Package 'gear'

April 10, 2020

Type Package

Title Geostatistical Analysis in R

Version 0.3.4

Date 2020-04-06

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>).

License GPL (>= 2)

Imports autoimage, stats, optimx, Rcpp

Suggests sp, sf, testthat, Matrix, geoR, gstat, spam, ggplot2, lattice

Encoding UTF-8

NeedsCompilation yes

RoxygenNote 7.1.0

LinkingTo Rcpp

Repository CRAN

Date/Publication 2020-04-10 21:20:02 UTC

R topics documented:

angle2d 2
autoplot.evgram 3
cmod.man 4
cmod.std 5
cmod_man 6
cmod_std 7
co 9
decomp_cov 11
estimate 12
estimate.geolm_cmodStd 12

eval.cmod . . . 15
evaluate.cmodStd . . . 16
evgram . . . 17
fitted.geolm . . . 19
ganiso_d . . . 20
gear . . . 20
geardf . . . 21
geodist . . . 22
geolm . . . 23
geolm_fit . . . 24
mle . . . 25
ploglik_xycholv . . . 27
plot.evgram . . . 29
plot.geardf . . . 30
predict.geolm_cmodMan . . . 31
predict.geolm_cmodStd . . . 34
print.evgram . . . 36
residuals.geolm . . . 37
solve_chol . . . 38
toydata . . . 39
update.geolm . . . 39
vgram . . . 41
xyplot.evgram . . . 43

Index **45**

`angle2d` *Determine angle*

Description

`angle2d` determines the angle between pairs of coordinates in degrees or radians. The coordinates are assumed to be in two-dimensional space.

Usage

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

Arguments

`coords1`	An $N \times 2$ matrix of spatial coordinates.
`coords2`	An $N \times 2$ matrix of spatial coordinates.
`radians`	A logical value indicating whether the angles returned should be in degrees or radians. The default is `FALSE`, indicating that the returned angles are in degrees.
`invert`	A logical value indicating whether the axes of the coordinates should be inverted (i.e., the x- and y-axis are switched). The default is `TRUE` to mimic results from other geostatistical R packages like `gstat`, `geoR`, and other software like `GSLIB` and `GeoEAS`. Set to `FALSE` to use the typical x- and y-axes.

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 (0, 1) and (1, 1) is 90 degrees, not 45. The 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)
```

---

autoplot.evgram *Plot* evgram *object*

Description

Plot an evgram object produced by the evgram function. The plotting function internally calls the autoplot function. Note: the ggplot2 package must be loaded (i.e., library(autplot) or ggplot2::autoplot must be specifically called for this function to work. See Examples.

Usage

```
autoplot.evgram(object, ..., split = FALSE)
```

Arguments

object	An evgram object produced by the evgram function.
...	Not used
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

Examples

```
data(co)
v = evgram(Al ~ 1, co, ~ easting + northing)
if (requireNamespace("ggplot2")) {
   ggplot2::autoplot(v)
}
v2 = evgram(Al ~ 1, co, ~ easting + northing, angle = 22.5, ndir = 4)
# ggplot2 must manually be loaded for this to work
if (requireNamespace("ggplot2")) {
   ggplot2::autoplot(v2)
   ggplot2::autoplot(v2, split = TRUE)
}
```

---

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

Description

cmod.man manually creates a covariance matrix object (cmodMan) for geostatistical data. This function will be deprecated in the future. Please update your code to use the cmod_man function.

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 process and the variance matrix of the errors is ve = diag(evar/weights), where the weights come from the geolm object the cmodMan 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. This function will be deprecated in the future. Please update your code to use the cmod_std function, which also allows the user to specify geometric anisotropy.

Usage

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

Arguments

model	A covariance model (e.g., "exponential"). See Details for the complete list of choices.
psill	The partial sill of the model. Must be a positive 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.

Examples

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

---

cmod_man *Manual covariance models for geostatistical data.*

Description

cmod_man manually creates a covariance model 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 process and the variance matrix of the errors is ve = diag(evar/weights), where the weights come from the geolm object the cmodMan object is associated with.

Value

Returns a cmodMan object.

Author(s)

Joshua French

Examples

```
coords = matrix(runif(6), 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,
  longlat = FALSE,
  angle = 0,
  ratio = 1,
  radians = FALSE,
  invert = TRUE
)
```

Arguments

model	A covariance model (e.g., "exponential"). See Details for the complete list of choices.
psill	The partial sill of the model. Must be a positive 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.
longlat	A logical value indicating whether great circle distance should be used. The default is FALSE.
angle	The major axis of geometric anisotropy (the direction of strongest spatial dependence). Must be between [0, 180) if radians = FALSE, otherwise it must be between [0, π).
ratio	The ratio of the minor axis range over the major axis range. The value must be between (0, 1].
radians	A logical value indicating whether the angles returned should be in degrees or radians. The default is FALSE, indicating that the returned angles are in degrees.
invert	A logical value indicating whether the axes of the coordinates should be inverted (i.e., the x- and y-axis are switched). The default is TRUE to mimic results from other geostatistical R packages like gstat, geoR, and other software like GSLIB and GeoEAS. Set to FALSE to use the typical x- and y-axes.

Details

The general, isotropic form of the specified covariance function is psill * ρ(d; r) + (evar + fvar) * (d == 0), where ρ is the correlation function of the parametric models and d is the distance between the relevant coordinates.

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.

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:

```
library(sp)
lonlat = co[,c("longitude","latitude")]
coordinates(lonlat) = c("longitude","latitude")
proj4string(lonlat) = CRS("+proj=longlat +datum=WGS84")
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")

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 least stable method. 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
all.equal(v, tcrossprod(d1))
all.equal(v, tcrossprod(d2), check.attributes = FALSE)
all.equal(v, tcrossprod(d3), check.attributes = FALSE)
```

`estimate` *Estimate model parameters*

Description

`estimate` estimates the parameters of the specified model. The function is written to automatically adapt based on the class of `object`. Currently, the `estimate` function performs maximum likelihood estimation for objects produced by the `geolm` function.

Usage

```
estimate(object, ...)
```

Arguments

`object`	A model object produced by the `geolm` function.
`...`	Currently unimplemented

Author(s)

Joshua French

See Also

`estimate.geolm_cmodStd`,

Examples

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

`estimate.geolm_cmodStd`
Determine MLEs of model parameters for a geostatistical model

Description

`estimate` estimates the parameters of a geostatistical linear model of class `geolm_cmodStd` using maximum likelihood estimation.

Usage

```
## S3 method for class 'geolm_cmodStd'
estimate(
  object,
  reml = FALSE,
  noise_type = "e",
  lower = NULL,
  upper = NULL,
  method = "nlminb",
  itnmax = NULL,
  control = list(),
  est_nugget = TRUE,
  est_par3 = TRUE,
  est_angle = FALSE,
  est_ratio = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

object	A geostatistical linear model object produced 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 estimation is performed. The default is FALSE.
noise_type	A character vector indicating the type of noise (nugget) variance to estimate. The default is est = "e", indicating the error variance should be estimated. Alternatively, the user can specify est = "f", indicating that the finescale (microscale) variance should be estimated. The other type of noise variance is set to 0, otherwise the model is not identifiable. See Details.
lower	A named list with the names of the parameters you wish to set lower bounds for and the associated value. See Details.
upper	A named list with the names of the parameters you wish to set upper bounds for and the associated value.
method	The optimization method. The default is "nlminb". "L-BFGS-B" is another acceptable choice. See optimx for further choices.
itnmax	An integer indicating the maximum number of iterations to allow for the optimization procedure.
control	A list of control parameters passed internally to optimx.
est_nugget	A logical value indicating whether the nugget variance (evar or fvar) should be estimated. The default is TRUE, indicating that the nugget should be estimated.
est_par3	A logical value indicating whether par3 should be estimated (for an appropriate covariance model such as "matern" or "amatern"). The default is TRUE, indicating that this parameter should be estimated.

est_angle	A logical value indicating whether the geometric anisotropy angle should be estimated. The default is FALSE, indicating that this parameter should not be estimated.
est_ratio	A logical value indicating whether the geometric anisotropy ratio of minor axis length to major axis length should be estimated. The default is FALSE, indicating that this parameter should not be estimated.
verbose	A logical value indicating whether potentially informative messages should be printed. The default is FALSE.
...	Currently unimplemented

Details

The optimx function is used to find the MLEs. The control argument of optimx has a parameter kkt related to checking optimality conditions. This is internally set to FALSE. See optimx for Details.

Only the sum of evar and fvar is identifiable. Depending on the choice of noise_type, the covariance model is internally updated to estimate only one type of noise. e.g., if noise_type = "e", then internally we update evar so that evar = evar + fvar and fvar = 0. Estimation is then performed on evar alone. Alternatively, the analagous estimated would be made for fvar if noise_type = "fvar".

When est_nugget is true, the likelihood is profiled to simplify the optimization problem. In that case a parameter lambda = (evar + fvar)/psill is optimized. The optimal psill and noise variance are then determined.

The lower argument should be a named list with the names of the parameters you wish to set lower bounds for. If not specified, an attempt is made to specify reasonable lower bounds. The current choices are r = 0.001, psill = 0.001, lambda = 0, angle = 0, ratio = 0.001, par3 = 0.001.

The upper argument should be a named list with the names of the parameters you wish to set upper bounds for. If not specified, an attempt is made to specify reasonable upper bounds. The current choices are r = 5 * maximum intercentroid distance, psill = 5 * var(object$y), lambda = 5, angle = 179.99, ratio = 1, par3 = 3.

Author(s)

Joshua French

See Also

cmod_std, optimx

Examples

```
data(toydata, package = "gear")
# setup standard covariance model
mod_std = cmod_std("exponential", psill = 1, r = 1, evar = 0.1)
# setup  dataframe with data
# fit Std geolm
object = geolm(y ~ x1 + x2, data = toydata, mod = mod_std,
                  coordnames = c("x1", "x2"))
```

```
est_object = estimate(object, control = list(trace = 1),
                      verbose = TRUE,
                      lower = list(r = 0.05, lambda = 0.05))
```

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

Description

`eval.cmod` evaluates the covariance of a model based on the provided arguments. This function will be deprecated in the future. Please use the `evaluate` function.

Usage

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

Arguments

`mod`	A covariance or semivariance model.
`d`	An $n \times m$ matrix of distances.
`coords`	Not used.

Value

Returns the evaluated model with necessary components needed for `estimate` and `predict`.

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)
```

`evaluate.cmodStd` *Evaluate spatial dependence model*

Description

`evaluate` evaluates the spatial dependence model based on the provided arguments.

Usage

```
## S3 method for class 'cmodStd'
evaluate(mod, d, e = TRUE, f = TRUE)

evaluate(mod, d, e = TRUE, f = TRUE)
```

Arguments

`mod`	A covariance or semivariogram model.
`d`	An $n \times m$ matrix of distances. If `mod$ratio != 1`, i.e., if geometric anisotropy has been specified, then `d` must be produced by the `ganiso_d` function.
`e`	A single logical value indicating whether the error variance should be added to the returned covariance matrix. Default is `TRUE`.
`f`	A single logical value indicating whether the finescale/microscale variance should be added to the returned covariance matrix. Default is `TRUE`.

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 `estimate` and `predict`.

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)
evaluate(cmod, d)
```

evgram *Empirical (semi-)variogram*

Description

evgram computes the empirical semivariogram of data based on the specified formula indicating the response and trend. See Details. The variogram is twice the semivariogram. If a trend is specified, then the semivariogram is constructed using the residuals of lm(formula,data).

Usage

```
evgram(
  formula,
  data,
  coordnames = NULL,
  nbins = 10,
  maxd = NULL,
  angle = 0,
  ndir = 1,
  type = "standard",
  npmin = 2,
  longlat = FALSE,
  verbose = TRUE,
  invert = TRUE
)
```

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.
coordnames	The columns of data containing the spatial coordinates, provided as a formula (e.g., ~ x + y), column numbers (e.g., c(1,2)), or column names (e.g., c("x","y")). The default is NULL.
nbins	The number of bins (tolerance regions) to use when estimating the empirical 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 empirical semivariogram. The default is 1, meaning calculate an omnidirectional 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.
verbose	Logical value indicating whether computation information should be printed. Default is TRUE.
invert	A logical value indicating whether the axes of the coordinates should be inverted (i.e., the x- and y-axis are switched). The default is TRUE to mimic results from other geostatistical R packages like gstat, geoR, and other software like GSLIB and GeoEAS. Set to FALSE to use the typical x- and y-axes.

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). If invert = TRUE, the directions should be the same.

Computing the empirical semivariogram for the residuals of lm(response ~ 1) will produce identical results to simply computing the empirical semivariogram from the original response. However, if a trend is specified (the righthand side of ~ has non-trival covariates), then the empirical 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 an evgram.

Author(s)

Joshua French

Examples

```
data(co)
v = evgram(Al ~ 1, co, ~ easting + northing)
plot(v)
v2 = evgram(Al ~ 1, co, c("easting", "northing"), angle = 22.5, ndir = 4)
plot(v2)
```

fitted.geolm *Extract fitted values from a* geolm *object*

Description

Extract the fitted values, i.e., the estimated mean values, for an object produced by the geolm function for a specified set of covariates, x. If x is NULL, then then x is taken from object.

Usage

```
## S3 method for class 'geolm'
fitted(object, x, ...)
```

Arguments

object	An object produced by the geolm function.
x	A $m \times p$ matrix of covariates for the locations where fitted values are desired. If NULL, object$x is used.
...	Not currently implemented.

Details

If the object has a known mean, i.e., object$mu is not NULL, then the function returns the vector rep(object$mu,m). If object has estimated coefficients, then x %*% object$coeff is returned.

If x is missing, then object$x is used for x. Naturally, ncol(x) must equal length(object$coeff). If x is NULL and object$mu is not NULL, then m is taken to be 1.

Value

The vector of fitted values.

Author(s)

Joshua French

See Also

fitted

Examples

```
data = data.frame(y = rnorm(10), x1 = runif(10),
                  x2 = runif(10))
d = as.matrix(dist(data[,c("x1", "x2")]))
mod = cmod_man(v = exp(-d), evar = 1)
gearmod = geolm(y ~ x1, data = data,
                coordnames = ~ x1 + x2, mod = mod)
# fitted values for original observations
```

```
fitted(gearmod)
# fitted values for new observations
fitted(gearmod, x = cbind(1, rnorm(20)))
```

`ganiso_d` *Anisotropic distance-related characteristics*

Description

Computes necessary distance-related characteristics when there is geometric anisotropy. This is essentially an internal function to `evaluate` a `cmodStd` object produced by `cmod_std` when the anisotropy ratio differs from 1.

Usage

```
ganiso_d(coords1, coords2, radians = TRUE, invert = TRUE)
```

Arguments

`coords1`	An $N \times 2$ matrix of spatial coordinates.
`coords2`	An $M \times 2$ matrix of spatial coordinates. Is missing, then `coords2 = coords1`.
`radians`	A logical value indicating whether the angles returned should be in degrees or radians. The default is `FALSE`, indicating that the returned angles are in degrees.
`invert`	A logical value indicating whether the axes of the coordinates should be inverted (i.e., the x- and y-axis are switched). The default is `TRUE` to mimic results from other geostatistical R packages like `gstat`, `geoR`, and other software like `GSLIB` and `GeoEAS`. Set to `FALSE` to use the typical x- and y-axes.

Value

A `ganisoD` object with components `d` and `angles`, which is the distance matrix between the coordinates and the angles between the coordinates. The angles are returned in radians.

Examples

```
ganiso_d(cbind(0, 0), cbind(1, 1))
```

`gear` *gear*

Description

*Ge*ostatistical *A*nalysis in *R*.

geardf *Construct a* geardf

Description

geardf constructs a geardf, which is simply a data.frame with an attribute indicating which columns refer to the coordinates. The function either combines x and coords if coordnames isn't provided, or identifies the columns of x to which coordnames refers. The geardf class is only provided to make plotting results of the predict.geolm* functions simple without depending on the sp or sf packages. See plot.geardf for easy plotting.

Usage

```
geardf(data, coords, coordnames)
```

Arguments

data	A data.frame
coords	A data.frame or matrix with column names
coordnames	The columns of data containing the spatial coordinates, provided as a formula (e.g., ~ x + y), column numbers (e.g., c(1,2)), or column names (e.g., c("x","y")). The default is NULL.

Value

A geardf

See Also

plot.geardf

Examples

```
dtf = data.frame(a = 1:2, b = 3:4)

# create geardf with matrix coords (note column names)
coords = matrix(rnorm(4), ncol = 2)
colnames(coords) = c("u", "v")
geardf(dtf, coords)

# create geardf with data.frame coords
coords = as.data.frame(coords)
geardf(dtf, coords)

# create geardf using coordnames
dtf2 = cbind(dtf, u = rnorm(2), v = rnorm(2))
# vector form of coordnames
geardf(dtf2, coordnames = c("u", "v"))
```

```
# formula form of coordnames
geardf(dtf2, coordnames = ~ u + v)
# column index forum of coordnames
geardf(dtf2, coordnames = 3:4)

gdf = geardf(dtf2, coordnames = 3:4)
# looks like a data.frame
gdf
# but slightly more complicated
class(gdf)
attr(gdf, "coordnames")
```

`geodist` *Compute distance for geographic coordinates*

Description

`geodist` computes the distance between the coordinates in `coords1` and `coords2`. If `coords2` isn't supplied, then the distances are computed between the coordinates in `coords1` alone. Otherwise, the pairwise distances between then points in `coords1` and `coords2` is computed. If `longlat = TRUE`, then the great circle distance is computed.

Usage

```
geodist(coords1, coords2, longlat = FALSE)
```

Arguments

`coords1`	A two-dimensional matrix of coordinates.
`coords2`	A two-dimensional matrix of coordinates.
`longlat`	A logical value indicating whether Euclidean distance (`longlat = FALSE`) or great circle distance (`longlat = FALSE`) should be computed. The default is `longlat = FALSE`.

Details

The algorithm used when `longlat = TRUE` is a C++ port of the C code written by Roger Bivand for `spDists`, which appears to be based on a special case of the Vincenty formula with a slight correction based on the WGS84 flattening constant. See https://en.wikipedia.org/wiki/Great-circle_distance.

Value

A matrix of distances

Examples

```
coords = matrix(runif(10), ncol = 2)
d = geodist(coords)
all.equal(d, as.matrix(dist(coords)), check.attributes = FALSE)
```

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,
  mod,
  weights = NULL,
  mu = NULL,
  longlat = NULL,
  cmod = NULL
)
```

Arguments

formula	An object of class formula providing a symbolic description of the model to be fitted. See Details of this function and lm.
data	A data frame containing the response, covariates, and location coordinates.
coordnames	The columns of data containing the spatial coordinates, provided as a formula (e.g., ~ x + y), column numbers (e.g., c(1,2)), or column names (e.g., c("x","y"))
mod	A model object produced by one of the cmod_* functions, e.g., cmod_std.
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.
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.
longlat	A logical indicating whether Euclidean (FALSE) or Great Circle distance (WGS84 ellipsoid) (longlat = TRUE) should be used. Default is FALSE.
cmod	Retained for backwards compatibility. A model object produced by one of the cmod_* functions, e.g., cmod_std.

Details

Note: for the multiresolution Gaussian process model, if `cmod$est == "f"` (i.e., if the nugget is finescale instead of measurement error), then the `weights` argument is internally set to `rep(1,n)`, where `n` is the number of observations.

`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, where `*` depends on `mod`.

Author(s)

Joshua French

Examples

```
data = data.frame(y = rnorm(10), x1 = runif(10),
                  x2 = runif(10))
d = geodist(data[,c("x1", "x2")])
mod = cmod_man(v = exp(-d), evar = 1)
gearmod = geolm(y ~ x1, data = data,
                coordnames = ~ x1 + x2, mod = mod)
```

---

geolm_fit *Fit a* geolm

Description

`geolm_fit` fits a `geolm` based on the specified `mod`. This is effectively an internal function.

Usage

```
geolm_fit(
  mod,
  x,
  y,
  coords,
  mu,
  weights,
  formula,
  coordnames,
  n,
  call,
  coeff_names
)
```

Arguments

mod	A model object produced by one of the cmod_* functions, e.g., cmod_std.
x	The matrix of covariates.
y	The vector of observed responses.
coords	The coordinates of the observed data set.
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.
weights	A vector that is proportional to the reciprocal variances of the errors.
formula	An object of class formula providing a symbolic description of the model to be fitted. See Details of this function and lm.
coordnames	A vector of length 2 with the names of the columns in data containing the coordinates, e.g., c("long","lat").
n	The number of observations.
call	The match.call for the geolm.
coeff_names	A character string with the variable names associated with the coefficents.

Value

Returns a geolm with necessary components needed for estimate and predict.

Author(s)

Joshua French

Examples

```
# no examples since you shouldn't be using this function!
```

---

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

Description

mle estimates the parameters of a geostatistical linear model.

mle estimates the parameters of a geostatistical linear model. The mle function will be deprecated in the future. Please update your code to use the estimate function.

Usage

```
mle(
  object,
  reml = FALSE,
  est = "e",
  lower = NULL,
  upper = NULL,
  method = "nlminb",
  itnmax = NULL,
  control = list(),
  ...
)

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.
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.
...	Currently unimplemented.

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

Joshua French

See Also

estimate,

Examples

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

---

ploglik_xycholv *Compute the log-likelihood of a model*

Description

ll_xycholv computes the log-likelihood of multivariate normal data using components typically found in a geolm. For ploglik_xycholv, cholv is the cholesky decomposition of the covariance matrix for the observed data after dividing the matrix by the (estimated) psill. See the examples below. Depending on parameter choices, the function can return the log-likelihood, the restricted log-likelihood, -2 times the log-likelihood or restricted log-likelihood, or the estimated partial sill for both a maximum likelihood and restricted maximum likelihood setting. This is intended to be an internal function, so minimal error checking is done.

Usage

```
ploglik_xycholv(
  x,
  y,
  cholv,
  mu = NULL,
```

```
  reml = FALSE,
  minus2 = TRUE,
  return_ll = TRUE
)

ll_xycholv(
  x,
  y,
  cholv,
  mu = NULL,
  reml = FALSE,
  minus2 = TRUE,
  return_ll = TRUE
)
```

Arguments

x	The matrix of covariates.
y	The vector of observed responses.
cholv	The cholesky decomposition of the covariance matrix of y (or of that matrix divided by psill for loglik_xycholv.
mu	A single numeric value indicating the assumed mean of the underlying process.
reml	A logical value. Should the Restricted Maximum Likelihood be returned. The default is FALSE.
minus2	A logical value. Should -2 times the log-likelihood be returned. The default is TRUE.
return_ll	A logical value. Should the log-liklihood be returned? Default is TRUE. If FALSE, the estimated partial sill is returned.

Value

A likelihood value, -2 times the likelihood value, or the estimated partial sill, depending on the user's argument choices.

References

Statistical Methods for Spatial Data Analysis. Oliver Schabenberger and Carol A. Gotway (Chapman & Hall/CRC Press) 2005. pp. 259-263

Examples

```
y = rnorm(10)
x = matrix(rep(1, length(y)))
coords = matrix(runif(length(y) * 2), ncol = 2)
d = as.matrix(dist(coords))
pv = exp(-d/3) + 0.1 * diag(length(y))
est_psill = ploglik_xycholv(x, y, chol(pv), return_ll = FALSE)
v = pv * est_psill
```

```
# same result
ploglik_xycholv(x, y, chol(pv), minus2 = FALSE)
ll_xycholv(x, y, chol(v), minus2 = FALSE)
```

---

plot.evgram *Plot* evgram *object*

Description

Plots evgram object produced by evgram using the plot function.

Usage

```
## S3 method for class 'evgram'
plot(x, ..., split = FALSE, add_legend = TRUE, args_legend = list())
```

Arguments

x	An evgram object produced by the evgram function.
...	Additional arguments to pass the plot 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.
add_legend	A logical value indicating whether a legend should be included for a plot of directional semivariograms when split = FALSE.
args_legend	A list with arguments for legend.

Details

The arguments after ... are only considered if a directional semivariogram is provided, i.e., when x$ndir != 1.

split will split directional semivariograms into different plots automatically using autosize.

add_legend and args_legend are only used for a directional semivariogram when split = FALSE

Author(s)

Joshua French

See Also

xyplot, evgram

Examples

```
data(co)
# omnidirectional example
v = evgram(Al ~ 1, co, ~ easting + northing)
plot(v)
plot(v, main = "semivariogram of Al")

# directional semivariograms overlaid
v2 = evgram(Al ~ 1, co, ~ easting + northing,
            angle = 22.5, ndir = 4)
plot(v2)
plot(v2, ylab = "semi-variance", pch = 2:5, type = "p")
plot(v2, lty = 2:5, type = "l",
     args_legend = list(x = "bottomright",
                        legend = c("22.5", "67.5", "112.5", "157.5")))
# directional semivariograms split
plot(v2, split = TRUE)
plot(v2, split = TRUE, col = 2, pch = 3, type = "b",
     main = c("(a)", "(b)", "(c)", "(d)"))
```

`plot.geardf` *Plot* `geardf` *object*

Description

Plot a `geardf` object produced by the `geardf` or `predict.geolm*` functions. See the `autopoints` and `autoimage` functions for advanced options.

Usage

```
## S3 method for class 'geardf'
plot(x, zcol = names(x), interp = FALSE, common.legend = FALSE, ...)
```

Arguments

`x`	A `geardf` object produced by the `geardf` or `predict.geolm*` functions.
`zcol`	The names of the columns of `x` to plot. The coordinate columns are automatically stripped from `zcol`.
`interp`	A logical value indicating whether the values should be interpolated onto a grid. If `FALSE`, then the `autopoints` function is used to construct a heated scatterplot. If `TRUE`, then the `autoimage` function is used to create a heat map. The default is `FALSE`.
`common.legend`	A logical value indicating whether a common legend should be used for the scatterplots/images. The default is `FALSE`.
`...`	Additional arguments to passed to the `autopoints` or `autoimage` functions.

Author(s)

Joshua French

See Also

autopoints, autoimage

Examples

```
data(toydata)
# newdata must have columns with prediction coordinates
newdata = data.frame(x1 = runif(10), x2 = runif(10))

# specify a standard covariance model
mod = cmod_std(model = "exponential", psill = 1, r = 1)

# geolm for universal kriging
geolm_uk = geolm(y ~ x1 + x2, data = toydata, mod = mod,
             coordnames = c("x1", "x2"))

# prediction for universal kriging
pred_uk = predict(geolm_uk, newdata, return_type = "geardf")
# heated scatterplot
plot(pred_uk)
# interpolated image of results
plot(pred_uk, interp = TRUE)
# plot only predictions and rmspe with different colors
plot(pred_uk, c("pred", "rmspe"), col = cm.colors(5))
#'plot only predictions with coarser interpolation grid
plot(pred_uk, "pred", interp = TRUE,
     interp.args = list(no.X = 10, no.Y = 10))
```

---

predict.geolm_cmodMan *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 'geolm_cmodMan'
predict(
  object,
  newdata,
  nsim = 0,
  vop,
  vp,
```

```
  return_type = "SpatialPointsDataFrame",
  dmethod = "chol",
  compute_mspe = TRUE,
  sp = NULL,
  ...
)
```

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.
return_type	A character string indicating the type of object that should be returned. The default is "SpatialPointsDataFrame" for easy plotting of results (see Examples). Other options include "data.frame", "geardf", and "sf".
dmethod	The method used to decompose the covariance matrix for conditional simulation. Valid options are "chol", "eigen", and "svd". The default is "chol".
compute_mspe	A logical value indicating whether the mean square prediction error should be calculated. Default is TRUE.
sp	This argument will be deprecated in the future. Please use the return_type argument. A logical value indicating whether to object returned should be of class SpatialPointsDataFrame for easier plotting with the sp package. Default is NULL.
...	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

A data.frame, SpatialPointsDataFrame, geardf, or sf object with the kriging predictions pred, kriging variance/mean-square prediction error (mspe), the root mean-square prediction error mspe (rmspe), and the conditional simulations sim.1, sim.2, etc. sim.1, sim.2, etc.

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 an exponential covariance model
v = psill * exp(-d/r) + (evar + fvar) * diag(10)
mod_man = cmod_man(v = v, evar = evar)

# coordinate names
cnames = c("x1", "x2")

# geolm for universal kriging
gearmod_uk = geolm(y ~ x1 + x2, data = data, mod = mod_man,
                 coordnames = cnames)

# 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[, cnames]
# create vop and vp using distances
dop = geodist(as.matrix(coords), as.matrix(newcoords))
dp = geodist(newcoords)

# manually create cross-covariance and covariance for
# prediction locations
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 cmod_std

# prediction for universal kriging, with conditional simulation
mod_std = cmod_std("exponential", psill = psill, r = r,
                   evar = evar, fvar = fvar)
gearmod_uk2 = geolm(y ~ x1 + x2, data = data, mod = mod_std,
```

```
                    coordnames = c("x1", "x2"))
pred_uk_std = predict(gearmod_uk2, newdata, nsim = 2, dmethod = "svd")

# compare results
all.equal(pred_uk_man$pred, pred_uk_std$pred)
all.equal(pred_uk_man$mspe, pred_uk_std$mspe)
```

`predict.geolm_cmodStd` *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 'geolm_cmodStd'
predict(
  object,
  newdata,
  nsim = 0,
  return_type = "SpatialPointsDataFrame",
  dmethod = "chol",
  compute_mspe = TRUE,
  sp = NULL,
  ...
)
```

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.
`return_type`	A character string indicating the type of object that should be returned. The default is `"SpatialPointsDataFrame"` for easy plotting of results (see Examples). Other options include `"data.frame"`, `"geardf"`, and `"sf"`.
`dmethod`	The method used to decompose the covariance matrix for conditional simulation. Valid options are `"chol"`, `"eigen"`, and `"svd"`. The default is `"chol"`.
`compute_mspe`	A logical value indicating whether the mean square prediction error should be calculated. Default is `TRUE`.
`sp`	This argument will be deprecated in the future. Please use the `return_type` argument. A logical value indicating whether to object returned should be of class `SpatialPointsDataFrame` for easier plotting with the `sp` package. Default is `NULL`.
`...`	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

A data.frame, SpatialPointsDataFrame, geardf, or sf object with the kriging predictions pred, kriging variance/mean-square prediction error (mspe), the root mean-square prediction error mspe (rmspe), and the conditional simulations sim.1, sim.2, etc. sim.1, sim.2, etc.

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
mod = cmod_std(model = "exponential", psill = 1, r = 1)

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

# demonstrate plotting abilities if return_type == "geardf"
 pred_geardf = predict(gearmod_uk, newdata,
               return_type = "geardf")
 plot(pred_geardf, "pred")
 plot(pred_geardf, interp = TRUE)

# demonstrate plotting abilities if sp package installed
if (requireNamespace("sp", quietly = TRUE)) {
 pred_spdf = predict(gearmod_uk, newdata,
             return_type = "SpatialPointsDataFrame")
 sp::spplot(pred_spdf, "pred")
 sp::spplot(pred_spdf)
}
# demonstrate plotting abilities if sf package installed
```

```
if (requireNamespace("sf", quietly = TRUE)) {
 pred_sfdf = predict(gearmod_uk, newdata,
             return_type = "sf")
 plot(pred_sfdf["pred"])
 plot(pred_sfdf)
}

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

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

`print.evgram` *Print* `evgram` *object*

Description

Print an object of class `evgram` produced by the `evgram` function.

Usage

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

Arguments

`x`	An `evgram` object produced by the `evgram` function.
`...`	Not currently implemented.

Author(s)

Joshua French

Examples

```
data(co)
evgram(Al ~ 1, co, ~ easting + northing)
```

`residuals.geolm` *Extract residuals from a* `geolm` *object*

Description

Extract the residuals for an `object` produced by the `geolm`.

Usage

```
## S3 method for class 'geolm'
residuals(object, ...)
```

Arguments

`object`	An object produced by the `geolm` function.
`...`	Not currently implemented.

Value

The vector of residuals.

Author(s)

Joshua French

See Also

`residuals`

Examples

```
data = data.frame(y = rnorm(10), x1 = runif(10),
                  x2 = runif(10))
d = as.matrix(dist(data[,c("x1", "x2")]))
mod = cmod_man(v = exp(-d), evar = 1)
gearmod = geolm(y ~ x1, data = data,
                coordnames = ~ x1 + x2, mod = mod)
# fitted values for original observations
residuals(gearmod)
```

solve_chol *Solve using cholesky decomposition*

Description

solve_chol solves a system of equations using the cholesky decomposition of a positive definite matrix A, i.e., using a = chol(A).

Usage

```
solve_chol(a, b, ...)

## S3 method for class 'chol'
solve(a, b, ...)
```

Arguments

a	The cholesky decomposition of a positive definite matrix.
b	A numeric or complex vector or matrix giving the right-hand side(s) of the linear system. If missing, b is taken to be an identity matrix and solve will return the inverse of a.
...	Currently unused.

Details

Note: Unless you have good reason to suspect that the cholesky decomposition of your matrix will be stable, it is recommended that you use solve or perform the solve using the qr decomposition.

Author(s)

Joshua French

See Also

solve, chol, qr, solve.qr

Examples

```
set.seed(10)
# create positive definite matrix a
a = crossprod(matrix(rnorm(25^2), nrow = 25))
# create vector x and matrix b
# x can be used to check the stability of the solution
x = matrix(rnorm(25))
b = a %*% x

# standard solve
x1 = solve(a, b)
```

```
all.equal(x, x1)

# solve using cholesky decomposition
chola = chol(a)
x2 = solve_chol(chola, b)
all.equal(x, x2)

# solve using qr decomposition
qra = qr(a)
x3 = solve.qr(qra, b)
all.equal(x, x3)

# compare direct inversion
ai1 = solve(a)
ai2 = solve_chol(chola) #using cholesky decomposition
all.equal(ai1, ai2) # should be TRUE
```

`toydata` *A toy data set for examples*

Description

This is a toy data set of 25 observations generated on a [0, 1] x [0, 1] domain.

Usage

```
data(toydata)
```

Format

A data frame with 25 rows and 3 columns:

y response

x1 coordinate dimension 1 values

x2 coordinate dimension 2 values

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

Description

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

Usage

```
## S3 method for class 'geolm'
update(object, mod, ...)

## S3 method for class 'geolm_cmodMan'
update(object, mod, ...)

## S3 method for class 'geolm_cmodStd'
update(object, mod, ...)
```

Arguments

object	An object produced by the geolm function.
mod	A spatial dependence model object obtained from one of the cmod_* functions or the estimate function.
...	Not implemented.

Value

Returns an object of the same class as object.

Author(s)

Joshua French

See Also

update

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, mod = cmod_std,
              coordnames = c("x1", "x2"))

gear2 = geolm(y ~ x1 + x2, data = data, mod = cmod_std2,
              coordnames = c("x1", "x2"))
gear2b = update(gear1, cmod_std2)
gear2$call = NULL
gear2b$call = NULL
identical(gear2, gear2b)
```

---

vgram *Empirical variogram*

Description

vgram calculates an empirical variogram. Note that, by convention, the empirical variogram actually estimates the semivariogram, not the theoretical variogram (which is twice the semivariogram).

Usage

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

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.

coordnames	The columns of data containing the spatial coordinates, provided as a formula (e.g., ~ x + y), column numbers (e.g., c(1, 2)), or column names (e.g., c("x", "y")). The default is NULL, but this must be specified if data is of class data.frame.
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.
verbose	Print computation information. Default is TRUE.
coords	A deprecated argument.

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 empirical variogram 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 empirical variogram 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 an evgram object with components:

Author(s)

Joshua French

Examples

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

`xyplot.evgram` *Plot* `evgram` *object*

Description

Plots `evgram` object produced by the `evgram` function using the `xyplot` function. Note: the `lattice` package must be loaded (i.e., `library(lattice)` or `lattice::xyplot` must be specifically called for this function to work. See Examples.

Usage

```
xyplot.evgram(x, ..., split = FALSE)
```

Arguments

`x`	An `evgram` object produced by the `evgram` function.
`...`	Additional arguments to pass the `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`, `evgram`

Examples

```
data(co)
v = evgram(Al ~ 1, co, ~ easting + northing)
if (requireNamespace("lattice")) {
   lattice::xyplot(v)
}
v2 = evgram(Al ~ 1, co, ~ easting + northing, angle = 22.5, ndir = 4)
if (requireNamespace("lattice")) {
   lattice::xyplot(v2)
   # show how attributes can be changed using different
   # arguments available in lattice::xyplot.
```

```
    lattice::xyplot(v2, col = 2:5)
    lattice::xyplot(v2, col = 2:5, pch = 1:4)
    lattice::xyplot(v2, col = 2:5, pch = 1:4, lty = 2:5, type = "b")
    lattice::xyplot(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)))
    lattice::xyplot(v2, split = TRUE)
}
```

Index

angle2d, 2
autoimage, *30, 31*
autoplot, *3*
autoplot.evgram, 3
autopoints, *30, 31*
autosize, *29*

chol, *38*
cmod.man, 4
cmod.std, 5
cmod_man, *4*, 6
cmod_std, *5*, 7, *14*, *20*, *23*, *25*
co, 9

data.frame, *32*, *34, 35*
decomp.cov *(*decomp_cov*)*, 11
decomp_cov, 11

estimate, 12, *15, 16*, *25*, *27*
estimate.geolm_cmodStd, *12*, 12
eval.cmod, 15
evaluate, *15*, *20*
evaluate *(*evaluate.cmodStd*)*, 16
evaluate.cmodStd, 16
evgram, *3*, 17, *29*, *36*, *43*

fitted, *19*
fitted.geolm, 19
formula, *23*, *25*

ganiso_d, *16*, 20
gear, 20
geardf, 21, *30*, *32*, *34, 35*
geodist, 22
geolm, *12*, *19*, 23, *37*
geolm_fit, 24

legend, *29*
ll_xycholv *(*ploglik_xycholv*)*, 27
lm, *23*, *25*

match.call, *25*
mle, 25

optimx, *13, 14*, *26*

ploglik_xycholv, 27
plot, *29*
plot.evgram, 29
plot.geardf, *21*, 30
predict, *15, 16*, *25*
predict.geolm_cmodMan, 31
predict.geolm_cmodStd, 34
print.evgram, 36

qr, *38*

residuals, *37*
residuals.geolm, 37

sf, *32*, *34, 35*
solve, *38*
solve.chol *(*solve_chol*)*, 38
solve.qr, *38*
solve_chol, 38
SpatialPointsDataFrame, *32*, *34, 35*
spDists, *22*

toydata, 39

update, *40*
update.geolm, 39
update.geolm_cmodMan *(*update.geolm*)*, 39
update.geolm_cmodStd *(*update.geolm*)*, 39

vgram, 41

xyplot, *29*, *43*
xyplot.evgram, 43