

Package ‘georob’

June 4, 2021

Type Package

Title Robust Geostatistical Analysis of Spatial Data

Version 0.3-14

Date 2021-06-04

Depends R(>= 2.14.0), graphics, parallel, snowfall, sp(>= 0.9-60), stats, utils

Imports abind, constrainedKriging(>= 0.2-1), fields, lmtest, methods, nlme, nleqslv, quantreg, RandomFields(>= 3.3.6), robustbase(>= 0.90-2)

Suggests gstat, multcomp, lattice

Description Provides functions for efficiently fitting linear models with spatially correlated errors by robust and Gaussian (Restricted) Maximum Likelihood and for computing robust and customary point and block external-drift Kriging predictions, along with utility functions for variogram modelling in ad hoc geostatistical analyses, model building, model evaluation by cross-validation, (conditional) simulation of Gaussian processes, unbiased back-transformation of Kriging predictions of log-transformed data.

License GPL (>= 2) | LGPL (>= 2)

NeedsCompilation no

Author Andreas Papritz [cre, aut],
Cornelia Schwierz [ctb]

Maintainer Andreas Papritz <andreas.papritz@env.ethz.ch>

Repository CRAN

Date/Publication 2021-06-04 09:30:02 UTC

R topics documented:

compress	2
control.georob	3
cv	9

cv.georob	10
default.aniso	15
elevation	16
fit.variogram.model	17
georob	23
georobModelBuilding	30
georobObject	34
georobPackage	37
georobS3methods	41
georobSimulation	45
lgpp	49
param.names	54
plot.georob	55
pmm	58
predict.georob	60
profilelogLik	64
sample.variogram	66
validate.predictions	71
wolfcamp	75

Index	77
--------------	-----------

compress

Compact Storage of Symmetric and Triangular Matrices

Description

The utility function `compress` stores symmetric or triangular matrices compactly by retaining only the diagonal and either the lower or upper off-diagonal elements. The function `expand` restores such compressed matrices again to a square form.

Usage

`compress(m)`

`expand(object)`

Arguments

<code>m</code>	either a single symmetric, lower or upper triangular matrix or a list of such matrices. The type of <code>m</code> (or of its component matrices) must be defined by the attribute <code>struc</code> with possible values "sym" (symmetric), "lt" (lower triangular) or "ut" (upper triangular).
<code>object</code>	a single compressed matrix or a list of such matrices generated by <code>compress</code> , see <i>Value</i> . The type of <code>object</code> (or of its components) must be defined by the attribute <code>struc</code> with possible values "sym" (symmetric), "lt" (lower triangular) or "ut" (upper triangular).

Value

If `m` is a single square matrix then `compress` generates a compressed matrix, which is a list with two components:

`diag` a vector with the diagonal elements of `m`.
`tri` a vector with non-redundant off-diagonal elements.

If `m` is a list of square matrices then the result is also a list of compressed matrices.

`expand` creates a square matrix if `object` is a list with components `diag` and `tri` and a list of square matrices if `object` is a list of such lists. If `m` or `objects` are lists that contain other components than squared or compressed matrices then these additional components are returned unchanged.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

See Also

[georob](#) for (robust) fitting of spatial linear models.

Examples

```
## Not run:

data(meuse)

r.logzn.rob <- georob(log(zinc) ~ sqrt(dist) + ffreq, data = meuse, locations = ~ x + y,
  variogram.model = "RMexp",
  param = c(variance = 0.15, nugget = 0.05, scale = 200),
  tuning.psi = 1)

cov2cor(expand(r.logzn.rob[["cov"]][["cov.betahat"]]))

## End(Not run)
```

control.georob

Control Parameters for georob

Description

This page documents parameters used to control [georob](#). It describes the arguments of the functions `control.georob`, `param.transf`, `fwd.transf`, `dfwd.transf`, `bwd.transf`, `control.rq`, `control.nleqslv`, `control.nlminb` and `control.optim`, which all serve to control the behaviour of [georob](#).

Usage

```

control.georob(ml.method = c("REML", "ML"), reparam = TRUE,
  maximizer = c("nlminb", "optim"), initial.param = TRUE,
  initial.fixef = c("lmrob", "rq", "lm"), bhat = NULL,
  min.rweight = 0.25,
  param.tf = param.transf(), fwd.tf = fwd.transf(),
  deriv.fwd.tf = dfwd.transf(), bwd.tf = bwd.transf(),
  psi.func = c("logistic", "t.dist", "huber"),
  irwls.maxiter = 50,
  irwls.ftol = 1.e-5, force.gradient = FALSE,
  min.condnum = 1.e-12, zero.dist = sqrt(.Machine[["double.eps"]]),
  error.family.estimation = c("gaussian", "long.tailed"),
  error.family.cov.effects = c("gaussian", "long.tailed"),
  error.family.cov.residuals = c("gaussian", "long.tailed"),
  cov.bhat = TRUE, full.cov.bhat = FALSE, cov.betahat = TRUE,
  cov.delta.bhat = TRUE, full.cov.delta.bhat = TRUE,
  cov.delta.bhat.betahat = TRUE,
  cov.ehat = TRUE, full.cov.ehat = FALSE,
  cov.ehat.p.bhat = FALSE, full.cov.ehat.p.bhat = FALSE,
  hessian = TRUE,
  rq = control.rq(), lmrob = lmrob.control(),
  nleqslv = control.nleqslv(),
  optim = control.optim(), nlminb = control.nlminb(),
  pcmp = control.pcmp(), ...)

param.transf(variance = "log", snugget = "log", nugget = "log", scale = "log",
  alpha = c(
    RMaskey = "log", RMdewijsian = "logit2", RMfbm = "logit2", RMgencauchy = "logit2",
    RMgenfbm = "logit2", RMLgd = "identity", RMqexp = "logit1", RMstable = "logit2"
  ),
  beta = c(RMdagum = "logit1", RMgencauchy = "log", RMLgd = "log"),
  delta = "logit1", gamma = c(RMcauchy = "log", RMdagum = "logit1"),
  kappa = "logit3", lambda = "log", mu = "log", nu = "log",
  f1 = "log", f2 = "log", omega = "identity", phi = "identity", zeta = "identity")

fwd.transf(...)

dfwd.transf(...)

bwd.transf(...)

control.rq(tau = 0.5, rq.method = c("br", "fnb", "pfn"),
  rq.alpha = 0.1, ci = FALSE, iid = TRUE,
  interp = TRUE, tcrit = TRUE, rq.beta = 0.99995, eps = 1e-06,
  Mm.factor = 0.8, max.bad.fixup = 3, ...)

control.nleqslv(method = c("Broyden", "Newton"),
  global = c("dbl dog", "pwldog", "qline", "gline", "none"),

```

```
xscalm = c("fixed", "auto"), control = list(ftol = 1e-04), ...)

control.optim(method = c("BFGS", "Nelder-Mead", "CG",
  "L-BFGS-B", "SANN", "Brent"), lower = -Inf, upper = Inf,
  control = list(reltol = 1e-05), ...)

control.nlminb(control = list(rel.tol = 1.e-5), lower = -Inf,
  upper = Inf, ...)
```

Arguments

ml.method	character keyword defining whether non-robust maximum likelihood (ML) or restricted maximum likelihood (REML default) estimates will be computed (ignored if tuning.psi <= tuning.psi.nr).
reparam	logical. If TRUE (default) the re-parametrized variance parameters σ_B^2 , η and ξ are estimated by Gaussian (RE)ML, otherwise the original parameters τ^2 , σ_n^2 and σ^2 (cf. subsection <i>Estimating variance parameters by Gaussian (RE)ML</i> , section <i>Details</i> of <code>georob</code>).
maximizer	character keyword defining whether the Gaussian (restricted) log-likelihood is maximized by <code>nlminb</code> (default) or <code>optim</code> .
initial.param	logical, controlling whether initial values of variogram parameters are computed for solving the estimating equations of the variogram and anisotropy parameters. If <code>initial.param = TRUE</code> (default) robust initial values of parameters are computed by discarding outlying observations based on the “robustness weights” of the initial fit of the regression model by <code>lmrob</code> and fitting the spatial linear model by Gaussian REML to the pruned data set. For <code>initial.param = FALSE</code> no initial parameter values are computed and the estimating equations are solved with the initial values passed by <code>param</code> and <code>aniso</code> to <code>georob</code> (see <i>Details</i> of <code>georob</code>).
initial.fixef	character keyword defining whether the function <code>lmrob</code> or <code>rq</code> is used to compute robust initial estimates of the regression parameters β (default “ <code>lmrob</code> ”). If the fixed effects model matrix has not full columns rank, then <code>lm</code> is used to compute initial values of the regression coefficients.
bhat	initial values for the spatial random effects \hat{B} , with $\hat{B} = \mathbf{0}$ if <code>bhat</code> is equal to NULL (default).
min.rweight	positive numeric. “Robustness weight” of the initial <code>lmrob</code> fit that observations must exceed to be used for computing robust initial estimates of variogram parameters by setting <code>initial.param = TRUE</code> (see <code>georob</code> ; default 0.25).
param.tf	a function such as <code>param.transf</code> , which returns a named list of character strings that define the transformations to be applied to the variogram parameters for model fitting, see <i>Details</i> .
fwd.tf	a function such as <code>fwd.transf</code> , which returns a named list of invertible functions to be used to transform variogram parameters, see <i>Details</i> .
deriv.fwd.tf	a function such as <code>dfwd.transf</code> , which returns a named list of functions corresponding to the first derivatives of <code>fwd.tf</code> , see <i>Details</i> .

bwd.tf	a function such as bwd.transf, which returns the named list of inverse functions corresponding to fwd.tf, see <i>Details</i> .
psi.func	character keyword defining what ψ_c -function should be used for robust model fitting. Possible values are "logistic" (a scaled and shifted logistic CDF, default), "t.dist" (re-descending ψ_c -function associated with Student t -distribution with c degrees of freedom) and "huber" (Huber's ψ_c -function).
irwls.maxiter	positive integer equal to the maximum number of IRWLS iterations to solve the estimating equations of \mathbf{B} and β (default 50).
irwls.ftol	positive numeric convergence criterion for IRWLS. Convergence is assumed if the objective function changes in one IRWLS iteration does not exceed ftol.
force.gradient	logical controlling whether the estimating equations or the gradient of the Gaussian restricted log-likelihood are evaluated even if all variogram parameters are fixed (default FALSE).
min.condnum	positive numeric. Minimum acceptable ratio of smallest to largest singular value of the model matrix \mathbf{X} (default $1.e-12$).
zero.dist	positive numeric equal to the maximum distance, separating two sampling locations that are still considered as being coincident.
error.family.estimation	character keyword, defining the probability distribution for ε (default: "gaussian") that is used to approximate the covariance of $\hat{\mathbf{B}}$ when solving the estimating equations, see <i>Details</i> .
error.family.cov.effects	character keyword, defining the probability distribution for ε (default: "gaussian") that is used to approximate the covariances of $\hat{\beta}$, $\hat{\mathbf{B}}$ and $\mathbf{B} - \hat{\mathbf{B}}$, see <i>Details</i> .
error.family.cov.residuals	character keyword, defining the probability distribution for ε (default: "long.tailed") that is used to approximate the covariances of $\hat{\varepsilon} = \mathbf{Y} - \mathbf{X}\hat{\beta} - \hat{\mathbf{B}}$ and $\hat{\varepsilon} + \hat{\mathbf{B}} = \mathbf{Y} - \mathbf{X}\hat{\beta}$, see <i>Details</i> .
cov.bhat	logical controlling whether the covariances of $\hat{\mathbf{B}}$ are returned by georob (default FALSE).
full.cov.bhat	logical controlling whether the full covariance matrix (TRUE) or only the variance vector of $\hat{\mathbf{B}}$ is returned (default FALSE).
cov.betahat	logical controlling whether the covariance matrix of $\hat{\beta}$ is returned (default TRUE).
cov.delta.bhat	logical controlling whether the covariances of $\mathbf{B} - \hat{\mathbf{B}}$ are returned (default TRUE).
full.cov.delta.bhat	logical controlling whether the full covariance matrix (TRUE) or only the variance vector of $\mathbf{B} - \hat{\mathbf{B}}$ is returned (default TRUE).
cov.delta.bhat.betahat	logical controlling whether the covariance matrix of $\mathbf{B} - \hat{\mathbf{B}}$ and $\hat{\beta}$ is returned (default TRUE).
cov.ehat	logical controlling whether the covariances of $\hat{\varepsilon} = \mathbf{Y} - \mathbf{X}\hat{\beta} - \hat{\mathbf{B}}$ are returned (default TRUE).

full.cov.ehat	logical controlling whether the full covariance matrix (TRUE) or only the variance vector of $\hat{\varepsilon} = \mathbf{Y} - \mathbf{X}\hat{\beta} - \hat{\mathbf{B}}$ is returned (default FALSE).
cov.ehat.p.bhat	logical controlling whether the covariances of $\hat{\varepsilon} + \hat{\mathbf{B}} = \mathbf{Y} - \mathbf{X}\hat{\beta}$ are returned (default FALSE).
full.cov.ehat.p.bhat	logical controlling whether the full covariance matrix (TRUE) or only the variance vector of $\hat{\varepsilon} + \hat{\mathbf{B}} = \mathbf{Y} - \mathbf{X}\hat{\beta}$ is returned (default FALSE).
hessian	logical scalar controlling whether for Gaussian (RE)ML the Hessian should be computed at the MLEs.
rq	a list of arguments passed to rq or a function such as control.rq that generates such a list (see rq for allowed arguments).
lmrob	a list of arguments passed to the control argument of lmrob or a function such as lmrob.control that generates such a list (see lmrob.control for allowed arguments).
nleqslv	a list of arguments passed to nleqslv or a function such as control.nleqslv that generates such a list (see nleqslv for allowed arguments).
nlminb	a list of arguments passed to nlminb or a function such as control.nlminb that generates such a list (see nlminb for allowed arguments).
optim	a list of arguments passed to optim or a function such as control.optim that generates such a list (see optim for allowed arguments).
pcmp	a list of arguments, passed e.g. to pmm or a function such as control.pcmp that generates such a list (see control.pcmp for allowed arguments).
...	for fwd.transf, dfwd.transf and bwd.transf a named vectors of functions, extending the definition of transformations for variogram parameters (see <i>Details</i>).
variance, snugget, nugget, scale, alpha, beta, delta, gamma, kappa, lambda, mu, nu	character strings with names of transformation functions of the variogram parameters.
f1, f2, omega, phi, zeta	character strings with names of transformation functions of the variogram parameters.
tau, rq.method, rq.alpha, ci, iid, interp, tcrit	arguments passed as ... to rq. Note that only "br", "fnn" and "pfn" methods of rq() are currently supported.
rq.beta, eps, Mm.factor, max.bad.fixup	arguments passed as ... to rq.
method, global, xscalm, control, lower, upper, reltol, rel.tol	arguments passed to related arguments of nleqslv, nlminb and optim, respectively.

Details

Parameter transformations:

The arguments `param.tf`, `fwd.tf`, `deriv.fwd.tf`, `bwd.tf` define the transformations of the variogram parameters for RE(ML) estimation. Implemented are currently "log", "logit1", "logit2", "logit3" (various variants of logit-transformation, see code of function `fwd.transf`) and "identity" (= no) transformations. These are the possible values that the many arguments of the function `param.transf` accept (as quoted character strings) and these are the names of the list components returned by `fwd.transf`, `dfwd.transf` and `bwd.transf`. Additional transformations can be implemented by:

1. Extending the function definitions by arguments like


```
fwd.tf = fwd.transf(my.fun = function(x) your transformation),
deriv.fwd.tf = dfwd.transf(my.fun = function(x) your derivative),
bwd.tf = bwd.transf(my.fun = function(x) your back-transformation),
```
2. Assigning to a given argument of `param.transf` the name of the new function, e.g.


```
variance = "my.fun".
```

Note the values given for the arguments of `param.transf` must match the names of the functions returned by `fwd.transf`, `dfwd.transf` and `bwd.transf`.

Approximation of covariances of fixed and random effects and residuals:

The robustified estimating equations of robust REML depend on the covariances of \hat{B} . These covariances (and the covariances of $B - \hat{B}$, $\hat{\beta}$, $\hat{\varepsilon}$, $\hat{\varepsilon} + \hat{B}$) are approximated by expressions that in turn depend on the variances of ε , $\psi(\varepsilon/\tau)$ and the expectation of $\psi'(\varepsilon/\tau)$ ($= \partial/\partial\varepsilon \psi(\varepsilon/\tau)$). The arguments `error.family.estimation`, `error.family.cov.effects` and `error.family.cov.residuals` control what parametric distribution for ε is used to compute the variance of ε , $\psi(\varepsilon/\tau)$ and the expectation of $\psi'(\varepsilon/\tau)$ when

- solving the estimating equations (`error.family.estimation`),
- computing the covariances of $\hat{\beta}$, \hat{B} and $B - \hat{B}$ (`error.family.cov.effects`) and
- computing the covariances of $\hat{\varepsilon} = Y - X\hat{\beta} - \hat{B}$ and $\hat{\varepsilon} + \hat{B} = Y - X\hat{\beta}$ (`error.family.cov.residuals`).

Possible options are: "gaussian" or "long.tailed". In the latter case the PDF of ε is assumed to be proportional to $1/\tau \exp(-\rho(\varepsilon/\tau))$, where $\psi(x) = \rho'(x)$.

Value

`control.georob`, `control.rq`, `control.nleqslv`, `control.optim` and `control.nlminb` all create lists with control parameters passed to `georob`, `rq`, `nleqslv`, `optim`, `nlminb`, respectively. Note that the list returned by `code.georob` contains some components (`irwls.initial`, `tuning.psi.nr`, `cov.bhat.betahat`, `aux.cov.pred.target`) that cannot be changed by the user.

`param.transf` generates a list with character strings that define what transformations are used for estimating the variogram parameters, and `fwd.transf`, `bwd.transf` and `dfwd.transf` return lists of functions with forward and backward transformations and the first derivatives of the forward transformations.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms; [georob](#) for (robust) fitting of spatial linear models;

[georobObject](#) for a description of the class `georob`;

[profilelogLik](#) for computing profiles of Gaussian likelihoods;

[plot.georob](#) for display of RE(ML) variogram estimates;

[georobModelBuilding](#) for stepwise building models of class `georob`;

[cv.georob](#) for assessing the goodness of a fit by `georob`;

[georobMethods](#) for further methods for the class `georob`;

[predict.georob](#) for computing robust Kriging predictions;

[lgnp](#) for unbiased back-transformation of Kriging prediction of log-transformed data;

[georobSimulation](#) for simulating realizations of a Gaussian process from model fitted by `georob`; and finally

[sample.variogram](#) and [fit.variogram.model](#) for robust estimation and modelling of sample variograms.

Examples

```
## Not run:
data(meuse)

r.logzn.rob <- georob(log(zinc) ~ sqrt(dist), data = meuse, locations = ~ x + y,
  variogram.model = "RMexp",
  param = c(variance = 0.15, nugget = 0.05, scale = 200),
  tuning.psi = 1, control = control.georob(cov.bhat = TRUE,
  cov.ehat.p.bhat = TRUE, initial.fixef = "rq"), verbose = 2)

qqnorm(rstandard(r.logzn.rob, level = 0)); abline(0, 1)
qqnorm(ranef(r.logzn.rob, standard = TRUE)); abline(0, 1)

## End(Not run)
```

Description

Generic function for cross-validating models.

Usage

```
cv(object, ...)
```

Arguments

object any model object.
 ... additional arguments as required by the methods.

Value

will depend on the method function used; see the respective documentation.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

See Also

[georob](#) for (robust) fitting of spatial linear models;
[cv.georob](#) for assessing the goodness of a model fitted by georob.

 cv.georob

Cross-Validating a Spatial Linear Model Fitted by georob

Description

This function assesses the goodness-of-fit of a spatial linear model by K -fold cross-validation. In more detail, the model is re-fitted K times by robust (or Gaussian) (RE)ML, excluding each time $1/K$ th of the data. The re-fitted models are used to compute robust (or customary) external Kriging predictions for the omitted observations. If the response variable is log-transformed then the Kriging predictions can be optionally transformed back to the original scale of the measurements. S3methods for evaluating and plotting diagnostic summaries of the cross-validation errors are described for the function [validate.predictions](#).

Usage

```
## S3 method for class 'georob'
cv(object, formula = NULL, subset = NULL,
    method = c("block", "random"), nset = 10L, seed = NULL,
    sets = NULL, duplicates.in.same.set = TRUE, re.estimate = TRUE,
    param = object[["variogram.object"]][[1]][["param"]],
    fit.param = object[["variogram.object"]][[1]][["fit.param"]],
    aniso = object[["variogram.object"]][[1]][["aniso"]],
    fit.aniso = object[["variogram.object"]][[1]][["fit.aniso"]],
    variogram.object = NULL,
    use.fitted.param = TRUE, return.fit = FALSE,
    reduced.output = TRUE, lgn = FALSE,
    mfl.action = c("offset", "stop"),
    ncores = min(nset, detectCores()), verbose = 0, ...)
```

Arguments

object	an object of class of "georob", see georobObject .
formula	an optional formula for the regression model passed by update to georob .
subset	an optional vector specifying a subset of observations to be used in the fitting process.
method	keyword, controlling whether subsets are formed by partitioning data set into blocks by kmeans (default) or randomly. Ignored if sets is non-NULL.
nset	positive integer defining the number K of subsets into which the data set is partitioned (default: $nset = 10$). Ignored if sets is non-NULL.
seed	optional integer seed to initialize random number generation, see set.seed . Ignored if sets is non-NULL.
sets	an optional vector of the same length as the response vector of the fitted model and with positive integers taking values in $(1, 2, \dots, K)$, defining in this way the K subsets into which the data set is split. If sets = NULL (default) the partition is randomly generated by kmeans or runif (using possibly seed).
duplicates.in.same.set	logical controlling whether replicated observations at a given location are assigned to the same subset when partitioning the data (default TRUE).
re.estimate	logical controlling whether the model is re-fitted to the reduced data sets before computing the Kriging predictions (TRUE, default) or whether the model passed in object is used to compute the predictions for the omitted observations, see <i>Details</i> .
param	a named numeric vector or a matrix or data frame with initial values of variogram parameters passed by update to georob . If param is a matrix (or a data frame) then it must have nset rows and <code>length(object[["variogram.object"]][[1]][["param"]])</code> columns with initial values of variogram parameters for the nset cross-validation sets, and <code>colnames(param)</code> must match <code>names(object[["variogram.object"]][[1]][["param"]])</code> .
fit.param	a named logical vector or a matrix or data frame defining which variogram parameters should be adjusted by update . If fit.param is a matrix (or a data frame) then it must have nset rows and <code>length(object[["variogram.object"]][[1]][["fit.param"]])</code> columns with variogram parameter fitting flags for the nset cross-validation sets, and <code>colnames(param)</code> must match <code>names(object[["variogram.object"]][[1]][["fit.param"]])</code> .
aniso	a named numeric vector or a matrix or data frame with initial values of anisotropy parameters passed by update to georob . If aniso is a matrix (or a data frame) then it must have nset rows and <code>length(object[["variogram.object"]][[1]][["aniso"]])</code> columns with initial values of anisotropy parameters for the nset cross-validation sets, and <code>colnames(aniso)</code> must match <code>names(object[["variogram.object"]][[1]][["aniso"]])</code> .
fit.aniso	a named logical vector or a matrix or data frame defining which anisotropy parameters should be adjusted by update . If fit.aniso is a matrix (or a data

frame) then it must have nset rows and `length(object[["variogram.object"]][[1]][["fit.aniso"]])` columns with anisotropy parameter fitting flags for the nset cross-validation sets, and `colnames(param)` must match `names(object[["variogram.object"]][[1]][["fit.aniso"]])`.

`variogram.object`

an optional list that gives initial values of for fitting a possibly nested variogram model for the cross-validation sets. Each component is a list with the following components:

- `param`: an optional named numeric vector or a matrix or data frame with initial values of variogram parameters passed by `update` to `georob`. If `param` is a matrix (or a data frame) then it must have nset rows and `length(object[["variogram.object"]][[i]][["param"]])` columns with initial values of variogram parameters for the nset cross-validation sets (i is the i th variogram structure), and `colnames(param)` must match `names(object[["variogram.object"]][[i]][["param"]])`.
- `fit.param`: an optional named logical vector or a matrix or data frame defining which variogram parameters should be adjusted by `update`. If `fit.param` is a matrix (or a data frame) then it must have nset rows and `length(object[["variogram.object"]][[i]][["fit.param"]])` columns with variogram parameter fitting flags for the nset cross-validation sets (i is the i th variogram structure), and `colnames(param)` must match `names(object[["variogram.object"]][[i]][["fit.param"]])`.
- `aniso`: an optional named numeric vector or a matrix or data frame with initial values of anisotropy parameters passed by `update` to `georob`. If `aniso` is a matrix (or a data frame) then it must have nset rows and `length(object[["variogram.object"]][[i]][["aniso"]])` columns with initial values of anisotropy parameters for the nset cross-validation sets (i is the i th variogram structure), and `colnames(aniso)` must match `names(object[["variogram.object"]][[i]][["aniso"]])`.
- `fit.aniso`: an optional named logical vector or a matrix or data frame defining which anisotropy parameters should be adjusted by `update`. If `fit.aniso` is a matrix (or a data frame) then it must have nset rows and `length(object[["variogram.object"]][[i]][["fit.aniso"]])` columns with anisotropy parameter fitting flags for the nset cross-validation sets (i is the i th variogram structure), and `colnames(param)` must match `names(object[["variogram.object"]][[i]][["fit.aniso"]])`.

`use.fitted.param`

logical scalar controlling whether fitted values of `param` (and `aniso`) are used as initial values when variogram parameters are fitted for the cross-validation sets (default TRUE).

`return.fit`

logical controlling whether information about the fit should be returned when re-estimating the model with the reduced data sets (default FALSE).

`reduced.output`

logical controlling whether the complete fitted model objects, fitted to the reduced data sets, are returned (FALSE) or only some components (TRUE, default, see *Value*). Ignored if `return.fit` = FALSE.

lgn	logical controlling whether Kriging predictions of a log-transformed response should be transformed back to the original scale of the measurements (default FALSE).
mfl.action	character controlling what is done when some levels of factor(s) are not present in any of the subsets used to fit the model. The function either stops ("stop") or treats the respective factors as model offset ("offset", default).
ncores	positive integer controlling how many cores are used for parallelized computations, see <i>Details</i> .
verbose	positive integer controlling logging of diagnostic messages to the console during model fitting. Passed by <code>update</code> to <code>georob</code> .
...	additional arguments passed by <code>update</code> to <code>georob</code> , see <i>Details</i> .

Details

Note that *the data frame passed as data argument to georob must exist in the user workspace when calling cv.georob*.

cv.georob then uses the packages **parallel**, **snow** and **snowfall** for parallelized computations. By default, the function uses K CPUs but not more than are physically available (as returned by `detectCores`).

cv.georob uses the function `update` to re-estimated the model with the reduced data sets. Therefore, any argument accepted by `georob` except data can be changed when re-fitting the model. Some of them (e.g. formula, subset, etc.) are explicit arguments of cv.georob, but also the remaining ones can be passed by ... to the function.

Practitioners in geostatistics commonly cross-validate a fitted model without re-estimating the model parameters with the reduced data sets. This is clearly an unsound practice (see Hastie et al., 2009, sec. 7.10). Therefore, the argument `re.estimate` should always be set to TRUE. The alternative is provided only for historic reasons.

Value

An object of class cv.georob, which is a list with the two components `pred` and `fit`.

`pred` is a data frame with the coordinates and the cross-validation prediction results with the following variables:

subset	an integer vector defining to which of the K subsets an observation was assigned.
data	the values of the (possibly log-transformed) response.
pred	the Kriging predictions.
se	the Kriging standard errors.

If `lgn = TRUE` then `pred` has the additional variables:

lgn.data	the untransformed response.
lgn.pred	the unbiased back-transformed predictions of a log-transformed response.
lgn.se	the Kriging standard errors of the back-transformed predictions of a log-transformed response.

The second component `fit` contains either the full outputs of `georob`, fitted for the K reduced data sets (`reduced.output = FALSE`), or K lists with the components `tuning.psi`, `converged`, `convergence.code`, `gradient`, `variogram.object`, coefficients along with the standard errors of $\hat{\beta}$, see `georobObject`.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

References

Hastie, T., Tibshirani, R. and Friedman, J. (2009) *The Elements of Statistical Learning; Data Mining, Inference and Prediction*. New York: Springer-Verlag.

See Also

`georobIntro` for a description of the model and a brief summary of the algorithms;
`georob` for (robust) fitting of spatial linear models;
`georobObject` for a description of the class `georob`;
`profilelogLik` for computing profiles of Gaussian likelihoods;
`plot.georob` for display of RE(ML) variogram estimates;
`control.georob` for controlling the behaviour of `georob`;
`georobModelBuilding` for stepwise building models of class `georob`;
`georobMethods` for further methods for the class `georob`;
`predict.georob` for computing robust Kriging predictions;
`validate.predictions` for validating Kriging predictions;
`lgnpp` for unbiased back-transformation of Kriging prediction of log-transformed data;
`georobSimulation` for simulating realizations of a Gaussian process from model fitted by `georob`;
and finally
`sample.variogram` and `fit.variogram.model` for robust estimation and modelling of sample variograms.

Examples

```
## Not run:
data(meuse)

r.logzn <- georob(log(zinc) ~ sqrt(dist), data = meuse, locations = ~ x + y,
  variogram.model = "RMexp",
  param = c(variance = 0.15, nugget = 0.05, scale = 200),
  tuning.psi = 1)

r.logzn.cv.1 <- cv(r.logzn, seed = 1, lgn = TRUE, ncores = 1, verbose = 1)
## using multiple cores
## r.logzn.cv.1 <- cv(r.logzn, seed = 1, lgn = TRUE)
r.logzn.cv.2 <- cv(r.logzn, formula = .~. + ffreq, seed = 1, lgn = TRUE,
```

```

    ncores = 1, verbose = 1)
## using multiple cores
## r.logzn.cv.2 <- cv(r.logzn, formula = .~. + ffreq, seed = 1, lgn = TRUE,
##    ncores = 1, verbose = 1)

plot(r.logzn.cv.1, type = "bs")
plot(r.logzn.cv.2, type = "bs", add = TRUE, col = "red")

legend("topright", lty = 1, col = c("black", "red"), bty = "n",
      legend = c("log(Zn) ~ sqrt(dist)", "log(Zn) ~ sqrt(dist) + ffreq"))
## End(Not run)

```

default.aniso *Setting Default Values of Variogram Parameters*

Description

Helper functions to set sensible default values for anisotropy parameters and for controlling what variogram and anisotropy parameters should be estimated.

Usage

```
default.aniso(f1 = 1., f2 = 1., omega = 90., phi = 90., zeta = 0.)
```

```
default.fit.param(
  variance = TRUE, snugget = FALSE, nugget = TRUE, scale = TRUE,
  alpha = FALSE, beta = FALSE, delta = FALSE, gamma = FALSE,
  kappa = FALSE, lambda = FALSE, mu = FALSE, nu = FALSE)

```

```
default.fit.aniso(f1 = FALSE, f2 = FALSE, omega = FALSE,
  phi = FALSE, zeta = FALSE)

```

Arguments

variance	variance (sill σ^2) of the auto-correlated component of the Gaussian random field $B(\mathbf{s})$.
snugget	variance (spatial nugget σ_n^2) of the seemingly spatially uncorrelated component of $B(\mathbf{s})$ (micro-scale spatial variation; default value snugget = 0).
nugget	variance (nugget τ^2) of the independent errors $\varepsilon(\mathbf{s})$.
scale	range parameter (α) of the variogram.
alpha, beta, delta, gamma, kappa, lambda, mu, nu	names of additional variogram parameters such as the smoothness parameter ν of the Whittle-Mat'ern model (see RMmodel and param.names).
f1	positive ratio f_1 of lengths of second and first semi-principal axes of an ellipsoidal surface with constant semi-variance in \mathbb{R}^3 (default f1 = 1).

f2	positive ratio f_2 of lengths of third and first semi-principal axes of the semi-variance ellipsoid (default f2 = 1).
omega	azimuth in degrees of the first semi-principal axis of the semi-variance ellipsoid (default omega = 90).
phi	90 degrees minus altitude of the first semi-principal axis of the semi-variance ellipsoid (default phi = 90).
zeta	angle in degrees between the second semi-principal axis and the direction of the line defined by the intersection between the x - y -plane and the plane orthogonal to the first semi-principal axis of the semi-variance ellipsoid through the origin (default zeta = 0).

Value

Either a named numeric with initial values of anisotropy parameters (`default.aniso`) or named logical vector, controlling what parameters should be estimated (`default.fit.param`, `default.fit.aniso`).

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms;
[georob](#) for (robust) fitting of spatial linear models.

Examples

```
default.aniso(f1 = 0.5, omega = 45)
default.fit.param(scale=FALSE, alpha = TRUE)
default.fit.aniso(f1 = TRUE, omega = TRUE)
```

elevation

Elevation Data

Description

Surface elevation data taken from Davis (1972).

Usage

```
data(elevation)
```


Format

A data frame with 52 observations on the following 3 variables:

x a numeric vector with the easting coordinate in multiples of 50 feet.

y a numeric vector with the northing coordinate in multiples of 50 feet..

height a numeric vector with the elevation in multiples of 10 feet.

Note

The data were imported from the package **geoR**.

Source

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

Examples

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

fit.variogram.model *Fitting Model Functions to Sample Variograms*

Description

The function `fit.variogram.model` fits a variogram model to a sample variogram by (weighted) non-linear least squares. There are `print`, `summary` and `lines` methods for summarizing and displaying fitted variogram models.

Usage

```
fit.variogram.model(sv,
  variogram.model = c("RMexp", "RMaskey", "RMBessel", "RMcauchy",
    "RMcircular", "RMcubic", "RMDagum", "RMDampedcos", "RMDewijsian",
    "RMfbm", "RMgauss", "RMgencauchy", "RMgenfbm", "RMgengneiting",
    "RMgneiting", "RMIgd", "RMmatern", "RMpenta", "RMqexp",
    "RMspheric", "RMstable", "RMwave", "RMwhittle"),
  param, fit.param = default.fit.param()[names(param)],
  aniso = default.aniso(), fit.aniso = default.fit.aniso(),
  variogram.object = NULL,
  max.lag = max(sv[["lag.dist"]]), min.npairs = 30,
  weighting.method = c("cressie", "equal", "npairs"),
  control = control.fit.variogram.model(),
  verbose = 0)

control.fit.variogram.model(maximizer = c("nlminb", "optim"),
  param.tf = param.transf(), fwd.tf = fwd.transf()),
```

```

deriv.fwd.tf = dfwd.transf(), bwd.tf = bwd.transf(),
hessian = TRUE, optim = control.optim(), nlminb = control.nlminb())

## S3 method for class 'fitted.variogram'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'fitted.variogram'
summary(object, correlation = FALSE, signif = 0.95, ...)

## S3 method for class 'fitted.variogram'
lines(x, what = c("variogram", "covariance", "correlation"),
      from = 1.e-6, to, n = 501, xy.angle = 90, xz.angle = 90,
      col = 1:length(xy.angle), pch = 1:length(xz.angle), lty = "solid", ...)

```

Arguments

- sv** an object of class `sample.variogram`, see [sample.variogram](#).
- variogram.model** a character keyword defining the variogram model to be fitted. Currently, most basic variogram models provided by the package **RandomFields** can be fitted (see *Details* of [georob](#) and [RMmodel](#)).
- param** a named numeric vector with initial values of the variogram parameters. The following parameter names are allowed (see *Details* of [georob](#) and [georobIntro](#) for information about the parametrization of variogram models):
- **variance**: variance (sill σ^2) of the auto-correlated component of the Gaussian random field $B(s)$.
 - **snugget**: variance (spatial nugget σ_n^2) of the seemingly spatially uncorrelated component of $B(s)$ (micro-scale spatial variation; default value `snugget = 0`).
 - **nugget**: variance (nugget τ^2) of the independent errors $\varepsilon(s)$.
 - **scale**: range parameter (α) of the variogram.
 - **names** of additional variogram parameters such as the smoothness parameter ν of the Whittle-Mat'ern model (see [RMmodel](#) and [param.names](#)).
- fit.param** a named logical vector (or a function such as [default.fit.param](#) that creates this vector) with the same names as used for `param`, defining which parameters are adjusted (TRUE) and which are kept fixed at their initial values (FALSE) when fitting the model.
- aniso** a named numeric vector with initial values (or a function such as [default.aniso](#) that creates this vector) for fitting geometrically anisotropic variogram models. The names of `aniso` are matched against the following names (see *Details* and [georobIntro](#) for information about the parametrization of variogram models):
- **f1**: ratio f_1 of lengths of second and first semi-principal axes of an ellipsoidal surface with constant semi-variance in \mathbb{R}^3 (default `f1 = 1`).
 - **f2**: ratio f_2 of lengths of third and first semi-principal axes of the semi-variance ellipsoid (default `f2 = 1`).

	<ul style="list-style-type: none"> • omega: azimuth in degrees of the first semi-principal axis of the semi-variance ellipsoid (default omega = 90). • phi: 90 degrees minus altitude of the first semi-principal axis of the semi-variance ellipsoid (default phi = 90). • zeta: angle in degrees between the second semi-principal axis and the direction of the line defined by the intersection between the x-y-plane and the plane orthogonal to the first semi-principal axis of the semi-variance ellipsoid through the origin (default zeta = 0).
<code>fit.aniso</code>	a named logical vector (or a function such as <code>default.fit.aniso</code> that creates this vector) with the same names as used for <code>aniso</code> , defining which parameters are adjusted (TRUE) and which are kept fixed at their initial values (FALSE) when fitting the model.
<code>variogram.object</code>	<p>an optional list that defines a possibly nested variogram model. Each component is itself a list with the following components:</p> <ul style="list-style-type: none"> • <code>variogram.model</code>: a character keyword defining the variogram model, see respective argument above. • <code>param</code>: a named numeric vector with initial values of the variogram parameters, see respective argument above. • <code>fit.param</code>: a named logical vector defining which parameters are adjusted, see respective argument above. • <code>aniso</code>: a named numeric vector with initial values for fitting geometrically anisotropic variogram models, see respective argument above. • <code>fit.param</code>: a named logical vector defining which anisotropy parameters are adjusted, see respective argument above. <p>Note that the arguments <code>variogram.model</code>, <code>param</code>, <code>fit.param</code>, <code>aniso</code> and <code>fit.aniso</code> are ignored when <code>variogram.object</code> is passed to <code>fit.variogram.model</code>.</p>
<code>max.lag</code>	a positive numeric defining the maximum lag distance to be used for fitting or plotting variogram models (default all lag classes).
<code>min.npairs</code>	a positive integer defining the minimum number of data pairs required so that a lag class is used for fitting a variogram model (default 30).
<code>weighting.method</code>	<p>a character keyword defining the weights for non-linear least squares. Possible values are:</p> <ul style="list-style-type: none"> • "equal": no weighting , • "npairs": weighting by number of data pairs in a lag class, • "cressie": "Cressie's weights" (default, see Cressie, 1993, sec. 2.6.2).
<code>verbose</code>	positive integer controlling logging of diagnostic messages to the console during model fitting.
<code>control</code>	a list with the components <code>maximizer</code> , <code>param.tf</code> , <code>fwd.tf</code> , <code>bwd.tf</code> , <code>hessian</code> , <code>optim</code> and <code>nlminb</code> or a function such as <code>control.fit.variogram.model</code> that generates such a list. See code control.georob for information on <code>codemaximizer</code> , <code>param.tf</code> , <code>fwd.tf</code> , <code>bwd.tf</code> , <code>hessian</code> , <code>optim</code> and <code>nlminb</code> .

maximizer	character keyword defining the optimizer for nonlinear least squares. Possible values are <code>nlminb</code> (default) or <code>optim</code> .
hessian	logical scalar controlling whether the Hessian should be computed at the nonlinear least squares estimates.
param.tf	a function such as <code>param.transf</code> , which returns a named vector of character strings that define the transformations to be applied to the variogram parameters for model fitting, see <code>control.georob</code> .
fwd.tf	a function such as <code>fwd.transf</code> , which returns a named list of invertible functions to be used to transform variogram parameters, see <code>control.georob</code> .
deriv.fwd.tf	a function such as <code>dfwd.transf</code> , which returns a named list of functions corresponding to the first derivatives of <code>fwd.tf</code> , see <i>Details</i> .
bwd.tf	a function such as <code>bwd.transf</code> , which returns the named list of inverse functions corresponding to <code>fwd.tf</code> , see <code>control.georob</code> .
nlminb	a list of arguments passed to <code>nlminb</code> or a function such as <code>control.nlminb</code> that generates such a list (see <code>nlminb</code> for allowed arguments).
optim	a list of arguments passed to <code>optim</code> or a function such as <code>control.optim</code> that generates such a list (see <code>optim</code> for allowed arguments).
object, x	an object of class <code>fitted.variogram</code> .
digits	positive integer indicating the number of decimal digits to print.
correlation	logical controlling whether the correlation matrix of the fitted variogram parameters is computed (default FALSE).
signif	confidence level for computing confidence intervals for variogram parameters (default 0.95).
what	the quantity that should be displayed (default "variogram").
from	numeric, minimal lag distance used in plotting variogram models.
to	numeric, maximum lag distance used in plotting variogram models (default: largest lag distance of current plot).
n	positive integer specifying the number of equally spaced lag distances for which semi-variances are evaluated in plotting variogram models (default 501).
xy.angle	numeric (vector) with azimuth angles (in degrees, clockwise positive from north) in x - y -plane for which semi-variances should be plotted.
xz.angle	numeric (vector) with angles in x - z -plane (in degrees, clockwise positive from zenith to south) for which semi-variances should be plotted.
col	color of curves to distinguish curves relating to different azimuth angles in x - y -plane.
pch	type of plotting symbols added to lines to distinguish curves relating to different angles in x - z -plane.
lty	line type for plotting variogram models.
...	additional arguments passed to methods.

Details

The parametrization of geometrically anisotropic variograms is described in detail in [georobIntro](#), and the section *Details* of [georob](#) describes how the parameter estimates are constrained to permissible ranges. The same mechanisms are used in `fit.variogram.model`.

The method `summary` computes confidence intervals of the estimated variogram and anisotropy parameters from the Hessian matrix of the residual sums of squares, based on the asymptotic normal distribution of least squares estimate estimates. Note that the Hessian matrix with respect to the *transformed* variogram and anisotropy parameters is used for this. Hence the inverse Hessian matrix is the covariance matrix of the transformed parameters, confidence intervals are first computed for the transformed parameters and the limits of these intervals are transformed back to the original scale of the parameters. Optionally, `summary` reports the correlation matrix of the *transformed* parameters, also computed from the Hessian matrix.

Value

The function `fit.variogram.model` generates an object of class `fitted.variogram` which is a list with the following components:

<code>sse</code>	the value of the object function (weighted residual sum of squares) evaluated at the solution.
<code>variogram.object</code>	the estimated parameters of a possibly nested variograms model. This is a list that contains for each variogram model structure the following components: <ul style="list-style-type: none"> • <code>variogram.model</code>: the name of the fitted parametric variogram model. • <code>param</code>: a named numeric vector with the (estimated) variogram parameters. • <code>fit.param</code>: a named logical vector with the flags defining what variogram parameters were estimated. • <code>isotropic</code>: logical indicating whether an isotropic variogram was fitted. • <code>aniso</code>: a named numeric vector with the (estimated) anisotropy parameters. • <code>fit.aniso</code>: a named logical vector with the flags defining what anisotropy parameters were estimated. • <code>sincos</code>: a list with <code>sin</code> and <code>cos</code> of the angles ω, ϕ and ζ that define the orientation of the anisotropy ellipsoid. • <code>rotmat</code>: the matrix (C_1, C_2, C_3) (see georobIntro). • <code>sclmat</code>: a vector with the elements $1, 1/f_1, 1/f_2$ (see georobIntro).
<code>param.tf</code>	a character vector listing the transformations of the variogram parameters used for model fitting.
<code>fwd.tf</code>	a list of functions for variogram parameter transformations.
<code>bwd.tf</code>	a list of functions for <i>inverse</i> variogram parameter transformations.
<code>converged</code>	logical indicating whether numerical maximization by <code>optim</code> converged.
<code>convergence.code</code>	a diagnostic integer issued by <code>optim</code> (component convergence) about convergence.
<code>iter</code>	a named integer vector of length two with the number of function and gradient evaluations by <code>optim</code> .

call	the matched call.
residuals	a numeric vector with the residuals, that is the sample semi-variance minus the fitted values.
fitted	a numeric vector with the modelled semi-variances.
weights	a numeric vector with the weights used for fitting.
hessian	a symmetric matrix with the Hessian at the solution (missing if hessian is false).
hessian.tfpa	a symmetric matrix with the Hessian at the solution with respect to the transformed variogram and anisotropy parameters (missing if hessian is false). This Hessian is used by <code>summary.fitted.variogram</code> to compute confidence intervals for the estimated parameters.
hessian.ntfpa	a symmetric matrix with the Hessian at the solution with respect to the non-transformed variogram and anisotropy parameters (missing if hessian is false).

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

References

Cressie, N. A. C. (1993) *Statistics for Spatial Data*. New York: John Wiley & Sons.

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms; [georob](#) for (robust) fitting of spatial linear models;
[georobObject](#) for a description of the class `georob`;
[profilelogLik](#) for computing profiles of Gaussian likelihoods;
[plot.georob](#) for display of RE(ML) variogram estimates;
[control.georob](#) for controlling the behaviour of `georob`;
[georobModelBuilding](#) for stepwise building models of class `georob`;
[cv.georob](#) for assessing the goodness of a fit by `georob`;
[georobMethods](#) for further methods for the class `georob`;
[predict.georob](#) for computing robust Kriging predictions;
[lgpp](#) for unbiased back-transformation of Kriging prediction of log-transformed data;
[georobSimulation](#) for simulating realizations of a Gaussian process from model fitted by `georob`.

Examples

```
data(wolfcamp)

## fitting an isotropic IRF(0) model
r.sv.iso <- sample.variogram(pressure~1, data = wolfcamp,
  locations = ~x + y, lag.dist.def = seq(0, 200, by = 15))

## Not run:
```

```

r.irf0.iso <- fit.variogram.model(r.sv.iso, variogram.model = "RMfbm",
  param = c(variance = 100, nugget = 1000, scale = 1., alpha = 1.),
  fit.param = default.fit.param(scale = FALSE, alpha = TRUE))
summary(r.irf0.iso, correlation = TRUE)

plot(r.sv.iso, type = "l")
lines(r.irf0.iso, line.col = "red")
## End(Not run)

## fitting an anisotropic IRF(0) model
r.sv.aniso <- sample.variogram(pressure~1, data = wolfcamp,
  locations = ~x + y, lag.dist.def = seq(0, 200, by = 15),
  xy.angle.def = c(0., 22.5, 67.5, 112.5, 157.5, 180.))

## Not run:
r.irf0.aniso <- fit.variogram.model(r.sv.aniso, variogram.model = "RMfbm",
  param = c(variance = 100, nugget = 1000, scale = 1., alpha = 1.5),
  fit.param = default.fit.param(scale = FALSE, alpha = TRUE),
  aniso = default.aniso(f1 = 0.4, omega = 135.),
  fit.aniso = default.fit.aniso(f1 = TRUE, omega = TRUE),
  control = control.fit.variogram.model(
    maximizer = "optim",
    optim = control.optim(
      method = "BFGS", hessian = TRUE, control = list(maxit = 5000)
    )
  )
  ))
summary(r.irf0.aniso, correlation = TRUE)

plot(r.sv.aniso, type = "l")
lines(r.irf0.aniso, xy.angle = seq(0, 135, by = 45))
## End(Not run)

```

Description

The function `georob` fits a linear model with spatially correlated errors to geostatistical data that are possibly contaminated by independent outliers. The regression coefficients and the parameters of the variogram model are estimated by robust or Gaussian restricted maximum likelihood (REML) or by Gaussian maximum likelihood (ML).

Usage

```

georob(formula, data, subset, weights, na.action, model = TRUE,
  x = FALSE, y = FALSE, contrasts = NULL, offset, locations,
  variogram.model = c("RMexp", "RMaskey", "RMBessel", "RMcauchy",
    "RMcircular", "RMcubic", "RMDagum", "RMDampedcos", "RMDewijsian",
    "RMfbm", "RMgauss", "RMgencauchy", "RMgenfbm", "RMgengneiting",

```

```

"RMgneiting", "RMlgd", "RMmatern", "RMpenta", "RMqexp",
"RMspheric", "RMstable", "RMwave", "RMwhittle"),
param, fit.param = default.fit.param()[names(param)],
aniso = default.aniso(), fit.aniso = default.fit.aniso(),
variogram.object = NULL,
tuning.psi = 2, control = control.georob(),
verbose = 0, ...)

```

Arguments

formula	a symbolic description of the regression model for the external drift to be fit (mandatory argument). See <code>lm</code> and <code>formula</code> for more details.
data	an optional data frame, a <code>SpatialPointsDataFrame</code> , list or environment (or another object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model and the coordinates where the data was recorded. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>georob</code> is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of weights to be used in the fitting process, currently ignored.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the <code>na.action</code> argument of <code>options</code> , and is <code>na.fail</code> if that is unset. The “factory-fresh” default is <code>na.omit</code> . Another possible value is <code>NULL</code> , no action. Value <code>na.exclude</code> can be useful.
model, x, y	logicals. If <code>TRUE</code> the corresponding components of the fit (the model frame, the model matrix, the response) are returned. The model frame is augmented by the coordinates.
contrasts	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
offset	this optional argument can be used to specify an <i>a priori</i> known component to be included in the linear predictor during fitting. An <code>offset</code> term can be included in the formula instead or as well, and if both are specified their sum is used.
locations	a one-sided formula defining the variables that are used as coordinates of the locations where the data was recorded (mandatory argument).
variogram.model	a character keyword defining the variogram model to be fitted. Currently, most basic variogram models provided by the package RandomFields can be fitted (see <i>Details</i> and <code>RMmodel</code>).
param	a named numeric vector with initial values of the variogram parameters (mandatory argument). The names of <code>param</code> are matched against the following names (see <i>Details</i> and <code>georobIntro</code> for information about the parametrization of variogram models): <ul style="list-style-type: none"> • variance: variance (sill σ^2) of the auto-correlated component of the Gaussian random field $B(s)$. • snugget: variance (spatial nugget σ_n^2) of the seemingly spatially uncorrelated component of $B(s)$ (micro-scale spatial variation; default value snugget = 0).

	<ul style="list-style-type: none"> • nugget: variance (nugget τ^2) of the independent errors $\varepsilon(\mathbf{s})$. • scale: range parameter (α) of the variogram. • names of additional variogram parameters such as the smoothness parameter ν of the Whittle-Matérn model (see RMmodel and param.names).
<code>fit.param</code>	a named logical vector (or a function such as default.fit.param that creates this vector) with the same names as used for <code>param</code> , defining which parameters are adjusted (TRUE) and which are kept fixed at their initial values (FALSE) when fitting the model.
<code>aniso</code>	a named numeric vector with initial values (or a function such as default.aniso that creates this vector) for fitting geometrically anisotropic variogram models. The names of <code>aniso</code> are matched against the following names (see Details and georobIntro for information about the parametrization of variogram models): <ul style="list-style-type: none"> • <code>f1</code>: ratio f_1 of lengths of second and first semi-principal axes of an ellipsoidal surface with constant semi-variance in \mathbb{R}^3 (default <code>f1 = 1</code>). • <code>f2</code>: ratio f_2 of lengths of third and first semi-principal axes of the semi-variance ellipsoid (default <code>f2 = 1</code>). • <code>omega</code>: azimuth in degrees of the first semi-principal axis of the semi-variance ellipsoid (default <code>omega = 90</code>). • <code>phi</code>: 90 degrees minus altitude of the first semi-principal axis of the semi-variance ellipsoid (default <code>phi = 90</code>). • <code>zeta</code>: angle in degrees between the second semi-principal axis and the direction of the line defined by the intersection between the x-y-plane and the plane orthogonal to the first semi-principal axis of the semi-variance ellipsoid through the origin (default <code>zeta = 0</code>).
<code>fit.aniso</code>	a named logical vector (or a function such as default.fit.aniso that creates this vector) with the same names as used for <code>aniso</code> , defining which parameters are adjusted (TRUE) and which are kept fixed at their initial values (FALSE) when fitting the model.
<code>variogram.object</code>	an optional list that defines a possibly nested variogram model. Each component is itself a list with the following components: <ul style="list-style-type: none"> • <code>variogram.model</code>: a mandatory character keyword defining the variogram model, see respective argument above. • <code>param</code>: a mandatory named numeric vector with initial values of the variogram parameters, see respective argument above. • <code>fit.param</code>: an optional named logical vector defining which parameters are adjusted, see respective argument above. • <code>aniso</code>: an optional named numeric vector with initial values for fitting geometrically anisotropic variogram models, see respective argument above. • <code>fit.aniso</code>: an optional named logical vector defining which anisotropy parameters are adjusted, see respective argument above. <p>Note that the arguments <code>variogram.model</code>, <code>param</code>, <code>fit.param</code>, <code>aniso</code> and <code>fit.aniso</code> are ignored when <code>variogram.object</code> is passed to <code>georob</code>.</p>
<code>tuning.psi</code>	positive numeric. The tuning constant c of the ψ_c -function of the robust REML algorithm.

control	a list specifying parameters that control the behaviour of georob. Use the function <code>control.georob</code> and see its help page for the components of control.
verbose	positive integer controlling logging of diagnostic messages to the console during model fitting. <code>verbose = 0</code> largely suppresses such messages and <code>verbose = 4</code> asks for most verbose output (see control arguments of <code>nleqslv</code> , <code>nlminb</code> and <code>optim</code> and <code>control.georob</code> for information how to fine tuning diagnostic output generated by <code>nleqslv</code> , <code>nlminb</code> and <code>optim</code>).
...	further arguments passed to function (e.g. <code>object</code> . used internally for updating georob objects).

Details

georob fits a spatial linear model by robust or Gaussian RE(ML) (Künsch *et al.*, 2011, Künsch *et al.*, in preparation). `georobIntro` describes the employed model and briefly sketches the robust REML estimation and the robust external drift Kriging method. Here, we describe further details of georob.

Implemented variograms:

Currently, most basic variogram models provided by the package **RandomFields** can be fitted by georob (see argument `variogram.model` for a list of implemented models). Some of these models have in addition to variance, `snugget`, `nugget` and `scale` further parameters. Initial values of these parameters (`param`) and fitting flags (`fit.param`) must be passed to georob by the same names as used by the functions `RM...` of the package **RandomFields** (see `RMmodel`). Use the function `param.names` to list additional parameters of a given `variogram.model`.

The arguments `fit.param` and `fit.aniso` are used to control what variogram and anisotropy parameters are estimated and which are kept at the constant initial values. The functions `default.fit.param` and `default.fit.aniso` set reasonable default values for these arguments. Note, as an aside, that the function `default.aniso` sets (default) values of the anisotropy parameters for an isotropic variogram.

Estimating parameters of power function variogram:

The intrinsic variogram model `RMfbm` is over-parametrized when both the variance (plus possibly `snugget`) and the `scale` are estimated. Therefore, to estimate the parameters of this model, `scale` must be kept fixed at an arbitrary value by using `fit.param["scale"] = FALSE`.

Estimating parameters of geometrically anisotropic variograms:

The subsection **Model** of `georobIntro` describes how such models are parametrized and gives definitions the various elements of `aniso`. Some additional remarks might be helpful:

- The first semi-principal axis points into the direction with the farthest reaching auto-correlation, which is described by the range parameter `scale` (α).
- The ranges in the direction of the second and third semi-principal axes are given by $f_1\alpha$ and $f_2\alpha$, with $0 < f_2 \leq f_1 \leq 1$.
- The default values for `aniso` ($f_1 = 1$, $f_2 = 1$) define an isotropic variogram model.
- Valid ranges for the angles characterizing the orientation of the semi-variance ellipsoid are (in degrees): ω $[0, 180]$, ϕ $[0, 180]$, ζ $[-90, 90]$.

Estimating variance of micro-scale variation:

Simultaneous estimation of the variance of the micro-scale variation (snugget, σ_n^2), appears seemingly as spatially uncorrelated with a given sampling design, and of the variance (nugget, τ^2) of the independent errors requires that for some locations s_i replicated observations are available. Locations less or equal than zero.dist apart are thereby considered as being coincident (see [control.georob](#)).

Constraining estimates of variogram parameters:

Parameters of variogram models can vary only within certain bounds (see [param.bounds](#) and [RMmodel](#) for allowed ranges). georob uses three mechanisms to constrain parameter estimates to permissible ranges:

1. *Parameter transformations:* By default, all variance (variance, snugget, nugget), the range scale, the anisotropy parameters f1 and f2 and many of the additional parameters are log-transformed before solving the estimating equations or maximizing the restricted log-likelihood and this warrants that the estimates are always positive (see [control.georob](#) for detailed explanations how to control parameter transformations).
2. *Checking permissible ranges:* The additional parameters of the variogram models such as the smoothness parameter ν of the Whittle-Matern model are forced to stay in the permissible ranges by signalling an error to `nleqslv`, `nlminb` or `optim` if the current trial values are invalid. These functions then graciously update the trial values of the parameters and carry their task on. However, it is clear that such a procedure likely gets stuck at a point on the boundary of the parameter space and is therefore just a workaround for avoiding runtime errors due to invalid parameter values.
3. *Exploiting the functionality of nlminb and optim:* If a spatial model is fitted non-robustly, then the arguments `lower`, `upper` (and `method` of `optim`) can be used to constrain the parameters (see [control.optim](#) how to pass them to `optim`). For `optim` one has to use the arguments `method = "L-BFGS-B"`, `lower = l`, `upper = u`, where `l` and `u` are numeric vectors with the lower and upper bounds of the *transformed* parameters in the order as they appear in `c(variance, snugget, nugget, scale, ...)[fit.param], aniso[fit.aniso])`, where `...` are additional parameters of isotropic variogram models (use `param.names(variogram.model)` to display the names and the order of the additional parameters for `variogram.model`). For `nlminb` one has to use the arguments `codelower = l`, `upper = u`, where `l` and `u` are numeric vectors as described above.

Computing robust initial estimates of parameters for robust REML:

To solve the robustified estimating equations for B and β the following initial estimates are used:

- $\hat{B} = \mathbf{0}$, if this turns out to be infeasible, initial values can be passed to georob by the argument `bhat` of [control.georob](#).
- $\hat{\beta}$ is either estimated robustly by the function `lmrob`, `rq` or non-robustly by `lm` (see argument `initial.fixef` of [control.georob](#)).

Finding the roots of the robustified estimating equations of the variogram and anisotropy parameters is more sensitive to a good choice of initial values than maximizing the Gaussian (restricted) log-likelihood with respect to the same parameters. If the initial values for `param` and `aniso` are not sufficiently close to the roots of the system of nonlinear equations, then `nleqslv` may fail to find them. Setting `initial.param = TRUE` (see [control.georob](#)) allows one to find initial values that are often sufficiently close to the roots so that `nleqslv` converges. This is achieved by:

1. Initial values of the regression parameters are computed by `lmrob` irrespective of the choice for `initial.fixef` (see `control.georob`).
2. Observations with “robustness weights” of the `lmrob` fit, satisfying $\psi_c(\hat{\varepsilon}_i/\hat{\tau})/(\hat{\varepsilon}_i/\hat{\tau}) \leq \text{min.rweight}$, are discarded (see `control.georob`).
3. The model is fit to the pruned data set by Gaussian REML using `nlminb` or `optim`.
4. The resulting estimates of the variogram parameters (`param`, `aniso`) are used as initial estimates for the subsequent robust fit of the model by `nleqslv`.

Note that for step 3 above, initial values of `param` and `aniso` must be provided to `georob`.

Estimating variance parameters by Gaussian (RE)ML:

Unlike robust REML, where robustified estimating equations are solved for the variance parameters nugget (τ^2), variance (σ^2), and possibly snugglet (σ_n^2), for Gaussian (RE)ML the variances can be re-parametrized to

- the signal variance $\sigma_B^2 = \sigma^2 + \sigma_n^2$,
- the inverse relative nugget $\eta = \sigma_B^2/\tau^2$ and
- the relative auto-correlated signal variance $\xi = \sigma^2/\sigma_B^2$.

`georob` maximizes then a (restricted) *profile log-likelihood* that depends only on η , ξ , α , \dots , and σ_B^2 is estimated by an explicit expression that depends on these parameters (e.g. Diggle and Ribeiro, 2006, p. 113). This is usually more efficient than maximizing the (restricted) log-likelihood with respect to the original variance parameters τ^2 , σ_n^2 and σ^2 . `georob` chooses the parametrization automatically, but the user can control it by the argument `reparam` of the function `control.georob`.

Value

An object of class `georob` representing a robust (or Gaussian) (RE)ML fit of a spatial linear model. See `georobObject` for the components of the fit.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>
with contributions by Cornelia Schwierz.

References

- Diggle, P. J. and Ribeiro, P. J. R. (2006) Model-based Geostatistics. Springer.
- Künsch, H. R., Papritz, A., Schwierz, C. and Stahel, W. A. (in preparation) Robust Geostatistics.
- Künsch, H. R., Papritz, A., Schwierz, C. and Stahel, W. A. (2011) Robust estimation of the external drift and the variogram of spatial data. Proceedings of the ISI 58th World Statistics Congress of the International Statistical Institute. doi: [10.3929/ethz009900710](https://doi.org/10.3929/ethz009900710)

See Also

`georobIntro` for a description of the model and a brief summary of the algorithms;
`georobObject` for a description of the class `georob`;
`profilelogLik` for computing profiles of Gaussian likelihoods;

[plot.georob](#) for display of RE(ML) variogram estimates;
[control.georob](#) for controlling the behaviour of georob;
[georobModelBuilding](#) for stepwise building models of class georob;
[cv.georob](#) for assessing the goodness of a fit by georob;
[georobMethods](#) for further methods for the class georob;
[predict.georob](#) for computing robust Kriging predictions;
[lgnpp](#) for unbiased back-transformation of Kriging prediction of log-transformed data;
[georobSimulation](#) for simulating realizations of a Gaussian process from model fitted by georob;
 and finally
[sample.variogram](#) and [fit.variogram.model](#) for robust estimation and modelling of sample variograms.

Examples

```

## Not run:

## meuse data ##
#####
data(meuse)

## Gaussian REML fit
r.logzn.reml <- georob(log(zinc) ~ sqrt(dist), data = meuse, locations = ~ x + y,
  variogram.model = "RMexp",
  param = c(variance = 0.15, nugget = 0.05, scale = 200),
  tuning.psi = 1000)
summary(r.logzn.reml, correlation = TRUE)

## robust REML fit
r.logzn.rob <- update(r.logzn.reml, tuning.psi = 1)

summary(r.logzn.rob, correlation = TRUE)

plot(r.logzn.reml, lag.dist.def = seq(0, 2000, by = 100))
lines(r.logzn.rob, col = "red")

#####
## wolfcamp data ##
#####
data(wolfcamp)

## fitting isotropic IRF(0) model

r.irf0.iso <- georob(pressure ~ 1, data = wolfcamp, locations = ~ x + y,
  variogram.model = "RMfbm",
  param = c(variance = 10, nugget = 1500, scale = 1, alpha = 1.5),
  fit.param = default.fit.param(scale = FALSE, alpha = TRUE),
  tuning.psi = 1000)

```

```

summary(r.irf0.iso)

## fitting anisotropic IRF(0) model

r.irf0.aniso <- georob(pressure ~ 1, data = wolfcamp, locations = ~ x + y,
  variogram.model = "RMfbm",
  param = c(variance = 5.9, nugget = 1450, scale = 1, alpha = 1),
  fit.param = default.fit.param(scale = FALSE, alpha = TRUE),
  aniso = default.aniso(f1 = 0.51, omega = 148.),
  fit.aniso = default.fit.aniso(f1 = TRUE, omega = TRUE),
  tuning.psi = 1000)
summary(r.irf0.aniso)

plot(r.irf0.iso, lag.dist.def = seq(0, 200, by = 7.5))
plot(r.irf0.aniso, lag.dist.def = seq(0, 200, by = 7.5),
  xy.angle.def = c(0, 22.5, 67.5, 112.5, 157.5, 180.),
  add = TRUE, col = 2:5)

pchisq(2*(r.irf0.aniso[["loglik"]] - r.irf0.iso[["loglik"]]), 2, lower = FALSE)
## End(Not run)

```

georobModelBuilding *S3 Methods for Stepwise Building Fixed-Effects Models for Class*
georob

Description

This page documents the methods `deviance`, `logLik`, `extractAIC`, `add1`, `drop1`, `step` and `waldtest` for the class `georob`. The package `georob` provides a generic `step` function and a default method which is identical with the (non-generic) function `step`.

Usage

```

## S3 method for class 'georob'
deviance(object, warn = TRUE, REML = FALSE, ...)

## S3 method for class 'georob'
logLik(object, warn = TRUE, REML = FALSE, ...)

## S3 method for class 'georob'
extractAIC(fit, scale = 0, k = 2, ...)

## S3 method for class 'georob'
add1(object, scope, scale = 0, test = c("none", "Chisq"), k = 2,
  trace = FALSE, fixed = TRUE, use.fitted.param = TRUE, verbose = 0,
  ncores = 1, ...)

## S3 method for class 'georob'

```

```

drop1(object, scope, scale = 0, test = c("none", "Chisq"), k = 2,
      trace = FALSE, fixed = TRUE, use.fitted.param = TRUE, verbose = 0,
      ncores = 1, ...)

step(object, ...)

## Default S3 method:
step(object, scope, scale = 0,
      direction = c("both", "backward", "forward"), trace = 1,
      keep = NULL, steps = 1000, k = 2, ...)

## S3 method for class 'georob'
step(object, scope, scale = 0,
      direction = c("both", "backward", "forward"), trace = 1,
      keep = NULL, steps = 1000, k = 2,
      fixed.add1.drop1 = TRUE, fixed.step = fixed.add1.drop1,
      use.fitted.param = TRUE, verbose = 0, ncores = 1, ...)

## S3 method for class 'georob'
waldtest(object, ..., vcov = NULL, test = c("F", "Chisq"),
          name = NULL)

```

Arguments

object, fit	an object of class <code>georob</code> , see georobObject .
direction	the mode of stepwise search, see step .
fixed, fixed.add1.drop1	logical controlling whether the variogram parameters are <i>not</i> adjusted when adding or dropping model terms by <code>add1</code> and <code>drop1</code> (default <code>TRUE</code>), see <i>Details</i> .
fixed.step	logical controlling whether the variogram parameters are <i>not</i> adjusted after having called <code>add1</code> and <code>drop1</code> in <code>step</code> (default <code>TRUE</code>), see <i>Details</i> .
k	numeric specifying the 'weight' of the equivalent degrees of freedom (=: edf) part in the AIC formula, see extractAIC .
keep	a filter function whose input is a fitted model object and the associated AIC statistic, and whose output is arbitrary, see step .
name	a function for extracting a suitable name/description from a fitted model object. By default the name is queried by calling formula , see waldtest .
ncores	integer specifying the number of cores used for parallelized execution of <code>add1</code> and <code>drop1</code> . If larger than one then the minimum of <code>ncores</code> , <code>detectCores()</code> and the number of terms to be added or dropped determines the number of cores that is actually used.
REML	logical controlling whether the restricted log-likelihood should be extracted (default <code>TRUE</code>).
scale	numeric, currently not used, see extractAIC .

scope	defines the range of models examined in the stepwise search. This should be either a single formula, or a list containing components upper and lower, both formulae, see step for details.
steps	the maximum number of steps to be considered (default is 1000), see step .
test	character keyword specifying whether to compute the large sample Chi-squared statistic (with asymptotic Chi-squared distribution) or the finite sample F statistic (with approximate F distribution), see waldtest .
trace	if positive, information is printed during the running of step, see step .
use.fitted.param	logical scalar controlling whether fitted values of param (and aniso are used as initial values when variogram parameters are fitted afresh for adding and dropping terms from the model (default TRUE).
vcov	a function for estimating the covariance matrix of the regression coefficients, see waldtest .
verbose	positive integer controlling logging of diagnostic messages to the console during model fitting, see georob (default 0).
warn	logical scalar controlling whether warnings should be suppressed.
...	additional arguments passed to methods (see in particular <code>waldtest.default</code>).

Details

For a non-robust fit the function deviance returns the residual deviance

$$(\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}})^T (\hat{\tau}^2 \mathbf{I} + \mathbf{\Gamma}_{\hat{\theta}})^{-1} (\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}})$$

(see [georobPackage](#) for an explanation of the notation). For a robust fit the deviance is not defined. The function then computes with a warning the deviance of an equivalent Gaussian model with heteroscedastic nugget τ^2/w where w are the “robustness weights” `rweights`, see [georobObject](#).

`logLik` returns the the maximized (restricted) log-likelihood. For a robust fit, the log-likelihood is not defined. The function then computes the (restricted) log-likelihood of an equivalent Gaussian model with heteroscedastic nugget (see above).

The methods `extractAIC`, `add1`, `drop1` and `step` are used for stepwise model building.

If `fixed==TRUE` or `fixed.add1.drop1==TRUE` (default) then the variogram parameters are kept fixed at the values of `object`. For `fixed==FALSE` or `fixed.add1.drop1==FALSE` the variogram parameters are fitted afresh for each model tested by `add1` and `drop1`. Then either the variogram parameters in `object$initial.objects` (`use.fitted.param==FALSE`) or the fitted parameters of `object` (`use.fitted.param==TRUE`) are used as initial values. For `fixed.step==TRUE` the variogram parameters are *not* fitted afresh by `step` after the calls to `drop1` and `add1` have been completed, unlike for `fixed.step==FALSE` where the parameters are estimated afresh for the new model that minimized AIC (BIC) in the previous step.

In addition, the functions of the R package **multcomp** can be used to test general linear hypotheses about the fixed effects of the model.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms;
[georob](#) for (robust) fitting of spatial linear models;
[georobObject](#) for a description of the class georob;
[profilelogLik](#) for computing profiles of Gaussian likelihoods;
[plot.georob](#) for display of RE(ML) variogram estimates;
[control.georob](#) for controlling the behaviour of georob;
[cv.georob](#) for assessing the goodness of a fit by georob;
[georobMethods](#) for further methods for the class georob;
[predict.georob](#) for computing robust Kriging predictions;
[lgnpp](#) for unbiased back-transformation of Kriging prediction of log-transformed data;
[georobSimulation](#) for simulating realizations of a Gaussian process from model fitted by georob;
 and finally
[sample.variogram](#) and [fit.variogram.model](#) for robust estimation and modelling of sample variograms.

Examples

```

## Not run:

data(meuse)

## Gaussian REML fit
r.logzn.reml <- georob(log(zinc) ~ sqrt(dist), data = meuse, locations = ~ x + y,
  variogram.model = "RMexp",
  param = c(variance = 0.15, nugget = 0.05, scale = 200),
  tuning.psi = 1000)
summary(r.logzn.reml, correlation = TRUE)

deviance(r.logzn.reml)
logLik(r.logzn.reml)

waldtest(r.logzn.reml, .~. + ffreq)

step(r.logzn.reml, ~ sqrt(dist) + ffreq + soil)

## robust REML fit
r.logzn.rob <- update(r.logzn.reml, tuning.psi = 1)

deviance(r.logzn.rob)
logLik(r.logzn.rob)
logLik(r.logzn.rob, REML=TRUE)

step(r.logzn.rob, ~ sqrt(dist) + ffreq + soil, fixed.step=FALSE, trace=2)

## End(Not run)

```

georobObject	<i>Fitted georob Object</i>
--------------	-----------------------------

Description

An object of class `georob` as returned by `georob` and representing a (robustly) fitted spatial linear model. Objects of this class have methods for model building (see `georobModelBuilding`) and cross-validation (see `cv.georob`), for computing (robust) Kriging predictions (see `predict.georob`), for plotting (see `plot.georob`) and for common generic functions (see `georobMethods`).

Value

A `georob` object is a list with following components:

<code>loglik</code>	the maximized (restricted) Gaussian log-likelihood of a non-robust (RE)ML fit or NA for a robust fit if <code>tuning.psi</code> is less than <code>tuning.psi.nr</code> .
<code>variogram.object</code>	the estimated parameters of a possibly nested variograms model. This is a list that contains for each variogram model structure the following components: <ul style="list-style-type: none"> • <code>variogram.model</code>: the name of the fitted parametric variogram model. • <code>param</code>: a named numeric vector with the (estimated) variogram parameters. • <code>fit.param</code>: a named logical vector with the flags defining what variogram parameters were estimated. • <code>isotropic</code>: logical indicating whether an isotropic variogram was fitted. • <code>aniso</code>: a named numeric vector with the (estimated) anisotropy parameters. • <code>fit.aniso</code>: a named logical vector with the flags defining what anisotropy parameters were estimated. • <code>sincos</code>: a list with <code>sin</code> and <code>cos</code> of the angles ω, ϕ and ζ that define the orientation of the anisotropy ellipsoid. • <code>rotmat</code>: the matrix (C_1, C_2, C_3) (see <code>georobIntro</code>). • <code>sclmat</code>: a vector with the elements $1, 1/f_1, 1/f_2$ (see <code>georobIntro</code>).
<code>gradient</code>	a named numeric vector with the estimating equations (robust REML) or the gradient of the maximized (restricted) log-likelihood (Gaussian (RE)ML) evaluated at the solution.
<code>tuning.psi</code>	the value of the tuning constant c of the ψ_c -function.
<code>coefficients</code>	a named vector with the estimated regression coefficients.
<code>fitted.values</code>	a named vector with the fitted values of the external drift $\mathbf{X}\hat{\boldsymbol{\beta}}$.
<code>bhat</code>	a named vector with the predicted spatial random effects $\hat{\mathbf{B}}$ at the data locations.
<code>residuals</code>	a named vector with the residuals $\hat{\boldsymbol{\varepsilon}} = \mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}} - \hat{\mathbf{B}}$.
<code>rweights</code>	a named numeric vector with the “robustness weights” $\psi(\hat{\boldsymbol{\varepsilon}}_i/\hat{\tau})/(\hat{\boldsymbol{\varepsilon}}_i/\hat{\tau})$.
<code>converged</code>	logical indicating whether numerical maximization of the (restricted) log-likelihood by <code>nlminb</code> or <code>optim</code> or root finding by <code>nleqslv</code> converged.

convergence.code	a diagnostic integer issued by <code>nlminb</code> , <code>optim</code> (component convergence) or <code>nleqslv</code> (component termcd) about convergence.
iter	a named integer vector of length two, indicating either <ul style="list-style-type: none"> • the number of function and gradient evaluations when maximizing the (restricted) Gaussian log-likelihood by <code>nlminb</code> or <code>optim</code>, or • the number of function and Jacobian evaluations when solving the robustified estimating equations by <code>nleqslv</code>.
Tmat	the compressed design matrix for replicated observations at coincident locations (integer vector that contains for each observation the row index of the respective unique location).
cov	a list with covariance matrices (or diagonal variance vectors). Covariance matrices are stored in <i>compressed form</i> (see <code>compress</code>) and can be expanded to square matrices by <code>expand</code> . What cov actually contains depends on the flags passed to <code>georob</code> for computing covariances (see <code>control.georob</code>). Possible components are: <ul style="list-style-type: none"> • <code>cov.bhat</code>: the covariances of $\widehat{\mathbf{B}}$. • <code>cov.betahat</code>: the covariances of $\widehat{\boldsymbol{\beta}}$. • <code>cov.delta.bhat</code>: the covariances of $\mathbf{B} - \widehat{\mathbf{B}}$. • <code>cov.delta.bhat.betahat</code>: the covariances of $\mathbf{B} - \widehat{\mathbf{B}}$ and $\widehat{\boldsymbol{\beta}}$. • <code>cov.ehat</code>: the covariances of $\widehat{\boldsymbol{\varepsilon}} = \mathbf{Y} - \mathbf{X}\widehat{\boldsymbol{\beta}} - \widehat{\mathbf{B}}$. • <code>cov.ehat.p.bhat</code>: the covariances of $\widehat{\boldsymbol{\varepsilon}} + \widehat{\mathbf{B}} = \mathbf{Y} - \mathbf{X}\widehat{\boldsymbol{\beta}}$. • <code>cov.pred.target</code>: a covariance term required for the back-transformation of Kriging predictions of log-transformed data.
expectations	a named numeric vector with the expectations of $\partial\psi_c(x)/\partial x$ (<code>dpsi</code>) and $\psi_c^2(x)$ (<code>psi2</code>) with respect to a standard normal distribution (<code>exp.gauss</code>) and the long-tailed distribution of ε (<code>exp.f0</code>) implied by the choice of the ψ_c -function.
Valphaxi.objects	a list of matrices in <i>compressed form</i> with (among others) the following components: <ul style="list-style-type: none"> • <code>Valpha</code>: a list with the (generalized) correlation matrices (<code>Valpha</code>) of the nested variogram models structures along with the constants (<code>gcr.constant</code>) added to the respective semivariances matrices. • <code>Valphaxi</code>: the (generalized) correlation matrix $\mathbf{V}_{\alpha,\xi} = \boldsymbol{\Gamma}_{\alpha,\xi}/(\sigma_n^2 + \sigma^2)$ that includes the spatial nugget effect. • <code>Valphaxi.inverse</code>: the inverse of $\mathbf{V}_{\alpha,\xi}$. • <code>log.det.Valphaxi</code>: $\log(\det(\mathbf{V}_{\alpha,\xi}))$.
zhat.objects	a list of matrices in (partly) <i>compressed form</i> with the following components: <ul style="list-style-type: none"> • <code>Aalphaxi</code>: the matrix $(\mathbf{X}^T \mathbf{V}_{\alpha,\xi}^{-1} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{V}_{\alpha,\xi}^{-1}$. • <code>Palphaxi</code>: the matrix $\mathbf{I} - \mathbf{X} \mathbf{A}_{\alpha,\xi}$. • <code>Valphaxi.inverse.Palphaxi</code>: the matrix $\mathbf{V}_{\alpha,\xi}^{-1} \mathbf{P}_{\alpha,\xi}$.
locations.object	a list with 3 components:

- `locations`: a formula indicating the coordinates of the measurement locations.
- `coordinates`: a numeric matrix with the coordinates of the measurement locations.
- `lag.vectors`: a numeric matrix with the lag vectors between any distinct pairs of measurement locations.

`initial.objects` a list with 3 components:

- `coefficients`: initial estimates of β computed either by `lmrob` or `rq`.
- `bhat`: initial predictions of B .
- `variogram.object`: the initial values of the parameters of a possibly nested variograms model. This is a list with the same structure as described above for the component `variogram.object`.

`hessian.tfpa` a symmetric matrix with the Hessian (observed Fisher information) at the solution with respect to the transformed variogram and anisotropy parameters if the model was fitted non-robustly with the argument `hessian = TRUE` (see `control.georob`). Missing otherwise. This Hessian is used by `summary.georob` to compute confidence intervals for the estimated parameters.

`hessian.ntfpa` a symmetric matrix with the Hessian (observed Fisher information) at the solution with respect to the non-transformed variogram and anisotropy parameters if the model was fitted non-robustly with the argument `hessian = TRUE` (see `control.georob`). Missing otherwise.

`control` a list with control parameters generated by `control.georob`.

`MD` optionally a matrix of robust distances in the space spanned by \mathbf{X} (see argument `compute.rd` of `lmrob.control` and `control.georob`).

`model, x, y` if requested the model frame, the model matrix and the response, respectively.

`na.action, offset, contrasts, xlevels, rank, df.residual, call, terms` further components of the fit as described for an object of class `lm`.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms;

[georob](#) for (robust) fitting of spatial linear models;

[profilelogLik](#) for computing profiles of Gaussian likelihoods;

[plot.georob](#) for display of RE(ML) variogram estimates;

[control.georob](#) for controlling the behaviour of `georob`;

[georobModelBuilding](#) for stepwise building models of class `georob`;

[cv.georob](#) for assessing the goodness of a fit by `georob`;

[georobMethods](#) for further methods for the class `georob`;

`predict.georob` for computing robust Kriging predictions;
`lgnpp` for unbiased back-transformation of Kriging prediction of log-transformed data;
`georobSimulation` for simulating realizations of a Gaussian process from model fitted by `georob`;
and finally
`sample.variogram` and `fit.variogram.model` for robust estimation and modelling of sample variograms.

georobPackage

The georob Package

Description

This is a summary of the features and functionality of **georob**, a package in **R** for robust geostatistical analyses.

Details

georob is a package for robust analyses of geostatistical data. Such data, say $y_i = y(\mathbf{s}_i)$, are recorded at a set of locations, $\mathbf{s}_i, i = 1, 2, \dots, n$, in a domain $G \in \mathbb{R}^d, d \in (1, 2, 3)$, along with covariate information $x_j(\mathbf{s}_i), j = 1, 2, \dots, p$.

Model: We use the following model for the data $y_i = y(\mathbf{s}_i)$:

$$Y(\mathbf{s}_i) = Z(\mathbf{s}_i) + \varepsilon = \mathbf{x}(\mathbf{s}_i)^T \boldsymbol{\beta} + B(\mathbf{s}_i) + \varepsilon_i,$$

where $Z(\mathbf{s}_i) = \mathbf{x}(\mathbf{s}_i)^T \boldsymbol{\beta} + B(\mathbf{s}_i)$ is the so-called signal, $\mathbf{x}(\mathbf{s}_i)^T \boldsymbol{\beta}$ is the external drift, $\{B(\mathbf{s})\}$ is an unobserved stationary or intrinsic spatial Gaussian random field with zero mean, and ε_i is an *i.i.d* error from a possibly long-tailed distribution with scale parameter τ (τ^2 is usually called nugget effect). In vector form the model is written as

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{B} + \boldsymbol{\varepsilon},$$

where \mathbf{X} is the model matrix with the rows $\mathbf{x}(\mathbf{s}_i)^T$.

The (generalized) covariance matrix of the vector of spatial Gaussian random effects \mathbf{B} is denoted by

$$E[\mathbf{B}\mathbf{B}^T] = \boldsymbol{\Gamma}_\theta = \sigma_n^2 \mathbf{I} + \sigma^2 \mathbf{V}_\alpha = \sigma_Z^2 \mathbf{V}_{\alpha, \xi} = \sigma_Z^2 ((1 - \xi) \mathbf{I} + \xi \mathbf{V}_\alpha),$$

where σ_n^2 is the variance of seemingly uncorrelated micro-scale variation in $B(\mathbf{s})$ that cannot be resolved with the chosen sampling design, σ^2 is the variance of the captured auto-correlated variation in $B(\mathbf{s})$, $\sigma_B^2 = \sigma_n^2 + \sigma^2$ is the signal variance, and $\xi = \sigma^2 / \sigma_B^2$. To estimate both σ_n^2 and τ^2 (and not only their sum), one needs replicated measurements for some of the \mathbf{s}_i .

We define \mathbf{V}_α to be the matrix with elements

$$(\mathbf{V}_\alpha)_{ij} = \gamma_0 - \gamma(|\mathbf{A}(\mathbf{s}_i - \mathbf{s}_j)|),$$

where the constant γ_0 is chosen large enough so that \mathbf{V}_α is positive definite, $\gamma(\cdot)$ is a valid stationary or intrinsic variogram, and $\mathbf{A} = \mathbf{A}(\alpha, f_1, f_2; \omega, \phi, \zeta)$ is a matrix that is used to model geometrically anisotropic auto-correlation. In more detail, \mathbf{A} maps an arbitrary point on an ellipsoidal

surface with constant semi-variance in \mathbb{R}^3 , centred on the origin, and having lengths of semi-principal axes, $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$, equal to $|\mathbf{p}_1| = \alpha$, $|\mathbf{p}_2| = f_1 \alpha$ and $|\mathbf{p}_3| = f_2 \alpha$, $0 < f_2 \leq f_1 \leq 1$, respectively, onto the surface of the unit ball centred on the origin.

The orientation of the ellipsoid is defined by the three angles ω , ϕ and ζ :

ω is the azimuth of \mathbf{p}_1 (= angle between north and the projection of \mathbf{p}_1 onto the x - y -plane, measured from north to south positive clockwise in degrees),

ϕ is 90 degrees minus the altitude of \mathbf{p}_1 (= angle between the zenith and \mathbf{p}_1 , measured from zenith to nadir positive clockwise in degrees), and

ζ is the angle between \mathbf{p}_2 and the direction of the line, say y' , defined by the intersection between the x - y -plane and the plane orthogonal to \mathbf{p}_1 running through the origin (ζ is measured from y' positive counter-clockwise in degrees).

The transformation matrix is given by

$$\mathbf{A} = \begin{pmatrix} 1/\alpha & 0 & 0 \\ 0 & 1/(f_1 \alpha) & 0 \\ 0 & 0 & 1/(f_2 \alpha) \end{pmatrix} (\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3,)$$

where

$$\mathbf{C}_1^T = (\sin \omega \sin \phi, -\cos \omega \cos \zeta - \sin \omega \cos \phi \sin \zeta, \cos \omega \sin \zeta - \sin \omega \cos \phi \cos \zeta)$$

$$\mathbf{C}_2^T = (\cos \omega \sin \phi, \sin \omega \cos \zeta - \cos \omega \cos \phi \sin \zeta, -\sin \omega \sin \zeta - \cos \omega \cos \phi \cos \zeta)$$

$$\mathbf{C}_3^T = (\cos \phi, \sin \phi \sin \zeta, \sin \phi \cos \zeta)$$

To model geometrically anisotropic variograms in \mathbb{R}^2 one has to set $\phi = 90$ and $f_2 = 1$, and for $f_1 = f_2 = 1$ one obtains the model for isotropic auto-correlation with range parameter α . Note that for isotropic auto-correlation the software processes data for which d may exceed 3.

Two remarks are in order:

1. Clearly, the (generalized) covariance matrix of the observations \mathbf{Y} is given by

$$\text{Cov}[\mathbf{Y}, \mathbf{Y}^T] = \tau^2 \mathbf{I} + \mathbf{\Gamma}_\theta.$$

2. Depending on the context, the term ‘‘variogram parameters’’ denotes sometimes all parameters of a geometrically anisotropic variogram model, but in places only the parameters of an isotropic variogram model, i.e. $\sigma^2, \dots, \alpha, \dots$ and f_1, \dots, ζ are denoted by the term ‘‘anisotropy parameters’’. In the sequel $\boldsymbol{\theta}$ is used to denote all variogram and anisotropy parameters except the nugget effect τ^2 .

Estimation: The unobserved spatial random effects \mathbf{B} at the data locations \mathbf{s}_i and the model parameters $\boldsymbol{\beta}$, τ^2 and $\boldsymbol{\theta}^T = (\sigma^2, \sigma_n^2, \alpha, \dots, f_1, f_2, \omega, \phi, \zeta)$ are unknown and are estimated in **georob** either by Gaussian or robust restricted maximum likelihood (REML) or Gaussian maximum likelihood (ML). Here \dots denote further parameters of the variogram such as the smoothness parameter of the Whittle-Matérn model.

In brief, the robust REML method is based on the insight that for given $\boldsymbol{\theta}$ and τ^2 the Kriging predictions (= BLUP) of \mathbf{B} and the generalized least squares (GLS = ML) estimates of $\boldsymbol{\beta}$ can be obtained simultaneously by maximizing

$$-\sum_i \left(\frac{y_i - \mathbf{x}(\mathbf{s}_i)^T \boldsymbol{\beta} - B(\mathbf{s}_i)}{\tau} \right)^2 - \mathbf{B}^T \mathbf{\Gamma}_\theta^{-1} \mathbf{B}$$

with respect to \mathbf{B} and $\boldsymbol{\beta}$ e.g. *Harville (1977)*.

Hence, the BLUP of \mathbf{B} , ML estimates of $\boldsymbol{\beta}$, $\boldsymbol{\theta}$ and τ^2 are obtained by maximizing

$$-\log(\det(\tau^2 \mathbf{I} + \boldsymbol{\Gamma}_\theta)) - \sum_i \left(\frac{y_i - \mathbf{x}(s_i)^\top \boldsymbol{\beta} - B(s_i)}{\tau} \right)^2 - \mathbf{B}^\top \boldsymbol{\Gamma}_\theta^{-1} \mathbf{B}$$

jointly with respect to \mathbf{B} , $\boldsymbol{\beta}$, $\boldsymbol{\theta}$ and τ^2 or by solving the respective estimating equations.

The estimating equations can then be robustified by

- replacing the standardized residuals, say ε_i/τ , by a bounded or re-descending ψ -function, $\psi_c(\varepsilon_i/\tau)$, of them (e.g. *Marona et al, 2006, chap. 2*) and by
- introducing suitable bias correction terms for Fisher consistency at the Gaussian model,

see *Kuensch et al. (2011)* for details. The robustified estimating equations are solved numerically by a combination of iterated re-weighted least squares (IRWLS) to estimate \mathbf{B} and $\boldsymbol{\beta}$ for given $\boldsymbol{\theta}$ and τ^2 and nonlinear root finding by the function `nleqslv` of the R package `nleqslv` to get $\boldsymbol{\theta}$ and τ^2 . The robustness of the procedure is controlled by the tuning parameter c of the ψ_c -function. For $c \geq 1000$ the algorithm computes Gaussian (RE)ML estimates and customary plug-in Kriging predictions. Instead of solving the Gaussian (RE)ML estimating equations, our software then maximizes the Gaussian (restricted) log-likelihood using `nlminb` or `optim`.

`georob` uses variogram models implemented in the R package `RandomFields` (see `RMmodel`).

Currently, estimation of the parameters of the following models is implemented:

"RMaskey", "RMBessel", "RMcauchy", "RMcircular", "RMcubic", "RMDagum",
 "RMDampedcos", "RMdewijsian", "RMexp" (default), "RMfbm", "RMgauss",
 "RMgencauchy", "RMgenfbm", "RMgengneiting", "RMgneiting", "RMLgd",
 "RMmatern", "RMpenta", "RMqexp", "RMspheric", "RMstable", "RMwave",
 "RMwhittle".

For most variogram parameters, closed-form expressions of $\partial\gamma/\partial\theta_i$ are used in the computations. However, for the parameter ν of the models "RMBessel", "RMmatern" and "RMwhittle" $\partial\gamma/\partial\nu$ is evaluated numerically by the function `numericDeriv`, and this results in an increase in computing time when ν is estimated.

Prediction: Robust plug-in external drift point Kriging predictions can be computed for an non-sampled location s_0 from the covariates $\mathbf{x}(s_0)$, the estimated parameters $\hat{\boldsymbol{\beta}}$, $\hat{\boldsymbol{\theta}}$ and the predicted random effects $\hat{\mathbf{B}}$ by

$$\hat{Y}(s_0) = \hat{Z}(s_0) = \mathbf{x}(s_0)^\top \hat{\boldsymbol{\beta}} + \boldsymbol{\gamma}_\theta^\top(s_0) \boldsymbol{\Gamma}_\theta^{-1} \hat{\mathbf{B}},$$

where $\boldsymbol{\Gamma}_\theta$ is the estimated (generalized) covariance matrix of \mathbf{B} and $\boldsymbol{\gamma}_\theta(s_0)$ is the vector with the estimated (generalized) covariances between \mathbf{B} and $B(s_0)$. Kriging variances can be computed as well, based on approximated covariances of $\hat{\mathbf{B}}$ and $\hat{\boldsymbol{\beta}}$ (see *Kuensch et al., 2011*, and appendices of *Nussbaum et al., 2014*, for details).

The package `georob` provides in addition software for computing robust external drift *block* Kriging predictions. The required integrals of the generalized covariance function are computed by functions of the R package `constrainedKriging`.

Functionality: For the time being, the functionality of `georob` is limited to robust geostatistical analyses of *single* response variables. No software is currently available for robust multivariate geostatistical analyses. `georob` offers functions for:

1. Robustly fitting a spatial linear model to data that are possibly contaminated by independent errors from a long-tailed distribution by robust REML (see functions `georob` — which also fits such models efficiently by Gaussian (RE)ML — `profilelogLik` and `control.georob`).
2. Extracting estimated model components (see `residuals.georob`, `rstandard.georob`, `ranef.georob`).
3. Robustly estimating sample variograms and for fitting variogram model functions to them (see `sample.variogram` and `fit.variogram.model`).
4. Model building by forward and backward selection of covariates for the external drift (see `waldtest.georob`, `step.georob`, `add1.georob`, `drop1.georob`, `extractAIC.georob`, `logLik.georob`, `deviance.georob`). For a robust fit, the log-likelihood is not defined. The function then computes the (restricted) log-likelihood of an equivalent Gaussian model with heteroscedastic nugget (see `deviance.georob` for details).
5. Assessing the goodness-of-fit and predictive power of the model by K -fold cross-validation (see `cv.georob` and `validate.predictions`).
6. Computing robust external drift point and block Kriging predictions (see `predict.georob`, `control.predict.georob`).
7. Unbiased back-transformation of both point and block Kriging predictions of log-transformed data to the original scale of the measurements (see `lgpp`).

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

References

- Nussbaum, M., Papritz, A., Baltensweiler, A. and Walthert, L. (2014) Estimating soil organic carbon stocks of Swiss forest soils by robust external-drift kriging. *Geoscientific Model Development*, 7, 1197–1210. doi: [10.5194/gmd711972014](https://doi.org/10.5194/gmd711972014), <https://gmd.copernicus.org/articles/7/1197/2014/>.
- Künsch, H. R., Papritz, A., Schwierz, C. and Stahel, W. A. (in preparation) Robust Geostatistics.
- Künsch, H. R., Papritz, A., Schwierz, C. and Stahel, W. A. (2011) Robust estimation of the external drift and the variogram of spatial data. Proceedings of the ISI 58th World Statistics Congress of the International Statistical Institute. doi: [10.3929/ethz009900710](https://doi.org/10.3929/ethz009900710)
- Maronna, R. A., Martin, R. D. and Yohai, V. J. (2006) Robust Statistics Theory and Methods, John Wiley & Sons.

See Also

`georob` for (robust) fitting of spatial linear models;
`georobObject` for a description of the class `georob`;
`profilelogLik` for computing profiles of Gaussian likelihoods;
`plot.georob` for display of RE(ML) variogram estimates;
`control.georob` for controlling the behaviour of `georob`;
`georobModelBuilding` for stepwise building models of class `georob`;

[cv.georob](#) for assessing the goodness of a fit by georob;
[georobMethods](#) for further methods for the class georob;
[predict.georob](#) for computing robust Kriging predictions;
[lgnpp](#) for unbiased back-transformation of Kriging prediction of log-transformed data;
[georobSimulation](#) for simulating realizations of a Gaussian process from model fitted by georob;
 and finally
[sample.variogram](#) and [fit.variogram.model](#) for robust estimation and modelling of sample variograms.

georobS3methods *Common S3 Methods for Class georob*

Description

This page documents the methods `coef`, `fixef`, `fixed.effects`, `model.frame`, `model.matrix`, `nobs`, `print`, `ranef`, `random.effects`, `resid`, `residuals`, `rstandard`, `summary` and `vcov` for the class `georob` which extract the respective components or summarize a `georob` object.

Usage

```

## S3 method for class 'georob'
coef(object, what = c("trend", "variogram"), ...)

## S3 method for class 'georob'
fixef(object, ...)

## S3 method for class 'georob'
fixed.effects(object, ...)

## S3 method for class 'georob'
model.frame(formula, ...)

## S3 method for class 'georob'
model.matrix(object, ...)

## S3 method for class 'georob'
nobs(object, ...)

## S3 method for class 'georob'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'georob'
ranef(object, standard = FALSE, ...)

## S3 method for class 'georob'

```

```

random.effects(object, standard = FALSE, ...)

## S3 method for class 'georob'
resid(object,
      type = c("working", "response", "deviance", "pearson", "partial"),
      terms = NULL,
      level = 1, ...)

## S3 method for class 'georob'
residuals(object,
          type = c("working", "response", "deviance", "pearson", "partial"),
          terms = NULL,
          level = 1, ...)

## S3 method for class 'georob'
rstandard(model, level = 1, ...)

## S3 method for class 'georob'
summary(object, correlation = FALSE, signif = 0.95, ...)

## S3 method for class 'georob'
vcov(object, ...)

```

Arguments

object, model, x	an object of class georob, see georobObject .
formula	a model formula or terms object or an object of class georob, see georobObject .
correlation	logical controlling whether the correlation matrix of the estimated regression coefficients and of the fitted variogram parameters (only for non-robust fits) is computed (default FALSE).
digits	positive integer indicating the number of decimal digits to print.
level	an optional integer giving the level for extracting the residuals from object. level = 0 extracts the regression residuals $\hat{B}(s) + \hat{\varepsilon}(s)$ and level = 1 (default) only the estimated errors $\hat{\varepsilon}(s)$.
signif	confidence level for computing confidence intervals for variogram parameters (default 0.95).
standard	logical controlling whether the spatial random effects B should be standardized (default FALSE).
type	character keyword indicating the type of residuals to compute, see residuals.lm . type = "huber" computes 'huberized' residuals $\hat{\sigma}/\gamma_1\psi(\hat{\varepsilon}(s)/\hat{\sigma})$.
terms	If type = "terms", which terms (default is all terms).
what	If what = "trend" (default) the function coef extracts the coefficients of the trend model and for what = "variogram" the variogram parameters.
...	additional arguments passed to methods.

Details

For robust REML fits deviance returns (possibly with a warning) the deviance of the Gaussian REML fit of the equivalent Gaussian spatial linear model with heteroscedastic nugget.

The methods `model.frame`, `model.matrix` and `nobs` extract the model frame, model matrix and the number of observations, see help pages of respective generic functions.

The methods `residuals` (and `resid`) extract either the estimated independent errors $\hat{\varepsilon}(s)$ or the sum of the latter quantities and the spatial random effects $\hat{B}(s)$. `rstandard` does the same but standardizes the residuals to unit variance. `ranef` (`random.effects`) extracts the spatial random effects with the option to standardize them as well, and `fixef` (`fixed.effects`) extracts the fitted fixed-effects regression coefficients, which may of course also be obtained by `coef`.

For Gaussian REML the method `summary` computes confidence intervals of the estimated variogram and anisotropy parameters from the Hessian matrix of the (restricted) log-likelihood (= observed Fisher information), based on the asymptotic normal distribution of (RE)ML estimates. Note that the Hessian matrix with respect to the *transformed* variogram and anisotropy parameters is used for this. Hence the inverse Hessian matrix is the covariance matrix of the transformed parameters, confidence intervals are first computed for the transformed parameters and the limits of these intervals are transformed back to the original scale of the parameters. Optionally, `summary` reports the correlation matrix of the *transformed* parameters, also computed from the Hessian matrix.

Note that the methods `coef` and `summary` generate objects of class `coef.georob` and `summary.georob`, respectively, for which only `print` methods are available.

Besides, the default methods of the generic functions `confint`, `df.residual`, `fitted`, `formula`, `termplot` and `update` can be used for objects of class `georob`.

Value

The methods `fixef`, `fixed.effects` and `coef` return the numeric vector of estimated fixed-effects regression coefficients or variogram and anisotropy parameters (`coef` only), and `vcov` returns the covariance matrix of the estimated regression coefficients.

The methods `resid`, `residuals` and `rstandard` return numeric vectors of (standardized) residuals, and `ranef` and `random.effects` the numeric vector of (standardized) spatial random effects, see *Details*.

The methods `model.frame` and `model.matrix` return a model frame and the fixed-effects model matrix, respectively, and `nobs` returns the number of observations used to fit a spatial linear model.

The method `summary` generates a list with components extracted directly from object (`call`, `residuals`, `bhat`, `rweights`, `converged`, `convergence.code`, `iter`, `loglik`, `variogram.object`, `gradient`, `tuning.psi`, `df.residual`, `control`, `terms`) and complements the list by the following components:

`scale` the square root of the estimated nugget effect τ^2 .

`coefficients` a 4-column matrix with estimated regression coefficients, their standard errors *t*-statistics and corresponding (two-sided) *p*-values.

`correlation` an optional lower-triagonal matrix with the Pearson correlation coefficients of the estimated regression coefficients.

`param.aniso` either a vector (robust REML) or a 3-column matrix (Gaussian REML) with estimated variogram and anisotropy parameters, complemented for Gaussian REML with confidence limits, see *Details*.

`cor.tf.param` an optional lower-triagonal matrix with the Pearson correlation coefficients of estimated transformed variogram and anisotropy parameters, see *Details*.
`se.residuals` a vector with the standard errors of the estimated ε .

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms;
[georob](#) for (robust) fitting of spatial linear models;
[georobObject](#) for a description of the class `georob`;
[profilelogLik](#) for computing profiles of Gaussian likelihoods;
[plot.georob](#) for display of RE(ML) variogram estimates;
[control.georob](#) for controlling the behaviour of `georob`;
[georobModelBuilding](#) for stepwise building models of class `georob`;
[cv.georob](#) for assessing the goodness of a fit by `georob`;
[predict.georob](#) for computing robust Kriging predictions;
[lgnpp](#) for unbiased back-transformation of Kriging prediction of log-transformed data;
[georobSimulation](#) for simulating realizations of a Gaussian process from model fitted by `georob`;
and finally
[sample.variogram](#) and [fit.variogram.model](#) for robust estimation and modelling of sample variograms.

Examples

```
## Not run:

data(meuse)

## Gaussian REML fit
r.logzn.reml <- georob(log(zinc) ~ sqrt(dist), data = meuse, locations = ~ x + y,
  variogram.model = "RMexp",
  param = c(variance = 0.15, nugget = 0.05, scale = 200),
  tuning.psi = 1000)
summary(r.logzn.reml, correlation = TRUE)

## robust REML fit
r.logzn.rob <- update(r.logzn.reml, tuning.psi = 1)

summary(r.logzn.rob, correlation = TRUE)

## residual diagnostics
old.par <- par(mfrow = c(2,3))

plot(fitted(r.logzn.reml), rstandard(r.logzn.reml))
```

```

abline(h = 0, lty = "dotted")
qqnorm(rstandard(r.logzn.reml))
abline(0, 1)
qqnorm(ranef(r.logzn.reml, standard = TRUE))
abline(0, 1)
plot(fitted(r.logzn.rob), rstandard(r.logzn.rob))
abline(h = 0, lty = "dotted")
qqnorm(rstandard(r.logzn.rob))
abline(0, 1)
qqnorm(ranef(r.logzn.rob, standard = TRUE))
abline(0, 1)

par(old.par)

## End(Not run)

```

georobSimulation	<i>Simulating Realizations of Gaussian Processes from Object of Class georob</i>
------------------	--

Description

This page documents the function `condsim` that simulates (un)conditional realizations of Gaussian processes from the parameters of a spatial linear model estimated by the function [georob](#).

Usage

```

condsim(object, newdata, nsim, seed,
        type = c("response", "signal"), locations, trend.coef = NULL,
        variogram.model = NULL, param = NULL, aniso = NULL, variogram.object = NULL,
        control = control.condsim(), verbose = 0)

```

```

control.condsim(use.grid = FALSE, grid.refinement = 2.,
               condsim = TRUE, include.data.sites = FALSE, means = FALSE,
               trend.covariates = FALSE, covariances = FALSE,
               ncores = detectCores(), pcmp = control.pcmp())

```

Arguments

object	an object of class <code>georob</code> (mandatory argument), see georobObject .
newdata	a mandatory data frame, SpatialPointsDataFrame , SpatialPixelsDataFrame , SpatialGridDataFrame , SpatialPoints , SpatialPixels or SpatialGrid object, with the coordinates of points for which simulations are computed and in which to look for variables required for computing fitted values or Kriging predictions, see predict.georob .
nsim	number of (conditional) realizations to compute (mandatory argument).
seed	integer seed to initialize random number generation, see set.seed (mandatory argument).

type	<p>character keyword defining what target quantity should be simulated. Possible values are</p> <ul style="list-style-type: none"> • "signal": the "signal" $Z(\mathbf{s}) = \mathbf{x}(\mathbf{s})^T \boldsymbol{\beta} + B(\mathbf{s})$ of the process, • "response": the observations $Y(\mathbf{s}) = Z(\mathbf{s}) + \varepsilon(\mathbf{s})$, (default), <p>see georobIntro for details on the model specification.</p>
locations	<p>an optional one-sided formula specifying what variables of newdata are the coordinates of the points for which simulated values are computed (default: <code>object[["locations.objects"]][["locations"]]</code>).</p>
trend.coef	<p>an optional numeric vector with the coefficients of the trend model to be used for computing the (conditional) mean function of the random process see <i>Details</i>.</p>
variogram.model	<p>an optional character keyword defining the variogram model to be used for the simulations, see georob and <i>Details</i>.</p>
param	<p>an optional named numeric vector with values of the variogram parameters used for the simulations, see georob and <i>Details</i>.</p>
aniso	<p>an optional named numeric vector with values of anisotropy parameters of a variogram used for the simulations, see georob and <i>Details</i>.</p>
variogram.object	<p>an optional list that defines a possibly nested variogram model used for the simulations, see georob and <i>Details</i>.</p>
control	<p>a list with the components <code>use.grid</code>, <code>grid.refinement</code>, <code>condsim</code>, <code>include.data.sites</code>, <code>means</code>, <code>trend.covariates</code>, <code>covariances</code>, <code>ncores</code>, and <code>pcmp</code> or a function such as <code>control.condsim</code> that generates such a list, see arguments of <code>control.condsim</code> for details.</p>
verbose	<p>positive integer controlling logging of diagnostic messages to the console. <code>verbose = 0</code> (default) suppresses such messages.</p>
use.grid	<p>logical scalar (default FALSE) to control whether (conditional) realizations are computed for a rectangular grid instead of the coordinates of points contained in newdata, see <i>Details</i>.</p>
grid.refinement	<p>numeric scalar that defines a factor by which the minimum differences of the coordinates between any pair of points in newdata are divided to setup the simulation grid, should be > 1 (default 2), see <i>Details</i>.</p>
condsim	<p>logical scalar (default TRUE) to control whether conditional (TRUE) or unconditional simulations (FALSE) are computed.</p>
include.data.sites	<p>logical scalar, to control whether (conditionally) simulated values are returned also for the points of the original data set used to estimate the model parameters.</p>
means	<p>logical scalar, to control whether the (un)conditional means are included in the output.</p>
trend.covariates	<p>logical scalar, to control whether the covariates required for the trend model are included in the output.</p>

covariances	logical scalar, to control whether the covariances between the points of the original data set used to estimate the model parameters (<code>attr gcvmat.d.d</code> , <code>compressed</code> matrix) and the covariances between the simulation and the original data points (<code>attr gcvmat.s.d</code> , matrix) are returned as attributes of the output. Note that these covariances are only returned if <code>use.grid == TRUE & condsim == TRUE</code> .
ncores	positive integer controlling how many cores are used for parallelized computations, defaults to all cores.
pcmp	a list of arguments, passed e.g. to <code>pmm</code> or a function such as <code>control.pcmp</code> that generates such a list (see <code>control.pcmp</code> for allowed arguments).

Details

`condsim` (conditionally) simulates from a Gaussian process that has a linear mean function with parameters β and an auto-correlation structure characterized by a parametric variogram model and variogram parameters τ^2 and θ (see `georobIntro` for the employed parametrization of the spatial linear model). The parameters of the mean and auto-correlation function are either taken from the the spatial linear model estimated by `georob` and passed by the argument object to `condsim` or from the optional arguments `trend.coef` (β) and `variogram.model`, `param`, `aniso` or `variogram.object` (τ^2 , θ). Note that in the former case the uncertainty in the estimated mean and auto-correlation parameters is not taken into account.

Simulated values are computed for the points in `newdata` by the function `RFsimulate` of the package **RandomFields**. Both unconditional and conditional simulations can be computed. In the latter cases, the simulated values are always conditioned to the response data used to fit the spatial linear model by `georob` and contained in `object`.

Unconditional simulation:

Unconditional realizations are either computed for the exact locations of the points in `newdata` (`use.grid == FALSE`), irrespective of the fact whether these are arranged on a regular grid, or for the (approximate) locations of the points in `newdata` matched to a rectangular simulation grid (`use.grid == TRUE`). The latter approach may be substantially faster for large problems because the simulation algorithm implemented in `RFsimulate` for grids is faster than for arbitrary geometries of the simulation points.

For `use.grid == TRUE`, a rectangular grid is constructed from the coordinates of the points in `newdata` and `object`. The spacing of the grid is equal to the minimum distance between any pair of points in `newdata`, divided by `grid.refinement`. The data related to the points in `newdata` (covariates for the trend model) and of the data in `object` (response values, covariates) are then assigned to the nodes of the grid that are closest to the respective points. If the same grid node is assigned to several points in `newdata` (or in `object`) then the data of the respective points are averaged. If the same node is assigned to a point in `object` and `newdata` then the point in `object` is kept and the concerned point in `newdata` is omitted.

Conditional simulation:

Simulations are conditioned to data either by exploiting the respective built-in functionality of `RFsimulate` (`use.grid == FALSE`) or by the Kriging method (`use.grid == TRUE`, see Chilès and Delfiner, 1999, sec. 7.3). The latter approach may again be faster for large problems because it exploits the larger speed of unconditional simulations for rectangular grids.

Parallelized computations:

condsim uses the packages **parallel**, **snow** and **snowfall** for parallelized computation of simulations. If there are m realizations to simulate, the task is split into $\text{ceiling}(m/ncores)$ sub-tasks that are then distributed to $ncores$ CPUs. Evidently, $ncores = 1$ suppresses parallel execution. By default, the function uses all available CPUs as returned by [detectCores](#).

Value

The output generated by condsim is an object of a “similar” class as newdata (data frame, [SpatialPointsDataFrame](#), [SpatialPixelsDataFrame](#), [SpatialGridDataFrame](#), [SpatialPolygonsDataFrame](#)).

The data frame or the data slot of the `Spatial...DataFrame` objects have the following components:

- the coordinates of the prediction points (only present if newdata is a data frame).
- expct: optionally the (un)conditional means.
- optionally the covariates required for the trend model.
- sim.1, sim.2, ...: the (un)conditionally simulated realizations.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

References

Chilès, J.-P. and Delfiner, P. (1999) *Geostatistics: Modeling Spatial Uncertainty*, John Wiley & Sons, New York.

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms;
[georob](#) for (robust) fitting of spatial linear models;
[georobObject](#) for a description of the class georob;
[profilelogLik](#) for computing profiles of Gaussian likelihoods;
[plot.georob](#) for display of RE(ML) variogram estimates;
[control.georob](#) for controlling the behaviour of georob;
[georobModelBuilding](#) for stepwise building models of class georob;
[cv.georob](#) for assessing the goodness of a fit by georob;
[georobMethods](#) for further methods for the class georob;
[predict.georob](#) for computing robust Kriging predictions;
[lgnp](#) for unbiased back-transformation of Kriging prediction of log-transformed data;
[sample.variogram](#) and [fit.variogram.model](#) for robust estimation and modelling of sample variograms.

Examples

```

## Not run:

data(meuse)
data(meuse.grid)

## convert to SpatialGridDataFrame
meuse.grid.sgdf <- meuse.grid
coordinates(meuse.grid.sgdf) <- ~ x + y
gridded(meuse.grid.sgdf) <- TRUE
fullgrid(meuse.grid.sgdf) <- TRUE

## Gaussian REML fit
r.logzn.reml <- georob(log(zinc) ~ sqrt(dist), data = meuse, locations = ~ x + y,
  variogram.model = "RMexp",
  param = c(variance = 0.15, nugget = 0.05, scale = 200),
  tuning.psi = 1000)

## Conditional simulations
r.sim <- condsim(r.logzn.reml, newdata = meuse.grid.sgdf, nsim = 100, seed = 1,
  control = control.condsim(ncores = 1))
## using multiple cores
## r.sim <- condsim(r.logzn.reml, newdata = meuse.grid.sgdf, nsim = 100, seed = 1)
str(r.sim, max=2)

## Display
spplot(r.sim, zcol = "sim.1", at = seq(3.5, 8.5, by = 0.5))
spplot(r.sim, zcol = "sim.2", at = seq(3.5, 8.5, by = 0.5))

library(lattice)
levelplot(sim.1 ~ x + y, as.data.frame(r.sim), aspect = "iso", at = seq(3.5, 8.5, by = 0.5),
  panel = function(x, y, z, subscripts, data.points, ... ){
    panel.levelplot( x, y, z, subscripts, ...)
    panel.xyplot(data.points$x, data.points$y, col = 1)
  }, data.points = meuse[, c("x", "y")])
)
levelplot(sim.2 ~ x + y, as.data.frame(r.sim), aspect = "iso", at = seq(3.5, 8.5, by = 0.5),
  panel = function(x, y, z, subscripts, data.points, ... ){
    panel.levelplot( x, y, z, subscripts, ...)
    panel.xyplot(data.points$x, data.points$y, col = 1)
  }, data.points = meuse[, c("x", "y")])
)

## End(Not run)

```

Description

The function `lgnpp` back-transforms point or block Kriging predictions of a log-transformed response variable computed by `predict.georob`. Alternatively, the function averages log-normal point Kriging predictions for a block and approximates the mean squared prediction error of the block mean.

Usage

```
lgnpp(object, newdata, locations, is.block = FALSE, all.pred = NULL,
       extended.output = FALSE)
```

Arguments

<code>object</code>	an object with Kriging predictions of a log-transformed response variable as obtained by <code>predict(georob-object,...)</code> .
<code>newdata</code>	an optional object as passed as argument <code>newdata</code> to <code>predict.georob</code> , see <i>Details</i> .
<code>locations</code>	an optional one-sided formula specifying what variables of <code>newdata</code> are the coordinates of the prediction points, see <code>predict.georob</code> .
<code>is.block</code>	an optional logical (default <code>FALSE</code>) specifying whether point predictions contained in <code>object</code> are considered to belong to a single block and should be averaged after back-transformation. Ignored if <code>object</code> contains block Kriging predictions, see <i>Details</i> .
<code>all.pred</code>	an optional positive integer or an object as obtained by <code>lgnpp(predict(georob-object,...))</code> , see <i>Details</i> .
<code>extended.output</code>	logical controlling whether the covariance matrix of the errors of the back-transformed point predictions is added as an attribute to the result, see <i>Details</i> .

Details

The function `lgnpp` performs three tasks:

1. Back-transformation of point Kriging predictions of a log-transformed response:

The usual, marginally unbiased back-transformation for log-normal point Kriging is used:

$$\widehat{U}(\mathbf{s}) = \exp(\widehat{Z}(\mathbf{s}) + 1/2(\text{Var}_{\hat{\theta}}[Z(\mathbf{s})] - \text{Var}_{\hat{\theta}}[\widehat{Z}(\mathbf{s})])),$$

$$\text{Cov}_{\hat{\theta}}[U(\mathbf{s}_i) - \widehat{U}(\mathbf{s}_i), U(\mathbf{s}_j) - \widehat{U}(\mathbf{s}_j)] = \mu_{\hat{\theta}}(\mathbf{s}_i)\mu_{\hat{\theta}}(\mathbf{s}_j)$$

$$\times \{\exp(\text{Cov}_{\hat{\theta}}[Z(\mathbf{s}_i), Z(\mathbf{s}_j)]) - 2 \exp(\text{Cov}_{\hat{\theta}}[\widehat{Z}(\mathbf{s}_i), Z(\mathbf{s}_j)]) + \exp(\text{Cov}_{\hat{\theta}}[\widehat{Z}(\mathbf{s}_i), \widehat{Z}(\mathbf{s}_j)])\},$$

where \widehat{Z} and \widehat{U} denote the log- and back-transformed predictions of the signal, and

$$\mu_{\hat{\theta}}(\mathbf{s}) \approx \exp(\mathbf{x}(\mathbf{s})^T \widehat{\boldsymbol{\beta}} + 1/2 \text{Var}_{\hat{\theta}}[Z(\mathbf{s})]).$$

The expressions for the required covariance terms can be found in the Appendices of Nussbaum et al. (2014). Instead of the signal $Z(s)$, predictions of the log-transformed response $Y(s)$ or the estimated trend $\mathbf{x}(s)^T \hat{\boldsymbol{\beta}}$ of the log-transformed data can be back-transformed (see [georobIntro](#)). The above transformations are used if object contains point Kriging predictions (see `predict.georob, Value`) and if `is.block = FALSE` and `all.pred` is missing.

2. Back-transformation of block Kriging predictions of a log-transformed response:

Block Kriging predictions of a log-transformed response variable are back-transformed by the approximately unbiased transformation proposed by Cressie (2006, Appendix C)

$$\hat{U}(A) = \exp(\hat{Z}(A) + 1/2\{\text{Var}_{\hat{\theta}}[Z(s)] + \hat{\boldsymbol{\beta}}^T \mathbf{M}(A) \hat{\boldsymbol{\beta}} - \text{Var}_{\hat{\theta}}[\hat{Z}(A)]\}),$$

$$E_{\hat{\theta}}[\{U(A) - \hat{U}(A)\}^2] = \mu_{\hat{\theta}}(A)^2 \{\exp(\text{Var}_{\hat{\theta}}[Z(A)]) - 2 \exp(\text{Cov}_{\hat{\theta}}[\hat{Z}(A), Z(A)]) + \exp(\text{Var}_{\hat{\theta}}[\hat{Z}(A)])\}$$

where $\hat{Z}(A)$ and $\hat{U}(A)$ are the log- and back-transformed predictions of the block mean $U(A)$, respectively, $\mathbf{M}(A)$ is the spatial covariance matrix of the covariates

$$\mathbf{M}(A) = 1/|A| \int_A (\mathbf{x}(s) - \mathbf{x}(A))(\mathbf{x}(s) - \mathbf{x}(A))^T ds$$

within the block A where

$$\mathbf{x}(A) = 1/|A| \int_A \mathbf{x}(s) ds$$

and

$$\mu_{\hat{\theta}}(A) \approx \exp(\mathbf{x}(A)^T \hat{\boldsymbol{\beta}} + 1/2 \text{Var}_{\hat{\theta}}[Z(A)]).$$

This back-transformation is based on the assumption that both the point data $U(s)$ and the block means $U(A)$ follow log-normal laws, which strictly cannot hold. But for small blocks the assumption works well as the bias and the loss of efficiency caused by this assumption are small (Cressie, 2006; Hofer et al., 2013).

The above formulae are used by `lgnpp` if object contains block Kriging predictions in the form of a `SpatialPolygonsDataFrame`. To approximate $\mathbf{M}(A)$, one needs the covariates on a fine grid for the whole study domain in which the blocks lie. The covariates are passed `lgnpp` as argument `newdata`, where `newdata` can be any spatial data frame accepted by `predict.georob`. For evaluating $\mathbf{M}(A)$ the geometry of the blocks is taken from the polygons slot of the `SpatialPolygonsDataFrame` passed as object to `lgnpp`.

3. Back-transformation and averaging of point Kriging predictions of a log-transformed response:

`lgnpp` allows as a further option to back-transform and *average* point Kriging predictions passed as object to the function. One then assumes that the predictions in object refer to points that lie in a *single* block. Hence, one uses the approximation

$$\hat{U}(A) \approx \frac{1}{K} \sum_{s_i \in A} \hat{U}(s_i)$$

to predict the block mean $U(A)$, where K is the number of points in A . The mean squared prediction error can be approximated by

$$E_{\hat{\theta}}[\{U(A) - \hat{U}(A)\}^2] \approx \frac{1}{K^2} \sum_{s_i \in A} \sum_{s_j \in A} \text{Cov}_{\hat{\theta}}[U(s_i) - \hat{U}(s_i), U(s_j) - \hat{U}(s_j)].$$

In most instances, the evaluation of the above double sum is not feasible because a large number of points is used to discretize the block A . lgnpp then uses the following approximations to compute the mean squared error (see also Appendix E of Nussbaum et al., 2014):

- Point prediction results are passed as object to lgnpp only for a *random sample of points in* A (of size k), for which the evaluation of the above double sum is feasible.
- The prediction results for the *complete set of points* within the block are passed as argument `all.pred` to lgnpp. These results are used to compute $\hat{U}(A)$.
- The mean squared error is then approximated by

$$E_{\hat{\theta}}[\{U(A) - \hat{U}(A)\}^2] \approx \frac{1}{K^2} \sum_{s_i \in A} E_{\hat{\theta}}[\{U(s_i) - \hat{U}(s_i)\}^2] \\ + \frac{K-1}{Kk(k-1)} \sum_{s_i \in \text{sample}} \sum_{s_j \in \text{sample}, s_j \neq s_i} \text{Cov}_{\hat{\theta}}[U(s_i) - \hat{U}(s_i), U(s_j) - \hat{U}(s_j)].$$

The first term of the RHS (and $\hat{U}(A)$) can be computed from the point Kriging results contained in `all.pred`, and the double sum is evaluated from the full covariance matrices of the predictions and the respective targets, passed to lgnpp as object (one has to use the arguments `control=control.predict.georob(full.covmat=TRUE)` for `predict.georob` when computing the point Kriging predictions stored in object).

- If the prediction results are not available for the complete set of points in A then `all.pred` may be equal to K . The block mean is then approximated by

$$\hat{U}(A) \approx \frac{1}{k} \sum_{s_i \in \text{sample}} \hat{U}(s_i)$$

and the first term of the RHS of the expression for the mean squared error by

$$\frac{1}{kK} \sum_{s_i \in \text{sample}} E_{\hat{\theta}}[\{U(s_i) - \hat{U}(s_i)\}^2].$$

- By drawing samples repeatedly and passing the related Kriging results as object to lgnpp, one can reduce the error of the approximation of the mean squared error.

Value

If `is.block` is FALSE and `all.pred` is equal to NULL an updated object of the same class as object (see section *Value* of `predict.georob`). The data frame with the point or block Kriging predictions is complemented by lgnpp with the following new components:

- `lgn.pred`: the back-transformed Kriging predictions of a log-transformed response.

- `lgn.se`: the standard errors of the back-transformed predictions.
- `lgn.lower`, `lgn.upper`: the bounds of the back-transformed prediction intervals.

If `is.block` is TRUE or `all.pred` not equal to NULL a named numeric vector with two elements:

- `mean`: the back-transformed block Kriging estimate, see *Details*.
- `se`: the (approximated) block Kriging standard error, see *Details*.

If `extended.output` is TRUE then the vector is supplemented with the attribute `mse.lgn.pred` that contains the full covariance matrix of the back-transformed point prediction errors.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

References

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

Hofer, C., Borer, F., Bono, R., Kayser, A. and Papritz, A. 2013. Predicting topsoil heavy metal content of parcels of land: An empirical validation of customary and constrained lognormal block Kriging and conditional simulations. *Geoderma*, **193–194**, 200–212.

Nussbaum, M., Papritz, A., Baltensweiler, A. and Walthert, L. (2014) Estimating soil organic carbon stocks of Swiss forest soils by robust external-drift kriging. *Geoscientific Model Development*, **7**, 1197–1210. doi: [10.5194/gmd711972014](https://doi.org/10.5194/gmd711972014), <https://gmd.copernicus.org/articles/7/1197/2014/>.

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms;

[georob](#) for (robust) fitting of spatial linear models;

[predict.georob](#) for computing robust Kriging predictions.

Examples

```
## Not run:
data(meuse)

data(meuse.grid)
coordinates(meuse.grid) <- ~x+y
meuse.grid.pixdf <- meuse.grid
gridded(meuse.grid.pixdf) <- TRUE

library(constrainedKriging)
data(meuse.blocks)

r.logzn.rob <- georob(log(zinc) ~ sqrt(dist), data = meuse, locations = ~ x + y,
  variogram.model = "RMexp", param = c(variance = 0.15, nugget = 0.05, scale = 200),
  tuning.psi = 1., control = control.georob(cov.bhat = TRUE, full.cov.bhat = TRUE))
```

```

## point predictions of log(Zn)
r.pred.points <- predict(r.logzn.rob, newdata = meuse.grid.pixdf,
  control = control.predict.georob(extended.output = TRUE, full.covmat = TRUE))
str(r.pred.points$pred@data)

## back-transformation of point predictions
r.backtf.pred.points <- lgnpp(r.pred.points)
str(r.backtf.pred.points$pred@data)

spplot(r.backtf.pred.points[["pred"]], zcol = "lgn.pred", main = "Zn content")

## predicting mean Zn content for whole area
r.block <- lgnpp(r.pred.points, is.block = TRUE, all.pred = r.backtf.pred.points[["pred"]])
r.block

## block predictions of log(Zn)
r.pred.block <- predict(r.logzn.rob, newdata = meuse.blocks,
  control = control.predict.georob(extended.output = TRUE,
  pwidth = 75, pheight = 75))
r.backtf.pred.block <- lgnpp(r.pred.block, newdata = meuse.grid)

spplot(r.backtf.pred.block, zcol = "lgn.pred", main = "block means Zn content")
## End(Not run)

```

param.names

Names and Permissible Ranges of Variogram Parameters

Description

Helper functions to query names and permissible ranges of variogram parameters.

Usage

```

param.names(model)

param.bounds(model, d)

```

Arguments

model	character keyword denoting a valid variogram, see georob and georobIntro .
d	number of dimensions of the survey domain.

Value

Either a character vector with the names of the additional variogram parameters such as the smoothness parameter of the Whittle-Matérn model (`param.names`) or a named list with the lower and upper bounds of permissible parameter ranges.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms;
[georob](#) for (robust) fitting of spatial linear models.

Examples

```
param.names("RMgengneiting")
param.bounds("RMgengneiting", d = 2)
```

plot.georob

Plot Methods for Class georob

Description

The plot and lines methods for class georob plot the variogram model, estimated by (robust) restricted maximum likelihood. plot.georob computes and plots in addition the sample variogram of the (robust) regression residuals and can be used to generate residual diagnostics plots (Tukey-Anscombe plot, normal QQ plots of residuals and random effects).

Usage

```
## S3 method for class 'georob'
plot(x, what = c("variogram", "covariance", "correlation",
  "ta", "sl", "qq.res", "qq.ranef" ), add = FALSE, lag.dist.def,
  xy.angle.def = c(0, 180), xz.angle.def = c(0, 180), max.lag = Inf,
  estimator = c("mad", "qn", "ch", "matheron"), mean.angle = TRUE,
  level = what != "ta", smooth = what == "ta" || what == "sl",
  id.n = 3, labels.id = names(residuals(x)), cex.id = 0.75,
  label.pos = c(4,2), col, pch, xlab, ylab, main, lty = "solid", ...)

## S3 method for class 'georob'
lines(x, what = c("variogram", "covariance", "correlation"),
  from = 1.e-6, to, n = 501, xy.angle = 90, xz.angle = 90,
  col = 1:length(xy.angle), pch = 1:length(xz.angle), lty = "solid", ...)
```

Arguments

x an object of class georob, see [georobObject](#).

what character keyword for the quantity that should be displayed. Possible values are:

- "variogram": the variogram
- "covariance": the covariance function
- "correlation": the correlation function

	<ul style="list-style-type: none"> • "scale-location": square root of absolute regression residuals plotted against fitted values (Scale-Location plot) • "ta": regression residuals plotted against fitted values (Tukey-Anscombe plot) • "qq.res": normal Q-Q plot of standardized errors $\hat{\varepsilon}$ • "qq.ranef": normal Q-Q plot of standardized random effects \hat{B}
add	logical controlling whether a new plot should be generated (FALSE, default) or whether the information should be added to the current plot (TRUE).
lag.dist.def	an optional numeric scalar defining a constant bin width for grouping the lag distances or an optional numeric vector with the upper bounds of a set of contiguous bins for computing the sample variogram of the regression residuals, see sample.variogram . If missing then the sample variogram is not computed.
xy.angle.def	an numeric vector defining angular classes in the horizontal plane for computing directional variograms. <code>xy.angle.def</code> must contain an ascending sequence of azimuth angles in degrees from north (positive clockwise to south), see sample.variogram . Omnidirectional variograms are computed with the default <code>c(0, 180)</code> .
xz.angle.def	an numeric vector defining angular classes in the x - z -plane for computing directional variograms. <code>xz.angle.def</code> must contain an ascending sequence of angles in degrees from zenith (positive clockwise to nadir), see sample.variogram . Omnidirectional variograms are computed with the default <code>c(0, 180)</code> .
max.lag	positive numeric defining the largest lag distance for which semi-variances should be computed (default no restriction).
estimator	character keyword defining the estimator for computing the sample variogram. Possible values are: <ul style="list-style-type: none"> • "qn": Genton's robust Q_n-estimator (default, Genton, 1998), • "mad": Dowd's robust MAD-estimator (Dowd, 1984), • "matheron": non-robust method-of-moments estimator, • "ch": robust Cressie-Hawkins estimator (Cressie and Hawkins, 1980).
mean.angle	logical controlling whether the mean lag vector (per combination of lag distance and angular class) is computed from the mean angles of all the lag vectors falling into a given class (TRUE, default) or from the mid-angles of the respective angular classes (FALSE).
level	an integer giving the level for extracting the residuals from object for what = "ta" or what = "qq.res". <code>level = 0</code> (default for what == "ta") extracts the regression residuals $\hat{B}(s) + \hat{\varepsilon}(s)$ and <code>level = 1</code> (default for what == "qq.res") only the estimated errors $\hat{\varepsilon}(s)$.
smooth	logical controlling whether a loess.smooth is added to the Tukey-Anscombe plot (default TRUE).
id.n	number of points to be labelled in each plot, starting with the most extreme (see plot.lmrob).
labels.id	vector of labels, from which the labels for extreme points will be chosen (see plot.lmrob). NULL uses observation numbers.

<code>cex.id</code>	magnification of point labels (see plot.lmrob).
<code>label.pos</code>	positioning of labels, for the left half and right half of the graph respectively (see plot.lmrob).
<code>from</code>	numeric, minimal lag distance for plotting variogram models.
<code>to</code>	numeric, maximum lag distance for plotting variogram models (default: largest lag distance of current plot).
<code>n</code>	positive integer specifying the number of equally spaced lag distances for which semi-variances are evaluated in plotting variogram models (default 501).
<code>xy.angle</code>	numeric (vector) with azimuth angles (in degrees, clockwise positive from north) in x - y -plane for which semi-variances should be plotted.
<code>xz.angle</code>	numeric (vector) with angles in x - z -plane (in degrees, clockwise positive from zenith to south) for which semi-variances should be plotted.
<code>col</code>	optional color of points and curves to distinguish items relating to different azimuth angles in x - y -plane.
<code>pch</code>	optional symbol for points and curves to distinguish items relating to different azimuth angles in x - z -plane.
<code>lty</code>	line type for plotting variogram models.
<code>xlab, ylab, main</code>	test annotation, see plot .
<code>...</code>	additional arguments passed to plot.sample.variogram , loess.smooth and graphical methods.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms;
[georob](#) for (robust) fitting of spatial linear models;
[georobObject](#) for a description of the class `georob`;
[profilelogLik](#) for computing profiles of Gaussian likelihoods;
[control.georob](#) for controlling the behaviour of `georob`;
[georobModelBuilding](#) for stepwise building models of class `georob`;
[cv.georob](#) for assessing the goodness of a fit by `georob`;
[georobMethods](#) for further methods for the class `georob`;
[predict.georob](#) for computing robust Kriging predictions;
[lgnpp](#) for unbiased back-transformation of Kriging prediction of log-transformed data;
[georobSimulation](#) for simulating realizations of a Gaussian process from model fitted by `georob`;
and finally
[sample.variogram](#) and [fit.variogram.model](#) for robust estimation and modelling of sample variograms.

Examples

```
## Not run:
#####
## meuse data ##
#####
data(meuse)

## Gaussian REML fit
r.logzn.reml <- georob(log(zinc) ~ sqrt(dist), data = meuse, locations = ~ x + y,
  variogram.model = "RMexp",
  param = c(variance = 0.15, nugget = 0.05, scale = 200),
  tuning.psi = 1000)
summary(r.logzn.reml, correlation = TRUE)

## robust REML fit
r.logzn.rob <- update(r.logzn.reml, tuning.psi = 1)

summary(r.logzn.rob, correlation = TRUE)

plot(r.logzn.reml, lag.dist.def = seq(0, 2000, by = 100))
lines(r.logzn.rob, col = "red")
## End(Not run)
```

pmm

Parallelized Matrix Multiplication

Description

This page documents the function `pmm` for parallelized matrix multiplication and the function `control.pcmp`, which controls the behaviour of `pmm` and other functions that execute tasks in parallel.

Usage

```
pmm(A, B, control = control.pcmp())

control.pcmp(pmm.ncores = 1, gcr.ncores = 1, max.ncores = detectCores(),
  f = 1, sfstop = FALSE, allow.recursive = TRUE,
  fork = !identical(.Platform[["OS.type"]], "windows"), ...)
```

Arguments

<code>A, B</code>	matrices to be multiplied.
<code>control</code>	a list of with the arguments <code>pmm.ncores</code> , <code>gcr.ncores</code> , <code>max.ncores</code> , <code>f</code> , <code>sfstop</code> and <code>allow.recursive</code> or a function such as <code>control.pcmp</code> that generates such a list.
<code>pmm.ncores</code>	number (integer, default 1) of cores used for parallelized matrix multiplication.

<code>gcr.ncores</code>	number (integer, default 1) of cores used for parallelized computation of semi-variance matrix.
<code>max.ncores</code>	maximum number of cores (integer, default all cores of a machine) used for parallelized computations.
<code>f</code>	number (integer, default 2) of tasks assigned to each core in parallelized operations.
<code>sfstop</code>	logical controlling whether the SNOW socket cluster is stopped after each parallelized matrix multiplication on windows OS (default FALSE).
<code>allow.recursive</code>	logical controlling whether nested parallelized computation should be allowed (default TRUE).
<code>fork</code>	logical controlling whether forking should be used for parallel computations (default TRUE on unix and FALSE on windows operating systems). Note that setting <code>fork == TRUE</code> on windows suppresses parallel computations.
<code>...</code>	further arguments, currently not used.

Details

Parallelized matrix multiplication shortens computing time for large data sets ($n > 1000$). However, spawning child processes requires itself resources and increasing the number of cores for parallel matrix multiplication does not always result in reduced computing time. A sensible default for the number of cores is likely `pmm.ncores=2`.

Note, however, that very substantial reductions in computing time results when one uses the **OpenBLAS** library instead of the reference BLAS library that ships with R, see <https://www.openblas.net/> and R FAQ for your OS. With OpenBLAS no gains are obtained by using more than one core for matrix multiplication, and one should therefore use the default argument `pmm.ncores = 1` for `control.pcmp()`.

`max.ncores` controls how many child processes are spawned in total. This can be used to prevent that child processes spawn themselves children which may result in a considerable number of child processes.

Value

`pmm`: the matrix product `A %%% B`,

`control.pcmp`: a list with components `pmm.ncores`, `gcr.ncores`, `max.ncores`, `f`, `sfstop`, `allow.recursive`.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms;
[georob](#) for (robust) fitting of spatial linear models.

Examples

```
## Not run:
A <- as.matrix(dist(rnorm(2000)))
B <- as.matrix(dist(rnorm(2000)))
system.time(C <- pmm(A, B, control = control.pcmp(pmm.ncores = 1)))
## using 4 cores
## system.time(C <- pmm(A, B, control = control.pcmp(pmm.ncores = 4)))

## End(Not run)
```

predict.georob

Predict Method for Robustly Fitted Spatial Linear Models

Description

Robust and customary external drift Kriging prediction based on a spatial linear models fitted by georob. The predict method for the class georob computes fitted values, point and block Kriging predictions as well as model terms for display by [termplot](#).

Usage

```
## S3 method for class 'georob'
predict(object, newdata, type = c("signal", "response", "trend", "terms"),
        terms = NULL, se.fit = TRUE, signif = 0.95, locations,
        variogram.model = NULL, param = NULL, aniso = NULL, variogram.object = NULL,
        control = control.predict.georob(), verbose = 0, ...)

control.predict.georob(full.covmat = FALSE, extended.output = FALSE,
                       mmax = 10000, ncores = pcmp[["max.ncores"]], pwidth = NULL, pheight = NULL,
                       napp = 1, pcmp = control.pcmp())
```

Arguments

object	an object of class "georob" (mandatory argument), see georobObject .
newdata	an optional data frame, SpatialPointsDataFrame , SpatialPixelsDataFrame , SpatialGridDataFrame , SpatialPolygonsDataFrame or an (optional) object of class SpatialPoints , SpatialPixels or SpatialGrid , in which to look for variables with which to compute fitted values or Kriging predictions, see Details . If newdata is a SpatialPolygonsDataFrame then block Kriging predictions are computed, otherwise point Kriging predictions.
type	character keyword defining what target quantity should be predicted (computed). Possible values are <ul style="list-style-type: none"> "signal": the "signal" $Z(\mathbf{s}) = \mathbf{x}(\mathbf{s})^T \boldsymbol{\beta} + B(\mathbf{s})$ of the process (default), "response": the observations $Y(\mathbf{s}) = Z(\mathbf{s}) + \varepsilon(\mathbf{s})$,

- "trend": the external drift $x(s)^T \beta$,
- "terms": the model terms.

terms	If type = "terms", which terms (default is all terms).
se.fit	logical, only used if type is equal to "terms", see predict.lm .
signif	positive numeric equal to the tolerance or confidence level for computing respective intervals. If NULL no intervals are computed.
locations	an optional one-sided formula specifying what variables of newdata are the coordinates of the prediction points (default: <code>object[["locations.objects"]][["locations"]]</code>).
variogram.model	an optional character keyword defining the variogram model to be used for Kriging, see georob and <i>Details</i> .
param	an optional named numeric vector with values of the variogram parameters used for Kriging, see georob and <i>Details</i> .
aniso	an optional named numeric vector with values of anisotropy parameters of a variogram used for Kriging, see georob and <i>Details</i> .
variogram.object	an optional list that defines a possibly nested variogram model used for Kriging, see georob and <i>Details</i> .
control	a list with the components <code>full.covmat</code> , <code>extended.output</code> , <code>mmax</code> , <code>ncores</code> , <code>pwidth</code> , <code>pheight</code> , <code>napp</code> and <code>pcmp</code> or a function such as <code>control.predict.georob</code> that generates such a list.
full.covmat	logical controlling whether the full covariance matrix of the prediction errors is returned (TRUE) or only the vector with its diagonal elements (FALSE, default), see <i>Value</i> for an explanation of the effect of <code>full.covmat</code> .
extended.output	logical controlling whether the covariance matrices of the Kriging predictions and of the data should be computed, see <i>Details</i> (default FALSE).
mmax	integer equal to the maximum number (default 10000) of prediction items, computed in a sub-task, see <i>Details</i> .
ncores	positive integer controlling how many cores are used for parallelized computations, see <i>Details</i> .
pwidth, pheight, napp	numeric scalars, used to tune numeric integration of semi-variances for block Kriging, see preCKrige .
pcmp	a list of arguments passed to <code>pmm</code> or a function such as <code>control.pcmp</code> that generates such a list (see <code>control.pcmp</code> for allowed arguments).
verbose	positive integer controlling logging of diagnostic messages to the console. <code>verbose = 0</code> (default) largely suppresses such messages.
...	arguments passed to <code>control.predict.georob</code> .

Details

If `newdata` is an object of class `SpatialPoints`, `SpatialPixels` or `SpatialGrid` then the drift model may only use the coordinates as covariates (universal Kriging). Furthermore the names used for the coordinates in `newdata` must be the same as in `data` when creating object (argument `locations` of `predict.georob` should not be used). Note that the result returned by `predict.georob` is then an object of class `SpatialPointsDataFrame`, `SpatialPixelsDataFrame` or `SpatialGridDataFrame`.

The `predict` method for class `georob` uses the packages **parallel**, **snow** and **snowfall** for parallelized computation of Kriging predictions. If there are m items to predict, the task is split into $\text{ceiling}(m/m_{\text{max}})$ sub-tasks that are then distributed to `ncores` CPUs. Evidently, `ncores = 1` suppresses parallel execution. By default, the function uses all available CPUs as returned by `detectCores`.

Note that if `full.covmat` is `TRUE` `mmax` must exceed m (and parallel execution is not possible).

The argument `extended.output = TRUE` is used to compute all quantities required for (approximately) unbiased back-transformation of Kriging predictions of log-transformed data to the original scale of the measurements by `lgnp`. In more detail, the following items are computed:

- `trend`: the fitted values, $\mathbf{x}(\mathbf{s})^T \hat{\boldsymbol{\beta}}$,
- `var.pred`: the variances of the Kriging predictions, $\text{Var}_{\hat{\theta}}[\hat{Y}(\mathbf{s})]$ or $\text{Var}_{\hat{\theta}}[\hat{Z}(\mathbf{s})]$,
- `cov.pred.target`: the covariances between the predictions and the prediction targets, $\text{Cov}_{\hat{\theta}}[\hat{Y}(\mathbf{s}), Y(\mathbf{s})]$ or $\text{Cov}_{\hat{\theta}}[\hat{Z}(\mathbf{s}), Z(\mathbf{s})]$,
- `var.target`: the variances of the prediction targets $\text{Var}_{\theta}[Y(\mathbf{s})]$ or $\text{Var}_{\theta}[Z(\mathbf{s})]$.

Note that the component `var.pred` is also present if `type` is equal to "trend", irrespective of the choice for `extended.output`. This component contains then the variances of the fitted values.

Value

If `type` is equal to "terms" then a vector, a matrix, or a list with prediction results along with bounds and standard errors, see `predict.lm`. Otherwise, the structure and contents of the output generated by `predict.georob` are determined by the class of `newdata` and the logical flags `full.covmat` and `extended.output`:

If `full.covmat` is `FALSE` then the result is an object of a "similar" class as `newdata` (data frame, `SpatialPointsDataFrame`, `SpatialPixelsDataFrame`, `SpatialGridDataFrame`, `SpatialPolygonsDataFrame`).

The data frame or the data slot of the `Spatial...DataFrame` objects have the following components:

- the coordinates of the prediction points (only present if `newdata` is a data frame).
- `pred`: the Kriging predictions (or fitted values).
- `se`: the root mean squared prediction errors (Kriging standard errors).
- `lower`, `upper`: the limits of tolerance/confidence intervals,
- `trend`, `var.pred`, `cov.pred.target`, `var.target`: only present if `extended.output` is `TRUE`, see *Details*.

If `full.covmat` is `TRUE` then `predict.georob` returns a list with the following components:

- `pred`: a data frame or a `Spatial...DataFrame` object as described above for `full.covmat = FALSE`.
- `mse.pred`: the full covariance matrix of the prediction errors, $Y(\mathbf{s}) - \hat{Y}(\mathbf{s})$ or $Z(\mathbf{s}) - \hat{Z}(\mathbf{s})$ see *Details*.
- `var.pred`: the full covariance matrix of the Kriging predictions, see *Details*.
- `cov.pred.target`: the full covariance matrix of the predictions and the prediction targets, see *Details*.
- `var.target`: the full covariance matrix of the prediction targets, see *Details*.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

References

Nussbaum, M., Papritz, A., Baltensweiler, A. and Walther, L. (2014) Estimating soil organic carbon stocks of Swiss forest soils by robust external-drift kriging. *Geoscientific Model Development*, 7, 1197–1210. doi: [10.5194/gmd711972014](https://doi.org/10.5194/gmd711972014), <https://gmd.copernicus.org/articles/7/1197/2014/>.

Künsch, H. R., Papritz, A., Schwierz, C. and Stahel, W. A. (2011) Robust estimation of the external drift and the variogram of spatial data. Proceedings of the ISI 58th World Statistics Congress of the International Statistical Institute. doi: [10.3929/ethz009900710](https://doi.org/10.3929/ethz009900710)

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms;

[georob](#) for (robust) fitting of spatial linear models;

[georobObject](#) for a description of the class `georob`;

[profilelogLik](#) for computing profiles of Gaussian likelihoods;

[plot.georob](#) for display of RE(ML) variogram estimates;

[control.georob](#) for controlling the behaviour of `georob`;

[georobModelBuilding](#) for stepwise building models of class `georob`;

[cv.georob](#) for assessing the goodness of a fit by `georob`;

[georobMethods](#) for further methods for the class `georob`;

[lgnpp](#) for unbiased back-transformation of Kriging prediction of log-transformed data;

[georobSimulation](#) for simulating realizations of a Gaussian process from model fitted by `georob`; and finally

[sample.variogram](#) and [fit.variogram.model](#) for robust estimation and modelling of sample variograms.

Examples

```

## Not run:
data(meuse)

data(meuse.grid)
coordinates(meuse.grid) <- ~x+y
meuse.grid.pixdf <- meuse.grid
gridded(meuse.grid.pixdf) <- TRUE

library(constrainedKriging)
data(meuse.blocks)

r.logzn.rob <- georob(log(zinc) ~ sqrt(dist), data = meuse, locations = ~ x + y,
  variogram.model = "RMexp", param = c(variance = 0.15, nugget = 0.05, scale = 200),
  tuning.psi = 1., control = control.georob(cov.bhat = TRUE, full.cov.bhat = TRUE))

## point predictions of log(Zn)
r.pred.points <- predict(r.logzn.rob, newdata = meuse.grid.pixdf,
  control = control.predict.georob(extended.output = TRUE, full.covmat = TRUE))
str(r.pred.points$pred@data)

## back-transformation of point predictions
r.backtf.pred.points <- lgnpp(r.pred.points)
str(r.backtf.pred.points$pred@data)

splot(r.backtf.pred.points[["pred"]], zcol = "lgn.pred", main = "Zn content")

## predicting mean Zn content for whole area
r.block <- lgnpp(r.backtf.pred.points, is.block = TRUE, all.pred = r.backtf.pred.points[["pred"]])
r.block

## block predictions of log(Zn)
r.pred.block <- predict(r.logzn.rob, newdata = meuse.blocks,
  control = control.predict.georob(extended.output = TRUE,
  pwidth = 75, pheight = 75))
r.backtf.pred.block <- lgnpp(r.pred.block, newdata = meuse.grid)

splot(r.backtf.pred.block, zcol = "lgn.pred", main = "block means Zn content")
## End(Not run)

```

Description

The function `profilelogLik` computes for an array of fixed variogram parameters the profile log-likelihood by maximizing the (restricted) log-likelihood with respect to the remaining variogram parameters, the fixed and random effects.

Usage

```
profilelogLik(object, values, use.fitted = TRUE, verbose = 0,
              ncores = min(detectCores(), NROW(values)))
```

Arguments

object	an object of class "georob" (mandatory argument), see georobObject .
values	a data.frame or a matrix with the values of the variogram and anisotropy parameters that should be kept fixed (mandatory argument, see georob and georobIntro for information about the parametrization of variogram models). The names of the columns of values must match the names of variogram and anisotropy parameters.
use.fitted	logical scalar controlling whether the fitted variogram parameters of object should be used as initial values (default TRUE) when maximizing the profile log-likelihood or the initial values used to generate object.
verbose	positive integer controlling logging of diagnostic messages to the console during model fitting, see georob .
ncores	positive integer controlling how many cores are used for parallelized computations, see <i>Details</i> .

Details

For robust REML fits `profilelogLik` returns (possibly with a warning) the log-likelihood of the Gaussian (RE)ML fit of the equivalent Gaussian spatial linear model with heteroscedastic nugget.

Note that *the data frame passed as data argument to georob must exist in the user workspace when calling profilelogLik*.

`profilelogLik` uses the packages **parallel**, **snow** and **snowfall** for parallelized computation of the profile likelihood. By default, the function uses `NROW(values)` CPUs but not more than are physically available (as returned by `detectCores`).

`profilelogLik` uses the function `update` to re-estimated the model with partly fixed variogram parameters. Therefore, any argument accepted by `georob` except data can be changed when re-fitting the model. Some of them (e.g. `verbose`) are explicit arguments of `profilelogLik`, but also the remaining ones can be passed by `...` to the function.

Value

A data.frame with the columns of values, a column `loglik` (contains the maximized [restricted] log-likelihood), columns with the estimated variogram and fixed effect parameters, columns with the gradient of the (restricted) log-likelihood (or the roots of the estimating equations) and a column `converged`, indicating whether convergence has occurred `converged == 1` when fitting the respective model.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms;
[georob](#) for (robust) fitting of spatial linear models;
[georobObject](#) for a description of the class `georob`;
[plot.georob](#) for display of RE(ML) variogram estimates;
[control.georob](#) for controlling the behaviour of `georob`;
[georobModelBuilding](#) for stepwise building models of class `georob`;
[cv.georob](#) for assessing the goodness of a fit by `georob`;
[georobMethods](#) for further methods for the class `georob`;
[predict.georob](#) for computing robust Kriging predictions;
[lgnpp](#) for unbiased back-transformation of Kriging prediction of log-transformed data;
[georobSimulation](#) for simulating realizations of a Gaussian process from model fitted by `georob`;
 and finally
[sample.variogram](#) and [fit.variogram.model](#) for robust estimation and modelling of sample variograms.

Examples

```

## Not run:

data(meuse)

r.logzn.ml <- georob(log(zinc)~sqrt(dist), data=meuse, locations=~x+y,
  variogram.model="RMexp", param=c(variance=0.15, nugget=0.05, scale=200),
  tuning.psi=1000, control=control.georob(ml.method="ML"))

r.prflik <- profilelogLik(r.logzn.ml, values=expand.grid(scale=seq(75, 600, by=25)), ncores = 1)
## using multiple cores
## r.prflik <- profilelogLik(r.logzn.ml, values=expand.grid(scale=seq(75, 600, by=25)))
plot(loglik~scale, r.prflik, type="l")
abline(v=r.logzn.ml$variogram.object[[1]]$param["scale"], lty="dotted")
abline(h=r.logzn.ml$loglik-0.5*qchisq(0.95, 1), lty="dotted")

## End(Not run)

```

sample.variogram

Computing (Robust) Sample Variograms of Spatial Data

Description

The function `sample.variogram` computes the sample (empirical) variogram of a spatial variable by the method-of-moment and three robust estimators. Both omnidirectional and direction-dependent variograms can be computed, the latter for observation locations in a three-dimensional domain. There are summary and plot methods for summarizing and displaying sample variograms.

Usage

```

sample.variogram(object, ...)

## Default S3 method:
sample.variogram(object, locations, lag.dist.def,
  xy.angle.def = c(0, 180), xz.angle.def = c(0, 180), max.lag = Inf,
  estimator = c("qn", "mad", "matheron", "ch"), mean.angle = TRUE, ...)

## S3 method for class 'formula'
sample.variogram(object, data, subset, na.action,
  locations, lag.dist.def, xy.angle.def = c(0, 180),
  xz.angle.def = c(0, 180), max.lag = Inf,
  estimator = c("qn", "mad", "matheron", "ch"), mean.angle = TRUE, ...)

## S3 method for class 'georob'
sample.variogram(object, lag.dist.def,
  xy.angle.def = c(0, 180), xz.angle.def = c(0, 180), max.lag = Inf,
  estimator = c("qn", "mad", "matheron", "ch"), mean.angle = TRUE, ...)

## S3 method for class 'sample.variogram'
summary(object, ...)

## S3 method for class 'sample.variogram'
plot(x, type = "p", add = FALSE,
  xlim = c(0, max(x[["lag.dist"]])),
  ylim = c(0, 1.1 * max(x[["gamma"]])), col, pch, lty, cex = 0.8,
  xlab = "lag distance", ylab = "semivariance",
  annotate.npairs = FALSE, npairs.pos = 3, npairs.cex = 0.7,
  legend = nlevels(x[["xy.angle"]]) > 1 || nlevels(x[["xz.angle"]]) > 1,
  legend.pos = "topleft", ...)

```

Arguments

object	a numeric vector with the values of the response for which the sample variogram should be computed (<code>sample.variogram.default</code>), a formula, specifying in its left part the response variable (right part of formula is ignored, <code>sample.variogram.formula</code>), an object of class <code>georob</code> (<code>sample.variogram.georob</code>) or an object of class <code>sample.variogram.summary.sample.variogram</code>).
locations	a numeric matrix with the coordinates of the locations where the response was observed (<code>sample.variogram.default</code>) or a one-sided formula specifying the coordinates (<code>sample.variogram.formula</code>). The matrix may have an arbitrary number of columns for an omnidirectional variogram, but at most 3 columns if a directional variogram is computed.
data	an optional data frame, list or environment (or another object coercible by as.data.frame to a data frame) containing the response variable and the co-

	ordinates where the data was recorded. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>sample.variogram</code> is called.
<code>subset</code>	an optional vector specifying a subset of observations to be used for estimating the variogram.
<code>na.action</code>	a function which indicates what should happen when the data contain NAs. The default is set by the <code>na.action</code> argument of <code>options</code> , and is <code>na.fail</code> if that is unset. The “factory-fresh” default is <code>na.omit</code> . Another possible value is <code>NULL</code> , no action. Value <code>na.exclude</code> can be useful.
<code>lag.dist.def</code>	a numeric scalar defining a constant bin width for grouping the lag distances or a numeric vector with the bounds of a set of contiguous bins (upper bounds of bins except for the first element of <code>lag.dist.def</code> which is the lower bound of the first bin). This argument is mandatory.
<code>xy.angle.def</code>	an numeric vector defining angular classes in the horizontal plane for computing directional variograms. <code>xy.angle.def</code> must contain an ascending sequence of azimuth angles in degrees from north (positive clockwise to south), see <i>Details</i> . Omnidirectional variograms are computed with the default <code>c(0, 180)</code> .
<code>xz.angle.def</code>	an numeric vector defining angular classes in the x - z -plane for computing directional variograms. <code>xz.angle.def</code> must contain an ascending sequence of angles in degrees from zenith (positive clockwise to nadir), see <i>Details</i> . Omnidirectional variograms are computed with the default <code>c(0, 180)</code> .
<code>max.lag</code>	positive numeric defining the largest lag distance for which semi variances should be computed (default no restriction).
<code>estimator</code>	character keyword defining the estimator for computing the sample variogram. Possible values are: <ul style="list-style-type: none"> • “qn”: Genton’s robust Q_n-estimator (default, Genton, 1998), • “mad”: Dowd’s robust MAD-estimator (Dowd, 1984), • “matheron”: non-robust method-of-moments estimator, • “ch”: robust Cressie-Hawkins estimator (Cressie and Hawkins, 1980).
<code>mean.angle</code>	logical controlling whether the mean lag vector (per combination of lag distance and angular class) is computed from the mean angles of all the lag vectors falling into a given class (<code>TRUE</code> , default) or from the mid-angles of the respective angular classes (<code>FALSE</code>).
<code>x</code>	an object of class <code>sample.variogram</code> .
<code>type, xlim, ylim, xlab, ylab</code>	see respective arguments of <code>plot.default</code> .
<code>add</code>	logical controlling whether a new plot should be generated (<code>FALSE</code> , default) or whether the information should be added to the current plot (<code>TRUE</code>).
<code>col</code>	the color of plotting symbols for distinguishing semi variances for angular classes in the x - y -plane.
<code>pch</code>	the type of plotting symbols for distinguishing semi variances for angular classes in the x - z -plane.
<code>lty</code>	the line type.

cex	character expansion factor for plotting symbols.
annotate.npairs	logical controlling whether the plotting symbols should be annotated by the number of data pairs per lag class.
npairs.pos	integer defining the position where text annotation about number of pairs should be plotted, see text .
npairs.cex	numeric defining the character expansion for text annotation about number of pairs.
legend	logical controlling whether a legend should be plotted.
legend.pos	a character keyword defining where to place the legend, see legend for possible values.
...	additional arguments passed to plot.formula .

Details

The angular classes in the x - y - and x - z -plane are defined by vectors of ascending angles on the half circle. The i th angular class is defined by the vector elements, say l and u , with indices i and $i + 1$. A lag vector belongs to the i th angular class if its azimuth (or angle from zenith), say φ , satisfies $l < \varphi \leq u$. If the first and the last element of `xy.angle.def` or `xz.angle.def` are equal to 0 and 180 degrees, respectively, then the first and the last angular class are “joined”, i.e., if there are K angles, there will be only $K - 2$ angular classes and the first class is defined by the interval (`xy.angle.def[K-1]-180`, `xy.angle.def[2]`] and the last class by (`xy.angle.def[K-2]`, `xy.angle.def[K-1]`].

Value

An object of class `sample.variogram`, which is a data frame with the following components:

lag.dist	the mean lag distance of the lag class,
xy.angle	the angular class in the x - y -plane,
xz.angle	the angular class in the x - z -plane,
gamma	the estimated semi-variance of the lag class,
npairs	the number of data pairs in the lag class,
lag.x	the x -component of the mean lag vector of the lag class,
lag.y	the y -component of the mean lag vector of the lag class,
lag.z	the z -component of the mean lag vector of the lag class.

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

References

- Cressie, N. and Hawkins, D. M. (1980) Robust Estimation of the Variogram: I. *Mathematical Geology*, **12**, 115–125.
- Dowd, P. A. (1984) The variogram and Kriging: Robust and resistant estimators. In *Geostatistics*

for *Natural Resources Characterization*, Verly, G., David, M., Journel, A. and Marechal, A. (Eds.) Dordrecht: D. Reidel Publishing Company, Part I, 1, 91–106.

Genton, M. (1998) Highly Robust Variogram Estimation. *Mathematical Geology*, **30**, 213–220.

See Also

[georobIntro](#) for a description of the model and a brief summary of the algorithms;
[georob](#) for (robust) fitting of spatial linear models;
[georobObject](#) for a description of the class `georob`;
[profilelogLik](#) for computing profiles of Gaussian likelihoods;
[plot.georob](#) for display of RE(ML) variogram estimates;
[control.georob](#) for controlling the behaviour of `georob`;
[georobModelBuilding](#) for stepwise building models of class `georob`;
[cv.georob](#) for assessing the goodness of a fit by `georob`;
[georobMethods](#) for further methods for the class `georob`;
[predict.georob](#) for computing robust Kriging predictions;
[lgnpp](#) for unbiased back-transformation of Kriging prediction of log-transformed data;
[georobSimulation](#) for simulating realizations of a Gaussian process from model fitted by `georob`.

Examples

```
data(wolfcamp)

## fitting an isotropic IRF(0) model
r.sv.iso <- sample.variogram(pressure~1, data = wolfcamp,
  locations = ~x + y, lag.dist.def = seq(0, 200, by = 15))

## Not run:
plot(r.sv.iso, type = "l")
## End(Not run)

## fitting an anisotropic IRF(0) model
r.sv.aniso <- sample.variogram(pressure~1, data = wolfcamp,
  locations = ~x + y, lag.dist.def = seq(0, 200, by = 15),
  xy.angle.def = c(0., 22.5, 67.5, 112.5, 157.5, 180.))
## Not run:
plot(r.sv.aniso, type = "l", add = TRUE, col = 2:5)
## End(Not run)
```

 validate.predictions *Summary Statistics of (Cross-)Validation Prediction Errors*

Description

Functions to compute and plot summary statistics of prediction errors to (cross-)validate fitted spatial linear models by the criteria proposed by Gneiting et al. (2007) for assessing probabilistic forecasts.

Usage

```
validate.predictions(data, pred, se.pred,
  statistic = c("crps", "pit", "mc", "bs", "st"), ncutoff = NULL)

## S3 method for class 'cv.georob'
plot(x,
  type = c("sc", "lgn.sc", "ta", "qq", "hist.pit", "ecdf.pit", "mc", "bs"),
  smooth = TRUE, span = 2/3, ncutoff = NULL, add = FALSE,
  col, pch, lty, main, xlab, ylab, ...)

## S3 method for class 'cv.georob'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'cv.georob'
summary(object, se = FALSE, ...)
```

Arguments

data	a numeric vector with observations about a response (mandatory argument).
pred	a numeric vector with predictions for the response (mandatory argument).
se.pred	a numeric vector with prediction standard errors (mandatory argument).
statistic	character keyword defining what statistic of the prediction errors should be computed. Possible values are (see <i>Details</i>): <ul style="list-style-type: none"> • "crps": continuous ranked probability score (default), • "pit": probability integral transform, • "mc": average predictive distribution (marginal calibration), • "bs": Brier score, • "st": mean and dispersion statistics of (standardized) prediction errors.
ncutoff	positive integer (N) giving the number of quantiles, for which CDFs are evaluated (type = "mc"), or the number of thresholds for which the Brier score is computed (type = "bs"), see <i>Details</i> (default: $\min(500, \text{length}(\text{data}))$).
x, object	objects of class cv.georob.

digits	positive integer indicating the number of decimal digits to print.
type	character keyword defining what type of plot is created by the <code>plot.cv.georob</code> . Possible values are: <ul style="list-style-type: none"> • "sc": a scatter-plot of the (possibly log-transformed) response vs. the respective predictions (default). • "lgn.sc": a scatter-plot of the untransformed response against back-transformed predictions of the log-transformed response. • "ta": Tukey-Anscombe plot (plot of standardized prediction errors vs. predictions). • "qq": normal QQ plot of standardized prediction errors. • "hist.pit": histogram of probability integral transform, see <i>Details</i>. • "ecdf.pit": empirical CDF of probability integral transform, see <i>Details</i>. • "mc": a marginal calibration plot, see <i>Details</i>, • "bs": a plot of Brier score vs. threshold, see <i>Details</i>.
smooth	control whether scatter plots of data vs. predictions should be smoothed by loess.smooth .
span	smoothness parameter for loess (see loess.smooth).
add	logical controlling whether the current high-level plot is added to an existing graphics without cleaning the frame before (default: FALSE).
main, xlab, ylab	title and axes labels of plot.
col, pch, lty	color, symbol and line type.
se	logical controlling if the standard errors of the averaged continuous ranked probability score and of the mean and dispersion statistics of the prediction errors (see <i>Details</i>) are computed from the respective values computed for the K cross-validation subsets (default: FALSE).
...	additional arguments passed to the methods.

Details

`validate.predictions` computes the items required to evaluate (and plot) the diagnostic criteria proposed by Gneiting et al. (2007) for assessing the *calibration* and the *sharpness* of probabilistic predictions of (cross-)validation data. To this aim, `validate.predictions` uses the assumption that the prediction errors $Y(s) - \hat{Y}(s)$ follow normal distributions with zero mean and standard deviations equal to the Kriging standard errors. This assumption is an approximation if the errors ε come from a long-tailed distribution. Furthermore, for the time being, the Kriging variance of the *response* Y is approximated by adding the estimated nugget $\hat{\tau}^2$ to the Kriging variance of the signal Z . This approximation likely underestimates the mean squared prediction error of the response if the errors come from a long-tailed distribution. Hence, for robust Kriging, the standard errors of the (cross-)validation errors are likely too small.

Notwithstanding these difficulties and imperfections, `validate.predictions` computes

- the *probability integral transform* (PIT),

$$\text{PIT}_i = F_i(y_i),$$

where $F_i(y_i)$ denotes the (plug-in) predictive CDF evaluated at y_i , the value of the i th (cross-)validation datum,

- the *average predictive CDF* (plug-in)

$$\bar{F}_n(y) = 1/n \sum_{i=1}^n F_i(y),$$

where n is the number of (cross-)validation observations and the F_i are evaluated at N quantiles equal to the set of distinct y_i (or a subset of size N of them),

- the *Brier Score* (plug-in)

$$\text{BS}(y) = 1/n \sum_{i=1}^n (F_i(y) - I(y_i \leq y))^2,$$

where $I(x)$ is the indicator function for the event x , and the Brier score is again evaluated at the unique values of the (cross-)validation observations (or a subset of size N of them),

- the *averaged continuous ranked probability score*, CRPS, a strictly proper scoring criterion to rank predictions, which is related to the Brier score by

$$\text{CRPS} = \int_{-\infty}^{\infty} \text{BS}(y) dy.$$

Gneiting et al. (2007) proposed the following plots to validate probabilistic predictions:

- A histogram (or a plot of the empirical CDF) of the PIT values. For ideal predictions, with observed coverages of prediction intervals matching nominal coverages, the PIT values have a uniform distribution.
- Plots of $\bar{F}_n(y)$ and of the empirical CDF of the data, say $\hat{G}_n(y)$, and of their difference, $\bar{F}_n(y) - \hat{G}_n(y)$ vs. y . The forecasts are said to be *marginally calibrated* if $\bar{F}_n(y)$ and $\hat{G}_n(y)$ match.
- A plot of $\text{BS}(y)$ vs. y . Probabilistic predictions are said to be *sharp* if the area under this curve, which equals CRPS, is minimized.

The plot method for class `cv.georob` allows to create these plots, along with scatter-plots of observations and predictions, Tukey-Anscombe and normal QQ plots of the standardized prediction errors.

`summary.cv.georob` computes the mean and dispersion statistics of the (standardized) prediction errors (by a call to `validate.prediction` with argument `statistic = "st"`, see *Value*) and the averaged continuous ranked probability score (`crps`). If present in the `cv.georob` object, the error statistics are also computed for the errors of the unbiasedly back-transformed predictions of a log-transformed response. If `se` is TRUE then these statistics are evaluated separately for the K cross-validation subsets and the standard errors of the means of these statistics are returned as well.

The `print` method for class `cv.georob` returns the mean and dispersion statistics of the (standardized) prediction errors.

Value

Depending on the argument `statistic`, the function `validate.predictions` returns

- a numeric vector of PIT values if `statistic` is equal to "pit",
- a named numeric vector with summary statistics of the (standardized) prediction errors if `statistic` is equal to "st". The following statistics are computed:

me	mean prediction error
mede	median prediction error
rmse	root mean squared prediction error
made	median absolute prediction error
qne	Qn dispersion measure of prediction errors (see Qn)
msse	mean squared standardized prediction error
medsse	median squared standardized prediction error

- a data frame if `statistic` is equal to "mc" or "bs" with the components (see *Details*):

z	the sorted unique (cross-)validation observations (or a subset of size <code>ncutoff</code> of them)
ghat	the empirical CDF of the (cross-)validation observations $\hat{G}_n(y)$
fbar	the average predictive distribution $\bar{F}_n(y)$
bs	the Brier score $BS(y)$

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>.

References

Gneiting, T., Balabdaoui, F. and Raftery, A. E. (2007) Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society Series B* **69**, 243–268.

See Also

[georob](#) for (robust) fitting of spatial linear models;
[cv.georob](#) for assessing the goodness of a fit by `georob`.

Examples

```
## Not run:
data(meuse)

r.logzn <- georob(log(zinc) ~ sqrt(dist), data = meuse, locations = ~ x + y,
  variogram.model = "RMexp",
  param = c(variance = 0.15, nugget = 0.05, scale = 200),
  tuning.psi = 1)

r.logzn.cv.1 <- cv(r.logzn, seed = 1, lgn = TRUE, ncores = 1, verbose = 1)
```

```

r.logzn.cv.2 <- cv(r.logzn, formula = .~. + ffreq, seed = 1, lgn = TRUE,
  ncores = 1, verbose = 1)

summary(r.logzn.cv.1, se = TRUE)
summary(r.logzn.cv.2, se = TRUE)

op <- par(mfrow = c(2,2))
plot(r.logzn.cv.1, type = "lgn.sc")
plot(r.logzn.cv.2, type = "lgn.sc", add = TRUE, col = "red")
abline(0, 1, lty= "dotted")
plot(r.logzn.cv.1, type = "ta")
plot(r.logzn.cv.2, type = "ta", add = TRUE, col = "red")
abline(h=0, lty= "dotted")
plot(r.logzn.cv.2, type = "mc", col = "red")
plot(r.logzn.cv.1, type = "bs")
plot(r.logzn.cv.2, type = "bs", add = TRUE, col = "red")
legend("topright", lty = 1, col = c("black", "red"), bty = "n",
  legend = c("log(Zn) ~ sqrt(dist)", "log(Zn) ~ sqrt(dist) + ffreq"))
par(op)
## End(Not run)

```

 wolfcamp

Wolfcamp Aquifer Data

Description

Piezometric head measurements taken at the Wolfcamp Aquifer, Texas, USA. See *Cressie (1993, p. 212–214)* for description of the scientific problem and the data. Original data were converted to SI units: coordinates are given in kilometers and pressure heads in meters.

Usage

```
data(wolfcamp)
```

Format

A data frame with 85 observations on the following 3 variables:

- x a numeric vector with the easting coordinate in kilometers.
- y a numeric vector with the northing coordinate in kilometers.
- pressure a numeric vector with the piezometric head in meters.

Note

The data were imported from the package **geoR**.

Source

Harper, W.V. and Furr, J.M. (1986) Geostatistical analysis of potentiometric data in the Wolfcamp Aquifer of the Palo Duro Basin, Texas. *Technical Report BMI/ONWI-587, Bettelle Memorial Institute, Columbus, OH.*

References

Cressie, N.A.C (1993) *Statistics for Spatial Data*. New York: Wiley.

Papritz, A. and Moyeed, R. (2001) Parameter uncertainty in spatial prediction: checking its importance by cross-validating the Wolfcamp and Rongelap data sets. *GeoENV 2000: Geostatistical for Environmental Applications*. Ed. P. Monestiez, D. Allard, R. Froidevaux. Kluwer.

Examples

```
data(wolfcamp)
summary(wolfcamp)
```

Index

- * **datasets**
 - elevation, 16
 - wolfcamp, 75
- * **methods**
 - georobS3methods, 41
- * **models**
 - compress, 2
 - control.georob, 3
 - cv, 9
 - cv.georob, 10
 - default.aniso, 15
 - fit.variogram.model, 17
 - georob, 23
 - georobModelBuilding, 30
 - georobObject, 34
 - georobPackage, 37
 - georobS3methods, 41
 - georobSimulation, 45
 - lgnpp, 49
 - param.names, 54
 - plot.georob, 55
 - pmm, 58
 - predict.georob, 60
 - profilelogLik, 64
 - sample.variogram, 66
 - validate.predictions, 71
- * **package**
 - georobPackage, 37
- * **print**
 - georobS3methods, 41
- * **robust**
 - compress, 2
 - control.georob, 3
 - default.aniso, 15
 - fit.variogram.model, 17
 - georob, 23
 - georobModelBuilding, 30
 - georobObject, 34
 - georobPackage, 37
 - georobS3methods, 41
 - georobSimulation, 45
 - lgnpp, 49
 - param.names, 54
 - plot.georob, 55
 - pmm, 58
 - predict.georob, 60
 - profilelogLik, 64
 - sample.variogram, 66
 - validate.predictions, 71
- * **spatial**
 - compress, 2
 - control.georob, 3
 - cv, 9
 - cv.georob, 10
 - default.aniso, 15
 - fit.variogram.model, 17
 - georob, 23
 - georobModelBuilding, 30
 - georobObject, 34
 - georobPackage, 37
 - georobS3methods, 41
 - georobSimulation, 45
 - lgnpp, 49
 - param.names, 54
 - plot.georob, 55
 - pmm, 58
 - predict.georob, 60
 - profilelogLik, 64
 - sample.variogram, 66
 - validate.predictions, 71
- add1.georob, 40
- add1.georob (georobModelBuilding), 30
- as.data.frame, 24, 67
- bwd.transf, 20
- bwd.transf (control.georob), 3
- coef.georob (georobS3methods), 41

- compress, [2](#), [35](#), [47](#)
 condsim (georobSimulation), [45](#)
 confint, [43](#)
 control.condsim (georobSimulation), [45](#)
 control.fit.variogram.model
 (fit.variogram.model), [17](#)
 control.georob, [3](#), [14](#), [19](#), [20](#), [22](#), [26–29](#), [33](#),
 [35](#), [36](#), [40](#), [44](#), [48](#), [57](#), [63](#), [66](#), [70](#)
 control.nleqslv (control.georob), [3](#)
 control.nlminb, [20](#)
 control.nlminb (control.georob), [3](#)
 control.optim, [20](#), [27](#)
 control.optim (control.georob), [3](#)
 control.pcmp, [7](#), [47](#), [61](#)
 control.pcmp (pmm), [58](#)
 control.predict.georob, [40](#)
 control.predict.georob
 (predict.georob), [60](#)
 control.rq (control.georob), [3](#)
 cv, [9](#)
 cv.georob, [9](#), [10](#), [10](#), [22](#), [29](#), [33](#), [34](#), [36](#), [40](#),
 [41](#), [44](#), [48](#), [57](#), [63](#), [66](#), [70](#), [74](#)
- default.aniso, [15](#), [18](#), [25](#), [26](#)
 default.fit.aniso, [19](#), [25](#), [26](#)
 default.fit.aniso (default.aniso), [15](#)
 default.fit.param, [18](#), [25](#), [26](#)
 default.fit.param (default.aniso), [15](#)
 detectCores, [13](#), [48](#), [62](#), [65](#)
 deviance.georob, [40](#)
 deviance.georob (georobModelBuilding),
 [30](#)
 df.residual, [43](#)
 dfwd.transf (control.georob), [3](#)
 drop1.georob, [40](#)
 drop1.georob (georobModelBuilding), [30](#)
- elevation, [16](#)
 expand, [35](#)
 expand (compress), [2](#)
 extractAIC, [31](#)
 extractAIC.georob, [40](#)
 extractAIC.georob
 (georobModelBuilding), [30](#)
- fit.variogram.model, [9](#), [14](#), [17](#), [29](#), [33](#), [37](#),
 [40](#), [41](#), [44](#), [48](#), [57](#), [63](#), [66](#)
 fitted, [43](#)
 fixed.effects (georobS3methods), [41](#)
- fixef (georobS3methods), [41](#)
 formula, [24](#), [31](#), [42](#), [43](#)
 fwd.transf, [20](#)
 fwd.transf (control.georob), [3](#)
- georob, [3](#), [5](#), [8–14](#), [16](#), [18](#), [21](#), [22](#), [23](#), [32–34](#),
 [36](#), [40](#), [44–48](#), [53–55](#), [57](#), [59](#), [61](#), [63](#),
 [65](#), [66](#), [70](#), [74](#)
 georobIntro, [9](#), [14](#), [16](#), [18](#), [21](#), [22](#), [24–26](#), [28](#),
 [33](#), [34](#), [36](#), [44](#), [46–48](#), [51](#), [53–55](#), [57](#),
 [59](#), [63](#), [65](#), [66](#), [70](#)
 georobIntro (georobPackage), [37](#)
 georobMethods, [9](#), [14](#), [22](#), [29](#), [33](#), [34](#), [36](#), [41](#),
 [48](#), [57](#), [63](#), [66](#), [70](#)
 georobMethods (georobS3methods), [41](#)
 georobModelBuilding, [9](#), [14](#), [22](#), [29](#), [30](#), [34](#),
 [36](#), [40](#), [44](#), [48](#), [57](#), [63](#), [66](#), [70](#)
 georobObject, [9](#), [11](#), [14](#), [22](#), [28](#), [31–33](#), [34](#),
 [40](#), [42](#), [44](#), [45](#), [48](#), [55](#), [57](#), [60](#), [63](#), [65](#),
 [66](#), [70](#)
 georobPackage, [32](#), [37](#)
 georobS3methods, [41](#)
 georobSimulation, [9](#), [14](#), [22](#), [29](#), [33](#), [37](#), [41](#),
 [44](#), [45](#), [57](#), [63](#), [66](#), [70](#)
- kmeans, [11](#)
- legend, [69](#)
 lgnpp, [9](#), [14](#), [22](#), [29](#), [33](#), [37](#), [40](#), [41](#), [44](#), [48](#), [49](#),
 [57](#), [62](#), [63](#), [66](#), [70](#)
- lines.fitted.variogram
 (fit.variogram.model), [17](#)
 lines.georob (plot.georob), [55](#)
 lm, [5](#), [24](#), [27](#), [36](#)
 lmrob, [5](#), [27](#), [28](#), [36](#)
 lmrob.control, [7](#), [36](#)
 loess.smooth, [56](#), [57](#), [72](#)
 logLik.georob, [40](#)
 logLik.georob (georobModelBuilding), [30](#)
- model.frame.georob (georobS3methods), [41](#)
 model.matrix.default, [24](#)
 model.matrix.georob (georobS3methods),
 [41](#)
- na.exclude, [24](#), [68](#)
 na.fail, [24](#), [68](#)
 na.omit, [24](#), [68](#)
 nleqslv, [7](#), [8](#), [26–28](#), [35](#), [39](#)

- nlsminb, [5](#), [7](#), [8](#), [20](#), [26](#), [28](#), [35](#), [39](#)
- nobs.georob (georobS3methods), [41](#)
- numericDeriv, [39](#)
- offset, [24](#)
- optim, [5](#), [7](#), [8](#), [20](#), [21](#), [26–28](#), [35](#), [39](#)
- options, [24](#), [68](#)
- param.bounds, [27](#)
- param.bounds (param.names), [54](#)
- param.names, [15](#), [18](#), [25](#), [26](#), [54](#)
- param.transf, [20](#)
- param.transf (control.georob), [3](#)
- plot, [57](#)
- plot.cv.georob (validate.predictions), [71](#)
- plot.default, [68](#)
- plot.formula, [69](#)
- plot.georob, [9](#), [14](#), [22](#), [29](#), [33](#), [34](#), [36](#), [40](#), [44](#), [48](#), [55](#), [63](#), [66](#), [70](#)
- plot.lmrob, [56](#), [57](#)
- plot.sample.variogram, [57](#)
- plot.sample.variogram (sample.variogram), [66](#)
- pmm, [7](#), [47](#), [58](#), [61](#)
- preCKrige, [61](#)
- predict, [50](#)
- predict.georob, [9](#), [14](#), [22](#), [29](#), [33](#), [34](#), [37](#), [40](#), [41](#), [44](#), [45](#), [48](#), [50](#), [52](#), [53](#), [57](#), [60](#), [66](#), [70](#)
- predict.lm, [61](#), [62](#)
- print.coef.georob (georobS3methods), [41](#)
- print.cv.georob (validate.predictions), [71](#)
- print.fitted.variogram (fit.variogram.model), [17](#)
- print.georob (georobS3methods), [41](#)
- print.summary.cv.georob (validate.predictions), [71](#)
- print.summary.fitted.variogram (fit.variogram.model), [17](#)
- print.summary.georob (georobS3methods), [41](#)
- print.summary.sample.variogram (sample.variogram), [66](#)
- profilelogLik, [9](#), [14](#), [22](#), [28](#), [33](#), [36](#), [40](#), [44](#), [48](#), [57](#), [63](#), [64](#), [70](#)
- Qn, [56](#), [68](#), [74](#)
- random.effects (georobS3methods), [41](#)
- raneef (georobS3methods), [41](#)
- raneef.georob, [40](#)
- resid.georob (georobS3methods), [41](#)
- residuals.georob, [40](#)
- residuals.georob (georobS3methods), [41](#)
- residuals.lm, [42](#)
- RFsimulate, [47](#)
- RMmodel, [15](#), [18](#), [24–27](#), [39](#)
- rq, [5](#), [7](#), [8](#), [27](#), [36](#)
- rstandard.georob, [40](#)
- rstandard.georob (georobS3methods), [41](#)
- runif, [11](#)
- sample.variogram, [9](#), [14](#), [18](#), [29](#), [33](#), [37](#), [40](#), [41](#), [44](#), [48](#), [56](#), [57](#), [63](#), [66](#), [66](#)
- set.seed, [11](#), [45](#)
- SpatialGrid, [45](#), [60](#)
- SpatialGridDataFrame, [45](#), [48](#), [60](#), [62](#)
- SpatialPixels, [45](#), [60](#)
- SpatialPixelsDataFrame, [45](#), [48](#), [60](#), [62](#)
- SpatialPoints, [45](#), [60](#)
- SpatialPointsDataFrame, [24](#), [45](#), [48](#), [60](#), [62](#)
- SpatialPolygonsDataFrame, [48](#), [51](#), [60](#), [62](#)
- step, [30–32](#)
- step (georobModelBuilding), [30](#)
- step.georob, [40](#)
- summary.cv.georob (validate.predictions), [71](#)
- summary.fitted.variogram (fit.variogram.model), [17](#)
- summary.georob, [36](#)
- summary.georob (georobS3methods), [41](#)
- summary.sample.variogram (sample.variogram), [66](#)
- termpplot, [43](#), [60](#)
- terms, [42](#)
- text, [69](#)
- update, [11–13](#), [43](#), [65](#)
- validate.predictions, [10](#), [14](#), [40](#), [71](#)
- vcov.georob (georobS3methods), [41](#)
- waldtest, [31](#), [32](#)
- waldtest (georobModelBuilding), [30](#)
- waldtest.georob, [40](#)
- wolfcamp, [75](#)