 230.3647)
PSI <- 0.0005346756; RAN <- 1012.6411; NUG <- 0.0005137079
m.esf <- vgm(PSI, "Sph", RAN, NUG)
(m.f.esf <- fit.variogram(ve, m.esf))

## Optimize the location of the first additional point
## Only 15 generations are evaluated in this example (using ordinary kriging)
## Users can visualize how the location of the additional point is optimized
## if plotMap is set to TRUE
old.par <- par(no.readonly = TRUE)
par(ask=FALSE)
optpt <- seqPtsOptNet(CorT~ 1, loc=~x+y, COSha10, m.f.esf, popSize=30,
    generations=15, xmin=bbox(lalib)[1], ymin=bbox(lalib)[2], xmax=bbox(lalib)[3],
    ymax=bbox(lalib)[4], plotMap=TRUE, spMap=lalib)
par(old.par)

## Summary of the genetic algorithm results
summary(optpt, echo=TRUE)

## Graph of best and mean evaluation value of the objective function
## (average standard error) along generations
plot(optpt)

## Find and plot the best set of additional points (best chromosome) in
## the population in the last generation
(bnet1 <- bestnet(optpt))
l1 = list("sp.polygons", lalib)
l2 = list("sp.points", bnet1, col="green", pch="*", cex=5)
spplot(COSha10map, "var1.pred", main="Location of the optimized point",
    col.regions=bpy.colors(100), scales = list(draw =TRUE), xlab ="East (m)",
    ylab = "North (m)", sp.layout=list(l1,l2))

## Average standard error of the optimized sequential point
min(optpt$evaluations)

## Optimize the location of the second sequential point, taking into account
## the first one
plot(lalib)
old.par <- par(no.readonly = TRUE)
par(ask=FALSE)
optpt2 <- seqPtsOptNet(CorT~ 1, loc=~x+y, COSha10, m.f.esf, prevSeqs=bnet1,
    popSize=30, generations=15, xmin=bbox(lalib)[1], ymin=bbox(lalib)[2],
    xmax=bbox(lalib)[3], ymax=bbox(lalib)[4], plotMap=TRUE, spMap=lalib)
par(old.par)

## Find the second optimal sequential point and use it, along with the first
## one, to find another optimal sequential point, and so on iteratively

bnet2 <- bestnet(optpt2)
bnet <- rbind(bnet1, bnet2)
```

```
old.par <- par(no.readonly = TRUE)
par(ask=FALSE)
optpt3 <- seqPtsOptNet(CorT~ 1, loc=~x+y, COSha10, m.f.esf, prevSeqs=bnet,
    popSize=30, generations=25, xmin=bbox(lalib)[1], ymin=bbox(lalib)[2],
    xmax=bbox(lalib)[3], ymax=bbox(lalib)[4], plotMap=TRUE, spMap=lalib)
par(old.par)

## End(Not run)

## Multivariate prediction is also enabled:
## Not run:
ve <- variogram(CorT~ DA10, loc=~x+y, data=COSha10, width = 230.3647)
(m.f.esf <- fit.variogram(ve, m.esf))

optptMP <- seqPtsOptNet(CorT~ DA10, loc=~x+y, COSha10, m.f.esf, popSize=30,
    generations=25, xmin=bbox(lalib)[1], ymin=bbox(lalib)[2], xmax=bbox(lalib)[3],
    ymax=bbox(lalib)[4], plotMap=TRUE, spMap=lalib)

## End(Not run)
```

`simPtsOptNet` *Design of optimal sampling networks through the simultaneous points method*

Description

Search for an optimum set of simultaneous additional points to an initial network that minimize the average standard error of kriging, using a genetic algorithm. It takes, as input for the optimization, the minimum and maximum values of the coordinates that enclose the study area. This function uses previous samples information to direct additional sampling for minimum average standard error. The algorithm generates random sampling schemes.

Usage

```
simPtsOptNet(formula, loc=NULL, data, fitmodel, BLUE=FALSE, n, popSize,
generations, xmin, ymin, xmax, ymax, plotMap=FALSE, spMap=NULL, ...)
```

Arguments

`formula`	formula that defines the interpolation method to be used. In this parameter, a dependent variable is defined as a linear model of independent variables. Suppose the dependent variable has name `z`, for ordinary and simple kriging use the formula `z~1`; for simple kriging also define `beta`; for universal kriging, suppose `z` is linearly dependent on `x` and `y`, use the formula `z~x+y`. See the `gstat` package for details
`loc`	object of class Spatial, or (deprecated) formula that defines the spatial data locations (coordinates) such as `~x+y`; see the `gstat` package for details

data	data frame containing the dependent variable, independent variables, and coordinates; see the gstat package for details
fitmodel	variogram model of dependent variable (or its residuals), defined by a call to vgm or fit.variogram; see the gstat package for details
BLUE	logical; if TRUE return the BLUE trend estimates only, if FALSE return the BLUP predictions (kriging); see predict.gstat in the gstat package for details
n	number of additional points to be added to the original network
popSize	population size; see the genalg package for details
generations	number of iterations. Usually, hundreds or thousands are required. See the genalg package for details
xmin	minimum *x*-coordinate of the study area
ymin	minimum *y*-coordinate of the study area
xmax	maximum *x*-coordinate of the study area
ymax	maximum *y*-coordinate of the study area
plotMap	logical; if TRUE, the optimized spatial locations of additional points are plotted
spMap	an object of class Spatial; it must be provided if plotMap is set to TRUE
...	other arguments to be passed to gstat or rbga

Value

an object of class rbga containing the population and the evaluation of the objective function for each chromosome in the last generation, the best and mean evaluation value in each generation, and additional information

References

Santacruz, A., Rubiano, Y., Melo, C., 2014. *Evolutionary optimization of spatial sampling networks designed for the monitoring of soil carbon*. In: Hartemink, A., McSweeney, K. (Eds.). *Soil Carbon. Series: Progress in Soil Science*. (pp. 77-84). Springer. [link]

Santacruz, A., 2011. *Evolutionary optimization of spatial sampling networks. An application of genetic algorithms and geostatistics for the monitoring of soil organic carbon*. Editorial Academica Espanola. 183 p. ISBN: 978-3-8454-9815-7 (In Spanish)

Delmelle, E., 2005. *Optimization of second-phase spatial sampling using auxiliary information*. Ph.D. Thesis, Dept. Geography, State University of New York, Buffalo, NY.

See Also

See rbga in the genalg package and krige in the gstat package

Examples

```
## Not run:
## Load data
data(COSha30)
data(COSha30map)
data(lalib)

## Calculate the sample variogram for data, generate the variogram model and
## fit ranges and/or sills from the variogram model to the sample variogram
ve <- variogram(CorT~ 1, loc=~x+y, data=COSha30, width = 236.0536)
PSI <- 0.0001531892; RAN <- 1031.8884; NUG <- 0.0001471817
m.esf <- vgm(PSI, "Sph", RAN, NUG)
(m.f.esf <- fit.variogram(ve, m.esf))

## Number of additional points to be added to the network
npoints <- 5

## Optimize the location of the additional points
## Only 20 generations are evaluated in this example (using ordinary kriging)
## Users can visualize how the location of the additional points is optimized
## if plotMap is set to TRUE
old.par <- par(no.readonly = TRUE)
par(ask=FALSE)
optnets <- simPtsOptNet(CorT~ 1, loc=~x+y, COSha30, m.f.esf, n=npoints,
    popSize=30, generations=20, xmin=bbox(lalib)[1], ymin=bbox(lalib)[2],
    xmax=bbox(lalib)[3], ymax=bbox(lalib)[4], plotMap=TRUE, spMap=lalib)
par(old.par)

## Summary of the genetic algorithm results
summary(optnets, echo=TRUE)

## Graph of best and mean evaluation value of the objective function
## (average standard error) along generations
plot(optnets)

## Find and plot the best set of additional points (best chromosome) in
## the population in the last generation
(bnet <- bestnet(optnets))
l1 = list("sp.polygons", lalib)
l2 = list("sp.points", bnet, col="green", pch="*", cex=5)
spplot(COSha30map, "var1.pred", main="Location of the optimized points",
    col.regions=bpy.colors(100), scales = list(draw =TRUE), xlab ="East (m)",
    ylab = "North (m)", sp.layout=list(l1,l2))

## Average standard error of the optimized additional points
min(optnets$evaluations)

## End(Not run)

## Multivariate prediction is also enabled:
## Not run:
ve <- variogram(CorT~ DA30, loc=~x+y, data=COSha30, width = 236.0536)
```

```
(m.f.esf <- fit.variogram(ve, m.esf))

optnetsMP <- simPtsOptNet(CorT~ DA30, loc=~x+y, COSha30, m.f.esf, n=npoints,
    popSize=30, generations=25, xmin=bbox(lalib)[1], ymin=bbox(lalib)[2],
    xmax=bbox(lalib)[3], ymax=bbox(lalib)[4], plotMap=TRUE, spMap=lalib)

## End(Not run)
```

Index