# Package: geostatsp (via r-universe)

October 11, 2026

**Type** Package

**Title** Geostatistical Modelling with Likelihood and Bayes

**Version** 2.2.0

**Date** 2026-09-08

**Depends** Matrix, terra, R (>= 3.5.0)

**Imports** abind, numDeriv, methods, stats

**Suggests** parallel, mapmisc, pracma, knitr

**Enhances** ellipse, RandomFields, INLA, diseasemapping, geoR, mvtnorm

**LinkingTo** Matrix (>= 1.6-2)

**Additional_repositories** https://inla.r-inla-download.org/R/testing

**Description** Geostatistical modelling facilities using 'SpatRaster' and 'SpatVector' objects are provided. Non-Gaussian models are fit using 'INLA', and Gaussian geostatistical models use Maximum Likelihood Estimation. For details see Brown (2015) <doi:10.18637/jss.v063.i12>. The 'RandomFields' package is available at <https://web.archive.org/web/20250719184025/https://www.wim.uni-mannheim.de/schlather/publications/software> and <https://github.com/cran/RandomFields>.

**License** GPL (>= 2)

**Encoding** UTF-8

**URL** https://diseasemapping.r-forge.r-project.org/, https://r-forge.r-project.org/projects/diseasemapping/

**BugReports** https://r-forge.r-project.org/projects/diseasemapping/

**NeedsCompilation** yes

**VignetteBuilder** knitr

**Author** Patrick Brown [aut, cre, cph]

**Maintainer** Patrick Brown <patrick.brown@utoronto.ca>

**Repository** https://cran.r-universe.dev

**Date/Publication** 2026-09-10 16:50:06 UTC

**RemoteUrl** https://github.com/cran/geostatsp

**RemoteRef** HEAD

**RemoteSha** d0b17d960d6f73c92e731bb68413ac7844bec867

# Contents

conditionalGmrf . . . 2
excProb . . . 3
gambiaUTM . . . 5
geostatData . . . 6
glgm-methods . . . 7
inla.models . . . 11
inlaAvailable . . . 11
krigeLgm . . . 12
lgm-methods . . . 14
likfitLgm . . . 18
loaloa . . . 22
matern . . . 24
maternGmrfPrec . . . 27
murder . . . 30
pcPriorRange . . . 31
postExp . . . 32
profLlgm . . . 33
RFsimulate . . . 35
rongelapUTM . . . 38
simLgcp . . . 39
spatialRoc . . . 40
squareRaster-methods . . . 41
stackRasterList . . . 42
swissRain . . . 43
swissRainR . . . 45
variog . . . 45
wheat . . . 46

**Index** **47**

---

`conditionalGmrf` *Conditional distribution of GMRF*

---

## Description

Distribution of Gaussian Markov Random Field conditional on data observed with noise on the same grid.

## Usage

```
conditionalGmrf(param, Yvec, Xmat, NN,
 template = NULL, mc.cores = 1,
 cellsPerLoop = 10, ...)
```

## Arguments

| | |
|---|---|
| param | vector of named parameters |
| Yvec | vector of observed data, or matrix with each column being a realisation. |
| Xmat | Matrix of covariates. |
| NN | nearest neighbour matrix |
| template | Raster on which the GMRF is defined |
| mc.cores | passed to mclapply |
| cellsPerLoop | number of cells to compute simultaneously. Larger values consume more memory but result in faster computation. |
| ... | additional arguments passed to maternGmrfPrec |

## Value

Raster image with layers containing conditional mean and standard deviation.

## Author(s)

Patrick Brown

## See Also

maternGmrfPrec, lgm

---

# excProb *Exceedance probabilities*

---

## Description

Calculate exceedance probabilities pr(X > threshold) from a fitted geostatistical model.

## Usage

```
excProb(x, threshold=0, random=FALSE, template=NULL, templateIdCol=NULL,
nuggetInPrediction=TRUE)
```

## Arguments

| | |
|---|---|
| x | Output from either the lgm or glgm functions, or a list of two-column matrices with columns named x and y containing the posterior distributions of random effects, as produced by inla. |
| threshold | the value which the exceedance probability is calculated with respect to. |
| random | Calculate exceedances for the random effects, rather than the predicted observations (including fixed effects). |
| template | A SpatRaster or SpatVector object which the results will be contained in. |
| templateIdCol | The data column in template corresponding to names of marginals |
| nuggetInPrediction | If TRUE, calculate exceedance probabilities of new observations by adding the nugget effect. Otherwise calculate probabilities for the latent process. Ignored if x is output from glgm. |

## Details

When x is the output from lgm, pr(Y>threshold) is calculated using the Gaussian distribution using the Kriging mean and conditional variance. When x is from the glgm function, the marginal posteriors are numerically integrated to obtain pr(X > threshold).

## Value

Either a vector of exceedance probabilities or an object of the same class as template.

## Examples

```
data('swissRain')
swissRain = unwrap(swissRain)
swissAltitude = unwrap(swissAltitude)
swissBorder = unwrap(swissBorder)
swissFit =  lgm("rain", swissRain, grid=30,
 boxcox=0.5,fixBoxcox=TRUE, covariates=swissAltitude)
swissExc = excProb(swissFit, 20)
mycol = c("green","yellow","orange","red")
mybreaks = c(0, 0.2, 0.8, 0.9, 1)
plot(swissBorder)
plot(swissExc, breaks=mybreaks, col=mycol,add=TRUE,legend=FALSE)
plot(swissBorder, add=TRUE)
legend("topleft",legend=mybreaks, col=c(NA,mycol))



if(inlaAvailable() ) {
  inlaSetThreads(2)

 swissRain$sqrtrain = sqrt(swissRain$rain)
 swissFit2 =  glgm(formula="sqrtrain",data=swissRain, grid=40,
 covariates=swissAltitude,family="gaussian")
 swissExc = excProb(swissFit2, threshold=sqrt(30))
```

```
 swissExc = excProb(swissFit2$inla$marginals.random$space, 0,
  template=swissFit2$raster)

}
```

---

gambiaUTM *Gambia data*

---

## Description

This data-set was used by Diggle, Moyeed, Rowlingson, and Thomson (2002) to demonstrate how the model-based geostatistics framework of Diggle et al. (1998) could be adapted to assess the source(s) of extrabinomial variation in the data and, in particular, whether this variation was spatially structured. The malaria prevalence data set consists of measurements of the presence of malarial parasites in blood samples obtained from children in 65 villages in the Gambia. Other child- and village-level indicators include age, bed net use, whether the bed net is treated, whether or not the village belonged to the primary health care structure, and a measure of 'greenness' using a vegetation index.

## Usage

```
data(gambiaUTM)
```

## Format

A `SpatVector` , with column `pos` being the binary response for a malaria diagnosis, as well as other child-level indicators such as `netuse` and `treated` being measures of bed net use and whether the nets were treated. The column `green` is a village-level measure of greenness. A UTM coordinate reference system is used, where coordinates are in metres.

## Source

https://web.archive.org/web/20240110054727/http://www.leg.ufpr.br/doku.php/pessoais:paulojus:mbgbook:datasets. For further details on the malaria data, see Thomson et al. (1999).

## References

Diggle, P. J., Moyeed, R. A., Rowlingson, R. and Thomson, M. (2002). Childhood Malaria in the Gambia: A case-study in model-based geostatistics. Journal of the Royal Statistical Society. Series C (Applied Statistics), 51(4): 493-506.

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

Thomson, M. C., Connor, S. J., D'Alessandro, U., Rowlingson, B., Diggle, P., Creswell, M. and Greenwood, B. (2004). Predicting malaria infection in Gambian children from satellite data and bed net use surveys: the importance of spatial correlation in the interpretation of results. American Journal of Tropical Medicine and Hygiene, 61: 2-8.

## Examples

```
data("gambiaUTM")
gambiaUTM = unwrap(gambiaUTM)

plot(gambiaUTM, main="gambia data")

if(require('mapmisc', quietly=TRUE)) {
  gambiaTiles = openmap(gambiaUTM, zoom=6, buffer=50*1000)
  oldpar=map.new(gambiaTiles)
  plot(gambiaTiles, add=TRUE)
  plot(gambiaUTM, add=TRUE)
  scaleBar(gambiaUTM, 'topright')

  par(oldpar)
}
```

---

# geostatData *Prepare observations and covariates for a geostatistical model*

---

## Description

Extracts covariates at observation locations, aligns covariate rasters to a prediction grid, and returns the data structures used by `glgm` and `lgm`.

## Usage

```
geostatData(formula, data, grid, covariates, buffer = 0)
## S3 method for class 'SpatVector'
geostatData(formula, data, grid, covariates, buffer = 0)
## S3 method for class 'SpatRaster'
geostatData(formula, data, grid = data, covariates = NULL, buffer = 0)
## Default S3 method:
geostatData(formula, data, ...)
```

## Arguments

| | |
|---|---|
| `formula` | Model formula. Names on the right-hand side are taken from `data` when present and otherwise extracted from `covariates`. |
| `data` | A `SpatVector` of point observations, or a `SpatRaster` of observations on a lattice. |
| `grid` | A `SpatRaster` giving the prediction grid (and the lattice for the spatial random effect). For the `SpatRaster` method this defaults to `data`. |
| `covariates` | A `SpatRaster`, a multi-layer `SpatRaster`, or a list of `SpatRasters`. Layer or list names must match names in `formula`. |
| `buffer` | Extra space padded around `grid`, passed to the same argument of `squareRaster`. |
| `...` | Ignored. The default method reports an error if `data` is not a `SpatVector` or `SpatRaster`. |

## Value

A list with

| | |
|---|---|
| data | For SpatVector input, a SpatVector with extracted covariates and a space cell identifier. For SpatRaster input, a data.frame of cell values. |
| grid | The (possibly buffered) square SpatRaster used as the model lattice. |
| covariates | A data.frame of covariate values on grid. |
| formula | Returned only by the SpatRaster method. Equal to the supplied formula, or a rewritten formula when log-offsets are aggregated. |

## See Also

glgm, lgm, squareRaster

## Examples

```
data("loaloa")
loaloa = unwrap(loaloa)
elevationLoa = unwrap(elevationLoa)
eviLoa = unwrap(eviLoa)

res = geostatData(
    y ~ elev + evi,
    data = loaloa,
    covariates = list(elev = elevationLoa, evi = eviLoa),
    grid = squareRaster(loaloa, cells = 20, buffer = 1e4)
)
names(res)
names(res$data)
res$grid
```

---

# glgm-methods *Generalized Linear Geostatistical Models*

---

## Description

Fits a generalized linear geostatistical model or a log-Gaussian Cox process using inla

## Usage

```
## S4 method for signature 'ANY,ANY,ANY,ANY'
glgm(formula, data,  grid, covariates, buffer=0, shape=1, prior, ...)
## S4 method for signature 'formula,SpatRaster,ANY,ANY'
glgm(formula, data,  grid, covariates, buffer=0, shape=1, prior, ...)
## S4 method for signature 'formula,SpatVector,ANY,ANY'
glgm(formula, data,  grid, covariates, buffer=0, shape=1, prior, ...)
## S4 method for signature 'formula,data.frame,SpatRaster,data.frame'
glgm(formula, data,  grid, covariates, buffer=0, shape=1, prior, ...)
lgcp(formula=NULL, data,  grid, covariates=NULL, border, ...)
```

## Arguments

| | |
|---|---|
| data | An object of class SpatVector containing the data. |
| grid | Either an integer giving the number of cells in the x direction, or a raster object which will be used for the spatial random effect. If the cells in the raster are not square, the resolution in the y direction will be adjusted to make it so. |
| covariates | Either a single SpatRaster, a list of SpatRasters or a multi-layer SpatRaster containing covariate values used when making spatial predictions. Names of the raster layers or list elements correspond to names in the formula. If a covariate is missing from the data object it will be extracted from the rasters. Defaults to NULL for an intercept-only model. |
| formula | Model formula, defaults to a linear combination of each of the layers in the covariates object. The spatial random effect should not be supplied but the default can be overridden with a f(space,..) term. For glgm the response variable defaults to the first variable in the data object, and formula can be an integer or character string specifying the response variable. For lgcp, the formula should be one-sided. |
| prior | list with elements named range, sd, sdObs. See Details. |
| shape | Shape parameter for the Matern correlation function, must be 1 or 2. |
| buffer | Extra space padded around the data bounding box to reduce edge effects. |
| border | boundary of the region on which an LGCP is defined, passed to mask |
| ... | Additional options passed to inla in the INLA package |

## Details

This function performs Bayesian inference for generalized linear geostatistical models with INLA. The Markov random field approximation on a regular lattice is used for the spatial random effect. The range parameter is the distance at which the correlation is 0.13, or

$$cov[U(s+h), U(s)] = (2^{1-\nu}/Gamma(\nu))d^{\nu}besselK(d,\nu)$$

$$d = |h|\sqrt{8\nu}/range$$

where $\nu$ is the shape parameter. The range parameter produced by glgm multiplies the range parameter from INLA by the cell size.

Elements of prior can be named range, sd, or sdObs. Elements can consist of:

- a single value giving the prior median for penalized complexity priors (exponential on the sd or 1/range).
- a vector c(u=a, alpha=b) giving an quantile and probability for pc priors. For standard deviations alpha is an upper quantile, for the range parameter b = pr(1/range > 1/a).
- a vector c(lower=a, upper=b) giving a 0.025 and 0.975 quantiles for the sd or range.
- a list of the form list(prior='loggamma', param=c(1,2)) passed directly to inla.
- a two-column matrix of prior densities for the sd or range.

## Value

A list with two components named `inla`, `raster`, and `parameters`. `inla` contains the results of the call to the `inla` function. `raster` is a multi-layer SpatRaster with the following layers:

| | |
|---|---|
| `random.` | mean, sd, X0.0??quant: Posterior mean, standard deviation, and quantiles of the random effect |
| `predict.` | mean, sd, X0.0??quant: same for linear predictors, on the link scale |
| `predict.exp` | posterior mean of the exponential of the linear predictor |
| `predict.invlogit` | Only supplied if a binomial response variable was used. |

`parameters` contains a list with elements:

| | |
|---|---|
| `summary` | a table with parameter estimates and posterior quantiles |
| `range, sd` | prior and posterior distributions of range and standard deviations |

## See Also

https://www.r-inla.org

## Examples

```
## Not run:
# geostatistical model for the swiss rainfall data

if(inlaAvailable() ) {
  inlaSetThreads(2)
}

require("geostatsp")
data("swissRain")
swissRain = unwrap(swissRain)
swissAltitude = unwrap(swissAltitude)
swissBorder = unwrap(swissBorder)

swissRain$lograin = log(swissRain$rain)
swissFit =  glgm(formula="lograin", data=swissRain,
 grid=30,
 covariates=swissAltitude, family="gaussian",
 buffer=2000,
 prior = list(sd=1, range=100*1000, sdObs = 2),
 control.inla = list(strategy='gaussian')
 )

if(!is.null(swissFit$parameters) ) {

 swissExc = excProb(swissFit, threshold=log(25))

 swissExcRE = excProb(swissFit$inla$marginals.random$space,
  log(1.5),template=swissFit$raster)
```

```
 swissFit$parameters$summary

 matplot(
  swissFit$parameters$range$postK[,'x'],
  swissFit$parameters$range$postK[,c('y','prior')],
  type="l", lty=1, xlim = c(0, 1000),
  xlab = 'km', ylab='dens')
 legend('topright', lty=1, col=1:2, legend=c('post','prior'))

 plot(swissFit$raster[["predict.exp"]])

 mycol = c("green","yellow","orange","red")
 mybreaks = c(0, 0.2, 0.8, 0.95, 1)
 plot(swissBorder)
 plot(swissExc, breaks=mybreaks, col=mycol,add=TRUE,legend=FALSE)
 plot(swissBorder, add=TRUE)
 legend("topleft",legend=mybreaks, fill=c(NA,mycol))


 plot(swissBorder)
 plot(swissExcRE, breaks=mybreaks, col=mycol,add=TRUE,legend=FALSE)
 plot(swissBorder, add=TRUE)
 legend("topleft",legend=mybreaks, fill=c(NA,mycol))
}




# a log-Gaussian Cox process example

myPoints = vect(cbind(rbeta(100,2,2), rbeta(100,3,4)))


mycov = rast(matrix(rbinom(100, 1, 0.5), 10, 10), extent=ext(0, 1, 0, 1))
names(mycov)="x1"


inlaSetThreads(2)

res = lgcp(
 formula=~factor(x1),
 data=myPoints,
 grid=squareRaster(ext(0,1,0,1), 20), covariates=mycov,
 prior=list(sd=c(0.9, 1.1), range=c(0.4, 0.41),
 control.inla = list(strategy='gaussian'), verbose=TRUE)
)
if(length(res$parameters)) {
 plot(res$raster[["predict.exp"]])
 plot(myPoints,add=TRUE,col="#0000FF30",cex=0.5)
}

## End(Not run)
```

## inla.models *Valid models in INLA*

### Description

calls the function of the same name in INLA

### Usage

```
inla.models()
```

### Value

a list

## inlaAvailable *Check whether INLA is usable*

### Description

Tests whether the INLA package is installed and its binaries work, and optionally sets INLA thread options.

### Usage

```
inlaAvailable()
inlaSetThreads(num.threads = 2L)
```

### Arguments

| | |
|---|---|
| num.threads | Number of threads passed to INLA::inla.setOption. |

### Details

requireNamespace("INLA") can succeed even when the INLA program binaries are missing or broken. inlaAvailable additionally probes INLA::inla.getOption so examples and tests can skip safely. inlaSetThreads is a no-op when INLA is unavailable.

### Value

inlaAvailable returns a logical. inlaSetThreads returns TRUE if options were set, FALSE otherwise, invisibly.

### Examples

```
inlaAvailable()
inlaSetThreads(2)
```

krigeLgm *Spatial prediction, or Kriging*

## Description

Perform spatial prediction, producing a raster of predictions and conditional standard deviations.

## Usage

```
krigeLgm(formula, data, grid,  covariates = NULL,
 param,
    expPred = FALSE, nuggetInPrediction = TRUE,
    mc.cores=getOption("mc.cores", 1L))
```

## Arguments

| | |
|---|---|
| formula | Either a model formula, or a data frame of linear covariates. |
| data | A SpatVector containing the data to be interpolated |
| grid | Either a SpatRaster , or a single integer giving the number of cells in the X direction which predictions will be made on. If the later the predictions will be a raster of square cells covering the bounding box of data. |
| covariates | The spatial covariates used in prediction, either a SpatRaster stack or list of rasters. |
| param | A vector of named model parameters, as produced by likfitLgm |
| expPred | Should the predictions be exponentiated, defaults to FALSE. |
| nuggetInPrediction | If TRUE, predict new observations by adding the nugget effect. The prediction variances will be adjusted accordingly, and the predictions on the natural scale for logged or Box Cox transformed data will be affected. Otherwise predict fitted values. |
| mc.cores | passed to mclapply if greater than 1. |

## Details

Given the model parameters and observed data, conditional means and variances of the spatial random field are computed.

## Value

A raster is returned with the following layers:

| | |
|---|---|
| fixed | Estimated means from the fixed effects portion of the model |
| random | Predicted random effect |
| krige.var | Conditional variance of predicted random effect (on the transformed scale if applicable) |

| | |
|---|---|
| predict | Prediction of the response, sum of fixed and random effects. If exp.pred is TRUE, gives predictions on the exponentiated scale, and half of krige.var is added prior to exponentiating |
| predict.log | If exp.pred=TRUE, the prediction of the logged process. |
| predict.boxcox | If a box cox transformation was used, the prediction of the process on the transformed scale. |

If the prediction locations are different for fixed and random effects (typically coarser for the random effects), a list with two multi-layer SpatRasters is returned.

| | |
|---|---|
| prediction | A SpatRaster as above, though the random effect prediction is resampled to the same locations as the fixed effects. |
| random | the predictions and conditional variance of the random effects, on the same raster as newdata |

## See Also

lgm

## Examples

```
data('swissRain')
swissAltitude = unwrap(swissAltitude)
swissRain = unwrap(swissRain)
swissRain$lograin = log(swissRain$rain)
swissRain[[names(swissAltitude)]] = extract(swissAltitude, swissRain, ID=FALSE)

swissFit = likfitLgm(data=swissRain,
   formula=lograin~ CHE_alt,
   param=c(range=46500, nugget=0.05,shape=1,
     anisoAngleDegrees=35, anisoRatio=12),
   paramToEstimate = c("range","nugget",
    "anisoAngleDegrees", "anisoRatio")
)
myTrend = swissFit$model$formula
myParams = swissFit$param


swissBorder = unwrap(swissBorder)

swissKrige = krigeLgm(
 data=swissRain,
 formula = myTrend,
 covariates = swissAltitude,
 param=myParams,
 grid = squareRaster(swissBorder, 40), expPred=TRUE,
 mc.cores=1L)

plot(swissKrige[["predict"]], main="predicted rain")
plot(swissBorder, add=TRUE)
```

lgm-methods *Linear Geostatistical Models*

## Description

Calculate MLE's of model parameters and perform spatial prediction.

## Usage

```
## S4 method for signature 'missing,ANY,ANY,ANY'
lgm(
formula, data, grid, covariates,
buffer=0, shape=1, boxcox=1, nugget = 0,
expPred=FALSE, nuggetInPrediction=TRUE,
reml=TRUE,mc.cores=1,
aniso=FALSE,
fixShape=TRUE,
fixBoxcox=TRUE,
fixNugget = FALSE,
...)
## S4 method for signature 'numeric,ANY,ANY,ANY'
lgm(
formula, data, grid, covariates,
buffer=0, shape=1, boxcox=1, nugget = 0,
expPred=FALSE, nuggetInPrediction=TRUE,
reml=TRUE,mc.cores=1,
aniso=FALSE,
fixShape=TRUE,
fixBoxcox=TRUE,
fixNugget = FALSE,
...)
## S4 method for signature 'character,ANY,ANY,ANY'
lgm(
formula, data, grid, covariates,
buffer=0, shape=1, boxcox=1, nugget = 0,
expPred=FALSE, nuggetInPrediction=TRUE,
reml=TRUE,mc.cores=1,
aniso=FALSE,
fixShape=TRUE,
fixBoxcox=TRUE,
fixNugget = FALSE,
...)
## S4 method for signature 'formula,SpatVector,numeric,ANY'
```

```
lgm(
formula, data, grid, covariates,
buffer=0, shape=1, boxcox=1, nugget = 0,
expPred=FALSE, nuggetInPrediction=TRUE,
reml=TRUE,mc.cores=1,
aniso=FALSE,
fixShape=TRUE,
fixBoxcox=TRUE,
fixNugget = FALSE,
...)
## S4 method for signature 'formula,SpatVector,SpatRaster,missing'
lgm(
formula, data, grid, covariates,
buffer=0, shape=1, boxcox=1, nugget = 0,
expPred=FALSE, nuggetInPrediction=TRUE,
reml=TRUE,mc.cores=1,
aniso=FALSE,
fixShape=TRUE,
fixBoxcox=TRUE,
fixNugget = FALSE,
...)
## S4 method for signature 'formula,SpatVector,SpatRaster,list'
lgm(
formula, data, grid, covariates,
buffer=0, shape=1, boxcox=1, nugget = 0,
expPred=FALSE, nuggetInPrediction=TRUE,
reml=TRUE,mc.cores=1,
aniso=FALSE,
fixShape=TRUE,
fixBoxcox=TRUE,
fixNugget = FALSE,
...)
## S4 method for signature 'formula,SpatVector,SpatRaster,SpatRaster'
lgm(
formula, data, grid, covariates,
buffer=0, shape=1, boxcox=1, nugget = 0,
expPred=FALSE, nuggetInPrediction=TRUE,
reml=TRUE,mc.cores=1,
aniso=FALSE,
fixShape=TRUE,
fixBoxcox=TRUE,
fixNugget = FALSE,
...)
## S4 method for signature 'formula,SpatVector,SpatRaster,data.frame'
lgm(
formula, data, grid, covariates,
buffer=0, shape=1, boxcox=1, nugget = 0,
expPred=FALSE, nuggetInPrediction=TRUE,
```

```
reml=TRUE,mc.cores=1,
aniso=FALSE,
fixShape=TRUE,
fixBoxcox=TRUE,
fixNugget = FALSE,
...)
## S4 method for signature 'formula,SpatRaster,ANY,ANY'
lgm(
formula, data, grid, covariates,
buffer=0, shape=1, boxcox=1, nugget = 0,
expPred=FALSE, nuggetInPrediction=TRUE,
reml=TRUE,mc.cores=1,
aniso=FALSE,
fixShape=TRUE,
fixBoxcox=TRUE,
fixNugget = FALSE,
...)
## S4 method for signature 'formula,data.frame,SpatRaster,data.frame'
lgm(
formula, data, grid, covariates,
buffer=0, shape=1, boxcox=1, nugget = 0,
expPred=FALSE, nuggetInPrediction=TRUE,
reml=TRUE,mc.cores=1,
aniso=FALSE,
fixShape=TRUE,
fixBoxcox=TRUE,
fixNugget = FALSE,
...)
```

## Arguments

| | |
|---|---|
| formula | A model formula for the fixed effects, or a character string specifying the response variable. |
| data | A SpatVector or SpatRaster layer, brick or stack containing the locations and observations, and possibly covariates. |
| grid | Either a SpatRaster, or a single integer giving the number of cells in the X direction which predictions will be made on. If the later the predictions will be a raster of square cells covering the bounding box of data. |
| covariates | The spatial covariates used in prediction, either a SpatRaster stack or list of rasters. Covariates in formula but not in data will be extracted from covariates. |
| shape | Order of the Matern correlation |
| boxcox | Box-Cox transformation parameter (or vector of parameters), set to 1 for no transformation. |
| nugget | Value for the nugget effect (observation error) variance, or vector of such values. |
| expPred | Should the predictions be exponentiated, defaults to FALSE. |

`nuggetInPrediction`
: If `TRUE`, predict new observations by adding the nugget effect. The prediction variances will be adjusted accordingly, and the predictions on the natural scale for logged or Box Cox transformed data will be affected. Otherwise predict fitted values.

`reml`
: If `TRUE` (the default), use restricted maximum likelihood.

`mc.cores`
: If `mc.cores>1`, this argument is passed to `mclapply` and computations are done in parallel where possible.

`aniso`
: Set to `TRUE` to use geometric anisotropy.

`fixShape`
: Set to `FALSE` to estimate the Matern order

`fixBoxcox`
: Set to `FALSE` to estimate the Box-Cox parameter.

`fixNugget`
: Set to `FALSE` to estimate the nugget effect parameter.

`buffer`
: Extra distance to add around `grid`.

`...`
: Additional arguments passed to `likfitLgm`. Starting values can be specified with a vector `param` of named elements

## Details

When `data` is a `SpatVector`, parameters are estimated using `optim` to maximize the log-likelihood function computed by `likfitLgm` and spatial prediction accomplished with `krigeLgm`.

With `data` being a `Raster` object, a Markov Random Field approximation to the Matern is used (experimental). Parameters to be estimated should be provided as vectors of possible values, with optimization only considering the parameter values supplied.

## Value

A list is returned which includes a `SpatRaster` named `predict` having layers:

`fixed`
: Estimated means from the fixed effects portion of the model

`random`
: Predicted random effect

`krigeSd`
: Conditional standard deviation of predicted random effect (on the transformed scale if applicable)

`predict`
: Prediction of the response, sum of predicted fixed and random effects. For Box-Cox or log-transformed data on the natural (untransformed) scale.

`predict.log`
: If `exp.pred=TRUE`, the prediction of the logged process.

`predict.boxcox`
: If a box cox transformation was used, the prediction of the process on the transformed scale.

In addition, the element `summery` contains a table of parameter estimates and confidence intervals. `optim` contains the output from the call to the `optim` function.

## See Also

`likfitLgm`, `krigeLgm`

## Examples

```
data("swissRain")
swissRain = unwrap(swissRain)
swissAltitude = unwrap(swissAltitude)
swissBorder = unwrap(swissBorder)

swissRes =  lgm( formula="rain",
 data=swissRain[1:60,], grid=20,
 covariates=swissAltitude, boxcox=0.5, fixBoxcox=TRUE,
 shape=1, fixShape=TRUE,
 aniso=FALSE, nugget=0, fixNugget=FALSE,
 nuggetInPrediction=FALSE
)

swissRes$summary

plot(swissRes$predict[["predict"]], main="predicted rain")
plot(swissBorder, add=TRUE)
```

---

# likfitLgm *Likelihood Based Parameter Estimation for Gaussian Random Fields*

---

## Description

*Maximum likelihood* (ML) or *restricted maximum likelihood* (REML) parameter estimation for (transformed) Gaussian random fields.

## Usage

```
likfitLgm(formula, data,
  paramToEstimate = c("range","nugget"),
  reml=TRUE,
  coordinates=data,
  param=NULL,
  upper=NULL,lower=NULL, parscale=NULL,
  verbose=FALSE)


loglikLgm(param,
  data, formula, coordinates=data,
  reml=TRUE,
  minustwotimes=TRUE,
  moreParams=NULL)
```

## Arguments

| | |
|---|---|
| formula | A formula for the fixed effects portion of the model, specifying a response and covariates. Alternately, data can be a vector of observations and formula can be a model matrix. |
| data | An object of class SpatVect, a vector of observations, or a data frame containing observations and covariates. |
| coordinates | A SpatVect object containing the locations of each observation, which defaults to data. Alternately, coordinates can be a symmetricMatrix-class or dist object reflecting the distance matrix of these coordinates (though this is only permitted if the model is isotropic). |
| param | A vector of model parameters, with named elements being amongst range, nugget, boxcox, shape, anisoAngleDegrees, anisoAngleRadians,anisoRatio, and possibly variance (see matern). When calling likfitLgm this vector is a combination of starting values for parameters to be estiamated and fixed values of parameters which will not be estimated. For loglikLgm, it is the covariance parameters for which the likelihood will be evaluated. |
| reml | Whether to use Restricted Likelihood rather than Likelihood, defaults to TRUE. |
| paramToEstimate | Vector of names of model parameters to estimate, with parameters excluded from this list being fixed. The variance parameter and regression coefficients are always estimated even if not listed. |
| lower | Named vector of lower bounds for model parameters passed to optim, defaults are used for parameters not specified. |
| upper | Upper bounds, as above. |
| parscale | Named vector of scaling of parameters passed as control=list(parscale=parscale) to optim. |
| minustwotimes | Return -2 times the log likelihood rather than the likelihood |
| moreParams | Vector of additional parameters, combined with param. Used for passing fixed parameters to loglikLgm from within optim. |
| verbose | if TRUE information is printed by optim. |

## Value

likfitLgm produces list with elements

| | |
|---|---|
| parameters | Maximum Likelihood Estimates of model parameters |
| varBetaHat | Variance matrix of the estimated regression parameters |
| optim | results from optim |
| trend | Either formula for the fixed effects or names of the columns of the model matrix, depending on trend supplied. |
| summary | a table of parameter estimates, standard errors, confidence intervals, p values, and a logical value indicating whether each parameter was estimated as opposed to fixed. |

`resid` residuals, being the observations minus the fixed effects, on the transformed scale.

`loglikLgm` returns a scalar value, either the log likelihood or -2 times the log likelihood. Attributes of this result include the vector of parameters (including the MLE's computed for the variance and coefficients), and the variance matrix of the coefficient MLE's.

## See Also

`lgm`

## Examples

```
n=40
mydat = vect(
 cbind(runif(n), seq(0,1,len=n)),
 atts=data.frame(cov1 = rnorm(n), cov2 = rpois(n, 0.5))
 )

# simulate a random field
trueParam = c(variance=2^2, range=0.35, shape=2, nugget=0.5^2)
set.seed(1)

oneSim = RFsimulate(model=trueParam,x=mydat)

values(mydat) = cbind(values(mydat) , values(oneSim))

# add fixed effects
mydat$Y = -3 + 0.5*mydat$cov1 + 0.2*mydat$cov2 +
 mydat$sim + rnorm(length(mydat), 0, sd=sqrt(trueParam["nugget"]))

plot(mydat, "sim", col=rainbow(10), main="U")
plot(mydat, "Y", col=rainbow(10), main="Y")


myres = likfitLgm(
 formula=Y ~ cov1 + cov2,
 data=mydat,
 param=c(range=0.1,nugget=0.1,shape=2),
 paramToEstimate = c("range","nugget")
 )

myres$summary[,1:4]


# plot variograms of data, true model, and estimated model
myv = variog(mydat, formula=Y ~ cov1 + cov2,option="bin", max.dist=0.5)
# myv will be NULL if geoR isn't installed
if(!is.null(myv)){
plot(myv, ylim=c(0, max(c(1.2*sum(trueParam[c("variance", "nugget")]),myv$v))),
 main="variograms")
distseq = seq(0, 0.5, len=50)
lines(distseq,
```

```
 sum(myres$param[c("variance", "nugget")]) - matern(distseq, param=myres$param),
 col='blue', lwd=3)
lines(distseq,
 sum(trueParam[c("variance", "nugget")]) - matern(distseq, param=trueParam),
 col='red')

legend("bottomright", fill=c("black","red","blue"),
 legend=c("data","true","MLE"))
}

# without a nugget
myresNoN = likfitLgm(
 formula=Y ~ cov1 + cov2,
 data=mydat,
 param=c(range=0.1,nugget=0,shape=1),
 paramToEstimate = c("range")
 )

myresNoN$summary[,1:4]


# plot variograms of data, true model, and estimated model
myv = variog(mydat, formula=Y ~ cov1 + cov2,option="bin", max.dist=0.5)

if(!is.null(myv)){
plot(myv, ylim=c(0, max(c(1.2*sum(trueParam[c("variance", "nugget")]),myv$v))),
 main="variograms")

distseq = seq(0, 0.5, len=50)
lines(distseq,
 sum(myres$param[c("variance", "nugget")]) - matern(distseq, param=myres$param),
 col='blue', lwd=3)
lines(distseq,
 sum(trueParam[c("variance", "nugget")]) - matern(distseq, param=trueParam),
 col='red')

lines(distseq,
 sum(myresNoN$param[c("variance", "nugget")]) -
   matern(distseq, param=myresNoN$param),
 col='green', lty=2, lwd=3)
legend("bottomright", fill=c("black","red","blue","green"),
 legend=c("data","true","MLE","no N"))
}


# calculate likelihood
temp=loglikLgm(param=myres$param,
  data=mydat,
  formula = Y ~ cov1 + cov2,
  reml=FALSE, minustwotimes=FALSE)
```

```
# an anisotropic example


trueParamAniso = param=c(variance=2^2, range=0.2, shape=2,
  nugget=0,anisoRatio=4,anisoAngleDegrees=10, nugget=0)

mydat$U = geostatsp::RFsimulate(trueParamAniso,mydat)$sim

mydat$Y = -3 + 0.5*mydat$cov1 + 0.2*mydat$cov2 +
 mydat$U + rnorm(length(mydat), 0, sd=sqrt(trueParamAniso["nugget"]))

oldpar = par(no.readonly = TRUE)

par(mfrow=c(1,2), mar=rep(0.1, 4))

plot(mydat, col=as.character(cut(mydat$U, breaks=50, labels=heat.colors(50))),
 pch=16, main="aniso")

plot(mydat, col=as.character(cut(mydat$Y, breaks=50, labels=heat.colors(50))),
 pch=16,main="iso")



myres = likfitLgm(
 formula=Y ~ cov1 + cov2,
 data=mydat,
 param=c(range=0.1,nugget=0,shape=2, anisoAngleDegrees=0, anisoRatio=2),
 paramToEstimate = c("range","nugget","anisoRatio","anisoAngleDegrees")
 )

myres$summary

par(oldpar)
par(mfrow=c(1,2))

myraster = rast(nrows=30,ncols=30,xmin=0,xmax=1,ymin=0,ymax=1)
covEst = matern(myraster, y=c(0.5, 0.5), par=myres$param)
covTrue = matern(myraster, y=c(0.5, 0.5), par=trueParamAniso)

plot(covEst, main="estimate")
plot(covTrue, main="true")

par(oldpar)
```

---

`loaloa` *Loaloa prevalence data from 197 village surveys*

---

## Description

Location and prevalence data from villages, elevation an vegetation index for the study region.

## Usage

```
data("loaloa")
```

## Format

`loaloa` is a `SpatVector` containing the data, with columns `N` being the number of individuals tested and `y` being the number of positives. `elevationLoa` is a raster of elevation data. `eviLoa` is a raster of vegetation index for a specific date. `ltLoa` is land type. `ltLoa` is a raster of land types. 1 2 5 6 7 8 9 10 11 12 13 14 15 `tempLoa` is a raster of average temperature in degrees C.

## Source

https://web.archive.org/web/20240110054727/http://www.leg.ufpr.br/doku.php/pessoais:paulojus:mbgbook:datasets for the loaloa data, https://web.archive.org/web/20241129120557/https://lpdaac.usgs.gov/data/ for EVI and land type and https://srtm.csi.cgiar.org for the elevation data.

## Examples

```
data("loaloa")
loaloa = unwrap(loaloa)
plot(loaloa, main="loaloa villages")

# elevation
elevationLoa = unwrap(elevationLoa)
plot(elevationLoa, col=terrain.colors(100), main="elevation")
points(loaloa)

# vegetation index
eviLoa = unwrap(eviLoa)
plot(eviLoa, main="evi")
points(loaloa)

tempLoa = unwrap(tempLoa)
plot(tempLoa, main="temperature")
points(loaloa)



# land type, a categorical variable
ltLoa = unwrap(ltLoa)
plot(ltLoa)
if(requireNamespace("mapmisc")){
 mapmisc::legendBreaks("bottomleft",ltLoa, bty='n')
}
points(loaloa)
```

matern *Evaluate the Matern correlation function*

## Description

Returns the Matern covariance for the distances supplied.

## Usage

```
 matern( x, param=c(range=1, variance=1, shape=1),
 type=c('variance','cholesky','precision', 'inverseCholesky'),
 y=NULL)
 ## S3 method for class 'SpatVector'
matern(x,  param,
 type=c('variance','cholesky','precision', 'inverseCholesky'),
 y=NULL)
 ## Default S3 method:
matern( x, param,
 type=c('variance','cholesky','precision', 'inverseCholesky'),
 y=NULL)
 ## S3 method for class 'dist'
matern( x, param,
 type=c('variance','cholesky','precision', 'inverseCholesky'),
 y=NULL)
 ## S3 method for class 'SpatRaster'
matern( x,  param,
 type=c('variance','cholesky','precision', 'inverseCholesky'),
 y=NULL)
 fillParam(param)
```

## Arguments

| | |
|---|---|
| x | A vector or matrix of distances, or SpatRaster or SpatVector of locations, see Details below. |
| param | A vector of named model parameters with, at a minimum names range and shape (see Details), and optionally variance (defaults to 1) and nugget (defaults to zero). For Geometric Anisotropy add anisoRatio and either anisoAngleDegrees or anisoAngleRadians |
| type | specifies if the variance matrix, the Cholesky decomposition of the variance matrix, the precision matrix, or the inverse of the Cholesky L matrix is returned. |
| y | Covariance is calculated for the distance between locations in x and y. If y=NULL, covariance of x with itself is produced. However, if x is a matrix or vector it is assumed to be a set of distances and y is ignored. |

## Details

The formula for the Matern correlation function is

$$M(x) = \frac{variance}{\Gamma(shape)} 2^{1-shape} \left( \frac{x\sqrt{8shape}}{range} \right)^{shape} besselK(x\sqrt{8shape}/range, shape)$$

The range argument is sqrt(8*shape)*phi.geoR, sqrt(8*shape)*scale.whittle.RandomFields, and 2*scale.matern.RandomFields.

Geometric anisotropy is only available when x is a SpatRaster or SpatVector. The parameter 'anisoAngle' refers to rotation of the coordinates anti-clockwise by the specified amount prior to calculating distances, which has the effect that the contours of the correlation function are rotated clockwise by this amount. anisoRatio is the amount the Y coordinates are divided by by post rotation prior to calculating distances. A large value of anisoRatio makes the Y coordinates smaller and increases the correlation in the Y direction.

When x or y are rasters, cells are indexed row-wise starting at the top left.

## Value

When x is a vector or matrix or object of class dist, a vector or matrix of covariances is returned. With x being SpatVector , y must also be SpatVector and a matrix of correlations between x and y is returned. When x is a Raster, and y is a single location a Raster of covariances between each pixel centre of x and y is returned.

## Examples

```
param=c(shape=2.5,range=1,variance=1)
u=seq(0,4,len=200)
uscale = sqrt(8*param['shape'])* u / param['range']

theMaterns = cbind(
 dist=u,
 manual= param['variance']* 2^(1- param['shape']) *
   ( 1/gamma(param['shape'])  )  *
   uscale^param['shape'] * besselK(uscale , param['shape']),
 geostatsp=geostatsp::matern(u, param=param)
)
head(theMaterns)
matplot(theMaterns[,'dist'],
 theMaterns[,c('manual','geostatsp')],
 col=c('red','blue'), type='l',
 xlab='dist', ylab='var')
legend('topright', fill=c('red','blue'),
 legend=c('manual','geostatsp'))



# example with raster
myraster = rast(nrows=40,ncols=60,extent=ext(-3, 3,-2,2))
param = c(range=2, shape=2, anisoRatio=2,
 anisoAngleDegrees=-25,variance=20)
```

```
# plot correlation of each cell with the origin
myMatern = matern(myraster, y=c(0,0), param=param)


plot(myMatern, main="anisortopic matern")


# correlation matrix for all cells with each other
myraster = rast(nrows=4,ncols=6,extent = ext(-3, 3, -2, 2))
myMatern = matern(myraster, param=c(range=1, shape=2, variance=0.5^2))
dim(myMatern)

# plot the cell ID's
values(myraster) = seq(1, ncell(myraster))
mydf = as.data.frame(myraster, xy=TRUE)
plot(mydf$x, mydf$y, type='n', main="cell ID's")
text(mydf$x, mydf$y, mydf$lyr.1)
# correlation between bottom-right cell and top right cell is
myMatern[6,24]

# example with points
mypoints = vect(
 cbind(runif(8), runif(8))
 )
# variance matrix from points
m1=matern(mypoints,
 param=c(range=2,shape=1.4,variance=4,nugget=1))
# cholesky of variance from distances
c2=matern(dist(crds(mypoints)), param=c(range=2,shape=1.4,variance=4,nugget=1),type='cholesky')

# check it's correct
quantile(as.vector(m1- tcrossprod(c2)))

# example with vector of distances
range=3
distVec = seq(0, 2*range, len=100)
shapeSeq = c(0.5, 1, 2,20)
theCov = NULL
for(D in shapeSeq) {
 theCov = cbind(theCov, matern(distVec, param=c(range=range, shape=D)))
}
matplot(distVec, theCov, type='l', lty=1, xlab='distance', ylab='correlation',
 main="matern correlations")
legend("right", fill=1:length(shapeSeq), legend=shapeSeq,title='shape')
# exponential

distVec2 = seq(0, max(distVec), len=20)
points(distVec2, exp(-2*(distVec2/range)),cex=1.5, pch=5)
# gaussian
points(distVec2, exp(-2*(distVec2/range)^2), col='blue',cex=1.5, pch=5)
legend("bottomleft", pch=5, col=c('black','blue'), legend=c('exp','gau'))
```

```
# comparing to geoR and RandomFields

if (requireNamespace("RandomFields", quietly = TRUE) &
requireNamespace("geoR", quietly = TRUE)
) {

covGeoR = covRandomFields = NULL

for(D in shapeSeq) {
 covGeoR = cbind(covGeoR,
  geoR::matern(distVec, phi=range/sqrt(8*D), kappa=D))
 covRandomFields = cbind(covRandomFields,
  RandomFields::RFcov(x=distVec,
  model=RandomFields::RMmatern(nu=D, var=1,
    scale=range/2) ))
}



matpoints(distVec, covGeoR, cex=0.5, pch=1)
matpoints(distVec, covRandomFields, cex=0.5, pch=2)

legend("topright", lty=c(1,NA,NA), pch=c(NA, 1, 2),
 legend=c("geostatsp","geoR","RandomFields"))
}
```

---

`maternGmrfPrec` *Precision matrix for a Matern spatial correlation*

---

## Description

Produces the precision matrix for a Gaussian random field on a regular square lattice, using a Markov random field approximation.

## Usage

```
maternGmrfPrec(N, ...)
## S3 method for class 'dgCMatrix'
maternGmrfPrec(N,
 param=c(variance=1, range=1, shape=1, cellSize=1),
  adjustEdges=FALSE,...)
## Default S3 method:
maternGmrfPrec(N, Ny=N,
  param=c(variance=1, range=1, shape=1, cellSize=1),
  adjustEdges=FALSE, ...)
NNmat(N, Ny=N, nearest=3, adjustEdges=FALSE)
## S3 method for class 'SpatRaster'
NNmat(N, Ny=N, nearest=3, adjustEdges=FALSE)
```

```
## Default S3 method:
NNmat(N, Ny=N, nearest=3, adjustEdges=FALSE)
```

## Arguments

| | |
|---|---|
| N | Number of grid cells in the x direction, or a matrix denoting nearest neighbours. |
| Ny | Grid cells in the y direction, defaults to N for a square grid |
| param | Vector of model parameters, with named elements: scale, scale parameter for the correlation function; prec, precision parameter; shape, Matern differentiability parameter (0, 1, or 2); and cellSize, the size of the grid cells. Optionally, variance and range can be given in place of prec and scale, when the former are present and the latter are missing the reciprocal of the former are taken. |
| adjustEdges | If TRUE, adjust the precision matrix so it does not implicitly assume the field takes values of zero outside the specified region. Defaults to FALSE. Can be a character string specifying the parameters to use for the correction, such as 'optimal' or 'optimalShape', with TRUE equivalent to 'theo' |
| nearest | Number of nearest neighbours to compute |
| ... | Additional arguments passed to maternGmrfPrec.dsCMatrix |

## Details

The numbering of cells is consistent with the terra package. Cell 1 is the top left cell, with cell 2 being the cell to the right and numbering continuing row-wise.

The nearest neighbour matrix N has: N[i,j]=1 if i=j; takes a value 2 if i and j are first 'rook' neighbours; 3 if they are first 'bishop' neighbours; 4 if they are second 'rook' neighbours; 5 if 'knight' neighbours; and 6 if third 'rook' neighbours.

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    0    0    0    6    0    0    0
[2,]    0    0    5    4    5    0    0
[3,]    0    5    3    2    3    5    0
[4,]    6    4    2    1    2    4    6
[5,]    0    5    3    2    3    5    0
[6,]    0    0    5    4    5    0    0
[7,]    0    0    0    6    0    0    0
```

## Value

A sparse matrix dsCMatrix-class object, containing a precision matrix for a Gaussian random field or (from the NNmat function) a matrix denoting neighbours.

## Examples

```
# produces the matrix above
 matrix(NNmat(11, 11, nearest=5)[,11*5+6],11, 11)

 params=c(range = 3, shape=2, variance=5^2)
```

```
myGrid = squareRaster(ext(0,20,0,10), 40)

# precision matrix without adjusting for edge effects
precMat =maternGmrfPrec(N=myGrid, param=params)

attributes(precMat)$info$precisionEntries

midcell = cellFromRowCol(myGrid,
 round(nrow(myGrid)/2), round(ncol(myGrid)/2)) # the middle cell
edgeCell = cellFromRowCol(myGrid, 5,5)# cell near corner

# show precision of middle cell
precMid=matrix(precMat[,midcell],
 nrow(myGrid), ncol(myGrid), byrow=TRUE)

precMid[round(nrow(precMid)/2) + seq(-5, 5),
 round(ncol(precMid)/2) + seq(-3, 3)]

# and with the adjustment
precMatCorr =maternGmrfPrec(
 N = myGrid, param=params,
 adjustEdges=TRUE)



# variance matrices
varMat = Matrix::solve(precMat)
varMatCorr = Matrix::solve(precMatCorr)

# compare covariance matrix to the matern
xseq = seq(-ymax(myGrid), ymax(myGrid), len=1000)/1.5
plot(xseq, matern(xseq, param=params),
 type = 'l',ylab='cov', xlab='dist',
 ylim=c(0, params["variance"]*1.1),
 main="matern v gmrf")

# middle cell
varMid=matrix(varMat[,midcell],
 nrow(myGrid), ncol(myGrid), byrow=TRUE)
varMidCorr=matrix(varMatCorr[,midcell],
 nrow(myGrid), ncol(myGrid), byrow=TRUE)
xseqMid = yFromRow(myGrid) - yFromCell(myGrid, midcell)
points(xseqMid, varMid[,colFromCell(myGrid, midcell)],
 col='red')
points(xseqMid, varMidCorr[,colFromCell(myGrid, midcell)],
  col='blue', cex=0.5)

# edge cells
varEdge=matrix(varMat[,edgeCell],
  nrow(myGrid), ncol(myGrid), byrow=TRUE)
varEdgeCorr = matrix(varMatCorr[,edgeCell],
  nrow(myGrid), ncol(myGrid), byrow=TRUE)
xseqEdge = yFromRow(myGrid) - yFromCell(myGrid, edgeCell)
```

```
points(xseqEdge,
 varEdge[,colFromCell(myGrid, edgeCell)],
 pch=3,col='red')
points(xseqEdge,
  varEdgeCorr[,colFromCell(myGrid, edgeCell)],
  pch=3, col='blue')

legend("topright", lty=c(1, NA, NA, NA, NA),
  pch=c(NA, 1, 3, 16, 16),
 col=c('black','black','black','red','blue'),
 legend=c('matern', 'middle','edge','unadj', 'adj')
 )


# construct matern variance matrix

myraster = attributes(precMat)$raster
covMatMatern = matern(myraster, param=params)

 prodUncor = crossprod(covMatMatern, precMat)
 prodCor = crossprod(covMatMatern, precMatCorr)

 quantile(Matrix::diag(prodUncor),na.rm=TRUE)
 quantile(Matrix::diag(prodCor),na.rm=TRUE)

 quantile(prodUncor[lower.tri(prodUncor,diag=FALSE)],na.rm=TRUE)
 quantile(prodCor[lower.tri(prodCor,diag=FALSE)],na.rm=TRUE)
```

---

`murder` *Murder locations*

---

## Description

Locations of murders in Toronto 1990-2014

## Usage

```
data("murder")
```

## Format

`murder` is a `SpatVector` object of murder locations. `torontoPdens`, `torontoIncome`, and `torontoNight` are rasters containing population density (per hectare), median household income, and ambient light respectively. `torontoBorder` is a `SpatVector` of the boundary of the city of Toronto.

## Source

Murder data:https://geo.scholarsportal.info/#r/details/_uri@=1882586248,

Lights: https://www.earthdata.nasa.gov/data/instruments/viirs/land-near-real-time-data

Boundary files: https://www150.statcan.gc.ca/n1/en/catalogue/92-160-X

Income: https://www150.statcan.gc.ca/n1/en/catalogue/97-551-X2006007

## Examples

```
data("murder")
murder= unwrap(murder)
torontoBorder = unwrap(torontoBorder)

plot(torontoBorder)
points(murder, col="#0000FF40", cex=0.5)

data("torontoPop")
torontoNight = unwrap(torontoNight)
torontoIncome = unwrap(torontoIncome)
torontoPdens = unwrap(torontoPdens)

# light
plot(torontoNight, main="Toronto ambient light")
plot(torontoBorder, add=TRUE)
points(murder, col="#0000FF40", cex=0.5)


# income
plot(torontoIncome, main="Toronto Income")
points(murder, col="#0000FF40", cex=0.5)
plot(torontoBorder, add=TRUE)

# population density
plot(torontoPdens, main="Toronto pop dens")
points(murder, col="#0000FF40", cex=0.5)
plot(torontoBorder, add=TRUE)
```

---

# pcPriorRange *PC prior for range parameter*

---

## Description

Creates a penalized complexity prior for the range parameter

## Usage

```
pcPriorRange(q, p=0.5, cellSize=1)
```

### Arguments

| | |
|---|---|
| q | Lower quantile for the range parameter |
| p | probability that the range is below this quantile, defaults to the median |
| cellSize | size of grid cells, can be a raster. |

### Details

q is the quantile in spatial units, usually meters, and the scale parameter follows an exponential distribution. A prior PC prior distribution for the range parameter in units of grid cells, which INLA requires, is computed.

### Value

A list with

| | |
|---|---|
| lambda | parameter for the exponential distribution (for scale in units of cells), in the same parametrization as dexp |
| priorScale | matrix with x and y columns with prior of scale parameter |
| priorRange | matris with x and y columns with prior of range parameter, in meters (or original spatial units) |
| inla | character string specifying this prior in inla's format |

### Examples

```
# pr(range < 100km) = 0.1, 200m grid cells
 x = pcPriorRange(q=100*1000, p=0.1, cellSize = 200)
 rangeSeq = seq(0, 1000, len=1001)
 plot(rangeSeq, x$dprior$range(rangeSeq*1000)*1000,
   type='l', xlab="range, 1000's km", ylab='dens')
 cat(x$inla)
```

---

## postExp *Exponentiate posterior quantiles*

---

### Description

Converts a summary table for model parameters on the log scale to the natural or exponentiated scale.

### Usage

```
postExp(x,
 exclude = grep('^(range|aniso|shape|boxcox)', rownames(x)),
 invLogit=FALSE)
```

## Arguments

| | |
|---|---|
| x | a matrix or data frame as returned by glgm |
| exclude | vector of parameters not transformed, defaults to the range parameter |
| invLogit | Converts intercept parameter to inverse-logit scale when TRUE. Can also be a vector of parameters to inverse-logit transform. |

## Value

a summary table for log or exponentially transformed model parameters

## Examples

```
require("geostatsp")
data("swissRain")
swissRain = unwrap(swissRain)
swissAltitude = unwrap(swissAltitude)

swissRain$lograin = log(swissRain$rain)

if(inlaAvailable()) {
  inlaSetThreads(2)
swissFit =  glgm(formula="lograin", data=swissRain,
 grid=20,
 covariates=swissAltitude/1000, family="gaussian",
 prior = list(sd=1, range=100*1000, sdObs = 2),
 control.inla = list(strategy='gaussian', int.strategy='eb'),
 control.mode = list(theta=c(1.6542995, 0.7137123,2.2404179))
 )
if(length(swissFit$parameters)) {
 postExp(swissFit$parameters$summary)
}

}
```

---

# profLlgm *Joint confidence regions*

---

## Description

Calculates profile likelihoods and approximate joint confidence regions for covariance parameters in linear geostatistical models.

## Usage

```
profLlgm(fit, mc.cores = 1, ...)
informationLgm(fit,  ...)
```

## Arguments

| | |
|---|---|
| fit | Output from the lgm function |
| mc.cores | Passed to mclapply |
| ... | For profLlgm, one or more vectors of parameter values at which the profile likelihood will be calculated, with names corresponding to elements of fit$param. For informationLgm, arguments passed to hessian |

## Value

one or more vectors of parameter values

| | |
|---|---|
| logL | A vector, matrix, or multi-dimensional array of profile likelihood values for every combination of parameter values supplied. |
| full | Data frame with profile likelihood values and estimates of model parameters |
| prob, breaks | vector of probabilities and chi-squared derived likelihood values associated with those probabilities |
| MLE, maxLogL | Maximum Likelihood Estimates of parameters and log likelihood evaluated at these values |
| basepars | combination of starting values for parameters re-estimated for each profile likelihood and values of parameters which are fixed. |
| col | vector of colours with one element fewer than the number of probabilities |
| ci, ciLong | when only one parameter is varying, a matrix of confidence intervals (in both wide and long format) is returned. |

## Author(s)

Patrick Brown

## See Also

lgm, mcmapply, hessian

## Examples

```
# this example is time consuming
# the following 'if' statement ensures the CRAN
# computer doesn't run it
if(interactive() | Sys.info()['user'] =='patrick') {

library('geostatsp')
data('swissRain')
swissRain = unwrap(swissRain)
swissAltitude = unwrap(swissAltitude)

swissFit = lgm(data=swissRain, formula=rain~ CHE_alt,
  grid=10, covariates=swissAltitude,
  shape=1,  fixShape=TRUE,
```

```
  boxcox=0.5, fixBoxcox=TRUE,
  aniso=TRUE,reml=TRUE,
  param=c(anisoAngleDegrees=37,anisoRatio=7.5,
  range=50000))


x=profLlgm(swissFit,
  anisoAngleDegrees=seq(30, 43 , len=4)
)


plot(x[[1]],x[[2]], xlab=names(x)[1],
  ylab='log L',
  ylim=c(min(x[[2]]),x$maxLogL),
  type='n')
abline(h=x$breaks[-1],
  col=x$col,
  lwd=1.5)
axis(2,at=x$breaks,labels=x$prob,line=-1.2,
 tick=FALSE,
  las=1,padj=1.2,hadj=0)
abline(v=x$ciLong$par,
  lty=2,
  col=x$col[as.character(x$ciLong$prob)])
lines(x[[1]],x[[2]], col='black')



}
```

---

RFsimulate *Simulation of Random Fields*

---

## Description

This function simulates conditional and unconditional Gaussian random fields, calling the function in the RandomFields package of the same name.

## Usage

```
## S4 method for signature 'ANY,SpatRaster'
RFsimulate(model, x, data=NULL,
 err.model=NULL, n = 1, ...)
## S4 method for signature 'numeric,SpatRaster'
RFsimulate(model, x,data=NULL,
 err.model=NULL, n = 1, ...)
## S4 method for signature 'numeric,SpatVector'
RFsimulate(model, x, data=NULL,
```

```
err.model=NULL, n = 1, ...)
## S4 method for signature 'RMmodel,SpatRaster'
RFsimulate(model, x, data=NULL,
 err.model=NULL, n = 1, ...)
## S4 method for signature 'RMmodel,SpatVector'
RFsimulate(model, x, data=NULL,
 err.model=NULL, n = 1, ...)
## S4 method for signature 'matrix,SpatRaster'
RFsimulate(model, x,  data=NULL,
 err.model=NULL, n = nrow(model), ...)
## S4 method for signature 'matrix,SpatVector'
RFsimulate(model, x, data=NULL,
 err.model=NULL, n = nrow(model), ...)
## S4 method for signature 'data.frame,ANY'
RFsimulate(model, x, data=NULL,
 err.model=NULL, n = nrow(model), ...)
modelRandomFields(param, includeNugget=FALSE)
```

## Arguments

| | |
|---|---|
| model | object of class RMmodel, a vector of named model parameters, or a matrix where each column is a model parameter |
| x | Object of type SpatRaster or SpatVector. |
| data | For conditional simulation and random imputing only. If data is missing, unconditional simulation is performed. Object of class SpatVector; coordinates and response values of measurements in case that conditional simulation is to be performed |
| err.model | For conditional simulation and random imputing only. Usually err.model=RMnugget(var=var), or not given at all (error-free measurements). |
| n | number of realizations to generate. |
| ... | for advanced use: further options and control parameters for the simulation that are passed to and processed by RFoptionsin the RandomFieldspackage |
| param | A vector of named parameters |
| includeNugget | If FALSE, the nugget parameter is ignored. |

## Details

If model is a matrix, a different set of parameters is used for each simulation. If data has the same number of columns as model has rows, a different column i is used with parameters in row i.

## Value

An object of the same class as x.

## Author(s)

Patrick E. Brown <patrick.brown@utoronto.ca>

## See Also

RFsimulatein the RandomFieldspackage

## Examples

```
# the following prevents some errors on the cran check
Sys.unsetenv(c("OMP_THREAD_LIMIT", "KMP_DEVICE_THREAD_LIMIT", "KMP_TEAMS_THREAD_LIMIT"))
Sys.setenv(
  OMP_NUM_THREADS = "2",
  OMP_DYNAMIC = "FALSE",
  OPENBLAS_NUM_THREADS = "2",
  MKL_NUM_THREADS = "2",
  MKL_DYNAMIC = "FALSE",
  BLIS_NUM_THREADS = "2",
  VECLIB_MAXIMUM_THREADS = "2"
)
options(mc.cores = 2)


library('geostatsp')

# exclude this line to use the RandomFields package
options(useRandomFields = FALSE)

model1 <- c(var=5, range=1,shape=0.5)


myraster = rast(nrows=20,ncols=30,extent = ext(0,6,0,4),
 crs="+proj=utm +zone=17 +datum=NAD27 +units=m +no_defs")

set.seed(0)

simu <- RFsimulate(model1, x=myraster, n=3)

plot(simu[['sim2']])

xPoints = suppressWarnings(as.points(myraster))
# conditional simulation
firstSample = RFsimulate(
 c(model1, nugget=1),
 x=xPoints[seq(1,ncell(myraster), len=100), ],
 n=3
)

secondSample = RFsimulate(
 model = cbind(var=5:3, range=seq(0.05, 0.25, len=3), shape=seq(0.5, 1.5, len=3)),
  err.model = 1,
 x= myraster,
 data=firstSample,n=4
)

plot(secondSample)
```

rongelapUTM *Rongelap data*

## 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 $\gamma$-ray counts at 157 locations.

## Usage

```
data(rongelapUTM)
```

## Format

A SpatVector, with columns count being the radiation count and time being the length of time the measurement was taken for. A UTM coordinate reference system is used, where coordinates are in metres.

## Source

https://web.archive.org/web/20240110054727/http://www.leg.ufpr.br/doku.php/pessoais:paulojus:mbgbook:datasets. 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.

## Examples

```
data("rongelapUTM")
rongelapUTM = unwrap(rongelapUTM)
plot(rongelapUTM, main="Rongelap island")

if(require('mapmisc')) {
  bgMap = openmap(rongelapUTM, buffer=300, maxTiles=2)
  plot(bgMap)
```

```
  points(rongelapUTM, cex=0.4)
  scaleBar(rongelapUTM, 'left')

}
```

---

simLgcp *Simulate a log-Gaussian Cox process*

---

## Description

Give covariates and model parameters, simulates a log-Gaussian Cox process

## Usage

```
simLgcp(param, covariates=NULL, betas=NULL,
offset=NULL,
  rasterTemplate=covariates[[1]], n=1, ...)
simPoissonPP(intensity)
```

## Arguments

| | |
|---|---|
| param | A vector of named model parameters with, at a minimum names range and shape (see Details), and optionally variance (defaults to 1). For Geometric Anisotropy add anisoRatio and either anisoAngleDegrees or anisoAngleRadians |
| covariates | Either a multi-layer SpatRaster or list of SpatRasters and SpatVectors (with the latter having only a single data column). |
| betas | Coefficients for the covariates |
| offset | Vector of character strings corresponding to elements of covariates which are offsets |
| rasterTemplate | Raster on which the latent surface is simulated, defaults to the first covariate. |
| n | number of realisations to simulate |
| ... | additional arguments, see RFsimulatein the RandomFieldspackage. |
| intensity | Raster of the intensity of a Poisson point process. |

## Value

A list with a events element containing the event locations and a SpatRaster element containing a multi-layer SpatRaster of the covariates, spatial random effect, and intensity.

## Examples

```
mymodel = c(mean=-0.5, variance=0.4^2,
    range=2, shape=2)

myraster = rast(nrows=15,ncols=20,xmin=0,xmax=10,ymin=0,ymax=7.5)

# some covariates, deliberately with a different resolution than myraster
covA = covB = myoffset = rast(ext(myraster), 10, 10)
values(covA) = as.vector(matrix(1:10, 10, 10))
values(covB) = as.vector(matrix(1:10, 10, 10, byrow=TRUE))
values(myoffset) = round(seq(-1, 1, len=ncell(myoffset)))

myCovariate = list(a=covA, b=covB, offsetFooBar = myoffset)

myLgcp=simLgcp(param=mymodel,
 covariates=myCovariate,
 betas=c(a=-0.1, b=0.25),
 offset='offsetFooBar',
 rasterTemplate=myraster)

plot(myLgcp$raster[["intensity"]], main="lgcp")
points(myLgcp$events)

myIntensity = exp(-1+0.2*myCovariate[["a"]])
myPoissonPP = simPoissonPP(myIntensity)[[1]]
plot(myIntensity, main="Poisson pp")
points(myPoissonPP)
```

---

`spatialRoc` *Sensitivity and specificity*

---

## Description

Calculate ROC curves using model fits to simulated spatial data

## Usage

```
spatialRoc(fit, rr = c(1, 1.2, 1.5, 2), truth, border=NULL,
 random = FALSE, prob = NULL, spec = seq(0,1,by=0.01))
```

## Arguments

| | |
|---|---|
| `fit` | A fitted model from the `lgcp` function |
| `rr` | Vector of relative risks exceedance probabilities will be calculated for. Values are on the natural scale, with `spatialRoc` taking logs when appropriate. |
| `truth` | True value of the spatial surface, or result from `simLgcp` function. Assumed to be on the log scale if `random=TRUE` and on the natural scale otherwise. |

| | |
|---|---|
| border | optional, SpatVector specifying region that calculations will be restricted to. |
| random | compute ROC's for relative intensity (FALSE) or random effect (TRUE) |
| prob | Vector of exceedance probabilities |
| spec | Vector of specificities for the resulting ROC's to be computed for. |

## Details

Fitted models are used to calculate exceedance probabilities, and a location is judged to be above an rr threshold if this exceedance probability is above a specified probability threshold. Each raster cell of the true surface is categorized as being either true positive, false positive, true negative, and false negative and sensitivity and specificity computed. ROC curves are produced by varying the probability threshold.

## Value

An array, with dimension 1 being probability threshold, dimension 2 being the relative risk threshold, dimension 3 being sensitivity and specificity. If fit is a list of model fits, dimension 4 corresponds to elements of fit.

## Author(s)

Patrick Brown

## See Also

lgcp, simLgcp, excProb

---

# squareRaster-methods *Create a raster with square cells*

---

## Description

Given a raster object, an extent, or a bounding box, a raster of with square cells and having the extent and number of cells specified is returned.

## Usage

```
## S4 method for signature 'matrix'
squareRaster(x,cells=NULL, buffer=0)
## S4 method for signature 'SpatRaster'
squareRaster(x,cells=NULL, buffer=0)
## S4 method for signature 'SpatVector'
squareRaster(x,cells=NULL, buffer=0)
## S4 method for signature 'SpatExtent'
squareRaster(x,cells=NULL, buffer=0)
```

### Arguments

| | |
|---|---|
| x | A spatial object |
| cells | The number of cells in the x direction. If NULL the number of columns of x is used. |
| buffer | Additional area to add around the resulting raster |

### Value

A SpatRaster with square cells

### Examples

```
myraster = rast(matrix(0,10,10),extent=c(0,10,0,12.3))

squareRaster(myraster)

squareRaster(myraster, buffer=3, cells=20)

squareRaster(ext(myraster), cells=10)
```

---

## stackRasterList *Converts a list of SpatRasters, possibly with different projections and resolutions, to a single multi-layer SpatRaster.*

---

### Description

This function is intended to be used prior to passing covariates to krigeLgm in order for the rasters for all covariates to have the same projection and same resolution.

### Usage

```
stackRasterList(x, template = x[[1]], method = "near", mc.cores=NULL)
spdfToBrick(x,
    template,
    logSumExpected=FALSE,
    pattern = '^expected_[[:digit:]]+$'
)
```

### Arguments

| | |
|---|---|
| x | A list of SpatRaster or SpatVectors for stackRasterList and spdfToBrick respectively |
| template | A raster whose projection and resolution all other rasters will be aligned with. |

| | |
|---|---|
| method | The method to use, either "near", or "bilinear". Can be a vector of the same length as x to specify different methods for each raster. If method has names which correspond to the names of x, the names will be used instead of the order to assign methods to rasters. |
| mc.cores | If non-null, mclapply is used with this argument specifying the number of cores. |
| logSumExpected | return the log of the sum of offsets |
| pattern | expression to identify layers to rasterize in x |

## Value

A multi-layer SpatRaster, with one layer for each variable.

## Examples

```
myCrs = crs("+proj=utm +zone=17 +ellps=GRS80 +units=m +no_defs")
x = list(a=rast(matrix(1:9, 3, 3), extent=ext(0,1,0,1),
       crs=myCrs),
 b=rast(matrix(1:25, 5, 5), extent=ext(-1, 2, -1, 2),
    crs=myCrs)
 )
mystack = stackRasterList(x)
mystack



mylist = list(
 a=rast(matrix(1:36, 6, 6,byrow=TRUE), extent=ext(0,1000,0,1000),
       crs=myCrs),
 b=rast(matrix(1:144, 12, 12), extent=ext(-200, 200, -200, 200),
    crs=myCrs),
 c=rast(matrix(1:100, 10, 10), extent=ext(-5000,5000,-5000,5000),
       crs=myCrs)
 )

 mystack = stackRasterList(mylist, mc.cores=1)
 mystack

plot(mystack[["b"]], main="stack b")
plot(mystack[['a']],add=TRUE,col=grey(seq(0,1,len=12)),alpha=0.8,legend=FALSE)
```

---

# swissRain *Swiss rainfall data*

---

## Description

Data from the SIC-97 project: Spatial Interpolation Comparison.

## Usage

```
data("swissRain")
```

## Format

swissRain is a SpatVector 100 daily rainfall measurements made in Switzerland on the 8th of May 1986. swissAltitude is a raster of elevation data, and swissLandType is a raster of land cover types.

## Source

https://web.archive.org/web/20241008015622/https://wiki.52north.org/AI_GEOSTATS/AI_GEOSTATSData and https://srtm.csi.cgiar.org and https://web.archive.org/web/20241129120557/https://lpdaac.usgs.gov/data/

## Examples

```
data("swissRain")
swissRain = unwrap(swissRain)
swissAltitude = unwrap(swissAltitude)
swissBorder = unwrap(swissBorder)
swissLandType = unwrap(swissLandType)
plot(swissAltitude, main="elevation")
points(swissRain)
plot(swissBorder, add=TRUE)


# land type, a categorical variable
commonValues  = sort(table(values(swissLandType)),decreasing=TRUE)[1:5]
commonValues=commonValues[!names(commonValues)==0]

thelevels = levels(swissLandType)[[1]]$ID
thebreaks = c(-0.5, 0.5+thelevels)
thecol = rep(NA, length(thelevels))
names(thecol) = as.character(thelevels)

thecol[names(commonValues)] = rainbow(length(commonValues))

plot(swissLandType, breaks=thebreaks, col=thecol,legend=FALSE,
 main="land type")
points(swissRain)
plot(swissBorder, add=TRUE)


legend("left",fill=thecol[names(commonValues)],
  legend=substr(levels(swissLandType)[[1]][
      match(as.integer(names(commonValues)),
        levels(swissLandType)[[1]]$ID),
      "Category"], 1,12),
      bg= 'white'
    )
```

## swissRainR *Raster of Swiss rain data*

### Description

A raster image of Swiss rain and elevation, and a nearest neighbour matrix corresponding to this raster.

### Usage

```
data(swissRainR)
```

### Format

`swissRainR` is a `SpatRaster` (packed) of Swiss elevation and precipitation, and `swissNN` is a matrix of nearest neighbours.

### Source

See examples

### Examples

```
data('swissRainR')
swissRainR = unwrap(swissRainR)
plot(swissRainR[['prec7']])
plot(swissRainR[['alt']])

swissNN[1:4,1:5]
```

## variog *Compute Empirical Variograms and Permutation Envelopes*

### Description

These are wrappers for `variog`in the `geoR`package and `variog.mc.env` in the `geoR` package.

### Usage

```
variog(geodata, ...)
## S3 method for class 'SpatVector'
variog(geodata, formula, ...)
## Default S3 method:
variogMcEnv(geodata, ...)
## S3 method for class 'SpatVector'
variogMcEnv(geodata, formula, ...)
```

## Arguments

| | |
|---|---|
| geodata | An object of class SpatVector or of a class suitable for variog in the geoR package |
| formula | A formula specifying the response variable and fixed effects portion of the model. The variogram is performed on the residuals. |
| ... | additional arguments passed to variog in the geoR package |

## Value

As variog in the geoR package and variog.mc.env in the geoR package

## See Also

variogin the geoRpackage and variog.mc.env in the geoR package

## Examples

```
data("swissRain")
swissRain = unwrap(swissRain)
swissRain$lograin = log(swissRain$rain)
swissv= variog(swissRain, formula=lograin ~ 1,option="bin")
swissEnv = variogMcEnv(swissRain, lograin ~ 1, obj.var=swissv,nsim=9)
if(!is.null(swissv)){
 plot(swissv, env=swissEnv, main = "Swiss variogram")
}
```

---

# wheat *Mercer and Hall wheat yield data*

---

## Description

Mercer and Hall wheat yield data, based on version in Cressie (1993), p. 455.

## Usage

```
data(wheat)
```

## Format

wheat is a raster where the values refer to wheat yields.

## Examples

```
 data("wheat")
 wheat = unwrap(wheat)
 plot(wheat, main="Mercer and Hall Data")
```

# Index


∗ **datasets**
gambiaUTM, 5
loaloa, 22
murder, 30
rongelapUTM, 38
swissRain, 43
swissRainR, 45
wheat, 46
∗ **models**
conditionalGmrf, 2
spatialRoc, 40
∗ **spatial**
RFsimulate, 35

conditionalGmrf, 2

elevationLoa *(*loaloa*)*, 22
eviLoa *(*loaloa*)*, 22
excProb, 3, *41*

fillParam *(*matern*)*, 24

gambiaUTM, 5
geostatData, 6
glgm, *6, 7*, *33*
glgm *(*glgm-methods*)*, 7
glgm,ANY,ANY,ANY,ANY-method *(*glgm-methods*)*, 7
glgm,formula,data.frame,SpatRaster,data.frame-method *(*glgm-methods*)*, 7
glgm,formula,SpatRaster,ANY,ANY-method *(*glgm-methods*)*, 7
glgm,formula,SpatVector,ANY,ANY-method *(*glgm-methods*)*, 7
glgm-methods, 7

hessian, *34*

informationLgm *(*profLlgm*)*, 33
inla.models, 11
inlaAvailable, 11
inlaSetThreads *(*inlaAvailable*)*, 11

krigeLgm, 12, *17*, *42*

lgcp, *40, 41*
lgcp *(*glgm-methods*)*, 7
lgm, *3, 4*, *6, 7*, *13*, *20*, *34*
lgm *(*lgm-methods*)*, 14
lgm,character,ANY,ANY,ANY-method *(*lgm-methods*)*, 14
lgm,formula,data.frame,SpatRaster,data.frame-method *(*lgm-methods*)*, 14
lgm,formula,SpatRaster,ANY,ANY-method *(*lgm-methods*)*, 14
lgm,formula,SpatVector,numeric,ANY-method *(*lgm-methods*)*, 14
lgm,formula,SpatVector,SpatRaster,data.frame-method *(*lgm-methods*)*, 14
lgm,formula,SpatVector,SpatRaster,list-method *(*lgm-methods*)*, 14
lgm,formula,SpatVector,SpatRaster,missing-method *(*lgm-methods*)*, 14
lgm,formula,SpatVector,SpatRaster,SpatRaster-method *(*lgm-methods*)*, 14
lgm,missing,ANY,ANY,ANY-method *(*lgm-methods*)*, 14
lgm,numeric,ANY,ANY,ANY-method *(*lgm-methods*)*, 14
lgm-methods, 14
likfitLgm, *12*, *17*, 18
loaloa, 22
loglikLgm *(*likfitLgm*)*, 18
ltLoa *(*loaloa*)*, 22

mask, *8*
matern, *19*, 24
maternGmrfPrec, *3*, 27
mclapply, *3*, *12*, *17*, *34*, *43*
mcmapply, *34*
modelRandomFields *(*RFsimulate*)*, 35

murder, 30

NNmat *(maternGmrfPrec)*, 27

optim, *17*, *19*

pcPrior *(pcPriorRange)*, 31
pcPriorRange, 31
postExp, 32
profLlgm, 33

RFsimulate, 35
RFsimulate,ANY,SpatRaster-method *(RFsimulate)*, 35
RFsimulate,data.frame,ANY-method *(RFsimulate)*, 35
RFsimulate,matrix,SpatRaster-method *(RFsimulate)*, 35
RFsimulate,matrix,SpatVector-method *(RFsimulate)*, 35
RFsimulate,numeric,SpatRaster-method *(RFsimulate)*, 35
RFsimulate,numeric,SpatVector-method *(RFsimulate)*, 35
RFsimulate,RMmodel,SpatRaster-method *(RFsimulate)*, 35
RFsimulate,RMmodel,SpatVector-method *(RFsimulate)*, 35
RFsimulate-methods *(RFsimulate)*, 35
rongelapUTM, 38

simLgcp, 39, *40*, *41*
simPoissonPP *(simLgcp)*, 39
spatialRoc, 40
SpatRaster, *12*, *16*, *36*
SpatVector, *36*
spdfToBrick *(stackRasterList)*, 42
squareRaster, *6*, *7*
squareRaster *(squareRaster-methods)*, 41
squareRaster,matrix-method *(squareRaster-methods)*, 41
squareRaster,SpatExtent-method *(squareRaster-methods)*, 41
squareRaster,SpatRaster-method *(squareRaster-methods)*, 41
squareRaster,SpatVector-method *(squareRaster-methods)*, 41
squareRaster-methods, 41
stackRasterList, 42

swissAltitude *(swissRain)*, 43
swissBorder *(swissRain)*, 43
swissLandType *(swissRain)*, 43
swissNN *(swissRainR)*, 45
swissRain, 43
swissRainR, 45

tempLoa *(loaloa)*, 22
torontoBorder *(murder)*, 30
torontoIncome *(murder)*, 30
torontoNight *(murder)*, 30
torontoPdens *(murder)*, 30

variog, 45
variogMcEnv *(variog)*, 45

wheat, 46