Package 'georob'

January 24, 2017

Type Package

Title Robust Geostatistical Analysis of Spatial Data

Version 0.3-3

Date 2017-01-24

Depends R(>= 2.14.0), sp(>= 0.9-60)

Imports constrainedKriging(>= 0.2-1), fields, lmtest, methods, nlme, nleqslv, quantreg, RandomFields(>= 3.0.10), robustbase(>= 0.90-2), snowfall(>= 1.84-6)

Suggests geoR, 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 and for unbiased back-transformation of Kriging predictions of log-transformed data.

License GPL (>= 2)

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2017-01-24 14:20:38

R topics documented:

`georob-package` *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 $y_i = y(\boldsymbol{s}_i)$, are recorded at a set of locations, $\boldsymbol{s}_i$, $i = 1, 2, \ldots, n$, in a domain $G \in \mathbb{R}^d$, $d \in (1, 2, 3)$, along with covariate information $x_j(\boldsymbol{s}_i)$, $j = 1, 2, \ldots, p$.

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

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

where $Z(\boldsymbol{s}_i) = \boldsymbol{x}(\boldsymbol{s}_i)^{\mathrm{T}}\boldsymbol{\beta} + B(\boldsymbol{s}_i)$ is the so-called signal, $\boldsymbol{x}(\boldsymbol{s}_i)^{\mathrm{T}}\boldsymbol{\beta}$ is the external drift, $\{B(\boldsymbol{s})\}$ 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

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

where $\boldsymbol{X}$ is the model matrix with the rows $\boldsymbol{x}(\boldsymbol{s}_i)^{\mathrm{T}}$.
The (generalized) covariance matrix of the vector of spatial Gaussian random effects $\boldsymbol{B}$ is denoted by

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

where σ^2_{n} is the variance of seemingly uncorrelated micro-scale variation in $B(\boldsymbol{s})$ that cannot be resolved with the chosen sampling design, σ^2 is the variance of the captured auto-correlated

variation in $B(\boldsymbol{s})$, $\sigma_Z^2 = \sigma_{\mathrm{n}}^2 + \sigma^2$ is the signal variance, and $\xi = \sigma^2/\sigma_Z^2$. To estimate both σ_{n}^2 and τ^2 (and not only their sum), one needs replicated measurements for some of the $\boldsymbol{s}_i$.

We define $\boldsymbol{V}_\alpha$ to be the matrix with elements

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

where the constant γ_0 is chosen large enough so that $\boldsymbol{V}_\alpha$ is positive definite, $\gamma(\cdot)$ is a valid stationary or intrinsic variogram, and $\boldsymbol{A} = \boldsymbol{A}(\alpha, f_1, f_2; \omega, \phi, \zeta)$ is a matrix that is used to model geometrically anisotropic auto-correlation. In more detail, $\boldsymbol{A}$ 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, $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, $\boldsymbol{p}_3$, equal to $|\boldsymbol{p}_1| = \alpha$, $|\boldsymbol{p}_2| = f_1\,\alpha$ and $|\boldsymbol{p}_3| = f_2\,\alpha$, $0 < f_2 \leq f_1 \leq 1$, 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 $\boldsymbol{p}_1$ (= angle between north and the projection of $\boldsymbol{p}_1$ onto the x-y-plane, measured from north to south positive clockwise in degrees),

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

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

The transformation matrix is given by

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

where

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

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

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

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 $\boldsymbol{Y}$ is given by

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

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. $\sigma^2, \ldots, \alpha, \ldots$ and $f_1, \ldots, \zeta$ 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 $\boldsymbol{B}$ at the data locations $\boldsymbol{s}_i$ and the model parameters $\boldsymbol{\beta}$, τ^2 and $\boldsymbol{\theta}^{\mathrm{T}} = (\sigma^2, \sigma_{\mathrm{n}}^2, \alpha, \ldots, f_1, f_2, \omega, \phi, \zeta)$ are unknown and are estimated in **georob** either by Gaussian or robust restricted maximum likelihood (REML) or Gaussian maximum likelihood (ML). Here ... denote further parameters of the variogram such as the smoothness parameter of the Whittle-Mat¬érn 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 $\boldsymbol{B}$ and the generalized least squares (GLS = ML) estimates of $\boldsymbol{\beta}$ can be obtained simultaneously by maximizing

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

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

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

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

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

The estimating equations can then by 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 *K¬ünsch et al. (2011)* for details. The robustified estimating equations are solved numerically by a combination of iterated re-weighted least squares (IRWLS) to estimate $\boldsymbol{B}$ 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 $\boldsymbol{s}_0$ from the covariates $\boldsymbol{x}(\boldsymbol{s}_0)$, the estimated parameters $\widehat{\boldsymbol{\beta}}$, $\widehat{\boldsymbol{\theta}}$ and the predicted random effects $\widehat{\boldsymbol{B}}$ by

$$\widehat{Y}(\boldsymbol{s}_0) = \widehat{Z}(\boldsymbol{s}_0) = \boldsymbol{x}(\boldsymbol{s}_0)^{\mathrm{T}}\widehat{\boldsymbol{\beta}} + \boldsymbol{\gamma}_{\widehat{\theta}}^{\mathrm{T}}(\boldsymbol{s}_0)\boldsymbol{\Gamma}_{\widehat{\theta}}^{-1}\widehat{\boldsymbol{B}},$$

where $\boldsymbol{\Gamma}_{\widehat{\theta}}$ is the estimated (generalized) covariance matrix of $\boldsymbol{B}$ and $\boldsymbol{\gamma}_{\widehat{\theta}}(s_0)$ is the vector with the estimated (generalized) covariances between $\boldsymbol{B}$ and $B(s_0)$. Kriging variances can be computed as well, based on approximated covariances of $\widehat{\boldsymbol{B}}$ and $\widehat{\boldsymbol{\beta}}$ (see *K¬ünsch et al., 2011*, and appendices of *Nussbaum et al., 2012*, 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 `lgnpp`).

Author(s)

Andreas Papritz <andreas.papritz@env.ethz.ch>
http://www.step.ethz.ch/people/scientific-staff/andreas-papritz.html

References

Nussbaum, M., Papritz, A., Baltensweiler, A. and Walthert, L. (2012) *Organic Carbon Stocks of Swiss Forest Soils*, Institute of Terrestrial Ecosystems, ETH Zurich and Swiss Federal Institute for Forest, Snow and Landscape Research (WSL), pp. 51. http://e-collection.library.ethz.ch/eserv/eth:6027/eth-6027-01.pdf

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. http://e-collection.library.ethz.ch/eserv/eth:7080/eth-7080-01.pdf

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; 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; predict.georob for computing robust Kriging predictions; and finally georobModelBuilding for stepwise building models of class georob; georobMethods for further methods for the class georob, sample.variogram and fit.variogram.model for robust estimation and modelling of sample variograms.

---

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

m	either a single symmetric, lower or upper triangular matrix or a list of such matrices. The type of m (or of its component matrices) must be defined by the attribute struc with possible values "sym" (symmetric), "lt" (lower triangular) or "ut" (upper triangular).
object	a single compressed matrix or a list of such matrices generated by compress, see *Value*. The type of object (or of its components) must be defined by the attribute struc 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 further components than squares 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 *Tuning 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 = "br", 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("dbldog", "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_Z^2, η and ξ are estimated by Gaussian (RE)ML, otherwise the original parameters τ^2, σ_{n}^2 and σ^2 (cf. subsection *Estimating variance parameters by Gaussian (RE)ML*, section *Details* of georob).
maximizer	character keyword defining the Gaussian (restricted) log-likelihood is maximized by nlminb (default) or optim.
initial.param	logical, controlling whether initial values of variogram parameters are computed for solving the estimating equations of the variogram and anisotropy parameters. If initial.param = TRUE (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 lmrob and fitting the spatial linear model by Gaussian REML to the pruned data set. For initial.param = FALSE no initial parameter values are computed and the estimating equations are solved with the initial values passed by param and aniso to georob (see *Details* of georob.
initial.fixef	character keyword defining whether the function lmrob or rq is used to compute robust initial estimates of the regression parameters $\boldsymbol{\beta}$ (default "lmrob"). If the fixed effects model matrix has not full columns rank, then lm is used to compute initial values of the regression coefficients.
bhat	initial values for the spatial random effects $\widehat{\boldsymbol{B}}$, with $\widehat{\boldsymbol{B}} = \mathbf{0}$ if bhat is equal to NULL (default).
min.rweight	positive numeric. "Robustness weight" of the initial lmrob fit that observations must exceed to be used for computing robust initial estimates of variogram parameters by setting initial.param = TRUE (see georob; default 0.25).
param.tf	a function such as param.transf, which returns a named vector of character strings that define the transformations to be applied to the variogram parameters for model fitting, see *Details*.
fwd.tf	a function such as fwd.transf, which returns a named list of invertible functions to be used to transform variogram parameters, see *Details*.
deriv.fwd.tf	a function such as dfwd.transf, which returns a named list of functions corresponding to the first derivatives of fwd.tf, see *Details*.
bwd.tf	a function such as bwd.transf, which returns the named list of inverse functions corresponding to fwd.tf, see *Details*.
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 $\boldsymbol{B}$ and $\boldsymbol{\beta}$ (default 50).
irwls.ftol	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 $\boldsymbol{X}$ (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 $\widehat{\boldsymbol{B}}$ when solving the estimating equations, see *Details*.

`error.family.cov.effects`
character keyword, defining the probability distribution for ε (default: `"gaussian"`) that is used to approximate the covariances of $\widehat{\boldsymbol{\beta}}$, $\widehat{\boldsymbol{B}}$ and $\boldsymbol{B} - \widehat{\boldsymbol{B}}$, see *Details*.

`error.family.cov.residuals`
character keyword, defining the probability distribution for ε (default: `"long.tailed"`) that is used to approximate the covariances of $\widehat{\boldsymbol{\varepsilon}} = \boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}} - \widehat{\boldsymbol{B}}$ and $\widehat{\boldsymbol{\varepsilon}} + \widehat{\boldsymbol{B}} = \boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}$, see *Details*.

`cov.bhat` logical controlling whether the covariances of $\widehat{\boldsymbol{B}}$ are returned by `georob` (default `FALSE`).

`full.cov.bhat` logical controlling whether the full covariance matrix (`TRUE`) or only the variance vector of $\widehat{\boldsymbol{B}}$ is returned (default `FALSE`).

`cov.betahat` logical controlling whether the covariance matrix of $\widehat{\boldsymbol{\beta}}$ is returned (default `TRUE`).

`cov.delta.bhat` logical controlling whether the covariances of $\boldsymbol{B} - \widehat{\boldsymbol{B}}$ are returned (default `TRUE`).

`full.cov.delta.bhat`
logical controlling whether the full covariance matrix (`TRUE`) or only the variance vector of $\boldsymbol{B} - \widehat{\boldsymbol{B}}$ is returned (default `TRUE`).

`cov.delta.bhat.betahat`
logical controlling whether the covariance matrix of $\boldsymbol{B} - \widehat{\boldsymbol{B}}$ and $\widehat{\boldsymbol{\beta}}$ is returned (default `TRUE`).

`cov.ehat` logical controlling whether the covariances of $\widehat{\boldsymbol{\varepsilon}} = \boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}} - \widehat{\boldsymbol{B}}$ are returned (default `TRUE`).

`full.cov.ehat` logical controlling whether the full covariance matrix (`TRUE`) or only the variance vector of $\widehat{\boldsymbol{\varepsilon}} = \boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}} - \widehat{\boldsymbol{B}}$ is returned (default `FALSE`).

`cov.ehat.p.bhat`
logical controlling whether the covariances of $\widehat{\boldsymbol{\varepsilon}} + \widehat{\boldsymbol{B}} = \boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}$ are returned (default `FALSE`).

`full.cov.ehat.p.bhat`
logical controlling whether the full covariance matrix (`TRUE`) or only the variance vector of $\widehat{\boldsymbol{\varepsilon}} + \widehat{\boldsymbol{B}} = \boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}$ 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 *Details*).

`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`.

`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 $\widehat{\boldsymbol{B}}$. These covariances (and the covariances of $\boldsymbol{B}-\widehat{\boldsymbol{B}}$, $\widehat{\boldsymbol{\beta}}$, $\widehat{\boldsymbol{\varepsilon}}$, $\widehat{\boldsymbol{\varepsilon}}+\widehat{\boldsymbol{B}}$) 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 $\widehat{\boldsymbol{\beta}}$, $\widehat{\boldsymbol{B}}$ and $\boldsymbol{B}-\widehat{\boldsymbol{B}}$ (`error.family.cov.effects`) and
- computing the covariances of $\widehat{\boldsymbol{\varepsilon}}=\boldsymbol{Y}-\boldsymbol{X}\widehat{\boldsymbol{\beta}}-\widehat{\boldsymbol{B}}$ and $\widehat{\boldsymbol{\varepsilon}}+\widehat{\boldsymbol{B}}=\boldsymbol{Y}-\boldsymbol{X}\widehat{\boldsymbol{\beta}}$ (`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; `plot.georob` for display of RE(ML) variogram estimates; `predict.georob` for computing robust Kriging predictions; and finally `georobMethods` for further methods for the class georob.

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)
```

---

cv *Generic Cross-validation*

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 = 10, 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 $(1, 2, \ldots, K)$, 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 *Details*.
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 length(object[["variogram.object"]][[1]][["param"]]) columns with initial values of variogram parameters for the nset cross-validation sets, and colnames(param) must match names(object[["variogram.object"]][[1]][["param"]]).
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 length(object[["variogram.object"]][[1]][["fit.param"
columns with variogram parameter fitting flags for the nset cross-validation
sets, and colnames(param) must match names(object[["variogram.object"]][[1]][["fit.param"

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 length(object[["variogram.object"]][[1]][["aniso"]]) columns with initial values of anisotropy parameters for the nset cross-validation sets, and colnames(aniso) must match names(object[["variogram.object"]][[1]][["aniso"]]).

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"
columns with initial values of variogram parameters for the nset cross-validation sets (i is the ith 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"
columns with variogram parameter fitting flags for the nset cross-validation sets (i is the ith 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"
columns with initial values of anisotropy parameters for the nset cross-validation sets (i is the ith 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"
columns with anisotropy parameter fitting flags for the nset cross-validation sets(i is the ith 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 *Details*.
verbose	positive integer controlling logging of diagnostic messages to the console during model fitting. Passed by update to georob.
...	additional arguments passed by update to georob, see *Details*.

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 package **parallel** 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.model`, `param`, `aniso[["aniso"]]`, `coefficients` along with the standard errors of $\widehat{\boldsymbol{\beta}}$, 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

`validate.predictions` for computing statistics of the cross-validation errors; `georob` for (robust) fitting of spatial linear models; `georobObject` for a description of the class `georob`; `predict.georob` for computing robust Kriging predictions.

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

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 $B(\boldsymbol{s})$.
`snugget`	variance (spatial nugget σ_{n}^2) of the seemingly spatially uncorrelated component of $B(\boldsymbol{s})$ (micro-scale spatial variation; default value `snugget = 0`).
`nugget`	variance (nugget τ^2) of the independent errors $\varepsilon(\boldsymbol{s})$.
`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-Matérn model (see `RMmodel` and `param.names`).
`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`).
`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)
```

---

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", "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,
    max.lag = max(sv[["lag.dist"]]), min.npairs = 30,
    weighting.method = c("cressie", "equal", "npairs"), hessian = TRUE,
    verbose = 0, ...)

## 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 $B(\boldsymbol{s})$.
- `snugget`: variance (spatial nugget σ^2_{n}) of the seemingly spatially uncorrelated component of $B(\boldsymbol{s})$ (micro-scale spatial variation; default value `snugget = 0`).
- `nugget`: variance (nugget τ^2) of the independent errors $\varepsilon(\boldsymbol{s})$.
- `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`).
- `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`).

`fit.aniso` a named logical vector (or a function such as `default.fit.aniso` that creates this vector) with the same names as used for `aniso`, defining which parameters are adjusted (`TRUE`) and which are kept fixed at their initial values (`FALSE`) when fitting the model.

`variogram.object`
an optional list that defines a possibly nested variogram model. Each component is itself a list with the following components:

- variogram.model: a character keyword defining the variogram model, see respective argument above.
- param: a named numeric vector with initial values of the variogram parameters, see respective argument above.
- fit.param: a named logical vector defining which parameters are adjusted, see respective argument above.
- aniso: a named numeric vector with initial values for fitting geometrically anisotropic variogram models, see respective argument above.
- fit.param: a named logical vector defining which anisotropy parameters are adjusted, see respective argument above.

Note that the arguments `variogram.model`, `param`, `fit.param`, `aniso` and `fit.aniso` are ignored when `variogram.object` is passed to `fit.variogram.model`.

`max.lag` a positive numeric defining the maximum lag distance to be used for fitting or plotting variogram models (default all lag classes).

`min.npairs` 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`).

`weighting.method`
a character keyword defining the weights for non-linear least squares. Possible values are:

- `"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).

`hessian` logical controlling whether the hessian is computed by `optim`.

`verbose` positive integer controlling logging of diagnostic messages to the console during model fitting.

`object, x` an object of class `fitted.variogram`.

`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 optim or 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.

Value

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

sse	the value of the object function (weighted residual sum of squares) evaluated at the solution.
variogram.model	the name of the fitted parametric variogram model.
param	a named vector with the (estimated) variogram parameters of the fitted model.
fit.param	logical vector indicating which variogram parameters were fitted.
aniso	a list with the following components: • isotropic: logical indicating whether an isotropic variogram was fitted. • aniso: a named numeric vector with the (estimated) anisotropy parameters of the fitted model. • fit.aniso: logical vector indicating which anisotropy parameters were fitted. • sincos: a list with sin and cos of the angles ω, ϕ and ζ that define the orientation of the anisotropy ellipsoid. • rotmat: the matrix $(\boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3)$ (see georobIntro). • sclmat: a vector with the elements 1, $1/f_1$, $1/f_2$ (see georobIntro).
param.tf	a character vector listing the transformations of the variogram parameters used for model fitting.
fwd.tf	a list of functions for variogram parameter transformations.
bwd.tf	a list of functions for *inverse* variogram parameter transformations.
converged	logical indicating whether numerical maximization by optim converged.
convergence.code	a diagnostic integer issued by optim (component convergence) about convergence.
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 giving an estimate of the Hessian at the solution (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; sample.variogram for computing sample variograms.

Examples

```
data(wolfcamp, package = "geoR")

## fitting an isotropic IRF(0) model
r.sv.iso <- sample.variogram(wolfcamp[["data"]], locations = wolfcamp[[1]],
    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),
    method = "Nelder-Mead", hessian = FALSE, control = list(maxit = 5000))
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(wolfcamp[["data"]],
    locations = wolfcamp[[1]], 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),
    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)
```

---

georob *Robust Fitting of Spatial Linear Models*

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 lm and formula for more details.
data	an optional data frame, a SpatialPointsDataFrame, list or environment (or another object coercible by as.data.frame 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 environment(formula), typically the environment from which georob 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 na.action argument of options, and is na.fail if that is unset. The "factory-fresh" default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.

`model, x, y` logicals. If `TRUE` 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 `contrasts.arg` of `model.matrix.default`.

`offset` this optional argument can be used to specify an *a priori* known component to be included in the linear predictor during fitting. An `offset` 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 were 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 *Details* and `RMmodel`).

`param` a named numeric vector with initial values of the variogram parameters (mandatory argument). The names of `param` are matched against the following names (see *Details* 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 $B(\boldsymbol{s})$.
- `snugget`: variance (spatial nugget σ_{n}^2) of the seemingly spatially uncorrelated component of $B(\boldsymbol{s})$ (micro-scale spatial variation; default value `snugget = 0`).
- `nugget`: variance (nugget τ^2) of the independent errors $\varepsilon(\boldsymbol{s})$.
- `scale`: range parameter (α) of the variogram.
- names of additional variogram parameters such as the smoothness parameter ν of the Whittle-Mat¬érn 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`).
- `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`).

`fit.aniso` a named logical vector (or a function such as `default.fit.aniso` that creates this vector) with the same names as used for `aniso`, defining which parameters are adjusted (`TRUE`) and which are kept fixed at their initial values (`FALSE`) when fitting the model.

`variogram.object`
an optional list that defines a possibly nested variogram model. Each component is itself a list with the following components:

- variogram.model: a mandatory character keyword defining the variogram model, see respective argument above.
- param: a mandatory named numeric vector with initial values of the variogram parameters, see respective argument above.
- fit.param: an optional named logical vector defining which parameters are adjusted, see respective argument above.
- aniso: an optional named numeric vector with initial values for fitting geometrically anisotropic variogram models, see respective argument above.
- fit.param: an optional named logical vector defining which anisotropy parameters are adjusted, see respective argument above.

Note that the arguments `variogram.model`, `param`, `fit.param`, `aniso` and `fit.aniso` are ignored when `variogram.object` is passed to `georob`.

`tuning.psi` 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 `control.georob` and see its help page for the components of `control`.

`verbose` positive integer controlling logging of diagnostic messages to the console during model fitting. `verbose = 0` largely suppresses such messages and `verbose = 4` asks for most verbose output (see `control` arguments of `nleqslv`, `nlminb` and `optim` and `control.georob` for information how to fine tuning diagnostic output generated by `nleqslv`, `nlminb` and `optim`).

`...` further arguments passed to function (e.g. `object.` used internally for updating `georob` objects).

Details

`georob` fits a spatial linear model by robust or Gaussian RE(ML) (*K¬ünsch et al., 2011*, *K¬ünsch 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^2_{n}), 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 $\boldsymbol{s}_i$ 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-Mat¬érn 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`).

Computing robust initial estimates of parameters for robust REML:

To solve the robustified estimating equations for $\boldsymbol{B}$ and $\boldsymbol{\beta}$ the following initial estimates are used:

- $\widehat{\boldsymbol{B}} = \mathbf{0}$, if this turns out to be infeasible, initial values can be passed to `georob` by the argument `bhat` of `control.georob`.
- $\widehat{\boldsymbol{\beta}}$ 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` 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 $\psi_c(\widehat{\varepsilon}_i/\widehat{\tau})/(\widehat{\varepsilon}_i/\widehat{\tau}) \leq$ `min.rweight`, are discarded (see `control.georob`).
3. The model is fit to the pruned data set by Gaussian REML using `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 `snugget` ($\sigma^2_{\rm n}$), for Gaussian (RE)ML the variances can be re-parametrized to

- the signal variance $\sigma^2_Z = \sigma^2 + \sigma^2_{\rm n}$,
- the inverse relative nugget $\eta = \sigma^2_Z/\tau^2$ and
- the relative auto-correlated signal variance $\xi = \sigma^2/\sigma^2_Z$.

`georob` maximizes then a (restricted) *profile log-likelihood* that depends only on η, ξ, α, ..., and σ^2_Z 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, $\sigma^2_{\rm n}$ 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>
http://www.step.ethz.ch/people/scientific-staff/andreas-papritz.html
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. http://e-collection.library.ethz.ch/eserv/eth:7080/eth-7080-01.pdf

See Also

georobIntro for a description of the model and a brief summary of the algorithms; 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; cv.georob for assessing the goodness of a fit by georob; predict.georob for computing robust Kriging predictions; and finally georobModelBuilding for stepwise building models of class georob; georobMethods for further methods for the class georob.

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, package = "geoR")
d.wolfcamp <- data.frame(x = wolfcamp[[1]][,1], y = wolfcamp[[1]][,2],
    pressure = wolfcamp[[2]])

## fitting isotropic IRF(0) model

r.irf0.iso <- georob(pressure ~ 1, data = d.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 = d.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)
```

`georob-S3methods` *Common S3 Methods for Class* `georob`

Description

This page documents the methods `fixef`, `fixed.effects`, `model.frame`, `model.matrix`, `nobs`, `print`, `ranef`, `random.effects`, `resid`, `residuals`, `rstandard`, `summary` and `vcov` for the class `georob`.

Usage

```
## 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 $\widehat{B}(\boldsymbol{s})+\widehat{\varepsilon}(\boldsymbol{s})$ and level = 1 (default) only the estimated errors $\widehat{\varepsilon}(\boldsymbol{s})$.
signif	confidence level for computing confidence intervals for variogram parameters (default 0.95).
standard	logical controlling whether the spatial random effects $\boldsymbol{B}$ should be standardized (default FALSE).
type	character keyword indicating the type of residuals to compute, see residuals.lm. type = "huber" computes 'huberized' residuals $\widehat{\sigma}/\gamma_1\psi(\widehat{\varepsilon}(\boldsymbol{s})/\widehat{\sigma})$.
terms	If type = "terms", which terms (default is all terms).
...	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 $\widehat{\varepsilon}(\boldsymbol{s})$ or the sum of the latter quantities and the spatial random effects $\widehat{B}(\boldsymbol{s})$. 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 regression coefficients, which may of course also be obtained by coef.

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

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; georobModelBuilding for stepwise building models of class georob; georobObject for a description of the class georob.

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,
    control = control.georob(cov.bhat = TRUE, cov.ehat.p.bhat = TRUE))
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)
```

`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, data = NULL, 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, data = NULL, 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, data = NULL,
    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 georob, see georobObject.
data	an optional data frame.
direction	the mode of stepwise search, see step.
fixed,fixed.add1.drop1	logical controlling whether the variogram parameters are *not* adjusted when adding or dropping model terms by add1 and drop1 (default TRUE), see *Details*.
fixed.step	logical controlling whether the variogram parameters are *not* adjusted after having called add1 and drop1 in step (default TRUE), see *Details*.
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 add1 and drop1. If larger than one then the minimum of ncores, detectCores() 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 TRUE).
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 waldtest.default).

Details

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

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

(see georob-package 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 $\tau^2/\boldsymbol{w}$ where $\boldsymbol{w}$ 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; georobMethods for further methods for the class georob.

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,
    control = control.georob(cov.bhat = TRUE, cov.ehat.p.bhat = TRUE))
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 *Fitted georob Object*

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:

loglik — the maximized (restricted) Gaussian log-likelihood of a non-robust (RE)ML fit or NA for a robust fit if tuning.psi is less than tuning.psi.nr.

variogram.object — the estimated parameters of a possibly nested variograms model. This is a list that contains for each variogram model structure the following components:

- variogram.model: the name of the fitted parametric variogram model.
- param: a named numeric vector with the (estimated) variogram parameters.
- fit.param: a named logical vector with the flags defining what variogram parameters were estimated.
- isotropic: logical indicating whether an isotropic variogram was fitted.
- aniso: a named numeric vector with the (estimated) anisotropy parameters.
- sincos: a list with sin and cos of the angles ω, ϕ and ζ that define the orientation of the anisotropy ellipsoid.
- rotmat: the matrix $(\boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3)$ (see georobIntro).
- sclmat: a vector with the elements 1, $1/f_1$, $1/f_2$ (see georobIntro).

gradient — 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 .

tuning.psi — the value of the tuning constant c of the ψ_c-function.

coefficients — a named vector with the estimated regression coefficients.

fitted.values — a named vector with the fitted values of the external drift $\boldsymbol{X}\widehat{\boldsymbol{\beta}}$.

bhat — a named vector with the predicted spatial random effects $\widehat{\boldsymbol{B}}$ at the data locations.

residuals — a named vector with the residuals $\widehat{\boldsymbol{\varepsilon}} = \boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}} - \widehat{\boldsymbol{B}}$.

rweights — a named numeric vector with the "robustness weights" $\psi(\widehat{\varepsilon}_i/\widehat{\tau})/(\widehat{\varepsilon}_i/\widehat{\tau})$.

converged — logical indicating whether numerical maximization of the (restricted) log-likelihood by nlminb or optim or root finding by nleqslv converged.

convergence.code — a diagnostic integer issued by nlminb, optim (component convergence) or nleqslv (component termcd) about convergence.

`iter` a named integer vector of length two, indicating either

- the number of function and gradient evaluations when maximizing the (restricted) Gaussian log-likelihood by `nlminb` or `optim`, or
- the number of function and Jacobian evaluations when solving the robustified estimating equations by `nleqslv`.

`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 *compressed form* (see `compress`) and can be expanded to square matrices by `expand`. What `cov` actually contains depends on the flags passed to `georob` for computing covariances (see `control.georob`). Possible components are:

- `cov.bhat`: the covariances of $\widehat{\boldsymbol{B}}$.
- `cov.betahat`: the covariances of $\widehat{\boldsymbol{\beta}}$.
- `cov.delta.bhat`: the covariances of $\boldsymbol{B} - \widehat{\boldsymbol{B}}$.
- `cov.delta.bhat.betahat`: the covariances of $\boldsymbol{B} - \widehat{\boldsymbol{B}}$ and $\widehat{\boldsymbol{\beta}}$.
- `cov.ehat`: the covariances of $\widehat{\boldsymbol{\varepsilon}} = \boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}} - \widehat{\boldsymbol{B}}$.
- `cov.ehat.p.bhat`: the covariances of $\widehat{\boldsymbol{\varepsilon}} + \widehat{\boldsymbol{B}} = \boldsymbol{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}$.
- `cov.pred.target`: a covariance term required for the back-transformation of Kriging predictions of log-transformed data.

`expectations` a named numeric vector with the expectations of $\partial\psi_c(x)/\partial x$ (`dpsi`) and $\psi_c^2(x)$ (`psi2`) with respect to a standard normal distribution (`exp.gauss`) and the long-tailed distribution of ε (`exp.f0`) implied by the choice of the ψ_c-function.

`Valphaxi.objects`

a list of matrices in *compressed form* with (among others) the following components:

- `Valpha`: a list with the (generalized) correlation matrices (`Valpha`) of the nested variogram models structures along with the constants (`gcr.constant`) added to the respective semivariances matrices.
- `Valphaxi`: the (generalized) correlation matrix $\boldsymbol{V}_{\alpha,\xi} = \boldsymbol{\Gamma}_{\alpha,\xi}/(\sigma_{\mathrm{n}}^2 + \sigma^2)$ that includes the spatial nugget effect.
- `Valphaxi.inverse`: the inverse of $\boldsymbol{V}_{\alpha,\xi}$.
- `log.det.Valphaxi`: $\log(\det(\boldsymbol{V}_{\alpha,\xi}))$.

`zhat.objects` a list of matrices in (partly) *compressed form* with the following components:

- `Aalphaxi`: the matrix $(\boldsymbol{X}^T\boldsymbol{V}_{\alpha,\xi}^{-1}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{V}_{\alpha,\xi}^{-1}$.
- `Palphaxi`: the matrix $\boldsymbol{I} - \boldsymbol{X}\boldsymbol{A}_{\alpha,\xi}$.
- `Valphaxi.inverse.Palphaxi`: the matrix $\boldsymbol{V}_{\alpha,\xi}^{-1}\boldsymbol{P}_{\alpha,\xi}$.

`locations.object`

a list with 3 components:

- `locations`: a formula indicating the coordinates of the measurement locations.

- `locations.coords`: 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 $\boldsymbol{\beta}$ computed either by `lmrob` or `rq`.
  - `bhat`: initial predictions of $\boldsymbol{B}$.
  - 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`
: a symmetric matrix giving an estimate of the Hessian at the solution 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 $\boldsymbol{X}$ (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; `control.georob` for controlling the behaviour of georob; `plot.georob` for display of (RE)ML variogram estimates; `cv.georob` for assessing the goodness of a fit by georob; `predict.georob` for computing robust Kriging predictions; and finally `georobModelBuilding` for stepwise building models of class georob; `georobMethods` for further methods for the class georob.

---

lgnpp *Unbiased Back-Transformations for Log-normal Kriging*

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

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

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:

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

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

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

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

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

The expressions for the required covariance terms can be found in the Appendices of Nussbaum et al. (2012). Instead of the signal $Z(\boldsymbol{s})$, predictions of the log-transformed response $Y(\boldsymbol{s})$ or the estimated trend $\boldsymbol{x}(\boldsymbol{s})^{\mathrm{T}}\widehat{\boldsymbol{\beta}}$ 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)

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

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

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

$$\boldsymbol{M}(A) = 1/|A| \int_A (\boldsymbol{x}(\boldsymbol{s}) - \boldsymbol{x}(A))(\boldsymbol{x}(\boldsymbol{s}) - \boldsymbol{x}(A))^{\mathrm{T}}\, d\boldsymbol{s}$$

within the block A where

$$\boldsymbol{x}(A) = 1/|A| \int_A \boldsymbol{x}(\boldsymbol{s})\, d\boldsymbol{s}$$

and

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

This back-transformation is based on the assumption that both the point data $U(\boldsymbol{s})$ 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 $\boldsymbol{M}(A)$, 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 $\boldsymbol{M}(A)$ 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

$$\widehat{U}(A) \approx \frac{1}{K}\sum_{s_i \in A} \widehat{U}(\boldsymbol{s}_i)$$

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

$$\mathrm{E}_{\hat{\theta}}[\{U(A) - \widehat{U}(A)\}^2] \approx \frac{1}{K^2}\sum_{s_i \in A}\sum_{s_j \in A} \mathrm{Cov}_{\hat{\theta}}[U(\boldsymbol{s}_i) - \widehat{U}(\boldsymbol{s}_i), U(\boldsymbol{s}_j) - \widehat{U}(\boldsymbol{s}_j)].$$

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., 2012):

- 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 $\widehat{U}(A)$.
- The mean squared error is then approximated by

$$\mathrm{E}_{\hat{\theta}}[\{U(A) - \widehat{U}(A)\}^2] \approx \frac{1}{K^2} \sum_{s_i \in A} \mathrm{E}_{\hat{\theta}}[\{U(\boldsymbol{s}_i) - \widehat{U}(\boldsymbol{s}_i)\}^2]$$

$$+ \frac{K-1}{Kk(k-1)} \sum_{s_i \in \text{sample}} \sum_{s_j \in \text{sample}, s_j \neq s_i} \mathrm{Cov}_{\hat{\theta}}[U(\boldsymbol{s}_i) - \widehat{U}(\boldsymbol{s}_i), U(\boldsymbol{s}_j) - \widehat{U}(\boldsymbol{s}_j)].$$

  The first term of the RHS (and $\widehat{U}(A)$) 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

$$\widehat{U}(A) \approx \frac{1}{k} \sum_{s_i \in \text{sample}} \widehat{U}(\boldsymbol{s}_i)$$

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

$$\frac{1}{kK} \sum_{s_i \in \text{sample}} \mathrm{E}_{\hat{\theta}}[\{U(\boldsymbol{s}_i) - \widehat{U}(\boldsymbol{s}_i)\}^2].$$

- 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. (2012) *Organic Carbon Stocks of Swiss Forest Soils*, Institute of Terrestrial Ecosystems, ETH Zurich and Swiss Federal Institute for Forest, Snow and Landscape Research (WSL), pp. 51. http://e-collection.library.ethz.ch/eserv/eth:6027/eth-6027-01.pdf

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`	integer equal 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 • "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. `xy.angle.def` 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 `c(0,180)`.

`xz.angle.def` an numeric vector defining angular classes in the x-z-plane for computing directional variograms. `xz.angle.def` 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 `c(0,180)`.

`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:

- `"qn"`: Genton's robust `Qn`-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"`. `level = 0` (default for `what == "ta"`) extracts the regression residuals $\widehat{B}(\boldsymbol{s}) + \widehat{\varepsilon}(\boldsymbol{s})$ and `level = 1` (default for `what == "qq.res"`) only the estimated errors $\widehat{\varepsilon}(\boldsymbol{s})$.

`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.

`cex.id` magnification of point labels (see `plot.lmrob`).

`label.pos` positioning of labels, for the left half and right half of the graph respectively (see `plot.lmrob`).

`from` numeric, minimal lag distance for plotting variogram models.

`to` numeric, maximum lag distance for 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	optional color of points and curves to distinguish items relating to different azimuth angles in x-y-plane.
pch	optional symbol for points and curves to distinguish items relating to different azimuth angles in x-z-plane.
lty	line type for plotting variogram models.
xlab, ylab, main	test annotation, see plot.
...	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; sample.variogram for computing 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, ...)
```

Arguments

A, B	matrices to be multiplied.
control	a list of with the arguments pmm.ncores, gcr.ncores, max.ncores, f, sfstop and allow.recursive or a function such as control.pcmp that generates such a list.
pmm.ncores	number (integer, default 1) of cores used for parallelized matrix multiplication.
gcr.ncores	number (integer, default 1) of cores used for parallelized computation of semi-variance matrix.
max.ncores	maximum number of cores (integer, default all cores of a machine) used for parallelized computations.
f	number (integer, default 2) of tasks assigned to each core in parallelized operations.
sfstop	logical controlling whether the SNOW socket cluster is stopped after each parallelized matrix multiplication on windows OS (default FALSE).
allow.recursive	logical controlling whether nested parallelized computation should be allowed (default TRUE).
...	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 http://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>

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)))
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,
    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", see georobObject.
newdata	an optional data frame, SpatialPointsDataFrame, SpatialPixelsDataFrame, SpatialGridDataFrame, SpatialPolygonsDataFrame or an (optioanal) 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 • "signal": the "signal" $Z(\boldsymbol{s}) = \boldsymbol{x}(\boldsymbol{s})^{\mathrm{T}}\boldsymbol{\beta} + B(\boldsymbol{s})$ of the process (default), • "response": the observations $Y(\boldsymbol{s}) = Z(\boldsymbol{s}) + \varepsilon(\boldsymbol{s})$, • "trend": the external drift $\boldsymbol{x}(\boldsymbol{s})^{\mathrm{T}}\boldsymbol{\beta}$, • "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: object[["locations.objects"]]$locations).
control	a list with the arguments full.covmat, extended.output, mmax, ncores, pwidth, pheight, napp and pcmp or a function such as control.predict.georob 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 *Value* for an explanation of the effect of full.covmat.
extended.output	logical controlling whether the covariance matrices of the Kriging predictions and of the data should be computed, see *Details* (default FALSE).
mmax	integer equal to the maximum number (default 10000) of prediction items, computed in a sub-task, see *Details*.
ncores	positive integer controlling how many cores are used for parallelized computations, see *Details*.
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 pmm or a function such as control.pcmp that generates such a list (see control.pcmp for allowed arguments).
verbose	positive integer controlling logging of diagnostic messages to the console. verbose = 0 (default) largely suppresses such messages.
...	arguments passed to control.predict.georob.

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 package **parallel** for parallelized computation of Kriging predictions. If there are m items to predict, the task is split into ceiling(m/mmax) 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 lgnpp. In more detail, the following items are computed:

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

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, $Y(\boldsymbol{s}) - \widehat{Y}(\boldsymbol{s})$ or $Z(\boldsymbol{s}) - \widehat{Z}(\boldsymbol{s})$ 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 Walthert, L. (2012) *Organic Carbon Stocks of Swiss Forest Soils*, Institute of Terrestrial Ecosystems, ETH Zurich and Swiss Federal Institute for Forest, Snow and Landscape Research (WSL), pp. 51. http://e-collection.library.ethz.ch/eserv/eth:6027/eth-6027-01.pdf

K¬ünsch, 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. http://e-collection.library.ethz.ch/eserv/eth:7080/eth-7080-01.pdf

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.

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.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)
```

`profilelogLik` *Profile Likelihood*

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 *Details*.

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 package **parallel** 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 refitting 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

georob for (robust) fitting of spatial linear models.

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)))
plot(loglik~scale, r.prflik, type="l")
abline(v=r.logzn.ml$param["scale"], lty="dotted")
abline(h=r.logzn.ml$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 (sample.variogram.default), a formula, specifying in its left part the response variable (right part of formula is ignored, sample.variogram.formula), an object of class georob (sample.variogram.georob) or an object of class sample.variogram (summary.sample.variogram).
locations	a numeric matrix with the coordinates of the locations where the response was observed (sample.variogram.default) or a one-sided formula specifying the coordinates (sample.variogram.formula). 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 coordinates where the data was recorded. If not found in data, the variables are taken from environment(formula), typically the environment from which sample.variogram is called.
subset	an optional vector specifying a subset of observations to be used for estimating the variogram.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action argument of options, and is na.fail if that is unset. The "factory-fresh" default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
lag.dist.def	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 lag.dist.def which is the lower bound of the first bin).
xy.angle.def	an numeric vector defining angular classes in the horizontal plane for computing directional variograms. xy.angle.def must contain an ascending sequence of azimuth angles in degrees from north (positive clockwise to south), see *Details*. Omnidirectional variograms are computed with the default c(0,180).
xz.angle.def	an numeric vector defining angular classes in the x-z-plane for computing directional variograms. xz.angle.def must contain an ascending sequence of angles in degrees from zenith (positive clockwise to nadir), see *Details*. Omnidirectional variograms are computed with the default c(0,180).
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: • "qn": Genton's robust Qn-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).

`x` an object of class `sample.variogram`.

`type, xlim, ylim, xlab, ylab`
see respective arguments of `plot.default`.

`add` logical controlling whether a new plot should be generated (FALSE, default) or whether the information should be added to the current plot (TRUE).

`col` the color of plotting symbols for distinguishing semi variances for angular classes in the x-y-plane.

`pch` the type of plotting symbols for distinguishing semi variances for angular classes in the x-z-plane.

`lty` 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 ith angular class is defined by the vector elements, say l and u, with indices i and $i + 1$. A lag vector belongs to the ith 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.x	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; fit.variogram.model for fitting variogram models to sample variograms.

Examples

```
data(wolfcamp, package = "geoR")

## fitting an isotropic IRF(0) model
r.sv.iso <- sample.variogram(wolfcamp[["data"]], locations = wolfcamp[[1]],
    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(wolfcamp[["data"]],
    locations = wolfcamp[[1]], 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 *Details*): • "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 *Details* (default: min(500, length(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 plot.cv.georob. Possible values are: • "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 *Details*. • "ecdf.pit": empirical CDF of probability integral transform, see *Details*. • "mc": a marginal calibration plot, see *Details*, • "bs": a plot of Brier score vs. threshold, see *Details*.
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 *Details*) 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 $Y(\boldsymbol{s}) - \widehat{Y}(\boldsymbol{s})$ 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 $\widehat{\tau}^2$ 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),

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

where $F_i(y_i)$ denotes the (plug-in) predictive CDF evaluated at y_i, the value of the ith (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)

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

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

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

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 an uniform distribution.
- Plots of $\bar{F}_n(y)$ and of the empirical CDF of the data, say $\widehat{G}_n(y)$, and of their difference, $\bar{F}_n(y) - \widehat{G}_n(y)$ vs y. The forecasts are said to be *marginally calibrated* if $\bar{F}_n(y)$ and $\widehat{G}_n(y)$ match.
- A plot of $\mathrm{BS}(y)$ 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 `ncutoff` of them)
`ghat`	the empirical CDF of the (cross-)validation observations $\widehat{G}_n(y)$
`fbar`	the average predictive distribution $\bar{F}_n(y)$
`bs`	the Brier score $\mathrm{BS}(y)$

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)
```

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

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", add = TRUE, 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)
```

Index