Package ‘SpatialGEV’

October 12, 2022

Title Fit Spatial Generalized Extreme Value Models

Version 1.0.0

Maintainer Meixi Chen <`meixi.chen@uwaterloo.ca`>

Description
Fit latent variable models with the GEV distribution as the data likelihood and the GEV parameters following latent Gaussian processes. The models in this package are built using the template model builder 'TMB' in R, which has the fast ability to integrate out the latent variables using Laplace approximation. This package allows the users to choose in the fit function which GEV parameter(s) is considered as a spatially varying random effect following a Gaussian process, so the users can fit spatial GEV models with different complexities to their dataset without having to write the models in 'TMB' by themselves. This package also offers methods to sample from both fixed and random effects posteriors as well as the posterior predictive distributions at different spatial locations. Methods for fitting this class of models are described in Chen, Ramezan, and Lysy (2021) <arXiv:2110.07051>.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports TMB (>= 1.7.16), mvtnorm, evd, stats

LinkingTo TMB

RoxygenNote 7.1.2

Suggests INLA, testthat, knitr, rmarkdown

Additional_repositories `https://inla.r-inla-download.org/R/stable/`

VignetteBuilder knitr

NeedsCompilation yes

Author Meixi Chen [aut, cre],
Martin Lysy [aut],
Reza Ramezan [ctb]

Repository CRAN

Date/Publication 2022-04-05 09:10:02 UTC

R topics documented:

`grid_location` *Grid the locations with fixed cell size*

Description

Grid the locations with fixed cell size

Usage

```
grid_location(
  lon,
  lat,
  sp.resolution = 2,
  lon.range = range(lon),
  lat.range = range(lat)
)
```

Arguments

`lon`	Numeric, `n` longitude values
`lat`	Numeric, `n` latitude values
`sp.resolution`	Numeric, must be a single value that indicates the minimal unit length of a grid cell.
`lon.range`	Optional vector that indicates the range of `lon`. Default is `range(lon)`.
`lat.range`	Optional vector that indicates the range of `lat`. Default is `range(lat)`.

Details

The longitude and latitude of each grid cell are the coordinate of the cell center. For example, if `sp.resolution=1`, then `cell_lon=55.5` and `cell_lat=22.5` correspond to the square whose left boundary is 55, right boundary is 56, upper boundary is 23, and lower boundary is 22.

Value

An `n x 3` data frame containing three variables: `cell_ind` corresponds to unique id for each grid cell, `cell_lon` is the longitude of the grid cell, `cell_lat` is the latitude of the grid cell. Since the output data frame retains the order of the input coordinates, the original coordinate dataset and the output have can be linked one-to-one by the row index.

Examples

```
longitude <- runif(20, -90, 80)
latitude <- runif(20, 40, 60)
grid_locs <- grid_location(longitude, latitude, sp.resolution=0.5)
cbind(longitude, latitude, grid_locs)
```

`kernel_exp` *Exponential covariance function*

Description

Exponential covariance function

Usage

```
kernel_exp(x, sigma, ell, X1 = NULL, X2 = NULL)
```

Arguments

`x`	Distance measure.
`sigma`	The amplitude parameter (scalar) with the constraint of `sigma` > `0`
`ell`	The smoothness parameter (scalar) with the constraint of `ell` > `0`.
`X1`	A `n1 x 2` matrix containing the coordinates of location set 1. If `x` is not provided, `X1` and `X2` should be provided for calculating their distance.
`X2`	A `n2 x 2` coordinate matrix.

Details

Let x = dist(x_i, x_j).

```
cov(i,j) = sigma*exp(-x/ell)
```

Value

A matrix or a scalar of exponential covariance depending on the type of x or whether X1 and X2 are used instead.

Examples

```
X1 <- cbind(runif(10, 1, 10), runif(10, 10, 20))
X2 <- cbind(runif(5, 1, 10), runif(5, 10, 20))

kernel_exp(sigma=2, ell=1, X1=X1, X2=X2)

kernel_exp(as.matrix(stats::dist(X1)), sigma=2, ell=1)
```

---

kernel_matern *Matern covariance function*

Description

Matern covariance function

Usage

```
kernel_matern(x, sigma, kappa, nu = 1, X1 = NULL, X2 = NULL)
```

Arguments

x	Distance measure.
sigma	Positive parameter. (This is in fact sigma^2)
kappa	Positive parameter.
nu	Range parameter default to 1.
X1	A n1 x 2 matrix containing the coordinates of location set 1. If x is not provided, X1 and X2 should be provided for calculating their distance.
X2	A n2 x 2 coordinate matrix.

Details

Let x = dist(x_i, x_j).

```
cov(i,j) = sigma * 2^(1-nu)/gamma(nu) * (kappa*x)^nu * K_v(kappa*x)
```

Note that when nu=0.5, the Matern kernel corresponds to the absolute exponential kernel.

Value

A matrix or a scalar of Matern covariance depending on the type of x or whether X1 and X2 are used instead.

Examples

```
X1 <- cbind(runif(10, 1, 10), runif(10, 10, 20))
X2 <- cbind(runif(5, 1, 10), runif(5, 10, 20))

kernel_matern(sigma=2, kappa=1, X1=X1, X2=X2)

kernel_matern(as.matrix(stats::dist(X1)), sigma=2, kappa=1)
```

`matern_pc_prior` *Helper funcion to specify a Penalized Complexity (PC) prior on the Matern hyperparameters*

Description

Helper funcion to specify a Penalized Complexity (PC) prior on the Matern hyperparameters

Usage

```
matern_pc_prior(rho_0, p_rho, sig_0, p_sig)
```

Arguments

`rho_0`	Hyperparameter for PC prior on the range parameter. Must be positive. See details.
`p_rho`	Hyperparameter for PC prior on the range parameter. Must be between 0 and 1. See details.
`sig_0`	Hyperparameter for PC prior on the range parameter. Must be positive. See details.
`p_sig`	Hyperparameter for PC prior on the range parameter. Must be between 0 and 1. See details.

Details

The joint prior on `rho` and `sig` achieves

`P(rho < rho_0) = p_rho,`

and

`P(sig > sig_0) = p_sig,`

where `rho` = `sqrt(8*nu)/kappa` and `sig` = `sqrt(sigma)`.

Value

A list to provide to the `matern_pc_prior` argument of `spatialGEV_fit`.

References

Simpson, D., Rue, H., Riebler, A., Martins, T. G., & Sørbye, S. H. (2017). Penalising model component complexity: A principled, practical approach to construct priors. Statistical Science.

Examples

```
y <- simulatedData2$y
locs <- simulatedData2$locs
n_loc <- nrow(locs)
fit <- spatialGEV_fit(y = y, locs = locs, random = "abs",
                      init_param = list(a = rep(0, n_loc),
                                        log_b = rep(0, n_loc),
                                        s = rep(-2, n_loc),
                                        beta_a = 0,
                                        beta_b = 0,
                                        beta_s = -2,
                                        log_sigma_a = 0,
                                        log_kappa_a = 0,
                                        log_sigma_b = 0,
                                        log_kappa_b = 0,
                                        log_sigma_s = 0,
                                        log_kappa_s = 0
                                        ),
                       reparam_s = "positive",
                       kernel = "matern",
                       beta_prior = list(beta_a=c(0,100), beta_b=c(0,10),
                                         beta_s=c(0,10)),
                       matern_pc_prior = list(
                                               matern_a=matern_pc_prior(1e5,0.95,5,0.1),
                                               matern_b=matern_pc_prior(1e5,0.95,3,0.1),
                                               matern_s=matern_pc_prior(1e2,0.95,1,0.1)
                                               ))
```

ONsnow *Monthly total snowfall in Ontario, Canada from 1987 to 2021.*

Description

A dataset containing the monthly total snowfall (in cm) in Ontario, Canada from 1987 to 2021.

Usage

```
ONsnow
```

Format

A data frame with 63945 rows and 7 variables with each row corresponding to a monthly record at a weather location:

LATITUDE	Numeric. Latitude of the weather station
LONGITUDE	Numeric. Longitude of the weather station
STATION_NAME	Character. Name of the weather station
CLIMATE_IDENTIFIER	Character. Unique id of each station
LOCAL_YEAR	Integer from 1987 to 2021. Year of the record
LOCAL_MONTH	Integer from 1 to 12. Month of the record
TOTAL_SNOWFALL	Positive number. Total monthly snowfall at a station in cm

Source

`https://climate-change.canada.ca/climate-data/#/monthly-climate-summaries`

`print.spatialGEVfit` *Print method for spatialGEVfit*

Description

Print method for spatialGEVfit

Usage

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

Arguments

`x`	Model object of class `spatialGEVfit` returned by `spatialGEV_fit`.
`...`	More arguments for `print`.

Value

Information about the fitted model containing number of fixed/random effects, fitting time, convergence information, etc.

`print.spatialGEVpred` *Print method for spatialGEVpred*

Description

Print method for spatialGEVpred

Usage

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

Arguments

`x`	Object of class `spatialGEVpred` returned by `spatialGEV_predict`.
`...`	Additional arguments for `print`.

Value

Information about the prediction.

`print.spatialGEVsam` *Print method for spatialGEVsam*

Description

Print method for spatialGEVsam

Usage

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

Arguments

`x`	Object of class `spatialGEVsam` returned by `spatialGEV_sample`.
`...`	Additional arguments for `print`.

Value

Information about the object including dimension and direction to use `summary` on the object.

r_nll *Calculate the negative marginal loglikelihood of the GEV-GP model.*

Description

Calculate the negative marginal loglikelihood of the GEV-GP model.

Usage

```
r_nll(
  y,
  dd,
  a,
  log_b,
  s,
  hyperparam_a,
  hyperparam_b,
  hyperparam_s,
  kernel = "exp",
  beta_a = NULL,
  beta_b = NULL,
  beta_s = NULL,
  X_a = NULL,
  X_b = NULL,
  X_s = NULL,
  f_s = function(x) {     x },
  ...
)
```

Arguments

`y`	List of `n` locations each with `n_obs[i]` independent GEV realizations.
`dd`	An `n x n` distance matrix.
`a`	Vector of `n` location paramter
`log_b`	A numeric value or a vector of `n` log-transformed scale parameters if considered as a random effect.
`s`	A numeric value or a vector of `n` shape parameters
`hyperparam_a`	A vector of hyperparameters for a. See details.
`hyperparam_b`	A vector of hyperparameters for b. Must be provided if `log_b` is a vector. See details.
`hyperparam_s`	A vector of hyperparameters for f(s), where f() is a transformation function for `s` specifided using the `f_s` argument. Must be provided if `s` is a vector.
`kernel`	"exp" or "matern". Kernel function used to compute the covariance matrix for spatial random effects. Default is "exp".

`beta_a`	Numeric. Coefficients for mean of GP(a).
`beta_b`	Numeric. Coefficients for mean of GP(log_b).
`beta_s`	Numeric. Coefficients for mean of GP(s).
`X_a`	Design matrix for a. If not provided, this will a `n_loc x 1` column matrix of 1s.
`X_b`	Design matrix for log(b). If not provided and logb is a random effect, this will a `n_loc x 1` column matrix of 1s.
`X_s`	Design matrix for s. If not provided, this will a `n_loc x 1` column matrix of 1s.
`f_s`	A function f() used to transform s such that f(s) ~ GP(X_s*beta_s, Sigma(hyperparam_s)). Default is identitfy function: `function(x){x}`.
`...`	Additional arguments to pass to the kernel function, e.g. `nu` for the matern.

Details

This function is used to test if TMB and R output the same negative loglikelihood. If `kernel="exp`, `hyperparam_a/b/s` should be `c(sigma_a/b/s, ell_a/b/s)`, where `sigma` is the amplitude hyperparameter and `ell` is the smoothness hyperparameter for the exponential kernel. If `kernel="matern`, `hyperparam_a/b/s` should be `c(sigma_a/b, kappa_a/b/s)`, where `sigma` and `kappa` are hyperparameters for the Matern kernel. If only `a` is a spatial random effect and `b` is fixed, only `hyperparam_a` needs to be provided.

This function is used as the ground truth for testing hpp model likelihood.

Value

Scalar value of the negative marginal loglikelihood:

```
-logL(Data; spatial_random_effects, fixed_hyperparameters)
```

Examples

```
library(SpatialGEV)
a <- simulatedData$a
logb <- simulatedData$logb
logs <- simulatedData$logs
s <- exp(logs)
y <- simulatedData$y
locs <- simulatedData$locs
dd <- as.matrix(stats::dist(locs))
log_sigma_a <- -1; log_ell_a <- 5
log_sigma_b <- -2; log_ell_b <- 10
beta_a <- mean(a); beta_b <- mean(logb)
# Negative marginal log-likelihood produced in R using the exponential kernel
nll_r <- r_nll(y, dd, a=a, log_b=logb, s=s,
               hyperparam_a=c(exp(log_sigma_a), exp(log_ell_a)),
               hyperparam_b=c(exp(log_sigma_b), exp(log_ell_b)),
               kernel="exp", beta_a=beta_a, beta_b=beta_b)
# Negative marg loglik produced by TMB template
init_param <- list(beta_a=beta_a, beta_b=beta_b,
                   a=a, log_b=logb, s=log(s),
                       log_sigma_a=log_sigma_a,
                       log_ell_a=log_ell_a,
                       log_sigma_b=log_sigma_b,
                       log_ell_b=log_ell_b)
adfun <- spatialGEV_fit(y, locs, random="ab",
                        init_param=init_param,
                        reparam_s="positive",
                        kernel="exp",
                        adfun_only=TRUE,
                        ignore_random=TRUE,
                        silent=TRUE)
nll_tmb <- adfun$fn(unlist(init_param))
nll_r - nll_tmb
```

`simulatedData` *Simulated dataset 1*

Description

A list of data used for package testing and demos. Both `a` and `logb` are simulated on smooth deterministic surfaces.

Usage

```
simulatedData
```

Format

A list containing the simulation parameters and simulated data on a 20x20 grid:

locs A 400x2 matrix. First column contains longitudes and second contains latitudes

a A length 400 vector. GEV location parameters

logb A length 400 vector. Log-transformed GEV scale parameters

logs A scalar. Log-transformed GEV shape parameter shared across space

y A length 400 list of vectors which are observations simulated at each location

---

`simulatedData2` *Simulated dataset 2*

Description

A list of data used for package testing and demos. `a`, `logb`, `logs` are simulated from respective Gaussian random fields and thus are nonsmooth.

Usage

```
simulatedData2
```

Format

A list containing the simulation parameters and simulated data on a 20x20 grid:

locs A 400x2 matrix. First column contains longitudes and second contains latitudes

a A length 400 vector. GEV location parameters

logb A length 400 vector. Log-transformed GEV scale parameters

logs A length 400 vector. Log-transformed GEV shape parameters

y A length 400 list of vectors which are observations simulated at each location

---

`sim_cond_normal` *Create a helper function to simulate from the conditional normal distribution of new data given old data*

Description

Create a helper function to simulate from the conditional normal distribution of new data given old data

Usage

```
sim_cond_normal(joint.mean, a, locs_new, locs_obs, kernel, ...)
```

Arguments

`joint.mean`	The length `n` mean vector of the MVN distribution. By default mu1 is the first `m` elements of `joint.mean`
`a`	A vector of length `n-m`, the values of mu2 to condition on
`locs_new`	A matrix containing the coordiantes of new locations
`locs_obs`	A matrix containing the coordinates of observed locations
`kernel`	A function (kernel function) that returns a matrix containing the similarity between the two arguments.
`...`	Hyperparameters to pass to the kernel function.

Details

This serves as a helper function for `spatialGEV_predict`. The notations are consistent to the notations on the MVN wikipedia page

Value

A function that takes in one argument `n` as the number of samples to draw from the condition normal distribution of `locs_new` given `locs_obs`: either from `rmvnorm` for MVN or `rnorm` for univariate normal. The old and new data are assumed to follow a joint multivariate normal distribution.

spatialGEV_fit *Fit a GEV-GP model.*

Description

Fit a GEV-GP model.

Usage

```
spatialGEV_fit(
  y,
  locs,
  random,
  init_param,
  reparam_s,
  kernel = "exp",
  X_a = NULL,
  X_b = NULL,
  X_s = NULL,
  nu = 1,
  s_prior = NULL,
  beta_prior = NULL,
  matern_pc_prior = NULL,
  sp_thres = -1,
  adfun_only = FALSE,
  ignore_random = FALSE,
  silent = FALSE,
  mesh_extra_init = list(a = 0, log_b = -1, s = 0.001),
  ...
)
```

Arguments

`y`	List of `n` locations each with `n_obs[i]` independent GEV realizations.
`locs`	`n x 2` matrix of longitude and latitude of the corresponding response values.
`random`	Either "a", "ab", or "abs", where `a` indicates the location parameter, `b` indicates the scale parameter, `s` indicates the shape parameter. This tells the model which GEV parameters are considered as random effects.
`init_param`	A list of initial parameters. See details.
`reparam_s`	A flag indicating whether the shape parameter is "zero", "unconstrained", constrained to be "negative", or constrained to be "positive". If model "abs" is used, `reparam_s` cannot be zero. See details.
`kernel`	Kernel function for spatial random effects covariance matrix. Can be "exp" (exponential kernel), "matern" (Matern kernel), or "spde" (Matern kernel with SPDE approximation described in Lindgren el al. 2011). To use the SPDE approximation, the user must first install the INLA R package.

`X_a`	`n x r` design matrix for a, where `r-1` is the number of covariates. If not provided, a `n x 1` column matrix of 1s is used.
`X_b`	`n x r` design matrix for log(b). Does not need to be provided if b is fixed.
`X_s`	`n x r` design matrix for g(s), where g() is a transformation function of `s`. Does not need to be provided if s is fixed.
`nu`	Hyperparameter of the Matern kernel. Default is 1.
`s_prior`	Optional. A length 2 vector where the first element is the mean of the normal prior on s or log(s) and the second is the standard deviation. Default is NULL, meaning a uniform prior is put on s if s is fixed, or a GP prior is applied if s is a random effect.
`beta_prior`	Optional named list that specifies normal priors on the GP mean function coefficients `betas`. Each element of the list should be a named length 2 vector in which the first element is mean and second element is sd. E.g. `beta_prior=list(beta_a=c(0,100), beta_b=c(0,10), beta_s=c(-2,5))`. Default is NULL, which means imposing a noninformative uniform flat prior.
`matern_pc_prior`	Optional named list that specifies Penalized complexity priors on the GP Matern covariance hyperparameters `sig` and `rho`, where `sig = sqrt(sigma)` and `rho = sqrt(8*nu)/kappa`. Names must be `matern_a`, `matern_b`, or `matern_s`. E.g. `matern_pc_prior=list(matern_s=matern_pc_prior(100, 0.9, 2, 0.1))`. Default is NULL, which means a flat prior. See `?matern_pc_prior` for more details.
`sp_thres`	Optional. Thresholding value to create sparse covariance matrix. Any distance value greater than or equal to `sp_thres` will be set to 0. Default is -1, which means not using sparse matrix. Caution: hard thresholding the covariance matrix often results in bad convergence.
`adfun_only`	Only output the ADfun constructed using TMB? If TRUE, model fitting is not performed and only a TMB tamplate `adfun` is returned (along with the created mesh if kernel is "spde"). This can be used when the user would like to use a different optimizer other than the default `nlminb`. E.g., call `optim(adfun$par, adfun$fn, adfun$gr)` for optimization.
`ignore_random`	Ignore random effect? If TRUE, spatial random effects are not integrated out in the model. This can be helpful for checking the marginal likelihood.
`silent`	Do not show tracing information?
`mesh_extra_init`	A named list of scalars. Used when the SPDE kernel is used. The list provides the initial values for a, log(b), and s on the extra triangles created in the mesh. The default is `list(a=1, log_b=0, s=0.001)`.
`...`	Arguments to pass to `INLA::inla.mesh.2d()`. See details `?inla.mesh.2d()` and Section 2.1 of Lindgren & Rue (2015) JSS paper. This is used specifically for when `kernel="spde"`, in which case a mesh needs to be constructed on the spatial domain. When no arguments are passed to `inla.mesh.2d()`, a default argument is `max.edge=2`, which simply specifies the largest allowed triangle edge length. It is strongly suggested that the user should specify these arguments if they would like to use the SPDE kernel. Please make sure INLA package is installed before using the SPDE approximation.

Details

This function adopts Laplace approximation using TMB model to integrate out the random effects.

The random effects are assumed to follow Gaussian processes with mean 0 and covariance matrix defined by the chosen kernel function. E.g., using the exponential kernel function:

```
cov(i,j) = sigma*exp(-|x_i - x_j|/ell)
```

When specifying the initial parameters to be passed to `init_param`, care must be taken to count the number of parameters. Described below is how to specify `init_param` under different settings of `random` and `kernel`. Note that the order of the parameters must match the descriptions below (initial values specified below such as 0 and 1 are only examples).

- random = "a", kernel = "exp": `a` should be a vector and the rest are scalars. `log_sigma_a` and `log_ell_a` are hyperparameters in the exponential kernel for the Gaussian process describing the spatial variation of `a`.

```
init_param = list(a = rep(1,n_locations), log_b = 0, s = 1,
                  beta_a = rep(0, n_covariates),
                  log_sigma_a = 0, log_ell_a = 0)
```

Note that even if `reparam_s=="zero"`, an initial value for `s` still must be provided, even though in this case the value does not matter anymore.

- random = "ab", kernel = "exp": When `b` is considered a random effect, its corresponding GP hyperparameters `log_sigma_b` and `log_ell_b` need to be specified.

```
init_param = list(a = rep(1,n_locations),
                  log_b = rep(0,n_locations), s=1,
                  beta_a = rep(0, n_covariates), beta_b = rep(0, n_covariates),
                  log_sigma_a = 0,log_ell_a = 0,
                  log_sigma_b = 0,log_ell_b = 0).
```

- random = "abs", kernel = "exp":

```
init_param = list(a = rep(1,n_locations),
                  log_b = rep(0,n_locations),
                  s = rep(0,n_locations),
                  beta_a = rep(0, n_covariates),
                  beta_b = rep(0, n_covariates),
                  beta_s = rep(0, n_covariates),
                  log_sigma_a = 0,log_ell_a = 0,
                  log_sigma_b = 0,log_ell_b = 0).
                  log_sigma_s = 0,log_ell_s = 0).
```

- random = "abs", kernel = "matern" or "spde": When the Matern or SPDE kernel is used, hyperparameters for the GP kernel are `log_sigma_a/b/s` and `log_kappa_a/b/s` for each spatial random effect.

```
init_param = list(a = rep(1,n_locations),
                  log_b = rep(0,n_locations),
                  s = rep(0,n_locations),
                  beta_a = rep(0, n_covariates),
                  beta_b = rep(0, n_covariates),
                  beta_s = rep(0, n_covariates),
                  log_sigma_a = 0,log_kappa_a = 0,
                  log_sigma_b = 0,log_kappa_b = 0).
                  log_sigma_s = 0,log_kappa_s = 0).
```

raparam_s allows the user to reparametrize the GEV shape parameter s. For example,

- if the data is believed to be right-skewed and lower bounded, this means s>0 and one should use reparam_s = "positive";
- if the data is believed to be left-skewed and upper bounded, this means s<0 and one should use reparam_s="negative".
- When reparam_s = "zero", the data likelihood is a Gumbel distribution. In this case the data has no upper nor lower bound. Finally, specify reparam_s = "unconstrained" if no sign constraint should be imposed on s.

Note that when reparam_s = "negative" or "postive", the initial value of s in init_param should be that of log(|s|).

When the SPDE kernel is used, a mesh on the spatial domain is created using INLA::inla.mesh.2d(), which extends the spatial domain by adding additional triangles in the mesh to avoid boundary effects in estimation. As a result, the number of a and b will be greater than the number of locations due to these additional triangles: each of them also has their own a and b values. Therefore, the fit function will return a vector meshidxloc to indicate the positions of the observed coordinates in the random effects vector.

Value

If adfun_only=TRUE, this function outputs a list returned by TMB::MakeADFun(). This list contains components par, fn, gr and can be passed to an R optimizer. If adfun_only=FALSE, this function outputs an object of class spatialGEVfit, a list

- An adfun object
- A fit object given by calling nlminb() on the adfun
- An object of class sdreport from TMB which contains the point estimates, standard error, and precision matrix for the fixed and random effects
- Other helpful information about the model: kernel, data coordinates matrix, and optionally the created mesh if ‘kernel="spde" (See details).

Examples

```
library(SpatialGEV)
a <- simulatedData$a
logb <- simulatedData$logb
```

```
logs <- simulatedData$logs
y <- simulatedData$y
locs <- simulatedData$locs
n_loc = nrow(locs)
# No covariates are included, only intercept is inlcuded.
fit <- spatialGEV_fit(y = y, locs = locs, random = "ab",
                      init_param = list(a = rep(0, n_loc),
                                        log_b = rep(0, n_loc),
                                        s = 0,
                                        beta_a = 0,
                                        beta_b = 0,
                                        log_sigma_a = 0,
                                        log_kappa_a = 0,
                                        log_sigma_b = 0,
                                        log_kappa_b = 0),
                      reparam_s = "positive",
                      kernel = "matern",
                      X_a = matrix(1, nrow=n_loc, ncol=1),
                      X_b = matrix(1, nrow=n_loc, ncol=1),
                      silent = TRUE)
print(fit)

# To use a different optimizer other than the default `nlminb()`, create
# an object ready to be passed to optimizer functions using `adfun_only=TRUE`
obj <- spatialGEV_fit(y = y, locs = locs, random = "ab",
                      init_param = list(a = rep(0, n_loc),
                                        log_b = rep(0, n_loc),
                                        s = 0,
                                        beta_a = 0,
                                        beta_b = 0,
                                        log_sigma_a = 0,
                                        log_kappa_a = 0,
                                        log_sigma_b = 0,
                                        log_kappa_b = 0),
                      reparam_s = "positive",
                      kernel = "matern",
                      X_a = matrix(1, nrow=n_loc, ncol=1),
                      X_b = matrix(1, nrow=n_loc, ncol=1),
                      adfun_only = TRUE)
fit <- optim(obj$par, obj$fn, obj$gr)


# Using the SPDE kernel (SPDE approximation to the Matern kernel)
# Make sure the INLA package is installed before using `kernel="spde"`
## Not run:
library(INLA)
y <- simulatedData2$y
locs <- simulatedData2$locs
n_loc <- nrow(locs)
fit_spde <- spatialGEV_fit(y = y, locs = locs, random = "abs",
                           init_param = list(a = rep(0, n_loc),
                                             log_b = rep(0, n_loc),
                                             s = rep(-2, n_loc),
                                             beta_a = 0,
                                             beta_b = 0,
                                             beta_s = -2,
                                             log_sigma_a = 0,
                                             log_kappa_a = 0,
                                             log_sigma_b = 0,
                                             log_kappa_b = 0,
                                             log_sigma_s = 0,
                                             log_kappa_s = 0
                                             ),
                           reparam_s = "positive",
                           kernel = "spde",
                           beta_prior = list(beta_a=c(0,100), beta_b=c(0,10),
                                             beta_s=c(0,10)),
                           matern_pc_prior = list(
                                               matern_a=matern_pc_prior(1e5,0.95,5,0.1),
                                               matern_b=matern_pc_prior(1e5,0.95,3,0.1),
                                                matern_s=matern_pc_prior(1e2,0.95,1,0.1)
                                                 ))
plot(fit_spde$mesh) # Plot the mesh
points(locs[,1], locs[,2], col="red", pch=16) # Plot the locations

## End(Not run)
```

---

spatialGEV_predict — *Draw from the posterior predictive distributions at new locations based on a fitted GEV-GP model*

Description

Draw from the posterior predictive distributions at new locations based on a fitted GEV-GP model

Usage

```
spatialGEV_predict(
  model,
  locs_new,
  n_draw,
  X_a_new = NULL,
  X_b_new = NULL,
  X_s_new = NULL,
  parameter_draws = NULL
)
```

Arguments

`model`	A fitted spatial GEV model object of class `spatialGEVfit`
`locs_new`	A `n_test x 2` matrix containing the coordinates of the new locations

`n_draw`	Number of draws from the posterior predictive distribution
`X_a_new`	`n_test x r1` design matrix for a at the new locations. If not provided, the default is a column matrix of all 1s.
`X_b_new`	`n_test x r2` design matrix for log(b) at the new locations
`X_s_new`	`n_test x r2` design matrix for (possibly transformed) s at the new locations
`parameter_draws`	Optional. A `n_draw x n_parameter` matrix. If `spatialGEV_sample()` has already been called, the output matrix of parameter draws can be supplied here to avoid doing sampling of parameters again. Make sure the number of rows of `parameter_draws` is the same as `n_draw`.

Value

An object of class `spatialGEVpred`, which is a list of the following components:

- An `n_draw x n_test` matrix `pred_y_draws` containing the draws from the posterior predictive distributions at `n_test` new locations
- An `n_test x 2` matrix `locs_new` containing the coordinates of the test data
- An `n_train x 2` matrix `locs_obs` containing the coordinates of the observed data

Examples

```
set.seed(123)
library(SpatialGEV)
a <- simulatedData$a
logb <- simulatedData$logb
logs <- simulatedData$logs
y <- simulatedData$y
locs <- simulatedData$locs
n_loc <- nrow(locs)
n_test <- 20
test_ind <- sample(1:n_loc, n_test)

# Obtain coordinate matrices and data lists
locs_test <- locs[test_ind,]
y_test <- y[test_ind]
locs_train <- locs[-test_ind,]
y_train <- y[-test_ind]

# Fit the GEV-GP model to the training set
train_fit <- spatialGEV_fit(y = y_train, locs = locs_train, random = "ab",
                            init_param = list(beta_a = mean(a),
                                              beta_b = mean(logb),
                                              a = rep(0, n_loc-n_test),
               log_b = rep(0, n_loc-n_test),
s = 0,
log_sigma_a = 1,
                                              log_kappa_a = -2,
log_sigma_b = 1,
                                             log_kappa_b = -2),
               reparam_s = "positive",
       kernel = "matern",
       silent = TRUE)
pred <- spatialGEV_predict(model = train_fit, locs_new = locs_test, n_draw = 100)
summary(pred)
```

---

spatialGEV_sample *Get posterior parameter draws from a fitted GEV-GP model.*

Description

Get posterior parameter draws from a fitted GEV-GP model.

Usage

```
spatialGEV_sample(model, n_draw, observation = FALSE, loc_ind = NULL)
```

Arguments

`model`	A fitted spatial GEV model object of class `spatialGEVfit`
`n_draw`	Number of draws from the posterior distribution
`observation`	whether to draw from the posterior distribution of the GEV observation?
`loc_ind`	A vector of location indices to sample from. Default is all locations.

Value

An object of class `spatialGEVsam`, which is a list of matrices containing the joint posterior draws of the parameters and optionally the GEV observations.

Examples

```
library(SpatialGEV)
a <- simulatedData$a
logb <- simulatedData$logb
logs <- simulatedData$logs
y <- simulatedData$y
locs <- simulatedData$locs
n_loc <- nrow(locs)
beta_a <- mean(a); beta_b <- mean(logb)
fit <- spatialGEV_fit(y = y, locs = locs, random = "ab",
                      init_param = list(beta_a = beta_a,
                                        beta_b = beta_b,
                                        a = rep(0, n_loc),
                                        log_b = rep(0, n_loc),
                                        s = 0,
                                        log_sigma_a = 0,
                                        log_kappa_a = 0,
                                        log_sigma_b = 0,
                                        log_kappa_b = 0),
                      reparam_s = "positive",
                      kernel = "matern",
                      silent = TRUE)
sam <- spatialGEV_sample(model = fit, n_draw = 100,
                         observation = TRUE, loc_ind=1:10)
print(sam)
summary(sam)
```

summary.spatialGEVpred

Summary method for spatialGEVpred

Description

Summary method for spatialGEVpred

Usage

```
## S3 method for class 'spatialGEVpred'
summary(object, q = c(0.025, 0.25, 0.5, 0.75, 0.975), ...)
```

Arguments

`object`	Object of class `spatialGEVpred` returned by `spatialGEV_predict`.
`q`	A vector of quantile values used to summarize the samples. Default is `c(0.025, 0.25, 0.5, 0.75, 0.975)`.
`...`	Additional arguments for `summary`.

Value

Summary statistics of the posterior predictive samples.

summary.spatialGEVsam *Summary method for spatialGEVsam*

Description

Summary method for spatialGEVsam

Usage

```
## S3 method for class 'spatialGEVsam'
summary(object, q = c(0.025, 0.25, 0.5, 0.75, 0.975), ...)
```

Arguments

`object`	Object of class `spatialGEVsam` returned by `spatialGEV_sample`.
`q`	A vector of quantile values used to summarize the samples. Default is `c(0.025, 0.25, 0.5, 0.75, 0.975)`.
`...`	Additional arguments for `summary`. Not used.

Value

Summary statistics of the posterior samples.

Index