Package 'gear'

June 15, 2019

Type Package

Title Geostatistical Analysis in R

Version 0.1.4

Date 2019-06-14

Author Joshua French

Maintainer Joshua French <joshua.french@ucdenver.edu>

Description Implements common geostatistical methods in a clean, straightforward, efficient manner. The methods are discussed in Schabenberger and Gotway (2004, <ISBN:9781584883227>) and Waller and Gotway (2004, <ISBN:9780471387718>). This package is a quasi-reboot of the 'SpatialTools' package.

License GPL (>= 2)

Imports sp, parallel, lattice, stats, optimx

Suggests testthat, gstat, geoR, spam, Matrix

Encoding UTF-8

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-06-14 22:50:14 UTC

R topics documented:

`angle2d` *Determine angle*

Description

`angle2d` determines the angle between pairs of coordinates in degrees or radians. The coordinats are assumed to be in 2d space.

Usage

```
angle2d(coords1, coords2, radians = FALSE)
```

Arguments

`coords1`	An $N \times 2$ matrix of spatial coordinates.
`coords2`	An $N \times 2$ matrix of spatial coordinates.
`radians`	A logical indicating whether degrees or radians should be returned. Default is FALSE, meaning return angle in degrees.

Details

Note that the angle is between the actual pairs of points, not the angle between the vectors extending from the origin to the points. e.g., the angle between cbind(0, 1) and cbind(1, 1) would be 90 degrees, not 45. Sign of the direction not accounted for, e.g., a -135 degree angle is rotated by 180 degrees to become a 45 degree angle. All angles returned are in the interval [0, 180].

Value

Returns a vector of angles.

Author(s)

Joshua French

Examples

```
coords1 = matrix(0, nrow = 8, ncol = 2)
coords2 = cbind(c(2, 2, 0, -2, -2, -2, 0, 2), c(0, 2, 2, 2, 0, -2, -2, -2))
angle2d(coords1, coords2)
angle2d(coords1, coords2, radians = TRUE)
```

cmod.man *Standard covariance models for geostatistical data.*

Description

cmod.man manually creates a covariance matrix object (cmodMan) for geostatistical data.

Usage

```
cmod.man(v, evar = 0)
```

Arguments

v	The covariance matrix of the observed data, including any errors. The matrix should be square, symmetric, and positive definite, though that latter two conditions are not checked.
evar	The variance of the errors. Must be non-negative number. The default is 0.

Details

Note that v includes the error variance, i.e., v = vz + ve, where vz is the covariance matrix of the filtered (non-noisy) process, and the variance matrix of the errors is ve = diag(evar/weights), where the weights come from the geolm object the covariance object is associated with.

Value

Returns a cmodMan object.

Author(s)

Joshua French

Examples

```
coords = matrix(runif(20), ncol = 2)
d = as.matrix(dist(coords))
cmod.man(v = exp(-d), evar = 1)
```

cmod.std *Standard covariance models for geostatistical data.*

Description

Creates a standard covariance model (cmodStd) object for geostatistical data.

Usage

```
cmod.std(model, psill, r, evar = 0, fvar = 0, par3 = 0.5)
```

Arguments

model	A standard semivariance model type.
psill	The partial sill of the model. Must be a postive number.
r	The range parameter r. Must be a positive number.
evar	The variance of the errors. Must be non-negative number. The default is 0.
fvar	The finescale variance (microscale error). Must be a non-negative number. The default is 0.
par3	The value of the third parameter for 3 parameter models. Must be a positive number. The default is 0.5.

Details

The general form of the specified covariance function is psill * ρ(d; r) + (evar + fvar)*(d==0), where ρ is the covariance function of the parametric models.

For the exponential model, ρ(d; r) is exp(-d/r).

For the gaussian model, ρ(d; r) is exp(-d^2/r^2).

For the matern model, ρ(d; r) is 2^(1-par3)/gamma(par3)*sd^par3*besselK(sd, nu = par3), where sd = d/r.

For the amatern (alternative Matern) model, ρ(d; r) is 2^(1-par3)/gamma(par3)*sd^par3*besselK(sd, nu = par3), where sd = 2 * sqrt(par3) * d/r.

For the spherical model, ρ(d; r) is 1 - 1.5*sd + 0.5*(sd)^3 if d < r, and 0 otherwise, with sd = d/r.

For the wendland1 model, ρ(d; r) is (1 - sd)^4 * (4*sd + 1) if d < r, and 0 otherwise, with sd = d/r.

For the wendland2 model, ρ(d; r) is (1 - sd)^6 * (35*sd^2 + 18*sd + 3))/3 if d < r, and 0 otherwise, with sd = d/r.

For the wu1 model, ρ(d; r) is (1 - sd)^3 * (1 + 3*sd + sd^2) if d < r, and 0 otherwise, with sd = d/r.

For the wu2 model, ρ(d; r) is (1 - sd)^4*(4 + 16*sd + 12*sd^2 + 3*sd^3))/4 if d < r, and 0 otherwise, with sd = d/r.

For the wu3 model, ρ(d; r) is (1 - sd)^6 * (1 + 6*sd + 41/3*sd^2 + 12*sd^3 + 5*sd^4 + 5/6*sd^5) if d < r, and 0 otherwise, with sd = d/r.

Value

Returns a cmodStd object.

Author(s)

Joshua French

References

Waller, L. A., & Gotway, C. A. (2004). Applied Spatial Statistics for Public Health Data. John Wiley & Sons.

See Also

covmat

Examples

```
cmod.std(model = "exponential", psill = 1, r = 1)
```

---

co *Geochemical measurements for 960 sites in Colorado.*

Description

These data were collected by the United States Geological Survey (USGS). Their description is as follows: In 2006, soil samples were collected at 960 sites (1 site per 280 square kilometers) throughout the state of Colorado. These samples were collected from a depth of 0 to 15 centimeters and, following a near-total multi-acid digestion, were analyzed for a suite of more than 40 major and trace elements. The resulting data set provides a baseline for the natural variation in soil geochemistry for Colorado and forms the basis for detecting changes in soil composition that might result from natural processes or anthropogenic activities.

Latitude and Longitude determined by hand-held GPS instrument using WGS84 datum. The longitude and latitude coordinates were converted to UTM coordinates using the following commands from the sp package: lonlat = co[, c("longitude", "latitude")]; coordinates(lonlat) <- c("longitude", "latitude") proj4string(lonlat) <- CRS("+proj=longlat +datum=WGS84") ## for example xy <- spTransform(lonlat, CRS("+proj=utm +zone=13 ellps=WGS84"))

Usage

```
data(co)
```

Format

A data frame with 960 rows and 31 columns:

easting Easting (m)
northing Northing (m)
latitude The latitude of the site.
longitude The longitude of the site.
Al aluminum (percent)
Ca calcium (percent)
Fe iron (percent)
K potassium (percent)
Mg magnesium (percent)
Na sodium (percent)
Ti titanium (percent)
Be beryllium (mg/kg)
Ce cerium (mg/kg)
Co cobalt (mg/kg)
Cr chromium (mg/kg)
Cu copper (mg/kg)
Ga gallium (mg/kg)
La lanthanum (mg/kg)
Li lithium (mg/kg)
Mo manganese (mg/kg)
Nb molybdenum (mg/kg)
Ni niobium (mg/kg)
Rb rubidium (mg/kg)
Sc scandium (mg/kg)
Sn tin (mg/kg)
Th thorium (mg/kg)
Tl thallium (mg/kg)
U uranium (mg/kg)
V vanadium (mg/kg)
W tungsten (mg/kg)
Y yttrium (mg/kg)

Source

U.S. Geological Survey, Data Series 520, 9 p. http://pubs.usgs.gov/ds/520/.

References

Smith, D.B., Ellefsen, K.J., and Kilburn, J.E., 2010, Geochemical data for Colorado soils – Results from the 2006 state-scale geochemical survey: U.S. Geological Survey, Data Series 520, 9p.

decomp.cov *Decompose covariance matrix*

Description

decomp.cov decomposes a covariance matrix v. If A = decomp.cov(v), then tcrossprod(A, A) == v.

Usage

```
decomp.cov(v, method = "eigen")
```

Arguments

v	An $n \times n$ covariance matrix.
method	The method used to decompose v. valid options are "chol", "eigen", or "svd".

Details

The "chol" method is the fastest, but must unstable. The "eigen" method is slower, but more stable. The "svd" method is the slowest method, but should be the most stable.

Value

Returns an $n \times n$ matrix.

Author(s)

Joshua French

Examples

```
# generate data
n = 100
coords = matrix(runif(n*2), nrow = n, ncol = 2)
d = as.matrix(dist(coords))
# create covariance matrix
v = 3*exp(-d/2) + 0.1*diag(n)

# decompose v using the three methods
d1 = decomp.cov(v, "chol")
d2 = decomp.cov(v, "eigen")
d3 = decomp.cov(v, "svd")

# verify accuracy of decompositions
range(v - tcrossprod(d1))
range(v - tcrossprod(d2))
range(v - tcrossprod(d3))
```

`eval.cmod` *Evaluate covariance or semivariance model.*

Description

`eval.cmod` evaluates the covariance or semivariance of a model based on the provided arguments. See Details for what the function returns, as it changes depending on the class of `mod`.

Usage

```
eval.cmod(mod, d = NULL, coords = NULL)

## S3 method for class 'cmodStd'
eval.cmod(mod, d, coords = NULL)
```

Arguments

`mod`	A covariance or semivariance model.
`d`	An $n \times m$ matrix of distances.
`coords`	An numeric object with two columns.

Details

If `mod` is of class `cmodStd` (from the `cmod.std` function), then the function returns an $n \times m$ matrix with the evaluated standard covariance function.

Value

Returns the evaluated model with necessary components needed for `mlefit`, `predict`, or `loglik` functions.

Author(s)

Joshua French

Examples

```
n = 10
coords = matrix(runif(2*n), nrow = n, ncol = 2)
d = as.matrix(dist(coords))
cmod = cmod.std(model = "exponential", psill = 1, r = 1)
eval.cmod(cmod, d)
```

geolm *Linear model for geostatistical data.*

Description

geolm creates a geostatistical linear model object of the appropriate class based on the arguments, especially the cmod arguments.

Usage

```
geolm(formula, data, coordnames, cmod = NULL, vmod = NULL,
  weights = NULL, longlat = FALSE, mu = NULL)
```

Arguments

formula	An object of class formula. See Details.
data	A data frame containing the response, covariates, and location coordinates.
coordnames	A vector of length 2 with the names of the columns in data containing the coordinates, e.g., c("long", "lat").
cmod	A covariance model object obtained from one of the cmod.* functions, e.g., cmod.std.
vmod	A semivariance model object obtained from one of the vmod.* functions. Not currently implemented.
weights	An optional vector of weights for the errors to be used in the fitting process. A vector that is proportional to the reciprocal variances of the errors, i.e., errors are assumed to be uncorrelated with variances evar/weights. Default is NULL, meaning that the weights are uniformly 1.
longlat	A logical value. Default is FALSE. If TRUE, Great Circle distances (WGS84 ellipsoid) are calcualted between locations. Otherwise, Euclidean.
mu	A single numeric value indicating the consant mean of the spatial process if simple kriging is desired. Default is NULL, meaning that ordinary or universal kriging should be used.

Details

formula should be specified after the form y ~ x1 + x2, where y is the response variable and x1 and x2 are the covariates of interest. If mu is provided, the variables to the right of ~ are ignored.

Value

Returns a geolm object.

Author(s)

Joshua French

Examples

```
data = data.frame(y = rnorm(10), x1 = runif(10),
                  x2 = runif(10))
cmod = cmod.std(model = "exponential", psill = 1,
                  r = 1)
gearmod = geolm(y ~ x1, data = data,
                  coordnames = c("x1", "x2"),
                  cmod = cmod)
```

`mle` *Finds maximum likelihood estimates of model parameters for a geostatistical model*

Description

`mle` estimates the parameters of a geostatistical linear model. The function is written to automatically adapt based on the class of `object`. See Details.

Usage

```
mle(object, reml = FALSE, est = "e", ...)

## S3 method for class 'geolmStd'
mle(object, reml = FALSE, est = "e", lower = NULL,
  upper = NULL, method = "nlminb", itnmax = NULL, control = list(),
  ...)
```

Arguments

`object`	A geostatistical linear model object producted by the `geolm` function.
`reml`	A logical value indicating whether standard maximum likelihood estimation should be performed (`reml = FALSE`). If `reml = TRUE`, then restricted maximum likelihood is performed. Defaul is `FALSE`.
`est`	A character vector indicator whether the error variance (`est="e"`) or finescale variance (`est = "f"`) should be estimated. The other component of the nugget variance is held constant, and in the case of a `geolmStd` object, is set to 0.
`...`	Currently unimplemented.
`lower`	A vector of 2 or 3 specifying the lowerbound of parameter values. See Details.
`upper`	lower A vector of 2 or 3 specifying the lowerbound of parameter values. See Details.
`method`	The optimization method. Default is `"nlminb"`, with `"L-BFGS-B"` being another acceptable choice. See `optimx` for details.
`itnmax`	An integer indicating the maximum number of iterations to allow for the optimization prodedure.
`control`	A list of control parameters passed internally to `optimx`. See `optimx` for details.

Details

In the case of a geolmStd object, the likelihood has been concentrated so that only the range parameter r and a scale parameter lambda = nugget/psill need to be estimated.

If object is a geolmStd, then lower is of length 2 if the covariance model of cmod is not matern or amatern. Otherwise, it should be of length 3. The first parameter is related to the range parameter r, the second to the scale parameter lambda, and the third to par3, if applicable. If lower = NULL, then the lower bounds are 0.001, 0, and 0.1, respectively. A similar pattern holds for upper, with the default being 3 * max(d), where d is the matrix of distances between coordinates, 5, and 2.5.

The kkt argument in the control list is set to be FALSE.

Author(s)

Joshua French

Examples

```
set.seed(10)
n = 100
```

---

plot.vgram *Plot vgram object*

Description

Plots object of class 'vgram' produced by the vgram function. The plot is based on the lattice::xyplot function.

Usage

```
## S3 method for class 'vgram'
plot(x, ..., split = FALSE)
```

Arguments

x	A vgram object produced by the vgram function.
...	Additional arguments to pass the lattice::xyplot function to change aspects of the plot.
split	A logical value indicating whether, for a directional semivariogram, the directional semivariograms should be displayed in a single or split panels. Default is FALSE, for a single panel.

Author(s)

Joshua French

See Also

xyplot, vgram

Examples

```
data(co)
v = vgram(Al ~ 1, co, ~ easting + northing)
plot(v)
v2 = vgram(Al ~ 1, co, ~ easting + northing, angle = 22.5, ndir = 4)
plot(v2)
# show how attributes can be changed using different arguments
# available in \code{lattice::xyplot}.
plot(v2, col = 2:5)
plot(v2, col = 2:5, pch = 1:4)
plot(v2, col = 2:5, pch = 1:4, lty = 2:5, type = "b")
plot(v2, col = 2:5, pch = 1:4, lty = 2:5, type = "b",
 key=list(text=list(levels(as.factor(v2$semi$angle))),
 space='right', points=list(pch=1:4, col=2:5),
 lines=list(col=2:5, lty = 2:5)))
plot(v2, split = TRUE)
```

---

predict.geolmMan *Predict method for geostatistical models*

Description

predict calculates the predicted values at specified locations. The method can additionally provide the mean square prediction error (mspe) and perform conditional simulation.

Usage

```
## S3 method for class 'geolmMan'
predict(object, newdata, nsim = 0, vop, vp,
  sp = TRUE, dmethod = "chol", ...)
```

Arguments

object	An object produced by the geolm function.
newdata	An optional data frame in which to look for the coordinates at which to predict. If omitted, the observed data locations are used.
nsim	A non-negative integer indicating the number of realizations to sample at the specified coordinates using conditional simulation.
vop	The cross-covariance matrix between the observed responses and the responses to predict.
vp	The covariance matrix of the responses to predict.
sp	A logical value indicating whether to object returned should be of class SpatialPointsDataFrame for easier plotting with the sp package. Default is TRUE.

dmethod	The method used to decompose the covariance matrix for conditional simulation. Valid options are "chol", "eigen", and "svd". The default is "chol".
...	Currently unimplemented.

Details

The newdata data frame must include the relevant covariates for the prediction locations, where the covariates are specified on the right side of the ~ in object$formula. newdata must also include the coordinates of the prediction locations, with these columns having the names provided in object$coordnames.

Value

If sp = TRUE, then a SpatialPointDataFrame from the sp package is returned, with components including the prediction coordinates, the predicted responses pred, the mean square prediction error (mspe), the root mean square prediction error (rmspe), and the conditional realizations, if application (sim.1, sim.2, ...). If sp = FALSE, then a list of class gearKrige is returned, with components pred, mspe, rmspe, and sim, if relevant.

Author(s)

Joshua French

Examples

```
# generate response
y = rnorm(10)
# generate coordinates
x1 = runif(10); x2 = runif(10)

# data frame for observed data
data = data.frame(y, x1, x2)
coords = cbind(x1, x2)
d = as.matrix(dist(coords))
psill = 2 # partial sill
r = 4 # range parameter
evar = .1 # error variance
fvar = .1 # add finescale variance
# one can't generally distinguish between evar and fvar, but
# this is done for illustration purposes

# manually specify a an expoential covariance model
v = psill * exp(-d/r) + (evar + fvar) * diag(10)

cmod_man = cmod.man(v = v, evar = evar)

#' # geolm for universal kriging
gearmod_uk = geolm(y ~ x1 + x2, data = data,
                   coordnames = c("x1", "x2"),
                   cmod = cmod_man)
```

```
# newdata must have columns with prediction coordinates
# add 5 unsampled sites to sampled sites
newdata = data.frame(x1 = c(x1, runif(5)), x2 = c(x2, runif(5)))
newcoords = newdata[,c("x1", "x2")]
# create vop and vp using distances
dop = sp::spDists(as.matrix(coords), as.matrix(newcoords))
dp = as.matrix(dist(newcoords))

vop = psill * exp(-dop/r) + fvar * (dop == 0)
vp = psill * exp(-dp/r) + fvar * diag(nrow(newcoords))

# prediction for universal kriging, with conditional simulation,
# using manual covariance matrices
pred_uk_man = predict(gearmod_uk, newdata, nsim = 2, vop = vop, vp = vp,
                      dmethod = "svd")

# do the same thing, but using the std covariance function

# prediction for universal kriging, with conditional simulation
cmod_std = cmod.std("exponential", psill = psill, r = r, evar = evar, fvar = fvar)
gearmod_uk2 = geolm(y ~ x1 + x2, data = data, coordnames = c("x1", "x2"),
                    cmod = cmod_std)
pred_uk_std = predict(gearmod_uk2, newdata, nsim = 2, dmethod = "svd")

# compare results
range(pred_uk_man$pred - pred_uk_std$pred)
range(pred_uk_man$mspe - pred_uk_std$mspe)
```

`predict.geolmStd` *Predict method for geostatistical models*

Description

`predict` calculates the predicted values at specified locations. The method can additionally provide the mean square prediction error (mspe) and perform conditional simulation.

Usage

```
## S3 method for class 'geolmStd'
predict(object, newdata, nsim = 0, sp = TRUE,
  dmethod = "chol", ...)
```

Arguments

`object`	An object produced by the `geolm` function.
`newdata`	An optional data frame in which to look for the coordinates at which to predict. If omitted, the observed data locations are used.

nsim	A non-negative integer indicating the number of realizations to sample at the specified coordinates using conditional simulation.
sp	A logical value indicating whether to object returned should be of class SpatialPointsDataFrame for easier plotting with the sp package. Default is TRUE.
dmethod	The method used to decompose the covariance matrix for conditional simulation. Valid options are "chol", "eigen", and "svd". The default is "chol".
...	Currently unimplemented.

Details

The newdata data frame must include the relevant covariates for the prediction locations, where the covariates are specified on the right side of the ~ in object$formula. newdata must also include the coordinates of the prediction locations, with these columns having the names provided in object$coordnames.

Value

If sp = TRUE, then a SpatialPointDataFrame from the sp package is returned, with components including the prediction coordinates, the predicted responses pred, the mean square prediction error (mspe), the root mean square prediction error (rmspe), and the conditional realizations, if application (sim.1, sim.2, ...). If sp = FALSE, then a list of class gearKrige is returned, with components pred, mspe, rmspe, and sim, if relevant.

Author(s)

Joshua French

Examples

```
# generate response
y = rnorm(10)
# generate coordinates
x1 = runif(10); x2 = runif(10)

# data frame for observed data
data = data.frame(y, x1, x2)
# newdata must have columns with prediction coordinates
newdata = data.frame(x1 = runif(5), x2 = runif(5))

# specify a standard covariance model
cmod = cmod.std(model = "exponential", psill = 1,
                r = 1)

# geolm for universal kriging
gearmod_uk = geolm(y ~ x1 + x2, data = data,
                   coordnames = c("x1", "x2"),
                   cmod = cmod)
# prediction for universal kriging, with conditional simulation
pred_uk = predict(gearmod_uk, newdata, nsim = 2)
```

```
# geolm for ordinary kriging
gearmod_ok = geolm(y ~ 1, data = data,
                   coordnames = c("x1", "x2"),
                   cmod = cmod)
# prediction for ordinary kriging
pred_ok = predict(gearmod_ok, newdata)

# geolm for simple kriging
gearmod_ok = geolm(y ~ 1, data = data,
                   coordnames = c("x1", "x2"),
                   cmod = cmod, mu = 1)
# prediction for simple kriging
pred_sk = predict(gearmod_ok, newdata)
```

`update.geolmStd` *Update linear model for geostatistical data.*

Description

`update` updates a geostatistical linear model based on the given covariance model.

Usage

```
## S3 method for class 'geolmStd'
update(object, cmod, ...)

## S3 method for class 'geolmMan'
update(object, cmod, ...)
```

Arguments

`object`	An object produced by the `geolm` function.
`cmod`	A covariance model object obtained from one of the `cmod.*` functions or the `mle` function.
`...`	Not implemented.

Value

Returns an object of the same class as `object`.

Author(s)

Joshua French

Examples

```
# generate response
y = rnorm(10)
# generate coordinates
x1 = runif(10); x2 = runif(10)

# data frame for observed data
data = data.frame(y, x1, x2)
coords = cbind(x1, x2)
psill = 2 # partial sill
r = 4 # range parameter
evar = .1 # error variance
fvar = .1 # add finescale variance
# one can't generally distinguish between evar and fvar, but
# this is done for illustration purposes

cmod_std = cmod.std("exponential", psill = psill, r = r,
                    evar = evar, fvar = fvar)

cmod_std2 = cmod.std("exponential", psill = psill + 1, r = r + .5,
                     evar = evar + .01, fvar = fvar)

# check geolm update for universal kriging
gear1 = geolm(y ~ x1 + x2, data = data,
              coordnames = c("x1", "x2"),
              cmod = cmod_std)
gear2 = geolm(y ~ x1 + x2, data = data,
              coordnames = c("x1", "x2"),
              cmod = cmod_std2)
gear2b = update(gear1, cmod_std2)
identical(gear2, gear2b)
```

---

vgram *Sample semivariogram*

Description

vgram calculates the sample semivariogram.

Usage

```
vgram(formula, data, coords = NULL, nbins = 10, maxd = NULL,
  angle = 0, ndir = 1, type = "standard", npmin = 2,
  longlat = FALSE)
```

Arguments

formula	A formula describing the relationship between the response and any covariates of interest, e.g., response ~ 1. The variogram is computed for the residuals of the linear model lm(formula, data).

data	A data.frame, SpatialPointsDataFrame, SpatialPixelsDataFrame, or SpatialGridDataFrame object.
coords	A formula specifiying (on the righthand side of ~) the coordinates in data. Default is NULL, but this must be specified if data is of class data.frame. This will probably look something like "~ x + y".
nbins	The number of bins (tolerance regions) to use when estimating the sample semivariogram.
maxd	The maximum distance used when calculating the semivariogram. Default is NULL, in which case half the maximum distance between coordinates is used.
angle	A single value (in degrees) indicating the starting direction for a directional variogram. The default is 0.
ndir	The number of directions for which to calculate a sample semivariogram. The default is 1, meaning calculate an omnidirection semivariogram.
type	The name of the estimator to use in the estimation process. The default is "standard", the typical method-of-moments estimator. Other options include "cressie" for the robust Cressie-Hawkins estimator, and "cloud" for a semivariogram cloud based on the standard estimator. If "cloud" is specified, the nbins argument is ignored.
npmin	The minimum number of pairs of points to use in the semivariogram estimator. For any bins with fewer points, the estimate for that bin is dropped.
longlat	A logical indicating whether Euclidean (FALSE) or Great Circle distance (WGS84 ellipsoid) (longlat = TRUE) should be used. Default is FALSE.

Details

Note that the directions may be different from other packages (e.g., gstat or geoR packages) because those packages calculate angles clockwise from the y-axis, which is a convention frequently seen in geostatistics (e.g., the GSLIB software library).

Additionally, note that calculating the sample semivariogram for the residuals of lm(response ~ 1) will produce identical results to simply computing the sample semivariogram from the original response. However, if a trend is specified (the righthand side of ~ has non-trival covariates), then the sample semivariogram of the residuals will differ from that of the original response. A trend should be specified when the mean is non-stationary over the spatial domain.

Value

Returns a vgram object with components:

Author(s)

Joshua French

Examples

```
data(co)
v = vgram(Al ~ 1, co, ~ easting + northing)
plot(v)
```

```
v2 = vgram(Al ~ 1, co, ~ easting + northing, angle = 22.5, ndir = 4)
plot(v2)
```

Index