Package ‘lgcp’

May 17, 2022

Maintainer  Benjamin M. Taylor <benjamin.taylor.software@gmail.com>
License  GPL-2 | GPL-3
Title  Log-Gaussian Cox Process
Type  Package
LazyLoad  yes
Author  Benjamin M. Taylor, Tilman M. Davies, Barry S. Rowlingson, Peter J. Diggle. Additional code contributions from Edzer Pebesma, Dominic Schumacher.
Version  1.8
Date  2022-05-16
Imports  spatstat.geom, spatstat.core, spatstat.utils, sp, raster, tcltk, iterators, ncdf4, methods, rpanel (>= 1.1-3), fields, maptools, Matrix, rgeos
Suggests  sparr, rgdal, gpclib
RoxygenNote  7.2.0
Encoding  UTF-8
NeedsCompilation  no
Depends  R (>= 2.10)
Repository  CRAN
Date/Publication  2022-05-17 09:20:05 UTC

R topics documented:

  lgcp-package ................................................................. 9
  .onAttach ................................................................. 11
  add.list ................................................................. 12
R topics documented:

addTemporalCovariates .................................................. 12
affine.fromFunction ...................................................... 13
affine.fromSPDF .......................................................... 14
affine.fromXYZ ............................................................ 14
affine.SpatialPolygonsDataFrame ....................................... 15
affine.stpp ............................................................... 15
aggCovInfo ................................................................. 16
aggCovInfo.ArealWeightedMean ......................................... 16
aggCovInfo.ArealWeightedSum ........................................... 17
aggCovInfo.Majority ....................................................... 17
aggregateCovariateInfo .................................................. 18
aggregateformulaList ..................................................... 18
andrieuthomsh .............................................................. 19
as.array.lgcpgrid .......................................................... 20
as.fromXYZ ................................................................. 20
as.fromXYZ/fromFunction ................................................. 21
as.im/fromFunction ......................................................... 22
as.im/fromSPDF ............................................................ 22
as.im/fromXYZ ............................................................. 23
as.list.lgcpgrid ............................................................ 24
as.owin.stapp .............................................................. 24
as.owinlist ................................................................. 25
as.owinlist.SpatialPolygonsDataFrame ................................. 25
as.owinlist.stapp .......................................................... 26
as.ppp.mstppp .............................................................. 26
as.ppp.stppp ............................................................... 27
as.SpatialGridDataFrame .................................................. 27
as.SpatialGridDataFrame.fromXYZ ....................................... 28
as.SpatialPixelsDataFrame .............................................. 28
as.SpatialPixelsDataFrame.lgcpgrid .................................... 29
as.stpp .......................................................... 29
as.stppp.stapp ........................................................... 30
assigninterp .............................................................. 30
at ................................................................. 32
autocorr ................................................................. 32
autocorrMultitype ........................................................ 33
BetaParameters ............................................................ 34
betavals ................................................................. 34
blockcircbase ............................................................. 35
blockcircbaseFunction .................................................... 36
bt.scalar ................................................................. 36
checkObsWin ............................................................. 37
chooseCellwidth .......................................................... 37
circulant ................................................................. 38
circulant.matrix .......................................................... 39
circulant.numeric ........................................................ 39
clearinterp ............................................................... 40
computeGradtruncSpatial ................................................. 40
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>computeGradtruncSpatioTemporal</td>
<td>41</td>
</tr>
<tr>
<td>condProbs</td>
<td>42</td>
</tr>
<tr>
<td>constant</td>
<td>43</td>
</tr>
<tr>
<td>constantInTime</td>
<td>44</td>
</tr>
<tr>
<td>constantInTime.numeric</td>
<td>45</td>
</tr>
<tr>
<td>constantInTime.stppp</td>
<td>46</td>
</tr>
<tr>
<td>cov.interp.fft</td>
<td>46</td>
</tr>
<tr>
<td>CovarianceFct</td>
<td>47</td>
</tr>
<tr>
<td>covEffects</td>
<td>48</td>
</tr>
<tr>
<td>CovFunction</td>
<td>49</td>
</tr>
<tr>
<td>CovFunction.function</td>
<td>49</td>
</tr>
<tr>
<td>CovParameters</td>
<td>50</td>
</tr>
<tr>
<td>Cvb</td>
<td>50</td>
</tr>
<tr>
<td>d.func</td>
<td>51</td>
</tr>
<tr>
<td>density.stppp</td>
<td>52</td>
</tr>
<tr>
<td>discreteWindow</td>
<td>52</td>
</tr>
<tr>
<td>discreteWindow.lgcpPredict</td>
<td>53</td>
</tr>
<tr>
<td>dump2dir</td>
<td>53</td>
</tr>
<tr>
<td>eigenfrombase</td>
<td>54</td>
</tr>
<tr>
<td>etavals</td>
<td>54</td>
</tr>
<tr>
<td>EvaluatePrior</td>
<td>55</td>
</tr>
<tr>
<td>exceedProbs</td>
<td>56</td>
</tr>
<tr>
<td>exceedProbsAggregated</td>
<td>56</td>
</tr>
<tr>
<td>expectation</td>
<td>57</td>
</tr>
<tr>
<td>expectation.lgcpPredict</td>
<td>58</td>
</tr>
<tr>
<td>expectation.lgcpPredictSpatialOnlyPlusParameters</td>
<td>59</td>
</tr>
<tr>
<td>exponentialCovFct</td>
<td>59</td>
</tr>
<tr>
<td>extendspatialAtRisk</td>
<td>60</td>
</tr>
<tr>
<td>extract</td>
<td>60</td>
</tr>
<tr>
<td>extract.lgcpPredict</td>
<td>61</td>
</tr>
<tr>
<td>Extract.mstppp</td>
<td>62</td>
</tr>
<tr>
<td>Extract.stppp</td>
<td>62</td>
</tr>
<tr>
<td>fftgrid</td>
<td>63</td>
</tr>
<tr>
<td>ftfitinterpolate</td>
<td>64</td>
</tr>
<tr>
<td>ftfitinterpolate.fromFunction</td>
<td>64</td>
</tr>
<tr>
<td>ftfitinterpolate.fromSPDF</td>
<td>65</td>
</tr>
<tr>
<td>ftfitinterpolate.fromXYZ</td>
<td>66</td>
</tr>
<tr>
<td>fftmultiply</td>
<td>66</td>
</tr>
<tr>
<td>formulaList</td>
<td>67</td>
</tr>
<tr>
<td>GAfinalise</td>
<td>67</td>
</tr>
<tr>
<td>GAfinalise.MonteCarloAverage</td>
<td>68</td>
</tr>
<tr>
<td>GAfinalise.nullAverage</td>
<td>68</td>
</tr>
<tr>
<td>GAinitialise</td>
<td>69</td>
</tr>
<tr>
<td>GAinitialise.MonteCarloAverage</td>
<td>70</td>
</tr>
<tr>
<td>GAinitialise.nullAverage</td>
<td>70</td>
</tr>
<tr>
<td>GammafromY</td>
<td>71</td>
</tr>
<tr>
<td>GAreturnvalue</td>
<td>71</td>
</tr>
<tr>
<td>GAreturnvalue.MonteCarloAverage</td>
<td>72</td>
</tr>
</tbody>
</table>
R topics documented:

- GAReturnvalue.nullAverage
- GAUpdate
- GAUpdate.MonteCarloAverage
- GAUpdate.nullAverage
- GaussianPrior
- genFFTgrid
- getCellCounts
- getCounts
- getCovParameters
- getCovParameters.GPrealisation
- getCovParameters.list
- getinterp
- getLHsformulaList
- getpolyol
- getRotation
- getRotation.default
- getRotation.stppp
- getup
- getZmat
- getZmats
- GFfinalise
- GFfinalise.dump2dir
- GFfinalise.nullFunction
- GFinitialise
- GFinitialise.dump2dir
- GFinitialise.nullFunction
- GFreturnvalue
- GFreturnvalue.dump2dir
- GFreturnvalue.nullFunction
- GFupdate
- GFupdate.dump2dir
- GFupdate.nullFunction
- ginhomAverage
- gOverlay
- GPdrv
- GPdrv2
- GPdrv2_Multitype
- GPlist2array
- GPrealisation
- grid2spdf
- grid2spix
- grid2spoly
- grid2spts
- gridav
- gridav.lgcpPredict
- gridfun
- gridfun.lgcpPredict
R topics documented:

lgcpSim .......................................................... 154
lgcpSimMultitypeSpatialCovariates ............................. 157
lgcpSimSpatial .................................................... 158
lgcpSimSpatialCovariates ................................. 159
lgcpvignette ............................................... 160
loc2poly .................................................. 161
LogGaussianPrior ............................................. 161
loop.mcmc ................................................. 162
ltar ..................................................... 162
MALAlgcp ................................................ 163
MALAlgcpAggregateSpatial.PlusPars ...................... 165
MALAlgcpMultitypeSpatial.PlusPars ...................... 166
MALAlgcpSpatial ........................................... 168
MALAlgcpSpatial.PlusPars .................................. 170
MALAlgcpSpatioTemporal.PlusPars ....................... 171
matchcovariance .............................................. 173
maternCovFct15 ............................................ 174
maternCovFct25 ............................................ 175
mcmcLoop .................................................. 175
mcmcpars ............................................... 176
mcmcProgressNone ......................................... 176
mcmcProgressPrint .......................................... 177
mcmcProgressTextBar ....................................... 177
mcmcProgressTk .......................................... 178
mcmctrace .................................................. 178
mcmctrace.lgcpPredict .................................... 179
meanfield .................................................. 179
meanfield.lgcpPredict .................................... 180
meanfield.lgcpPredictINLA ................................ 180
MonteCarloAverage .......................................... 181
mstppp ................................................... 182
mstppp.list .............................................. 183
mstppp.ppp ................................................ 183
mstppp.stppp ............................................. 184
muEst ..................................................... 184
multiply.list ............................................... 185
neatatable .................................................. 186
neigh2D .................................................... 186
nextStep .................................................... 187
nullAverage ............................................... 187
nullFunction .............................................. 188
numCases .................................................. 188
osppp2latlon .............................................. 189
osppp2merc ............................................... 189
paramprec .................................................. 190
paramprecbase ........................................... 190
parautocorr ............................................... 191
parsummary ................................................ 191
<table>
<thead>
<tr>
<th>R topics documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>segProbs</td>
</tr>
<tr>
<td>seintens</td>
</tr>
<tr>
<td>seintens.lgcP predict</td>
</tr>
<tr>
<td>selectObsWindow</td>
</tr>
<tr>
<td>selectObsWindow.default</td>
</tr>
<tr>
<td>selectObsWindow.stppp</td>
</tr>
<tr>
<td>serr</td>
</tr>
<tr>
<td>serr.lgcP predict</td>
</tr>
<tr>
<td>setoutput</td>
</tr>
<tr>
<td>setTxtProgressBar2</td>
</tr>
<tr>
<td>showGrid</td>
</tr>
<tr>
<td>showGrid.default</td>
</tr>
<tr>
<td>showGrid.lgcP predict</td>
</tr>
<tr>
<td>showGrid.stppp</td>
</tr>
<tr>
<td>smultiply.list</td>
</tr>
<tr>
<td>sparsebase</td>
</tr>
<tr>
<td>spatialAtRisk</td>
</tr>
<tr>
<td>spatialAtRisk.bivden</td>
</tr>
<tr>
<td>spatialAtRisk.default</td>
</tr>
<tr>
<td>spatialAtRisk.fromXYZ</td>
</tr>
<tr>
<td>spatialAtRisk.function</td>
</tr>
<tr>
<td>spatialAtRisk.im</td>
</tr>
<tr>
<td>spatialAtRisk.lgcPgrid</td>
</tr>
<tr>
<td>spatialAtRisk.SpatialGridDataFrame</td>
</tr>
<tr>
<td>spatialIntensities</td>
</tr>
<tr>
<td>spatialIntensities.fromSPDF</td>
</tr>
<tr>
<td>spatialIntensities.fromXYZ</td>
</tr>
<tr>
<td>spatialparsEst</td>
</tr>
<tr>
<td>SpatialPolygonsDataFrame.stapp</td>
</tr>
<tr>
<td>SpikedExponentialCovFct</td>
</tr>
<tr>
<td>stapp</td>
</tr>
<tr>
<td>stapp.list</td>
</tr>
<tr>
<td>stapp.SpatialPolygonsDataFrame</td>
</tr>
<tr>
<td>stGPrealisation</td>
</tr>
<tr>
<td>stppp</td>
</tr>
<tr>
<td>stppp.list</td>
</tr>
<tr>
<td>summary.lgcPgrid</td>
</tr>
<tr>
<td>summary.mcmc</td>
</tr>
<tr>
<td>target.and.grad.AggregateSpatialPlusPars</td>
</tr>
<tr>
<td>target.and.grad.MultitypespatialPlusPars</td>
</tr>
<tr>
<td>target.and.grad.spatial</td>
</tr>
<tr>
<td>target.and.grad.spatialPlusPars</td>
</tr>
<tr>
<td>target.and.grad.spatiotemporal</td>
</tr>
<tr>
<td>target.and.grad.SpatioTemporalPlusPars</td>
</tr>
<tr>
<td>temporalAtRisk</td>
</tr>
<tr>
<td>temporalAtRisk.function</td>
</tr>
</tbody>
</table>

Page 8
Description

An R package for spatiotemporal prediction and forecasting for log-Gaussian Cox processes.
Usage

lgcp

Format

An object of class logical of length 1.

Details

This package was not yet installed at build time.

Index: This package was not yet installed at build time.

For examples and further details of the package, type vignette("lgcp"), or refer to the paper associated with this package.

The content of lgcp can be broken up as follows:

Datasets wpopdata.rda, wtowncoords.rda, wtowns.rda. Giving regional and town populations as well as town coordinates, are provided by Wikipedia and The Office for National Statistics under respectively the Creative Commons Attribution-ShareAlike 3.0 Unported License and the Open Government Licence.

Data manipulation

Model fitting and parameter estimation

Unconditional and conditional simulation

Summary statistics, diagnostics and visualisation

Dependencies

The lgcp package depends upon some other important contributions to CRAN in order to operate; their uses here are indicated:

spatstat, sp, RandomFields, iterators, ncdf, methods, tcltk, rgl, rpanel, fields, rgdal, maptools, rgeos, raster

Citation

To see how to cite lgcp, type citation("lgcp") at the console.
**Author(s)**

Benjamin Taylor, Health and Medicine, Lancaster University. Tilman Davies, Institute of Fundamental Sciences - Statistics, Massey University, New Zealand. Barry Rowlingson, Health and Medicine, Lancaster University. Peter Diggle, Health and Medicine, Lancaster University.

**References**


---

**.onAttach**  

**.onAttach function**

---

**Description**

A function to print a welcome message on loading package

**Usage**

```
.onAttach(libname, pkgname)
```

**Arguments**

- `libname`  
  - libname argument
- `pkgname`  
  - pkgname argument

**Value**

...
add.list

Description
This function adds the elements of two list objects together and returns the result in another list object.

Usage
add.list(list1, list2)

Arguments
- list1: a list of objects that could be summed using "+
- list2: a list of objects that could be summed using "+

Value
a list with ith entry the sum of list1[[i]] and list2[[i]]

addTemporalCovariates

Description
A function to 'bolt on' temporal data onto a spatial covariate design matrix. The function takes a spatial design matrix, Z(s) and converts it to a spatiotemporal design matrix Z(s,t) when the effects can be separably decomposed i.e.,

\[ Z(s,t)\beta = Z_1(s)\beta_1 + Z_2(t)\beta_2 \]

An example of this function in action is given in the vignette "Bayesian_lgcp", in the section on spatiotemporal data.

Usage
addTemporalCovariates(temporal.formula, T, laglength, tdata, Zmat)

Arguments
- temporal.formula: a formula of the form \( t \sim tvar1 + tvar2 \) etc. Where the left hand side is a "t". Note there should not be an intercept term in both of the the spatial and temporal components.
- T: the time point of interest
affine.fromFunction

laglength  the number of previous time points to include in the analysis
tdata     a data frame with variable t minimally including times (T-laglength):T and var1, var2 etc.
Zmat      the spatial covariates Z(s), obtained by using the getZmat function.

Details

The main idea of this function is: having created a spatial Z(s) using getZmat, to create a dummy dataset tdata and temporal formula corresponding to the temporal component of the separable effects. The entries in the model matrix Z(s,t) corresponding to the time covariates are constant over the observation window in space, but in general vary from time-point to time-point.

Note that if there is an intercept in the spatial part of the model e.g., X ~ var1 + var2, then in the temporal model, the intercept should be removed i.e., t ~ tvar1 + tvar2 - 1

Value

A list of design matrices, one for each time, Z(s,t) for t in (T-laglength):T

See Also

chooseCellwidth, getpolyol, guessinterp, getZmat, lgcpPrior, lgcpInits, CovFunction lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatialTemporalPlusPars, lgcpPredictMultitypeSpatialPlusPars

Description

An affine transformation of an object of class fromFunction

Usage

## S3 method for class 'fromFunction'
affine(X, mat, ...)

Arguments

X       an object of class fromFunction
mat     matrix of affine transformation
...     additional arguments

Value

the object acted on by the transformation matrix
affine.fromSPDF  
**affine.fromSPDF function**

---

**Description**

An affine transformation of an object of class `fromSPDF`.

**Usage**

```r
## S3 method for class 'fromSPDF'
affine(X, mat, ...)
```

**Arguments**

- `X`: an object of class `fromSPDF`
- `mat`: matrix of affine transformation
- `...`: additional arguments

**Value**

The object acted on by the transformation matrix.

---

affine.fromXYZ  
**affine.fromXYZ function**

---

**Description**

An affine transformation of an object of class `fromXYZ`. Nearest Neighbour interpolation.

**Usage**

```r
## S3 method for class 'fromXYZ'
affine(X, mat, ...)
```

**Arguments**

- `X`: an object of class `fromFunction`
- `mat`: matrix of affine transformation
- `...`: additional arguments

**Value**

The object acted on by the transformation matrix.
affine.SpatialPolygonsDataFrame

affine.SpatialPolygonsDataFrame function

Description
An affine transformation of an object of class SpatialPolygonsDataFrame

Usage

## S3 method for class 'SpatialPolygonsDataFrame'
affine(X, mat, ...)

Arguments

X an object of class fromFunction
mat matrix of affine transformation
... additional arguments

Value
the object acted on by the transformation matrix

affine.stppp

affine.stppp function

Description
An affine transformation of an object of class stppp

Usage

## S3 method for class 'stppp'
affine(X, mat, ...)

Arguments

X an object of class stppp
mat matrix of affine transformation
... additional arguments

Value
the object acted on by the transformation matrix
aggCovInfo function

Description

Generic function for aggregation of covariate information.

Usage

```r
aggCovInfo(obj, ...)  
```

Arguments

- `obj`: an object
- `...`: additional arguments

Value

method `aggCovInfo`

---

aggCovInfo.ArealWeightedMean function

Description

Aggregation via weighted mean.

Usage

```r
## S3 method for class 'ArealWeightedMean'  
aggCovInfo(obj, regwts, ...)  
```

Arguments

- `obj`: an `ArealWeightedMean` object
- `regwts`: regional (areal) weighting vector
- `...`: additional arguments

Value

Areal weighted mean.
**Description**

Aggregation via weighted sum. Use to sum up population counts in regions.

**Usage**

```r
## S3 method for class 'ArealWeightedSum'
aggCovInfo(obj, regwts, ...)
```

**Arguments**

- `obj` an ArealWeightedSum object
- `regwts` regional (areal) weighting vector
- `...` additional arguments

**Value**

Areal weighted Sum.

---

**Description**

Aggregation via majority.

**Usage**

```r
## S3 method for class 'Majority'
aggCovInfo(obj, regwts, ...)
```

**Arguments**

- `obj` an Majority object
- `regwts` regional (areal) weighting vector
- `...` additional arguments

**Value**

The most popular cell type.
aggregateCovariateInfo

*aggregateCovariateInfo function*

**Description**

A function called by cov.interp.fft to allocate and perform interpolation of covariate information onto the FFT grid

**Usage**

```r
aggregateCovariateInfo(cellidx, cidx, gidx, df, fftovl, classes, polyareas)
```

**Arguments**

- `cellidx`: the index of the cell
- `cidx`: index of covariate, no longer used
- `gidx`: grid index
- `df`: the data frame containing the covariate information
- `fttovl`: an overlay of the fft grid onto the SpatialPolygonsDataFrame or SpatialPixelsDataFrame objects
- `classes`: vector of class attributes of the dataframe
- `polyareas`: polygon areas of the SpatialPolygonsDataFrame or SpatialPixelsDataFrame objects

**Value**

the interpolated covariate information onto the FFT grid

aggregateformulaList

*aggregateformulaList function*

**Description**

An internal function to collect terms from a formulalist. Not intended for general use.

**Usage**

```r
aggregateformulaList(x, ...)
```

**Arguments**

- `x`: an object of class "formulaList"
- `...`: other arguments
**Value**

a formula of the form $X \sim \text{var1} + \text{var2}$ tec.

---

**Description**

A Robbins-Munro stochastic approximation update is used to adapt the tuning parameter of the proposal kernel. The idea is to update the tuning parameter at each iteration of the sampler:

$$h^{(i+1)} = h^{(i)} + \eta^{(i+1)} (\alpha^{(i)} - \alpha_{opt}),$$

where $h^{(i)}$ and $\alpha^{(i)}$ are the tuning parameter and acceptance probability at iteration $i$ and $\alpha_{opt}$ is a target acceptance probability. For Gaussian targets, and in the limit as the dimension of the problem tends to infinity, an appropriate target acceptance probability for MALA algorithms is 0.574. The sequence $\{\eta^{(i)}\}$ is chosen so that $\sum_{i=0}^{\infty} \eta^{(i)}$ is infinite whilst $\sum_{i=0}^{\infty} (\eta^{(i)})^{1+\epsilon}$ is finite for $\epsilon > 0$. These two conditions ensure that any value of $h$ can be reached, but in a way that maintains the ergodic behaviour of the chain. One class of sequences with this property is,

$$\eta^{(i)} = \frac{C}{i^\alpha},$$

where $\alpha \in (0, 1]$ and $C > 0$. The scheme is set via the `mcmcpars` function.

**Usage**

```r
andrieuthomsh(inith, alpha, C, targetacceptance = 0.574)
```

**Arguments**

- `inith`: initial $h$
- `alpha`: parameter $\alpha$
- `C`: parameter $C$
- `targetacceptance`: target acceptance probability

**Value**

an object of class `andrieuthomsh`

**References**

See Also
mcmcpars, lgcpPredict

Examples
andrieuthomsh(inith=1, alpha=0.5, C=1, targetacceptance=0.574)

---

as.array.lgcpgrid  
as.array.lgcpgrid function

Description
Method to convert an lgcpgrid object into an array.

Usage
## S3 method for class 'lgcpgrid'
as.array(x, ...)

Arguments
x  
an object of class lgcpgrid
...
other arguments

Value
conversion from lgcpgrid to array

---

as.fromXYZ  
as.fromXYZ function

Description
Generic function for conversion to a fromXYZ object (eg as would have been produced by spatialAtRisk for example.)

Usage
as.fromXYZ(X, ...)

Arguments
X  
an object
...
additional arguments
as.fromXYZ.fromFunction

Value
generic function returning method as.fromXYZ

See Also
as.im.fromXYZ, as.im.fromSPDF, as.im.fromFunction, as.fromXYZ

as.fromXYZ.fromFunction

Description
Method for converting from the fromFunction class of objects to the fromXYZ class of objects. Clearly this requires the user to specify a grid onto which to compute the discretised version.

Usage
## S3 method for class 'fromFunction'
as.fromXYZ(X, xyt, M = 100, N = 100, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>an object of class fromFunction</td>
</tr>
<tr>
<td>xyt</td>
<td>and objects of class stppp</td>
</tr>
<tr>
<td>M</td>
<td>number of cells in x direction</td>
</tr>
<tr>
<td>N</td>
<td>number of cells in y direction</td>
</tr>
<tr>
<td>...</td>
<td>additional arguments</td>
</tr>
</tbody>
</table>

Value
object of class im containing normalised intensities

See Also
as.im.fromXYZ, as.im.fromSPDF, as.im.fromFunction, as.fromXYZ
as.im.fromFunction  

Description

Convert an object of class fromFunction (created by spatialAtRisk for example) into a spatstat im object.

Usage

## S3 method for class 'fromFunction'
as.im(X, xyt, M = 100, N = 100, ...)

Arguments

X an object of class fromSPDF
xyt and objects of class stppp
M number of cells in x direction
N number of cells in y direction
... additional arguments

Value

object of class im containing normalised intensities

See Also

as.im.fromXYZ, as.im.fromSPDF, as.im.fromFunction, as.fromXYZ

as.im.fromSPDF  

Description

Convert an object of class fromSPDF (created by spatialAtRisk for example) into a spatstat im object.

Usage

## S3 method for class 'fromSPDF'
as.im(X, ncells = 100, ...)
as.im.fromXYZ

Arguments

X an object of class fromSPDF
ncells number of cells to divide range into; default 100
... additional arguments

Value

object of class im containing normalised intensities

See Also

as.im.fromXYZ, as.im.fromSPDF, as.im.fromFunction, as.fromXYZ

as.im.fromXYZ

as.im.fromXYZ function

Description

Convert an object of class fromXYZ (created by spatialAtRisk for example) into a spatstat im object.

Usage

## S3 method for class 'fromXYZ'
as.im(X, ...)

Arguments

X object of class fromXYZ
... additional arguments

Value

object of class im containing normalised intensities

See Also

as.im.fromSPDF, as.im.fromFunction, as.fromXYZ
as.list.lgcpgrid function

Description
Method to convert an lgcpgrid object into a list of matrices.

Usage

## S3 method for class 'lgcpgrid'
as.list(x, ...)

Arguments

x 
an object of class lgcpgrid
...
other arguments

Value
conversion from lgcpgrid to list

See Also
lgcpgrid.list, lgcpgrid.array, print.lgcpgrid, summary.lgcpgrid, quantile.lgcpgrid, image.lgcpgrid, plot.lgcpgrid

as.owin.stapp function

Description
A function to extract the SpatialPolygons part of W and return it as an owin object.

Usage

## S3 method for class 'stapp'
as.owin(W, ..., fatal = TRUE)

Arguments

W 
see ?as.owin
...
see ?as.owin
fatal 
see ?as.owin

Value
an owin object
as.owinlist  

Description

Generic function for creating lists of owin objects

Usage

as.owinlist(obj, ...)

Arguments

obj an object

... additional arguments

Value

method as.owinlist

as.owinlist.SpatialPolygonsDataFrame

Description

A function to create a list of owin objects from a SpatialPolygonsDataFrame

Usage

## S3 method for class 'SpatialPolygonsDataFrame'
as.owinlist(obj, dmin = 0, check = TRUE, subset = rep(TRUE, length(obj)), ...)

Arguments

obj a SpatialPolygonsDataFrame object
dmin purpose is to simplify the SpatialPolygons. A numeric value giving the smallest permissible length of an edge. See ?simplify.owin
check whether or not to use spatstat functions to check the validity of SpatialPolygons objects
subset logical vector. Subset of regions to extract and convert to owin objects. By default, all regions are extracted.

... additional arguments

Value

a list of owin objects corresponding to the constituent Polygons objects
as.owinlist.stapp  

as.owinlist.stapp function

Description
A function to create a list of owin objects from a stapp

Usage

## S3 method for class 'stapp'
as.owinlist(obj, dmin = 0, check = TRUE, ...)

Arguments

obj an stapp object
dmin purpose is to simplify the SpatialPolygons. A numeric value giving the smallest permissible length of an edge. See ? simplify.owin
check whether or not to use spatstat functions to check the validity of SpatialPolygons objects
...
additional arguments

Value

a list of owin objects corresponding to the constituent Polygons objects

as.ppp.mstppp  

as.ppp.mstppp function

Description
Convert from mstppp to ppp. Can be useful for data handling.

Usage

## S3 method for class 'mstppp'
as.ppp(X, ..., fatal = TRUE)

Arguments

X an object of class mstppp
... additional arguments
fatal logical value, see details in generic ?as.ppp

Value

a ppp object without observation times
**as.ppp.stppp**

### as.ppp.stppp function

**Description**
Convert from stppp to ppp. Can be useful for data handling.

**Usage**
```r
## S3 method for class 'stppp'
as.ppp(X, ..., fatal = TRUE)
```

**Arguments**
- `X`: an object of class stppp
- `...`: additional arguments
- `fatal`: logical value, see details in generic `as.ppp`

**Value**
a ppp object without observation times

---

**as.SpatialGridDataFrame**

### as.SpatialGridDataFrame function

**Description**
Generic method for converting to an object of class SpatialGridDataFrame.

**Usage**
```r
as.SpatialGridDataFrame(obj, ...)
```

**Arguments**
- `obj`: an object
- `...`: additional arguments

**Value**
method as.SpatialGridDataFrame

**See Also**

`as.SpatialGridDataFrame.fromXYZ`
### as.SpatialGridDataFrame

**as.SpatialGridDataFrame.fromXYZ**

Method for converting objects of class fromXYZ into those of class SpatialGridDataFrame

#### Usage

```r
## S3 method for class 'fromXYZ'
as.SpatialGridDataFrame(obj, ...)
```

#### Arguments

- `obj`: an object of class spatialAtRisk
- `...`: additional arguments

#### Value

an object of class SpatialGridDataFrame

#### See Also

- as.SpatialGridDataFrame

---

### as.SpatialPixelsDataFrame

**as.SpatialPixelsDataFrame**

Generic function for conversion to SpatialPixels objects.

#### Usage

```r
as.SpatialPixelsDataFrame(obj, ...)
```

#### Arguments

- `obj`: an object
- `...`: additional arguments

#### Value

method as.SpatialPixels
### as.SpatialPixelsDataFrame.lgpgrid

**Description**
Method to convert lgpgrid objects to SpatialPixelsDataFrame objects.

**Usage**
```r
## S3 method for class 'lgpgrid'
as.SpatialPixelsDataFrame(obj, ...)
```

**Arguments**
- `obj`: an lgpgrid object
- `...`: additional arguments to be passed to SpatialPoints, eg a proj4string

**Value**
Either a SpatialPixelsDataFrame, or a list consisting of SpatialPixelsDataFrame objects.

---

### as.stppp

**Description**
Generic function for converting to stpp objects

**Usage**
```r
as.stppp(obj, ...)
```

**Arguments**
- `obj`: an object
- `...`: additional arguments

**Value**
method as.stppp
**as.stppp.stapp**  
*as.stppp.stapp function*

**Description**

A function to convert stapp objects to stppp objects for use in lgcpPredict. The regional counts in the stapp object are assigned a random location within each areal region proportional to a population density (if that is available) else the counts are distributed uniformly across the observation windows.

**Usage**

```r
## S3 method for class 'stapp'
as.stppp(obj, popden = NULL, n = 100, dmin = 0, check = TRUE, ...)
```

**Arguments**

- `obj` an object of class stapp
- `popden` a 'spatialAtRisk' of sub-class 'fromXYZ' object representing the population density, or for better results, lambda(s) can also be used here. Cases are distributed across the spatial region according to popden. NULL by default, which has the effect of assigning counts uniformly.
- `n` if popden is NULL, then this parameter controls the resolution of the uniform. Otherwise if popden is of class 'fromFunction', it controls the size of the imputation grid used for sampling. Default is 100.
- `dmin` If any reginal counts are missing, then a set of polygonal 'holes' in the observation window will be computed for each. dmin is the parameter used to control the simplification of these holes (see ?simplify.owin). default is zero.
- `check` logical. If any reginal counts are missing, then roughly speaking, check specifies whether to check the 'holes'.
- `...` additional arguments

**Value**

...
assigninterp

Usage

assigninterp(df, vars, value)

Arguments

df a data frame
vars character vector giving name of variables
value an interpolation type, possible options are given by typing interptypes(), see ?interptypes

Details

The three types of interpolation method employed in the package lgcp are:

1. 'Majority' The interpolated value corresponds to the value of the covariate occupying the largest area of the computational cell.
2. 'ArealWeightedMean' The interpolated value corresponds to the mean of all covariate values contributing to the computational cell weighted by their respective areas.
3. 'ArealWeightedSum' The interpolated value is the sum of all contributing covariates weighed by the proportion of area with respect to the covariate polygons. For example, suppose region A has the same area as a computational grid cell and has 500 inhabitants. If that region occupies half of a computational grid cell, then this interpolation type assigns 250 inhabitants from A to the computational grid cell.

Value

assigns an interpolation type to a variable

See Also

chooseCellwidth, getpolyol, guessinterp, getZmat, addTemporalCovariates, lgcpPrior, lgcpInits, CovFunction lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatioTemporalPlusPars, lgcpPredictMultitypeSpatialPlusPars

Examples

## Not run: spdf a SpatialPolygonsDataFrame
## Not run: spdf@data <- assigninterp(df=spdf@data,vars="pop",value="ArealWeightedSum")
at

description

at function

Usage

at(t, mu, theta)

Arguments

t change in time parameter, see Brix and Diggle (2001)
mu mean
theta parameter beta in Brix and Diggle

Value

...

autocorr

autocorr function

Description

This function requires data to have been dumped to disk: see ?dump2dir and ?setoutput. The routine autocorr.lgcpPredict computes cellwise selected autocorrelations of Y. Since computing the quantiles is an expensive operation, the option to output the quantiles on a subregion of interest is also provided (by setting the argument inWindow, which has a sensible default).

Usage

autocorr(
  x,
  lags,
  tidx = NULL,
  inWindow = x$xyt$window,
  crop2parentwindow = TRUE,
  ...
)
autocorrMultitype

Arguments

- **x**: an object of class `lgcpPredict`.  
- **lags**: a vector of the required lags.  
- **tidx**: the index number of the time interval of interest, default is the last time point.  
- **inWindow**: an observation owin window on which to compute the autocorrelations, can speed up calculation. Default is `x$sxyt$window`, set to `NULL` for full grid.  
- **crop2parentwindow**: logical: whether to only compute autocorrelations for cells inside `x$sxyt$window` (the 'parent window').  
- **...**: additional arguments

Value

An array, the [.,i]th slice being the grid of cell-wise autocorrelations.

See Also

`lgcpPredict`, `dump2dir`, `setoutput`, `plot.lgcpAutocorr`, `ltar`, `parautocorr`, `traceplots`, `parsummary`, `textsummary`, `priorpost`, `postcov`, `exceedProbs`, `betavals`, `etavals`

Description

A function to compute cell-wise autocorrelation in the latent field at specified lags

Usage

```r
autocorrMultitype(
  x,
  lags,
  fieldno,
  inWindow = x$sxyt$window,
  crop2parentwindow = TRUE,
  ...
)
```

Arguments

- **x**: an object of class `lgcpPredictMultitypeSpatialPlusParameters`.  
- **lags**: the lags at which to compute the autocorrelation.  
- **fieldno**: the index of the latent field, the i in Y_i, see the help file for `lgcpPredictMultitypeSpatialPlusParameters`. In diagnostic checking, this command should be called for each field in the model.
inWindow

an observation owin window on which to compute the autocorrelations, can speed up calculation. Default is x$xyt$window, set to NULL for full grid.

crop2parentwindow

logical: whether to only compute autocorrelations for cells inside x$xyt$window (the 'parent window')

... other arguments

Value

an array, the [,j]th slice being the grid of cell-wise autocorrelations.

BetaParameters

BetaParameters function

Description

An internal function to declare a vector a parameter vector for the main effects.

Usage

BetaParameters(beta)

Arguments

beta a vector

Value

...

betavals

betavals function

Description

A function to return the sampled beta from a call to the function lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatioTemporalPlusPars or lgcpPredictMultitypeSpatialPlusPars

Usage

betavals(lg)
**blockcircbase**

**Arguments**

- `lg` an object produced by a call to `lgcpPredictSpatialPlusPars`, `lgcpPredictAggregateSpatialPlusPars`, `lgcpPredictSpatioTemporalPlusPars` or `lgcpPredictMultitypeSpatialPlusPars`

**Value**

the posterior sampled beta

**See Also**

- `ltar`, `autocorr`, `parautocorr`, `traceplots`, `parsummary`, `textsummary`, `priorpost`, `postcov`, `exceedProbs`, `etavals`

---

**blockcircbase**  
*blockcircbase function*

**Description**

Compute the base matrix of a continuous Gaussian field. Computed as a block circulant matrix on a torus where x and y is the x and y centroids (must be equally spaced)

**Usage**

`blockcircbase(x, y, sigma, phi, model, additionalparameters, inverse = FALSE)`

**Arguments**

- `x` x centroids, an equally spaced vector
- `y` y centroids, an equally spaced vector
- `sigma` spatial variance parameter
- `phi` spatial decay parameter
- `model` covariance model, see ?CovarianceFct
- `additionalparameters` additional parameters for chosen covariance model. See ?CovarianceFct
- `inverse` logical. Whether to return the base matrix of the inverse covariance matrix (ie the base matrix for the precision matrix), default is FALSE

**Value**

the base matrix of a block circulant matrix representing a stationary covariance function on a toral grid.
blockcircbaseFunction  *blockcircbaseFunction function*

**Description**

Compute the base matrix of a continuous Gaussian field. Computed as a block circulant matrix on a torus where x and y is the x and y centroids (must be equally spaced). This is an extension of the function blockcircbase to extend the range of covariance functions that can be fitted to the model.

**Usage**

```
blockcircbaseFunction(x, y, CovFunction, CovParameters, inverse = FALSE)
```

**Arguments**

- **x**  
  x centroids, an equally spaced vector
- **y**  
  y centroids, an equally spaced vector
- **CovFunction**  
  a function of distance, returning the covariance between points that distance apart
- **CovParameters**  
  an object of class CovParameters, see ?CovParameters
- **inverse**  
  logical. Whether to return the base matrix of the inverse covariance matrix (ie the base matrix for the precision matrix), default is FALSE

**Value**

the base matrix of a block circulant matrix representing a stationary covariance function on a toral grid.

**See Also**

chooseCellwidth, getpolyol, guessinterp, getZmat, addTemporalCovariates, lgcpPrior, lgcpInits, lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatioTemporalPlusPars, lgcpPredictMultitypeSpatialPlusPars

---

bt.scalar  *bt.scalar function*

**Description**

bt.scalar function

**Usage**

```
bt.scalar(t, theta)
```
Arguments

\( t \)  
change in time, see Brix and Diggle (2001)

\( \theta \)  
parameter beta in Brix and Diggle

Value

...

checkObsWin function

Description

A function to run on an object generated by the "selectObsWindow" function. Plots the observation window with grid, use as a visual aid to check the choice of cell width is correct.

Usage

checkObsWin(ow)

Arguments

\( ow \)  
an object generated by selectObsWindow, see ?selectObsWindow

Value

a plot of the observation window and grid

See Also

chooseCellwidth

chooseCellwidth function

Description

A function to help choose the cell width (the parameter "cellwidth" in lgcpPredictSpatialPlusPars, for example) prior to setting up the FFT grid, before an MCMC run.

Usage

chooseCellwidth(obj, cwidth)
Arguments

obj an object of class ppp, stppp, SpatialPolygonsDataFrame, or owin
cwinit the cell width

Details

Ideally this function should be used after having made a preliminary guess at the parameters of the latent field. The idea is to run `chooseCellwidth` several times, adjusting the parameter "cwinit" so as to balance available computational resources with output grid size.

Value

produces a plot of the observation window and computational grid.

See Also

generic function for constructing circulant matrices

Usage

circulant(x, ...)

Arguments

x an object
... additional arguments

Value

method circulant
circulant.matrix

circulant.matrix function

Description
If x is a matrix whose columns are the bases of the sub-blocks of a block circulant matrix, then this function returns the block circulant matrix of interest.

Usage
## S3 method for class 'matrix'
circulant(x, ...)

Arguments
x a matrix object
...

Value
If x is a matrix whose columns are the bases of the sub-blocks of a block circulant matrix, then this function returns the block circulant matrix of interest.

circulant.numeric
circulant.numeric function

Description
returns a circulant matrix with base x

Usage
## S3 method for class 'numeric'
circulant(x, ...)

Arguments
x an numeric object
...

Value
a circulant matrix with base x
clearinterp  

**clearinterp function**

**Description**

A function to remove the interpolation methods from a data frame.

**Usage**

```r
clearinterp(df)
```

**Arguments**

- `df`: a data frame

**Value**

removes the interpolation methods

---

computeGradtruncSpatial  

**computeGradtruncSpatial function**

**Description**

**Advanced use only.** A function to compute a gradient truncation parameter for 'spatial only' MALA via simulation. The function requires an FFT 'grid' to be pre-computed, see `fftgrid`.

**Usage**

```r
computeGradtruncSpatial(
  nsims = 100,
  scale = 1,
  nis,
  mu,
  rootQeigs,
  invrootQeigs,
  scaleconst,
  spatial,
  cellarea
)
```
computeGradtruncSpatioTemporal

Arguments

nsims          The number of simulations to use in computation of gradient truncation.
scale          multiplicative scaling constant, returned value is scale (times) max(gradient over
                simulations). Default scale is 1.
nis            cell counts on the extended grid
mu             parameter of latent field, mu
rootQeigs      root of eigenvalues of precision matrix of latent field
invrootQeigs   reciprocal root of eigenvalues of precision matrix of latent field
scaleconst     expected number of cases, or ML estimate of this quantity
spatial        spatial at risk interpolated onto grid of requisite size
cellarea       cell area

Value

gradient truncation parameter

See Also

fftgrid

computeGradtruncSpatioTemporal

computeGradtruncSpatioTemporal function

Description

**Advanced use only.** A function to compute a gradient truncation parameter for 'spatial only'
MALA via simulation. The function requires an FFT 'grid' to be pre-computed, see fftgrid.

Usage

computeGradtruncSpatioTemporal(
    nsims = 100,
    scale = 1,
    nis,
    mu,
    rootQeigs,
    invrootQeigs,
    spatial,
    temporal,
    bt,
    cellarea
)
condProbs

Arguments

- `nsims`: The number of simulations to use in computation of gradient truncation.
- `scale`: multiplicative scaling constant, returned value is scale (times) max(gradient over simulations). Default scale is 1.
- `nis`: cell counts on the extended grid
- `mu`: parameter of latent field, mu
- `rootQeigs`: root of eigenvalues of precision matrix of latent field
- `invrootQeigs`: reciprocal root of eigenvalues of precision matrix of latent field
- `spatial`: spatial at risk interpolated onto grid of requisite size
- `temporal`: fitted temporal values
- `bt`: vectoer of variances b(delta t) in Brix and Diggle 2001
- `cellarea`: cell area

Value

gradient truncation parameter

See Also

fftgrid

condProbs

condProbs function

Description

A function to compute the conditional type-probabilities from a multivariate LGCP. See the vignette “Bayesian_lgcp” for a full explanation of this.

Usage

condProbs(obj)

Arguments

- `obj`: an lgcpPredictMultitypeSpatialPlusParameters object
Details

We suppose there are \( K \) point types of interest. The model for point-type \( k \) is as follows:

\[
X_k(s) \sim \text{Poisson}[R_k(s)]
\]

\[
R_k(s) = C_A \lambda_k(s) \exp[Z_k(s)\beta_k + Y_k(s)]
\]

Here \( X_k(s) \) is the number of events of type \( k \) in the computational grid cell containing the point \( s \), \( R_k(s) \) is the Poisson rate, \( C_A \) is the cell area, \( \lambda_k(s) \) is a known offset, \( Z_k(s) \) is a vector of measured covariates and \( Y_i(s) \) where \( i = 1, \ldots, K+1 \) are latent Gaussian processes on the computational grid. The other parameters in the model are \( \beta_k \), the covariate effects for the \( k \)th type; and \( \eta_i = [\log(\sigma_i), \log(\phi_i)] \), the parameters of the process \( Y_i \) for \( i = 1, \ldots, K+1 \) on an appropriately transformed (again, in this case log) scale.

The term ‘conditional probability of type \( k \)’ means the probability that at a particular location there will be an event of type \( k \), which denoted \( p_k \).

Value

an \text{lgcpgrid} object containing the conditional type-probabilities for each type

See Also

\text{segProbs}, \text{postcov.lgcpPredictSpatialOnlyPlusParameters}, \text{postcov.lgcpPredictAggregateSpatialPlusParameters}, \text{postcov.lgcpPredictSpatioTemporalPlusParameters}, \text{postcov.lgcpPredictMultitypeSpatialPlusParameters}, \text{ltar}, \text{autocorr}, \text{parautocorr}, \text{traceplots}, \text{parsummary}, \text{textsummary}, \text{priorpost}, \text{postcov}, \text{exceedProbs}, \text{betavals}, \text{etavals}

\begin{verbatim}
constanth

constanth function

Description

This function is used to set up a constant acceptance scheme in the argument \text{mcmc.control} of the function \text{lgcpPredict}. The scheme is set via the \text{mcmcpars} function.

Usage

\text{constanth}(h)

Arguments

\( h \) an object

Value

object of class constanth
constantInTime

See Also

mcmcpars, lgcpPredict

Examples

constantInTime(0.01)

constantInTime function

Description

Generic function for creating constant-in-time temporalAtRisk objects, that is for models where mu(t) can be assumed to be constant in time. The assumption being that the global at-risk population does not change in size over time.

Usage

constantInTime(obj, ...)

Arguments

obj an object

... additional arguments

Details

For further details of temporalAtRisk objects, see ?temporalAtRisk>

Value

method constantInTime

See Also

temporalAtRisk, spatialAtRisk, temporalAtRisk.numeric, temporalAtRisk.function, constantInTime.numeric, constantInTime.stppp, print.temporalAtRisk, plot.temporalAtRisk
**constantInTime.numeric function**

**Description**

Create a constant-in-time temporalAtRisk object from a numeric object of length 1. The returned temporalAtRisk object is assumed to have been scaled correctly by the user so that \( \mu(t) = E(\text{number of cases in a unit time interval}) \).

**Usage**

```r
## S3 method for class 'numeric'
constantInTime(obj, tlim, warn = TRUE, ...)
```

**Arguments**

- `obj` numeric constant
- `tlim` vector of length 2 giving time limits
- `warn` Issue a warning if the given temporal intensity treated is treated as 'known'? 
- `...` additional arguments

**Details**

For further details of temporalAtRisk objects, see `?temporalAtRisk`

**Value**

A function \( f(t) \) giving the (constant) temporal intensity at time \( t \) for integer \( t \) in the interval \([tlim[1],tlim[2]]\) of class temporalAtRisk

**See Also**

temporalAtRisk, spatialAtRisk, temporalAtRisk.numeric, temporalAtRisk.function, constantInTime, constantInTime.stppp, print.temporalAtRisk, plot.temporalAtRisk,
constantInTime.stppp  \textit{constantInTime.stppp function}

\textbf{Description}

Create a constant-in-time temporalAtRisk object from an stppp object. The returned temporalAtRisk object is scaled to return \( \mu(t) = E(\text{number of cases in a unit time interval}) \).

\textbf{Usage}

\begin{verbatim}
## S3 method for class 'stppp'
constantInTime(obj, ...)
\end{verbatim}

\textbf{Arguments}

- \texttt{obj}  
  an object of class stppp.

- \texttt{...}  
  additional arguments

\textbf{Details}

For further details of temporalAtRisk objects, see \?temporalAtRisk>

\textbf{Value}

a function \( f(t) \) giving the (constant) temporal intensity at time \( t \) for integer \( t \) in the interval \([t\text{lim}[1], t\text{lim}[2]]\) of class temporalAtRisk

\textbf{See Also}

temporalAtRisk, spatialAtRisk, temporalAtRisk.numeric, temporalAtRisk.function, constantInTime, constantInTime.numeric, print.temporalAtRisk, plot.temporalAtRisk,

\cov.interp.fft  \textit{cov.interp.fft function}

\textbf{Description}

A function to interpolate covariate values onto the fft grid, ready for analysis
**Usage**

cov.interp.fft(
    formula,
    W,
    regionalcovariates = NULL,
    pixelcovariates = NULL,
    mcens,
    ncens,
    cellInside,
    overl = NULL
)

**Arguments**

- **formula**: an object of class formula (or one that can be coerced to that class) starting with `X ~ (eg X-var1+var2 *NOT for example* Y-var1+var2): a symbolic description of the model to be fitted.
- **W**: an owin observation window
- **regionalcovariates**: an optional SpatialPolygonsDataFrame
- **pixelcovariates**: an optional SpatialPixelsDataFrame
- **mcens**: x-coordinates of output grid centroids (not fft grid centroids ie *not* the extended grid)
- **ncens**: y-coordinates of output grid centroids (not fft grid centroids ie *not* the extended grid)
- **cellInside**: a 0-1 matrix indicating which computational cells are inside the observation window
- **overl**: an overlay of the computational grid onto the SpatialPolygonsDataFrame or SpatialPixelsDataFrame.

**Value**

The interpolated design matrix, ready for analysis

---

**CovarianceFct**  
*CovarianceFct function*

**Description**

A function to

**Usage**

CovarianceFct(u, sigma, phi, model, additionalparameters)
Arguments

- **u**: distance
- **sigma**: parameter sigma
- **phi**: parameter phi
- **model**: character string, the model
- **additionalparameters**: additional parameters for the covariance function that will be fixed.

Value

the covariance function evaluated at the specified distances

covEffects
covEffects function

Description

A function used in conjunction with the function "expectation" to compute the main covariate effects,
\( \lambda(s) \exp[Z(s)\beta] \)
in each computational grid cell. Currently only implemented for spatial processes (lgcpPredictSpatialPlusPars and lgcpPredictAggregateSpatialPlusPars).

Usage

covEffects(Y, beta, eta, Z, otherargs)

Arguments

- **Y**: the latent field
- **beta**: the main effects
- **eta**: the parameters of the latent field
- **Z**: the design matrix
- **otherargs**: other arguments to the function (see vignette "Bayesian_lgcp" for an explanation)

Value

the main effects

See Also

expectation, lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars

Examples

```r
## Not run: ex <- expectation(lg,covEffects)[[1]] # lg is output from spatial LGCP MCMC
```
CovFunction

CovFunction function

Description

A Generic method used to specify the choice of covariance function for use in the MCMC algorithm. For further details and examples, see the vignette "Bayesian_lgcp".

Usage

CovFunction(obj, ...)

Arguments

obj an object
...

additional arguments

Value

method CovFunction

See Also

CovFunction.function, exponentialCovFct, RandomFieldsCovFct, SpikedExponentialCovFct

CovFunction.function

CovFunction.function function

Description

A function used to define the covariance function for the latent field prior to running the MCMC algorithm

Usage

## S3 method for class 'function'
CovFunction(obj, ...)

Arguments

obj a function object
...

additional arguments

Value

the covariance function ready to run the MCMC routine.
See Also

exponentialCovFct, RandomFieldsCovFct, SpikedExponentialCovFct, CovarianceFct

Examples

```r
## Not run: cf1 <- CovFunction(exponentialCovFct)
## Not run: cf2 <- CovFunction(RandomFieldsCovFct(model="matern",additionalparameters=1))
```

---

**CovParameters function**

Description

A function to provide a structure for the parameters of the latent field. Not intended for general use.

Usage

```r
CovParameters(list)
```

Arguments

- `list`: a list

Value

an object used in the MCMC routine.

---

**Cvb function**

Description

This function is used in thetaEst to estimate the temporal correlation parameter, theta.

Usage

```r
Cvb(xyt, spatial.intensity, N = 100, spatial.covmodel, covpars)
```

Arguments

- `xyt`: object of class stppp
- `spatial.intensity`: bivariate density estimate of lambda, an object of class im (produced from density.ppp for example)
- `N`: number of integration points
- `spatial.covmodel`: spatial covariance model
- `covpars`: additional covariance parameters
Value

a function, see below. Computes Monte carlo estimate of function $C(v;\beta)$ in Brix and Diggle 2001 pp 829 (... note later corrigendum to paper (2003) corrects the expression given in this paper)

References


See Also

thetaEst

d.func function

d.func(mat1il, mat2jk, i, j, l, k)

Arguments

mat1il matrix 1
mat2jk matrix 2
i index matrix 1 number 1
j index matrix 2 number 1
l index matrix 1 number 2
k index matrix 2 number 2

Value

...
density.stppp

density.stppp function

Description

A wrapper function for density.ppp.

Usage

## S3 method for class 'stppp'

density(x, bandwidth = NULL, ...)

Arguments

x an stppp object
bandwidth 'bandwidth' parameter, equivalent to parameter sigma in ?density.ppp i.e. standard deviation of isotropic Gaussian smoothing kernel.
...
additional arguments to be passed to density.ppp

Value

bivariate density estimate of xyt; not this is a wrapper function for density.ppp

See Also

density.ppp

discreteWindow

discreteWindow function

Description

Generic function for extracting the FFT discrete window.

Usage

discreteWindow(obj, ...)

Arguments

obj an object
...
additional arguments

Value

method discreteWindow
discreteWindow.lgcpPredict

See Also
discreteWindow.lgcpPredict

discreteWindow.lgcpPredict

discreteWindow.lgcpPredict function

Description
A function for extracting the FFT discrete window from an lgcpPredict object.

Usage
## S3 method for class 'lgcpPredict'
discreteWindow(obj, inclusion = "touching", ...)

Arguments
obj
an lgcpPredict object
inclusion
criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.
...
additional arguments

Value
...

dump2dir

dump2dir function

Description
This function, when set by the gridfunction argument of setoutput, in turn called by the argument output.control of lgcpPredict facilitates the dumping of data to disk. Data is dumped to a netCDF file, simout.nc, stored in the directory specified by the user. If the directory does not exist, then it will be created. Since the requested data dumped to disk may be very large in a run of lgcpPredict, by default, the user is prompted as to whether to proceed with prediction, this can be turned off by setting the option forceSave=TRUE detailed here. To save space, or increase the number of simulations that can be stored for a fixed disk space the option to only save the last time point is also available (lastonly=TRUE, which is the default setting).

Usage
dump2dir(dirname, lastonly = TRUE, forceSave = FALSE)
Arguments

dirname        character vector of length 1 containing the name of the directory to create
lastonly       only save output from time T? (see ?lgcpPredict for definition of T)
forceSave      option to override display of menu

Value

object of class dump2dir

See Also

setoutput, \GFinitialise, GFupdate, GFfinalise, GFreturnvalue

---

eigenfrombase  eigenfrombase function

Description

A function to compute the eigenvalues of an SPD block circulant matrix given the base matrix.

Usage

eigenfrombase(x)

Arguments

x            the base matrix

Value

the eigenvalues

---

etavals      etavals function

Description

A function to return the sampled eta from a call to the function lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatioTemporalPlusPars or lgcpPredictMultitypeSpatialPlusPars

Usage

etavals(lg)
EvaluatePrior

Arguments

\texttt{etaParameters} \hspace{1cm} \texttt{betaParameters} \hspace{1cm} \texttt{prior}

Value

the posterior sampled \texttt{eta}

See Also

\texttt{ltar}, \texttt{autocorr}, \texttt{parautocorr}, \texttt{traceplots}, \texttt{parsummary}, \texttt{textsummary}, \texttt{priorpost}, \texttt{postcov}, \texttt{exceedProbs}, \texttt{betavals}

Description

An internal function used in the MCMC routine to evaluate the prior for a given set of parameters

Usage

\texttt{EvaluatePrior(etaParameters, betaParameters, prior)}

Arguments

\texttt{etaParameters} \hspace{1cm} \texttt{the parameter eta}

\texttt{betaParameters} \hspace{1cm} \texttt{the parameter beta}

\texttt{prior} \hspace{1cm} \texttt{the prior}

Value

the prior evaluated at the given values.
exceedProbsAggregated function

Description

This function can be called using MonteCarloAverage (see fun3 the examples in the help file for MonteCarloAverage). It computes exceedance probabilities,

\[ P[\exp(Y_{1:t2}) > k], \]

that is the probability that the relative reisk exceeds threshold \( k \). Note that it is possible to pass vectors of tresholds to the function, and the exceedance probabilities will be computed for each of these.

Usage

\[
\text{exceedProbsAggregated}(\text{threshold, lg = NULL, lastonly = TRUE})
\]

Arguments

threshold vector of threshold levels for the indicator function
direction default 'upper' giving exceedance probabilities, alternative is 'lower', which gives 'subordinate probabilities'

Value

a function of Y that computes the indicator function \( I(\exp(Y) > \text{threshold}) \) evaluated for each cell of a matrix Y If several tresholds are specified an array is returned with the \([i]th\) slice equal to \( I(\exp(Y) > \text{threshold}[i]) \)

See Also

MonteCarloAverage, setoutput
**expectation**

**Arguments**

- **threshold**: vector of threshold levels for the indicator function
- **lg**: an object of class aggregatedPredict
- **lastonly**: logical, whether to only compute the exceedances for the last time point. default is TRUE

**Details**

This function computes regional exceedance probabilities after MCMC has finished, it requires the information to have been dumped to disk, and to have been computed using the function lgcpPredictAggregated

\[ P[\exp(Y_{t_1:t_2}) > k] \],

that is the probability that the relative risk exceeds threshold \( k \). Note that it is possible to pass vectors of thresholds to the function, and the exceedance probabilities will be computed for each of these.

**Value**

A function of \( Y \) that computes the indicator function \( I(\exp(Y) > \text{threshold}) \) evaluated for each cell of a matrix \( Y \), but with values aggregated to regions If several thresholds are specified an array is returned with the \([.,i]th\) slice equal to \( I(\exp(Y) > \text{threshold}[i]) \)

**See Also**

lgcpPredictAggregated

**Description**

Generic function used in the computation of Monte Carlo expectations.

**Usage**

```r
expectation(obj, ...)
```

**Arguments**

- **obj**: an object
- **...**: additional arguments

**Value**

method expectation
expectation.lgcpPredict

expectation.lgcpPredict function

Description

This function requires data to have been dumped to disk: see ?dump2dir and ?setoutput. This function computes the Monte Carlo Average of a function where data from a run of lgcpPredict has been dumped to disk.

Usage

## S3 method for class 'lgcpPredict'
expectation(obj, fun, maxit = NULL, ...)

Arguments

- **obj**: an object of class lgcpPredict
- **fun**: a function accepting a single argument that returns a numeric vector, matrix or array object
- **maxit**: Not used in ordinary circumstances. Defines subset of samples over which to compute expectation. Expectation is computed using information from iterations 1:maxit, where 1 is the first non-burn in iteration dumped to disk.
- **...**: additional arguments

Details

A Monte Carlo Average is computed as:

\[
E_{\pi}(Y_{t_1:t_2}|X_{t_1:t_2})[g(Y_{t_1:t_2})] \approx \frac{1}{n} \sum_{i=1}^{n} g(Y_{t_1:t_2}^{(i)})
\]

where \( g \) is a function of interest, \( Y_{t_1:t_2}^{(i)} \) is the \( i \)th retained sample from the target and \( n \) is the total number of retained iterations. For example, to compute the mean of \( Y_{t_1:t_2} \) set,

\[
g(Y_{t_1:t_2}) = Y_{t_1:t_2},
\]

the output from such a Monte Carlo average would be a set of \( t_2 - t_1 \) grids, each cell of which being equal to the mean over all retained iterations of the algorithm (NOTE: this is just an example computation, in practice, there is no need to compute the mean on line explicitly, as this is already done by default in lgcpPredict).

Value

- the expected value of that function

See Also

lgcpPredict, dump2dir, setoutput
expectation.lgcpPredictSpatialOnlyPlusParameters

**expectation.lgcpPredictSpatialOnlyPlusParameters function**

**Description**

This function requires data to have been dumped to disk: see ?dump2dir and ?setoutput. This function computes the Monte Carlo Average of a function where data from a run of lgcpPredict has been dumped to disk.

**Usage**

```
"expectation(obj, fun, maxit=NULL, ...)
```

**Arguments**

- **obj**: an object of class lgcpPredictSpatialOnlyPlusParameters
- **fun**: a function with arguments ‘Y’, ‘beta’, ‘eta’, ‘Z’ and ‘otherargs’. See vignette("Bayesian_lgcp") for an example
- **maxit**: Not used in ordinary circumstances. Defines subset of samples over which to compute expectation. Expectation is computed using information from iterations 1:maxit, where 1 is the first non-burn in iteration dumped to disk.
- **...**: additional arguments

**Value**

the expected value of that function

---

**exponentialCovFct**

**exponentialCovFct function**

**Description**

A function to declare and also evaluate an exponential covariance function.

**Usage**

```
exponentialCovFct(d, CovParameters)
```

**Arguments**

- **d**: total distance
- **CovParameters**: parameters of the latent field, an object of class "CovParameters".
Value

the exponential covariance function

See Also

CovFunction.function, RandomFieldsCovFct, SpikedExponentialCovFct

extendspatialAtRisk  extendspatialAtRisk function

Description

A function to extend a spatialAtRisk object, used in interpolating the fft grid NOTE THIS DOES NOT RETURN A PROPER spatialAtRisk OBJECT SINCE THE NORMALISING CONSTANT IS PUT BACK IN.

Usage

extendspatialAtRisk(spatial)

Arguments

spatial  a spatialAtRisk object inheriting class ’fromXYZ’

Value

the spatialAtRisk object on a slightly larger grid, with zeros appearing outside the original extent.

extract  extract function

Description

Generic function for extracting information dumped to disk. See extract.lgcpPredict for further information.

Usage

extract(obj, ...)

Arguments

obj  an object

...  additional arguments
Description

This function requires data to have been dumped to disk: see ?dump2dir and ?setoutput. extract.lgcpPredict extracts chunks of data that have been dumped to disk. The subset of data can either be specified using an (x,y,t,s) box or (window,t,s) region where window is a polygonal subregion of interest.

Usage

## S3 method for class 'lgcpPredict'
extract(
  obj,
  x = NULL,
  y = NULL,
  t,
  s = -1,
  inWindow = NULL,
  crop2parentwindow = TRUE,
  ...
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>an object of class lgcpPredict</td>
</tr>
<tr>
<td>x</td>
<td>range of x-indices: vector (eg c(2,4)) corresponding to desired subset of x coordinates. If equal to -1, then all cells in this dimension are extracted</td>
</tr>
<tr>
<td>y</td>
<td>range of y-indices as above</td>
</tr>
<tr>
<td>t</td>
<td>range of t-indices: time indices of interest</td>
</tr>
<tr>
<td>s</td>
<td>range of s-indices ie the simulation indices of interest</td>
</tr>
<tr>
<td>inWindow</td>
<td>an observation owin window over which to extract the data (alternative to specifying x and y).</td>
</tr>
<tr>
<td>crop2parentwindow</td>
<td>logical: whether to only extract cells inside obj$xyt$window (the 'parent window')</td>
</tr>
<tr>
<td>...</td>
<td>additional arguments</td>
</tr>
</tbody>
</table>
Extract.stppp

**Description**

Extracting subsets of an stppp object.

**Usage**

```
x[subset]
```

**Arguments**

- `x`: an object of class stppp
- `subset`: the subset to extract

**Value**

extracts subset of an stppp object

---

Extract.mstppp

**Description**

Extracting subsets of an mstppp object.

**Usage**

```
x[subset]
```

**Arguments**

- `x`: an object of class mstppp
- `subset`: subset to extract

**Value**

extracts subset of an mstppp object
Examples

```r
## Not run: xyt <- lgcpSim()
## Not run: xyt
## Not run: xyt[xyt$t>0.5]
```

fftgrid

fftgrid function

Description

! As of lgcp version 0.9-5, this function is no longer used!

Usage

`fftgrid(xyt, M, N, spatial, sigma, phi, model, covpars, inclusion = "touching")`

Arguments

- `xyt`: object of class stppp
- `M`: number of centroids in x-direction
- `N`: number of centroids in y-direction
- `spatial`: an object of class spatialAtRisk
- `sigma`: scaling parameter for spatial covariance function, see Brix and Diggle (2001)
- `phi`: scaling parameter for spatial covariance function, see Brix and Diggle (2001)
- `model`: correlation type see ?CovarianceFct
- `covpars`: vector of additional parameters for certain classes of covariance function (eg Matern), these must be supplied in the order given in ?CovarianceFct
- `inclusion`: criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.

Details

Advanced use only. Computes various quantities for use in lgcpPredict, lgcpSim.

Value

fft objects for use in MALA
fftinterpolate function

Description

Generic function used for computing interpolations used in the function fftgrid.

Usage

fftinterpolate(spatial, ...)

Arguments

spatial an object
...
additional arguments

Value

method fftinterpolate

See Also

fftgrid

fftinterpolate.fromFunction function

Description

This method performs interpolation within the function fftgrid for fromFunction objects.

Usage

## S3 method for class 'fromFunction'
fftinterpolate(spatial, mcens, ncens, ext, ...)

Arguments

spatial objects of class spatialAtRisk
mcens x-coordinates of interpolation grid in extended space
ncens y-coordinates of interpolation grid in extended space
ext integer multiple by which grid should be extended, default is 2. Generally this will not need to be altered, but if the spatial correlation decays slowly, increasing 'ext' may be necessary.
...
additional arguments
Description

This method performs interpolation within the function fftgrid for fromSPDF objects.

Usage

```r
## S3 method for class 'fromSPDF'
fftinterpolate(spatial, mcens, ncens, ext, ...)
```

Arguments

- `spatial`: objects of class spatialAtRisk
- `mcens`: x-coordinates of interpolation grid in extended space
- `ncens`: y-coordinates of interpolation grid in extended space
- `ext`: integer multiple by which grid should be extended, default is 2. Generally this will not need to be altered, but if the spatial correlation decays slowly, increasing 'ext' may be necessary.
- `...`: additional arguments

Value

matrix of interpolated values

See Also

fftgrid, spatialAtRisk.SpatialPolygonsDataFrame
Description

This method performs interpolation within the function fftgrid for fromXYZ objects.

Usage

## S3 method for class 'fromXYZ'
fftinterpolate(spatial, mcens, ncens, ext, ...)

Arguments

- `spatial`: objects of class spatialAtRisk
- `mcens`: x-coordinates of interpolation grid in extended space
- `ncens`: y-coordinates of interpolation grid in extended space
- `ext`: integer multiple by which grid should be extended, default is 2. Generally this will not need to be altered, but if the spatial correlation decays slowly, increasing 'ext' may be necessary.
- `...`: additional arguments

Value

matrix of interpolated values

See Also

fftgrid, spatialAtRisk.fromXYZ

Description

A function to pre-multiply a vector by a block cirulant matrix

Usage

fftmultiply(efb, vector)
**Arguments**

- **efb**
  - eigenvalues of the matrix

- **vector**
  - the vector

**Value**

- a vector: the product of the matrix and the vector.

---

**Description**

A function to create an object of class "formulaList" from a list of "formula" objects; use to define the model for the main effects prior to running the multivariate MCMC algorithm.

**Usage**

```r
formulaList(X)
```

**Arguments**

- **X**
  - a list object, each element of which is a formula

**Value**

- an object of class "formulaList"

---

**Description**

Generic function defining the finalisation step for the `gridAverage` class of functions. The function is called invisibly within `MALAlgcp` and facilitates the computation of Monte Carlo Averages online.

**Usage**

```r
GAfinalise(F, ...)
```

**Arguments**

- **F**
  - an object

- **...**
  - additional arguments
GAfinalise.nullAverage

Value
method GAfinalise

See Also
setoutput, GAinitialise, GAupdate, GAreturnvalue

GAfinalise.MonteCarloAverage

Description
Finalise a Monte Carlo averaging scheme. Divide the sum by the number of iterations.

Usage
## S3 method for class 'MonteCarloAverage'
GAfinalise(F, ...)

Arguments
F an object of class MonteCarloAverage
... additional arguments

Value
computes Monte Carlo averages

See Also
MonteCarloAverage, setoutput, GAinitialise, GAupdate, GAfinalise, GAreturnvalue

GAfinalise.nullAverage

Description
This is a null function and performs no action.

Usage
## S3 method for class 'nullAverage'
GAfinalise(F, ...)
GAinitialise

Arguments

F an object of class nullAverage

... additional arguments

Value

nothing

See Also

nullAverage, setoutput, GAinitialise, GAupdate, GAfinalise, GAreturnvalue

GAinitialise

Description

Generic function defining the the initialisation step for the gridAverage class of functions. The function is called invisibly within MALAlgcp and facilitates the computation of Monte Carlo Averages online.

Usage

GAinitialise(F, ...)

Arguments

F an object

... additional arguments

Value

method GAinitialise

See Also

setoutput, GAupdate, GAfinalise, GAreturnvalue
GAinitialise.MonteCarloAverage

GAinitialise.MonteCarloAverage function

Description
Initialise a Monte Carlo averaging scheme.

Usage

```r
## S3 method for class 'MonteCarloAverage'
GAinitialise(F, ...)  
```

Arguments

- `F`: an object of class MonteCarloAverage
- `...`: additional arguments

Value

nothing

See Also

MonteCarloAverage, setoutput, GAinitialise, GAnupdate, GAfinalise, GAreturnvalue

GAinitialise.nullAverage

GAinitialise.nullAverage function

Description
This is a null function and performs no action.

Usage

```r
## S3 method for class 'nullAverage'
GAinitialise(F, ...)  
```

Arguments

- `F`: an object of class nullAverage
- `...`: additional arguments
GammafromY

Value
nothing

See Also
nullAverage, setoutput, GAinitialise, GAupdate, GAfinalise, GAreturnvalue

GammafromY function

Description
A function to change Ys (spatially correlated noise) into Gammas (white noise). Used in the MALA algorithm.

Usage
GammafromY(Y, rootQeigs, mu)

Arguments
Y     Y matrix
rootQeigs  square root of the eigenvectors of the precision matrix
mu    parameter of the latent Gaussian field

Value
Gamma

GAreturnvalue function

Description
Generic function defining the returned value for the gridAverage class of functions. The function is called invisibly within MALAlgcp and facilitates the computation of Monte Carlo Averages online.

Usage
GAreturnvalue(F, ...)

Arguments
F    an object
...  additional arguments
GAreturnvalue.nullAverage

Value

method GAreturnvalue

See Also

setoutput, GA initialise, GA update, GA finalise

GAreturnvalue.MonteCarloAverage

Description

Returns the required Monte Carlo average.

Usage

## S3 method for class 'MonteCarloAverage'
GAreturnvalue(F, ...)

Arguments

F an object of class MonteCarloAverage

... additional arguments

Value

results from MonteCarloAverage

See Also

MonteCarloAverage, setoutput, GA initialise, GA update, GA finalise, GAreturnvalue

GAreturnvalue.nullAverage

Description

This is a null function and performs no action.

Usage

## S3 method for class 'nullAverage'
GAreturnvalue(F, ...)

**GAupdate**

**Arguments**

- **F** an object of class `nullAverage`
- ... additional arguments

**Value**

nothing

**See Also**

`nullAverage, setoutput, GA initialise, GAupdate, GAFinalise, GArturnvalue`

---

**GAupdate function**

**Description**

Generic function defining the update step for the `gridAverage` class of functions. The function is called invisibly within `MALAlgcp` and facilitates the computation of Monte Carlo Averages online.

**Usage**

`GAupdate(F, ...)`

**Arguments**

- **F** an object
- ... additional arguments

**Value**

method `GAupdate`

**See Also**

`setoutput, GA initialise, GAFinalise, GArturnvalue`
GAupdate.MonteCarloAverage

GAupdate.MonteCarloAverage function

Description
Update a Monte Carlo averaging scheme. This function performs the Monte Carlo sum online.

Usage
## S3 method for class 'MonteCarloAverage'
GAupdate(F, ...)

Arguments
F an object of class MonteCarloAverage
... additional arguments

Value
updates Monte Carlo sums

See Also
MonteCarloAverage, setoutput, GAinitialise, GAupdate, GAfinalise, GAreturnvalue

GAupdate.nullAverage

GAupdate.nullAverage function

Description
This is a null function and performs no action.

Usage
## S3 method for class 'nullAverage'
GAupdate(F, ...)

Arguments
F an object of class nullAverage
... additional arguments

Value
nothing
GaussianPrior

Description
A function to create a Gaussian prior.

Usage
GaussianPrior(mean, variance)

Arguments
- mean: a vector of length 2 representing the mean.
- variance: a 2x2 matrix representing the variance.

Value
an object of class LogGaussianPrior that can be passed to the function PriorSpec.

See Also
LogGaussianPrior, linkPriorSpec.list

Examples
## Not run: GaussianPrior(mean=rep(0,9),variance=diag(10^6,9))

genFFTgrid

description
A function to generate an FFT grid and associated quantities including cell dimensions, size of extended grid, centroids, cell area, cellInside matrix (a 0/1 matrix: is the centroid of the cell inside the observation window?)

Usage
genFFTgrid(study.region, M, N, ext, inclusion = "touching")
Arguments

study.region  an owin object
M  number of cells in x direction
N  number of cells in y direction
ext  multiplying constant: the size of the extended grid: ext*M by ext*N
inclusion  criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.

Value

a list

table

getCellCounts  getCellCounts function

Description

This function is used to count the number of observations falling inside grid cells.

Usage

getCellCounts(x, y, xgrid, ygrid)

Arguments

x  x-coordinates of events
y  y-coordinates of events
xgrid  x-coordinates of grid centroids
ygrid  y-coordinates of grid centroids

Value

The number of observations in each grid cell.
getCounts

**getCounts function**

**Description**

This function is used to count the number of observations falling inside grid cells, the output is used in the function `lgcpPredict`.

**Usage**

```r
getCounts(xyt, subset = rep(TRUE, xyt$n), M, N, ext)
```

**Arguments**

- **xyt**: stppp or ppp data object
- **subset**: Logical vector. Subset of data of interest, by default this is all data.
- **M**: number of centroids in x-direction
- **N**: number of centroids in y-direction
- **ext**: how far to extend the grid eg (M,N) to (ext*M,ext*N)

**Value**

The number of observations in each grid cell returned on a grid suitable for use in the extended FFT space.

**See Also**

- `lgcpPredict`

**Examples**

```r
require(spatstat.core)
xyt <- stppp(ppp(runif(100),runif(100)),t=1:100,tlim=c(1,100))
ccts <- getCounts(xyt,M=64,N=64,ext=2) # gives an output grid of size 128 by 128
cctssub <- ccts[1:64,1:64] # returns the cell counts in the observation
# window of interest
```
getCovParameters  
**getCovParameters function**

Description

Internal function for retrieving covariance parameters. Not intended for general use.

Usage

```r
getcovparameters(obj, ...)
```

Arguments

- `obj` an object
- `...` additional arguments

Value

Method `getCovParameters`

---

getcovparameters.GPrealisation  
**getCovParameters.GPrealisation function**

Description

Internal function for retrieving covariance parameters. Not intended for general use.

Usage

```r
## S3 method for class 'GPrealisation'
getcovparameters(obj, ...)
```

Arguments

- `obj` an GPrealisation object
- `...` additional arguments

Value

...
**getCovParameters.list function**

**Description**
Internal function for retrieving covariance parameters. Not intended for general use.

**Usage**
```r
## S3 method for class 'list'
getCovParameters(obj, ...)
```

**Arguments**
- `obj`: an list object
- `...`: additional arguments

**Value**
```
```

---

**getinterp function**

**Description**
A function to get the interpolation methods from a data frame.

**Usage**
```r
getinterp(df)
```

**Arguments**
- `df`: a data frame

**Details**
The three types of interpolation method employed in the package lgcp are:

1. 'Majority' The interpolated value corresponds to the value of the covariate occupying the largest area of the computational cell.
2. 'ArealWeightedMean' The interpolated value corresponds to the mean of all covariate values contributing to the computational cell weighted by their respective areas.
3. 'ArealWeightedSum' The interpolated value is the sum of all contributing covariates weighed by the proportion of area with respect to the covariate polygons. For example, suppose region A has the same area as a computational grid cell and has 500 inhabitants. If that region occupies half of a computational grid cell, then this interpolation type assigns 250 inhabitants from A to the computational grid cell.

**Value**

the interpolation methods

---

**getlgcpPredictSpatialINLA**

*getlgcpPredictSpatialINLA function*

**Description**

A function to download and 'install' lgcpPredictSpatialINLA into the lgcp namespace.

**Usage**

getlgcpPredictSpatialINLA()

**Value**

Does not return anything

---

**getLHSformulaList**

*getLHSformulaList function*

**Description**

A function to retrieve the dependent variables from a formulaList object. Not intended for general use.

**Usage**

getLHSformulaList(f1)

**Arguments**

* f1 an object of class "formulaList"

**Value**

the independent variables
getpolyol

Description

A function to perform polygon/polygon overlay operations and form the computational grid, on which inference will eventually take place. For details and examples of using this function, please see the package vignette "Bayesian_lgcp".

Usage

getpolyol(
  data,
  regionalcovariates = NULL,
  pixelcovariates = NULL,
  cellwidth,
  ext = 2,
  inclusion = "touching"
)

Arguments

data an object of class ppp or SpatialPolygonsDataFrame, containing the event counts, i.e. the dataset that will eventually be analysed

regionalcovariates an object of class SpatialPolygonsDataFrame containing regionally measured covariate information

pixelcovariates X an object of class SpatialPixelsDataFrame containing regionally measured covariate information

cellwidth the chosen cell width

ext the amount by which to extend the observation window in forming the FFT grid, default is 2. In the case that the point pattern has long range spatial correlation, this may need to be increased.

inclusion criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former, the default, includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.

Value

an object of class lgcppolyol, which can then be fed into the function getZmat.
getRotation.default

See Also

chooseCellwidth, guessinterp, getZmat, addTemporalCovariates, lgcpPrior, lgcpInits, CovFunction
lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatioTemporalPlusPars, lgcpPredictMultitypeSpatialPlusPars

getRotation
getRotation function

Description

Generic function for the computation of rotation matrices.

Usage

getRotation(xyt, ...)

Arguments

xyt an object
...
additional arguments

Value

method getRotation

See Also

getAddress.stppp

getAddress.default getRotation.default function

Description

Presently there is no default method, see ?getRotation.stppp

Usage

## Default S3 method:
getRotation(xyt, ...)

Arguments

xyt an object
...
additional arguments
### getRotation.stppp

**Value**

Currently no default implementation

**See Also**

getRotation.stppp

---

### getRotation.stppp function

**Description**

Compute rotation matrix if observation window is a polygonal boundary

**Usage**

```r
## S3 method for class 'stppp'
getRotation(xyt, ...)
```

**Arguments**

- `xyt` an object of class stppp
- `...` additional arguments

**Value**

The optimal rotation matrix and rotated data and observation window. Note it may or may not be advantageous to rotate the window, this information is displayed prior to the MALA routine when using lgcpPredict

---

### getup function

**Description**

A function to get an object from a parent frame.

**Usage**

```r
getup(n, lev = 1)
```

**Arguments**

- `n` a character string, the name of the object
- `lev` how many levels up the hierarchy to go (see the argument "envir" from the function "get"), default is 1.
getZmat

getZmat function

Description

A function to construct a design matrix for use with the Bayesian MCMC routines in lgcp. See the vignette "Bayesian_lgcp" for further details on how to use this function.

Usage

getzmat(
  formula,
  data,
  regionalcovariates = NULL,
  pixelcovariates = NULL,
  cellwidth,
  ext = 2,
  inclusion = "touching",
  overl = NULL
)

Arguments

formula a formula object of the form X ~ var1 + var2 etc. The name of the dependent variable must be "X". Only accepts 'simple' formulae, such as the example given.
data the data to be analysed (using, for example lgcpPredictSpatialPlusPars). Either an object of class ppp, or an object of class SpatialPolygonsDataFrame
regionalcovariates an optional SpatialPolygonsDataFrame object containing covariate information, if applicable
pixelcovariates an optional SpatialPixelsDataFrame object containing covariate information, if applicable
cellwidth the width of computational cells
ext integer multiple by which grid should be extended, default is 2. Generally this will not need to be altered, but if the spatial correlation decays slowly, increasing 'ext' may be necessary.
inclusion criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former, the default, includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.
getZmats

over1  an object of class "lgcppolyol", created by the function getpolyol. Such an object contains the FFT grid and a polygon/polygon overlay and speeds up computation massively.

Details

For example, a spatial LGCP model for the would have the form:

\[
X(s) \sim \text{Poisson}[R(s)]
\]

\[
R(s) = C_A \lambda(s) \exp[Z(s)\beta+Y(s)]
\]

The function getZmat helps create the matrix Z. The returned object is passed onto an MCMC function, for example lgcpPredictSpatialPlusPars or lgcpPredictAggregateSpatialPlusPars. This function can also be used to help construct Z for use with lgcpPredictSpatioTemporalPlusPars and lgcpPredictMultitypeSpatialPlusPars, but these functions require a list of such objects: see the vignette "Bayesian_lgcp" for examples.

Value

a design matrix for passing on to the Bayesian MCMC functions

See Also

chooseCellwidth, getpolyol, guessinterp, addTemporalCovariates, lgcpPrior, lgcpInits, CovFunction lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatioTemporalPlusPars, lgcpPredictMultitypeSpatialPlusPars

getzmats

getzmats function

Description

An internal function to create Z_k from an lgcpZmat object, for use in the multivariate MCMC algorithm. Not intended for general use.

Usage

getzmats(Zmat, formulaList)

Arguments

Zmat  an object of class "lgcpZmat"

formulaList  an object of class "formulaList"

Value

design matrices for each of the point types
**GFfinalise**  
**GFfinalise function**

**Description**
Generic function defining the the finalisation step for the gridFunction class of objects. The function is called invisibly within MALAlgcp and facilitates the dumping of data to disk.

**Usage**

GFfinalise(F, ...)

**Arguments**

F an object
...
additional arguments

**Value**

method GFfinalise

**See Also**

setoutput, GFinitialise, GFupdate, GFreturnvalue

---

**GFfinalise.dump2dir**  
**GFfinalise.dump2dir function**

**Description**
This function finalises the dumping of data to a netCDF file.

**Usage**

## S3 method for class 'dump2dir'
GFfinalise(F, ...)

**Arguments**

F an object
...
additional arguments

**Value**

nothing
See Also
dump2dir, setoutput, GFinitialise, GFupdate, GFfinalise, GFreturnvalue

GFfinalise.nullFunction

GFfinalise.nullFunction function

Description
This is a null function and performs no action.

Usage
## S3 method for class 'nullFunction'
GFfinalise(F, ...)

Arguments
F an object of class dump2dir
... additional arguments

Value
nothing

See Also
nullFunction, setoutput, GFinitialise, GFupdate, GFfinalise, GFreturnvalue

GFinitialise

GFinitialise function

Description
Generic function defining the the initialisation step for the gridFunction class of objects. The function is called invisibly within MALAlgcp and facilitates the dumping of data to disk.

Usage
GFinitialise(F, ...)

Arguments
F an object
... additional arguments
Value
method GFinitialise

See Also
setoutput, GFupdate, GFfinalise, GFreturnvalue

GFinitialise.dump2dir GFinitialise.dump2dir function

Description
Creates a directory (if necessary) and allocates space for a netCDF dump.

Usage

## S3 method for class 'dump2dir'
GFinitialise(F, ...)

Arguments

F an object of class dump2dir

... additional arguments

Value
creates initialisation file and folder

See Also
dump2dir, setoutput, GFinitialise, GFupdate, GFfinalise, GFreturnvalue

GFinitialise.nullFunction GFinitialise.nullFunction function

Description
This is a null function and performs no action.

Usage

## S3 method for class 'nullFunction'
GFinitialise(F, ...)

GFreturnvalue

Arguments

F an object of class dump2dir

... additional arguments

Value

nothing

See Also

nullFunction, setoutput, GFinitialise, GFupdate, GFfinalise, GFreturnvalue

GFreturnvalue

GFreturnvalue function

Description

Generic function defining the the returned value for the gridFunction class of objects. The function is called invisibly within MALAlgcp and facilitates the dumping of data to disk

Usage

GFreturnvalue(F, ...)

Arguments

F an object

... additional arguments

Value

method GFreturnvalue

See Also

setoutput, GFinitialise, GFupdate, GFfinalise
**GFreturnvalue.dump2dir**

**GFreturnvalue.dump2dir function**

**Description**

This function returns the name of the directory the netCDF file was written to.

**Usage**

```r
## S3 method for class 'dump2dir'
GFreturnvalue(F, ...)
```

**Arguments**

- `F` an object
- `...` additional arguments

**Value**

display where files have been written to

**See Also**

dump2dir, setoutput, GFinitialise, GFupdate, GFfinalise, GFreturnvalue

---

**GFreturnvalue.nullFunction**

**GFreturnvalue.nullFunction function**

**Description**

This is a null function and performs no action.

**Usage**

```r
## S3 method for class 'nullFunction'
GFreturnvalue(F, ...)
```

**Arguments**

- `F` an object of class dump2dir
- `...` additional arguments
GFupdate

Value
nothing

See Also
nullFunction, setoutput, GFinitialise, GFupdate, GFfinalise, GFreturnvalue

GFupdate

GFupdate function

Description
Generic function defining the update step for the gridFunction class of objects. The function is called invisibly within MALAlgcp and facilitates the dumping of data to disk.

Usage
GFupdate(F, ...)

Arguments
F
an object
...
additional arguments

Value
method GFupdate

See Also
setoutput, GFinitialise, GFfinalise, GFreturnvalue

GFupdate.dump2dir

GFupdate.dump2dir function

Description
This function gets the required information from MALAlgcp and writes the data to the netCDF file.

Usage
## S3 method for class 'dump2dir'
GFupdate(F, ...)

# S3 method for class 'dump2dir'
GFupdate(F, ...)
GFupdate.nullFunction

Arguments

F an object
...
additional arguments

Value

saves latent field

See Also

dump2dir, setoutput, GFinicialise, GFupdate, GFfinalise, GFreturnvalue

GFupdate.nullFunction function

Description

This is a null function and performs no action.

Usage

## S3 method for class 'nullFunction'
GFupdate(F, ...)

Arguments

F an object of class dump2dir
...
additional arguments

Value

nothing

See Also

nullFunction, setoutput, GFinicialise, GFupdate, GFfinalise, GFreturnvalue
ginhomAverage

Description

A function to estimate the inhomogeneous pair correlation function for a spatiotemporal point process. See equation (8) of Diggle P, Rowlingson B, Su T (2005).

Usage

ginhomAverage(
  xyt,
  spatial.intensity,
  temporal.intensity,
  time.window = xyt$tlim,
  rvals = NULL,
  correction = "iso",
  suppresswarnings = FALSE,
  ...
)

Arguments

xyt an object of class stpp
spatial.intensity A spatialAtRisk object
temporal.intensity A temporalAtRisk object
time.window time interval contained in the interval xyt$tlim over which to compute average. Useful if there is a lot of data over a lot of time points.
rvals Vector of values for the argument r at which g(r) should be evaluated (see ?pcfinhom). There is a sensible default.
correction choice of edge correction to use, see ?pcfinhom, default is Ripley isotropic correction
suppresswarnings Whether or not to suppress warnings generated by pcfinhom
...
other parameters to be passed to pcfinhom, see ?pcfinhom

Value

time average of inhomogenous pcf, equation (13) of Brix and Diggle 2001.
References


See Also

  KinhomAverage, spatialparsEst, thetaEst, lambdaEst, muEst

Description

A function to overlay the FFT grid, a SpatialPolygons object, onto a SpatialPolygonsDataFrame object.

Usage

gOverlay(grid, spdf)

Arguments

grid the FFT grid, a SpatialPolygons object
spdf a SpatialPolygonsDataFrame object

Details

this code was adapted from Roger Bivand:

Value

a matrix describing the features of the overlay: the originating indices of grid and spdf (all non-trivial intersections) and the area of each intersection.
Description

A function to compute the first derivatives of the log target with respect to the parameters of the latent field. Not intended for general purpose use.

Usage

GPdrv(
  GP,
  prior,
  Z,
  Zt,
  eta,
  beta,
  nis,
  cellarea,
  spatial,
  gradtrunc,
  fftgrid,
  covfunction,
  d,
  eps = 1e-06
)

Arguments

  GP                  an object of class GPrealisation
  prior               priors for the model
  Z                   design matrix on the FFT grid
  Zt                  transpose of the design matrix
  eta                 vector of parameters, eta
  beta                vector of parameters, beta
  nis                 cell counts on the extended grid
  cellarea            the cell area
  spatial             the poisson offset
  gradtrunc           gradient truncation parameter
  fftgrid             an object of class FFTgrid
  covfunction         the choice of covariance function, see ?CovFunction
  d                   matrix of toral distances
  eps                 the finite difference step size
Value

first derivatives of the log target at the specified parameters Y, eta and beta

GPdrv2  

GPdrv2 function

Description

A function to compute the second derivative of the log target with respect to the parameters of the latent field. Not intended for general purpose use.

Usage

GPdrv2(
  GP,
  prior,
  Z,
  Zt,
  eta,
  beta,
  nis,
  cellarea,
  spatial,
  gradtrunc,
  fftgrid,
  covfunction,
  d,
  eps = 1e-06
)

Arguments

GP  an object of class GPrealisation
prior  priors for the model
Z  design matrix on the FFT grid
Zt  transpose of the design matrix
eta  vector of parameters, eta
beta  vector of parameters, beta
nis  cell counts on the extended grid
cellarea  the cell area
spatial  the poisson offset
gradtrunc  gradient truncation parameter
fftgrid  an object of class FFTgrid
covfunction  the choice of covariance function, see ?CovFunction
d  matrix of toral distances
eps  the finite difference step size
GPdrv2_Multitype

Value

first and second derivatives of the log target at the specified parameters Y, eta and beta

GPdrv2_Multitype

GPdrv2_Multitype function

Description

A function to compute the second derivatives of the log target for the multivariate model with respect to the parameters of the latent field. Not intended for general use.

Usage

GPdrv2_Multitype(
  GPlist,
  priorlist,
  Zlist,
  Ztlist,
  etalist,
  betalist,
  nis,
  cellarea,
  spatial,
  gradtrunc,
  fftgrid,
  covfunction,
  d,
  eps = 1e-06,
  k
)

Arguments

GPlist a list of objects of class GPrealisation
priorlist list of priors for the model
Zlist list of design matrices on the FFT grid
Ztlist list of transpose design matrices
etalist list of parameters, eta, for each realisation
betalist clist of parameters, beta, for each realisation
nis cell counts of each type the extended grid
cellarea the cell area
spatial list of poisson offsets for each type
gradtrunc gradient truncation parameter
**GPrealisation**

 fftgrid      an object of class FFTgrid
 covfunction  list giving the choice of covariance function for each type, see ?CovFunction
d           matrix of toral distances
eps         the finite difference step size
k           index of type for which to compute the gradient and hessian

**Value**

first and second derivatives of the log target for type k at the specified parameters Y, eta and beta

---

**GPlist2array**

*GPlist2array function*

**Description**

An internal function for turning a list of GPrealisation objects into an array by a particular common element of the GPrealisation object

**Usage**

GPlist2array(GPlist, element)

**Arguments**

GPlist an object of class GPrealisation
element the name of the element of GPlist[[1]] (for example) to extract, e.g. "Y"

**Value**

an array

---

**GPrealisation**

*GPrealisation function*

**Description**

A function to store a realisation of a spatial gaussian process for use in MCMC algorithms that include Bayesian parameter estimation. Stores not only the realisation, but also computational quantities.

**Usage**

GPrealisation(gamma, fftgrid, covFunction, covParameters, d)
Arguments

- **gamma**: the transformed (white noise) realisation of the process
- **fftgrid**: an object of class FFTgrid, see ?genFFTgrid
- **covFunction**: an object of class function returning the spatial covariance
- **covParameters**: an object of class CovParameters, see ?CovParameters
- **d**: matrix of grid distances

Value

- a realisation of a spatial Gaussian process on a regular grid

Description

A function to convert a regular (x,y) grid of centroids into a SpatialPoints object

Usage

```
grid2spdf(xgrid, ygrid, proj4string = CRS(as.character(NA)))
```

Arguments

- **xgrid**: vector of x centroids (equally spaced)
- **ygrid**: vector of x centroids (equally spaced)
- **proj4string**: an optional proj4string, projection string for the grid, set using the function CRS

Value

- a SpatialPolygonsDataFrame
grid2spix  grid2spix function

Description
A function to convert a regular (x,y) grid of centroids into a SpatialPixels object

Usage
grid2spix(xgrid, ygrid, proj4string = CRS(as.character(NA)))

Arguments
xgrid  vector of x centroids (equally spaced)
ygrid  vector of x centroids (equally spaced)
proj4string an optional proj4string, projection string for the grid, set using the function CRS

Value
a SpatialPixels object

grid2spoly  grid2spoly function

Description
A function to convert a regular (x,y) grid of centroids into a SpatialPolygons object

Usage
grid2spoly(xgrid, ygrid, proj4string = CRS(as.character(NA)))

Arguments
xgrid  vector of x centroids (equally spaced)
ygrid  vector of x centroids (equally spaced)
proj4string  proj 4 string: specify in the usual way

Value
a SpatialPolygons object
grid2spts function

Description
A function to convert a regular (x,y) grid of centroids into a SpatialPoints object

Usage
grid2spts(xgrid, ygrid, proj4string = CRS(as.character(NA)))

Arguments
- xgrid: vector of x centroids (equally spaced)
- ygrid: vector of y centroids (equally spaced)
- proj4string: an optional proj4string, projection string for the grid, set using the function CRS

Value
a SpatialPoints object

gridav function

Description
A generic function for returning gridmeas objects.

Usage
gridav(obj, ...)

Arguments
- obj: an object
- ...: additional arguments

Value
method gridav

See Also
setoutput, lgcpgrid
gridav.lgcpPredict function

Description

Accessor function for lgcpPredict objects: returns the gridmeans argument set in the output.control argument of the function lgcpPredict.

Usage

## S3 method for class 'lgcpPredict'
gridav(obj, fun = NULL, ...)

Arguments

obj an object of class lgcpPredict
fun an optional character vector of length 1 giving the name of a function to return Monte Carlo average of
... additional arguments

Value

returns the output from the gridmeans option of the setoutput argument of lgcpPredict

See Also

setoutput, lgcpgrid

ggridfun function

Description

A generic function for returning gridfunction objects.

Usage

ggridfun(obj, ...)

Arguments

obj an object
... additional arguments

gridfun
Description

Accessor function for lgcpPredict objects: returns the gridfunction argument set in the output.control argument of the function lgcpPredict.

Usage

```r
## S3 method for class 'lgcpPredict'
gridfun(obj, ...)
```

Arguments

- `obj` an object of class lgcpPredict
- `...` additional arguments

Value

returns the output from the gridfunction option of the setoutput argument of lgcpPredict

See Also

setoutput, lgcpgrid

Description

For the grid defined by x-coordinates, xvals, and y-coordinates, yvals, and an owin object W, this function just returns a logical matrix M, whose [i,j] entry is TRUE if the point(xvals[i], yvals[j]) is inside the observation window.

Usage

```r
gridInWindow(xvals, yvals, win, inclusion = "touching")
```
Arguments

xvals  x coordinates
yvals  y coordinates
win    owin object
inclusion  criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former includes all cells that touch the observation window; the latter includes all cells whose centroids are inside the observation window.

Value

matrix of TRUE/FALSE, which elements of the grid are inside the observation window win

gu  
gu function

Description

gu function

Usage

gu(u, sigma, phi, model, additionalparameters)

Arguments

u  distance
sigma  variance parameter, see Brix and Diggle (2001)
phi  scale parameter, see Brix and Diggle (2001)
model  correlation type, see ?CovarianceFct
additionalparameters  vector of additional parameters for certain classes of covariance function (eg Matern), these must be supplied in the order given in ?CovarianceFct

Value

this is just a wrapper for CovarianceFct
guessinterp

guessinterp function

Description

A function to guess provisional interpolational methods to variables in a data frame. Numeric variables are assigned interpolation by areal weighted mean (see below); factor, character and other types of variable are assigned interpolation by majority vote (see below). Not that the interpolation type ArealWeightedSum is not assigned automatically.

Usage

guessinterp(df)

Arguments

df a data frame

Details

The three types of interpolation method employed in the package lgcp are:

1. 'Majority' The interpolated value corresponds to the value of the covariate occupying the largest area of the computational cell.
2. 'ArealWeightedMean' The interpolated value corresponds to the mean of all covariate values contributing to the computational cell weighted by their respective areas.
3. 'ArealWeightedSum' The interpolated value is the sum of all contributing covariates weighed by the proportion of area with respect to the covariate polygons. For example, suppose region A has the same area as a computational grid cell and has 500 inhabitants. If that region occupies half of a computational grid cell, then this interpolation type assigns 250 inhabitants from A to the computational grid cell.

Value

the data frame, but with attributes describing the interpolation method for each variable

See Also

chooseCellwidth, getpolyol, getZmat, addTemporalCovariates, lgcpPrior, lgcpInits, CovFunction lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatialPlusPars, lgcpPredictMultitypeSpatialPlusPars

Examples

## Not run: spdf a SpatialPolygonsDataFrame
## Not run: spdf@data <- guessinterp(spdf@data)
hasNext  

**generic hasNext method**

**Description**

test if an iterator has any more values to go

**Usage**

hasNext(obj)

**Arguments**

obj an iterator

hasNext.iter  

**hasNext.iter function**

**Description**

method for iter objects test if an iterator has any more values to go

**Usage**

## S3 method for class 'iter'

hasNext(obj)

**Arguments**

obj an iterator

hvals  

**hvals function**

**Description**

Generic function to return the values of the proposal scaling $h$ in the MCMC algorithm.

**Usage**

hvals(obj, ...)

Arguments

obj an object

... additional arguments

Value

method hvals

hvals.lgcpPredict

hvals.lgcpPredict function

Description

Accessor function returning the value of h, the MALA proposal scaling constant over the iterations of the algorithm for objects of class lgcpPredict

Usage

## S3 method for class 'lgcpPredict'
hvals(obj, ...)

Arguments

obj an object of class lgcpPredict

... additional arguments

Value

returns the values of h taken during the progress of the algorithm

See Also

lgcpPredict
identify.lgcpPredict function

Description

Identifies the indices of grid cells on plots of lgcpPredict objects. Can be used to identify a small number of cells for further information e.g. trace or autocorrelation plots (provided data has been dumped to disk). On calling identify(lg) for example (see code below), the user can click multiply with the left mouse button on the graphics device; once the user has selected all points of interest, the right button is pressed, which returns them.

Usage

## S3 method for class 'lgcpPredict'
identify(x, ...)

Arguments

x an object of class lgcpPredict
...
additional arguments

Value

a 2 x n matrix containing the grid indices of the points of interest, where n is the number of points selected via the mouse.

See Also

lgcpPredict, loc2poly

Examples

## Not run: plot(lg) # lg an lgcpPredict object  
## Not run: pt_indices <- identify(lg)

identifygrid function

Description

Identifies the indices of grid cells on plots of objects.

Usage

identifygrid(x, y)
**image.lgcpgrid**

**Arguments**

- `x`  the x grid centroids
- `y`  the y grid centroids

**Value**

- A 2 x n matrix containing the grid indices of the points of interest, where n is the number of points selected via the mouse.

**See Also**

- `lgcpPredict`, `loc2poly`, `identify.lgcpPredict`
initialiseAMCMC

Description

A generic to be used for the purpose of user-defined adaptive MCMC schemes, initialiseAMCMC tells the MALA algorithm which value of h to use first. See lgcp vignette, codevignette("lgcp"), for further details on writing adaptive MCMC schemes.

Usage

initialiseAMCMC(obj, ...)

Arguments

obj an object
...
additional arguments

Value

method initialiseAMCMC

See Also

initialiseAMCMC.constanth, initialiseAMCMC.andrieuthomsh

initialiseAMCMC.andrieuthomsh

Description

Initialises the andrieuthomsh adaptive scheme.

Usage

## S3 method for class 'andrieuthomsh'
initialiseAMCMC(obj, ...)

Arguments

obj an object
...
additional arguments
**initialiseAMCMC.constanth**

**Value**

initial h for scheme

**References**


**See Also**

andrieuthomsh

---

**initialiseAMCMC.constanth function**

**Description**

Initialises the **constanth** adaptive scheme.

**Usage**

```r
## S3 method for class 'constanth'
initialiseAMCMC(obj, ...)
```

**Arguments**

- `obj` an object
- `...` additional arguments

**Value**

initial h for scheme

**See Also**

**constanth**
integerise

integerise function

Description

Generic function for converting the time variable of an stppp object.

Usage

integerise(obj, ...)

Arguments

obj an object
... additional arguments

Value

method integerise

See Also

integerise.stppp

integerise.mstppp

integerise.mstppp function

Description

Function for converting the times and time limits of an mstppp object into integer values.

Usage

## S3 method for class 'mstppp'
integerise(obj, ...)

Arguments

obj an mstppp object
... additional arguments

Value

The mstppp object, but with integerised times.
integerise.stppp

integerise.stppp function

Description

Function for converting the times and time limits of an stppp object into integer values. Do this before estimating \( \mu(t) \), and hence before creating the temporalAtRisk object. Not taking this step is possible in lgcp, but can cause minor complications connected with the scaling of \( \mu(t) \).

Usage

```r
## S3 method for class 'stppp'
integerise(obj, ...)
```

Arguments

- `obj`: an stppp object
- `...`: additional arguments

Value

The stppp object, but with integerised times.

intens

intens function

Description

Generic function to return the Poisson Intensity.

Usage

```r
intens(obj, ...)
```

Arguments

- `obj`: an object
- `...`: additional arguments

Value

method intens

See Also

lgcpPredict, intens.lgcpPredict
intens.lgcpPredict

**intens.lgcpPredict function**

**Description**

Accessor function returning the Poisson intensity as an lgcpgrid object.

**Usage**

```r
## S3 method for class 'lgcpPredict'
intens(obj, ...)
```

**Arguments**

- `obj` an lgcpPredict object
- `...` additional arguments

**Value**

the cell-wise mean Poisson intensity, as computed by MCMC.

**See Also**

lgcpPredict

---

intens.lgcpSimMultitypeSpatialPlusParameters

**intens.lgcpSimMultitypeSpatialPlusParameters function**

**Description**

A function to return the cellwise Poisson intensity used during in constructing the simulated data.

**Usage**

```r
"intens(obj, ...)"
```

**Arguments**

- `obj` an object of class lgcpSimMultitypeSpatialPlusParameters
- `...` other parameters

**Value**

the Poisson intensity
**intens.lgcpSimSpatialPlusParameters**

**intens.lgcpSimSpatialPlusParameters function**

**Description**

A function to return the cellwise Poisson intensity used during in constructing the simulated data.

**Usage**

```r
## S3 method for class 'lgcpSimSpatialPlusParameters'
intens(obj, ...)
```

**Arguments**

- `obj`: an object of class `lgcpSimSpatialPlusParameters`
- `...`: other parameters

**Value**

the Poisson intensity

---

**interptypes**

**interptypes function**

**Description**

A function to return the types of covariate interpolation available.

**Usage**

```r
interptypes()
```

**Details**

The three types of interpolation method employed in the package `lgcp` are:

1. 'Majority' The interpolated value corresponds to the value of the covariate occupying the largest area of the computational cell.
2. 'ArealWeightedMean' The interpolated value corresponds to the mean of all covariate values contributing to the computational cell weighted by their respective areas.
3. 'ArealWeightedSum' The interpolated value is the sum of all contributing covariates weighed by the proportion of area with respect to the covariate polygons. For example, suppose region A has the same area as a computational grid cell and has 500 inhabitants. If that region occupies half of a computational grid cell, then this interpolation type assigns 250 inhabitants from A to the computational grid cell.
Value

character string of available interpolation types

inversebase  inversebase function

Description

A function to compute the base of the inverse of a block circulant matrix, given the base of the matrix

Usage

inversebase(x)

Arguments

x  the base matrix of a block circulant matrix

Value

the base matrix of the inverse of the circulant matrix

is.burnin  is this a burn-in iteration?

Description

if this mcmc iteration is in the burn-in period, return TRUE

Usage

is.burnin(obj)

Arguments

obj  an mcmc iterator

Value

TRUE or FALSE
is.pow2

is.pow2 function

Description
Tests whether a number is a power of 2.

Usage
is.pow2(num)

Arguments
num a numeric

Value
logical: is num a power of 2?

Examples
is.pow2(128) # TRUE
is.pow2(64.9) # FALSE

is.retain

do we retain this iteration?

Description
if this mcmc iteration is one not thinned out, this is true

Usage
is.retain(obj)

Arguments
obj an mcmc iterator

Value
TRUE or FALSE
is.SPD

Description

A function to compute whether a block circulant matrix is symmetric positive definite (SPD), given its base matrix.

Usage

is.SPD(base)

Arguments

base base matrix of a block circulant matrix

Value

logical, whether the circulant matrix the base represents is SPD

iteration

Description

within a loop, this is the iteration number we are currently doing.

Usage

iteration(obj)

Arguments

obj an mcmc iterator

Details

get the iteration number

Value

integer iteration number, starting from 1.
KinhomAverage

**KinhomAverage function**

**Description**

A function to estimate the inhomogeneous K function for a spatiotemporal point process. The method of computation is similar to `ginhomAverage`, see eq (8) Diggle P, Rowlingson B, Su T (2005) to see how this is computed.

**Usage**

```r
KinhomAverage(
  xyt, 
  spatial.intensity, 
  temporal.intensity, 
  time.window = xyt$tlim, 
  rvals = NULL, 
  correction = "iso", 
  suppresswarnings = FALSE
)
```

**Arguments**

- **xyt**: an object of class stppp
- **spatial.intensity**: A spatialAtRisk object
- **temporal.intensity**: A temporalAtRisk object
- **time.window**: time interval contained in the interval xyt$tlim over which to compute average. Useful if there is a lot of data over a lot of time points.
- **rvals**: Vector of values for the argument r at which the inhomogeneous K function should be evaluated (see ?Kinhom). There is a sensible default.
- **correction**: choice of edge correction to use, see ?Kinhom, default is Ripley isotropic correction
- **suppresswarnings**: Whether or not to suppress warnings generated by Kinhom

**Value**

time average of inhomogenous K function.

**References**


See Also

ginhomAverage, spatialparsEst, thetaEst, lambdaEst, muEst

---

**lambdaEst**

**Description**

Generic function for estimating bivariate densities by eye. Specific methods exist for stppp objects and ppp objects.

**Usage**

`lambdaEst(xyt, ...)`

**Arguments**

- `xyt`: an object
- `...`: additional arguments

**Value**

method lambdaEst

**See Also**

lambdaEst.stppp, lambdaEst.ppp
**Description**

A tool for the visual estimation of lambda(s) via a 2 dimensional smoothing of the case locations. For parameter estimation, the alternative is to estimate lambda(s) by some other means, convert it into a spatialAtRisk object and then into a pixel image object using the build in coercion methods, this im object can then be fed to `ginhomAverage`, `KinhomAverage` or `thetaEst` for instance.

**Usage**

```r
## S3 method for class 'ppp'
lambdaEst(xyt, weights = c(), edge = TRUE, bw = NULL, ...)
```

**Arguments**

- `xyt` object of class stppp
- `weights` Optional vector of weights to be attached to the points. May include negative values. See `?density.ppp`.
- `edge` Logical flag: if TRUE, apply edge correction. See `?density.ppp`.
- `bw` optional bandwidth. Set to NULL by default, which calls teh resolve.2D.kernel function for computing an initial value of this
- `...` arguments to be passed to plot

**Details**

The function `lambdaEst` is built directly on the `density.ppp` function and as such, implements a bivariate Gaussian smoothing kernel. The bandwidth is initially that which is automatically chosen by the default method of `density.ppp`. Since image plots of these kernel density estimates may not have appropriate colour scales, the ability to adjust this is given with the slider 'colour adjustment'. With colour adjustment set to 1, the default image.plot for the equivalent pixel image object is shown and for values less than 1, the colour scheme is more spread out, allowing the user to get a better feel for the density that is being fitted. NOTE: colour adjustment does not affect the returned density and the user should be aware that the returned density will 'look like' that displayed when colour adjustment is set equal to 1.

**Value**

This is an rpanel function for visual choice of lambda(s), the output is a variable, varname, with the density *per unit time* the variable varname can be fed to the function `ginhomAverage` or `KinhomAverage` as the argument density (see for example `?ginhomAverage`), or into the function `thetaEst` as the argument spatial.intensity.
References


See Also

spatialAtRisk, ginhomAverage, KinhomAverage, spatialparsEst, thetaEst, muEst

lambdaEst.stppp  lambdaEst.stppp function

Description

A tool for the visual estimation of lambda(s) via a 2 dimensional smoothing of the case locations. For parameter estimation, the alternative is to estimate lambda(s) by some other means, convert it into a spatialAtRisk object and then into a pixel image object using the build in coercion methods, this im object can then be fed to ginhomAverage, KinhomAverage or thetaEst for instance.

Usage

## S3 method for class 'stppp'
lambdaEst(xyt, weights = c(), edge = TRUE, bw = NULL, ...)

Arguments

xyt object of class stppp
weights Optional vector of weights to be attached to the points. May include negative values. See ?density.ppp.
edge Logical flag: if TRUE, apply edge correction. See ?density.ppp.
bw optional bandwidth. Set to NULL by default, which calls teh resolve.2D.kernel function for computing an initial value of this
...
arguments to be passed to plot

Details

The function lambdaEst is built directly on the density.ppp function and as such, implements a bivariate Gaussian smoothing kernel. The bandwidth is initially that which is automatically chosen by the default method of density.ppp. Since image plots of these kernel density estimates may not have appropriate colour scales, the ability to adjust this is given with the slider ‘colour adjustment’. With colour adjustment set to 1, the default image.plot for the equivalent pixel image object is shown and for values less than 1, the colour scheme is more spread out, allowing the user to get a better feel for the density that is being fitted. NOTE: colour adjustment does not affect the returned density and the user should be aware that the returned density will 'look like' that displayed when colour adjustment is set equal to 1.
This is an rpanel function for visual choice of lambda(s), the output is a variable, varname, with the density *per unit time* the variable varname can be fed to the function ginhomAverage or KinhomAverage as the argument density (see for example ?ginhomAverage), or into the function thetaEst as the argument spatial.intensity.

References


See Also

spatialAtRisk, ginhomAverage, KinhomAverage, spatialparsEst, thetaEst, muEst
lgcpForecast function

Description

Function to produce forecasts for the mean field \( Y \) at times beyond the last time point in the analysis (given by the argument \( T \) in the function \texttt{lgcpPredict}).

Usage

\begin{verbatim}
lgcpForecast(
  lg, ptimes, spatial.intensity, temporal.intensity, inclusion = "touching"
)
\end{verbatim}

Arguments

- \( lg \) : an object of class \texttt{lgcpPredict}
- \( ptimes \) : vector of time points for prediction. Must start strictly after last inferred time point.
- \( spatial.intensity \) : the fixed spatial component: an object of that can be coerced to one of class \texttt{spatialAtRisk}
- \( temporal.intensity \) : the fixed temporal component: either a numeric vector, or a function that can be coerced into an object of class \texttt{temporalAtRisk}
- \( inclusion \) : criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.

Value

Forecasted relative risk, Poisson intensities and \( Y \) values over grid, together with approximate variance.

References


See Also

\texttt{lgcpPredict}
Description

Generic function for the handling of list objects where each element of the list is a matrix. Each matrix is assumed to have the same dimension. Such objects arise from the various routines in the package lgcp.

Usage

lgcpgrid(grid, ...)

Arguments

grid a list object with each member of the list being a numeric matrix, each matrix having the same dimension

... other arguments

Details

lgcpgrid objects are list objects with names len, nrow, ncol, grid, xvals, yvals, zvals. The first three elements of the list store the dimension of the object, the fourth element, grid, is itself a list object consisting of matrices in which the data is stored. The last three arguments can be used to give what is effectively a 3 dimensional array a physical reference.

For example, the mean of Y from a call to lgcpPredict, obj$y.mean for example, is stored in an lgcpgrid object. If several time points have been stored in the call to lgcpPredict, then the grid element of the lgcpgrid object contains the output for each of the time points in succession. So the first element, obj$y.mean$grid[[1]], contains the output from the first time point and so on.

Value

method lgcpgrid

See Also

lgcpgrid.list, lgcpgrid.array, lgcpgrid.matrix
lgcpgrid.array

lgcpgrid.array function

Description

Creates an lgcp grid object from an 3-dimensional array.

Usage

```r
## S3 method for class 'array'
lgcpgrid(
  grid,
  xvals = 1:dim(grid)[1],
  yvals = 1:dim(grid)[2],
  zvals = 1:dim(grid)[3],
  ...
)
```

Arguments

- `grid`  
  a three dimensional array object
- `xvals`  
  optional vector of x-coordinates associated to grid. By default, this is the cell index in the x direction.
- `yvals`  
  optional vector of y-coordinates associated to grid. By default, this is the cell index in the y direction.
- `zvals`  
  optional vector of z-coordinates (time) associated to grid. By default, this is the cell index in the z direction.
- `...`  
  other arguments

Value

an object of class lgcpgrid

See Also

lgcpgrid.list, as.list.lgcpgrid, print.lgcpgrid, summary.lgcpgrid, quantile.lgcpgrid, image.lgcpgrid, plot.lgcpgrid
**lgcpgrid.list function**

**Description**

Creates an lgcpgrid object from a list object plus some optional coordinates. Note that each element of the list should be a matrix, and that each matrix should have the same dimension.

**Usage**

```r
## S3 method for class 'list'
lgcpgrid(
  grid,
  xvals = 1:dim(grid[[1]])[1],
  yvals = 1:dim(grid[[1]])[2],
  zvals = 1:length(grid),
  ...
)
```

**Arguments**

- `grid` a list object with each member of the list being a numeric matrix, each matrix having the same dimension
- `xvals` optional vector of x-coordinates associated to grid. By default, this is the cell index in the x direction.
- `yvals` optional vector of y-coordinates associated to grid. By default, this is the cell index in the y direction.
- `zvals` optional vector of z-coordinates (time) associated to grid. By default, this is the cell index in the z direction.
- `...` other arguments

**Value**

an object of class lgcpgrid

**See Also**

lgcpgrid.array, as.list.lgcpgrid, print.lgcpgrid, summary.lgcpgrid, quantile.lgcpgrid, image.lgcpgrid, plot.lgcpgrid
### lgcpgrid.matrix

**lgcpgrid.matrix function**

Description

Creates an lgcp grid object from an 2-dimensional matrix.

Usage

```r
# S3 method for class 'matrix'
lgcpgrid(grid, xvals = 1:nrow(grid), yvals = 1:ncol(grid), ...)
```

Arguments

- **grid**
  - a three dimensional array object
- **xvals**
  - optional vector of x-coordinates associated to grid. By default, this is the cell index in the x direction.
- **yvals**
  - optional vector of y-coordinates associated to grid. By default, this is the cell index in the y direction.
- **...**
  - other arguments

Value

- an object of class lgcpgrid

See Also

- `lgcpgrid.list`, `as.list.lgcpgrid`, `print.lgcpgrid`, `summary.lgcpgrid`, `quantile.lgcpgrid`, `image.lgcpgrid`, `plot.lgcpgrid`

---

### lgcpInits

**lgcpInits function**

Description

A function to declare initial values for a run of the MCMC routine. If specified, the MCMC algorithm will calibrate the proposal density using these as provisional estimates of the parameters.

Usage

```r
lgcpInits(etainit = NULL, betainit = NULL)
```
Arguments

etainit a vector, the initial value of eta to use
betainit a vector, the initial value of beta to use, this vector must have names the same as the variable names in the formula in use, and in the same order.

Details

It is not necessary to supply initial values to the MCMC routine, by default the functions lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatioTemporalPlusPars and lgcpPredictMultitypeSpatialPlusPars will initialise the MCMC as follows. For eta, if no initial value is specified then the initial value of eta in the MCMC run will be the prior mean. For beta, if no initial value is specified then the initial value of beta in the MCMC run will be estimated from an overdispersed Poisson fit to the cell counts, ignoring spatial correlation. The user cannot specify an initial value of Y (or equivalently Gamma), as a sensible value is chosen by the MCMC function.

A secondary function of specifying initial values is to help design the MCMC proposal matrix, which is based on these initial estimates.

Value

an object of class lgcpInits used in the MCMC routine.

See Also


Examples

## Not run: INITS <- lgcpInits(etainit=log(c(sqrt(1.5),275)), betainit=NULL)

---

lgcppars function

Description

A function for setting the parameters sigma, phi and theta for lgcpPredict. Note that the returned set of parameters also features mu=-0.5*sigma^2, gives mean(exp(Y)) = 1.

Usage

lgcppars(sigma = NULL, phi = NULL, theta = NULL, mu = NULL, beta = NULL)
Arguments

sigma  
sigma parameter

phi  
phi parameter

theta  
this is 'beta' parameter in Brix and Diggle (2001)

mu  
the mean of the latent field, if equal to NULL, this is set to -sigma^2/2

beta  
ONLY USED IN case where there is covariate information.

See Also

lgcpPredict

Description

The function lgcpPredict performs spatiotemporal prediction for log-Gaussian Cox Processes

Usage

lgcpPredict(
  xyt,
  T,
  laglength,
  model.parameters = lgcppars(),
  spatial.covmodel = "exponential",
  covpars = c(),
  cellwidth = NULL,
  gridsize = NULL,
  spatial.intensity,
  temporal.intensity,
  mcmc.control,
  output.control = setoutput(),
  missing.data.areas = NULL,
  autorotate = FALSE,
  gradtrunc = Inf,
  ext = 2,
  inclusion = "touching"
)

Arguments

xyt  
a spatio-temporal point pattern object, see ?stppp

T  
time point of interest
laglength

specifies lag window, so that data from and including time (T-laglength) to time T is used in the MALA algorithm.

model.parameters

values for parameters, see ?lgcppars

spatial.covmodel

correlation type see ?CovarianceFct

covpars

vector of additional parameters for certain classes of covariance function (eg Matern), these must be supplied in the order given in ?CovarianceFct

cellwidth

width of grid cells on which to do MALA (grid cells are square) in same units as observation window. Note EITHER gridsize OR cellwidth must be specified.

gridsize

size of output grid required. Note EITHER gridsize OR cellwidth must be specified.

spatial.intensity

the fixed spatial component: an object of that can be coerced to one of class spatialAtRisk

temporal.intensity

the fixed temporal component: either a numeric vector, or a function that can be coerced into an object of class temporalAtRisk

mcmc.control

MCMC parameters, see ?mcmcpars

output.control

output choice, see ?setoutput

missing.data.areas

a list of owin objects (of length laglength+1) which has xyt$window as a base window, but with polygonal holes specifying spatial areas where there is missing data.

autorotate

logical: whether or not to automatically do MCMC on optimised, rotated grid.

gradtrunc

truncation for gradient vector equal to H parameter Moller et al 1998 pp 473. Default is Inf, which means no gradient truncation. Set to NULL to estimate this automatically (though note that this may not necessarily be a good choice). The default seems to work in most settings.

ext

integer multiple by which grid should be extended, default is 2. Generally this will not need to be altered, but if the spatial correlation decays very slowly (compared with the size of the observation window), increasing 'ext' may be necessary.

inclusion

criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.

further notes on autorotate argument: If set to TRUE, and the argument spatial is not NULL, then the argument spatial must be computed in the original frame of reference (ie NOT in the rotated frame). Autorotate performs bilinear interpolation (via interp.im) on an inverse transformed grid; if there is no computational advantage in doing this, a warning message will be issued. Note that best accuracy is achieved by manually rotating xyt and then computing spatial on the transformed xyt and finally feeding these in as arguments to the function lgcpPredict. By default autorotate is set to FALSE.
Details

The following is a mathematical description of a log-Gaussian Cox Process, it is best viewed in the
pdf version of the manual.

Let \( Y(s, t) \) be a spatiotemporal Gaussian process, \( W \subset \mathbb{R}^2 \) be an observation window in space
and \( T \subset \mathbb{R}_{\geq 0} \) be an interval of time of interest. Cases occur at spatio-temporal positions \((x, t) \in W \times T\) according to an inhomogeneous spatio-temporal Cox process, i.e. a Poisson process with
a stochastic intensity \( R(x, t) \). The number of cases, \( X_{S, [t_1, t_2]} \), arising in any \( S \subseteq W \) during the
interval \([t_1, t_2] \subseteq T\) is then Poisson distributed conditional on \( R(\cdot) \).

\[
X_{S, [t_1, t_2]} \sim \text{Poisson} \left\{ \int_S \int_{t_1}^{t_2} R(s, t) \, ds \, dt \right\}
\]

Following Brix and Diggle (2001) and Diggle et al (2005), the intensity is decomposed multiplica-
tively as

\[
R(s, t) = \lambda(s) \mu(t) \exp\{Y(s, t)\}.
\]

In the above, the fixed spatial component, \( \lambda : \mathbb{R}^2 \mapsto \mathbb{R}_{\geq 0} \), is a known function, proportional to the
population at risk at each point in space and scaled so that

\[
\int_W \lambda(s) \, ds = 1,
\]

whilst the fixed temporal component, \( \mu : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0} \), is also a known function with

\[
\mu(t) \delta t = E[X_{W, \delta t}],
\]

for \( t \) in a small interval of time, \( \delta t \), over which the rate of the process over \( W \) can be considered
constant.

NOTE: the xyt stpp object can be recorded in continuous time, but for the purposes
of prediction, discretisation must take place. For the time dimension, this is achieved invisibly
by \texttt{as.integer(xyt$t)} and \texttt{as.integer(xyt$tlim)}. Therefore, before running an analysis
please make sure that this is commensurate with the physical interpretation and requirements
of your output. The spatial discretisation is chosen with the argument cellwidth (or gridsize).
If the chosen discretisation in time and space is too coarse for a given set of parameters (sigma,
phi and theta) then the proper correlation structures implied by the model will not be captured
in the output.

Before calling this function, the user must decide on the time point of interest, the number of in-
tervals of data to use, the parameters, spatial covariance model, spatial discretisation, fixed spatial
(\( \lambda(s) \)) and temporal (\( \mu(t) \)) components, mcmc parameters, and whether or not any output is re-
quired.

Value

the results of fitting the model in an object of class \texttt{lgcpPredict}

References

of Statistical Software, 52(4), 1-40. URL http://www.jstatsoft.org/v52/i04/


See Also

KinhomAverage, ginhomAverage, lambdaEst, muEst, spatialparsEst, thetaEst, spatialAtRisk, temporalAtRisk, lgcppars, CovarianceFct, mcmcpars, setoutput print.lgcpPredict, xvals.lgcpPredict, yvals.lgcpPredict, plot.lgcpPredict, meanfield.lgcpPredict, rr.lgcpPredict, serr.lgcpPredict, intens.lgcpPredict, varfield.lgcpPredict, gridfun.lgcpPredict, gridav.lgcpPredict, hvals.lgcpPredict, window.lgcpPredict, mcmctrace.lgcpPredict, plotExceed.lgcpPredict, quantile.lgcpPredict, expectation.lgcpPredict, extract.lgcpPredict, showGrid.lgcpPredict

lgcpPredictAggregated function

Description

The function lgcpPredict performs spatiotemporal prediction for log-Gaussian Cox Processes for point process data where counts have been aggregated to the regional level. This is achieved by imputation of the regional counts onto a spatial continuum; if something is known about the underlying spatial density of cases, then this information can be added to improve the quality of the imputation, without this, the counts are distributed uniformly within regions.

Usage

```
lgcpPredictAggregated(
  app,
  popden = NULL,
  T,
  laglength,
  model.parameters = lgcppars(),
  spatial.covmodel = "exponential",
  covpars = c(),
  cellwidth = NULL,
  gridsize = NULL,
  spatial.intensity,
  temporal.intensity,
  mcmc.control,
  output.control = setoutput(),
  autorotate = FALSE,
)```

gradtrunc = NULL,
n = 100,
dmin = 0,
check = TRUE
)

Arguments

app a spatio-temporal aggregated point pattern object, see ?stapp
popden a spatialAtRisk object of class 'fromFunction' describing the population density, if known. Default is NULL, which gives a uniform density on each region.
T time point of interest
laglength specifies lag window, so that data from and including time (T-laglength) to time T is used in the MALA algorithm
model.parameters values for parameters, see ?lgcppars
spatial.covmodel correlation type see ?CovarianceFct
covpars vector of additional parameters for certain classes of covariance function (e.g. Matern), these must be supplied in the order given in ?CovarianceFct
cellwidth width of grid cells on which to do MALA (grid cells are square). Note EITHER gridsize OR cellwidth must be specified.
gridsize size of output grid required. Note EITHER gridsize OR cellwidth must be specified.
spatial.intensity the fixed spatial component: an object of that can be coerced to one of class spatialAtRisk
temporal.intensity the fixed temporal component: either a numeric vector, or a function that can be coerced into an object of class temporalAtRisk
mcmc.control MCMC parameters, see ?mcmcpars
output.control output choice, see ?setoutput
autorotate logical: whether or not to automatically do MCMC on optimised, rotated grid.
gradtrunc truncation for gradient vector equal to H parameter Moller et al 1998 pp 473. Set to NULL to estimate this automatically (default). Set to zero for no gradient truncation.
n parameter for as.stpp. If popden is NULL, then this parameter controls the resolution of the uniform. Otherwise if popden is of class 'fromFunction', it controls the size of the imputation grid used for sampling. Default is 100.
dmin parameter for as.stpp. If any reginal counts are missing, then a set of polygonal 'holes' in the observation window will be computed for each. dmin is the parameter used to control the simplification of these holes (see ?simplify.owin). default is zero.
check logical parameter for as.stppp. If any regional counts are missing, then roughly speaking, check specifies whether to check the 'holes'. Further notes on autorotate argument: If set to TRUE, and the argument spatial is not NULL, then the argument spatial must be computed in the original frame of reference (i.e., not in the rotated frame). Autorotate performs bilinear interpolation (via interp.im) on an inverse transformed grid; if there is no computational advantage in doing this, a warning message will be issued. Note that best accuracy is achieved by manually rotating xyt and then computing spatial on the transformed xyt and finally feeding these in as arguments to the function lgcpPredict. By default autorotate is set to FALSE.

Details

The following is a mathematical description of a log-Gaussian Cox Process, it is best viewed in the pdf version of the manual.

Let \( Y(s, t) \) be a spatiotemporal Gaussian process, \( W \subset \mathbb{R}^2 \) be an observation window in space and \( T \subset \mathbb{R}_{\geq 0} \) be an interval of time of interest. Cases occur at spatio-temporal positions \((x, t) \in W \times T\) according to an inhomogeneous spatio-temporal Cox process, i.e., a Poisson process with a stochastic intensity \( R(x, t) \). The number of cases, \( X_{S, [t_1, t_2]} \), arising in any \( S \subseteq W \) during the interval \([t_1, t_2] \subseteq T\) is then Poisson distributed conditional on \( R(\cdot) \),

\[
X_{S, [t_1, t_2]} \sim \text{Poisson} \left\{ \int_S \int_{t_1}^{t_2} R(s, t) \, ds \, dt \right\}
\]

Following Brix and Diggle (2001) and Diggle et al (2005), the intensity is decomposed multiplicatively as

\[
R(s, t) = \lambda(s) \mu(t) \exp\{Y(s, t)\}.
\]

In the above, the fixed spatial component, \( \lambda : \mathbb{R}^2 \mapsto \mathbb{R}_{\geq 0} \), is a known function, proportional to the population at risk at each point in space and scaled so that

\[
\int_W \lambda(s) \, ds = 1,
\]

whilst the fixed temporal component, \( \mu : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0} \), is also a known function with

\[
\mu(t) \delta t = \mathbb{E}[X_{W, \delta t}],
\]

for \( t \) in a small interval of time, \( \delta t \), over which the rate of the process over \( W \) can be considered constant.

NOTE: the xyt stppp object can be recorded in continuous time, but for the purposes of prediction, discretisation must take place. For the time dimension, this is achieved invisibly by \texttt{as.integer(xyt$t)} and \texttt{as.integer(xyt$tlim)}. Therefore, before running an analysis please make sure that this is commensurate with the physical interpretation and requirements of your output. The spatial discretisation is chosen with the argument cellwidth (or gridsize). If the chosen discretisation in time and space is too coarse for a given set of parameters (sigma, phi and theta) then the proper correlation structures implied by the model will not be captured in the output.

Before calling this function, the user must decide on the time point of interest, the number of intervals of data to use, the parameters, spatial covariance model, spatial discretisation, fixed spatial (\( \lambda(s) \)) and temporal (\( \mu(t) \)) components, mcmc parameters, and whether or not any output is required.
Value

the results of fitting the model in an object of class `lgcpPredict`

References


See Also

`KinhomAverage`, `ginhomAverage`, `lambdaEst`, `muEst`, `spatialparsEst`, `thetaEst`, `spatialAtRisk`, `temporalAtRisk`, `lgcppars`, `CovarianceFct`, `mcmcpars`, `setoutput`, `print.lgcpPredict`, `xvals.lgcpPredict`, `yvals.lgcpPredict`, `plot.lgcpPredict`, `meanfield.lgcpPredict`, `rr.lgcpPredict`, `serr.lgcpPredict`, `intens.lgcpPredict`, `varfield.lgcpPredict`, `gridfun.lgcpPredict`, `gridav.lgcpPredict`, `hvals.lgcpPredict`, `window.lgcpPredict`, `mcmctrace.lgcpPredict`, `plotExceed.lgcpPredict`, `quantile.lgcpPredict`, `identify.lgcpPredict`, `expectation.lgcpPredict`, `extract.lgcpPredict`, `showGrid.lgcpPredict`
Arguments

formula a formula object of the form X ~ var1 + var2 etc. The name of the dependent variable must be "X". Only accepts 'simple' formulae, such as the example given.

spdf a SpatialPolygonsDataFrame object with variable "X", the event counts per region.

Zmat design matrix Z (see below) constructed with getZmat

overlayInZmat if the covariate information in Zmat also comes from spdf, set to TRUE to avoid replicating the overlay operations. Default is FALSE.

model.priors model priors, set using lgcpPrior

model.inits model initial values. The default is NULL, in which case lgcp will use the prior mean to initialise eta and beta will be initialised from an oversispersed glm fit to the data. Otherwise use lgcpInits to specify.

spatial.covmodel choice of spatial covariance function. See ?CovFunction

cellwidth the width of computational cells

poisson.offset A SpatialAtRisk object defining lambda (see below)

mcmc.control MCMC parameters, see ?mcmcpars

output.control output choice, see ?setoutput

gradtrunc truncation for gradient vector equal to H parameter Moller et al 1998 pp 473. Default is Inf, which means no gradient truncation, which seems to work in most settings.

ext integer multiple by which grid should be extended, default is 2. Generally this will not need to be altered, but if the spatial correlation decays slowly, increasing 'ext' may be necessary.

Nfreq the sampling frequency for the cell counts. Default is every 101 iterations.

inclusion criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former, the default, includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.

overlapping logical does spdf contain overlapping polygons? Default is FALSE. If set to TRUE, spdf can contain a variable named 'sintens' that gives the sampling intensity for each polygon; the default is to assume that cases are evenly split between overlapping regions.
optional matrix of dimension \((NM) \times \text{number of regions in spdf}\) where \(M, N\) are the number of cells in the \(x\) and \(y\) directions (not the number of cells on the Fourier grid, rather the number of cell on the output grid). The \(i\)th row of this matrix are the probabilities that for the \(i\)th grid cell (in the same order as expand.grid(mcens,ncens)) a case belongs to each of the regions in spdf. Including this object overrides 'sintens' in the overlapping option above.

**Details**

See the vignette "Bayesian_lgcp" for examples of this code in use.

In this case, we OBSERVE case counts in the regions of a SpatialPolygonsDataFrame; the counts are stored as a variable, \(X\). The model for the UNOBSERVED data, \(X(s)\), is as follows:

\[
X(s) \sim \text{Poisson}[R(s)]
\]

\[
R(s) = C_A \lambda(s) \exp[Z(s)\beta+Y(s)]
\]

Here \(X(s)\) is the number of events in the cell of the computational grid containing \(s\), \(R(s)\) is the Poisson rate, \(C_A\) is the cell area, \(\lambda(s)\) is a known offset, \(Z(s)\) is a vector of measured covariates and \(Y(s)\) is the latent Gaussian process on the computational grid. The other parameters in the model are \(\beta\), the covariate effects; and \(\eta=[\log(\sigma),\log(\phi)]\), the parameters of the process \(Y\) on an appropriately transformed (in this case log) scale.

We recommend the user takes the following steps before running this method:

1. Compute approximate values of the parameters, \(\eta\), of the process \(Y\) using the function `minimum.contrast`. These approximate values are used for two main reasons: (1) to help inform the size of the computational grid, since we will need to use a cell width that enables us to capture the dependence properties of \(Y\) and (2) to help inform the proposal kernel for the MCMC algorithm.

2. Choose an appropriate grid on which to perform inference using the function `chooseCellwidth`; this will partly be determined by the results of the first stage and partly by the available computational resource available to perform inference.

3. Using the function `getpolyol`, construct the computational grid and polygon overlays, as required. As this can be an expensive step, we recommend that the user saves this object after it has been constructed and in future reference to the data, reloads this object, rather than having to re-compute it (provided the computational grid has not changed).

4. Decide on which covariates are to play a part in the analysis and use the `lgcp` function `getZmat` to interpolate these onto the computational grid. Note that having saved the results from the previous step, this is a relatively quick operation, and allows the user to quickly construct different design matrices, \(Z\), from different candidate models for the data.

5. If required, set up the population offset using `SpatialAtRisk` functions (see the vignette "Bayesian_lgcp"); specify the priors using `lgcpPrior`; and if desired, the initial values for the MCMC, using the function `lgcpInits`. 
6. Run the MCMC algorithm and save the output to disk. We recommend dumping information to disk using the dump2dir function in the output.control argument because it offers much greater flexibility in terms of MCMC diagnosis and post-processing.

7. Perform post-processing analyses including MCMC diagnostic checks and produce summaries of the posterior expectations we require for presentation. (see the vignette "Bayesian_lgcp" for further details). Functions of use in this step include traceplots, autocorr, parautocorr, itar, parsu- summary, priorpost, postcov, textsummary, expectation, exceedProbs and lgcp:::expectation.lgcpPredict

Value

an object of class lgcpPredictAggregateSpatialPlusParameters

References


See Also

linkchooseCellWidth, getpolyol, guessinterp, getZmat, addTemporalCovariates, lgcpPrior, lgcpInits, CovFunction lgcpPredictSpatialPlusPars, lgcpPredictSpatioTemporalPlusPars, lgcpPredictMultitypeSpatialPlusPars, ltar, autocorr, parautocorr, traceplots, parssummary, textsummary, priorpost, postcov, exceedProbs, betavals, etavals

lgcpPredictMultitypeSpatialPlusPars

lgcpPredictMultitypeSpatialPlusPars function

Description

A function to deliver fully Bayesian inference for a multitype spatial log-Gaussian Cox process.
Usage

lgcpPredictMultitypeSpatialPlusPars(
  formulaList,
  sd,
  typemark = NULL,
  Zmat = NULL,
  model.priorsList,
  model.initsList = NULL,
  spatial.covmodelList,
  cellwidth = NULL,
  poisson.offset = NULL,
  mcmc.control,
  output.control = setoutput(),
  gradtrunc = Inf,
  ext = 2,
  inclusion = "touching"
)

Arguments

formulaList an object of class formulaList, see ?formulaList. A list of formulae of the form t1 ~ var1 + var2 etc. The name of the dependent variable must correspond to the name of the point type. Only accepts 'simple' formulae, such as the example given.

ds a marked ppp object, the mark of interest must be able to be coerced to a factor variable

typemark if there are multiple marks, run the MCMC algorithm for spatial point process data. Not for general purpose use.is sets the name of the mark by which

Zmat design matrix including all covariate effects from each point type, constructed with getZmat

model.priorsList model priors, a list object of length the number of types, each element set using lgcpPrior

model.initsList list of model initial values (of length the number of types). The default is NULL, in which case lgcp will use the prior mean to initialise eta and beta will be initialised from an oversispersed glm fit to the data. Otherwise use lgcpInits to specify.

spatial.covmodelList list of spatial covariance functions (of length the number of types). See ?Cov-Function

cellwidth the width of computational cells

poisson.offset A list of SpatialAtRisk objects (of length the number of types) defining lambda_k

mcmc.control MCMC paramters, see ?mcmcpars

output.control output choice, see ?setoutput
gradtrunc truncation for gradient vector equal to H parameter Moller et al 1998 pp 473. Default is Inf, which means no gradient truncation, which seems to work in most settings.

ext integer multiple by which grid should be extended, default is 2. Generally this will not need to be altered, but if the spatial correlation decays slowly, increasing 'ext' may be necessary.

inclusion criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former, the default, includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.

Details

See the vignette "Bayesian_lgcp" for examples of this code in use.

We suppose there are K point types of interest. The model for point-type k is as follows:

\[ X_k(s) \sim \text{Poisson}[R_k(s)] \]

\[ R_k(s) = C_A \lambda_k(s) \exp[Z_k(s)\beta_k+Y_k(s)] \]

Here \( X_k(s) \) is the number of events of type k in the computational grid cell containing the point s, \( R_k(s) \) is the Poisson rate, \( C_A \) is the cell area, \( \lambda_k(s) \) is a known offset, \( Z_k(s) \) is a vector of measured covariates and \( Y_i(s) \) where \( i = 1,\ldots,K+1 \) are latent Gaussian processes on the computational grid. The other parameters in the model are \( \beta_k \), the covariate effects for the kth type; and \( \eta_i = [\log(\sigma_i),\log(\phi_i)] \), the parameters of the process \( Y_i \) for \( i = 1,\ldots,K+1 \) on an appropriately transformed (again, in this case log) scale.

We recommend the user takes the following steps before running this method:

1. Compute approximate values of the parameters, \( \eta \), of the process \( Y \) using the function \texttt{minimum.contrast}. These approximate values are used for two main reasons: (1) to help inform the size of the computational grid, since we will need to use a cell width that enables us to capture the dependence properties of \( Y \) and (2) to help inform the proposal kernel for the MCMC algorithm.

2. Choose an appropriate grid on which to perform inference using the function \texttt{chooseCellwidth}; this will partly be determined by the results of the first stage and partly by the available computational resource available to perform inference.

3. Using the function \texttt{getpolyol}, construct the computational grid and polygon overlays, as required. As this can be an expensive step, we recommend that the user saves this object after it has been constructed and in future reference to the data, reloads this object, rather than having to re-compute it (provided the computational grid has not changed).

4. Decide on which covariates are to play a part in the analysis and use the \texttt{lgcp} function \texttt{getZmat} to interpolate these onto the computational grid. Note that having saved the results from the previous step, this is a relatively quick operation, and allows the user to quickly construct different design matrices, \( Z \), from different candidate models for the data.
5. If required, set up the population offset using SpatialAtRisk functions (see the vignette "Bayesian_lgcp"); specify the priors using lgcpPrior; and if desired, the initial values for the MCMC, using the function lgcpInits.

6. Run the MCMC algorithm and save the output to disk. We recommend dumping information to disk using the dump2dir function in the output.control argument because it offers much greater flexibility in terms of MCMC diagnosis and post-processing.

7. Perform post-processing analyses including MCMC diagnostic checks and produce summaries of the posterior expectations we require for presentation. (see the vignette "Bayesian_lgcp" for further details). Functions of use in this step include traceplots, autocorr, parautocorr, ltar, parsummary, priorpost, postcov, textsummary, expectation, exceedProbs and lgcp:::expectation.lgcpPredict

Value

an object of class lgcpPredictMultitypeSpatialPlusParameters

References


See Also

linkchooseCellWidth, getpolyol, guessinterp, getZmat, addTemporalCovariates, lgcpPrior, lgcpInits, CovFunction lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatioTemporalPlusPars, ltar, autocorr, parautocorr, traceplots, parssummary, textsummary, expectation, exceedProbs and lgcp:::expectation.lgcpPredict

142

lgcpPredictSpatial

Description

The function lgcpPredictSpatial performs spatial prediction for log-Gaussian Cox Processes
Usage

\texttt{lgcpPredictSpatial(}
\  \texttt{sd,}
\  \texttt{model.parameters = lgcppars(),}
\  \texttt{spatial.covmodel = "exponential",}
\  \texttt{covpars = c(),}
\  \texttt{cellwidth = NULL,}
\  \texttt{gridsize = NULL,}
\  \texttt{spatial.intensity,}
\  \texttt{spatial.offset = NULL,}
\  \texttt{mcmc.control,}
\  \texttt{output.control = setoutput(),}
\  \texttt{gradtrunc = Inf,}
\  \texttt{ext = 2,}
\  \texttt{inclusion = "touching"}
\)

Arguments

\texttt{sd} \hspace{1cm} \text{a spatial point pattern object, see \texttt{?ppp}}

\texttt{model.parameters} \hspace{1cm} \text{values for parameters, see \texttt{?lgcppars}}

\texttt{spatial.covmodel} \hspace{1cm} \text{correlation type see \texttt{?CovarianceFct}}

\texttt{covpars} \hspace{1cm} \text{vector of additional parameters for certain classes of covariance function (eg Matern), these must be supplied in the order given in \texttt{?CovarianceFct}}

\texttt{cellwidth} \hspace{1cm} \text{width of grid cells on which to do MALA (grid cells are square) in same units as observation window. Note EITHER gridsize OR cellwidth must be specified.}

\texttt{gridsize} \hspace{1cm} \text{size of output grid required. Note EITHER gridsize OR cellwidth must be specified.}

\texttt{spatial.intensity} \hspace{1cm} \text{the fixed spatial component: an object of that can be coerced to one of class \texttt{spatialAtRisk}}

\texttt{spatial.offset} \hspace{1cm} \text{Numeric of length 1. Optional offset parameter, corresponding to the expected number of cases. NULL by default, in which case, this is estimated from the data.}

\texttt{mcmc.control} \hspace{1cm} \text{MCMC parameters, see \texttt{?mcmcpars}}

\texttt{output.control} \hspace{1cm} \text{output choice, see \texttt{?setoutput}}

\texttt{gradtrunc} \hspace{1cm} \text{truncation for gradient vector equal to H parameter Moller et al 1998 pp 473. Default is Inf, which means no gradient truncation. Set to NULL to estimate this automatically (though note that this may not necessarily be a good choice). The default seems to work in most settings.}

\texttt{ext} \hspace{1cm} \text{integer multiple by which grid should be extended, default is 2. Generally this will not need to be altered, but if the spatial correlation decays slowly, increasing 'ext' may be necessary.}
inclusion criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former, the default, includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.

Details

The following is a mathematical description of a log-Gaussian Cox Process, it is best viewed in the pdf version of the manual.

Let \( \mathcal{Y}(s) \) be a spatial Gaussian process and \( W \subset \mathbb{R}^2 \) be an observation window in space. Cases occur at spatial positions \( x \in W \) according to an inhomogeneous spatial Cox process, i.e. a Poisson process with a stochastic intensity \( R(x) \). The number of cases, \( X_S \), arising in any \( S \subseteq W \) is then Poisson distributed conditional on \( R(\cdot) \),

\[
X_S \sim \text{Poisson} \left\{ \int_S R(s) ds \right\}
\]

Following Brix and Diggle (2001) and Diggle et al (2005) (but ignoring temporal variation), the intensity is decomposed multiplicatively as

\[
R(s,t) = \lambda(s) \exp \{\mathcal{Y}(s,t)\}.
\]

In the above, the fixed spatial component, \( \lambda : \mathbb{R}^2 \rightarrow \mathbb{R}_{\geq 0} \), is a known function, proportional to the population at risk at each point in space and scaled so that

\[
\int_W \lambda(s) ds = 1.
\]

Before calling this function, the user must decide on the parameters, spatial covariance model, spatial discretisation, fixed spatial \( (\lambda(s)) \) component, mcmc parameters, and whether or not any output is required. Note there is no autorotate option for this function.

Value

the results of fitting the model in an object of class \texttt{lgcpPredict}

References

See Also

lgcpPredict KinhomAverage, ginhomAverage, lambdaEst, muEst, spatialparsEst, thetaEst, spatialAtRisk, temporalAtRisk, lgcppars, CovarianceFct, mcmcpars, setoutput, print.lgcpPredict, xvals.lgcpPredict, yvals.lgcpPredict, plot.lgcpPredict, meanfield.lgcpPredict, rr.lgcpPredict, serr.lgcpPredict, intens.lgcpPredict, varfield.lgcpPredict, gridfun.lgcpPredict, gridav.lgcpPredict, hvals.lgcpPredict, window.lgcpPredict, mcmctrace.lgcpPredict, plotExceed.lgcpPredict, quantile.lgcpPredict, identify.lgcpPredict, expectation.lgcpPredict, extract.lgcpPredict, showGrid.lgcpPredict

lgcpPredictSpatialINLA
lgcpPredictSpatialINLA function

Description

!IMPORTANT! after library(lgcp) this will be a dummy function. In order to use, type getlgcpPredictSpatialINLA() at the console. This will download and install the true function.

Usage

lgcpPredictSpatialINLA(
  sd,
  ns,
  model.parameters = lgcppars(),
  spatial.covmodel = "exponential",
  covpars = c(),
  cellwidth = NULL,
  gridsize = NULL,
  spatial.intensity,
  ext = 2,
  optimverbose = FALSE,
  inlaverbose = TRUE,
  generic@hyper = list(theta = list(initial = 0, fixed = TRUE)),
  strategy = "simplified.laplace",
  method = "Nelder-Mead"
)

Arguments

sd a spatial point pattern object, see ?ppp
ns size of neighbourhood to use for GMRF approximation ns=1 corresponds to 3^2=9 eight neighbours around each point, ns=2 corresponds to 5^2=25 neighbours etc ...
model.parameters values for parameters, see ?lgcppars
The function `lgcpPredictSpatialINLA` performs spatial prediction for log-Gaussian Cox Processes using the integrated nested Laplace approximation.

The following is a mathematical description of a log-Gaussian Cox Process, it is best viewed in the pdf version of the manual.

Let \( \mathcal{Y}(s) \) be a spatial Gaussian process and \( W \subseteq \mathbb{R}^2 \) be an observation window in space. Cases occur at spatial positions \( x \in W \) according to an inhomogeneous spatial Cox process, i.e. a Poisson process with a stochastic intensity \( R(x) \). The number of cases, \( X_S \), arising in any \( S \subseteq W \) is then Poisson distributed conditional on \( R(\cdot) \),

\[
X_S \sim \text{Poisson} \left\{ \int_S R(s) ds \right\}
\]

Following Brix and Diggle (2001) and Diggle et al (2005) (but ignoring temporal variation), the intensity is decomposed multiplicatively as

\[
R(s, t) = \lambda(s) \exp\{\mathcal{Y}(s, t)\}.
\]

In the above, the fixed spatial component, \( \lambda : \mathbb{R}^2 \to \mathbb{R}_{\geq 0} \), is a known function, proportional to the population at risk at each point in space and scaled so that

\[
\int_W \lambda(s) ds = 1.
\]
Before calling this function, the user must decide on the parameters, spatial covariance model, spatial discretisation, fixed spatial ($\lambda(s)$) component and whether or not any output is required. Note there is no autorotate option for this function.

Value

the results of fitting the model in an object of class `lgcpPredict`

References


See Also

`lgcpPredict KinhomAverage, ginhomAverage, lambdaEst, muEst, spatialparsEst, thetaEst, spatialAtRisk, temporalAtRisk, lgcppars, CovarianceFct, mcmcpars, setoutput print.lgcpPredict, xvals.lgcpPredict, yvals.lgcpPredict, plot.lgcpPredict, meanfield.lgcpPredict, rr.lgcpPredict, serr.lgcpPredict, intens.lgcpPredict, varfield.lgcpPredict, gridfun.lgcpPredict, gridav.lgcpPredict, hvals.lgcpPredict, window.lgcpPredict, mcmctrace.lgcpPredict, plotExceed.lgcpPredict, quantile.lgcpPredict, identify.lgcpPredict, expectation.lgcpPredict, extract.lgcpPredict, showGrid.lgcpPredict`

---

**Description**

A function to deliver fully Bayesian inference for the spatial log-Gaussian Cox process.

**Usage**

```r
lgcpPredictSpatialPlusPars(
  formula,
  sd,
  Zmat = NULL,
  model.priors,
)```

model.inits = lgcpInits(),
spatial.covmodel,
cellwidth = NULL,
poisson.offset = NULL,
mcmc.control,
output.control = setoutput(),
gradtrunc = Inf,
ext = 2,
inclusion = "touching"
)

Arguments

formula a formula object of the form X ~ var1 + var2 etc. The name of the dependent variable must be "X". Only accepts 'simple' formulae, such as the example given.
sd a spatstat ppp object
Zmat design matrix Z (see below) constructed with getZmat
model.priors model priors, set using lgcpPrior
model.inits model initial values. The default is NULL, in which case lgcp will use the prior mean to initialise eta and beta will be initialised from an oversispersed glm fit to the data. Otherwise use lgcpInits to specify.
spatial.covmodel choice of spatial covariance function. See ?CovFunction
cellwidth the width of computational cells
poisson.offset A SpatialAtRisk object defining lambda (see below)
mcmc.control MCMC paramters, see ?mcmcpars
output.control output choice, see ?setoutput
gradtrunc truncation for gradient vector equal to H parameter Moller et al 1998 pp 473. Default is Inf, which means no gradient truncation, which seems to work in most settings.
ext integer multiple by which grid should be extended, default is 2. Generally this will not need to be altered, but if the spatial correlation decays slowly, increasing 'ext' may be necessary.
inclusion criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former, the default, includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.

Details

See the vignette "Bayesian_lgcp" for examples of this code in use.

The model for the data is as follows:
X(s) ~ Poisson[R(s)]

R(s) = C_A \lambda(s) \exp[Z(s)\beta+Y(s)]

Here X(s) is the number of events in the cell of the computational grid containing s, R(s) is the Poisson rate, C_A is the cell area, \lambda(s) is a known offset, Z(s) is a vector of measured covariates and Y(s) is the latent Gaussian process on the computational grid. The other parameters in the model are \beta, the covariate effects; and \eta=[\log(\sigma),\log(\phi)], the parameters of the process Y on an appropriately transformed (in this case log) scale.

We recommend the user takes the following steps before running this method:

1. Compute approximate values of the parameters, \eta, of the process Y using the function minimum.contrast. These approximate values are used for two main reasons: (1) to help inform the size of the computational grid, since we will need to use a cell width that enables us to capture the dependence properties of Y and (2) to help inform the proposal kernel for the MCMC algorithm.

2. Choose an appropriate grid on which to perform inference using the function chooseCellwidth; this will partly be determined by the results of the first stage and partly by the available computational resource available to perform inference.

3. Using the function getpolyol, construct the computational grid and polygon overlays, as required. As this can be an expensive step, we recommend that the user saves this object after it has been constructed and in future reference to the data, reloads this object, rather than having to re-compute it (provided the computational grid has not changed).

4. Decide on which covariates are to play a part in the analysis and use the lgcp function getZmat to interpolate these onto the computational grid. Note that having saved the results from the previous step, this is a relatively quick operation, and allows the user to quickly construct different design matrices, Z, from different candidate models for the data.

5. If required, set up the population offset using SpatialAtRisk functions (see the vignette "Bayesian_lgcp"); specify the priors using lgcpPrior; and if desired, the initial values for the MCMC, using the function lgcpInits.

6. Run the MCMC algorithm and save the output to disk. We recommend dumping information to disk using the dump2dir function in the output.control argument because it offers much greater flexibility in terms of MCMC diagnosis and post-processing.

7. Perform post-processing analyses including MCMC diagnostic checks and produce summaries of the posterior expectations we require for presentation. (see the vignette "Bayesian_lgcp" for further details). Functions of use in this step include traceplots, autocorr, parautocorr, ltr, parsummary, priorpost, postcov, textsummary, expectation, exceedProbs and lgcp:::expectation.lgcpPredict

Value

an object of class lgcpPredictSpatialOnlyPlusParameters

References


See Also

linkchooseCellWidth, getpolyol, guessinterp, getZmat, addTemporalCovariates, lgcpPrior, lgcpInits, CovFunction lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatioTemporalPlusPars, lgcpPredictMultitypeSpatialPlusPars, ltar, autocorr, paraautocorr, traceplots, parssummary, textsummary, priorpost, postcov, exceedProbs, betavals, etavals

---

lgcpPredictSpatioTemporalPlusPars

**lgcpPredictSpatioTemporalPlusPars function**

**Description**

A function to deliver fully Bayesian inference for the spatiotemporal log-Gaussian Cox process.

**Usage**

```r
lgcpPredictSpatioTemporalPlusPars(
  formula,  
  xyt,  
  T,  
  laglength,  
  ZmatList = NULL,  
  model.priors,  
  model.inits = lgcpInits(),  
  spatial.covmodel,  
  cellwidth = NULL,  
  poisson.offset = NULL,  
  mcmc.control,  
  output.control = setoutput(),  
  gradtrunc = Inf,  
  ext = 2,  
  inclusion = "touching"  
)
```
Arguments

- **formula**: A formula object of the form \( X \sim var1 + var2 \) etc. The name of the dependent variable must be "X". Only accepts 'simple' formulae, such as the example given.

- **xyt**: An object of class stppp

- **T**: The time point of interest

- **laglength**: The number of previous time points to include in the analysis

- **ZmatList**: A list of design matrices \( Z \) constructed with getZmat and possibly addTemporalCovariates see the details below and Bayesian_lgcp vignette for details on how to construct this.

- **model.priors**: Model priors, set using lgcpPrior

- **model.inits**: Model initial values. The default is NULL, in which case lgcp will use the prior mean to initialise \( \eta \) and \( \beta \) will be initialised from an oversispersed glm fit to the data. Otherwise use lgcpInits to specify.

- **spatial.covmodel**: Choice of spatial covariance function. See ?CovFunction

- **cellwidth**: The width of computational cells

- **poisson.offset**: A list of SpatialAtRisk objects (of length the number of types) defining \( \lambda_k \) (see below)

- **mcmc.control**: MCMC parameters, see ?mcmcpars

- **output.control**: Output choice, see ?setoutput

- **gradtrunc**: Truncation for gradient vector equal to \( H \) parameter Moller et al 1998 pp 473. Default is Inf, which means no gradient truncation, which seems to work in most settings.

- **ext**: Integer multiple by which grid should be extended, default is 2. Generally this will not need to be altered, but if the spatial correlation decays slowly, increasing 'ext' may be necessary.

- **inclusion**: Criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former, the default, includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.

Details

See the vignette "Bayesian_lgcp" for examples of this code in use.

The model for the data is as follows:

\[
X(s) \sim \text{Poisson}[R(s,t)]
\]

\[
R(s) = C_A \lambda(s,t) \exp[Z(s,t)\beta + Y(s,t)]
\]
Here $X(s,t)$ is the number of events in the cell of the computational grid containing $s$, $R(s,t)$ is the Poisson rate, $C_A$ is the cell area, $\lambda(s,t)$ is a known offset, $Z(s,t)$ is a vector of measured covariates and $Y(s,t)$ is the latent Gaussian process on the computational grid. The other parameters in the model are $\beta$, the covariate effects; and $\eta=\{\log(\sigma), \log(\phi), \log(\theta)\}$, the parameters of the process $Y$ on an appropriately transformed (in this case log) scale.

We recommend the user takes the following steps before running this method:

1. Compute approximate values of the parameters, $\eta$, of the process $Y$ using the function minimum.contrast. These approximate values are used for two main reasons: (1) to help inform the size of the computational grid, since we will need to use a cell width that enables us to capture the dependence properties of $Y$ and (2) to help inform the proposal kernel for the MCMC algorithm.

2. Choose an appropriate grid on which to perform inference using the function chooseCellWidth; this will partly be determined by the results of the first stage and partly by the available computational resource available to perform inference.

3. Using the function getpolyol, construct the computational grid and polygon overlays, as required. As this can be an expensive step, we recommend that the user saves this object after it has been constructed and in future reference to the data, reloads this object, rather than having to re-compute it (provided the computational grid has not changed).

4. Decide on which covariates are to play a part in the analysis and use the lgcp function getZmat to interpolate these onto the computational grid. Note that having saved the results from the previous step, this is a relatively quick operation, and allows the user to quickly construct different design matrices, $Z$, from different candidate models for the data.

5. If required, set up the population offset using SpatialAtRisk functions (see the vignette "Bayesian_lgcp"); specify the priors using lgcpPrior; and if desired, the initial values for the MCMC, using the function lgcpInits.

6. Run the MCMC algorithm and save the output to disk. We recommend dumping information to disk using the dump2dir function in the output.control argument because it offers much greater flexibility in terms of MCMC diagnosis and post-processing.

7. Perform post-processing analyses including MCMC diagnostic checks and produce summaries of the posterior expectations we require for presentation. (see the vignette "Bayesian_lgcp" for further details). Functions of use in this step include traceplots, autocorr, parautocorr, ltar, parsummary, priorpost, postcov, textsummary, expectation, exceedProbs and lgcp:::expectation.lgcpPredict.

The user must provide a list of design matrices to use this function. In the interpolation step above, there are three cases to consider:

1. where $Z(s,t)$ cannot be decomposed, i.e., $Z$ are true spatiotemporal covariates. In this case, each element of the list must be constructed separately using the function getZmat on the covariates for each time point.

2. $Z(s,t)\beta = Z_1(s)\beta_1 + Z_2(t)\beta_2$: the spatial and temporal effects are separable; in this case use the function addTemporalCovariates, to aid in the construction of the list.

3. $Z(s,t)\beta = Z(s)\beta$, in which case the user only needs to perform the interpolation using getZmat once, each of the elements of the list will then be identical.
4. \( Z(s,t)\beta = Z(t)\beta \) in this case we follow the procedure for the separable case above. For example, if \( \text{dotw} \) is a temporal covariate we would use formula \(< X \sim \text{dotw} \) for the main algorithm, formula.spatial \(< X \sim 1 \) to interpolate the spatial covariates using getZmat, followed by temporal.formula \(< t \sim \text{dotw} - 1 \) using addTemporalCovariates to construct the list of design matrices, \( Z\text{mat} \).

Value

an object of class \( \text{lgcpPredictSpatioTemporalPlusParameters} \)

References


See Also

\( \text{linkchooseCellWidth}, \text{getpolyol}, \text{guessinterp}, \text{getZmat}, \text{addTemporalCovariates}, \text{lgcpPrior}, \text{lgcpInits}, \text{CovFunction} \) \( \text{lgcpPredictSpatialPlusPars}, \text{lgcpPredictAggregateSpatialPlusPars}, \text{lgcpPredictMulti-typeSpatialPlusPars}, \text{ltar}, \text{autocorr}, \text{paraautocorr}, \text{traceplots}, \text{parsummary}, \text{textsummary}, \text{priorpost}, \text{postcov}, \text{exceedProbs}, \text{betavals}, \text{etavals} \)

---

\( \text{lgcpPrior} \) \( \text{lgcpPrior function} \)

Description

A function to create the prior for beta and eta ready for a run of the MCMC algorithm.

Usage

\( \text{lgcpPrior(etaprior = NULL, betaprior = NULL)} \)
Arguments

etaprior an object of class PriorSpec defining the prior for the parameters of the latent field, eta. See ?PriorSpec.list.

betaprior etaprior an object of class PriorSpec defining the prior for the parameters of main effects, beta. See ?PriorSpec.list.

Value

an R structure representing the prior density ready for a run of the MCMC algorithm.

See Also


Examples

lgcpPrior(etaprior=PriorSpec(LogGaussianPrior(mean=log(c(1,500)), variance=diag(0.15,2))),betaprior=PriorSpec(GaussianPrior(mean=rep(0,9), variance=diag(10^6,9))))

Section: lgcpSim

Description

Approximate simulation from a spatiotemporal log-Gaussian Cox Process. Returns an stppp object.

Usage

lgcpSim(
  owin = NULL,
  tlim = as.integer(c(0, 10)),
  spatial.intensity = NULL,
  temporal.intensity = NULL,
  cellwidth = 0.05,
  model.parameters = lgcppars(sigma = 2, phi = 0.2, theta = 1),
  spatial.covmodel = "exponential",
  covpars = c(),
  returnintensities = FALSE,
  progressbar = TRUE,
  ext = 2,
  plot = FALSE,
  ratepow = 0.25,
)
sleeptime = 0,
inclusion = "touching"
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>owin</td>
<td>polygonal observation window</td>
</tr>
<tr>
<td>tlim</td>
<td>time interval on which to simulate data</td>
</tr>
<tr>
<td>spatial.intensity</td>
<td>object that can be coerced into a spatialAtRisk object. if NULL then uniform</td>
</tr>
<tr>
<td>temporal.intensity</td>
<td>the fixed temporal component: either a numeric vector, or a function that can</td>
</tr>
<tr>
<td>cellwidth</td>
<td>width of cells in same units as observation window</td>
</tr>
<tr>
<td>model.parameters</td>
<td>parameters of model, see ?lgcppars.</td>
</tr>
<tr>
<td>spatial.covmodel</td>
<td>spatial covariance function, default is exponential, see ?CovarianceFct</td>
</tr>
<tr>
<td>covpars</td>
<td>vector of additional parameters for spatial covariance function, in order they</td>
</tr>
<tr>
<td>returnintensities</td>
<td>logigal, whether to return the spatial intensities and true field Y at each</td>
</tr>
<tr>
<td>progressbar</td>
<td>logical, whether to print a progress bar. Default TRUE.</td>
</tr>
<tr>
<td>ext</td>
<td>how much to extend the parameter space by. Default is 2.</td>
</tr>
<tr>
<td>plot</td>
<td>logical, whether to plot intensities.</td>
</tr>
<tr>
<td>ratepow</td>
<td>power that intensity is raised to for plotting purposes (makes the plot more</td>
</tr>
<tr>
<td>sleeptime</td>
<td>time in seconds to sleep between plots</td>
</tr>
<tr>
<td>inclusion</td>
<td>criterion for cells being included into observation window. Either ‘touching’</td>
</tr>
</tbody>
</table>

Details

The following is a mathematical description of a log-Gaussian Cox Process, it is best viewed in the pdf version of the manual.

Let $Y(s, t)$ be a spatiotemporal Gaussian process, $W \subseteq R^2$ be an observation window in space and $T \subseteq R_{\geq 0}$ be an interval of time of interest. Cases occur at spatio-temporal positions $(x, t) \in W \times T$ according to an inhomogeneous spatio-temporal Cox process, i.e. a Poisson process with a stochastic intensity $R(x, t)$. The number of cases, $X_{S,[t_1,t_2]}$, arising in any $S \subseteq W$ during the interval $[t_1, t_2] \subseteq T$ is then Poisson distributed conditional on $R(\cdot)$,

$$X_{S,[t_1,t_2]} \sim \text{Poisson} \left\{ \int_{S} \int_{t_1}^{t_2} R(s, t)dsdt \right\}$$
Following Brix and Diggle (2001) and Diggle et al (2005), the intensity is decomposed multiplicatively as

\[ R(s, t) = \lambda(s)\mu(t)\exp\{\mathcal{Y}(s, t)\}. \]

In the above, the fixed spatial component, \( \lambda : R^2 \mapsto R_{\geq 0} \), is a known function, proportional to the population at risk at each point in space and scaled so that

\[ \int_W \lambda(s) ds = 1, \]

whilst the fixed temporal component, \( \mu : R_{\geq 0} \mapsto R_{\geq 0} \), is also a known function with

\[ \mu(t)\delta t = E[X_{W,\delta t}], \]

for \( t \) in a small interval of time, \( \delta t \), over which the rate of the process over \( W \) can be considered constant.

Value

an stppp object containing the data

References


See Also

lgcpPredict, showGrid.stppp, stppp

Examples

## Not run: library(spatstat.core); library(spatstat.utils); xyt <- lgcpSim()
lgcpSimMultitypeSpatialCovariates

lgcpSimMultitypeSpatialCovariates function

Description

A function to Simulate multivariate point process models

Usage

lgcpSimMultitypeSpatialCovariates(
  formulaList,
  owin,
  regionalcovariates,
  pixelcovariates,
  betaList,
  spatial.offsetList = NULL,
  cellwidth,
  model.parameters,
  spatial.covmodel = "exponential",
  covpars = c(),
  ext = 2,
  plot = FALSE,
  inclusion = "touching"
)

Arguments

formulaList  a list of formulae objects
owin         a spatstat owin object on which to simulate the data
regionalcovariates  a SpatialPolygonsDataFrame object
pixelcovariates   a SpatialPixelsDataFrame object
betaList        list of beta parameters
spatial.offsetList  list of poisson offsets
cellwidth       cellwidth
model.parameters model parameters, a list eg list(sigma=1,phi=0.2)
spatial.covmodel the choice of spatial covariance model, can be anything from the RandomFields covariance function, CovarianceFct.
covpars          additional covariance parameters, for the chosen model, optional.
ext              number of times to extend the simulation window
whether to plot the results automatically

criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former, the default, includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.

Value

a marked ppp object, the simulated data

Description

A function to simulate from a log gaussian process

Usage

lgcpSimSpatial(
  owin = NULL,
  spatial.intensity = NULL,
  expectednumcases = 100,
  cellwidth = 0.05,
  model.parameters = lgcppars(sigma = 2, phi = 0.2),
  spatial.covmodel = "exponential",
  covpars = c(),
  ext = 2,
  plot = FALSE,
  inclusion = "touching"
)

Arguments

  owin observation window
  spatial.intensity an object that can be coerced to one of class spatialAtRisk
  expectednumcases the expected number of cases
  cellwidth width of cells in same units as observation window
  model.parameters parameters of model, see ?lgcppars. Only set sigma and phi for spatial model.
  spatial.covmodel spatial covariance function, default is exponential, see ?CovarianceFct
  covpars vector of additional parameters for spatial covariance function, in order they appear in chosen model in ?CovarianceFct


\textit{lgcpSimSpatialCovariates} function

\textbf{Description}

A function to simulate a spatial LGCP.

\textbf{Usage}

\begin{verbatim}
lgcpSimSpatialCovariates(
    formula,  
    owin,       
    regionalcovariates = NULL, 
    pixelcovariates = NULL, 
    Zmat = NULL, 
    beta,       
    poisson.offset = NULL, 
    cellwidth,  
    model.parameters, 
    spatial.covmodel = "exponential", 
    covpars = c(), 
    ext = 2,       
    plot = FALSE, 
    inclusion = "touching"
)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
\item \texttt{formula} \hspace{1cm} a formula of the form $X \sim \text{var1} + \text{var2}$ etc.
\item \texttt{owin} \hspace{1cm} the observation window on which to do the simulation
\item \texttt{regionalcovariates} \hspace{1cm} an optional object of class \texttt{SpatialPolygonsDataFrame} containing covariates
\item \texttt{pixelcovariates} \hspace{1cm} an optional object of class \texttt{SpatialPixelsDataFrame} containing covariates

\item \texttt{Zmat} \hspace{1cm} an optional matrix containing covariates
\item \texttt{beta} \hspace{1cm} a vector containing the parameter values
\item \texttt{poisson.offset} \hspace{1cm} a vector containing the offset values
\item \texttt{cellwidth} \hspace{1cm} the cell width for the grid
\item \texttt{model.parameters} \hspace{1cm} a list containing the model parameters
\item \texttt{spatial.covmodel} \hspace{1cm} the spatial covariance model
\item \texttt{covpars} \hspace{1cm} a vector containing the covariate parameters
\item \texttt{ext} \hspace{1cm} how much to extend the parameter space by. Default is 2.
\item \texttt{plot} \hspace{1cm} logical, whether to plot the latent field.
\item \texttt{inclusion} \hspace{1cm} criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.
\end{itemize}

\textbf{Value}

a \texttt{ppp} object containing the data
Zmat

optional design matrix, if the polygon/polygon overlays have already been computed

beta

the parameters, beta for the model

poisson.offset

the poisson offset, created using a SpatialAtRisk.fromXYZ class of objects

cellwidth

the width of cells on which to do the simulation

model.parameters

the parameters of the model eg list(sigma=1,phi=0.2)

spatial.covmodel

the choice of spatial covariance model, can be anything from the RandomFields covariance function, CovarianceFct.

covpars

additional covariance parameters, for the chosen model, optional.

ext

the amount by which to extend the observation grid in each direction, default is 2

plot

whether to plot the resulting data

inclusion

criterion for cells being included into observation window. Either 'touching' or 'centroid'. The former, the default, includes all cells that touch the observation window, the latter includes all cells whose centroids are inside the observation window.

Value

a ppp object containing the simulated data

Description

Display the introductory vignette for the lgcp package.

Usage

lgcpvignette()

Value

displays the vignette by calling browseURL
loc2poly function

Description

Converts a polygon selected via the mouse in a graphics window into an polygonal owin object. (Make sure the x and y scales are correct!) Points must be selected traversing the required window in one direction (ie either clockwise, or anticlockwise), points must not be overlapping. Select the sequence of edges via left mouse button clicks and store the polygon with a right click.

Usage

loc2poly(n = 512, type = "l", col = "black", ...)

Arguments

n
  the maximum number of points to locate

type
  same as argument type in function locator. see ?locator. Default draws lines

col
  colour of lines/points

...
  other arguments to pass to locate

Value

a polygonal owin object

See Also

lgcpPredict, identify.lgcpPredict

Examples

## Not run: plot(lg) # lg an lgcpPredict object
## Not run: subwin <- loc2poly()

LogGaussianPrior function

Description

A function to create a Gaussian prior on the log scale

Usage

LogGaussianPrior(mean, variance)
Arguments

- **mean**: a vector of length 2 representing the mean (on the log scale)
- **variance**: a 2x2 matrix representing the variance (on the log scale)

Value

- an object of class LogGaussianPrior that can be passed to the function PriorSpec.

See Also

- GaussianPrior, linkPriorSpec.list

Examples

```r
## Not run: LogGaussianPrior(mean=log(c(1,500)), variance=diag(0.15,2))
```

### loop.mcmc

**loop over an iterator**

**Description**

useful for testing progress bars

**Usage**

```r
loop.mcmc(object, sleep = 1)
```

**Arguments**

- **object**: an mcmc iterator
- **sleep**: pause between iterations in seconds

### ltar

**ltar function**

**Description**

A function to return the sampled log-target from a call to the function lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatioTemporalPlusPars or lgcpPredictMultitype-SpatialPlusPars. This is used as a convergence diagnostic.

**Usage**

```r
ltar(lg)
```
**MALAlgcp**

**Arguments**

- `lg`: an object produced by a call to `lgcpPredictSpatialPlusPars`, `lgcpPredictAggregateSpatialPlusPars`, `lgcpPredictSpatioTemporalPlusPars` or `lgcpPredictMultitypeSpatialPlusPars`

**Value**

- the log-target from each saved iteration of the MCMC chain.

**See Also**

- `autocorr`, `parautocorr`, `traceplots`, `parssummary`, `textsummary`, `priorpost`, `postcov`, `exceedProbs`, `betavals`, `etavals`

---

**MALAlgcp**  
**MALAlgcp function**

**Description**

ADVANCED USE ONLY A function to perform MALA for the spatial only case

**Usage**

```r
MALAlgcp(
  mcmcloop,
  inits,
  adaptivescheme,
  M,
  N,
  Mext,
  Next,
  sigma,
  phi,
  theta,
  mu,
  nis,
  cellarea,
  spatialvals,
  temporal.fitted,
  tdiff,
  scaleconst,
  rootQeigs,
  invrootQeigs,
  cellInside,
  MCMCdiag,
  gradtrunc,
  gridfun,
)```
Arguments

mcmcloop an mcmcLoop object
inits initial values from mcmc.control
adaptescheme adaptive scheme from mcmc.control
M number of cells in x direction on output grid
N number of cells in y direction on output grid
Mext number of cells in x direction on extended output grid
Nex number of cells in y direction on extended output grid
sigma spatial covariance parameter sigma
phi spatial covariance parameter phi
theta temporal correlation parameter theta
mu spatial covariance parameter mu
nis cell counts matrix
cellarea area of cells
spatialvals spatial at risk, function lambda, interpolated onto the requisite grid
temporal.fitted temporal fitted values representing mu(t)
tdiff vecto of time differences with convention that the first element is Inf
scaleconst expected number of observations
rootQeigs square root of eigenvalues of precision matrix
invrotQeigs inverse square root of eigenvalues of precision matrix
cellInside logical matrix dictating whether cells are inside the observation window
MCMCdiag defunct
gradtrunc gradient truncation parameter
gridfun grid functions
gridav grid average functions
mcens x-coordinates of cell centroids
ncens y-coordinates of cell centroids
aggtimes z-coordinates of cell centroids (ie time)

Value

object passed back to lgcpPredictSpatial
MALAlgcpAggregateSpatial.PlusPars

MALAlgcpAggregateSpatial.PlusPars function

Description

A function to run the MCMC algorithm for aggregated spatial point process data. Not for general purpose use.

Usage

MALAlgcpAggregateSpatial.PlusPars(
  mcmcloop,
  inits,
  adaptivescheme,
  M,
  N,
  Mext,
  Next,
  mcens,
  ncens,
  formula,
  Zmat,
  model.priors,
  model.inits,
  fftgrid,
  spatial.covmodel,
  nis,
  cellarea,
  spatialvals,
  cellInside,
  MCMCdiag,
  gradtrunc,
  gridfun,
  gridav,
  d,
  spdf,
  ol,
  Nfreq
)

Arguments

mcmcloop details of the mcmc loop
inits initial values
adaptivescheme the adaptive MCMC scheme
MALAlgcpMultitypeSpatial.PlusPars

Description

A function to run the MCMC algorithm for multivariate spatial point process data. Not for general purpose use.

Value

output from the MCMC run
Usage

MALAlgcpMultitypeSpatial.PlusPars(
    mcmcloop,
    inits,
    adaptivescheme,
    M,
    N,
    Mext,
    Next,
    mcens,
    ncens,
    formulaList,
    zml,
    Zmat,
    model.priorsList,
    model.initsList,
    fftgrid,
    spatial.covmodelList,
    nis,
    cellarea,
    spatialvals,
    cellInside,
    MCMCdiag,
    gradtrunc,
    gridfun,
    gridav,
    marks,
    ntypes,
    d
)

Arguments

mcmcloop       details of the mcmc loop
inits          initial values
adaptivescheme the adaptive MCMC scheme
M               number of grid cells in x direction
N               number of grid cells in y direction
Mext           number of extended grid cells in x direction
Next            number of extended grid cells in y direction
mcens          centroids in x direction
ncens          centroids in y direction
formulaList    a list of formula objects of the form X ~ var1 + var2 etc.
zml             list of design matrices
Zmat            a design matrix constructed using getZmat
MALAlgcpSpatial

model.priorsList
  list of model priors, see lgcpPriors

model.initsList
  list of model initial values, see lgcpInits

fftgrid
  an objects of class FFTgrid, see genFFTgrid

spatial.covmodelList
  list of spatial covariance models constructed using CovFunction

nis
  cell counts on the etended grid

cellarea
  the cell area

spatialvals
  interpolated poisson offset on fft grid

cellInside
  0-1 matrix indicating inclusion in the observation window

MCMCdiag
  not used

gradtrunc
  gradient truncation parameter

gridfun
  used to specify other actions to be taken, e.g. dumping MCMC output to disk.

gridav
  used for computing Monte Carlo expectations online

marks
  the marks from the marked ppp object

ntypes
  the number of types being analysed

d
  matrix of toral distances

Value

output from the MCMC run

MALAlgcpSpatial

MALAlgcpSpatial function

Description

ADVANCED USE ONLY A function to perform MALA for the spatial only case

Usage

MALAlgcpSpatial(
mcmcloop,
inits,
adaptescheme,
M,
N,
Mext,
Next,
sigma,
phi,
mu,
MALAlgcpSpatial

nis,
cellarea,
spatialvals,
scaleconst,
rootQeigs,
invrootQeigs,
cellInside,
MCMCdiag,
gradtrunc,
gridfun,
gridav,
mcens,
ncens
)

Arguments

mcmcloop an mcmcLoop object
inits initial values from mcmc.control
adaptivescheme adaptive scheme from mcmc.control
M number of cells in x direction on output grid
N number of cells in y direction on output grid
Mext number of cells in x direction on extended output grid
Next number of cells in y direction on extended output grid
sigma spatial covariance parameter sigma
phi spatial covariance parameter phi
mu spatial covariance parameter mu
nis cell counts matrix
cellarea area of cells
spatialvals spatial at risk, function lambda, interpolated onto the requisite grid
scaleconst expected number of observations
rootQeigs square root of eigenvalues of precision matrix
invrootQeigs inverse square root of eigenvalues of precision matrix
cellInside logical matrix dictating whether cells are inside the observation window
MCMCdiag defunct
gradtrunc gradient truncation parameter
gridfun grid functions
gridav grid average functions
mcens x-coordinates of cell centroids
ncens y-coordinates of cell centroids

Value

object passed back to lgcpPredictSpatial
MALAlgcpSpatial.PlusPars

**MALAlgcpSpatial.PlusPars function**

**Description**

A function to run the MCMC algorithm for spatial point process data. Not for general purpose use.

**Usage**

```r
MALAlgcpSpatial.PlusPars(
  mcmcloop,
  inits,
  adaptivescheme,
  M,
  N,
  Mext,
  Next,
  mcens,
  ncens,
  formula,
  Zmat,
  model.priors,
  model.inits,
  fftgrid,
  spatial.covmodel,
  nis,
  cellarea,
  spatialvals,
  cellInside,
  MCMCdiag,
  gradtrunc,
  gridfun,
  gridav,
  d
)
```

**Arguments**

- `mcmcloop`: details of the mcmc loop
- `inits`: initial values
- `adaptivescheme`: the adaptive MCMC scheme
- `M`: number of grid cells in x direction
- `N`: number of grid cells in y direction
- `Mext`: number of extended grid cells in x direction
MALAlgcpSpatioTemporal.PlusPars

Description

A function to run the MCMC algorithm for spatiotemporal point process data. Not for general purpose use.

Usage

MALAlgcpSpatioTemporal.PlusPars(
mcmclloop,  
inits,  
adaptivescheme,  
M,  
N,  
Mext,
Next, mcens, ncens, formula, ZmatList, model.priors, model.inits, fftgrid, spatial.covmodel, nis, tdiff, cellarea, spatialvals, cellInside, MCMCdiag, gradtrunc, gridfun, gridav, d, aggtimes, spatialOnlyCovariates
)

Arguments

mcmcloop details of the mcmc loop
inits initial values
adaptivescheme the adaptive MCMC scheme
M number of grid cells in x direction
N number of grid cells in y direction
Next number of extended grid cells in x direction
mcens centroids in x direction
ncens centroids in y direction
formula a formula object of the form X ~ var1 + var2 etc.
ZmatList list of design matrices constructed using getZmat
model.priors model priors, constructed using lgcpPrior
model.inits initial values for the MCMC
fftgrid an objects of class FFTgrid, see genFFTgrid
spatial.covmodel spatial covariance model, constructed with CovFunction
nis cell counts on the etended grid
tdiff vector of time differences
cellarea the cell area
**matchcovariance**

spatialvals  interpolated poisson offset on fft grid

cellInside  0-1 matrix indicating inclusion in the observation window

MCMCdiag  not used

gradtrunc  gradient truncation parameter

gridfun  used to specify other actions to be taken, e.g. dumping MCMC output to disk.

gridav  used for computing Monte Carlo expectations online

d  matrix of toral distances

aggtimes  the aggregate times

spatialOnlyCovariates  whether this is a 'spatial' only problem

**Value**

output from the MCMC run

**Description**

A function to match the covariance matrix of a Gaussian Field with an approximate GMRF with neighbourhood size ns.

**Usage**

```r
matchcovariance(
exg,
yg,
ns,
sigma,
phi,
model,
additionalparameters,
verbose = TRUE,
r = 1,
method = "Nelder-Mead"
)
```

**Arguments**

- `xg`  x grid must be equally spaced
- `yg`  y grid must be equally spaced
- `ns`  neighbourhood size
- `sigma`  spatial variability parameter
Description
A function to declare and also evaluate an Matern 1.5 covariance function.

Usage
maternCovFct15(d, CovParameters)

Arguments
d          toral distance
CovParameters  parameters of the latent field, an object of class "CovParamaters".

Value
the exponential covariance function

Author(s)
Dominic Schumacher

See Also
CovFunction.function, RandomFieldsCovFct, SpikedExponentialCovFct
**Description**

A function to declare and also evaluate an Matern 2.5 covariance function.

**Usage**

```r
maternCovFct25(d, CovParameters)
```

**Arguments**

- `d`: total distance
- `CovParameters`: parameters of the latent field, an object of class "CovParamaters".

**Value**

the exponential covariance function

**Author(s)**

Dominic Schumacher

**See Also**

CovFunction.function, RandomFieldsCovFct, SpikedExponentialCovFct

---

**Description**

control an MCMC loop with this iterator

**Usage**

```r
mcmcLoop(N, burnin, thin, trim = TRUE, progressor = mcmcProgressPrint)
```

**Arguments**

- `N`: number of iterations
- `burnin`: length of burn-in
- `thin`: frequency of thinning
- `trim`: whether to cut off iterations after the last retained iteration
- `progressor`: a function that returns a progress object
mcmcpars  

**mcmcpars function**

**Description**

A function for setting MCMC options in a run of `lgcpPredict` for example.

**Usage**

```r
mcmcpars(mala.length, burnin, retain, inits = NULL, adaptivescheme)
```

**Arguments**

- `mala.length`: default = 100,
- `burnin`: default = floor(mala.length/2),
- `retain`: thinning parameter eg operated on chain every 'retain' iteration (eg store output or compute some posterior functional)
- `inits`: optional initial values for MCMC
- `adaptivescheme`: the type of adaptive mcmc to use, see `?constanth` (constant h) or `?andrieuthomsh` (adaptive MCMC of Andrieu and Thoms (2008))

**Value**

mcmc parameters

**See Also**

- `lgcpPredict`

---

mcmcProgressNone  

**null progress monitor**

**Description**

a progress monitor that does nothing

**Usage**

```r
mcmcProgressNone(mcmcloop)
```

**Arguments**

- `mcmcloop`: an mcmc loop iterator

**Value**

a progress monitor
mcmcProgressPrint

printing progress monitor

Description
a progress monitor that prints each iteration

Usage
mcmcProgressPrint(mcmcloop)

Arguments
mcmcloop an mcmc loop iterator

Value
a progress monitor

mcmcProgressTextBar
text bar progress monitor

Description
a progress monitor that uses a text progress bar

Usage
mcmcProgressTextBar(mcmcloop)

Arguments
mcmcloop an mcmc loop iterator

Value
a progress monitor
**mcmcProgressTk**  
*graphical progress monitor*

**Description**

A progress monitor that uses tcltk dialogs

**Usage**

```r
mcmcProgressTk(mcmcloop)
```

**Arguments**

- `mcmcloop`  
an mcmc loop iterator

**Value**

A progress monitor

---

**mcmctrace**  
*mcmctrace function*

**Description**

Generic function to extract the information required to produce MCMC trace plots.

**Usage**

```r
mcmctrace(obj, ...)
```

**Arguments**

- `obj`  
an object
- `...`  
additional arguments

**Value**

Method `mcmctrace`
mcmctrace.lgcpPredict  mcmctrace.lgcpPredict function

Description
If MCMCdiag was positive when lgcpPredict was called, then this retrieves information from the chains stored.

Usage
## S3 method for class 'lgcpPredict'
mcmctrace(obj, ...)

Arguments
  obj        an object of class lgcpPredict
  ...        additional arguments

Value
returns the saved MCMC chains in an object of class mcmcdiag.

See Also
lgcpPredict, plot.mcmcdiag

meanfield  meanfield function

Description
Generic function to extract the mean of the latent field Y.

Usage
meanfield(obj, ...)

Arguments
  obj        an object
  ...        additional arguments

Value
method meanfield
Description

This is an accessor function for objects of class `lgcpPredict` and returns the mean of the field $Y$ as an `lgcpgrid` object.

Usage

```r
## S3 method for class 'lgcpPredict'
meanfield(obj, ...)
```

Arguments

- `obj`: an object of class `lgcpPredict`
- `...`: additional arguments

Value

returns the cell-wise mean of $Y$ computed via Monte Carlo.

See Also

`lgcpPredict`, `lgcpgrid`
MonteCarloAverage

MonteCarloAverage function

Description

This function creates an object of class MonteCarloAverage. The purpose of the function is to compute Monte Carlo expectations online in the function lgcpPredict, it is set in the argument gridmeans of the argument output.control.

Usage

MonteCarloAverage(funlist, lastonly = TRUE)

Arguments

funlist a character vector of names of functions, each accepting single argument Y
lastonly compute average using only time T? (see ?lgcpPredict for definition of T)

Details

A Monte Carlo Average is computed as:

\[ E_\pi(Y_{t_1:t_2}|X_{t_1:t_2})[g(Y_{t_1:t_2})] \approx \frac{1}{n} \sum_{i=1}^{n} g(Y_{t_1:t_2}^{(i)}) \]

where \( g \) is a function of interest, \( Y_{t_1:t_2}^{(i)} \) is the \( i \)th retained sample from the target and \( n \) is the total number of retained iterations. For example, to compute the mean of \( Y_{t_1:t_2} \) set,

\[ g(Y_{t_1:t_2}) = Y_{t_1:t_2}, \]

the output from such a Monte Carlo average would be a set of \( t_2 - t_1 \) grids, each cell of which being equal to the mean over all retained iterations of the algorithm (NOTE: this is just an example computation, in practice, there is no need to compute the mean on line explicitly, as this is already done by default in lgcpPredict). For further examples, see below. The option last=TRUE computes,

\[ E_\pi(Y_{t_1:t_2}|X_{t_1:t_2})[g(Y_{t_2})], \]

so in this case the expectation over the last time point only is computed. This can save computation time.

Value

object of class MonteCarloAverage

See Also

setoutput, lgcpPredict, GAinitialise, GAupdate, GAfinalise, GAreturnvalue, exceedProbs
Examples

fun1 <- function(x){return(x)} # gives the mean
fun2 <- function(x){return(x^2)} # computes E(X^2). Can be used with the
# mean to compute variances, since
# Var(X) = E(X^2) - E(X)^2
fun3 <- exceedProbs(c(1.5,2,3)) # exceedance probabilities,
# see ?exceedProbs
mca <- MonteCarloAverage(c("fun1","fun2","fun3"))
mca2 <- MonteCarloAverage(c("fun1","fun2","fun3"),lastonly=TRUE)

mstppp

mstppp function

Description

Generic function used in the construction of marked space-time planar point patterns. An mstppp object is like an stppp object, but with an extra component containing a data frame (the mark information).

Usage

mstppp(P, ...)

Arguments

P
an object

... additional arguments

Details

Observations are assumed to occur in the plane and the observation window is assumed not to change over time.

Value

method mstppp

See Also

mstppp, mstppp.ppp, mstppp.list
mstppp.list

**mstppp.list function**

**Description**

Construct a marked space-time planar point pattern from a list object

**Usage**

```r
## S3 method for class 'list'
mstppp(P, ...)
```

**Arguments**

- `P` list object containing $xyt$, an $(n \times 3)$ matrix corresponding to (x,y,t) values; $tlim$, a vector of length 2 giving the observation time window; $window$ giving an owin spatial observation window; see ?owin for more details; and $data$, a data frame containing the collection of marks
- `...` additional arguments

**Value**

an object of class mstppp

**See Also**

mstppp, mstppp.ppp.

mstppp.ppp

**mstppp.ppp function**

**Description**

Construct a marked space-time planar point pattern from a ppp object

**Usage**

```r
## S3 method for class 'ppp'
mstppp(P, t, tlim, data, ...)
```

**Arguments**

- `P` a spatstat ppp object
- `t` a vector of length `n`
- `tlim` a vector of length 2 specifying the observation time window
- `data` a data frame containing the collection of marks
- `...` additional arguments
Value

an object of class mstppp

See Also

mstppp, mstppp.list

---

**mstppp.stppp**

*mstppp.stppp function*

---

**Description**

Construct a marked space-time planar point pattern from an stppp object

**Usage**

```r
## S3 method for class 'stppp'
mstppp(P, data, ...)
```

**Arguments**

- `P` an lgcp stppp object
- `data` a data frame containing the collection of marks
- `...` additional arguments

**Value**

an object of class mstppp

**See Also**

mstppp, mstppp.list

---

**muEst**

*muEst function*

---

**Description**

Computes a non-parametric estimate of \( \mu(t) \). For the purposes of performing prediction, the alternatives are: (1) use a parameteric model as in Diggle P, Rowlingson B, Sù T (2005), or (2) a constantInTime model.

**Usage**

```r
muEst(xyt, ...)
```
multiply.list

Arguments

- `xyt` an stppp object
- `...` additional arguments to be passed to lowess

Value

object of class temporalAtRisk giving the smoothed mut using the lowess function

References


See Also
temporalAtRisk, constantInTime, ginhomAverage, KinhomAverage, spatialparsEst, thetaEst, lambdaEst

multiply.list

Description

This function multiplies the elements of two list objects together and returns the result in another list object.

Usage

`multiply.list(list1, list2)`

Arguments

- `list1` a list of objects that could be summed using "+
- `list2` a list of objects that could be summed using "+

Value

a list with ith entry the sum of list1[i] and list2[i]
neattable  

Description
Function to print right-aligned tables to the console.

Usage
neattable(mat, indent = 0)

Arguments
mat  a numeric or character matrix object
indent  indent

Value
prints to screen with specified indent

Examples
mat <- rbind(c("one", "two", "three"), matrix(round(runif(9), 3), 3, 3))
neattable(mat)

neigh2D  

Description
A function to compute the neighbours of a cell on a toral grid

Usage
neigh2D(i, j, ns, M, N)

Arguments
i  cell index i
j  cell index j
ns  number of neighbours either side
M  size of grid in x direction
N  size of grid in y direction

Value
the cell indices of the neighbours
### nextStep

*next step of an MCMC chain*

**Description**

just a wrapper for nextElem really.

**Usage**

```r
callNextStep(object)
```

**Arguments**

- `object` an mcmc loop object

### nullAverage

*nullAverage function*

**Description**

A null scheme, that does not perform any computation in the running of `lgcpPredict`, it is the default value of `gridmeans` in the argument `output.control`.

**Usage**

```r
nullAverage()
```

**Value**

object of class nullAverage

**See Also**

`setoutput, lgcpPredict, GA initialise, GA update, GA finalise, GAreturnvalue`
nullFunction

nullFunction function

Description

This is a null function and performs no action.

Usage

nullFunction()

Value

object of class nullFunction

See Also

setoutput, GFinitialise, GFupdate, GFfinalise, GFreturnvalue

numCases

numCases function

Description

A function used in conjunction with the function "expectation" to compute the expected number of cases in each computational grid cell. Currently only implemented for spatial processes (lgcpPredictSpatialPlusPars and lgcpPredictAggregateSpatialPlusPars).

Usage

numCases(Y, beta, eta, Z, otherargs)

Arguments

Y the latent field
beta the main effects
eta the parameters of the latent field
Z the design matrix
otherargs other arguments to the function (see vignette "Bayesian_lgcp" for an explanation)

Value

the number of cases in each cell
osppp2latlon

See Also
expectation, lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars

Examples

## Not run: ex <- expectation(lg,numCases)[[1]] # lg is output from spatial LGCP MCMC

osppp2latlon  osppp2latlon function

Description
A function to transform a ppp object in the OSGB projection (epsg:27700) to a ppp object in the latitude/longitude (epsg:4326) projection.

Usage
osppp2latlon(obj)

Arguments
obj a ppp object in OSGB

Value
a ppp object in Lat/Lon

osppp2merc  osppp2merc function

Description
A function to transform a ppp object in the OSGB projection (epsg:27700) to a ppp object in the Mercator (epsg:3857) projection.

Usage
osppp2merc(obj)

Arguments
obj a ppp object in OSGB

Value
a ppp object in Mercator
**paramprec**

**paramprec function**

### Description

A function to compute the precision matrix of a GMRF on an $M \times N$ toral grid with neighbourhood size $ns$. Note that the precision matrix is block circulant. The returned function operates on a parameter vector as in Rue and Held (2005) pp 187.

### Usage

```r
paramprec(ns, M, N)
```

### Arguments

- **ns**: neighbourhood size
- **M**: number of cells in x direction
- **N**: number of cells in y direction

### Value

A function that returns the precision matrix given a parameter vector.

---

**paramprecbase**

**paramprecbase function**

### Description

A function to compute the parametrised base matrix of a precision matrix of a GMRF on an $M \times N$ toral grid with neighbourhood size $ns$. Note that the precision matrix is block circulant. The returned function operates on a parameter vector as in Rue and Held (2005) pp 187.

### Usage

```r
paramprecbase(ns, M, N, inverse = FALSE)
```

### Arguments

- **ns**: neighbourhood size
- **M**: number of x cells
- **N**: number of y cells
- **inverse**: whether or not to compute the base matrix of the inverse precision matrix (ie the covariance matrix). default is FALSE

### Value

A function that returns the base matrix of the precision matrix.
parautocorr

parautocorr function

Description
A function to produce autocorrelation plots for the parameters beta and eta from a call to the function `lgcpPredictSpatialPlusPars`, `lgcpPredictAggregateSpatialPlusPars`, `lgcpPredictSpatioTemporalPlusPars` or `lgcpPredictMultitypeSpatialPlusPars`.

Usage

```r
parautocorr(obj, xlab = "Lag", ylab = NULL, main = "", ask = TRUE, ...)
```

Arguments

- `xlab`: optional label for x-axis, there is a sensible default.
- `ylab`: optional label for y-axis, there is a sensible default.
- `main`: optional title of the plot, there is a sensible default.
- `ask`: the parameter "ask", see `?par`.
- `...`: other arguments passed to the function "hist".

Value
produces autocorrelation plots of the parameters beta and eta.

See Also
- `ltar`, `autocorr`, `traceplots`, `parsummary`, `textsummary`, `priorpost`, `postcov`, `exceedProbs`, `betaval`, `etaval`

parsummary

parsummary function

Description
A function to produce a summary table for the parameters beta and eta from a call to the function `lgcpPredictSpatialPlusPars`, `lgcpPredictAggregateSpatialPlusPars`, `lgcpPredictSpatioTemporalPlusPars` or `lgcpPredictMultitypeSpatialPlusPars`.

Usage

```r
parsummary(obj, expon = TRUE, LaTeX = FALSE, ...)
```
plot.fromSPDF

Arguments

- **obj**: an object produced by a call to `lgcpPredictSpatialPlusPars`, `lgcpPredictAggregateSpatialPlusPars`, `lgcpPredictSpatioTemporalPlusPars` or `lgcpPredictMultitypeSpatialPlusPars`
- **expon**: whether to exponentiate the results, so that the parameters beta have the interpretation of "relative risk per unit increase in the covariate" default is TRUE
- **LaTeX**: whether to print parameter names using LaTeX symbols (if the table is later to be exported to a LaTeX document)
- ... other arguments

Value

a data frame containing the median, 0.025 and 0.975 quantiles.

See Also

- `ltar`, `autocorr`, `parautocorr`, `traceplots`, `textsummary`, `priorpost`, `postcov`, `exceedProbs`, `betavals`, `etavals`

---

plot.fromSPDF

**plot.fromSPDF function**

---

Description

Plot method for objects of class `fromSPDF`.

Usage

```r
## S3 method for class 'fromSPDF'
plot(x, ...)
```

Arguments

- **x**: an object of class `spatialAtRisk`
- ... additional arguments

Value

prints the object
### plot.fromXYZ function

**Description**
Plot method for objects of class fromXYZ.

**Usage**
```r
## S3 method for class 'fromXYZ'
plot(x, ...)  
```

**Arguments**
- `x`: object of class spatialAtRisk
- `...`: additional arguments

**Value**
an image plot

### plot.lgcpAutocorr function

**Description**
Plots lgcpAutocorr objects: output from autocorr

**Usage**
```r
## S3 method for class 'lgcpAutocorr'
plot(x, sel = 1:dim(x)[3], ask = TRUE, crop = TRUE, plotwin = FALSE, ...)  
```

**Arguments**
- `x`: an object of class lgcpAutocorr
- `sel`: vector of integers between 1 and grid$len: which grids to plot. Default NULL, in which case all grids are plotted.
- `ask`: logical; if TRUE the user is asked before each plot
- `crop`: whether or not to crop to bounding box of observation window
- `plotwin`: logical whether to plot the window attr(x,"window"), default is FALSE
- `...`: other arguments passed to image.plot
Value

a plot

See Also

autocorr

Examples

## Not run: ac <- autocorr(lg, qt=c(1,2,3))
# assumes that lg has class lgcpPredict
## Not run: plot(ac)

plot.lgcpgrid

plot.lgcpgrid function

Description

This is a wrapper function for image.lgcpgrid

Usage

## S3 method for class 'lgcpgrid'
plot(x, sel = 1:x$len, ask = TRUE, ...)

Arguments

x

an object of class lgcpgrid

sel

vector of integers between 1 and grid$len: which grids to plot. Default NULL, in which case all grids are plotted.

ask

logical; if TRUE the user is asked before each plot

... other arguments

Value

an image-type plot

See Also

lgcpgrid.list, lgcpgrid.array, as.list.lgcpgrid, print.lgcpgrid, summary.lgcpgrid, quantile.lgcpgrid, image.lgcpgrid
plot.lgcpPredict

plot.lgcpPredict function

Description

Simple plotting function for objects of class lgcpPredict.

Usage

## S3 method for class 'lgcpPredict'
plot(
  x,
  type = "relrisk",
  sel = 1:x$EY.mean$len,
  plotdata = TRUE,
  ask = TRUE,
  clipWindow = TRUE,
  ...)

Arguments

- **x**: an object of class lgcpPredict
- **type**: Character string: what type of plot to produce. Choices are "relrisk" (=exp(Y)); "serr" (standard error of relative risk); or "intensity" (=lambda*mu*exp(Y)).
- **sel**: vector of integers between 1 and grid$len: which grids to plot. Default NULL, in which case all grids are plotted.
- **plotdata**: whether or not to overlay the data
- **ask**: logical; if TRUE the user is asked before each plot
- **clipWindow**: whether to plot grid cells outside the observation window
- **...**: additional arguments passed to image.plot

Value

plots the Monte Carlo mean of quantities obtained via simulation. By default the mean relative risk is plotted.

See Also

lgcpPredict
plot.lgcpQuantiles  

*plot.lgcpQuantiles function*

### Description

Plots `lgcpQuantiles` objects: output from `quantiles.lgcpPredict`

### Usage

```r
## S3 method for class 'lgcpQuantiles'
plot(x, sel = 1:dim(x)[3], ask = TRUE, crop = TRUE, plotwin = FALSE, ...)
```

### Arguments

- **x**: an object of class `lgcpQuantiles`
- **sel**: vector of integers between 1 and `grid$len`: which grids to plot. Default `NULL`, in which case all grids are plotted.
- **ask**: logical; if TRUE the user is asked before each plot
- **crop**: whether or not to crop to bounding box of observation window
- **plotwin**: logical whether to plot the window `attr(x, "window")`, default is `FALSE`
- **...**: other arguments passed to `image.plot`

### Value

grid plotting This is a wrapper function for `image.lgcpgrid`

### See Also

`quantile.lgcpPredict`

### Examples

```r
## Not run: qtiles <- quantile(lg, qt=c(0.5,0.75,0.9), fun=exp)
# assumed that lg has class lgcpPredict
## Not run: plot(qtiles)
```
Description

A function to plot lgcpZmat objects

Usage

## S3 method for class 'lgcpZmat'
plot(
x, 
ask = TRUE,
pow = 1,
main = NULL,
miscol = "black",
obswin = NULL,
...
)

Arguments

x an lgcpZmat object, see ?getZmat
ask graphical parameter ask, see ?par
pow power parameter, raises the image values to this power (helps with visualisation, default is 1.)
main title for plot, default is null which gives an automatic title to the plot (the name of the covariate)
miscol colour to identify imputed grid cells, default is yellow
obswin optional observation window to add to plot using plot(obswin).
...
other parameters

Value

a sequence of plots of the interpolated covariate values
plot.mcmcdiag  plot.mcmcdiag function

Description

The command `plot(trace(lg))`, where `lg` is an object of class `lgcpPredict` will plot the mcmc traces of a subset of the cells, provided they have been stored, see `mcmpars`.

Usage

```r
## S3 method for class 'mcmcdiag'
plot(x, idx = 1:dim(x$trace)[2], ...)
```

Arguments

- `x`: an object of class mcmcdiag
- `idx`: vector of chain indices to plot, default plots all chains
- `...`: additional arguments passed to plot

Value

plots the saved MCMC chains

See Also

`mcmctrace.lgcpPredict`, `mcmpars`.

plot.mstppp  plot.mstppp function

Description

Plot method for mstppp objects

Usage

```r
## S3 method for class 'mstppp'
plot(x, cols = "red", ...)
```

Arguments

- `x`: an object of class mstppp
- `cols`: optional vector of colours to plot points with
- `...`: additional arguments passed to plot
Value
plots the stppp object x

Description
Plot method for stppp objects

Usage
## S3 method for class 'stppp'
plot(x, ...)

Arguments
x an object of class stppp
... additional arguments passed to plot

Value
plots the stppp object x

Description
Plot a temporalAtRisk object.

Usage
## S3 method for class 'temporalAtRisk'
plot(x, ...)

Arguments
x an object
... additional arguments

Value
print the object
plotExceed

plotExceed function

Description

A generic function for plotting exceedance probabilities.

Usage

plotExceed(obj, ...)

Arguments

obj
  an object

... additional arguments

Value

generic function returning method plotExceed

See Also

plotExceed.lgcpPredict, plotExceed.array

plotExceed.array

plotExceed.array function

Description

Function for plotting exceedance probabilities stored in array objects. Used in plotExceed.lgcpPredict.

Usage

## S3 method for class 'array'
plotExceed(
  obj,
  fun,
  lgcppredict = NULL,
  xvals = NULL,
  yvals = NULL,
  window = NULL,
)
plotExceed.array

```r
cases = NULL,
nlevel = 64,
ask = TRUE,
mapunderlay = NULL,
alpha = 1,
sub = NULL,
... )
```

Arguments

- **obj**: an object
- **fun**: the name of the function used to compute exceedances (character vector of length 1). Note that the named function must be in memory.
- **lgcppredict**: an object of class lgcpPredict that can be used to supply an observation window and x and y coordinates
- **xvals**: optional vector giving x coords of centroids of cells
- **yvals**: optional vector giving y coords of centroids of cells
- **window**: optional observation window
- **cases**: optional xy (n x 2) matrix of locations of cases to plot
- **nlevel**: number of colour levels to use in plot, default is 64
- **ask**: whether or not to ask for a new plot between plotting exceedances at different thresholds.
- **mapunderlay**: optional underlay to plot underneath maps of exceedance probabilities. Use in conjunction with rainbow parameter ‘alpha’ (eg alpha=0.3) to set transparency of exceedance layer.
- **alpha**: graphical parameter taking values in [0,1] controlling transparency of exceedance layer. Default is 1.
- **sub**: optional subtitle for plot
- **...**: additional arguments passed to image.plot

Value

generic function returning method plotExceed

See Also

plotExceed.lgcpPredict
plotExceed.lgcpPredict

plotExceed.lgcpPredict function

Description

Function for plotting exceedance probabilities stored in lgcpPredict objects.

Usage

## S3 method for class 'lgcpPredict'
plotExceed(
  obj,
  fun,
  nlevel = 64,
  ask = TRUE,
  plotcases = FALSE,
  mapunderlay = NULL,
  alpha = 1,
  ...
)

Arguments

obj an object

fun the name of the function used to compute exceedances (character vector of length 1). Note that the named function must be in memory.

nlevel number of colour levels to use in plot, default is 64

ask whether or not to ask for a new plot between plotting exceedances at different thresholds.

plotcases whether or not to plot the cases on the map

mapunderlay optional underlay to plot underneath maps of exceedance probabilities. Use in conjunction with rainbow parameter 'alpha' (eg alpha=0.3) to set transparency of exceedance layer.

alpha graphical parameter taking values in [0,1] controlling transparency of exceedance layer. Default is 1.

... additional arguments passed to image.plot

Value

plot of exceedances

See Also

lgcpPredict, MonteCarloAverage, setoutput
## Examples

```r
## Not run: exceedfun <- exceedProbs(c(1.5,2,4))
## Not run:
plot(lg,"exceedfun") # lg is an object of class lgcpPredict
# in which the Monte Carlo mean of
# "exceedfun" was computed
# see ?MonteCarloAverage and ?setoutput

## End(Not run)
```

## Description

A function to plot various objects. A developmental tool: not intended for general use

## Usage

```r
plotit(x)
```

## Arguments

- `x`: an a list, matrix, or GPrealisation object.

## Value

plots the objects.

## postcov

### postcov function

## Description

Generic function for producing plots of the posterior covariance function from a call to the function `lgcpPredictSpatialPlusPars`, `lgcpPredictAggregateSpatialPlusPars`, `lgcpPredictSpatialTemporalPlusPars` or `lgcpPredictMultitypeSpatialPlusPars`.

## Usage

```r
postcov(obj, ...)
```

## Arguments

- `obj`: an object
- `...`: additional arguments
postcov.lgcpPredictAggregateSpatialPlusParameters

Value

method postcov

See Also

postcov.lgcpPredictSpatialOnlyPlusParameters, postcov.lgcpPredictAggregateSpatialPlusParameters,
postcov.lgcpPredictSpatioTemporalPlusParameters, postcov.lgcpPredictMultitypeSpatialPlusParameters,
ltr, autocorr, parautocorr, traceplots, parssummary, textsummary, priorpost, exceedProbs, betavals,
etavals

postcov.lgcpPredictAggregateSpatialPlusParameters

postcov.lgcpPredictAggregateSpatialPlusParameters function

Description

A function for producing plots of the posterior covariance function.

Usage

"postcov(obj,qts=c(0.025,0.5,0.975),covmodel=NULL,ask=TRUE,...)"

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>an lgcpPredictAggregateSpatialPlusParameters object</td>
</tr>
<tr>
<td>qts</td>
<td>vector of quantiles of length 3, default is 0.025, 0.5, 0.975</td>
</tr>
<tr>
<td>covmodel</td>
<td>the assumed covariance model. NULL by default, this information is read in from the object obj, so generally does not need to be set.</td>
</tr>
<tr>
<td>ask</td>
<td>parameter &quot;ask&quot;, see ?par</td>
</tr>
<tr>
<td>...</td>
<td>additional arguments</td>
</tr>
</tbody>
</table>

Value

...

See Also

postcov.lgcpPredictSpatialOnlyPlusParameters, postcov.lgcpPredictAggregateSpatialPlusParameters,
postcov.lgcpPredictSpatioTemporalPlusParameters, postcov.lgcpPredictMultitypeSpatialPlusParameters,
ltr, autocorr, parautocorr, traceplots, parssummary, textsummary, priorpost, postcov, exceedProbs,
betavals, etavals
**Description**

A function for producing plots of the posterior covariance function.

**Usage**

```
"postcov(obj,qts=c(0.025,0.5,0.975),covmodel=NULL,ask=TRUE,...)"
```

**Arguments**

- **obj**: an lgcpPredictMultitypeSpatialPlusParameters object
- **qts**: vector of quantiles of length 3, default is 0.025, 0.5, 0.975
- **covmodel**: the assumed covariance model. NULL by default, this information is read in from the object obj, so generally does not need to be set.
- **ask**: parameter "ask", see ?par
- **...**: additional arguments

**Value**

plots of the posterior covariance function for each type.

**See Also**

- `postcov.lgcpPredictMultitypeSpatialOnlyPlusParameters`
- `postcov.lgcpPredictAggregateSpatialPlusParameters`
- `postcov.lgcpPredictSpatioTemporalPlusParameters`
- `postcov.lgcpPredictMultitypeSpatialPlusParameters`
- `ltar`, `autocorr`, `parautocorr`, `traceplots`, `parsummary`, `textsummary`, `priorpost`, `postcov`, `exceedProbs`, `betalvals`, `etavalts`
**postcov.lgcpPredictSpatioTemporalPlusParameters**

**Arguments**

- **obj**: an `lgcpPredictSpatioTemporalPlusParameters` object
- **qts**: vector of quantiles of length 3, default is 0.025, 0.5, 0.975
- **covmodel**: the assumed covariance model. NULL by default, this information is read in from the object `obj`, so generally does not need to be set.
- **ask**: parameter "ask", see `?par`
- ... additional arguments

**Value**

a plot of the posterior spatiotemporal covariance function.

**See Also**

- `postcov.lgcpPredictSpatialOnlyPlusParameters`
- `postcov.lgcpPredictAggregateSpatialPlusParameters`
- `postcov.lgcpPredictSpatioTemporalPlusParameters`
- `postcov.lgcpPredictMultitypeSpatialPlusParameters`
- `ltar`, `autocorr`, `parautocorr`, `traceplots`, `parsummary`, `textsummary`, `priorpost`, `postcov`, `exceedProbs`, `betavals`, `etavals`

---

**Description**

A function for producing plots of the posterior spatiotemporal covariance function.

**Usage**

```
"postcov(obj,qts=c(0.025,0.5,0.975),covmodel=NULL,ask=TRUE,...)"
```

**Arguments**

- **obj**: an `lgcpPredictSpatioTemporalPlusParameters` object
- **qts**: vector of quantiles of length 3, default is 0.025, 0.5, 0.975
- **covmodel**: the assumed covariance model. NULL by default, this information is read in from the object `obj`, so generally does not need to be set.
- **ask**: parameter "ask", see `?par`
- ... additional arguments

**Value**

a plot of the posterior spatial covariance function and temporal correlation function.
See Also

postcov.lgcpPredictSpatialOnlyPlusParameters, postcov.lgcpPredictAggregateSpatialPlusParameters, postcov.lgcpPredictSpatioTemporalPlusParameters, postcov.lgcpPredictMultitypeSpatialPlusParameters, ltar, autocorr, parautocorr, traceplots, parssummary, textsummary, priorpost, postcov, exceedProbs, betavals, etavals

print.dump2dir

Description

Display function for dump2dir objects.

Usage

## S3 method for class 'dump2dir'
print(x, ...)

Arguments

x

an object of class dump2dir

... additional arguments

Value

nothing

See Also

dump2dir,

print.fromFunction

Description

Print method for objects of class fromFunction.

Usage

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

---

**print.dump2dir**

Display function for dump2dir objects.

**Usage**

```r
## S3 method for class 'dump2dir'
print(x, ...)
```

**Arguments**

- `x`: an object of class `dump2dir`
- `...`: additional arguments

**Value**

- `nothing`

**See Also**

- `dump2dir`

---

**print.fromFunction**

Print method for objects of class fromFunction.

**Usage**

```r
## S3 method for class 'fromFunction'
print(x, ...)
```
Arguments

x an object of class spatialAtRisk
...

Value

prints the object

print.fromSPDF print.fromSPDF function

Description

Print method for objects of class fromSPDF.

Usage

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

Arguments

x an object of class spatialAtRisk
...

Value

prints the object

print.fromXYZ print.fromXYZ function

Description

Print method for objects of class fromXYZ.

Usage

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

Arguments

x an object of class spatialAtRisk
...

additional arguments
Description
Print method for gridaverage objects

Usage
## S3 method for class 'gridaverage'
print(x, ...)

Arguments
x an object of class gridaverage
...
other arguments

Value
just prints out details

See Also
lgcpgrid.list, lgcpgrid.array, as.list.lgcpgrid, summary.lgcpgrid, quantile.lgcpgrid, image.lgcpgrid, plot.lgcpgrid
print.lgcpPredict

print.lgcpPredict function

Description
Print method for lgcpPredict objects.

Usage
## S3 method for class 'lgcpPredict'
print(x, ...)

Arguments
x an object of class lgcpPredict
... additional arguments

Value
just prints information to the screen

See Also
lgcpPredict

print.mcmc

print.mcmc function

Description
print method print an mcmc iterator’s details

Usage
## S3 method for class 'mcmc'
print(x, ...)

Arguments
x a mcmc iterator
... other args
print.mstppp

**print.mstppp function**

**Description**
Print method for mstppp objects

**Usage**

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

**Arguments**
- `x` an object of class mstppp
- `...` additional arguments

**Value**
prints the mstppp object x

print.stapp

**print.stapp function**

**Description**
Print method for stapp objects

**Usage**

```r
## S3 method for class 'stapp'
print(x, printhead = TRUE, ...)  
```

**Arguments**
- `x` an object of class stapp
- `printhead` whether or not to print the head of the counts matrix
- `...` additional arguments

**Value**
prints the stapp object x
print.stppp  

**print.stppp function**

**Description**

Print method for stppp objects

**Usage**

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

**Arguments**

- `x` an object of class stppp
- `...` additional arguments

**Value**

prints the stppp object x

---

print.temporalAtRisk  

**print.temporalAtRisk function**

**Description**

Printing method for temporalAtRisk objects.

**Usage**

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

**Arguments**

- `x` an object
- `...` additional arguments

**Value**

print the object

**See Also**

 temporalAtRisk, spatialAtRisk, temporalAtRisk.numeric, temporalAtRisk.function, constantInTime, constantInTime.numeric, constantInTime.stppp, plot.temporalAtRisk
priorpost

priorpost function

Description

A function to plot the prior and posterior densities of the model parameters \( \eta \) and \( \beta \). The prior appears as a red line and the posterior appears as a histogram.

Usage

```r
priorpost(
  obj, 
  breaks = 30, 
  xlab = NULL, 
  ylab = "Density", 
  main = "", 
  ask = TRUE, 
  ... 
)
```

Arguments

- **obj**: an object produced by a call to `lgcpPredictSpatialPlusPars`, `lgcpPredictAggregateSpatialPlusPars`, `lgcpPredictSpatioTemporalPlusPars` or `lgcpPredictMultitypeSpatialPlusPars`
- **breaks**: "breaks" parameter from the function "hist"
- **xlab**: optional label for x-axis, there is a sensible default.
- **ylab**: optional label for y-axis, there is a sensible default.
- **main**: optional title of the plot, there is a sensible default.
- **ask**: the parameter "ask", see ?par
- **...**: other arguments passed to the function "hist"

Value

plots of the prior and posterior of the model parameters \( \eta \) and \( \beta \).

See Also

- `ltar`, `autocorr`, `parautocorr`, `traceplots`, `parssummary`, `textsummary`, `postcov`, `exceedProbs`, `betavals`, `etavals`
**PriorSpec**

*PriorSpec function*

**Description**

Generic for declaring that an object is of valid type for use as as prior in `lgcp`. For further details and examples, see the vignette "Bayesian_lgcp".

**Usage**

`PriorSpec(obj, ...)`

**Arguments**

- `obj` an object
- `...` additional arguments

**Value**

method `PriorSpec`

**See Also**

`PriorSpec.list`

---

**PriorSpec.list**

*PriorSpec.list function*

**Description**

Method for declaring a Bayesian prior density in `lgcp`. Checks to confirm that the object `obj` has the requisite components for functioning as a prior.

**Usage**

```
# S3 method for class 'list'
PriorSpec(obj, ...)
```

**Arguments**

- `obj` a list object defining a prior, see `?GaussianPrior` and `?LogGaussianPrior`
- `...` additional arguments

**Value**

an object suitable for use in a call to the MCMC routines
quantile.lgcpgrid

See Also

GaussianPrior, LogGaussianPrior

Examples

```r
## Not run: PriorSpec(LogGaussianPrior(mean=log(c(1,500)),variance=diag(0.15,2)))
## Not run: PriorSpec(GaussianPrior(mean=rep(0,9),variance=diag(10^6,9)))
```

quantile.lgcpgrid

quantile.lgcpgrid function

Description

Quantile method for lgcpg objects. This just applies the quantile function to each of the elements of x$grid

Usage

```r
## S3 method for class 'lgcpgrid'
quantile(x, ...)
```

Arguments

- `x` an object of class lgcpgrid
- `...` other arguments

Value

Quantiles per grid, see ?quantile for further options

See Also

lgcpgrid.list, lgcpgrid.array, as.list.lgcpgrid, print.lgcpgrid, summary.lgcpgrid, image.lgcpgrid, plot.lgcpgrid
quantile.lgcpPredict  quantile.lgcpPredict function

Description

This function requires data to have been dumped to disk: see `?dump2dir` and `?setoutput`. The routine `quantile.lgcpPredict` computes quantiles of functions of Y. For example, to get cell-wise quantiles of exceedance probabilities, set `fun=exp`. Since computing the quantiles is an expensive operation, the option to output the quantiles on a subregion of interest is also provided (by setting the argument `inWindow`, which has a sensible default).

Usage

```r
## S3 method for class 'lgcpPredict'
quantile(
  x, 
  qt, 
  tidx = NULL, 
  fun = NULL, 
  inWindow = x$xyt$window, 
  crop2parentwindow = TRUE, 
  startidx = 1, 
  sampcount = NULL, 
  ... 
)
```

Arguments

- `x`  an object of class `lgcpPredict`
- `qt` a vector of the required quantiles
- `tidx` the index number of the the time interval of interest, default is the last time point.
- `fun` a 1-1 function (default the identity function) to be applied cell-wise to the grid. Must be able to evaluate `sapply(vec,fun)` for vectors `vec`.
- `inWindow` an observation owin window on which to compute the quantiles, can speed up calculation. Default is `x$xyt$window`.
- `crop2parentwindow` logical: whether to only compute the quantiles for cells inside `x$xyt$window` (the 'parent window')
- `startidx` optional starting sample index for computing quantiles. Default is 1.
- `sampcount` number of samples to include in computation of quantiles after `startidx`. Default is all
- `...` additional arguments
RandomFieldsCovFct

Value

an array, the \([i,i]th\) slice being the grid of cell-wise quantiles, \(q[i]\), of \(\text{fun}(Y)\), where \(Y\) is the MCMC output dumped to disk.

See Also

\(\text{laptopsPredict, dump2dir, setoutput, plot.laptopsQuantiles}\)

Description

A function to declare and also evaluate an covariance function from the RandomFields Package. See ?CovarianceFct. Note that the present version of lgcp only offers estimation for sigma and phi, any additional parameters are treated as fixed.

Usage

RandomFieldsCovFct(model, additionalparameters = c())

Arguments

model the choice of model e.g. "matern"
additionalparameters additional parameters for chosen covariance model. See ?CovarianceFct

Value

a covariance function from the RandomFields package

See Also

\(\text{CovFunction.function, exponentialCovFct, SpikedExponentialCovFct, CovarianceFct}\)

Examples

## Not run: RandomFieldsCovFct(model="matern", additionalparameters=1)
raster.lgcpgrid  

Description

A function to convert lgcpgrid objects into either a raster object, or a RasterBrick object.

Usage

## S3 method for class 'lgcpgrid'
raster(x, crs = NA, transpose = FALSE, ...)

Arguments

x  
an lgcpgrid object

rcrs  
PROJ4 type description of a map projection (optional). See ?raster

transpose  
Logical. Transpose the data? See ?brick method for array

...  
additional arguments

Value

...

rescale.mstppp  

Description

Rescale an mstppp object. Similar to rescale.ppp

Usage

## S3 method for class 'mstppp'
rescale(X, s, unitname)

Arguments

x  
an object of class mstppp

s  
scale as in rescale.ppp: x and y coordinaes are scaled by 1/s

unitname  
parameter as defined in ?rescale

Value

a ppp object without observation times
rescale.stppp

Rescale an stppp object. Similar to rescale.ppp

Usage

## S3 method for class 'stppp'
rescale(X, s, unitname)

Arguments

X
an object of class stppp
s
scale as in rescale.ppp: x and y coordinaes are scaled by 1/s
unitname
parameter as defined in ?rescale

Value

a ppp object without observation times

resetLoop

reset iterator

call this to reset an iterator’s state to the initial

Usage

resetLoop(obj)

Arguments

obj
an mcmc iterator
rgauss

rgauss function

Description

A function to simulate a Gaussian field on a regular square lattice, the returned object is of class lgcpgrid.

Usage

rgauss(
  n = 1,
  range = c(0, 1),
  ncells = 128,
  spatial.covmodel = "exponential",
  model.parameters = lgcppars(sigma = 2, phi = 0.1),
  covpars = c(),
  ext = 2
)

Arguments

n the number of realisations to generate. Default is 1.
range a vector of length 2, defining the left-most and right most cell centroids in the x-direction. Note that the centroids in the y-direction are the same as those in the x-direction.
ncells the number of cells, typially a power of 2
spatial.covmodel spatial covariance function, default is exponential, see ?CovarianceFct
model.parameters parameters of model, see ?lgcppars. Only set sigma and phi for spatial model.
covpars vector of additional parameters for spatial covariance function, in order they appear in chosen model in ?CovarianceFct
ext how much to extend the parameter space by. Default is 2.

Value

an lgcp grid object containing the simulated field(s).
**roteffgain**

**roteffgain function**

**Description**

Compute whether there might be any advantage in rotating the observation window in the object `xyt` for a proposed cell width.

**Usage**

\[ \text{roteffgain}(\text{xyt}, \text{cellwidth}) \]

**Arguments**

- `xyt`: an object of class stppp
- `cellwidth`: size of grid on which to do MALA

**Value**

whether or not there woud be any efficiency gain in the MALA by rotating window

**See Also**

- `getRotation.stppp`

---

**rotmat**

**rotmat function**

**Description**

This function returns a rotation matrix corresponding to an anticlockwise rotation of \( \theta \) radians about the origin.

**Usage**

\[ \text{rotmat}(\text{theta}) \]

**Arguments**

- `theta`: an angle in radians

**Value**

the transformation matrix corresponding to an anticlockwise rotation of \( \theta \) radians about the origin
**rr**  
*rr function*

### Description
Generic function to return relative risk.

### Usage
```r
rr(obj, ...)
```

### Arguments
- `obj`: an object
- `...`: additional arguments

### Value
method `rr`

### See Also
- `lgcpPredict`, `rr.lgcpPredict`

---

**rr.lgcpPredict**  
*rr.lgcpPredict function*

### Description
Accessor function returning the relative risk = \( \exp(Y) \) as an `lgcpgrid` object.

### Usage
```r
## S3 method for class 'lgcpPredict'
rr(obj, ...)
```

### Arguments
- `obj`: an `lgcpPredict` object
- `...`: additional arguments

### Value
the relative risk as computed my MCMC

### See Also
- `lgcpPredict`
samplePosterior

**samplePosterior function**

**Description**

A function to draw a sample from the posterior of a spatial LGCP. Randomly selects an index i, and returns the ith value of eta, the ith value of beta and the ith value of Y as a named list.

**Usage**

```r
samplePosterior(x)
```

**Arguments**

- `x` an object of class `lgcpPredictSpatialOnlyPlusParameters` or `lgcpPredictAggregateSpatialPlusParameters`

**Value**

a sample from the posterior named list object with names elements "eta", "beta" and "Y".

---

segProbs

**segProbs function**

**Description**

A function to compute segregation probabilities from a multivariate LGCP. See the vignette "Bayesian_lgcP" for a full explanation of this.

**Usage**

```r
segProbs(obj, domprob)
```

**Arguments**

- `obj` an `lgcpPredictMultitypeSpatialPlusParameters` object
- `domprob` the threshold beyond which we declare a type as dominant e.g. a value of 0.8 would mean we would consider each type to be dominant if the conditional probability of an event of a given type at that location exceeded 0.8.
Details

We suppose there are K point types of interest. The model for point-type k is as follows:

\[ X_k(s) \sim \text{Poisson}[R_k(s)] \]

\[ R_k(s) = C_A \lambda_k(s) \exp[Z_k(s)\beta_k + Y_k(s)] \]

Here \( X_k(s) \) is the number of events of type k in the computational grid cell containing the point s, \( R_k(s) \) is the Poisson rate, \( C_A \) is the cell area, \( \lambda_k(s) \) is a known offset, \( Z_k(s) \) is a vector of measured covariates and \( Y_i(s) \) where \( i = 1, \ldots, K+1 \) are latent Gaussian processes on the computational grid. The other parameters in the model are \( \beta_k \), the covariate effects for the kth type; and \( \eta_i = [\log(\sigma_i), \log(\phi_i)] \), the parameters of the process \( Y_i \) for \( i = 1, \ldots, K+1 \) on an appropriately transformed (again, in this case log) scale.

The term 'conditional probability of type k' means the probability that at a particular location, \( x \), there will be an event of type k, we denote this \( p_k(x) \).

It is also of interest to scientists to be able to illustrate spatial regions where a genotype dominates a posteriori. We say that type k dominates at position \( x \) if \( p_k(x) > c \), where \( c \) (the parameter domprob) is a threshold is a threshold set by the user. Let \( A_k(c, q) \) denote the set of locations \( x \) for which \( P[p_k(x) > c \mid X] > q \).

As the quantities \( c \) and \( q \) tend to 1 each area \( A_k(c, q) \) shrinks towards the empty set; this happens more slowly in a highly segregated pattern compared with a weakly segregated one.

The function segProbs computes \( P[p_k(x) > c \mid X] \) for each type, from which plots of \( P[p_k(x) > c \mid X] > q \) can be produced.

Value

an lgcpgrid object containing the segregation probabilities.

seintens function

Description

Generic function to return the standard error of the Poisson Intensity.

Usage

seintens(obj, ...)

Arguments

obj an object

... additional arguments
seintens.lgcpPredict function

Description
Accessor function returning the standard error of the Poisson intensity as an lgcpgrid object.

Usage
## S3 method for class 'lgcpPredict'
seintens(obj, ...)

Arguments
obj an lgcpPredict object
... additional arguments

Value
the cell-wise standard error of the Poisson intensity, as computed by MCMC.

See Also
lgcpPredict

selectObsWindow function

Description
See ?selectObsWindow for further details on usage. This is a generic function for the purpose of selecting an observation window (or more precisely a bounding box) to contain the extended FFT grid.

Usage
selectObsWindow(xyt, ...)
selectObsWindow.default

Arguments

- `xyt`: an object
- `cellwidth`: size of the grid spacing in chosen units (equivalent to the cell width argument in `lgcpPredict`)
- `...`: additional arguments

Value

- `method selectObsWindow`

See Also

- `selectObsWindow.default`, `selectObsWindow.stppp`

Description

Default method, note at present, there is only an implementation for `stppp` objects.

Usage

```r
## Default S3 method:
selectObsWindow(xyt, cellwidth, ...)
```

Arguments

- `xyt`: an object
- `cellwidth`: size of the grid spacing in chosen units (equivalent to the cell width argument in `lgcpPredict`)
- `...`: additional arguments

Details

!!NOTE!! that this function also returns the grid ($xvals$ and $yvals$) on which the FFT (and hence MALA) will be performed. It is useful to define `spatialAtRisk` objects on this grid to prevent loss of information from the bilinear interpolation that takes place as part of the fitting algorithm.

Value

- `this is the same as selectObsWindow.stppp`

See Also

- `spatialAtRisk selectObsWindow.stppp`
selectObsWindow.stppp  selectObsWindow.stppp function

Description

This function computes an appropriate observation window on which to perform prediction. Since the FFT grid must have dimension $2^M \times 2^N$ for some $M$ and $N$, the window $xyt$window, is extended to allow this to be fit in for a given cell width.

Usage

```r
## S3 method for class 'stppp'
selectObsWindow(xyt, cellwidth, ...)
```

Arguments

- `xyt`: an object of class stppp
- `cellwidth`: size of the grid spacing in chosen units (equivalent to the cell width argument in `lgcpPredict`)
- `...`: additional arguments

Details

!!NOTE!! that this function also returns the grid ($xvals$ and $yvals$) on which the FFT (and hence MALA) will be performed. It is useful to define spatialAtRisk objects on this grid to prevent loss of information from the bilinear interpolation that takes place as part of the fitting algorithm.

Value

a resized stppp object together with grid sizes $M$ and $N$ ready for FFT, together with the FFT grid locations, can be useful for estimating lambda(s)

See Also

- `spatialAtRisk`
serr  

*serr function*

Description

Generic function to return standard error of relative risk.

Usage

`serr(obj, ...)`

Arguments

- `obj`: an object
- `...`: additional arguments

Value

method `serr`

See Also

`lgcpPredict, serr.lgcpPredict`

serr.lgcpPredict  

*serr.lgcpPredict function*

Description

Accessor function returning the standard error of relative risk as an lgcpgrid object.

Usage

```r
## S3 method for class 'lgcpPredict'

serr(obj, ...)
```

Arguments

- `obj`: an lgcpPredict object
- `...`: additional arguments

Value

Standard error of the relative risk as computed by MCMC.

See Also

`lgcpPredict`
setoutput

Description

Sets output functionality for *lgcpPredict* via the main functions *dump2dir* and *MonteCarloAverage*. Note that it is possible for the user to create their own gridfunction and gridmeans schemes.

Usage

```
setoutput(gridfunction = NULL, gridmeans = NULL)
```

Arguments

- **gridfunction**: what to do with the latent field, but default this set to nothing, but could save output to a directory, see ?dump2dir
- **gridmeans**: list of Monte Carlo averages to compute, see ?MonteCarloAverage

Value

output parameters

See Also

*lgcpPredict, dump2dir, MonteCarloAverage*

setTxtProgressBar2

set the progress bar

Description

update a text progress bar. See help(txtProgressBar) for more info.

Usage

```
setTxtProgressBar2(pb, value, title = NULL, label = NULL)
```

Arguments

- **pb**: text progress bar object
- **value**: new value
- **title**: ignored
- **label**: text for end of progress bar
showGrid

Description
Generic method for displaying the FFT grid used in computation.

Usage
showGrid(x, ...)

Arguments
x
  an object
...
  additional arguments

Value
generic function returning method showGrid

See Also
  showGrid.default, showGrid.lgcpPredict, showGrid.stppp

showGrid.default

Description
Default method for printing a grid to a screen. Arguments are vectors giving the x any y coordinates of the centroids.

Usage
## Default S3 method:
showGrid(x, y, ...)

Arguments
x
  an vector of grid values for the x coordinates
y
  an vector of grid values for the y coordinates
...
  additional arguments passed to points

Value
plots grid centroids on the current graphics device
showGrid.lgcpPredict

See Also

showGrid.lgcpPredict, showGrid.stppp

showGrid.lgcpPredict  showGrid.lgcpPredict function

Description

This function displays the FFT grid used on a plot of an lgcpPredict object. First plot the object using for example plot(lg), where lg is an object of class lgcpPredict, then for any of the plots produced, a call to showGrid(lg,pch="+",cex=0.5) will display the centroids of the FFT grid.

Usage

## S3 method for class 'lgcpPredict'
showGrid(x, ...)

Arguments

x              an object of class lgcpPredict
...            additional arguments passed to points

Value

plots grid centroids on the current graphics device

See Also

lgcpPredict, showGrid.default, showGrid.stppp

showGrid.stppp  showGrid.stppp function

Description

If an stppp object has been created via simulation, ie using the function lgcpSim, then this function will display the grid centroids that were used in the simulation

Usage

## S3 method for class 'stppp'
showGrid(x, ...)

smultiply.list

Arguments

x an object of class stppp. Note this function only applies to SIMULATED data.

... additional arguments passed to points

Value

plots grid centroids on the current graphics device. FOR SIMULATED DATA ONLY.

See Also

lgcpSim, showGrid.default, showGrid.lgcpPredict

Examples

## Not run: xyt <- lgcpSim()
## Not run: plot(xyt)
## Not run: showGrid(xyt,pch="+",cex=0.5)
sparsebase

**sparsebase function**

**Description**

A function that returns the full precision matrix in sparse format from the base of a block circulant matrix, see ?Matrix::sparseMatrix

**Usage**

\[
\text{sparsebase(base)}
\]

**Arguments**

- **base**: base matrix of a block circulant matrix

**Value**

...

spatialAtRisk

**spatialAtRisk function**

**Description**

The methods for this generic function: spatialAtRisk.default, spatialAtRisk.fromXYZ, spatialAtRisk.im, spatialAtRisk.function, spatialAtRisk.SpatialGridDataFrame, spatialAtRisk.SpatialPolygonsDataFrame and spatialAtRisk.bivden are used to represent the fixed spatial component, lambda(s) in the log-Gaussian Cox process model. Typically lambda(s) would be represented as a spatstat object of class im, that encodes population density information. However, regardless of the physical interpretation of lambda(s), in lgcp we assume that it integrates to 1 over the observation window. The above methods make sure this condition is satisfied (with the exception of the method for objects of class function), as well as providing a framework for manipulating these structures. lgcp uses bilinear interpolation to project a user supplied lambda(s) onto a discrete grid ready for inference via MCMC, this grid can be obtained via the selectObsWindow function.

**Usage**

\[
\text{spatialAtRisk}(X, \ldots)
\]

**Arguments**

- **X**: an object
- **\ldots**: additional arguments
Details

Generic function used in the construction of spatialAtRisk objects. The class of spatialAtRisk objects provide a framework for describing the spatial inhomogeneity of the at-risk population, \( \lambda(s) \). This is in contrast to the class of temporalAtRisk objects, which describe the global levels of the population at risk, \( \mu(t) \).

Unless the user has specified \( \lambda(s) \) directly by an R function (a mapping the from the real plane onto the non-negative real numbers, see \texttt{?spatialAtRisk.function}), then it is only necessary to describe the population at risk up to a constant of proportionality, as the routines automatically normalise the \( \lambda \) provided to integrate to 1.

For reference purposes, the following is a mathematical description of a log-Gaussian Cox Process, it is best viewed in the pdf version of the manual.

Let \( \mathcal{Y}(s, t) \) be a spatiotemporal Gaussian process, \( W \subset \mathbb{R}^2 \) be an observation window in space and \( T \subset \mathbb{R}_{\geq 0} \) be an interval of time of interest. Cases occur at spatio-temporal positions \( (x, t) \in W \times T \) according to an inhomogeneous spatio-temporal Cox process, i.e. a Poisson process with a stochastic intensity \( R(x, t) \). The number of cases, \( X_{S, [t_1, t_2]} \), arising in any \( S \subseteq W \) during the interval \( [t_1, t_2] \subseteq T \) is then Poisson distributed conditional on \( R(\cdot) \),

\[
X_{S, [t_1, t_2]} \sim \text{Poisson} \left\{ \int_{S} \int_{t_1}^{t_2} R(s, t)dsdt \right\}
\]

Following Brix and Diggle (2001) and Diggle et al (2005), the intensity is decomposed multiplicatively as

\[
R(s, t) = \lambda(s) \mu(t) \exp\{\mathcal{Y}(s, t)\}.
\]

In the above, the fixed spatial component, \( \lambda : \mathbb{R}^2 \mapsto \mathbb{R}_{\geq 0} \), is a known function, proportional to the population at risk at each point in space and scaled so that

\[
\int_{W} \lambda(s)ds = 1,
\]

whilst the fixed temporal component, \( \mu : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0} \), is also a known function with

\[
\mu(t)\delta t = E[X_{W, \delta t}],
\]

for \( t \) in a small interval of time, \( \delta t \), over which the rate of the process over \( W \) can be considered constant.

Value

method spatialAtRisk


See Also

selectObsWindow lgcpPredict, linklgcpSim, spatialAtRisk.default, spatialAtRisk.fromXYZ, spatialAtRisk.im, spatialAtRisk.function, spatialAtRisk.SpatialGridDataFrame, spatialAtRisk.SpatialPolygonsDataFrame, spatialAtRisk.bivden
spatialAtRisk.bivden  spatialAtRisk.bivden function

Description

Creates a spatialAtRisk object from a sparr bivden object

Usage

## S3 method for class 'bivden'
spatialAtRisk(X, ...)

Arguments

X              a bivden object
...
additional arguments

Value

object of class spatialAtRisk


See Also

lgcpPredict, linklgcpSim, spatialAtRisk.default, spatialAtRisk.fromXYZ, spatialAtRisk.im, spatialAtRisk.default, spatialAtRisk.function, spatialAtRisk.SpatialGridDataFrame, spatialAtRisk.SpatialPolygonsDataFrame

spatialAtRisk.default  spatialAtRisk.default function

Description

The default method for creating a spatialAtRisk object, which attempts to extract x, y and Zm values from the object using xvals, yvals and zvals.

Usage

## Default S3 method:
spatialAtRisk(X, ...)
spatialAtRisk.fromXYZ

Arguments

X  an object

... additional arguments

Value

object of class spatialAtRisk


See Also

lgcpPredict, linklgcpSim, spatialAtRisk.fromXYZ, spatialAtRisk.im, spatialAtRisk.function, spatialAtRisk.SpatialDataFrame, spatialAtRisk.SpatialPolygonsDataFrame, spatialAtRisk.bivden, xvals, yvals, zvals

spatialAtRisk.fromXYZ  spatialAtRisk.fromXYZ.function

Description

Creates a spatialAtRisk object from a list of X, Y, Zm giving respectively the x and y coordinates of the grid and the 'z' values ie so that Zm[i,j] is proportional to the at-risk population at X[i], Y[j].

Usage

## S3 method for class 'fromXYZ'
spatialAtRisk(X, Y, Zm, ...)

Arguments

X  vector of x-coordinates

Y  vector of y-coordinates

Zm  matrix such that Zm[i,j] = f(x[i],y[j]) for some function f

... additional arguments

Value

object of class spatialAtRisk

spatialAtRisk.function

See Also

lgcpPredict, linklgcpSim, spatialAtRisk.default, spatialAtRisk.im, spatialAtRisk.function, spatialAtRisk.SpatialGridDataFrame, spatialAtRisk.SpatialPolygonsDataFrame, spatialAtRisk.bivden

spatialAtRisk.function

spatialAtRisk.function function

Description

Creating a spatialAtRisk object from a function mapping \( \mathbb{R}^2 \) onto the non-negative reals. Note that for spatialAtRisk objects defined in this manner, the user is responsible for ensuring that the integral of the function is 1 over the observation window of interest.

Usage

## S3 method for class `function`
spatialAtRisk(X, warn = TRUE, ...)

Arguments

X
  a function that accepts arguments x and y that returns the at risk population at coordinate (x,y), which should be a numeric of length 1

warn
  whether to issue a warning or not

... additional arguments

Value

object of class spatialAtRisk NOTE The function provided is assumed to integrate to 1 over the observation window, the user is responsible for ensuring this is the case.


See Also

lgcpPredict, linklgcpSim, spatialAtRisk.default, spatialAtRisk.fromXYZ, spatialAtRisk.im, spatialAtRisk.SpatialGridDataFrame, spatialAtRisk.SpatialPolygonsDataFrame, spatialAtRisk.bivden
spatialAtRisk.im  

**spatialAtRisk.im function**

**Description**

Creates a spatialAtRisk object from a spatstat pixel image (im) object.

**Usage**

```r
## S3 method for class 'im'
spatialAtRisk(X, ...)  
```

**Arguments**

- `X`  
  object of class im
- `...`  
  additional arguments

**Value**

object of class spatialAtRisk


**See Also**

lgcpPredict, linklgcpSim, spatialAtRisk.default, spatialAtRisk.fromXYZ, spatialAtRisk.function, spatialAtRisk.SpatialGridDataFrame, spatialAtRisk.SpatialPolygonsDataFrame, spatialAtRisk.bivden

spatialAtRisk.lgcpgrid  

**spatialAtRisk.lgcpgrid function**

**Description**

Creates a spatialAtRisk object from an lgcpgrid object

**Usage**

```r
## S3 method for class 'lgcpgrid'
spatialAtRisk(X, idx = length(X$grid), ...)  
```

```
spatialAtRisk(SpatialGridDataFrame)

Arguments

- **X**: an lgcpgrid object
- **idx**: in the case that X$grid is a list of length > 1, this argument specifies which element of the list to convert. By default, it is the last.
- **...**: additional arguments

Value

object of class spatialAtRisk


See Also

lgcpPredict, linklgcpSim, spatialAtRisk.default, spatialAtRisk.fromXYZ, spatialAtRisk.im, spatialAtRisk.function, spatialAtRisk.SpatialGridDataFrame, spatialAtRisk.SpatialPolygonsDataFrame
spatialAtRisk.SpatialPolygonsDataFrame

spatialAtRisk.SpatialPolygonsDataFrame function

Description

Creates a spatialAtRisk object from a SpatialPolygonsDataFrame object.

Usage

## S3 method for class 'SpatialPolygonsDataFrame'
spatialAtRisk(X, ...)

Arguments

X               a SpatialPolygonsDataFrame object; one column of the data frame should have
                name "atrisk", containing the aggregate population at risk for that region

...            additional arguments

Value

object of class spatialAtRisk

   of the Royal Statistical Society, Series B, 63(4), 823-841.


See Also

lgcpPredict, linklgcpSim, spatialAtRisk.default, spatialAtRisk.fromXYZ, spatialAtRisk.im, spa-
   tialAtRisk.function, spatialAtRisk.SpatialPolygonsDataFrame, spatialAtRisk.bivden
spatialIntensities  

**spatialIntensities function**

**Description**
Generic method for extracting spatial intensities.

**Usage**

```r
spatialIntensities(X, ...)  
```

**Arguments**

- `X` an object
- `...` additional arguments

**Value**

method spatialintensities

**See Also**

- `spatialIntensities.fromXYZ`
- `spatialIntensities.fromSPDF`

---

spatialIntensities.fromSPDF  

**spatialIntensities.fromSPDF function**

**Description**
Extract the spatial intensities from an object of class fromSPDF (as would have been created by `spatialAtRisk.SpatialPolygonsDataFrame` for example).

**Usage**

```r  
## S3 method for class 'fromSPDF'
spatialIntensities(X, xyt, ...)  
```

**Arguments**

- `X` an object of class fromSPDF
- `xyt` object of class stppp or a list object of numeric vectors with names $x$, $y$
- `...` additional arguments
spatialIntensities.fromXYZ

**Value**

normalised spatial intensities

**See Also**

spatialIntensities, spatialIntensities.fromXYZ

---

spatialIntensities.fromXYZ

*spatialIntensities.fromXYZ function*

---

**Description**

Extract the spatial intensities from an object of class fromXYZ (as would have been created by spatialAtRisk for example).

**Usage**

```r
## S3 method for class 'fromXYZ'
spatialIntensities(X, xyt, ...)
```

**Arguments**

- `X` object of class fromXYZ
- `xyt` object of class stppp or a list object of numeric vectors with names $x, $y
- `...` additional arguments

**Value**

normalised spatial intensities

**See Also**

spatialIntensities, spatialIntensities.fromSPDF
spatialparsEst

spatialparsEst function

Description

Having estimated either the pair correlation or K functions using respectively ginhomAverage or KinhomAverage, the spatial parameters sigma and phi can be estimated. This function provides a visual tool for this estimation procedure.

Usage

spatialparsEst(
  gk,
  sigma.range,
  phi.range,
  spatial.covmodel,
  covpars = c(),
  guess = FALSE
)

Arguments

  gk an R object; output from the function KinhomAverage or ginhomAverage
  sigma.range range of sigma values to consider
  phi.range range of phi values to consider
  spatial.covmodel correlation type see ?CovarianceFct
  covpars vector of additional parameters for certain classes of covariance function (eg Matern), these must be supplied in the order given in ?CovarianceFct
  guess logical. Perform an initial guess at parameters? Alternative (the default) sets initial values in the middle of sigma.range and phi.range. NOTE: automatic parameter estimation can be can be unreliable.

Details

To get a good choice of parameters, it is likely that the routine will have to be called several times in order to refine the choice of sigma.range and phi.range.

Value

rpanel function to help choose sigma and phi by eye
References


See Also

ginhomAverage, KinhomAverage, thetaEst, lambdaEst, muEst

SpatialPolygonsDataFrame.stapp

Description

A function to return the SpatialPolygonsDataFrame part of an stapp object

Usage

SpatialPolygonsDataFrame.stapp(from)

Arguments

from stapp object

Value

an object of class SpatialPolygonsDataFrame
**SpikedExponentialCovFct**

*SpikedExponentialCovFct function*

**Description**

A function to declare and also evaluate a spiked exponential covariance function. Note that the present version of lgcp only offers estimation for sigma and phi, the additional parameter 'spikevar' is treated as fixed.

**Usage**

```r
SpikedExponentialCovFct(d, CovParameters, spikevar = 1)
```

**Arguments**

- `d` : total distance
- `CovParameters` : parameters of the latent field, an object of class "CovParamaters".
- `spikevar` : the additional variance at distance 0

**Value**

the spiked exponential covariance function; note that the spikevariance is currently not estimated as part of the MCMC routine, and is thus treated as a fixed parameter.

**See Also**

- `CovFunction.function`, `exponentialCovFct`, `RandomFieldsCovFct`

**stapp**

*stapp function*

**Description**

Generic function for space-time aggregated point-process data

**Usage**

```r
stapp(obj, ...)
```

**Arguments**

- `obj` : an object
- `...` : additional arguments
stapp.SpatialPolygonsDataFrame

Value

method stapp

stapp.list  stapp.list function

Description

A wrapper function for stapp.SpatialPolygonsDataFrame

Usage

## S3 method for class 'list'
stapp(obj, ...)

Arguments

obj an list object as described above, see ?stapp.SpatialPolygonsDataFrame for further details on the requirements of the list

... additional arguments

Details

Construct a space-time aggregated point-process (stapp) object from a list object. The first element of the list should be a SpatialPolygonsDataFrame, the second element of the list a counts matrix, the third element of the list a vector of times, the fourth element a vector giving the bounds of the temporal observation window and the fifth element a spatstat owin object giving the spatial observation window.

Value

an object of class stapp

stapp.SpatialPolygonsDataFrame

stapp.SpatialPolygonsDataFrame function

Description

Construct a space-time aggregated point-process (stapp) object from a SpatialPolygonsDataFrame (along with some other info)

Usage

## S3 method for class 'SpatialPolygonsDataFrame'
stapp(obj, counts, t, tlim, window, ...)

**stGPrealisation**

**Arguments**

- **obj** an SpatialPolygonsDataFrame object
- **counts** a (length(t) by N) matrix containing aggregated case counts for each of the geographical regions defined by the SpatialPolygonsDataFrame, where N is the number of regions
- **t** vector of times, for each element of t there should correspond a column in the matrix `counts`
- **tlim** vector giving the upper and lower bounds of the temporal observation window
- **window** the observation window, of class owin, see ?owin
- **...** additional arguments

**Value**

an object of class stapp

---

**Description**

A function to store a realisation of a spatiotemporal gaussian process for use in MCMC algorithms that include Bayesian parameter estimation. Stores not only the realisation, but also computational quantities.

**Usage**

stGPrealisation(gamma, fftgrid, covFunction, covParameters, d, tdiff)

**Arguments**

- **gamma** the transformed (white noise) realisation of the process
- **fftgrid** an object of class FFTgrid, see ?genFFTgrid
- **covFunction** an object of class function returning the spatial covariance
- **covParameters** an object of class CovParamaters, see ?CovParamaters
- **d** matrix of grid distances
- **tdiff** vector of time differences

**Value**

a realisation of a spatiotemporal Gaussian process on a regular grid
stppp

**stppp function**

Description

Generic function used in the construction of space-time planar point patterns. An stppp object is like a ppp object, but with extra components for (1) a vector giving the time at which the event occurred and (2) a time-window over which observations occurred. Observations are assumed to occur in the plane and the observation window is assumed not to change over time.

Usage

```r
stppp(P, ...)
```

Arguments

- `P`: an object
- `...`: additional arguments

Value

method stppp

See Also

stpp, stpp.ppp, stppp.list

stppp.list

**stppp.list function**

Description

Construct a space-time planar point pattern from a list object

Usage

```r
## S3 method for class 'list'
stppp(P, ...)
```

Arguments

- `P`: list object containing `data`, an (n x 3) matrix corresponding to (x,y,t) values; `tl`, a vector of length 2 giving the observation time window; and `window` giving an owin spatial observation window, see ?owin for more details
- `...`: additional arguments
Value

an object of class stppp

See Also

stppp, stppp.ppp

Description

Construct a space-time planar point pattern from a ppp object

Usage

```r
## S3 method for class 'ppp'
stppp(P, t, tlim, ...)
```

Arguments

- `P` a spatstat ppp object
- `t` a vector of length `P$n`
- `tlim` a vector of length 2 specifying the observation time window
- `...` additional arguments

Value

an object of class stppp

See Also

stppp, stppp.list
summary.lgcpgrid

summary.lgcpgrid function

Description
Summary method for lgcp objects. This just applies the summary function to each of the elements of object$grid.

Usage
## S3 method for class 'lgcpgrid'
summary(object, ...)

Arguments
object         an object of class lgcpgrid
...            other arguments

Value
Summary per grid, see ?summary for further options

See Also
lgcpgrid.list, lgcpgrid.array, as.list.lgcpgrid, print.lgcpgrid, quantile.lgcpgrid, image.lgcpgrid, plot.lgcpgrid

summary.mcmc

summary.mcmc function

Description
summary of an mcmc iterator print out values of an iterator and reset it. DONT call this in a loop that uses this iterator - it will reset it. And break.

Usage
## S3 method for class 'mcmc'
summary(object, ...)

Arguments
object         an mcmc iterator
...            other args
Description

A function to compute the target and gradient for the Bayesian aggregated point process model. Not for general use.

Usage

target.and.grad.AggregateSpatialPlusPars(
    GP,
    prior,
    Z,
    Zt,
    eta,
    beta,
    nis,
    cellarea,
    spatial,
    gradtrunc
)

Arguments

GP an object constructed using GPrealisation
prior the prior, created using lgcpPrior
Z the design matrix on the full FFT grid
Zt the transpose of the design matrix
eta the model parameter, eta
beta the model parameters, beta
nis cell counts on the FFT grid
cellarea the cell area
spatial the poisson offset
gradtrunc the gradient truncation parameter

Value

the target and gradient
target.and.grad.MultitypespatialPlusPars

**Description**

A function to compute the target and gradient for the Bayesian multivariate lgcp

**Usage**

```
target.and.grad.MultitypespatialPlusPars(
    GPlist,
    priorlist,
    Zlist,
    Ztlist,
    eta,
    beta,
    nis,
    cellarea,
    spatial,
    gradtrunc
)
```

**Arguments**

- **GPlist**: list of Gaussian processes
- **priorlist**: list of priors
- **Zlist**: list of design matrices on the FFT grid
- **Ztlist**: list of transposed design matrices
- **eta**: LGCP model parameter eta
- **beta**: LGCP model parameter beta
- **nis**: matrix of cell counts on the extended grid
- **cellarea**: the cell area
- **spatial**: the poisson offset interpolated onto the correct grid
- **gradtrunc**: gradient truncation parameter

**Value**

the target and gradient
Description

A function to compute the target and gradient for 'spatial only' MALA

Usage

target.and.grad.spatial(
  Gamma,
  nis,
  cellarea,
  rootQeigs,
  invrootQeigs,
  mu,
  spatial,
  logspat,
  scaleconst,
  gradtrunc
)

Arguments

- Gamma: current state of the chain, Gamma
- nis: matrix of cell counts
- cellarea: area of cells, a positive number
- rootQeigs: square root of the eigenvectors of the precision matrix
- invrootQeigs: inverse square root of the eigenvectors of the precision matrix
- mu: parameter of the latent Gaussian field
- spatial: spatial at risk function, lambda, interpolated onto correct grid
- logspat: log of spatial at risk function, lambda*scaleconst, interpolated onto correct grid
- scaleconst: the expected number of cases
- gradtrunc: gradient truncation parameter

Value

the back-transformed Y, its exponential, the log-target and gradient for use in MALA_gcpSpatial
target.and.grad.spatialPlusPars

**Description**

A function to compute the target and gradient for the Bayesian spatial LGCP

**Usage**

```r
target.and.grad.spatialPlusPars(
  GP,
  prior,
  Z,
  Zt,
  eta,
  beta,
  nis,
  cellarea,
  spatial,
  gradtrunc
)
```

**Arguments**

- **GP**: an object created using GPrealisation
- **prior**: the model priors, created using lgcpPrior
- **Z**: the design matrix on the FFT grid
- **Zt**: transpose of the design matrix
- **eta**: the parameters, eta
- **beta**: the parameters, beta
- **nis**: cell counts on the FFT grid
- **cellarea**: the cell area
- **spatial**: poisson offset
- **gradtrunc**: the gradient truncation parameter

**Value**

the target and gradient for this model
Description

A function to compute the target and gradient for 'spatial only' MALA

Usage

```r
target.and.grad.spatiotemporal(
  Gamma,
  nis,
  cellarea,
  rootQeigs,
  invrootQeigs,
  mu,
  spatial,
  logspat,
  temporal,
  bt,
  gt,
  gradtrunc
)
```

Arguments

- `Gamma`: current state of the chain, Gamma
- `nis`: matrix of cell counts
- `cellarea`: area of cells, a positive number
- `rootQeigs`: square root of the eigenvectors of the precision matrix
- `invrootQeigs`: inverse square root of the eigenvectors of the precision matrix
- `mu`: parameter of the latent Gaussian field
- `spatial`: spatial at risk function, lambda, interpolated onto correct grid
- `logspat`: log of spatial at risk function, lambda*scaleconst, interpolated onto correct grid
- `temporal`: fitted temporal values
- `bt`: in Brix and Diggle vector b(delta t)
- `gt`: in Brix and Diggle vector g(delta t) (ie the coefficient of R in G(t)), with convention that (deltat[1])=Inf
- `gradtrunc`: gradient truncation parameter

Value

the back-transformed Y, its exponential, the log-target and gradient for use in MALAlgcp
Description

A function to compute the target and gradient for the Bayesian spatiotemporal LGCP.

Usage

target.and.grad.SpatioTemporalPlusPars(
  GP,
  prior,
  Z,
  Zt,
  eta,
  beta,
  nis,
  cellarea,
  spatial,
  gradtrunc,
  ETA0,
  tdiff
)

Arguments

- **GP**: an object created using the stGPrealisation function
- **prior**: the priors for the model, created using lgcPrior
- **Z**: the design matrix on the FFT grid
- **Zt**: the transpose of the design matrix
- **eta**: the parameters eta
- **beta**: the parameters beta
- **nis**: the cell counts on the FFT grid
- **cellarea**: the cell area
- **spatial**: the poisson offset
- **gradtrunc**: the gradient truncation parameter
- **ETA0**: the initial value of eta
- **tdiff**: vector of time differences between time points

Value

the target and gradient for the spatiotemporal model.
Description

Generic function used in the construction of temporalAtRisk objects. A temporalAtRisk object describes the at risk population globally in an observation time window \([t_1, t_2]\). Therefore, for any \(t\) in \([t_1, t_2]\), a temporalAtRisk object should be able to return the global at risk population, \(\mu(t) = \mathbb{E}(\text{number of cases in the unit time interval containing } t)\). This is in contrast to the class of spatialAtRisk objects, which describe the spatial inhomogeneity in the population at risk, \(\lambda(s)\).

Usage

\[
temporalAtRisk(obj, \ldots)
\]

Arguments

- \(obj\): an object
- \(\ldots\): additional arguments

Details

Note that in the prediction routine, \(\text{lgcpPredict}\), and the simulation routine, \(\text{lgcpSim}\), time discretisation is achieved using \(\text{as.integer}\) on both observation times and time limits \(t_1\) and \(t_2\) (which may be stored as non-integer values). The functions that create temporalAtRisk objects therefore return piecewise constant step-functions that can be evaluated for any real \(t\) in \([t_1, t_2]\), but with the restriction that \(\mu(t_i) = \mu(t_j)\) whenever \(\text{as.integer}(t_i) = \text{as.integer}(t_j)\).

A temporalAtRisk object may be (1) ’assumed known’, or (2) scaled to a particular dataset. In the latter case, in the routines available (\text{temporalAtRisk.numeric} and \text{temporalAtRisk.function}), the stpp dataset of interest should be referenced, in which case the scaling of \(\mu(t)\) will be done automatically. Otherwise, for example for simulation purposes, no scaling of \(\mu(t)\) occurs, and it is assumed that the \(\mu(t)\) corresponds to the expected number of cases during the unit time interval containing \(t\). For reference purposes, the following is a mathematical description of a log-Gaussian Cox Process, it is best viewed in the pdf version of the manual.

Let \(\mathcal{Y}(s, t)\) be a spatiotemporal Gaussian process, \(W \subset \mathbb{R}^2\) be an observation window in space and \(T \subset \mathbb{R}_{\geq 0}\) be an interval of time of interest. Cases occur at spatio-temporal positions \((x, t)\) \(\in W \times T\) according to an inhomogeneous spatio-temporal Cox process, i.e. a Poisson process with a stochastic intensity \(R(x, t)\). The number of cases, \(X_{S_\cdot[t_1, t_2]}\), arising in any \(S \subseteq W\) during the interval \([t_1, t_2] \subseteq T\) is then Poisson distributed conditional on \(R(\cdot)\),

\[
X_{S_\cdot[t_1, t_2]} \sim \text{Poisson} \left\{ \int_S \int_{t_1}^{t_2} R(s, t) ds dt \right\}
\]

Following Brix and Diggle (2001) and Diggle et al (2005), the intensity is decomposed multiplicatively as

\[
R(s, t) = \lambda(s) \mu(t) \exp\{\mathcal{Y}(s, t)\},
\]
In the above, the fixed spatial component, $\lambda : R^2 \mapsto R_{\geq 0}$, is a known function, proportional to the population at risk at each point in space and scaled so that
\[ \int_W \lambda(s) ds = 1, \]
whilst the fixed temporal component, $\mu : R_{\geq 0} \mapsto R_{\geq 0}$, is also a known function with
\[ \mu(t) \delta t = E[X_{W,\delta t}], \]
for $t$ in a small interval of time, $\delta t$, over which the rate of the process over $W$ can be considered constant.

Value

method temporalAtRisk


See Also

spatialAtRisk, lgcpPredict, lgcpSim, temporalAtRisk.numeric, temporalAtRisk.function, constantInTime, constantInTime.numeric, constantInTime.stppp, print.temporalAtRisk, plot.temporalAtRisk

temporalAtRisk.function

temporalAtRisk.function function

Description

Create a temporalAtRisk object from a function.

Usage

```r
## S3 method for class `function`
 temporalAtRisk(obj, tlim, xyt = NULL, warn = TRUE, ...) 
```

Arguments

- `obj`: a function accepting single, scalar, numeric argument, t, that returns the temporal intensity for time t
- `tlim`: an integer vector of length 2 giving the time limits of the observation window
- `xyt`: an object of class stppp. If NULL (default) then the function returned is not scaled. Otherwise, the function is scaled so that f(t) = expected number of counts at time t.
- `warn`: Issue a warning if the given temporal intensity treated is treated as 'known'?
- `...`: additional arguments
Details

Note that in the prediction routine, lgcpPredict, and the simulation routine, lgcpSim, time discretisation is achieved using as.integer on both observation times and time limits t_1 and t_2 (which may be stored as non-integer values). The functions that create temporalAtRisk objects therefore return piecewise constant step-functions that can be evaluated for any real t in [t_1,t_2], but with the restriction that $\mu(t_i) = \mu(t_j)$ whenever as.integer(t_i)==as.integer(t_j).

A temporalAtRisk object may be (1) 'assumed known', corresponding to the default argument xyt=NULL; or (2) scaled to a particular dataset (argument xyt=[stppp object of interest]). In the latter case, in the routines available (temporalAtRisk.numeric and temporalAtRisk.function), the dataset of interest should be referenced, in which case the scaling of $\mu(t)$ will be done automatically. Otherwise, for example for simulation purposes, no scaling of $\mu(t)$ occurs, and it is assumed that the $\mu(t)$ corresponds to the expected number of cases during the unit time interval containing t.

Value

a function f(t) giving the temporal intensity at time t for integer t in the interval [tlim[1],tlim[2]] of class temporalAtRisk


See Also

temporalAtRisk, spatialAtRisk, temporalAtRisk.numeric, constantInTime, constantInTime.numeric, constantInTime.stppp, print.temporalAtRisk, plot.temporalAtRisk

temporalAtRisk.numeric

temporalAtRisk.numeric function

Description

Create a temporalAtRisk object from a numeric vector.

Usage

```r
## S3 method for class 'numeric'
temporalAtRisk(obj, tlim, xyt = NULL, warn = TRUE, ...)
```
temporalAtRisk.numeric

Arguments

**obj**
a numeric vector of length \((t_{lim}[2]-t_{lim}[1] + 1)\) giving the temporal intensity up to a constant of proportionality at each integer time within the interval defined by \(t_{lim}\)

**tlim**
an integer vector of length 2 giving the time limits of the observation window

**xyt**
an object of class stppp. If NULL (default) then the function returned is not scaled. Otherwise, the function is scaled so that \(f(t) = \text{expected number of counts at time } t\).

**warn**
Issue a warning if the given temporal intensity treated is treated as 'known'?

... additional arguments

Details

Note that in the prediction routine, lgcpPredict, and the simulation routine, lgcpSim, time discretisation is achieved using as.integer on both observation times and time limits \(t_1\) and \(t_2\) (which may be stored as non-integer values). The functions that create temporalAtRisk objects therefore return piecewise constant step-functions that can be evaluated for any real \(t\) in \([t_1, t_2]\), but with the restriction that \(\mu(t_i) = \mu(t_j)\) whenever \(\text{as.integer}(t_i) = \text{as.integer}(t_j)\).

A temporalAtRisk object may be (1) 'assumed known', corresponding to the default argument \(xyt=\text{NULL}\); or (2) scaled to a particular dataset (argument \(xyt=[\text{stppp object of interest}]\)). In the latter case, in the routines available (temporalAtRisk.numeric and temporalAtRisk.function), the dataset of interest should be referenced, in which case the scaling of \(\mu(t)\) will be done automatically. Otherwise, for example for simulation purposes, no scaling of \(\mu(t)\) occurs, and it is assumed that the \(\mu(t)\) corresponds to the expected number of cases during the unit time interval containing \(t\).

Value

a function \(f(t)\) giving the temporal intensity at time \(t\) for integer \(t\) in the interval as.integer(\([t_{lim}[1], t_{lim}[2]\]) of class temporalAtRisk


See Also

temporalAtRisk, spatialAtRisk, temporalAtRisk.function, constantInTime, constantInTime.numeric, constantInTime.stppp, print.temporalAtRisk, plot.temporalAtRisk
tempRaster

tempRaster function

Description
A function to create a temporary raster object from an x-y regular grid of cell centroids. Useful for projection from one raster to another.

Usage
tempRaster(mcens, ncens)

Arguments
- mcens: vector of equally-spaced coordinates of cell centroids in x-direction
- ncens: vector of equally-spaced coordinates of cell centroids in y-direction

Value
an empty raster object

textsummary

textsummary function

Description
A function to print a text description of the inferred parameters beta and eta from a call to the function lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatioTemporalPlusPars or lgcpPredictMultitypeSpatialPlusPars

Usage
textsummary(obj, digits = 3, scientific = -3, inclIntercept = FALSE, ...)

Arguments
- obj: an object produced by a call to lgcpPredictSpatialPlusPars, lgcpPredictAggregateSpatialPlusPars, lgcpPredictSpatioTemporalPlusPars or lgcpPredictMultitypeSpatialPlusPars
- digits: see the option "digits" in ?format
- scientific: see the option "scientific" in ?format
- inclIntercept: logical: whether to summarise the intercept term, default is FALSE.
- ...: other arguments passed to the function "format"
thetaEst

Value
A text summary, that can be pasted into a LaTeX document and later edited.

See Also
ltar, autocorr, paraautocorr, traceplots, parssummary, priorpost, postcov, exceedProbs, betavals, etavals

---

thetaEst

thetaEst function

Description
A tool to visually estimate the temporal correlation parameter theta; note that sigma and phi must have first been estimated.

Usage

thetaEst(
  xyt,
  spatial.intensity = NULL,
  temporal.intensity = NULL,
  sigma,
  phi,
  theta.range = c(0, 10),
  N = 100,
  spatial.covmodel = "exponential",
  covpars = c()
)

Arguments

xyt object of class stppp
spatial.intensity A spatial at risk object OR a bivariate density estimate of lambda, an object of class im (produced from density.ppp for example),
temporal.intensity either an object of class temporalAtRisk, or one that can be coerced into that form. If NULL (default), this is estimated from the data, seee ?muEst
sigma estimate of parameter sigma
phi estimate of parameter phi
theta.range range of theta values to consider
N number of integration points in computation of C(v,beta) (see Brix and Diggle 2003, corrigendum to Brix and Diggle 2001)
spatial.covmodel spatial covariance model
covpars additional covariance parameters
Value

An r panel tool for visual estimation of temporal parameter theta. NOTE if lambdaEst has been invoked to estimate lambda, then the returned density should be passed to thetaEst as the argument spatial.intensity.

References


See Also

ginhomAverage, KinhomAverage, spatialparsEst, lambdaEst, muEst

toral.cov.mat

toral.cov.mat function

Description

A function to compute the covariance matrix of a stationary process on a torus.

Usage

toral.cov.mat(xg, yg, sigma, phi, model, additionalparameters)

Arguments

xg x grid
yg y grid
sigma spatial variability parameter
phi spatial decay parameter
model model for covariance, see ?CovarianceFct
additionalparameters additional parameters for covariance structure

Value

circulant covariance matrix
**touchingowin**  
**touchingowin function**

**Description**

A function to compute which cells are touching an owin or spatial polygons object

**Usage**

```r
touchingowin(x, y, w)
```

**Arguments**

- `x`  
  grid centroids in x-direction note this will be expanded into a GRID of (x,y) values in the function

- `y`  
  grid centroids in y-direction note this will be expanded into a GRID of (x,y) values in the function

- `w`  
  an owin or SpatialPolygons object

**Value**

vector of TRUE or FALSE according to whether the cell

---

**traceplots**  
**traceplots function**

**Description**

A function to produce trace plots for the parameters beta and eta from a call to the function `lgcpPredictSpatialPlusPars`, `lgcpPredictAggregateSpatialPlusPars`, `lgcpPredictSpatioTemporalPlusPars` or `lgcpPredictMultitypeSpatialPlusPars`

**Usage**

```r
traceplots(obj, xlab = "Sample No.", ylab = NULL, main = "", ask = TRUE, ...)
```

**Arguments**

- `obj`  
  an object produced by a call to `lgcpPredictSpatialPlusPars`, `lgcpPredictAggregateSpatialPlusPars`, `lgcpPredictSpatioTemporalPlusPars` or `lgcpPredictMultitypeSpatialPlusPars`

- `xlab`  
  optional label for x-axis, there is a sensible default.

- `ylab`  
  optional label for y-axis, there is a sensible default.

- `main`  
  optional title of the plot, there is a sensible default.

- `ask`  
  the parameter "ask", see `?par`

- `...`  
  other arguments passed to the function "hist"
transblack function

Value
produces MCMC trace plots of the parameters beta and eta

See Also
ltar, autocorr, parautocorr, pars_summary, te_summary, priorpost, postcov, exceedProbs, betavals, etavals

transblack

Description
A function to return a transparent black colour.

Usage
transblack(alpha = 0.1)

Arguments
alpha transparency parameter, see ?rgb

Value
character string of colour

transblue

Description
A function to return a transparent blue colour.

Usage
transblue(alpha = 0.1)

Arguments
alpha transparency parameter, see ?rgb

Value
character string of colour
transgreen function

Description
A function to return a transparent green colour.

Usage
transgreen(alpha = 0.1)

Arguments
alpha transparency parameter, see ?rgb

Value
character string of colour

transred function

Description
A function to return a transparent red colour.

Usage
transred(alpha = 0.1)

Arguments
alpha transparency parameter, see ?rgb

Value
character string of colour
A text progress bar with label

Description
This is the base txtProgressBar but with a little modification to implement the label parameter for style=3. For full info see txtProgressBar

Usage

```r
txtProgressBar2(
  min = 0,
  max = 1,
  initial = 0,
  char = "=",
  width = NA,
  title = "",
  label = "",
  style = 1
)
```

Arguments

- `min`: min value for bar
- `max`: max value for bar
- `initial`: initial value for bar
- `char`: the character (or character string) to form the progress bar.
- `width`: progress bar width
- `title`: ignored
- `label`: text to put at the end of the bar
- `style`: bar style

updateAMCMC

`updateAMCMC` function

Description

A generic to be used for the purpose of user-defined adaptive MCMC schemes, `updateAMCMC` tells the MALA algorithm how to update the value of h. See lgcp vignette, codevignette("lgcp"), for further details on writing adaptive MCMC schemes.

Usage

```r
updateAMCMC(obj, ...)
```
updateAMCMC.andrieuthomsh

Arguments

obj an object
...
additional arguments

Value

method updateAMCMC

See Also

updateAMCMC.constanth, updateAMCMC.andrieuthomsh

updateAMCMC.andrieuthomsh

updateAMCMC.andrieuthomsh function

Description

Updates the andrieuthomsh adaptive scheme.

Usage

## S3 method for class 'andrieuthomsh'
updateAMCMC(obj, ...)

Arguments

obj an object
...
additional arguments

Value

update and return current h for scheme

References


See Also

andrieuthomsh
**updateAMCMC::constanth**

Description

Updates the `constanth` adaptive scheme.

Usage

```r
## S3 method for class 'constanth'
updateAMCMC(obj, ...)
```

Arguments

- `obj`: an object
- `...`: additional arguments

Value

update and return current h for scheme

See Also

- `constanth`

**varfield**

varfield function

Description

Generic function to extract the variance of the latent field Y.

Usage

```r
varfield(obj, ...)
```

Arguments

- `obj`: an object
- `...`: additional arguments

Value

method meanfield

See Also

- `lgcpPredict`
varfield.lgcpPredict function

Description

This is an accessor function for objects of class lgcpPredict and returns the variance of the field \( Y \) as an lgcpgrid object.

Usage

```r
## S3 method for class 'lgcpPredict'
varfield(obj, ...)
```

Arguments

- `obj`: an object of class lgcpPredict
- `...`: additional arguments

Value

returns the cell-wise variance of \( Y \) computed via Monte Carlo.

See Also

- `lgcpPredict`

varfield.lgcpPredictINLA function

Description

A function to return the variance of the latent field from a call to lgcpPredictINLA output.

Usage

```r
## S3 method for class 'lgcpPredictINLA'
varfield(obj, ...)
```

Arguments

- `obj`: an object of class lgcpPredictINLA
- `...`: other arguments

Value

the variance of the latent field
window.lgcpPredict

window.lgcpPredict function

Description
Accessor function returning the observation window from objects of class lgcpPredict. Note that for computational purposes, the window of an stpp object will be extended to accommodate the requirement that the dimensions must be powers of 2. The function window.lgcpPredict returns the extended window.

Usage
## S3 method for class 'lgcpPredict'
window(x, ...)

Arguments
x an object of class lgcpPredict
... additional arguments

Value
returns the observation window used during computation

See Also
lgcpPredict

wpopdata Population of Welsh counties

Description
Population of Welsh counties

Usage
data(wpopdata)

Format
matrix

Source
ONS
References

http://www.statistics.gov.uk/default.asp

wtowncoords

Welsh town details: location

Description

Welsh town details: location

Usage

data(wtowncoords)

Format

matrix

Source

Wikipedia

References

https://www.wikipedia.org/

wtowns

Welsh town details: population

Description

Welsh town details: population

Usage

data(wtowns)

Format

matrix

Source

ONS

References

http://www.statistics.gov.uk/default.asp
xvals

xvals function

Description

Generic for extracting the 'x values' from an object.

Usage

xvals(obj, ...)

Arguments

obj an object of class spatialAtRisk
...

Value

the xvals method

See Also

yvals, zvals, xvals.default, yvals.default, zvals.default, xvals.fromXYZ, yvals.fromXYZ, zvals.fromXYZ, xvals.SpatialGridDataFrame, yvals.SpatialGridDataFrame, zvals.SpatialGridDataFrame

xvals.default

xvals.default function

Description

Default method for extracting 'x values' looks for $X, $x in that order.

Usage

## Default S3 method:
xvals(obj, ...)

Arguments

obj an object
...

Value

the x values
xvals.lgcpPredict

xvals.lgcpPredict function

Description

Gets the x-coordinates of the centroids of the prediction grid.

Usage

## S3 method for class 'lgcpPredict'
xvals(obj, ...)

Arguments

obj an object of class lgcpPredict
... additional arguments

See Also

xvals, yvals, zvals, xvals.default, yvals.default, zvals.default, xvals.fromXYZ, yvals.fromXYZ, zvals.fromXYZ, xvals.SpatialGridDataFrame, yvals.SpatialGridDataFrame, zvals.SpatialGridDataFrame

xvals.fromXYZ

xvals.fromXYZ function

Description

Method for extracting 'x values' from an object of class fromXYZ

Usage

## S3 method for class 'fromXYZ'
xvals(obj, ...)

Arguments

obj a spatialAtRisk object
... additional arguments

Value

the x values

See Also

xvals, yvals, zvals, xvals.default, yvals.default, zvals.default, xvals.fromXYZ, yvals.fromXYZ, zvals.fromXYZ, xvals.SpatialGridDataFrame, yvals.SpatialGridDataFrame, zvals.SpatialGridDataFrame

xvals.lgcpPredict

xvals.lgcpPredict function

Description

Gets the x-coordinates of the centroids of the prediction grid.

Usage

## S3 method for class 'lgcpPredict'
xvals(obj, ...)

Arguments

obj an object of class lgcpPredict
... additional arguments

See Also

xvals, yvals, zvals, xvals.default, yvals.default, zvals.default, xvals.fromXYZ, yvals.fromXYZ, zvals.fromXYZ, xvals.SpatialGridDataFrame, yvals.SpatialGridDataFrame, zvals.SpatialGridDataFrame
Value

the x coordinates of the centroids of the grid

See Also

lgcpPredict
**YfromGamma**  

*YfromGamma function*

**Description**  
A function to change Gammas (white noise) into Ys (spatially correlated noise). Used in the MALA algorithm.

**Usage**  

YfromGamma(Gamma, invrootQeigs, mu)

**Arguments**  

- **Gamma**: Gamma matrix  
- **invrootQeigs**: inverse square root of the eigenvectors of the precision matrix  
- **mu**: parameter of the latent Gaussian field

**Value**  

Y

---

**yvals**  

*yvals function*

**Description**  
Generic for extractign the 'y values' from an object.

**Usage**  

yvals(obj, ...)

**Arguments**  

- **obj**: an object of class spatialAtRisk  
- **...**: additional arguments

**Value**  

the yvals method

**See Also**  

xvals, zvals, xvals.default, yvals.default, zvals.default, xvals.fromXYZ, yvals.fromXYZ, zvals.fromXYZ, xvals.SpatialGridDataFrame, yvals.SpatialGridDataFrame, zvals.SpatialGridDataFrame
Description

Default method for extracting 'y values' looks for $Y$, $y$ in that order.

Usage

```
## Default S3 method:
yvals(obj, ...)
```

Arguments

- **obj**: an object
- **...**: additional arguments

Value

the y values

See Also

xvals, yvals, zvals, xvals.default, zvals.default, xvals.fromXYZ, yvals.fromXYZ, zvals.fromXYZ, xvals.SpatialGridDataFrame, yvals.SpatialGridDataFrame, zvals.SpatialGridDataFrame
See Also

xvals, yvals, zvals, xvals.default, yvals.default, zvals.default, xvals.fromXYZ, zvals.fromXYZ, xvals.SpatialGridDataFrame, yvals.SpatialGridDataFrame, zvals.SpatialGridDataFrame

---

yvals.lgcpPredict  yvals.lgcpPredict function

Description

Gets the y-coordinates of the centroids of the prediction grid.

Usage

## S3 method for class 'lgcpPredict'
yvals(obj, ...)

Arguments

obj an object of class lgcpPredict
...
additional arguments

Value

the y coordinates of the centroids of the grid

See Also

lgcpPredict

---

yvals.SpatialGridDataFrame  yvals.SpatialGridDataFrame function

Description

Method for extracting 'y values' from an object of class SpatialGridDataFrame

Usage

## S3 method for class 'SpatialGridDataFrame'
yvals(obj, ...)

Arguments

obj an object
...
additional arguments
zvals

Value
the y values

See Also
xvals, yvals, zvals, xvals.default, yvals.default, zvals.default, xvals.fromXYZ, yvals.fromXYZ, zvals.fromXYZ,
xvals.SpatialGridDataFrame, yvals.SpatialGridDataFrame, zvals.SpatialGridDataFrame

zvals
zvals function

Description
Generic for extracting the 'z values' from an object.

Usage
zvals(obj, ...)

Arguments
obj an object
... additional arguments

Value
the zvals method

See Also
xvals, yvals, xvals.default, yvals.default, zvals.default, xvals.fromXYZ, yvals.fromXYZ, zvals.fromXYZ,
xvals.SpatialGridDataFrame, yvals.SpatialGridDataFrame, zvals.SpatialGridDataFrame

zvals.default
zvals.default function

Description
Default method for extracting 'z values' looks for $Zm, $Z, $z in that order.

Usage
## Default S3 method:
zvals(obj, ...)
Arguments

obj       an object
...      additional arguments

Value

class 'fromXYZ'

See Also

xvals, yvals, zvals, xvals.default, yvals.default, xvals.fromXYZ, yvals.fromXYZ, zvals.fromXYZ, xvals.SpatialGridDataFrame, yvals.SpatialGridDataFrame, zvals.SpatialGridDataFrame

Description

Method for extracting 'z values' from an object of class fromXYZ

Usage

## S3 method for class 'fromXYZ'
zvals(obj, ...)

Arguments

obj       a spatialAtRisk object
...      additional arguments

Value

class 'fromXYZ'

See Also

xvals, yvals, zvals, xvals.default, yvals.default, zvals.default, xvals.fromXYZ, yvals.fromXYZ, xvals.SpatialGridDataFrame, yvals.SpatialGridDataFrame, zvals.SpatialGridDataFrame
**zvals.SpatialGridDataFrame**

**zvals.SpatialGridDataFrame function**

---

**Description**

Method for extracting 'z values' from an object of class SpatialGridDataFrame

**Usage**

```r
## S3 method for class 'SpatialGridDataFrame'
zvals(obj, ...)
```

**Arguments**

- `obj` an object
- `...` additional arguments

**Value**

the z values

**See Also**

xvals, yvals, zvals, xvals.default, yvals.default, zvals.default, xvals.fromXYZ, yvals.fromXYZ, zvals.fromXYZ,
xvals.SpatialGridDataFrame, yvals.SpatialGridDataFrame
Index

* datasets
  wpopdata, 271
  wtowncoords, 272
  wtowns, 272
* package
  lgcp-package, 9
  .onAttach, 11
  [.mstppp (Extract.mstppp), 62
  [.stppp (Extract.stppp), 62
  add.list, 12
  addTemporalCovariates, 12, 31, 36, 38, 82, 85, 105, 129, 139, 142, 150, 153, 154
  affine.fromFunction, 13
  affine.fromSPDF, 14
  affine.fromXYZ, 14
  affine.SpatialPolygonsDataFrame, 15
  affine.stppp, 15
  aggCovInfo, 16
  aggCovInfo.ArealWeightedMean, 16
  aggCovInfo.ArealWeightedSum, 17
  aggCovInfo.Majority, 17
  aggregateCovariateInfo, 18
  aggregateformulaList, 18
  andrieuthomsh, 19, 110, 111, 268
  as.array.lgcpgrid, 20
  as.fromXYZ, 20, 21–23
  as.fromXYZ.fromFunction, 21
  as.im.fromFunction, 21, 22, 22, 23
  as.im.fromSPDF, 21, 22, 22, 23
  as.im.fromXYZ, 21–23, 23
  as.list.lgcpgrid, 24, 109, 126–128, 194, 209, 215, 250
  as.owin.stapp, 24
  as.owinlist, 25
  as.owinlist.SpatialPolygonsDataFrame, 25
  as.owinlist.stapp, 26
  as.ppp.mstppp, 26
  as.ppp.stppp, 27
  as.SpatialGridDataFrame, 27, 28
  as.SpatialGridDataFrame.fromXYZ, 27, 28
  as.SpatialPixelsDataFrame, 28
  as.SpatialPixelsDataFrame.lgcpgrid, 29, 29
  as.stppp, 29
  as.stppp.stapp, 30
  assigninterp, 30
  at, 32
  autocorr, 32, 35, 43, 55, 139, 142, 150, 153, 163, 191, 192, 194, 204–207, 213, 262, 265
  autocorrMultitype, 33
  BetaParameters, 34
  betavals, 33, 34, 43, 55, 139, 142, 150, 153, 163, 191, 192, 204–207, 213, 262, 265
  blockcircbase, 35
  blockcircbaseFunction, 36
  bt.scalar, 36
  checkObsWin, 37
  chooseCellwidth, 13, 31, 36, 37, 37, 82, 85, 105, 129, 154
  circulant, 38
  circulant.matrix, 39
  circulant.numeric, 39
  clearinterp, 40
  computeGradtruncSpatial, 40
  computeGradtruncSpatioTemporal, 41
  condProbs, 42
  constanth, 43, 111, 269
  constantInTime, 44, 45, 46, 184, 185, 200, 212, 258–260
  constantInTime.numeric, 44, 45, 46, 200, 212, 258–260
  constantInTime.stppp, 44, 45, 46, 200, 212, 258–260
  cov.interp.fft, 46
INDEX

CovFunction, 13, 31, 38, 49, 82, 85, 105, 129, 139, 142, 150, 153, 154
CovFunction.function, 49, 49, 60, 174, 175, 217, 245
CovParameters, 50
Cvb, 50

d.func, 51
density.ppp, 52
density.stppp, 52
discreteWindow, 52
discreteWindow.lgcpPredict, 53, 53
dump2dir, 33, 53, 58, 62, 87, 88, 90, 92, 207, 217, 229
eigenfrombase, 54
etavals, 33, 35, 43, 54, 139, 142, 150, 153, 163, 191, 192, 204–207, 213, 262, 265
EvaluatePrior, 55
exceedProbs, 33, 35, 43, 54, 55, 56, 139, 142, 150, 153, 163, 181, 191, 192, 204–207, 207, 213, 262, 265
exceedProbsAggregated, 56
expectation, 48, 57, 189
expectation.lgcpPredict, 58, 133, 136, 145, 147
expectation.lgcpPredictSpatialOnlyPlusParameters, 59
exponentialCovFct, 49, 50, 59, 217, 245
extendspatialAtRisk, 60
extract, 60
extract.lgcpPredict, 60, 61, 61, 133, 136, 145, 147
Extract.mstppp, 62
Extract.stppp, 62

fftgrid, 40–42, 63, 64–66
fftinterpolate, 64
fftinterpolate.fromFunction, 64
fftinterpolate.fromSPDF, 65
fftinterpolate.fromXYZ, 66
fftmultiply, 66
formulaList, 67
GAfinalise, 67, 68–75, 181, 187

G Afinalise.MonteCarloAverage, 68
G Afinalise.nullAverage, 68
G Ainitialise, 68, 69, 69, 70–75, 181, 187
G Ainitialise.MonteCarloAverage, 70
G Ainitialise.nullAverage, 70
GammaFromY, 71
GAreturnvalue, 68–71, 71, 72–75, 181, 187
GAreturnvalue.MonteCarloAverage, 72
GAreturnvalue.nullAverage, 72
G Aupdate, 68–73, 73, 74, 75, 181, 187
G Aupdate.MonteCarloAverage, 74
G Aupdate.nullAverage, 74
GaussianPrior, 75, 154, 162, 215
genFFTgrid, 75
getCellCounts, 76
getCounts, 77
getCovParameters, 78
getCovParameters.GPrealisation, 78
getCovParameters.list, 79
getinterp, 79
getlgcpPredictSpatialINLA, 80
getLHSformulaList, 80
getpolyol, 13, 31, 36, 38, 81, 85, 105, 129, 139, 142, 150, 153, 154
gerotation, 82
gerotation.default, 82
gerotation.stppp, 82, 83, 83, 221
getup, 83
getzmat, 13, 31, 36, 38, 82, 84, 105, 129, 139, 142, 150, 153, 154
getzmats, 85
gFFinalise, 54, 86, 87–92, 188
gFFinalise.dump2dir, 86
gFFinalise.nullFunction, 87
gFinitialise, 54, 86, 87, 87, 88–92, 188
gFinitialise.dump2dir, 88
gFinitialise.nullFunction, 88
gFreturnvalue, 54, 86–89, 89, 90–92, 188
gFreturnvalue.dump2dir, 90
gFreturnvalue.nullFunction, 90
gFupdate, 54, 86–91, 91, 92, 188
gFupdate.dump2dir, 91
gFupdate.nullFunction, 92
ginhomAverage, 93, 119–123, 133, 136, 145, 147, 185, 243, 244, 263
G Overlay, 94
GPdrv, 95
GPdrv2, 96
is.burnin
is.retain
iteration
is.SPD
interptypes
integerise.stppp
integerise.mstppp
integerise
initialiseAMCMC.andrieuthomsh
initialiseAMCMC
image.lgcpgrid
identifygrid
identify.lgcpPredict
hasNext
hasNext.iter
hvals
hvals.lgcpPredict
identify.lgcpPredict
identifygrid
image.lgcpgrid
initialiseAMCMC
initialiseAMCMC.andrieuthomsh
initialiseAMCMC.constanth
integerise
integerise.mstppp
integerise.stppp
intens
intens.lgcpPredict
intens.lgcpSimMultitypeSpatialPlusParameters
intens.lgcpSimSpatialPlusParameters
interptypes
inversebase
is.burnin
is.pow2
is.retain
is.SPD
iteration

KinhomAverage, 94, 119, 121–123, 133, 136, 145, 147, 185, 243, 244, 263
lambdaEst, 94, 120, 120, 133, 136, 145, 147, 185, 244, 263
lambdaEst.ppp, 120, 121
lambdaEst.stppp, 120, 122
lgcp (lgcp-package), 9
lgcp-package, 9
lgcpbayes, 123
lgcpgrid, 101–103, 125, 180
lgcpgrid.array, 24, 109, 125, 126, 127, 194, 209, 215, 250
lgcpgrid.list, 24, 109, 125, 126, 127, 128, 194, 209, 215, 250
lgcpgrid.matrix, 125, 128
lgcpInits, 13, 31, 36, 38, 82, 85, 105, 129, 139, 142, 150, 153, 154
lgcppars, 129, 133, 136, 145, 147
lgcpPredictAggregated, 57, 133
lgcpPredictAggregateSpatialPlusPars, 13, 31, 36, 38, 48, 82, 85, 105, 129, 136, 142, 150, 153, 154, 189
lgcpPredictMultitypeSpatialPlusPars, 13, 31, 36, 38, 82, 85, 105, 129, 139, 139, 150, 153, 154
lgcpPredictSpatial, 142
lgcpPredictSpatialINLA, 145
lgcpPredictSpatialPlusPars, 13, 31, 36, 38, 82, 85, 105, 129, 139, 142, 147, 153, 154, 189
lgcpSimMultitypeSpatialPlusParameters, 13, 31, 36, 38, 82, 85, 105, 129, 139, 142, 150, 150, 154
lgcpPrior, 13, 31, 36, 38, 82, 85, 105, 129, 139, 142, 150, 153, 153, 154
lgcpSim, 154, 232, 257–260
lgcpSimMultitypeSpatialCovariates, 157
lgcpSimSpatial, 158
lgcpSimSpatialCovariates, 159
lgcpvignette, 160
loc2poly, 62, 108, 109, 161
LogGaussianPrior, 75, 154, 161, 215
loop.mcmc, 162
ltar, 33, 35, 43, 55, 139, 142, 150, 153, 162, 191, 192, 204–207, 213, 262, 265
MALAlgcp, 163
MALAlgcpAggregateSpatial.PlusPars, 165
MALAlgcpMultitypeSpatial.PlusPars, 166
MALAlgcpSpatial, 168
MALAlgcpSpatial.PlusPars, 170
MALAlgcpSpatioTemporal.PlusPars, 171
matchcovariance, 173
maternCovFct15, 174
maternCovFct25, 175
mcmcLoop, 175
mcmcpars, 20, 44, 133, 136, 145, 147, 176, 198
mcmcProgressNone, 176
mcmcProgressPrint, 177
mcmcProgressTextBar, 177
mcmcProgressTk, 178
mcmctrace, 178
mcmctrace.lgcpPredict, 133, 136, 145, 147, 179, 198
meanfield, 179
meanfield.lgcpPredict, 133, 136, 145, 147, 180
meanfield.lgcpPredictINLA, 180
MonteCarloAverage, 56, 68, 70, 72, 74, 181, 202, 229
mstppp, 182, 182, 183, 184
mstppp.list, 182, 183, 184
mstppp.ppp, 182, 183, 183
mstppp.stppp, 184
muEst, 94, 120, 122, 123, 133, 136, 145, 147, 184, 244, 263
multiply.list, 185
neattable, 186
neigh2D, 186
nextStep, 187
nullAverage, 69, 71, 73, 75, 187
nullFunction, 87, 89, 91, 92, 188
numCases, 188
osppp2latlon, 189
osppp2merc, 189
paramprec, 190
paramprecbase, 190
parautocorr, 33, 35, 43, 55, 139, 142, 150, 153, 163, 191, 192, 204–207, 213, 262, 265
parsummary, 33, 35, 43, 55, 139, 142, 150, 153, 163, 191, 191, 204–207, 213, 262, 265
plot.fromSPDF, 192
plot.fromXYZ, 193
plot.lgcpAutocorr, 33, 193
plot.lgcpgrid, 24, 109, 126–128, 194, 209, 215, 250
plot.lgcpPredict, 133, 136, 145, 147, 195
plot.lgcpQuantiles, 196, 217
plot.lgcpZmat, 197
plot.mcmcdiag, 179, 198
plot.mstppp, 198
plot.stppp, 199
plot.temporalAtRisk, 44–46, 199, 212, 258–260
plotExceed, 200
plotExceed.array, 200, 200
plotExceed.lgcpPredict, 133, 136, 145, 147, 200, 201, 202
plotit, 203
postcov, 33, 35, 43, 55, 139, 142, 150, 153, 163, 191, 192, 203, 204–207, 213, 262, 265
postcov.lgcpPredictAggregateSpatialPlusParameters, 43, 204, 204, 205–207
postcov.lgcpPredictMultitypeSpatialPlusParameters, 43, 204, 205, 206, 207
postcov.lgcpPredictSpatialOnlyPlusParameters, 43, 204, 205, 206, 207
postcov.lgcpPredictSpatioTemporalPlusParameters, 43, 204–206, 206, 207
print.dump2dir, 207
print.fromFunction, 207
print.fromSPDF, 208
print.fromXYZ, 208
print.gridaverage, 209
print.lgcpgrid, 24, 109, 126–128, 194, 209, 215, 250
print.lgcpPredict, 133, 136, 145, 147, 210
print.mcmc, 210
print.mstppp, 211
print.stapp, 211
print.stppp, 212
print.temporalAtRisk, 44–46, 200, 212,
258–260
priorpost, 33, 35, 43, 55, 139, 142, 150, 153, 163, 191, 192, 204–207, 213, 262, 265
PriorSpec, 214
PriorSpec.list, 154, 214, 214
quantile.lgcpgrid, 24, 109, 126–128, 194, 209, 215, 250
quantile.lgcpgrid.default, 213, 181
RandomFieldsCovFct, 49, 50, 60, 174, 175, 217, 245
raster.lgcpgrid, 218
rescale.mstppp, 218
rescale.stppp, 219
resetLoop, 219
gauss, 220
roteffgain, 221
rotmat, 221
rr, 222
rr.lgcpgrid, 133, 136, 145, 147, 222, 222
samplePosterior, 223
segProbs, 43, 223
seintens, 224
seintens.lgcpgrid, 225, 225
selectObsWindow, 225, 233, 234
selectObsWindow.default, 226, 226
selectObsWindow.stppp, 226, 227
serr, 228
serr.lgcpgrid, 133, 136, 145, 147, 222, 228
setTxtProgressBar2, 229
showGrid, 230
showGrid.default, 230, 230, 231, 232
showGrid.lgcpgrid, 133, 136, 145, 147, 230, 231, 231, 232
showGrid.stppp, 156, 230, 231, 231
smultiply.list, 232
sparsebase, 233
spatialAtRisk, 44–46, 122, 123, 133, 136, 145, 147, 200, 212, 226, 227, 233, 257–260
spatialAtRisk.bivden, 233, 234, 235, 236–238, 240
spatialAtRisk.default, 233–235, 235, 237–240
spatialAtRisk.fromXYZ, 66, 233–236, 236, 237–240
spatialAtRisk.function, 65, 233–237, 237, 238–240
spatialAtRisk.im, 233–237, 238, 239, 240
spatialAtRisk.lgcpgrid, 238
spatialAtRisk.SpatialGridDataFrame, 233–239, 239, 240
spatialAtRisk.SpatialPolygonsDataFrame, 65, 233–240, 240
spatialIntensities, 241, 242
spatialIntensities.fromSPDF, 241, 241, 242
spatialIntensities.fromXYZ, 241, 242, 242
spatialparsEst, 94, 120, 122, 123, 133, 136, 145, 147, 185, 243, 263
SpatialPolygonsDataFrame.stppp, 244
SpikedExponentialCovFct, 49, 50, 60, 174, 175, 217, 245
stapp, 245
stapp.list, 246
stapp.SpatialPolygonsDataFrame, 246
stGPrealisation, 247
stppp, 156, 248, 248, 249
stppp.list, 248, 248, 249
stppp.ppp, 248, 249, 249
summary.lgcpgrid, 24, 109, 126–128, 194, 209, 215, 250
summary.mcmc, 250

target.spatialPlusPars, 291

INDEX
INDEX

temporalAtRisk.numeric, 44–46, 200, 212, 257–259, 259, 260
tempRaster, 261
textsummary, 33, 35, 43, 55, 139, 142, 150, 153, 163, 191, 192, 204–207, 213, 261, 265
thetaEst, 51, 94, 120–123, 133, 136, 145, 147, 185, 244, 262
toral.cov.mat, 263
touchingowin, 264
traceplots, 33, 35, 43, 55, 139, 142, 150, 153, 163, 191, 192, 204–207, 213, 262, 264
transblack, 265
transblue, 265
transgreen, 266
transred, 266
txtProgressBar2, 267
updateAMCMC, 267
updateAMCMC.andrieuthomsh, 268, 268
updateAMCMC.constanth, 268, 269
varfield, 269
varfield.lgcpPredict, 133, 136, 145, 147, 270
varfield.lgcpPredictINLA, 270
window.lgcpPredict, 133, 136, 145, 147, 271
wpopdata, 271
wtowncoords, 272
wtowns, 272
xvals, 273, 274–281
xvals.default, 273, 273, 274–281
xvals.fromXYZ, 273, 274, 274, 275–281
xvals.lgcpPredict, 133, 136, 145, 147, 274
xvals.SpatialGridDataFrame, 273, 274, 275, 276–281
YfromGamma, 276
yvals, 273–275, 276, 277–281
yvals.default, 273–276, 277, 278–281
yvals.fromXYZ, 273–277, 277, 279–281
yvals.lgcpPredict, 133, 136, 145, 147, 278
yvals.SpatialGridDataFrame, 273–278, 278, 279–281
zvals, 273–279, 279, 280, 281