Package 'convoSPAT'

October 12, 2022

Type Package

Title Convolution-Based Nonstationary Spatial Modeling

Version 1.2.7

Date 2021-01-15

Description Fits convolution-based nonstationary
Gaussian process models to point-referenced spatial data. The nonstationary covariance function allows the user to specify the underlying correlation structure and which spatial dependence parameters should be allowed to vary over space: the anisotropy, nugget variance, and process variance. The parameters are estimated via maximum likelihood, using a local likelihood approach. Also provided are functions to fit stationary spatial models for comparison, calculate the Kriging predictor and standard errors, and create various plots to visualize nonstationarity.

Depends R (>= 3.1.2)

License MIT + file LICENSE

LazyData TRUE

Imports stats, graphics, ellipse, fields, MASS, plotrix, StatMatch

URL http://github.com/markdrisser/convoSPAT

RoxygenNote 7.1.0

NeedsCompilation no

Author Mark D. Risser [aut, cre]

Maintainer Mark D. Risser <markdrisser@gmail.com>

Repository CRAN

Date/Publication 2021-01-16 00:50:04 UTC

R topics documented:

`Aniso_fit` *Fit the stationary spatial model*

Description

`Aniso_fit` estimates the parameters of the stationary spatial model. Required inputs are the observed data and locations. Optional inputs include the covariance model (exponential is the default).

Usage

```
Aniso_fit(
  sp.SPDF = NULL,
  coords = NULL,
  data = NULL,
  cov.model = "exponential",
  mean.model = data ~ 1,
  fixed.nugg2.var = NULL,
  method = "reml",
  fix.tausq = FALSE,
  tausq = 0,
  fix.kappa = FALSE,
  kappa = 0.5,
```

```
  local.pars.LB = NULL,
  local.pars.UB = NULL,
  local.ini.pars = NULL
)
```

Arguments

`sp.SPDF` A "SpatialPointsDataFrame" object, which contains the spatial coordinates and additional attribute variables corresponding to the spatoal coordinates

`coords` An N x 2 matrix where each row has the two-dimensional coordinates of the N data locations.

`data` A vector or matrix with N rows, containing the data values. Inputting a vector corresponds to a single replicate of data, while inputting a matrix corresponds to replicates. In the case of replicates, the model assumes the replicates are independent and identically distributed.

`cov.model` A string specifying the model for the correlation function; defaults to "exponential". Options available in this package are: "exponential", "matern", or "gaussian".

`mean.model` An object of class formula, specifying the mean model to be used. Defaults to an intercept only.

`fixed.nugg2.var`
Optional; describes the variance/covariance for a fixed (second) nugget term (represents a known error term). Either a vector of length N containing a station-specific variances (implying independent error) or an NxN covariance matrix (implying dependent error). Defaults to zero.

`method` Indicates the estimation method, either maximum likelihood ("ml") or restricted maximum likelihood ("reml").

`fix.tausq` Logical; indicates whether the default nugget term (tau^2) should be fixed (TRUE) or estimated (FALSE). Defaults to FALSE.

`tausq` Scalar; fixed value for the nugget variance (when fix.tausq = TRUE).

`fix.kappa` Logical; indicates if the kappa parameter should be fixed (TRUE) or estimated (FALSE). Defaults to FALSE (only valid for cov.model = "matern" and cov.model = "cauchy").

`kappa` Scalar; value of the kappa parameter. Only used if fix.kappa = TRUE.

`local.pars.LB, local.pars.UB`
Optional vectors of lower and upper bounds, respectively, used by the "L-BFGS-B" method option in the optim function for the local parameter estimation. Each vector must be of length five, containing values for lam1, lam2, tausq, sigmasq, and nu. Default for local.pars.LB is rep(1e-05,5); default for local.pars.UB is c(max.distance/2, max.distance/2, 4*resid.var, 4*resid.var, 100), where max.distance is the maximum interpoint distance of the observed data and resid.var is the residual variance from using lm with mean.model.

`local.ini.pars` Optional vector of initial values used by the "L-BFGS-B" method option in the optim function for the local parameter estimation. The vector must be of length five, containing values for lam1, lam2, tausq, sigmasq, and nu. Defaults to c(max.distance/10, max.distance/10, 0.1*resid.var, 0.9*resid.var, 1), where max.distance is the maximum interpoint distance of the observed data and resid.var is the residual variance from using lm with mean.model.

Value

A list with the following components:

MLEs.save	Table of local maximum likelihood estimates for each mixture component location.
data	Observed data values.
beta.GLS	Vector of generalized least squares estimates of beta, the mean coefficients.
beta.cov	Covariance matrix of the generalized least squares estimate of beta.
Mean.coefs	"Regression table" for the mean coefficient estimates, listing the estimate, standard error, and t-value.
Cov.mat	Estimated covariance matrix (N.obs x N.obs) using all relevant parameter estimates.
Cov.mat.chol	Cholesky of Cov.mat (i.e., chol(Cov.mat)), the estimated covariance matrix (N.obs x N.obs).
aniso.pars	Vector of MLEs for the anisotropy parameters lam1, lam2, eta.
aniso.mat	2 x 2 anisotropy matrix, calculated from aniso.pars.
tausq.est	Scalar maximum likelihood estimate of tausq (nugget variance).
sigmasq.est	Scalar maximum likelihood estimate of sigmasq (process variance).
kappa.MLE	Scalar maximum likelihood estimate for kappa (when applicable).
fixed.nugg2.var	N x N matrix with the fixed variance/covariance for the second (measurement error) nugget term (defaults to zero).
cov.model	String; the correlation model used for estimation.
coords	N x 2 matrix of observation locations.
global.loglik	Scalar value of the maximized likelihood from the global optimization (if available).
Xmat	Design matrix, obtained from using lm with mean.model.
fix.kappa	Logical, indicating if kappa was fixed (TRUE) or estimated (FALSE).
kappa	Scalar; fixed value of kappa.

Examples

```
## Not run:
# Using iid standard Gaussian data
aniso.fit <- Aniso_fit( coords = cbind(runif(100), runif(100)),
data = rnorm(100) )

## End(Not run)
```

cov_spatial *Calculate spatial covariance.*

Description

This function replaces the geoR function `cov.spatial`, which is now defunct. Options available in this package are: `"exponential"`, `"matern"`, and `"gaussian"`.

Usage

```
cov_spatial(
  Dist.mat,
  cov.model = "exponential",
  cov.pars = c(1, 1),
  kappa = 0.5
)
```

Arguments

`Dist.mat`	A matrix of scaled distances.
`cov.model`	A string specifying the model for the correlation function; defaults to `"exponential"`. Options available in this package are: `"exponential"`, `"matern"`, and `"gaussian"`.
`cov.pars`	Fixed values; not used in the function.
`kappa`	Scalar; value of the smoothness parameter.

Value

This function returns a correlation matrix.

Examples

```
Distmat <- as.matrix(dist(matrix(runif(20), ncol = 2), diag = TRUE, upper = TRUE))
C <- cov_spatial( Dist.mat = Distmat )
```

evaluate_CV *Evaluation criteria*

Description

Calculate three evaluation criteria – continuous rank probability score (CRPS), prediction mean square deviation ratio (pMSDR), and mean squared prediction error (MSPE) – comparing hold-out data and predictions.

Usage

```
evaluate_CV(holdout.data, pred.mean, pred.SDs)
```

Arguments

holdout.data	Observed/true data that has been held out for model comparison.
pred.mean	Predicted mean values corresponding to the hold-out locations.
pred.SDs	Predicted standard errors corresponding to the hold-out locations.

Value

A list with the following components:

CRPS	The CRPS averaged over all hold-out locations.
MSPE	The mean squared prediction error.
pMSDR	The prediction mean square deviation ratio.

Examples

```
## Not run:
evaluate_CV( holdout.data = simdata$sim.data[holdout.index],
pred.mean = pred.NS$pred.means, pred.SDs = pred.NS$pred.SDs )

## End(Not run)
```

---

f_mc_kernels *Calculate mixture component kernel matrices.*

Description

f_mc_kernels calculates spatially-varying mixture component kernels using generalized linear models for each of the eigenvalues (lam1 and lam2) and the angle of rotation (eta).

Usage

```
f_mc_kernels(
  y.min = 0,
  y.max = 5,
  x.min = 0,
  x.max = 5,
  N.mc = 3^2,
  lam1.coef = c(-1.3, 0.5, -0.6),
  lam2.coef = c(-1.4, -0.1, 0.2),
  logit.eta.coef = c(0, -0.15, 0.15)
)
```

Arguments

`y.min`	Lower bound for the y-coordinate axis.
`y.max`	Upper bound for the y-coordinate axis.
`x.min`	Lower bound for the y-coordinate axis.
`x.max`	Upper bound for the y-coordinate axis.
`N.mc`	Number of mixture component locations.
`lam1.coef`	Log-linear regression coefficients for lam1; the coefficients correspond to the intercept, longitude, and latitude.
`lam2.coef`	Log-linear regression coefficients for lam2; the coefficients correspond to the intercept, longitude, and latitude.
`logit.eta.coef`	Scaled logit regression coefficients for eta; the coefficients correspond to the intercept, longitude, and latitude.

Value

A list with the following components:

`mc.locations`	A `N.mc` x 2 matrix of the mixture component locations.
`mc.kernels`	A `N.mc` x 2 x 2 array of kernel matrices corresponding to each of the mixture component locations.

Examples

```
f_mc_kernels( y.min = 0, y.max = 5, x.min = 0,
x.max = 5, N.mc = 3^2, lam1.coef = c(-1.3, 0.5, -0.6),
lam2.coef = c(-1.4, -0.1, 0.2), logit.eta.coef = c(0, -0.15, 0.15) )
```

`kernel_cov` *Calculate a kernel covariance matrix.*

Description

`kernel_cov` calculates a 2 x 2 matrix based on the eigendecomposition components (two eigenvalues and angle of rotation).

Usage

```
kernel_cov(params)
```

Arguments

`params`	A vector of three parameters, corresponding to (lam1, lam2, eta). The eigenvalues (lam1 and lam2) must be positive.

Value

A 2 x 2 kernel covariance matrix.

Examples

```
kernel_cov(c(1, 2, pi/3))
```

---

make_global_loglik1 *Constructor functions for global parameter estimation.*

Description

This function generates another function to be used within `optim` to obtain maximum likelihood estimates of global variance parameters tausq, sigmasq with a fixed correlation matrix (smoothness is fixed).

Usage

```
make_global_loglik1(data, Xmat, Corr, nugg2.var)
```

Arguments

`data`	A vector or matrix of data to use in the likelihood calculation.
`Xmat`	The design matrix for the mean model.
`Corr`	The correlation matrix.
`nugg2.var`	Fixed values for the covariance of the second nugget term.

Value

This function returns another function for use in `optim`.

Examples

```
## Not run:
make_global_loglik1( data, Xmat, Corr, nugg2.var )

## End(Not run)
```

make_global_loglik1_kappa
Constructor functions for global parameter estimation.

Description

This function generates another function to be used within `optim` to obtain maximum likelihood estimates of global variance parameters tausq, sigmasq, and nu.

Usage

```
make_global_loglik1_kappa(data, Xmat, cov.model, Scalemat, Distmat, nugg2.var)
```

Arguments

`data`	A vector or matrix of data to use in the likelihood calculation.
`Xmat`	The design matrix for the mean model.
`cov.model`	String; the covariance model.
`Scalemat`	Matrix; contains the scaling quantities from the covariance function.
`Distmat`	Matrix; contains the scaled distances.
`nugg2.var`	Fixed values for the covariance of the second nugget term.

Value

This function returns another function for use in `optim`.

Examples

```
## Not run:
make_global_loglik1_kappa( data, Xmat, cov.model, Scalemat, Distmat, nugg2.var )

## End(Not run)
```

make_global_loglik2 *Constructor functions for global parameter estimation.*

Description

This function generates another function to be used within `optim` to obtain maximum likelihood estimates of global variance parameter sigmasq with a fixed correlation matrix (smoothness is fixed). The nugget variance is taken to be spatially-varing.

Usage

```
make_global_loglik2(data, Xmat, Corr, obs.nuggets, nugg2.var)
```

Arguments

data	A vector or matrix of data to use in the likelihood calculation.
Xmat	The design matrix for the mean model.
Corr	The correlation matrix.
obs.nuggets	A vector containing the spatially-varying nuggets corresponding to each data location.
nugg2.var	Fixed values for the covariance of the second nugget term.

Value

This function returns another function for use in optim.

Examples

```
## Not run:
make_global_loglik2( data, Xmat, Corr, obs.nuggets, nugg2.var )

## End(Not run)
```

---

make_global_loglik2_kappa

Constructor functions for global parameter estimation.

Description

This function generates another function to be used within optim to obtain maximum likelihood estimates of global variance parameters sigmasq and nu. The nugget variance is taken to be spatially-varying.

Usage

```
make_global_loglik2_kappa(
  data,
  Xmat,
  cov.model,
  Scalemat,
  Distmat,
  obs.nuggets,
  nugg2.var
)
```

Arguments

data	A vector or matrix of data to use in the likelihood calculation.
Xmat	The design matrix for the mean model.
cov.model	String; the covariance model.
Scalemat	Matrix; contains the scaling quantities from the covariance function.
Distmat	Matrix; contains the scaled distances.
obs.nuggets	A vector containing the spatially-varying nuggets corresponding to each data location.
nugg2.var	Fixed values for the covariance of the second nugget term.

Value

This function returns another function for use in optim.

Examples

```
## Not run:
make_global_loglik2_kappa( data, Xmat, cov.model, Scalemat, Distmat, obs.nuggets, nugg2.var )

## End(Not run)
```

---

make_global_loglik3 *Constructor functions for global parameter estimation.*

Description

This function generates another function to be used within optim to obtain maximum likelihood estimates of global variance parameter tausq with a fixed correlation matrix (smoothness is fixed). The process variance is taken to be spatially-varing.

Usage

```
make_global_loglik3(data, Xmat, Corr, obs.variance, nugg2.var)
```

Arguments

data	A vector or matrix of data to use in the likelihood calculation.
Xmat	The design matrix for the mean model.
Corr	The correlation matrix matrix.
obs.variance	A vector containing the spatially-varying variance corresponding to each data location.
nugg2.var	Fixed values for the covariance of the second nugget term.

Value

This function returns another function for use in `optim`.

Examples

```
## Not run:
make_global_loglik3( data, Xmat, Corr, obs.variance, nugg2.var )

## End(Not run)
```

---

make_global_loglik3_kappa

Constructor functions for global parameter estimation.

Description

This function generates another function to be used within `optim` to obtain maximum likelihood estimates of global variance parameters tausq and nu. The process variance is taken to be spatially-varying.

Usage

```
make_global_loglik3_kappa(
  data,
  Xmat,
  cov.model,
  Scalemat,
  Distmat,
  obs.variance,
  nugg2.var
)
```

Arguments

`data`	A vector or matrix of data to use in the likelihood calculation.
`Xmat`	The design matrix for the mean model.
`cov.model`	String; the covariance model.
`Scalemat`	Matrix; contains the scaling quantities from the covariance function.
`Distmat`	Matrix; contains the scaled distances.
`obs.variance`	A vector containing the spatially-varying variance corresponding to each data location.
`nugg2.var`	Fixed values for the covariance of the second nugget term.

Value

This function returns another function for use in optim.

Examples

```
## Not run:
make_global_loglik3_kappa( data, Xmat, cov.model, Scalemat, Distmat, obs.variance, nugg2.var )

## End(Not run)
```

---

make_global_loglik4_kappa
Constructor functions for global parameter estimation.

Description

This function generates another function to be used within optim to obtain maximum likelihood estimates of global variance parameters nu. The process variance and nugget variance are taken to be spatially-varying.

Usage

```
make_global_loglik4_kappa(
  data,
  Xmat,
  cov.model,
  Scalemat,
  Distmat,
  obs.variance,
  obs.nuggets,
  nugg2.var
)
```

Arguments

data	A vector or matrix of data to use in the likelihood calculation.
Xmat	The design matrix for the mean model.
cov.model	String; the covariance model.
Scalemat	Matrix; contains the scaling quantities from the covariance function.
Distmat	Matrix; contains the scaled distances.
obs.variance	A vector containing the spatially-varying variance corresponding to each data location.
obs.nuggets	A vector containing the spatially-varying nuggets corresponding to each data location.
nugg2.var	Fixed values for the covariance of the second nugget term.

Value

This function returns another function for use in optim.

Examples

```
## Not run:
make_global_loglik4_kappa( data, Xmat, cov.model, Scalemat, Distmat,
obs.variance, obs.nuggets, nugg2.var )

## End(Not run)
```

---

make_local_lik *Constructor functions for local parameter estimation.*

Description

This function generates another function to be used within optim to obtain maximum likelihood estimates of covariance (and possibly mean) parameters. The function includes options for (1) maximum likelihood ("ml") vs. restricted maximum likelihood ("reml"), (2) smoothness (kappa): models without smoothness vs. estimating the smoothness vs. using fixed smoothness, (3) locally isotropic vs. locally anisotropic, and (4) fixed nugget variance (tausq): fixed vs. estimated.

Usage

```
make_local_lik(
  locations,
  cov.model,
  data,
  Xmat,
  nugg2.var = matrix(0, nrow(locations), nrow(locations)),
  tausq = 0,
  kappa = 0.5,
  fixed = rep(FALSE, 6),
  method = "reml",
  local.aniso = TRUE,
  fix.tausq = FALSE,
  fix.kappa = FALSE
)
```

Arguments

locations	A matrix of locations.
cov.model	String; the covariance model.
data	A vector or matrix of data to use in the likelihood calculation.
Xmat	The design matrix for the mean model.

nugg2.var	Fixed values for the variance/covariance of the second nugget term; defaults to a matrix of zeros.
tausq	Scalar; fixed value for the nugget variance (when fix.tausq = TRUE).
kappa	Scalar; fixed value for the smoothness (when fix.kappa = TRUE).
fixed	Logical vector of FALSE values; length corresponds to the number of parameters to be estimated.
method	Indicates the estimation method, either maximum likelihood ("ml") or restricted maximum likelihood ("reml").
local.aniso	Logical; indicates if the local covariance should be anisotropic (TRUE) or isotropic (FALSE). Defaults to TRUE.
fix.tausq	Logical; indicates whether the default nugget term (tau^2) should be fixed (TRUE) or estimated (FALSE). Defaults to FALSE.
fix.kappa	Logical; indicates if the kappa parameter should be fixed (TRUE) or estimated (FALSE). Defaults to FALSE (only valid for cov.model = "matern" and cov.model = "cauchy").

Value

This function returns another function for use in optim.

Examples

```
## Not run:
make_local_lik( locations, cov.model, data, Xmat )

## End(Not run)
```

---

mc_N *Calculate local sample sizes.*

Description

mc_N calculates the number of observations (sample size) that fall within a certain fit radius for each mixture component location.

Usage

```
mc_N(coords, mc.locations, fit.radius)
```

Arguments

coords	A matrix of observation locations.
mc.locations	A matrix of the mixture component locations to use in the model fitting.
fit.radius	Scalar; defines the fitting radius for local likelihood estimation.

Value

A vector mc.N.fit, which summarizes the number of observation locations in coords that fall within the fit radius for each mixture component location.

Examples

```
## Not run:
mc_N( coords = simdata$sim.locations, mc.locations = simdata$mc.locations,
fit.radius = 1 )

## End(Not run)
```

NSconvo_fit *Fit the nonstationary spatial model*

Description

NSconvo_fit estimates the parameters of the nonstationary convolution-based spatial model. Required inputs are the observed data and locations. Optional inputs include mixture component locations (if not provided, the number of mixture component locations are required), the fit radius, the covariance model (exponential is the default), and whether or not the nugget and process variance will be spatially-varying.

Usage

```
NSconvo_fit(
  sp.SPDF = NULL,
  coords = NULL,
  data = NULL,
  cov.model = "exponential",
  mean.model = data ~ 1,
  mc.locations = NULL,
  N.mc = NULL,
  lambda.w = NULL,
  fixed.nugg2.var = NULL,
  mean.model.df = NULL,
  mc.kernels = NULL,
  fit.radius = NULL,
  ns.nugget = FALSE,
  ns.variance = FALSE,
  ns.mean = FALSE,
  local.aniso = TRUE,
  fix.tausq = FALSE,
  tausq = 0,
  fix.kappa = FALSE,
  kappa = 0.5,
```

```
    method = "reml",
    print.progress = TRUE,
    local.pars.LB = NULL,
    local.pars.UB = NULL,
    global.pars.LB = NULL,
    global.pars.UB = NULL,
    local.ini.pars = NULL,
    global.ini.pars = NULL
  )
```

Arguments

- `sp.SPDF` — A "SpatialPointsDataFrame" object, which contains the spatial coordinates and additional attribute variables corresponding to the spatoal coordinates
- `coords` — An N x 2 matrix where each row has the two-dimensional coordinates of the N data locations.
- `data` — A vector or matrix with N rows, containing the data values. Inputting a vector corresponds to a single replicate of data, while inputting a matrix corresponds to replicates. In the case of replicates, the model assumes the replicates are independent and identically distributed.
- `cov.model` — A string specifying the model for the correlation function; defaults to "exponential". Options available in this package are: "exponential", "matern", and "gaussian".
- `mean.model` — An object of class formula, specifying the mean model to be used. Defaults to an intercept only.
- `mc.locations` — Optional; matrix of mixture component locations.
- `N.mc` — Optional; if `mc.locations` is not specified, the function will create a rectangular grid of size `N.mc` over the spatial domain.
- `lambda.w` — Scalar; tuning parameter for the weight function. Defaults to be the square of one-half of the minimum distance between mixture component locations.
- `fixed.nugg2.var` — Optional; describes the variance/covariance for a fixed (second) nugget term (represents a known error term). Either a vector of length N containing a station-specific variances (implying independent error) or an NxN covariance matrix (implying dependent error). Defaults to zero.
- `mean.model.df` — Optional data frame; refers to the variables used in `mean.model`. Important when using categorical variables in `mean.model`, as a subset of the full design matrix will likely be rank deficient. Specifying `mean.model.df` allows `NSconvo_fit` to calculate a design matrix specific to the points used to fit each local model.
- `mc.kernels` — Optional specification of mixture component kernel matrices (based on expert opinion, etc.).
- `fit.radius` — Scalar; specifies the fit radius or neighborhood size for the local likelihood estimation.
- `ns.nugget` — Logical; indicates if the nugget variance (tausq) should be spatially-varying (`TRUE`) or constant (`FALSE`).

ns.variance
: Logical; indicates if the process variance (sigmasq) should be spatially-varying (TRUE) or constant (FALSE).

ns.mean
: Logical; indicates if the mean coefficeints (beta) should be spatially-varying (TRUE) or constant (FALSE).

local.aniso
: Logical; indicates if the local covariance should be anisotropic (TRUE) or isotropic (FALSE). Defaults to TRUE. In the case of a locally isotropic model, the bounds and initial values for lam will default to the first element of local.pars.LB, local.pars.UB, and local.ini.pars (while still required, the second and third elements of these vectors will be ignored.)

fix.tausq
: Logical; indicates whether the default nugget term (tau^2) should be fixed (TRUE) or estimated (FALSE). Defaults to FALSE.

tausq
: Scalar; fixed value for the nugget variance (when fix.tausq = TRUE).

fix.kappa
: Logical; indicates if the kappa parameter should be fixed (TRUE) or estimated (FALSE). Defaults to FALSE (only valid for cov.model = "matern" and cov.model = "cauchy").

kappa
: Scalar; value of the kappa parameter. Only used if fix.kappa = TRUE.

method
: Indicates the estimation method, either maximum likelihood ("ml") or restricted maximum likelihood ("reml").

print.progress
: Logical; if TRUE, text indicating the progress of local model fitting in real time.

local.pars.LB, local.pars.UB
: Optional vectors of lower and upper bounds, respectively, used by the "L-BFGS-B" method option in the optim function for the local parameter estimation. Each vector must be of length five, containing values for lam1, lam2, tausq, sigmasq, and nu. Default for local.pars.LB is rep(1e-05,5); default for local.pars.UB is c(max.distance/2, max.distance/2, 4*resid.var, 4*resid.var, 100), where max.distance is the maximum interpoint distance of the observed data and resid.var is the residual variance from using lm with mean.model.

global.pars.LB, global.pars.UB
: Optional vectors of lower and upper bounds, respectively, used by the "L-BFGS-B" method option in the optim function for the global parameter estimation. Each vector must be of length three, containing values for tausq, sigmasq, and nu. Default for global.pars.LB is rep(1e-05,3); default for global.pars.UB is c(4*resid.var, 4*resid.var, 100), where resid.var is the residual variance from using lm with mean.model.

local.ini.pars
: Optional vector of initial values used by the "L-BFGS-B" method option in the optim function for the local parameter estimation. The vector must be of length five, containing values for lam1, lam2, tausq, sigmasq, and nu. Defaults to c(max.distance/10, max.distance/10, 0.1*resid.var, 0.9*resid.var, 1), where max.distance is the maximum interpoint distance of the observed data and resid.var is the residual variance from using lm with mean.model.

global.ini.pars
: Optional vector of initial values used by the "L-BFGS-B" method option in the optim function for the global parameter estimation. The vector must be of length three, containing values for tausq, sigmasq, and nu. Defaults to c(0.1*resid.var, 0.9*resid.var, 1), where resid.var is the residual variance from using lm with mean.model.

Value

A "NSconvo" object, with the following components:

mc.locations
: Mixture component locations used for the simulated data.

mc.kernels
: Mixture component kernel matrices used for the simulated data.

MLEs.save
: Table of local maximum likelihood estimates for each mixture component location.

kernel.ellipses
: N.obs x 2 x 2 array, containing the kernel matrices corresponding to each of the simulated values.

data
: Observed data values.

beta.GLS
: Generalized least squares estimates of beta, the mean coefficients. For ns.mean = FALSE, this is a vector (containing the global mean coefficients); for ns.mean = TRUE, this is a matrix (one column for each mixture component location).

beta.cov
: Covariance matrix of the generalized least squares estimate of beta. For ns.mean = FALSE, this is a matrix (containing the covariance of theglobal mean coefficients); for ns.mean = TRUE, this is an array (one matrix for each mixture component location).

Mean.coefs
: "Regression table" for the mean coefficient estimates, listing the estimate, standard error, and t-value (for ns.mean = FALSE only).

tausq.est
: Estimate of tausq (nugget variance), either scalar (when ns.nugget = "FALSE") or a vector of length N (when ns.nugget = "TRUE"), which contains the estimated nugget variance for each observation location.

sigmasq.est
: Estimate of sigmasq (process variance), either scalar (when ns.variance = "FALSE") or a vector of length N (when ns.variance = "TRUE"), which contains the estimated process variance for each observation location.

beta.est
: Estimate of beta (mean coefficients), either a vector (when ns.mean = "FALSE") or a matrix with N rows (when ns.mean = "TRUE"), each row of which contains the estimated (smoothed) mean coefficients for each observation location.

kappa.MLE
: Scalar maximum likelihood estimate for kappa (when applicable).

Cov.mat
: Estimated covariance matrix (N.obs x N.obs) using all relevant parameter estimates.

Cov.mat.chol
: Cholesky of Cov.mat (i.e., chol(Cov.mat)), the estimated covariance matrix (N.obs x N.obs).

cov.model
: String; the correlation model used for estimation.

ns.nugget
: Logical, indicating if the nugget variance was estimated as spatially-varing (TRUE) or constant (FALSE).

ns.variance
: Logical, indicating if the process variance was estimated as spatially-varying (TRUE) or constant (FALSE).

fixed.nugg2.var
: N x N matrix with the fixed variance/covariance for the second (measurement error) nugget term (defaults to zero).

coords
: N x 2 matrix of observation locations.

global.loglik	Scalar value of the maximized likelihood from the global optimization (if available).
Xmat	Design matrix, obtained from using lm with mean.model.
lambda.w	Tuning parameter for the weight function.
fix.kappa	Logical, indicating if kappa was fixed (TRUE) or estimated (FALSE).
kappa	Scalar; fixed value of kappa.

Examples

```
## Not run:
# Using white noise data
fit.model <- NSconvo_fit( coords = cbind( runif(100), runif(100)),
data = rnorm(100), fit.radius = 0.4, N.mc = 4 )

## End(Not run)
```

NSconvo_sim *Simulate data from the nonstationary model.*

Description

NSconvo_sim simulates data from the nonstationary model, given mixture component kernel matrices. The function requires either a mixture component kernel object, from the function f.mc.kernels(), or a direct specification of the mixture component locations and mixture component kernels.

Usage

```
NSconvo_sim(
  grid = TRUE,
  y.min = 0,
  y.max = 5,
  x.min = 0,
  x.max = 5,
  N.obs = 20^2,
  sim.locations = NULL,
  mc.kernels.obj = NULL,
  mc.kernels = NULL,
  mc.locations = NULL,
  lambda.w = NULL,
  tausq = 0.1,
  sigmasq = 1,
  beta.coefs = 4,
  kappa = NULL,
  covariates = rep(1, N.obs),
  cov.model = "exponential"
)
```

Arguments

grid	Logical; indicates of the simulated data should fall on a grid (TRUE) or not (FALSE).
y.min	Lower bound for the y-coordinate axis.
y.max	Upper bound for the y-coordinate axis.
x.min	Lower bound for the y-coordinate axis.
x.max	Upper bound for the y-coordinate axis.
N.obs	Number of simulated data values.
sim.locations	Optional N.obs x 2 matrix; allows the user to specify the locations of the simulated data.
mc.kernels.obj	Object from the f_mc_kernels function.
mc.kernels	Optional specification of mixture component kernel matrices.
mc.locations	Optional specification of mixture component locations.
lambda.w	Scalar; tuning parameter for the weight function.
tausq	Scalar; true nugget variance.
sigmasq	Scalar; true process variance.
beta.coefs	Vector of true regression coefficients. Length must match the number of columns in covariates.
kappa	Scalar; true smoothness.
covariates	Matrix with N.obs rows, corresponding to covariate information for each of the simualted values.
cov.model	A string specifying the model for the correlation function; defaults to "exponential". Options available in this package are: "exponential", "matern", and "gaussian".

Value

A list with the following components:

sim.locations	Matrix of locations for the simulated values.
mc.locations	Mixture component locations used for the simulated data.
mc.kernels	Mixture component kernel matrices used for the simulated data.
kernel.ellipses	N.obs x 2 x 2 array, containing the kernel matrices corresponding to each of the simulated values.
Cov.mat	True covariance matrix (N.obs x N.obs) corresponding to the simulated data.
sim.data	Simulated data values.
lambda.w	Tuning parameter for the weight function.

Examples

```
## Not run:
NSconvo_sim( grid = TRUE, y.min = 0, y.max = 5, x.min = 0,
x.max = 5, N.obs = 20^2, sim.locations = NULL, mc.kernels.obj = NULL,
mc.kernels = NULL, mc.locations = NULL, lambda.w = NULL,
tausq = 0.1, sigmasq = 1, beta.coefs = 4, kappa = NULL,
covariates = rep(1,N.obs), cov.model = "exponential" )

## End(Not run)
```

`plot.Aniso` *Plot of the estimated correlations from the stationary model.*

Description

This function plots the estimated correlation between a reference point and all other prediction locations.

Usage

```
## S3 method for class 'Aniso'
plot(x, ref.loc = NULL, all.pred.locs = NULL, grid = TRUE, ...)
```

Arguments

`x`	An "Aniso" object, from Aniso_fit().
`ref.loc`	Vector of length 2; the reference location.
`all.pred.locs`	A matrix of all prediction locations.
`grid`	Logical; indicates if the `all.pred.locs` are on a rectangular grid (`TRUE`) or not (`FALSE`).
`...`	Arguments passed to `plot` functions.

Value

A plot of either the estimated ellipses or estimated correlation is printed.

Examples

```
## Not run:
plot.Aniso( Aniso.object )

## End(Not run)
```

`plot.NSconvo` *Plot from the nonstationary model.*

Description

This function plots either the estimated anisotropy ellipses for each of the mixture component locations or the estimated correlation between a reference point and all other prediction locations.

Usage

```
## S3 method for class 'NSconvo'
plot(
  x,
  plot.ellipses = TRUE,
  fit.radius = NULL,
  aniso.mat = NULL,
  true.mc = NULL,
  ref.loc = NULL,
  all.pred.locs = NULL,
  grid = TRUE,
  true.col = 1,
  aniso.col = 4,
  ns.col = 2,
  plot.mc.locs = TRUE,
  ...
)
```

Arguments

`x`	A "NSconvo" object, from NSconvo_fit().
`plot.ellipses`	Logical; indicates whether the estimated ellipses should be plotted (`TRUE`) or estiamted correlations (`FALSE`).
`fit.radius`	Scalar; defines the fit radius used for the local likelihood estimation.
`aniso.mat`	2 x 2 matrix; contains the estimated anisotropy ellipse from the stationary model (for comparison).
`true.mc`	The true mixture component ellipses, if known.
`ref.loc`	Vector of length 2; the reference location.
`all.pred.locs`	A matrix of all prediction locations.
`grid`	Logical; indicates if the `all.pred.locs` are on a rectangular grid (`TRUE`) or not (`FALSE`).
`true.col`	Color value for the true mixture component ellipses (if plotted).
`aniso.col`	Color value for the anisotropy ellipse (if plotted).
`ns.col`	Color value for the mixture component ellipses.
`plot.mc.locs`	Logical; indicates whether the mixture component locations should be plotted (`TRUE`) or not (`FALSE`).
`...`	Other options passed to `plot`.

Value

A plot of either the estimated ellipses or estimated correlation is printed.

Examples

```
## Not run:
plot.NSconvo( NSconvo.object )

## End(Not run)
```

---

predict.Aniso *Obtain predictions at unobserved locations for the stationary spatial model.*

Description

predict.Aniso calculates the kriging predictor and corresponding standard errors at unmonitored sites.

Usage

```
## S3 method for class 'Aniso'
predict(
  object,
  pred.coords,
  pred.covariates = NULL,
  pred.fixed.nugg2.var = NULL,
  ...
)
```

Arguments

object An "Aniso" object, from Aniso_fit.

pred.coords Matrix of locations where predictions are required.

pred.covariates
Matrix of covariates for the prediction locations, NOT including an intercept. The number of columns for this matrix must match the design matrix from mean.model in NSconvo_fit. Defaults to an intercept only.

pred.fixed.nugg2.var
An optional vector or matrix describing the the variance/covariance a fixed second nugget term (corresponds to fixed.nugg2.var in Aniso_fit; often useful if conducting prediction for held-out data). Defaults to zero.

... additional arguments affecting the predictions produced.

Value

A list with the following components:

pred.means Vector of the kriging predictor, for each location in pred.coords.

pred.SDs Vector of the kriging standard errors, for each location in pred.coords.

Examples

```
## Not run:
pred.S <- predict( Aniso.obj,
pred.coords = cbind(runif(300),runif(300)) )

## End(Not run)
```

---

predict.NSconvo *Obtain predictions at unobserved locations for the nonstationary spatial model.*

Description

predict.NSconvo calculates the kriging predictor and corresponding standard errors at unmonitored sites.

Usage

```
## S3 method for class 'NSconvo'
predict(
  object,
  pred.coords,
  pred.covariates = NULL,
  pred.fixed.nugg2.var = NULL,
  ...
)
```

Arguments

object A "NSconvo" object, from NSconvo_fit.

pred.coords Matrix of locations where predictions are required.

pred.covariates Matrix of covariates for the prediction locations, NOT including an intercept. The number of columns for this matrix must match the design matrix from mean.model in NSconvo_fit. Defaults to an intercept only.

pred.fixed.nugg2.var An optional vector or matrix describing the the variance/covariance a fixed second nugget term (corresponds to fixed.nugg2.var in NSconvo_fit; often useful if conducting prediction for held-out data). Defaults to zero.

... additional arguments affecting the predictions produced.

Value

A list with the following components:

- `pred.means` Vector of the kriging predictor, for each location in `pred.coords`.
- `pred.SDs` Vector of the kriging standard errors, for each location in `pred.coords`.

Examples

```
## Not run:
pred.NS <- predict( NSconvo.obj,
pred.coords = matrix(c(1,1), ncol=2),
pred.covariates = matrix(c(1,1), ncol=2) )

## End(Not run)
```

`simdata` *Simulated nonstationary dataset*

Description

A data set containing the necessary components to fit the nonstationary spatial model, simulated from the true model.

Usage

```
simdata
```

Format

A list with the following objects:

- **sim.locations** A matrix of longitude/latitude coordinates of the simulated locations.
- **mc.locations** A matrix of longitude/latitude coordinates of the mixture component locations.
- **mc.kernel** A three-dimensional array, containing the true 2 x 2 kernel covariance matrices for each mixture component location.
- **kernel.ellipses** A three-dimensional array, containing the true 2 x 2 kernel covariance matrices for each simulated location.
- **sim.data** A matrix of the simulated data; each of the ten columns correspond to an independent and identically distribured replicate.
- **lambda.w** Scalar; the value of the tuning parameter used in the weight function.
- **holdout.index** Vector; indicates which of the simulated locations should be used in the hold-out sample.

`summary.Aniso` *Summarize the stationary model fit.*

Description

`summary.Aniso` prints relevant output from the model fitting procedure.

Usage

```
## S3 method for class 'Aniso'
summary(object, ...)
```

Arguments

`object`	An "Aniso" object, from `Aniso_fit`.
`...`	additional arguments affecting the summary produced.

Value

Text containing the model fitting results.

Examples

```
## Not run:
summary.Aniso( Aniso.object )

## End(Not run)
```

`summary.NSconvo` *Summarize the nonstationary model fit.*

Description

`summary.NSconvo` prints relevant output from the model fitting procedure.

Usage

```
## S3 method for class 'NSconvo'
summary(object, ...)
```

Arguments

`object`	A "NSconvo" object, from `NSconvo_fit`.
`...`	additional arguments affecting the summary produced.

Value

Text containing the model fitting results.

Examples

```
## Not run:
summary.NSconvo( NSconvo.object )

## End(Not run)
```

US.mc.grids *Mixture component grids for the western United States*

Description

A list of two mixture component grids for fitting the nonstationary model to the western United States precipitation data.

Usage

```
US.mc.grids
```

Format

A list with two elements:

Element 1 Coarse mixture component grid.

Element 2 Fine mixture component grid.

US.prediction.locs *Prediction locations for the western United States*

Description

A matrix with two columns containing a fine grid of locations for which to make a filled-in prediction map for the western United States.

Usage

```
US.prediction.locs
```

Format

A matrix with two columns:

Column 1 Longitude of the prediction grid.

Column 2 Latitude of the prediction grid.

`USprecip97` *Annual precipitation measurements from the western United States, 1997*

Description

A data set containing the annual precipitation for 1270 locations in the western United States.

Usage

```
USprecip97
```

Format

A data frame with the following variables:

longitude Longitude of the monitoring site.

latitude Latitude of the monitoring site.

annual.ppt Annual precipitation for the monitoring site, in millimeters.

log.annual.ppt Annual precipitation for the monitoring site, in log millimeters.

Source

http://www.image.ucar.edu/GSP/Data/US.monthly.met/

Index