Package 'geoRglm'

July 2, 2014

Version 0.9-4

Date 2013-12-11

Title geoRglm - a package for generalised linear spatial models

Author Ole F. Christensen <OleF.Christensen@agrsci.dk> and Paulo J. Ribeiro Jr <Paulo.Ribeiro@est.ufpr.br>

Maintainer Ole Christensen <OleF.Christensen@agrsci.dk>

Depends R (>= 2.2.0), geoR (>= 1.6-25), stats

Suggests coda

Description Functions for inference in generalised linear spatial models. The posterior and predictive inference is based on Markov chain Monte Carlo methods. Package geoRglm is an extension to the package geoR, which must be installed first.

License GPL (>= 2)

URL http://gbi.agrsci.dk/~ofch/geoRglm

LazyLoad yes

LazyData yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-12-13 08:46:40

R topics documented:

asympvar 2
b50, p50 and b64 3
binom.krige 4
binom.krige.bayes 7
covariog 11

covariog.model.env . . . 13
create.mcmc.coda . . . 15
geoRglm-defunct . . . 16
glsm.krige . . . 16
glsm.mcmc . . . 18
hist.glm.krige.bayes . . . 21
image.glm.krige.bayes . . . 22
krige.glm.control . . . 23
likfit.glsm . . . 25
lines.covariomodel . . . 28
mcmc.control . . . 29
model.glm.control . . . 30
output.glm.control . . . 31
plot.covariogram . . . 32
pois.krige . . . 33
pois.krige.bayes . . . 36
prepare.likfit.glsm . . . 40
prior.glm.control . . . 41
proflik.glsm . . . 42
rongelap . . . 44
summary.likGLSM . . . 45

Index **47**

`asympvar` *Asymptotic Variance*

Description

Calculates the initial monotone/positive sequence estimate of the asymptotic variance from CLT (Geyer 92). Useful for estimation of the variance of a Markov Chain Monte Carlo estimate.

Usage

```
asympvar(timeseries, type="mon", lag.max = 100, messages)
```

Arguments

`timeseries`	a vector with a timeseries, or a matrix where the rows are different timeseries.
`type`	`"pos"` and `"mon"` gives the monotone and the positive sequence estimator, respectively, and `"all"` gives both. Default is `type="mon"`.
`lag.max`	maximum lag at which to calculate the asymptotic variance. Default is `lag.max = 100`.
`messages`	logical. If `TRUE`, the default, status messages are printed while the function is running.

Value

A number (or a vector) with the estimate, when `type="mon"` or `type="pos"`. A list with components `mon` and `pos` when `type="all"`

Author(s)

The orginal Splus version of this function was written by Rasmus Waagepetersen. R port by
Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

References

Geyer, C. (1992). Practical Monte Carlo (with discussion). *Statist. Sci.* **7**, 473-511.

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

Examples

```
data(p50)
## Not run:
test <- pois.krige(p50, krige = krige.glm.control(cov.pars = c(1,1), beta = 1),
      mcmc.input = mcmc.control(S.scale = 0.5, n.iter = 1000, thin = 1))
asympvar(test$intensity[45,])
ass <- asympvar(test$intensity[1:10,], type = "pos")

## End(Not run)
```

`b50, p50 and b64` *Simulated Data sets which Illustrate the Usage of the Package geoRglm*

Description

The simulated data sets `b50` and `p50` are used in several examples throughout the package documentation. The simulated data set `b64` was used in Diggle, Ribeiro and Christensen (2003), and Christensen and Ribeiro (2002).

Usage

```
data(b50)
data(p50)
data(b64)
```

Format

The object is a list with the following components:

`coords` the coordinates of data locations.

`data` the simulated data from the binomial-logit normal model for `b50` and `b64`, and the simulated data from the Poisson-log normal model for `p50`.

`units.m` n-dimensional vector giving the number of trials for `b50` and `b64`, and the observation times for `p50`.

`cov.model` the correlation model used for the simulation.

`nugget` the values of the `nugget` parameter used for the simulation.

`cov.pars` the covariance parameters used for the simulation.

`kappa` the value of the smoothness parameter κ used for the simulation.

Source

Simulated data sets.

References

Peter J. Diggle, Paulo J. Ribeiro and Ole F. Christensen (2003). An introduction to model-based geostatistics. In : *Spatial statistics and computational methods* (ed. J. Møller), Springer Verlag, 43-86.

Christensen, O. F. and Ribeiro Jr, P. J. (2002). geoRglm - a package for generalised linear spatial models. *R News*, 2(2), 26-28.

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

`binom.krige` *Conditional Simulation and Prediction for the Binomial-logit Spatial model*

Description

This function performs conditional simulation (by MCMC) and spatial prediction in the Binomial logit-normal model for fixed covariance parameters. Available types of prediction are: *SK* (simple kriging; fixed beta), *OK* (ordinary kriging; uniform prior on beta).

Usage

```
binom.krige(geodata, coords = geodata$coords, data = geodata$data,
              units.m = "default", locations = NULL, borders,
              mcmc.input, krige, output)
```

Arguments

geodata	a list containing elements coords and data as described next. Typically an object of the class "geodata" - a **geoR** data set. If not provided the arguments coords and data must be provided instead. The list may also contain an argument units.m as described below.
coords	an $n \times 2$ matrix, each row containing Euclidean coordinates of the n data locations. By default it takes the element coords of the argument geodata.
data	a vector with data values. By default it takes the element data of the argument geodata.
units.m	n-dimensional vector of observation times for the data. By default (units.m = "default"), it takes geodata$units.m in case this exist and else a vector of 1's.
locations	an $N \times 2$ matrix or data frame, or a list with the 2-D coordinates of the N prediction locations.
borders	optional. If a two column matrix defining a polygon is provided the prediction is performed only at locations inside this polygon.
mcmc.input	input parameter for the MCMC algorithm. It can take an output from mcmc.control or a list with elements as for the arguments in mcmc.control. See documentation for mcmc.control. ATTENTION: the argument S.scale is necessary while all the others have default values.
krige	defines the model components and the type of kriging. It can take an output from krige.glm.control or a list with elements as for the arguments in krige.glm.control. See documentation for krige.glm.control.
output	parameters for controlling the output. It can take an output from output.glm.control or a list with elements as for the arguments in output.glm.control. See documentation for output.glm.control.

Details

For simulating the conditional distribution of S given y, the Langevin-Hastings algorithm with the parametrisation in Papaspilliopoulus, Roberts and Skold (2003) is used. This algorithm is a Metropolis-Hastings algorithm, where the proposal distribution uses gradient information from the log-posterior distribution.

The proposal variance (called S.scale; see mcmc.control) for the algorithm needs to be scaled such that approximately 60 percent of the proposals are accepted. We also recommend that the user to studies plots of the autocorrelations.

The prediction part of the program consist of performing trans-Gaussian kriging on each of the simulated $g^{-1}(S)$-"datasets" from the conditional distribution. Afterwards the predictor is obtained by taking the mean of prediction means, and the prediction variance is obtained by taking the mean of the prediction variances plus the variance of the prediction means. The trans-Gaussian kriging is done by calling an internal function which is an extension of krige.conv allowing for more than one "data set", and using a second order Taylor approximation of the inverse logit function g^{-1}.

Value

A list with the following components:

`predict`	a vector with predicted values.
`krige.var`	a vector with predicted variances.
`mcmc.error`	estimated Monte Carlo errors on the predicted values.
`beta.est`	estimate of the β parameter. Not included in the output if `type.krige = "sk"`.
`prevalence`	an $n \times n.sim$ matrix with $n.sim$ being the number of MCMC simulations. containing $g^{-1}(S_i)$. Each column corresponds to a conditional simulation of the conditional distribution of $g^{-1}(S_i)$ at the data locations. Only returned when no prediction locations are given.
`acc.rate`	matrix with acceptance rates from MCMC. Only returned when no prediction locations are given.
`simulations`	an $ni \times n.sim$ matrix where ni is the number of prediction locations and $n.sim$ is the number of MCMC simulations. Each column corresponds to a conditional simulation of the predictive distribution $g^{-1}(S^*)$. Only returned if `sim.predict = TRUE`.
`call`	the function call.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

References

O. Papaspiliopoulus and G. O. Roberts and M. Skold (2003). Non-centered parameterizations for hierarchical models and data augmentation. *Bayesian statistics 7* (eds. J. M. Bernardo, S. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith and M. West), Oxford University Press, 307-326.

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

See Also

`binom.krige.bayes` for Bayesian prediction in the Binomial-normal model, `pois.krige` for prediction with fixed parameters in the Poisson normal model, and `krige.conv` for prediction in the linear Gaussian model.

Examples

```
if(!exists(".Random.seed", envir=.GlobalEnv, inherits = FALSE)) set.seed(1234)
data(b50)
# First we scale the algorithm, and study how well the chain is mixing.
test <- binom.krige(b50, krige = list(cov.pars = c(1,1), beta = 1),
      mcmc.input = mcmc.control(S.scale = 0.2, thin = 1))
```

```
plot(qlogis(test$prevalence[45,]), type = "l")
acf(qlogis(test$prevalence[45,]), type = "correlation", plot = TRUE)
## Not run: # Now we make prediction (we decide to thin to every 10, which is the default),
# where we now use S.scale = 0.7.
test2 <- binom.krige(b50, locations = cbind(c(0.5,0.5, 1, 1), c(0.4, 1, 0.4, 1)),
      krige = krige.glm.control(cov.pars = c(1,1), beta = 1),
      mcmc.input = mcmc.control(S.scale = 0.7))
image(test2)
contour(test2)
## End(Not run)
```

`binom.krige.bayes` *Bayesian Posterior Simulation and Prediction for the Binomial Spatial model*

Description

This function performs posterior simulation (by MCMC) and spatial prediction in the binomial-logit spatial model.

Usage

```
binom.krige.bayes(geodata, coords = geodata$coords, data = geodata$data,
         units.m = "default", locations = "no", borders,
         model, prior, mcmc.input, output)
```

Arguments

`geodata`	a list containing elements `coords` and `data` as described next. Typically an object of the class `"geodata"` - a **geoR** data set. If not provided the arguments `coords` and `data` must be given instead. The list may also contain an argument `units.m` as described below.
`coords`	an $n \times 2$ matrix, each row containing Euclidean coordinates of the n data locations. By default it takes the element `coords` of the argument `geodata`.
`data`	a vector with data values. By default it takes the element `data` of the argument `geodata`.
`units.m`	n-dimensional vector giving the number of trials for the data. By default (`units.m = "default"`), it takes `geodata$units.m` in case this exist and else a vector of 1's.
`locations`	an $N \times 2$ matrix or data frame, or a list with the 2-D coordinates of the N prediction locations.
`borders`	optional. If a two column matrix defining a polygon is provided the prediction is performed only at locations inside this polygon.
`model`	a list defining the components of the model. It can take an output from `model.glm.control` or a list with elements as for the arguments in `model.glm.control`. See documentation for `model.glm.control`. All arguments have default values

`prior` specification of priors for the model parameters. It can take an output from `prior.glm.control` or a list with elements as for the arguments in `prior.glm.control`. See documentation for `prior.glm.control`. ATTENTION: When `phi.prior = "fixed"` then `phi` must be specified, and when `phi.prior` is not `"fixed"` then `phi.discrete` must be specified. All other parameters have default values.

`mcmc.input` input parameter for the MCMC algorithm. It can take an output from `mcmc.control` or a list with elements as for the arguments in `mcmc.control`. See documentation for `mcmc.control`.
ATTENTION: the argument `S.scale` must be specified, the argument `phi.start` must specified when `prior$phi` is not `"fixed"`, while all the others have default values.

`output` parameters for controlling the output. It can take an output from `output.glm.control` or a list with elements as for the arguments in `output.glm.control`. See documentation for `output.glm.control`.

Details

`binom.krige.bayes` is a function for Bayesian geostatistical inference in the binomial-logit spatial model.

The Bayesian algorithm is using a discretized version of the prior distribution for the parameter ϕ. This means that the prior for ϕ is always a proper prior.

For simulating from the posterior distribution of S given y, we use a Langevin-Hastings type algorithm. This algorithm is a Metropolis-Hastings algorithm, where the proposal distribution uses gradient information from the posterior. The algorithm is described below. For shortness of presentation, we only present the MCMC algorithm for the case where β follows a uniform prior.

When β follows a uniform prior and the prior for σ^2 is a scaled inverse-χ^2 distribution, the marginalised improper density of S is

$$f_I(s) \propto |D^T V^{-1} D|^{1/2} |V|^{-1/2} \{n_\sigma S_\sigma^2 + s^T (V^{-1} - V^{-1} D (D^T V^{-1} D)^{-1} D^T V^{-1}) s\}^{-(n-p+n_\sigma)/2},$$

where V is the correlation matrix of S. The uniform prior for σ^2 corresponds to $S_\sigma^2 = 0$ and $n_\sigma = -2$, and the reciprocal prior for σ^2 corresponds to $S_\sigma^2 = 0$ and $n_\sigma = 0$.

We use the reparametrisation $S = Q\Gamma$, where Q is the Cholesky factorisation of V so that $V = QQ^T$. Posterior simulations of S are obtained by transforming MCMC simulations from the conditional distribution of Γ given $Y = y$.

The log posterior density of Γ given $Y = y$ is

$$\log f(\gamma|y) = const(y) - \frac{1}{2}\gamma^T (I_n - V^{-1/2} D (D^T V^{-1} D)^{-1} D^T V^{-1/2}) \gamma + \sum_{i=1}^{n} y_i s_i - \log(1 + \exp(s_i)),$$

where $(s_1, \ldots, s_n)^T = Q\gamma$.

For the Langevin-Hastings update we use the gradient of the log target density,

$$\nabla(\gamma)^{trunc} = -(I_n - Q^{-1} D (D^T V^{-1} D)^{-1} D^T (Q^{-1})^T) \gamma + Q^T \{y_i - \exp(s_i)/(1 + \exp(s_i))\}_{i=1}^n .$$

The proposal γ' follows a multivariate normal distribution with mean vector $\xi(\gamma) = \gamma + (h/2)\nabla(\gamma)^{trunc}$ and covariance matrix hI, where $h > 0$ is a user-specified "proposal variance" (called `S.scale`; see `mcmc.control`).

When `phi.prior` is not `"fixed"`, we update the parameter ϕ by a random walk Metropolis step. Here `mcmc.input$phi.scale` (see `mcmc.control`) is the proposal variance, which needs to be sufficiently large so that the algorithm easily can move between the discrete values in `prior$phi.discrete` (see `prior.glm.control`).

CONTROL FUNCTIONS

The function call includes auxiliary control functions which allows the user to specify and/or change the specification of 1) model components (using `model.glm.control`), 2) prior distributions (using `prior.glm.control`), 3) options for the MCMC algorithm (using `mcmc.control`), and 4) options for the output (using `output.glm.control`). Default values are available in most of the cases. The arguments for the control functions are described in their respective help files.

In the prediction part of the function we want to predict $\exp(S^*)/(1+\exp(S^*))$ at locations of interest. For the prediction part of the algorithm, we use the median of the predictive distribution as the predictor and 1/4 of the length of the 95 percent predictive interval as a measure of the prediction uncertainty. Below we describe the procedure for calculating these quantities.

First we perform a Bayesian Gaussian prediction with the given priors on β and σ^2 on each of the simulated S-"datasets" from the posterior distribution (and in case ϕ is not fixed, for each sampled ϕ value). This Gaussian prediction is done by an internal function which is an extension of `krige.bayes` allowing for more than one "data set".

For calculating the probability below a threshold for the predictive distribution given the data, we first calculate this probability for each of the simulated S-"datasets". This is done using the fact that the predictive distribution for each of the simulated S-"datasets" is a multivariate t-distribution. Afterwards the probability below a threshold is calculated by taking the empirical mean of these conditional probabilities.

Now the median and the 2.5 percent and 97.5 percent quantiles can be calculated by an iterative procedure, where first a guess of the value is made, and second this guess is checked by calculating the probability of being less than this value. In case the guess is too low, it is adjusted upwards, and vise verse.

Value

A list with the following components:

`posterior` A list with results for the posterior distribution of the model parameters and the random effects at the data locations. The components are:

- betasummary of posterior distribution for the parameter β.
- sigmasqsummary of the posterior distribution for the parameter σ^2.
- phisummary of the posterior distribution of the parameter ϕ.
- simulationssample from the posterior distribution of $\exp(S)/(1+\exp(S))$ at the data locations. Returned only if `keep.mcmc.sim = TRUE`.
- acc.rateThe acceptance rates.

`predictive` A list with results for the predictive distribution at the prediction locations (if provided). The components are:

- simulationsa numerical matrix. Each column contains a simulation from the predictive distribution. Returned only if `sim.predict = TRUE`.
- mediana vector with the estimated median at the prediction locations.
- uncertaintya vector with the estimated uncertainty at the prediction locations, defined as the length of the 95% prediction interval divided by 4.
- quantileA matrix or vector with quantile estimators.
- probabilityA matrix or vector with probabilities below a threshold. Returned only if the argument `threshold` is used.

`model`	model components used as defined by `model.glm.control`.
`prior`	priors used for the model parameters.
`mcmc.input`	input parameters used for the MCMC algorithm.
`.Random.seed`	system random seed before running the function. Allows reproduction of results. If the `.Random.seed` is set to this value and the function is run again, it will produce exactly the same results.
`call`	the function call.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

References

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

See Also

`binom.krige` for prediction with fixed parameters in the binomial-logit normal model, `pois.krige.bayes` for Bayesian prediction in the Poisson normal model, `krige.bayes` for Bayesian prediction in the Gaussian spatial model.

Examples

```
data(b50)

if(!exists(".Random.seed", envir=.GlobalEnv, inherits = FALSE)) set.seed(1234)
## Not run:
mcmc.10 <- mcmc.control(S.scale = 0.09, n.iter = 1000, phi.scale = 0.2,
             phi.start = 4.5)
prior.10 <- prior.glm.control(phi.discrete = seq(0.2,5,0.2))
test.10 <- binom.krige.bayes(b50, locations=t(cbind(c(2.5,3.5),c(-1,3.5),c(2.5,1.5),c(-1,1.5))),
             prior = prior.10, mcmc.input = mcmc.10)
image(test.10)
persp(test.10)
```

```
## End(Not run)
```

---

covariog *Empirical Covariogram for a Model with log-link and an Underlying Gaussian Field*

Description

Computes the sample empirical (sample) covariogram described in Christensen, Moller and Waagepetersen (2000). Output is returned as a `binned covariogram`. The function is NOT a general function for computing the covariogram, and it is in fact of very limited use.

Usage

```
covariog(geodata, coords = geodata$coords, data = geodata$data,
         units.m = "default", uvec = "default", bins.lim = "default",
         estimator.type = c("poisson", "not-poisson"),
         max.dist = NULL, pairs.min = 2)
```

Arguments

`geodata`	a list containing elements `data` and `coords` as described next. Typically an object of the class `"geodata"` - a **geoR** data set. If not provided the arguments `data` and `coords` must be provided instead. The list may also contain an argument `units.m` as described below.
`coords`	an $n \times 2$ matrix containing coordinates of the n data locations in each row. Default is `geodata$coords`, if provided.
`data`	a vector or matrix with data values. If a matrix is provided, each column is regarded as one variable or realization. Default is `geodata$data`, if provided.
`units.m`	n-dimensional vector of observation times for the data. By default (`units.m = "default"`), it takes `geodata$units.m` in case this exist and else a vector of 1's.
`uvec`	a vector with values defining the covariogram binning. The values of `uvec` defines the midpoints of the bins. If $uvec[1] > 0$ the first bin is: $0 < u <= uvec[2] - 0.5 * (uvec[2] - uvec[1])$. If $uvec[1] = 0$ first bin is: $0 < u <= 0.5 * uvec[2]$, and $uvec[1]$ is replaced by the midpoint of this interval. The default (`uvec = "default"`) is that $uvec[i] = max.dist * (i-1)/14$ for $i = 1, \ldots, 15$.
`bins.lim`	separating values for the binning. By default these values are defined via the argument of `uvec`.
`estimator.type`	`"poisson"` estimates the value $\hat{C}(0)$ using the Poisson assumption. `"not-poisson"` doesn't compute $\hat{C}(0)$.
`max.dist`	a number defining the maximal distance for the covariogram. Pairs of locations separated by a larger distance than this value are ignored in the covariogram calculation. Default is the maximum distance between pairs of data locations.

`pairs.min` An integer number defining the minimum number of pairs for the bins. Bins with number of pairs smaller than this value are ignored.

Details

Covariograms can be used in geostatistical analysis for exploratory purposes, to estimate covariance parameters and/or to compare theoretical and fitted models against the empirical covariogram.

The covariogram computed by this function assumes a specific model, a spatial GLMM, and furthermore it assumes that the link-function is the logarithm (i.e. it should not be used for the binomial-logistic model !).

Assume that the conditional distribution of Y_i given S_i has mean $t_i \exp(S_i)$, where the values of t_i are given in `units.m`. The estimator implemented is

$$\hat{C}(u) = \log\left(\frac{\frac{1}{|W_u^{\Delta}|}\sum_{(i,j)\in W_u^{\Delta}} Y(x_i)Y(x_j)/(t_i t_j)}{\left(\frac{1}{n}\sum_{i=1}^{n} Y(x_i)/t_i\right)^2}\right), \quad u > 0$$

When a Poisson distribution is assumed, then

$$\hat{C}(0) = \log\left(\frac{\frac{1}{n}\sum_{i=1}^{n} Y(x_i)(Y(x_i)-1)/t_i^2}{\left(\frac{1}{n}\sum_{i=1}^{n} Y(x_i)/t_i\right)^2}\right)$$

Value

An object of the class `covariogram` which is a list with the following components:

`u` a vector with distances.

`v` a vector with estimated covariogram values at distances given in `u`. When `estimator.type = "poisson"`, the first value in `v` is the estimate of σ^2, $\hat{C}(0)$.

`n` number of pairs in each bin. When `estimator.type = "poisson"`, the first value in `n` is `v0`.

`v0` the estimate of σ^2, $\hat{C}(0)$.

`bins.lim` Separating values for the binning provided in the function call.

`estimator.type` echoes the type of estimator used.

`call` The function call.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

References

Christensen, O. F., Moller, J. and Waagepetersen R. (2000). Analysis of spatial data using generalized linear mixed models and Langevin-type Markov chain Monte Carlo. *Research report R-00-2009*, Aalborg University.

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

See Also

covariog.model.env for covariogram envelopes and plot.covariogram for graphical output.

Examples

```
data(p50)
covar <- covariog(p50, uvec=c(1:10))
plot(covar)
## Now excluding the bin at zero (only assuming log-link).
covar2 <- covariog(p50,uvec=c(1:10), estimator.type="no")
plot(covar2)
```

---

covariog.model.env *Envelope for Empirical Covariogram for the Poisson-log normal model*

Description

Computes envelope for empirical covariogram by simulating data for given model parameters. This function is for the Poisson-log normal model.

Usage

```
covariog.model.env(geodata, coords = geodata$coords, units.m = "default",
     obj.covariog, model.pars, nsim = 500, prob = c(0.025, 0.975),
     messages)
```

Arguments

geodata	a list containing element coords as described next. Typically an object of the class "geodata" - a **geoR** data set. If not provided the argument coords must be given instead. The list may also contain an argument units.m as described below.
coords	an $n \times 2$ matrix, each row containing Euclidean coordinates of the n data locations. By default it takes the element coords of the argument geodata.
units.m	n-dimensional vector of observation times for the data. By default (units.m = "default"), it takes geodata$units.m in case this exist and else the value 1 for every observation.
obj.covariog	an object of the class "covariogram", typically an output of the function covariog.
model.pars	a list with model specification and parameter values. The required components of the list are: • beta, the mean parameter. Default is beta=0. • cov.model, the covariance model. Default is "exponential". • cov.pars, the covariance parameters σ^2 and ϕ.

- kappa, the extra covariance parameters for some of the covariance models. Default is kappa = 0.5.
- nugget, the relative nugget variance. Default is nugget = 0.

nsim	number of simulations used to compute the envelope. Default is nsim = 500.
prob	the quantiles used for constructing the envelopes. Default is 2.5% and 97.5%
messages	logical. If TRUE, the default, status messages are printed while the function is running.

Details

The envelope is computed assuming a Poisson-log normal model. Simulated values are generated at the data locations, given the model parameters. The empirical covariogram is computed for each simulation using the same binning as for the original covariogram of the data. The envelope is computed by taking, at each lag, the quantile-values of the covariograms for the simulated data.

Value

An object of the class "covariogram.envelope" which is a list with the components:

u	a vector with distances.
v.lower	a vector with the upper-quantile covariogram values for each distance in u.
v.upper	a vector with the lower-quantile covariogram values for each distance in u.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

References

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

See Also

covariog for covariogram calculation and plot.covariogram for graphical output.

Examples

```
data(p50)
covar <- covariog(p50, uvec = c(1:10))
parmval <- list(cov.model = "exponential", cov.pars = c(0.8,0.1),
                beta = 1)
class(parmval) <- "covariomodel"
konvol <- covariog.model.env(p50, obj.covariog = covar,
                             model.pars = parmval)
plot(covar, envelope.obj = konvol)
lines(parmval, max.dist = 10, lty = 1)
```

`create.mcmc.coda` *Create an mcmc object for the CODA package*

Description

This function creates an mcmc object for the CODA package for output from the functions `glsm.mcmc`, `binom.krige.bayes` and `pois.krige.bayes`. The functions in CODA can then be used to investigate convergence and mixing of the MCMC algorithm.

Usage

```
create.mcmc.coda(x, mcmc.input)
```

Arguments

`x`	an output object from `glsm.mcmc`, `binom.krige.bayes` or `pois.krige.bayes`.
`mcmc.input`	input parameters for the MCMC algorithm. It can take an output from `mcmc.control` or a list with elements. Only `thin` and `burn.in` are used, and both have default values (`thin=10`, `burn.in=0`).

Value

An object of class `mcmc` to be used for CODA.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

References

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

Examples

```
## see example in help file for glsm.mcmc
```

geoRglm-defunct *Defunct Functions in the Package geoRglm*

Description

The functions listed here are no longer part of the package geoRglm as they are not needed (any more).

Usage

```
geoRglmdefunct()
```

Details

The following functions are now defunct:

1. pois.log.krigerenamed to pois.krige. From geoRglm_0.6-0.
2. y50renamed to p50. From geoRglm_0.6-2.

glsm.krige *Prediction for a Generalised Linear Spatial Model*

Description

This function makes prediction for a generalised linear spatial model, using an output object from glsm.mcmc

Usage

```
glsm.krige(mcmc.output, locations, borders, trend.l,
micro.scale=NULL, dist.epsilon= 1e-10, output)
```

Arguments

mcmc.output	an output file from the function glsm.mcmc.
locations	an $N \times 2$ matrix or data frame, or a list with the 2-D coordinates of the N prediction locations.
borders	optional. If a two column matrix defining a polygon is provided the prediction is performed only at locations inside this polygon.
trend.l	specifies the trend (covariate) values at prediction locations. It must be of the same type as for trend.
micro.scale	micro-scale variance. If specified, the nugget is divided into 2 terms: *micro-scale variance* and *measurement error*. This has effect on prediction, since the the target for prediction is inverse link function of the "signal" part of S (without the measurement error part of the nugget). The default is micro.scale = nugget.

`dist.epsilon` a numeric value. Locations which are separated by a distance less than this value are considered co-located.

`output` parameters for controlling the output. It can take an output from `output.glm.control` or a list with elements as for the arguments in `output.glm.control`. See documentation for `output.glm.control`.

Details

This function makes prediction for fixed parameters using an output object from `glsm.mcmc` containing the model specification and simulations from the posterior values of S.

The prediction consist of performing trans-Gaussian kriging on each of the simulated $g^{-1}(S)$-"datasets" from the conditional distribution. Afterwards the predictor is obtained by taking the mean of prediction means, and the prediction variance is obtained by taking the mean of the prediction variances plus the variance of the prediction means. The trans-Gaussian kriging is done by calling an internal function which is an extension of `krige.conv` allowing for more than one "data set", and using a second order Taylor approximation of the inverse link function g^{-1}.

Value

A list with the following components:

`predict` a vector with predicted values.

`krige.var` a vector with predicted variances.

`mcmc.error` estimated Monte Carlo errors on the predicted values.

`simulations` an $ni \times n.sim$ matrix where ni is the number of prediction locations and $n.sim$ is the number of MCMC simulations. Each column corresponds to a conditional simulation of the predictive distribution $g^{-1}(S^*)$. Only returned if `sim.predict = TRUE`.

`message` messages about the type of prediction performed.

`call` the function call.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

References

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

See Also

`glsm.mcmc` for MCMC simulation in a generalised linear spatial model.

Examples

```
if(!exists(".Random.seed", envir=.GlobalEnv, inherits = FALSE)) set.seed(1234)
data(b50)
mcmc.5 <- mcmc.control(S.scale = 0.6, thin=1)
model.5 <- list(cov.pars=c(0.6, 0.1), beta=1, family="binomial")
outmcmc.5 <- glsm.mcmc(b50, model= model.5, mcmc.input = mcmc.5)
test2 <- glsm.krige(outmcmc.5, locations=matrix(c(0.15,0.15,0.005,0.05),2,2))
image(test2)
test3 <- glsm.krige(outmcmc.5, locations=matrix(c(0.15,0.15,0.005,0.05),2,2),
                    output=output.glm.control(sim.predict=TRUE, quantile=FALSE))
```

`glsm.mcmc` *Conditional Simulation for a generalised linear spatial model*

Description

This function performs conditional simulation (by MCMC) in a generalised linear spatial model for fixed parameters.

Usage

```
glsm.mcmc(geodata, coords = geodata$coords, data = geodata$data,
         units.m = "default",  model, mcmc.input, messages)
```

Arguments

`geodata`	a list containing elements `coords` and `data` as described next. Typically an object of the class `"geodata"` - a **geoR** data set. If not provided the arguments `coords` and `data` must be provided instead. The list may also contain an argument `units.m` as described below.
`coords`	an $n \times 2$ matrix, each row containing Euclidean coordinates of the n data locations. By default it takes the element `coords` of the argument `geodata`.
`data`	a vector with data values. By default it takes the element `data` of the argument `geodata`.
`units.m`	n-dimensional vector of observation times for the data. By default (`units.m = "default"`), it takes `geodata$units.m` in case this exist and else a vector of 1's.
`model`	defines the model components. Either an object of class `likGLSM`; typically output from `likfit.glsm`, or a list containing the arguments :

- trendspecifies the trend (covariate) values at the data locations. See documentation of `trend.spatial` for further details. Default is `trend = "cte"`.
- betanumerical value of the mean (vector) parameter.
- cov.modelstring indicating the name of the model for the correlation function. Further details in the documentation for `cov.spatial`.

- cov.parsa vector with the 2 covariance parameters σ^2, and ϕ for the underlying Gaussian field.
- kappaadditional smoothness parameter required by the following correlation functions: `"matern"`, `"powered.exponential"`, `"cauchy"` and `"gneiting.matern"`.
- nuggetthe value of the nugget parameter τ^2 for the underlying Gaussian field. Default is `nugget = 0`.
- aniso.parsparameters for geometric anisotropy correction. If `aniso.pars = FALSE` no correction is made, otherwise a two elements vector with values for the anisotropy parameters must be provided. Anisotropy correction consists of a transformation of the data and prediction coordinates performed by the function `coords.aniso`.
- familyequal to either `"poisson"` or `"binomial"`
- linkequal to either `"canonical"` (default), `"log"`, `"boxcox"` or `"logit"`. For `"canonical"` then in general the canonical link function is used (`"log"` for the Poisson distribution and `"logit"` for the binomial distribution), but when `lambda` is also specified then the Box-Cox class is used (a mis-use of the terminology "canonical", really).
- lambdanumeric value of the Box-Cox transformation parameter. The value $\lambda = 1$ corresponds to no transformation and the default value $\lambda = 0$ corresponds to the log-transformation. Only used when `family = "poisson"`

`mcmc.input` input parameter for the MCMC algorithm. It can take an output from `mcmc.control` or a list with elements as for the arguments in `mcmc.control`. See documentation for `mcmc.control`.
ATTENTION: the argument `S.scale` is necessary while all the others have default values.

`messages` logical. Indicates whether or not status messages are printed on the screen (or other output device) while the function is running.

Details

For simulating the conditional distribution of S given y, the Langevin-Hastings algorithm with the parametrisation in Papaspilliopoulus, Roberts and Skold (2003) is used. This algorithm is a Metropolis-Hastings algorithm, where the proposal distribution uses gradient information from the log-posterior distribution.

The proposal variance (called `S.scale`; see `mcmc.control`) for the algorithm needs to be scaled such that approximately 60 percent of the proposals are accepted. We also recommend that the user to studies plots of the autocorrelations.

Value

A list with the following components:

`simulations` an $n \times n.sim$ matrix with $n.sim$ being the number of MCMC simulations. containing S_i. Each column corresponds to a conditional simulation of the conditional distribution of S_i at the data locations.

`acc.rate` matrix with acceptance rates from MCMC. Only returned when no prediction locations are given.

model	Information about the model parameters, link function and error distribution family used.
geodata	Information about the data.
call	the function call.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

References

O. Papaspiliopoulus and G. O. Roberts and M. Skold (2003). Non-centered parameterizations for hierarchical models and data augmentation. *Bayesian statistics 7* (eds. J. M. Bernardo, S. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith and M. West), Oxford University Press, 307-326.

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

See Also

binom.krige for prediction with fixed parameters in the Binomial-normal model, pois.krige for prediction with fixed parameters in the Poisson normal model.

Examples

```
if(!exists(".Random.seed", envir=.GlobalEnv, inherits = FALSE)) set.seed(1234)
data(b50)
test <- glsm.mcmc(b50, model = list(family="binomial",
             cov.pars = c(1,1), beta = c(1,0), trend =~ rnorm(50),
             cov.model="spherical", nugget=0.3),
          mcmc.input = mcmc.control(S.scale = 0.2, thin = 1))
## visulalising the MCMC output using the coda package
test.coda <- create.mcmc.coda(test, mcmc.input = list(thin = 1))
## Not run:
plot(test.coda)
autocorr.plot(test.coda)

## End(Not run)
```

`hist.glm.krige.bayes` *Plots Sample from Posterior Distributions*

Description

Plots histograms and/or density estimation with samples from the posterior distribution of the model parameters for output from the functions `binom.krige.bayes` and `pois.krige.bayes`

Usage

```
## S3 method for class 'glm.krige.bayes'
hist(x, pars, density.est = TRUE, histogram = TRUE, ...)
```

Arguments

`x`	an object of the class `glm.krige.bayes`, with an output from the funtions `binom.krige.bayes` or `pois.krige.bayes`.
`pars`	a vector with the names of one or more of the model parameters. Defaults to all model parameters.
`density.est`	logical indication whether a line with the density estimation should be added to the plot.
`histogram`	logical indicating whether the histogram is included in the plot.
`...`	further arguments for the plotting functions and or for the density estimation.

Value

Produces a plot in the currently graphics device.
Returns an `invisible` list with the components:

`histogram`	with the output of the function `hist` for each parameter
`density.estimation`	with the output of the function `density` for each parameter

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

See Also

`binom.krige.bayes`, `pois.krige.bayes`, `hist.krige.bayes`.

Examples

```
## See documentation for binom.krige.bayes and pois.krige.bayes
```

image.glm.krige.bayes *Plots Results of the Predictive Distribution*

Description

This function produces an image or perspective plot of a selected element of the predictive distribution returned by the functions binom.krige.bayes and pois.krige.bayes.

Usage

```
## S3 method for class 'glm.krige.bayes'
image(x, locations, borders,
                  values.to.plot=c("median", "uncertainty",
                     "quantiles", "probabilities", "simulation"),
                  number.col, coords.data, x.leg, y.leg, messages,...)


## S3 method for class 'glm.krige.bayes'
persp(x, locations, borders,
                  values.to.plot=c("median", "uncertainty",
                     "quantiles", "probabilities", "simulation"),
                  number.col, messages, ...)
```

Arguments

x	an object of the class glm.krige.bayes, typically an output of the functions binom.krige.bayes or pois.krige.bayes.
locations	an $n \times 2$ matrix with the coordinates of the prediction locations, which should define a regular grid in order to be plotted by image or persp. By default does not need to be provided and evaluates the attribute "prediction.locations" from the input object.
borders	an $n \times 2$ matrix with the coordinates defining the borders of a region inside the grid defined by locations. Elements in the argument values are assigned to locations internal to the borders and NA's to the external ones.
values.to.plot	select the element of the predictive distribution to be plotted. See DETAILS below.
number.col	Specifies the number of the column to be plotted. Only used if previous argument is set to one of "quantiles", "probabilities" or "simulation".
coords.data	optional. If an $n \times 2$ matrix with the data coordinates is provided, points indicating the data locations are included in the plot.
x.leg, y.leg	limits for the legend in the horizontal and vertical directions.
messages	logical, if TRUE status messages are printed while running the function.
...	extra arguments to be passed to the plotting function image or persp.

Details

The functions binom.krige.bayes and pois.krige.bayes return summaries and other results about the predictive distributions. The argument values.to.plot specifies which result will be plotted. It can be passed to the function in two different forms:

- a vector with the object containing the values to be plotted, or
- one of the following options: "median", "uncertainty", "quantiles", "probability" or "simulation".

For the last three options, if the results are stored in matrices, a column number must be provided using the argument number.col.

The documentation for the functions binom.krige.bayes and pois.krige.bayes provide further details about these options.

Value

An image or persp plot is produced on the current graphics device. No values are returned.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

See Also

image.krige.bayes for plotting output from krige.bayes

Examples

```
#See examples in the documentation for the functions binom.krige.bayes and pois.krige.bayes .
```

---

krige.glm.control *Defines options and model for prediction*

Description

This auxiliary function defines options and model for pois.krige and binom.krige.

Usage

```
krige.glm.control(type.krige = "sk", trend.d = "cte", trend.l = "cte",
                  obj.model = NULL, beta, cov.model, cov.pars, kappa,
                  nugget, micro.scale, dist.epsilon = 1e-10,
                  aniso.pars, lambda)
```

Arguments

type.krige	type of prediction to be performed (minimal mean square error prediction). Options are "sk" and "ok" corresponding to prediction with fixed parameters (type.krige = "sk"), which is the default, or prediction with a uniform prior on β (type.krige = "ok"). Prediction using a model with covariates can be done by specifying the covariate model using the arguments trend.d and trend.l.
trend.d	specifies the trend (covariate) values at the data locations. See documentation of trend.spatial for further details. Default is trend.d = "cte".
trend.l	specifies the trend (covariate) values at prediction locations. It must be of the same type as for trend.d. Only used if prediction locations are provided in the argument locations.
obj.model	a list with the model parameters.
beta	numerical value of the mean (vector) parameter. Only used if type.krige="sk".
cov.model	string indicating the name of the model for the correlation function. Further details in the documentation for cov.spatial.
cov.pars	a vector with the 2 covariance parameters σ^2, and ϕ for the underlying Gaussian field.
kappa	additional smoothness parameter required by the following correlation functions: "matern", "powered.exponential", "cauchy" and "gneiting.matern".
nugget	the value of the nugget parameter τ^2 for the underlying Gaussian field. Default is nugget = 0.
micro.scale	micro-scale variance. If specified, the nugget is divided into 2 terms: *micro-scale variance* and *measurement error*. This has effect on prediction where the "signal" part of S (without the measurement error part of the nugget) is predicted. The default is micro.scale = nugget.
dist.epsilon	a numeric value. Locations which are separated by a distance less than this value are considered co-located.
aniso.pars	parameters for geometric anisotropy correction. If aniso.pars = FALSE no correction is made, otherwise a two elements vector with values for the anisotropy parameters must be provided. Anisotropy correction consists of a transformation of the data and prediction coordinates performed by the function coords.aniso.
lambda	numeric value of the Box-Cox transformation parameter for pois.krige. The value $\lambda = 1$ corresponds to no transformation and $\lambda = 0$ corresponds to the log-transformation. Prediction results are back-transformed and returned is the same scale as for the original data.

Value

A list with processed arguments to be passed to the main function.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

See Also

pois.krige and binom.krige.

---

likfit.glsm *Monte Carlo Maximum Likelihood Estimation in a Generalised Linear Spatial Model*

Description

This function performs Monte Carlo maximum likelihood in a generalised linear spatial model, based on a Monte Carlo sample from the conditional distribution.

Usage

```
likfit.glsm(mcmc.obj, trend = mcmc.obj$trend, cov.model = "matern",
       kappa = 0.5, ini.phi, fix.nugget.rel = FALSE, nugget.rel = 0,
       aniso.pars = NULL, fix.lambda = TRUE, lambda = NULL,
       limits = pars.limits(), messages, ...)
```

Arguments

mcmc.obj	object with the Monte Carlo simulations and corresponding approximating density. This object should be an output from the function prepare.likfit.glsm.
trend	specifies the covariate values at the data locations. See documentation of trend.spatial for further details. Default is that the trend is the same as in the mcmc.obj object.
cov.model	a string specifying the model for the correlation function. For further details see documentation for cov.spatial.
kappa	additional smoothness parameter required by the following correlation functions: "matern", "powered.exponential", "gneiting.matern" and "cauchy".
ini.phi	initial value for the covariance parameter ϕ.
fix.nugget.rel	logical, saying whether the parameter τ^2_R (relative nugget) should be regarded as fixed (fix.nugget.rel = TRUE) or should be estimated (fix.nugget.rel = FALSE). Default is fix.nugget.rel = FALSE.
nugget.rel	value of the relative nugget parameter. Regarded as a fixed value if fix.nugget.rel = TRUE, otherwise as the initial value for the maximization algorithm. Default is nugget.rel = 0.
aniso.pars	parameters for geometric anisotropy correction. If aniso.pars = NULL the correction will be the same as for the generated sample in mcmc.obj. Otherwise a two elements vector with values for the anisotropy parameters must be provided. Anisotropy correction consists of a transformation of the data and prediction coordinates performed by the function coords.aniso.
fix.lambda	logical, indicating whether the Box-Cox transformation parameter λ should be regarded as fixed (fix.lambda = TRUE) or should be be estimated (fix.lambda = FALSE). Default is fix.lambda = TRUE.

lambda	value of parameter λ in the Box-Cox class of link functions. Regarded as a fixed value if fix.lambda = TRUE, otherwise as the initial value for the minimization algorithm. Default is lambda = NULL, in which case the used link function will be the same as for the generated sample in mcmc.obj.
limits	values defining lower and upper limits for the model parameters used in the numerical minimization. The auxiliary function pars.limits is used to set the limits.
messages	logical. Indicates whether status messages should be printed on the screen (or output device) while the function is running.
...	additional parameters to be passed to the optimisation function. Typically arguments of the type control() which controls the behavior of the optimization algorithm. For further details, see the documentation for the minimization function optim.

Details

This function estimates the parameters in the Poisson/Binomial normal model, using a Monte Carlo approximation to the likelihood. Further details can be found in Christensen (2004).

Parameter estimation is done numerically using the R function optim with box-constraints, i.e. method="L-BFGS-B".

Lower and upper limits for parameter values can be specified using the function pars.limits(). For example, including limits = pars.limits(phi=c(0.01, 10)) in the function call will specify the limits for the parameter ϕ. Default values are used if the argument limits is not provided.

Only when the mcmc.obj object contains an object mu giving the intensity, is it possible to use other link functions than the link function used for the generated sample in mcmc.obj

We strongly recommend that the user does not provide self-made input objects for mcmc.obj, but only uses objects created by prepare.likfit.glsm. In case the user really wants to create his own objects, he should study the source code very carefully to understand how it works.

Summary and print methods for summarising and printing the output also exist.

Value

A list with the following components:

family	the error distribution (Poisson or Binomial).
link	the name of the link function.
cov.model	a string with the name of the correlation function.
beta	estimate of the parameter β. This can be a scalar or vector depending on the covariates (trend) specified in the model.
cov.pars	a vector with the estimates of the parameters σ^2 and ϕ, respectively.
nugget.rel	value of the relative nugget parameter τ^2_R. This is an estimate if fix.nugget.rel = FALSE, and otherwise a given fixed value.
kappa	value of the smoothness parameter. Valid only when the correlation function is one of: "matern", "powered.exponential", "cauchy" or "gneiting.matern".

`lambda`	values of the parameter for the Box-Cox class of link functions. A fixed value if `fix.lambda = TRUE`, otherwise the estimated value.
`aniso.pars`	values of the anisotropy parameters used.
`trend`	the trend
`parameters.summary`	a data-frame with all model parameters, their status (estimated or fixed) and values.
`loglik`	the value of the maximized likelihood.
`npars`	number of estimated parameters.
`info.minimisation`	results returned by the minimisation function.
`call`	the function call.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

References

Christensen, O. F. (2004). Monte Carlo maximum likelihood in model-based geostatistics. Journal of computational and graphical statistics **13** 702-718.

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

See Also

`prepare.likfit.glsm` on how to prepare the object `mcmc.obj`, `glsm.mcmc` for MCMC simulation in generalised linear spatial model, and `summary.likGLSM` for summarising the output. See also `likfit` for parameter estimation in the Gaussian spatial model.

Examples

```
data(p50)

## Not run:
mcmc.5 <- mcmc.control(S.scale = 0.6, thin=20, n.iter=50000, burn.in=1000)
model.5 <- list(cov.pars=c(0.6, 0.1), beta=1, family="poisson")
outmcmc.5 <- glsm.mcmc(p50, model= model.5, mcmc.input = mcmc.5)
mcmcobj.5 <- prepare.likfit.glsm(outmcmc.5)
lik.5 <- likfit.glsm(mcmcobj.5, ini.phi = 0.1, fix.nugget.rel = TRUE)
print(lik.5)
summary(lik.5)
lik.5.sph.nugget <- likfit.glsm(mcmcobj.5, ini.phi = 1,
                          cov.model = "spherical", nugget.rel = 0.385)
print(lik.5.sph.nugget)
summary(lik.5.sph.nugget)

## End(Not run)
```

`lines.covariomodel` *Line with a Covariogram Model*

Description

This function adds a theoretical covariogram to the current plot. The covariogram model to be added is typically with parameters estimated from the data.

Usage

```
## S3 method for class 'covariomodel'
lines(x, max.dist = x$max.dist, ...)
```

Arguments

`x`	an object of the class `covariomodel` which is a list containing information about the model parameters.
`max.dist`	maximum distance (x-axis) to compute and draw the line representing the covariogram model. The default is the distance given by `x$max.dist`.
`...`	arguments to be passed to the function `lines`.

Details

Allows theoretical covariogram(s) to be added to a plot. Together with `plot.covariogram` can be used to compare sample covariograms against fitted models.

Value

A line with a covariogram model is added to a plot on the current graphics device. No values are returned.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

References

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

See Also

`plot.covariogram`, `lines`.

Examples

```
sim <- grf(100, cov.pars = c(0.1, 0.2))
sim$data <- rpois(100, lambda = exp(sim$data+1))
# data generated from the poisson-log normal model
covario <- covariog(sim, max.dist = 1)  # sample covariogram
plot(covario)
parmval <- list(cov.model = "exponential", cov.pars = c(0.1, 0.1),
                max.dist = 1, nugget = 0.01)
class(parmval) <- "covariomodel"
lines(parmval, lty = 2)
```

`mcmc.control` *Defines options for the MCMC-algorithm*

Description

This auxiliary function defines options for the MCMC-algorithm used by `pois.krige.bayes`, `binom.krige.bayes`, `binom.krige` and `pois.krige`.

Usage

```
mcmc.control(S.scale, Htrunc, S.start, burn.in, thin, n.iter, phi.start,
             phi.scale)
```

Arguments

`S.scale`	proposal variance for the update of S in the algorithm.
`Htrunc`	truncation parameter for the MCMC-algorithm. Only used for the Poisson model. Either a number or a vector of the same dimension as the data. Default is Htrunc = 2*data + 5.
`S.start`	starting value for `S` (without the mean !!) in the MCMC-algorithm. Default value exist. Should either be a vector of the same length as the data, `S.start="random"`, or `S.start="default"`
`burn.in`	length of the burn-in period. Default is 0.
`thin`	thinning: only every `thin` scan is stored. Default is `thin=10`.
`n.iter`	number of iterations performed. Default is `n.iter=1000*thin`.
`phi.start`	starting value for ϕ. Default is the median of `prior$phi.discrete`.
`phi.scale`	proposal variance for the update of ϕ in the algorithm.

Value

A list with processed arguments to be passed to the main function.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

See Also

pois.krige.bayes, binom.krige.bayes, binom.krige and pois.krige.

---

model.glm.control *Defines model options*

Description

This auxiliary function defines model options for pois.krige.bayes and binom.krige.bayes.

Usage

```
model.glm.control(trend.d = "cte", trend.l = "cte", cov.model = "matern",
                  kappa = 0.5, aniso.pars = NULL, lambda = 0)
```

Arguments

trend.d	specifies the covariate values at the data locations. See documentation of trend.spatial for further details. Default is trend.d="cte".
trend.l	specifies the covariate values at prediction locations. It must be of the same type as for trend.d. Only used if prediction locations are provided in the argument locations.
cov.model	string indicating the name of the model for the correlation function. Further details in the documentation for cov.spatial.
kappa	additional smoothness parameter required by the following correlation functions: "matern", "powered.exponential", "gneiting.matern" and "cauchy".
aniso.pars	parameters for geometric anisotropy correction. If aniso.pars = NULL no correction is made, otherwise a two elements vector with values for the anisotropy parameters must be provided. Anisotropy correction consists of a transformation of the data and prediction coordinates performed by the function coords.aniso.
lambda	parameter controlling the link function for pois.krige.bayes. lambda = 0 is the default, which corresponds to a log link. Other values gives a link function from the Box-Cox class.

Value

A list with processed arguments to be passed to the main function.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

See Also

pois.krige.bayes and binom.krige.bayes.

---

output.glm.control *Defines output options*

Description

This auxiliary function defines output options for pois.krige.bayes and binom.krige.bayes.

Usage

```
output.glm.control(sim.posterior, sim.predict, keep.mcmc.sim, quantile,
                   threshold, inference, messages)
```

Arguments

sim.posterior	logical. Indicates whether or not the MCMC-sample from the posterior distribution of the parameters should be returned. Default is sim.posterior = TRUE.
sim.predict	logical. Defines whether simulations are drawn from the predictive distribution. Only valid if prediction locations are provided in the argument locations. Default is sim.predict = FALSE.
keep.mcmc.sim	logical. Indicates whether or not the MCMC-sample from the posterior distribution of the signal should be returned. Here the signal is the inverse link function of the random effect $g^{-1}(S)$, which equals $\exp(S)$ for the Poison model with log-link, $(\lambda S + 1)^{\lambda}$ for the Poison model with Box-Cox-link and $\exp(S)/(1 + \exp(S))$ for the binomial model with logit-link. Default is keep.mcmc.sim = TRUE.
quantile	indicates whether quantiles of the simulations from the predictive distributions are computed and returned. If a vector with numbers in the interval $[0, 1]$ is provided, the output includes the object quantile, which contains values of corresponding estimated quantiles. For example, if quantile = c(0.25,0.50,0.75) the function returns the quartiles of the distributions at each of the prediction locations. The default is quantile = TRUE where the values c(0.025, 0.5, 0.975) are used. If quantile = FALSE no quantiles are computed (and hence neither median nor uncertainty is included in the output). Only used if prediction locations are provided in the argument locations.
threshold	one or more values with threshold values can be given to this argument. If so, an object called probability is included in the output. This object contains, for each prediction location, the probability that the variable is less than or equal to the threshold value given in the argument. Default is that nothing is computed.
inference	logical. Indicates whether or not inference (summary of the parameters, and prediction) is performed or not. Default is inference=TRUE. Setting inference=FALSE is useful in an initial stage, when tuning the MCMC algorithm (choosing S.scale and phi.scale appropriate).
messages	logical. Indicates whether or not status messages are printed on the screen (or other output device) while the function is running.

Value

A list with processed arguments to be passed to the main function.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

See Also

pois.krige.bayes and binom.krige.bayes.

---

plot.covariogram *Plot Empirical Covariogram*

Description

Plots sample (empirical) covariogram computed using the function covariog.

Usage

```
## S3 method for class 'covariogram'
plot(x, max.dist = max(x$u), ylim = "default", type = "b",
     envelope.obj = NULL, ...)
```

Arguments

x	an object of the class "covariogram", typically an output of the function covariog.
max.dist	maximum distance for the x-axis. The default is the maximum distance for which the sample covariogram was computed.
ylim	limits for the covariogram values in the y-axis. The default is from the minimum to the maximum value in x$v.
type	type of line for the empirical covariogram. The default is "b" (dots and lines). For further details see documentation for lines.
envelope.obj	adds a covariogram envelope computed by the function covariog.model.env.
...	other arguments to be passed to the function plot.

Details

This function allows visualisation of the empirical covariogram. Together with lines.covariomodel it can be used to compare theoretical covariogram models against the empirical covariogram.

Value

Produces a plot with the sample covariogram on the current graphics device. No values are returned.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

See Also

covariog for covariogram calculations, lines.covariomodel for adding lines to the current plot, covariog.model.env for computation of covariogram envelopes, and plot for generic plot function.

Examples

```
data(p50)
covario <- covariog(p50, uvec = c(1:10))  # sample covariogram
plot(covario)
```

---

pois.krige *Conditional Simulation and Prediction for the Poisson Spatial model with a link function from the Box-Cox class*

Description

This function performs conditional simulation (by MCMC) and spatial prediction in the Poisson normal model with link function from the Box-Cox class for fixed covariance parameters. Available types of prediction are: *sk* (simple kriging; fixed beta), *ok* (ordinary kriging; uniform prior on beta).

Usage

```
pois.krige(geodata, coords = geodata$coords, data = geodata$data,
               units.m = "default", locations = NULL, borders,
               mcmc.input, krige, output)
```

Arguments

geodata	a list containing elements coords and data as described next. Typically an object of the class "geodata" - a **geoR** data set. If not provided the arguments coords and data must be provided instead. The list may also contain an argument units.m as described below.
coords	an $n \times 2$ matrix, each row containing Euclidean coordinates of the n data locations. By default it takes the element coords of the argument geodata.
data	a vector with data values. By default it takes the element data of the argument geodata.
units.m	n-dimensional vector of observation times for the data. By default (units.m = "default"), it takes geodata$units.m in case this exist and else a vector of 1's.
locations	an $N \times 2$ matrix or data frame, or a list with the 2-D coordinates of the N prediction locations.

borders	optional. If a two column matrix defining a polygon is provided the prediction is performed only at locations inside this polygon.
mcmc.input	input parameter for the MCMC algorithm. It can take an output from mcmc.control or a list with elements as for the arguments in mcmc.control. See documentation for mcmc.control. ATTENTION: the argument S.scale is necessary while all the others have default values.
krige	defines the model components and the type of kriging. It can take an output from krige.glm.control or a list with elements as for the arguments in krige.glm.control. See documentation for krige.glm.control.
output	parameters for controlling the output. It can take an output from output.glm.control or a list with elements as for the arguments in output.glm.control. See documentation for output.glm.control.

Details

For simulating the conditional distribution of S given y, the Langevin-Hastings algorithm with the parametrisation in Papaspilliopoulus, Roberts and Skold (2003) is used. This algorithm is a Metropolis-Hastings algorithm, where the proposal distribution uses gradient information from the log-posterior distribution.

The proposal variance (called S.scale; see mcmc.control) for the algorithm needs to be scaled such that approximately 60 percent of the proposals are accepted. We also recommend that the user to studies plots of the autocorrelations.

The prediction part of the program consist of performing trans-Gaussian kriging on each of the simulated $g^{-1}(S)$-"datasets" from the conditional distribution. Afterwards the predictor is obtained by taking the mean of prediction means, and the prediction variance is obtained by taking the mean of the prediction variances plus the variance of the prediction means. The trans-Gaussian kriging is done by calling an internal function which is an extension of krige.conv allowing for more than one "data set", and using a second order Taylor approximation of the inverse Box-Cox transformation function g^{-1} when the transformation parameter $lambda > 0$; for the exponential function, i.e. logarithmic link and $lambda = 0$, an exact formula is used instead of the Taylor approximation.

Value

A list with the following components:

predict	a vector with predicted values.
krige.var	a vector with predicted variances.
mcmc.error	estimated Monte Carlo errors on the predicted values.
beta.est	estimate of the β parameter. Not included in the output if type.krige = "sk".
intensity	an $n \times n.sim$ matrix with $n.sim$ being the number of MCMC simulations. containing $g^{-1}(S_i)$. Each column corresponds to a conditional simulation of the conditional distribution of $g^{-1}(S_i)$ at the data locations. Only returned when no prediction locations are given.
acc.rate	matrix with acceptance rates from MCMC. Only returned when no prediction locations are given.

`simulations` an $ni \times n.sim$ matrix where ni is the number of prediction locations and $n.sim$ is the number of MCMC simulations. Each column corresponds to a conditional simulation of the predictive distribution $g^{-1}(S^*)$. Only returned if `sim.predict = TRUE`.

`call` the function call.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

References

O. Papaspiliopoulus and G. O. Roberts and M. Skold (2003). Non-centered parameterizations for hierarchical models and data augmentation. *Bayesian statistics 7* (eds. J. M. Bernardo, S. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith and M. West), Oxford University Press, 307-326.

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

See Also

`pois.krige.bayes` for Bayesian prediction in the Poisson-normal model, `binom.krige` for prediction with fixed parameters in the binomial-logit normal model, `krige.conv` for prediction in the linear Gaussian model.

Examples

```
if(!exists(".Random.seed", envir=.GlobalEnv, inherits = FALSE)) set.seed(1234)
data(p50)
# First we scale the algorithm, and study how well the chain is mixing.
test <- pois.krige(p50, krige = list(cov.pars = c(1,1), beta = 1),
      mcmc.input = mcmc.control(S.scale = 0.2, thin = 1))
plot(log(test$intensity[45,]), type = "l")
acf(log(test$intensity[45,]), type = "correlation", plot = TRUE)
## Not run: # Now we make prediction (we decide to thin to every 10, which is the default),
# where we now use S.scale = 0.55.
test2 <- pois.krige(p50, locations = cbind(c(0.5,0.5), c(1,0.4)),
      krige = krige.glm.control(cov.pars = c(1,1), beta = 1),
      mcmc.input = mcmc.control(S.scale = 0.55))
## End(Not run)
```

`pois.krige.bayes` *Bayesian Posterior Simulation and Prediction for the Poisson Spatial model*

Description

This function performs posterior simulation (by MCMC) and spatial prediction in the Poisson spatial model (with link function from the Box-Cox class).

Usage

```
pois.krige.bayes(geodata, coords = geodata$coords, data = geodata$data,
        units.m = "default", locations = "no", borders,
        model, prior, mcmc.input, output)
```

Arguments

`geodata`	a list containing elements `coords` and `data` as described next. Typically an object of the class `"geodata"` - a **geoR** data set. If not provided the arguments `coords` and `data` must be given instead. The list may also contain an argument `units.m` as described below.
`coords`	an $n \times 2$ matrix, each row containing Euclidean coordinates of the n data locations. By default it takes the element `coords` of the argument `geodata`.
`data`	a vector with data values. By default it takes the element `data` of the argument `geodata`.
`units.m`	n-dimensional vector of observation times for the data. By default (`units.m = "default"`), it takes `geodata$units.m` in case this exist and else a vector of 1's.
`locations`	an $N \times 2$ matrix or data frame, or a list with the 2-D coordinates of the N prediction locations.
`borders`	optional. If a two column matrix defining a polygon is provided the prediction is performed only at locations inside this polygon.
`model`	a list defining the components of the model. It can take an output from `model.glm.control` or a list with elements as for the arguments in `model.glm.control`. See documentation for `model.glm.control`. All arguments have default values
`prior`	specification of priors for the model parameters. It can take an output from `prior.glm.control` or a list with elements as for the arguments in `prior.glm.control`. See documentation for `prior.glm.control`. ATTENTION: When `phi.prior = "fixed"` then `phi` must be specified, and when `phi.prior` is not `"fixed"` then `phi.discrete` must be specified. All other parameters have default values.
`mcmc.input`	input parameter for the MCMC algorithm. It can take an output from `mcmc.control` or a list with elements as for the arguments in `mcmc.control`. See documentation for `mcmc.control`. ATTENTION: the argument `S.scale` must be specified, the argument `phi.start` must specified when `prior$phi` is not `"fixed"`, while all the others have default values.

`output` parameters for controlling the output. It can take an output from `output.glm.control` or a list with elements as for the arguments in `output.glm.control`. See documentation for `output.glm.control`.

Details

`pois.krige.bayes` is a function for Bayesian geostatistical inference in the Poisson spatial model.

It can be used for an analysis with fixed values of the parameters. However, the function `pois.krige` may be preferable in this case.

The Bayesian algorithm is using a discretized version of the prior distribution for the parameter ϕ. This means that the prior for ϕ is always a proper prior.

For simulating from the posterior distribution of S given y, we use a Langevin-Hastings type algorithm. This algorithm is a Metropolis-Hastings algorithm, where the proposal distribution uses gradient information from the posterior. The algorithm is described below. For shortness of presentation, we present only the MCMC algorithm for the case where β follows a uniform prior and the link function is the canonical log-link.

When β follows a uniform prior and the prior for σ^2 is a scaled inverse-χ^2 distribution, the marginalised improper density of S is

$$f_I(s) \propto |D^T V^{-1} D|^{1/2} |V|^{-1/2} \{n_\sigma S_\sigma^2 + s^T (V^{-1} - V^{-1} D (D^T V^{-1} D)^{-1} D^T V^{-1}) s\}^{-(n-p+n_\sigma)/2},$$

where V is the correlation matrix of S. The uniform prior for σ^2 corresponds to $S_\sigma^2 = 0$ and $n_\sigma = -2$, and the reciprocal prior for σ^2 corresponds to $S_\sigma^2 = 0$ and $n_\sigma = 0$.

We use the reparametrisation $S = Q\Gamma$, where Q is the Cholesky factorisation of V so that $V = QQ^T$. Posterior simulations of S are obtained by transforming MCMC simulations from the conditional distribution of Γ given $Y = y$.

The log posterior density of Γ given $Y = y$ is

$$\log f(\gamma|y) = const(y) - \frac{1}{2}\gamma^T (I_n - V^{-1/2} D (D^T V^{-1} D)^{-1} D^T V^{-1/2}) \gamma + \sum_{i=1}^{n} y_i s_i - \exp(s_i),$$

where $(s_1, \ldots, s_n)^T = Q\gamma$.

For the truncated Langevin-Hastings update we use a truncated form of the gradient (truncating by H_i) of the log target density,

$$\nabla(\gamma)^{trunc} = -(I_n - Q^{-1} D (D^T V^{-1} D)^{-1} D^T (Q^{-1})^T)\gamma + Q^T \{y_i - \exp(s_i) \wedge H_i\}_{i=1}^n .$$

The proposal γ' follows a multivariate normal distribution with mean vector $\xi(\gamma) = \gamma + (h/2)\nabla(\gamma)^{trunc}$ and covariance matrix hI, where $h > 0$ is a user-specified "proposal variance" (called `S.scale`; see `mcmc.control`).

When `phi.prior` is not `"fixed"`, we update the parameter ϕ by a random walk Metropolis step. Here `mcmc.input$phi.scale` (see `mcmc.control`) is the proposal variance, which needs to be sufficiently large so that the algorithm easily can move between the discrete values in `prior$phi.discrete` (see `prior.glm.control`).

CONTROL FUNCTIONS

The function call includes auxiliary control functions which allows the user to specify and/or change the specification of 1) model components (using `model.glm.control`), 2) prior distributions (using

prior.glm.control), 3) options for the MCMC algorithm (using mcmc.control), and 4) options for the output (using output.glm.control). Default values are available in most of the cases. The arguments for the control functions are described in their respective help files.

In the prediction part of the function we want to predict $g_\lambda^{-1}(S^*)$ at locations of interest, where g_λ^{-1} is the inverse Box-Cox transformation. For the prediction part of the algorithm, we use the median of the predictive distribution as the predictor and 1/4 of the length of the 95 percent predictive interval as a measure of the prediction uncertainty. Below we describe the procedure for calculating these quantities.

First we perform a Bayesian Gaussian prediction with the given priors on β and σ^2 on each of the simulated S-"datasets" from the posterior distribution (and in case ϕ is not fixed, for each sampled ϕ value). This Gaussian prediction is done by calling an internal function which is an extension of krige.bayes allowing for more than one "data set".

For calculating the probability below a threshold for the predictive distribution given the data, we first calculate this probability for each of the simulated S-"datasets". This is done using the fact that the predictive distribution for each of the simulated S-"datasets" is a multivariate t-distribution. Afterwards the probability below a threshold is calculated by taking the empirical mean of these conditional probabilities.

Now the median and the 2.5 percent and 97.5 percent quantiles can be calculated by an iterative procedure, where first a guess of the value is made, and second this guess is checked by calculating the probability of being less than this value. In case the guess is too low, it is adjusted upwards, and vise verse.

Value

A list with the following components:

`posterior` A list with results for the posterior distribution of the model parameters and the random effects at the data locations. The components are:

- betasummary of posterior distribution for the parameter β.
- sigmasqsummary of the posterior distribution for the parameter σ^2.
- phisummary of the posterior distribution of the parameter ϕ.
- simulationssample from the posterior distribution of $g_\lambda^{-1}(S)$ at the data locations. Returned only if `keep.mcmc.sim = TRUE`.
- acc.rateThe acceptance rates.

`predictive` A list with results for the predictive distribution at the prediction locations (if provided). The components are:

- simulationsa numerical matrix. Each column contains a simulation from the predictive distribution. Returned only if `sim.predict = TRUE`.
- mediana vector with the estimated median at the prediction locations.
- uncertaintya vector with the estimated uncertainty at the prediction locations, defined as the length of the 95% prediction interval divided by 4.
- quantilesA matrix or vector with quantile estimators.

- probabilityA matrix or vector with probabilities below a threshold. Returned only if the argument `threshold` is used.

`model`	model components used as defined by `model.glm.control`.
`prior`	priors used for the model parameters.
`mcmc.input`	input parameters used for the MCMC algorithm.
`.Random.seed`	system random seed before running the function. Allows reproduction of results. If the `.Random.seed` is set to this value and the function is run again, it will produce exactly the same results.
`call`	the function call.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

References

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

See Also

`pois.krige` for prediction with fixed parameters in the Poisson normal model, `binom.krige.bayes` for Bayesian prediction in the Binomial-normal model, and `krige.bayes` for Bayesian prediction in the Gaussian spatial model.

Examples

```
data(p50)

if(!exists(".Random.seed", envir=.GlobalEnv, inherits = FALSE))
set.seed(1234)
## Not run:
## MCMC with fixed phi
prior.5 <- prior.glm.control(phi.prior = "fixed", phi = 0.1)
mcmc.5 <- mcmc.control(S.scale = 0.01, thin = 1)
test.5 <- pois.krige.bayes(p50, prior = prior.5, mcmc.input = mcmc.5)
par(mfrow=c(1,2))
hist(test.5)
## Now chose S.scale (Acc-rate=0.60 is preferable).
mcmc.5.new <- mcmc.control(S.scale = 0.08, thin = 100)
test.5.new <- pois.krige.bayes(p50,
               locations = t(cbind(c(2.5,3.5),c(-6,3.5),c(2.5,-3.5),c(-6,-3.5))),
               prior = prior.5, mcmc.input = mcmc.5.new,
               output = list(threshold = 10, quantile = c(0.49999,0.99)))
image(test.5.new)
persp(test.5.new)
## MCMC with random phi.
```

```
## Note here that we can start with the S.scale from above.
mcmc.6 <- mcmc.control(S.scale = 0.08, n.iter = 2000, thin = 100,
                       phi.scale = 0.01)
prior.6 <- prior.glm.control(phi.discrete = seq(0.02, 1, 0.02))
test.6 <- pois.krige.bayes(p50, prior = prior.6, mcmc.input = mcmc.6)
## Acc-rate=0.60 , acc-rate-phi = 0.25-0.30  are preferable
mcmc.6.new <- mcmc.control(S.scale=0.08, n.iter = 400000, thin = 200,
                       burn.in = 5000, phi.scale = 0.12, phi.start = 0.5)
prior.6 <- prior.glm.control(phi.prior = "uniform",
               phi.discrete = seq(0.02, 1, 0.02))
test.6.new <- pois.krige.bayes(p50,
                 locations = t(cbind(c(2.5,3.5), c(-60,-37))),
                 prior = prior.6, mcmc.input = mcmc.6.new)
par(mfrow=c(3,1))
hist(test.6.new)

## End(Not run)
```

`prepare.likfit.glsm` *Prepare for Monte Carlo MLE*

Description

This function takes an output object from `glsm.mcmc`, and the corresponding data object of class `geodata`, and prepares the variables needed for the Monte Carlo maximum likelihood function `likfit.glsm`.

Usage

```
prepare.likfit.glsm(mcmc.output, use.intensity = FALSE)
```

Arguments

`mcmc.output`	an output file from the function `glsm.mcmc`.
`use.intensity`	logical. If `use.intensity = TRUE` then the integration variable in the Monte Carlo approximation will not be S but the intensity $g_\lambda^{-1}(S)$. The latter makes it possible to use other link functions in `likfit.glsm` than the one used in `mcmc.output`. Default is `use.intensity = FALSE`.

Value

An object containing the sample and the approximating density to be used in `likfit.glsm`.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

See Also

likfit.glsm for how to use the output from this function, and glsm.mcmc for generating the object mcmc.output.

Examples

```
## Not run:
data(p50)
mcmc.4 <- mcmc.control(S.scale = 0.000035, n.iter = 1000)
kr4 <- list(family="poisson", cov.pars=c(1,1), beta=0)
condsim.4 <- glsm.mcmc(p50, mcmc.input = mcmc.4, model = kr4)
mcmcobj.4 <- prepare.likfit.glsm(condsim.4)
lik.4 <- likfit.glsm(mcmcobj.4, ini.phi = 10, fix.nugget.rel = TRUE)
lik.4.spherical.nugget <- likfit.glsm(mcmcobj.4, ini.phi = 5.59,
                              cov.model = "spherical", nugget.rel = 0.385)

## End(Not run)
```

---

prior.glm.control *Defines prior options*

Description

This auxiliary function defines prior options for pois.krige.bayes and binom.krige.bayes.

Usage

```
prior.glm.control(beta.prior = c("flat", "normal", "fixed"),
                  beta = NULL, beta.var.std = NULL,
                sigmasq.prior = c("uniform", "sc.inv.chisq", "reciprocal", "fixed"),
                  sigmasq = NULL, df.sigmasq = NULL,
                  phi.prior = c("uniform", "exponential","fixed",
                                "squared.reciprocal", "reciprocal"),
                  phi = NULL, phi.discrete = NULL,
                  tausq.rel = 0)
```

Arguments

beta.prior	prior distribution for the mean (vector) parameter β. The options are "flat" (default), "normal" or "fixed".
beta	hyper-parameter for the prior distribution of the mean (vector) parameter β. Only used if beta.prior = "normal" or beta.prior = "fixed". For the latter beta defines the value of the known mean.
beta.var.std	standardised (co)variance hyperparameter(s) for the prior for the mean (vector) parameter beta. The (co)variance matrix for beta is given by the multiplication of this matrix by $sigma^2$. Only used if 'beta.prior = "normal"'.

`sigmasq.prior`	prior distribution for the parameter σ^2. The options are `"uniform"` (default), `"sc.inv.chisq"`, `"reciprocal"` (gives improper posterior), or `"fixed"`.
`sigmasq`	fixed value of the parameter σ^2 when `sigmasq.prior = "fixed"`. Parameter S^2_σ in the scaled inverse-χ^2 prior distribution for σ^2.
`df.sigmasq`	parameter n_σ in the scaled inverse-χ^2 prior distribution for σ^2.
`phi.prior`	prior distribution for the range parameter ϕ. Options are: `"uniform"` ($\propto 1$), `"exponential"` ($\exp(-\nu*\phi)$), `"fixed"` (known value of ϕ), `"squared.reciprocal"` ($1/\phi^2$), `"reciprocal"` ($1/\phi$). Alternativelly, a user defined discrete distribution can be specified by providing a vector of probabilities. These probabilities corrresponds to a prior distribution with support `phi.discrete`. If the `"fixed"` the argument ϕ should be provided and it is regarded as fixed when performing predictions.
`phi`	fixed value of the parameter ϕ when `phi.prior = "fixed"`. Mean of the prior distribution when `phi.prior = "exponential"`.
`phi.discrete`	support points for the discretisation of the prior for the parameter ϕ.
`tausq.rel`	the value of the relative nugget parameter τ^2_R. Default is `tausq.rel = 0`.

Value

A list with processed arguments to be passed to the main function.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

See Also

`pois.krige.bayes` and `binom.krige.bayes`.

`proflik.glsm` *Computes Profile Likelihood for generalised linear spatial models*

Description

Computes two dimenaional profile likelihood for the parameters (phi, nugget.rel) for a model previously derived using the function `likfit.glsm`.

Usage

```
proflik.glsm(mcmc.obj, obj.likfit.glsm, phi.values, nugget.rel.values,
             messages, ...)
```

Arguments

mcmc.obj	object with the Monte Carlo simulations and corresponding approximating density. This object should be an output from the function prepare.likfit.glsm.
obj.likfit.glsm	Output file from likfit.glsm.
phi.values	set of values of the parameter phi for which the profile likelihood will be computed.
nugget.rel.values	set of values of the relative nugget parameter for which the profile likelihood will be computed. Only used if obj.likfit.glsm was created with the option fix.nugget = FALSE.
messages	logical. Indicates whether messages should be printed on the screen (or output device) while the function is running. Note that for this function additional messages can be obtained by setting the global option verbose=TRUE
...	additional parameters to be passed to the maximization function. Typically arguments of the type control() which controls the behavior of the minimization algorithm. For further details, see the documentation for the minimization function optim.

Value

An object of the class '"proflik"' which is a list. The element contains values of the pair of parameters and the corresponding value of the profile likelihood.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

References

Further information about **geoRglm** can be found at:
http://gbi.agrsci.dk/~ofch/geoRglm.

See Also

likfit.glsm for the parameter estimation, and proflik for the profile likelihood in the Gaussian spatial model.

Examples

```
data(p50)

## Not run:
mcmc.5 <- mcmc.control(S.scale = 0.6, thin=20, n.iter=50000, burn.in=1000)
model.5 <- list(cov.pars=c(0.6, 0.1), beta=1, family="poisson")
outmcmc.5 <- glsm.mcmc(p50, model= model.5, mcmc.input = mcmc.5)
mcmcobj.5 <- prepare.likfit.glsm(outmcmc.5)
```

```
lik.5.sph.nugget <- likfit.glsm(mcmcobj.5, ini.phi = 1,
             cov.model = "spherical", nugget.rel = 0.385)
pr.lik.5.sph.nugget <- proflik.glsm(mcmcobj.5, lik.5.sph.nugget,
         phi.values = seq(0.5,5,l=10), nugget.rel.values=seq(0.5,5,l=10))
plot(pr.lik.5.sph.nugget)

## End(Not run)
```

---

rongelap *Radionuclide Concentrations on Rongelap Island*

Description

This data-set was used by Diggle, Tawn and Moyeed (1998) to illustrate the model-based geostatistical methodology introduced in the paper. discussed in the paper. The radionuclide concentration data set consists of measurements of γ-ray counts at 157 locations.

Usage

```
data(rongelap)
```

Format

The object is a list with the following components:

`coords` the coordinates of data locations.

`data` the data.

`units.m` n-dimensional vector of observation-times for the data.

`borders` a matrix with the coordinates defining the coastline on Rongelap Island.

Source

For further details on the radionuclide concentration data, see Diggle, Harper and Simon (1997), Diggle, Tawn and Moyeed (1998) and Christensen (2004).

References

Christensen, O. F. (2004). Monte Carlo maximum likelihood in model-based geostatistics. Journal of computational and graphical statistics **13** 702-718.

Diggle, P. J., Harper, L. and Simon, S. L. (1997). Geostatistical analysis of residual contamination from nuclea testing. In: *Statistics for the environment 3: pollution assesment and control* (eds. V. Barnet and K. F. Turkmann), Wiley, Chichester, 89-107.

Diggle, P. J., Tawn, J. A. and Moyeed, R. A. (1998). Model-based geostatistics (with Discussion). Applied Statistics, 47, 299–350.

summary.likGLSM *Summarizes Parameter Estimation Results for Generalised linear Spatial Models*

Description

Summarizes results returned by the function likfit.glsm.

Functions are *methods* for summary and print for class likGLSM and summary.likGLSM.

Usage

```
## S3 method for class 'likGLSM'
summary(object,...)
## S3 method for class 'likGLSM'
print(x, digits = max(3, getOption("digits") - 3),...)
## S3 method for class 'summary.likGLSM'
print(x, digits = max(3, getOption("digits") - 3),...)
```

Arguments

object	an object of class likGLSM, typically a result of a call to likfit.glsm.
x	an object of class likGLSM or class summary.likGLSM, typically resulting from a call to likfit.glsm.
digits	the number of significant digits to use when printing.
...	extra arguments for print.

Details

A detailed summary of a object of the class likGLSM is produced by by summary.likGLSM and printed by print.summary.likGLSM. This includes model specification with values of fixed and estimated parameters. A simplified summary of the parameter estimation is printed by print.likGLSM.

Value

print.likGLSM prints the parameter estimates and the value of the maximized likelihood.
summary.likGLSM returns a list with main results of a call to likfit.glsm.
print.summary.likGLSM prints these results on the screen (or other output device) in a "nice" way.

Author(s)

Ole F. Christensen <OleF.Christensen@agrsci.dk>,
Paulo J. Ribeiro Jr. <Paulo.Ribeiro@est.ufpr.br>.

See Also

likfit.glsm, print, summary.

Examples

```
## See examples for the function likfit.glsm
```

Index

∗Topic **aplot**
lines.covariomodel, 28
∗Topic **datasets**
b50, p50 and b64, 3
rongelap, 44
∗Topic **dplot**
hist.glm.krige.bayes, 21
plot.covariogram, 32
∗Topic **print**
summary.likGLSM, 45
∗Topic **spatial**
asympvar, 2
binom.krige, 4
binom.krige.bayes, 7
covariog, 11
covariog.model.env, 13
create.mcmc.coda, 15
geoRglm-defunct, 16
glsm.krige, 16
glsm.mcmc, 18
hist.glm.krige.bayes, 21
image.glm.krige.bayes, 22
krige.glm.control, 23
likfit.glsm, 25
lines.covariomodel, 28
mcmc.control, 29
model.glm.control, 30
output.glm.control, 31
plot.covariogram, 32
pois.krige, 33
pois.krige.bayes, 36
prepare.likfit.glsm, 40
prior.glm.control, 41
proflik.glsm, 42
summary.likGLSM, 45
∗Topic **utilities**
geoRglm-defunct, 16
.NewtonRhapson.step *(*likfit.glsm*)*, 25
.Random.seed, *10*, *39*
.func.val *(*likfit.glsm*)*, 25
.lik.sim *(*likfit.glsm*)*, 25
.maxim.aux1 *(*likfit.glsm*)*, 25
.mcmc.aux *(*pois.krige*)*, 33
.mcmc.bayes.binom.logit *(*binom.krige.bayes*)*, 7
.mcmc.bayes.conj.binom.logit *(*binom.krige.bayes*)*, 7
.mcmc.bayes.conj.pois.boxcox *(*pois.krige.bayes*)*, 36
.mcmc.bayes.conj.pois.log *(*pois.krige.bayes*)*, 36
.mcmc.bayes.pois.boxcox *(*pois.krige.bayes*)*, 36
.mcmc.bayes.pois.log *(*pois.krige.bayes*)*, 36
.mcmc.binom.aux *(*binom.krige*)*, 4
.mcmc.binom.logit *(*binom.krige*)*, 4
.mcmc.boxcox.aux *(*pois.krige*)*, 33
.mcmc.pois.boxcox *(*pois.krige*)*, 33
.mcmc.pois.log *(*pois.krige*)*, 33

asympvar, 2

b50 *(*b50, p50 and b64*)*, 3
b50, p50 and b64, 3
b64 *(*b50, p50 and b64*)*, 3
binom.krige, 4, *10*, *20*, *23*, *25*, *29*, *30*, *35*
binom.krige.bayes, *6*, 7, *15*, *21–23*, *29–32*, *39*, *41*, *42*

coords.aniso, *19*, *24*, *25*, *30*
cov.spatial, *18*, *24*, *25*, *30*
covariog, 11, *13*, *14*, *32*, *33*
covariog.model.env, *13*, 13, *32*, *33*
create.mcmc.coda, 15

density, *21*

geoRglm-defunct, 16
geoRglmdefunct *(*geoRglm-defunct*)*, 16

glsm.krige, 16
glsm.mcmc, *15–17*, 18, *27*, *40*, *41*

hist, *21*
hist.glm.krige.bayes, 21
hist.krige.bayes, *21*

image, *22*, *23*
image.glm.krige.bayes, 22
image.krige.bayes, *23*
invisible, *21*

krige.bayes, *9*, *10*, *23*, *38*, *39*
krige.conv, *5*, *6*, *17*, *34*, *35*
krige.glm.control, *5*, 23, *34*

likfit, *27*
likfit.glsm, *18*, 25, *40–43*, *45*
lines, *28*, *32*
lines.covariomodel, 28, *32*, *33*

mcmc.control, *5*, *8*, *9*, *15*, *19*, 29, *34*, *36–38*
model.glm.control, *7*, *9*, *10*, 30, *36*, *37*, *39*

optim, *26*, *43*
output.glm.control, *5*, *8*, *9*, *17*, 31, *34*, *37*, *38*

p50, *16*
p50 *(*b50, p50 and b64*)*, 3
pars.limits, *26*
persp, *22*, *23*
persp.glm.krige.bayes *(*image.glm.krige.bayes*)*, 22
plot, *32*, *33*
plot.covariogram, *13*, *14*, *28*, 32
pois.krige, *6*, *16*, *20*, *23–25*, *29*, *30*, 33, *37*, *39*
pois.krige.bayes, *10*, *15*, *21–23*, *29–32*, *35*, *36*, *41*, *42*
pois.log.krige *(*geoRglm-defunct*)*, 16
prepare.likfit.glsm, *25–27*, 40, *43*
print, *45*
print.likGLSM *(*summary.likGLSM*)*, 45
print.summary.likGLSM *(*summary.likGLSM*)*, 45
prior.glm.control, *8*, *9*, *36–38*, 41
proflik, *43*
proflik.glsm, 42

rongelap, 44

summary, *45*
summary.likGLSM, 45

trend.spatial, *18*, *24*, *25*, *30*

y50 *(*geoRglm-defunct*)*, 16