Package 'INLA'

June 20, 2017

Type Package

Title Functions which Allow to Perform Full Bayesian Analysis of Latent Gaussian Models using Integrated Nested Laplace Approximations

Author Havard Rue, Sara Martino, Finn Lindgren, Daniel Simpson, Andrea Riebler, Elias Teixeira Krainski and Geir-Arne Fuglstad

Maintainer Havard Rue <hrue@r-inla.org>, Finn Lindgren <finn.lindgren@gmail.com>, Daniel Simpson <dp.simpson@gmail.com>, Andrea Riebler <andrea.riebler@math.ntnu.no>, Elias Teixeira Krainski <elias.krainski@math.ntnu.no> and Geir-Arne Fuglstad <fulgstad@math.ntnu.no>

Description This package contains functions which allow to perform full Bayesian analysis of latent Gaussian models using Integrated Nested Laplace Approximaxion, and is a front-end to the inla-program.

Depends sp, Matrix

Suggests mvtnorm, numDeriv, Rgraphviz, graph, fields, rgl, parallel, pixmap, splancs, orthopolynom, compiler, devtools, knitr, markdown, rmarkdown, shiny, rgdal, Deriv, HKprocess, FGN, MatrixModels

Imports stats, graphics, grDevices, utils, methods, splines

VignetteBuilder knitr

BuildVignettes true

ByteCompile no

LazyData true

License GPL (>= 2)

Version 17.06.20

Date 2017-06-20 (Version_17.06.20)

R topics documented:

`INLA-package` *Integrated Nested Laplace Approximation*

Description

Package to perform full Bayesian analysis on generalised additive mixed models using Integrated Nested Laplace Approximations.

Details

Package:	INLA
Type:	Package
Version:	0.0
Date:	TODAY
License:	GPL2
NOTE:	This package has no version number yet; it is to heavily developed at the moment; see `bitbucket.org/hrue/`

See the web-site *www.r-inla.org* for further details.

Author(s)

Havard Rue, Sara Martino, Finn Lindgren, Daniel Simpson and Andrea Riebler

---

as.inla.mesh.segment *Convert* sp *curve objects to* inla.mesh.segment *objects.*

Description

Convert sp curve objects to inla.mesh.segment objects.

Usage

```
as.inla.mesh.segment(sp, ...)
inla.sp2segment(sp, ...) ## For backwards compatibility

## S3 method for class 'Line'
as.inla.mesh.segment(sp, reverse=FALSE, crs=NULL, ...)
## S3 method for class 'Lines'
as.inla.mesh.segment(sp, join=TRUE, crs=NULL, ...)
## S3 method for class 'SpatialLines'
as.inla.mesh.segment(sp, join=TRUE, grp=NULL, ...)
## S3 method for class 'SpatialLinesDataFrame'
as.inla.mesh.segment(sp, ...)
## S3 method for class 'Polygon'
as.inla.mesh.segment(sp, crs=NULL, ...)
## S3 method for class 'Polygons'
as.inla.mesh.segment(sp, join=TRUE, crs=NULL, ...)
## S3 method for class 'SpatialPolygons'
as.inla.mesh.segment(sp, join=TRUE, grp=NULL, ...)
## S3 method for class 'SpatialPolygonsDataFrame'
as.inla.mesh.segment(sp, ...)
```

Arguments

sp	An sp polygon object of class Polygon, Polygons, SpatialPolygons, or SpatialPolygonsDataFra
join	If TRUE, join multiple polygons into a single segment (possibly non-simply connected).
grp	Group ID specification for each polygon, as used by inla.mesh.segment, one ID per polygon.
reverse	Logical, indicating if the line sequence should be traversed backwards.
crs	An optional CRS or inla.CRS object
...	Additional arguments passed on to other methods.

Value

A inla.mesh.segment object, or a list of inla.mesh.segment objects.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.mesh.segment

`BivMetaAnalysis` *Bivariate Meta Analysis*

Description

Data are taken from a meta-analysis to compare the utility of three types of diagnostic imaging - lymphangiography (LAG), computed tomography (CT) and magnetic resonance (MR) - to detect lymph node metastases in patients with cervical cancer. The dataset consists of a total of 46 studies: the first 17 for LAG, the following 19 for CT and the last 10 for MR.

Usage

```
data(BivMetaAnalysis)
```

Format

A data frame with 92 observations on the following 9 variables.

`N` a numeric vector

`Y` a numeric vector

`diid` a numeric vector

`lag.tp` a numeric vector

`lag.tn` a numeric vector

`ct.tp` a numeric vector

`ct.tn` a numeric vector

`mr.tp` a numeric vector

`mr.tn` a numeric vector

References

J. Scheidler and H. Hricak and K. K. Yu and L. Subak and M. R. Segal,"Radiological evaluation of lymph node metastases in patients with cervical cancer: a meta-analysis",JAMA 1997

Examples

```
data(BivMetaAnalysis)
```

`Cancer` *~~ data name/kind ... ~~*

Description

~~ A concise (1-5 lines) description of the dataset. ~~

Usage

```
data(Cancer)
```

Format

A data frame with 6690 observations on the following 4 variables.

- `Y` Number of cases
- `N` a numeric vector
- `Age` a numeric vector
- `region` a numeric vector

References

Rue, H and Held, L. (2005) *Gaussian Markov Random Fields - Theory and Applications* Chapman and Hall

---

`contrib.sd` *Computes the standard deviation for the structured (random) effects in an INLA model*

Description

Computes the posterior distribution of the standard deviations for the structured (random) effects in an INLA model, starting from the default output based on the precisions

Usage

```
inla.contrib.sd(model, nsamples=1000)
```

Arguments

`model`	An INLA model, fitted calling the `inla()`-function. The formula specified for the model should include at least one structured (random) effect in the form `f(variable, model="iid")`.
`nsamples`	The number of simulations from the posterior distribution of the standard deviations used to compute the summary statistics

Value

`inla.contrib.sd` returns a matrix `samples` including the simulated values from the posterior distributions as well as a summary table `hyper` reporting the mean, sd and 95% credible interval for the posterior distributions of each random effect.

Author(s)

Gianluca Baio <gianluca@stats.ucl.ac.uk>

See Also

inla

Examples

```
# Data generation
n=12
Ntrials = sample(c(80:100), size=n, replace=TRUE)
eta = rnorm(n,0,0.5)
prob = exp(eta)/(1 + exp(eta))
y = rbinom(n, size=Ntrials, prob = prob)
data=data.frame(y=y,z=1:n)
formula=y~f(z,model="iid")
m=inla(formula,data=data,family="binomial",Ntrials=Ntrials)
summary(m)
s=inla.contrib.sd(m)
s$hyper
hist(s$samples,xlab="standard deviation for z",main="")
```

---

control.compute *Control variables in control.compute*

Description

Control variables in `control.compute` for use in `inla`

Usage

```
inla.set.control.compute.default(...)
control.compute(config, cpo, dic, gdensity, graph, hyperpar, mlik, openmp.strategy, po, q, return.
```

Arguments

`...` Possible arguments

`openmp.strategy`
: The computational strategy to use: 'small', 'medium', 'large', 'huge' and 'default'. The difference is how the parallelisation is done, and is tuned for 'small'-sized models, 'medium'-sized models, etc. The default option tries to make an educated guess, but this allows to overide this selection. Default is 'default'

`hyperpar` A boolean variable if the marginal for the hyperparameters should be computed. Default TRUE.

`return.marginals`
: A boolean variable if the marginals for the latent field should be returned (although it is computed). Default TRUE

`dic` A boolean variable if the DIC-value should be computed. Default FALSE.

`mlik` A boolean variable if the marginal likelihood should be computed. Default FALSE.

`cpo`	A boolean variable if the cross-validated predictive measures (cpo, pit) should be computed
`po`	A boolean variable if the predictive ordinate should be computed
`waic`	A boolean variable if the Watanabe-Akaike information criteria should be computed
`q`	A boolean variable if binary images of the precision matrix, the reordered precision matrix and the Cholesky triangle should be generated. (Default FALSE.)
`config`	A boolean variable if the internal GMRF approximations be stored. (Default FALSE. EXPERIMENTAL)
`smtp`	The sparse-matrix solver, one of 'smtp' (default) or 'band'
`graph`	A boolean variable if the graph itself should be returned. (Default FALSE.)
`gdensity`	A boolean variable if the Gaussian-densities itself should be returned. (Default FALSE.)

Value

The function `control.compute` is used to TAB-complete arguments and returns a list of given arguments. The function `inla.set.control.compute.default` returns a list with all the default values of all parameters within this control statement.

See Also

`control.update`, `control.lincomb`, `control.group`, `control.mix`, `control.link`, `control.expert`, `control.compute`, `control.family`, `control.fixed`, `control.inla`, `control.predictor`, `control.results`, `control.mode`, `control.hazard`, `inla`

`control.expert` *Control variables in control.expert*

Description

Control variables in `control.expert` for use in `inla`

Usage

```
inla.set.control.expert.default(...)
control.expert(cpo.idx, cpo.manual, disable.gaussian.check, jp.func, jp.RData, jp.Rfile)
```

Arguments

`...`	Possible arguments
`cpo.manual`	A boolean variable to decide if the inla-program is to be runned in a manual-cpo-mode. (EXPERT OPTION: DO NOT USE)
`cpo.idx`	The index/indices of the data point(s) to remove. (EXPERT OPTION: DO NOT USE)
`disable.gaussian.check`	Disable the check for fast computations with a Gaussian likelihood and identity link

jp.RData	The R-data file that contains global variables to be used by jp.func
jp.Rfile	The R-file to be sourced to set up a joint prior for the hyperparameters to be evaluated by jp.func
jp.func	The R-function which returns the joint prior, to be defined in jp.Rfile

Value

The function control.expert is used to TAB-complete arguments and returns a list of given arguments. The function inla.set.control.expert.default returns a list with all the default values of all parameters within this control statement.

See Also

control.update, control.lincomb, control.group, control.mix, control.link, control.expert, control.compute, control.family, control.fixed, control.inla, control.predictor, control.results, control.mode, control.hazard, inla

---

control.family *Control variables in control.family*

Description

Control variables in control.family for use in inla

Usage

```
inla.set.control.family.default(...)
control.family(cenpoisson.I, control.link, control.mix, fixed, gev.scale.xi, hyper, initial, link
```

Arguments

...	Possible arguments
hyper	Definition of the hyperparameters
initial	(OBSOLETE!) Initial value for the hyperparameter(s) of the likelihood in the internal scale.
prior	(OBSOLETE!) The name of the prior distribution(s) for othe hyperparameter(s).
param	(OBSOLETE!) The parameters for the prior distribution
fixed	(OBSOLETE!) Boolean variable(s) to say if the hyperparameter(s) is fixed or random.
link	(OBSOLETE! Use control.link=list(model=) instead.) The link function to use.
sn.shape.max	Maximum value for the shape-parameter for Skew Normal observations
gev.scale.xi	The internal scaling of the shape-parameter for the GEV distribution. (default 0.01)
cenpoisson.I	The censoring interval for the censored Poisson
quantile	The quantile to be used for the quantile parameterised likelihoods (default 0.5)
variant	This variable is used to give options for various variants of the likelihood, like chosing different parameterisations for example. See the relevant likelihood documentations for options (does only apply to some likelihoods).
control.mix	See ?control.mix
control.link	See ?control.link

Value

The function `control.family` is used to TAB-complete arguments and returns a list of given arguments. The function `inla.set.control.family.default` returns a list with all the default values of all parameters within this control statement.

See Also

`control.update`, `control.lincomb`, `control.group`, `control.mix`, `control.link`, `control.expert`, `control.compute`, `control.family`, `control.fixed`, `control.inla`, `control.predictor`, `control.results`, `control.mode`, `control.hazard`, `inla`

`control.fixed` *Control variables in control.fixed*

Description

Control variables in `control.fixed` for use in `inla`

Usage

```
inla.set.control.fixed.default(...)
control.fixed(cdf, compute, correlation.matrix, expand.factor.strategy, mean, mean.intercept, pre
```

Arguments

`...` Possible arguments

`cdf` A list of values to compute the CDF for, for all fixed effects

`quantiles` A list of quantiles to compute for all fixed effects

`expand.factor.strategy`
The strategy used to expand factors into fixed effects based on their levels. The default strategy is us use the `model.matrix`-function for which NA's are not allowed (`expand.factor.strategy="model.matrix"`) and levels are possible removed. The alternative option (`expand.factor.strategy="inla"`) use an `inla`-spesific expansion which expand a factor into one fixed effects for each level, do allow for NA's and all levels are present in the model. In this case, factors MUST BE factors in the data.frame/list and NOT added as `.+factor(x1)+.` in the formula only.

`mean` Prior mean for all fixed effects except the intercept. Alternatively, a named list with specific means where name=default applies to unmatched names. For example `control.fixed=list(mean=list(a=1, b=2, default=0))` assign 'mean=1' to fixed effect 'a' , 'mean=2' to effect 'b' and 'mean=0' to all others.

`mean.intercept` Prior mean for the intercept

`prec` Default precision for all fixed effects except the intercept. Alternatively, a named list with specific means where name=default applies to unmatched names. For example `control.fixed=list(prec=list(a=1, b=2, default=0.01))` assign 'prec=1' to fixed effect 'a' , 'prec=2' to effect 'b' and 'prec=0.01' to all others.

`prec.intercept` Default precision the intercept (default 0.0) `compute` Compute marginals for the fixed effects ? (default TRUE)

`correlation.matrix`
Compute the posterior correlation matrix for all fixed effects? (default FALSE) OOPS: This option will set up appropriate linear combinations and the results are shown as the posterior correlation matrix of the linear combinations. This option will imply `control.inla=list(lincomb.derived.correlation.matrix=TRUE)`.

Value

The function `control.fixed` is used to TAB-complete arguments and returns a list of given arguments. The function `inla.set.control.fixed.default` returns a list with all the default values of all parameters within this control statement.

See Also

`control.update`, `control.lincomb`, `control.group`, `control.mix`, `control.link`, `control.expert`, `control.compute`, `control.family`, `control.fixed`, `control.inla`, `control.predictor`, `control.results`, `control.mode`, `control.hazard`, `inla`

`control.group` *Control variables in control.group*

Description

Control variables in `control.group` for use in `inla`

Usage

```
inla.set.control.group.default(...)
control.group(adjust.for.con.comp, cyclic, fixed, graph, hyper, initial, model, order, param, pric
```

Arguments

`...`	Possible arguments
`model`	Group model (one of 'exchangable', 'exchangablepos', 'ar1', 'ar', 'rw1', 'rw2', 'besag', or 'iid')
`order`	Defines the `order` of the model: for model `ar` this defines the order p, in AR(p). Not used for other models at the time being.
`cyclic`	Make the group model cyclic? (Only applies to models 'ar1', 'rw1' and 'rw2')
`graph`	The graph spesification (Only applies to model 'besag')
`scale.model`	Scale the intrinsic model (RW1, RW2, BESAG) so the generalized variance is 1. (Default `inla.getOption("scale.model.default")`.)
`adjust.for.con.comp`	Adjust for connected components when `scale.model=TRUE`?
`hyper`	Definition of the hyperparameter(s)
`initial`	(OBSOLETE!) The initial value for the group correlation or precision in the internal scale.
`fixed`	(OBSOLETE!) A boolean variable if the group correction or precision is assumed to be fixed or random.

prior	(OBSOLETE!) The name of the prior distribution for the group correlation or precision in the internal scale
param	(OBSOLETE!) Prior parameters

Value

The function control.group is used to TAB-complete arguments and returns a list of given arguments. The function inla.set.control.group.default returns a list with all the default values of all parameters within this control statement.

See Also

control.update, control.lincomb, control.group, control.mix, control.link, control.expert, control.compute, control.family, control.fixed, control.inla, control.predictor, control.results, control.mode, control.hazard, inla

---

control.hazard — *Control variables in control.hazard*

Description

Control variables in control.hazard for use in inla

Usage

```
inla.set.control.hazard.default(...)
control.hazard(constr, cutpoints, diagonal, fixed, hyper, initial, model, n.intervals, param, pric
```

Arguments

...	Possible arguments
model	The model for the baseline hazard model. One of 'rw1' or 'rw2'. (Default 'rw1'.)
hyper	The definition of the hyperparameters.
fixed	(OBSOLETE!) A boolean variable; is the precision for 'model' fixed? (Default FALSE.)
initial	(OBSOLETE!) The initial value for the precision.
prior	(OBSOLETE!) The prior distribution for the precision for 'model'
param	(OBSOLETE!) The parameters in the prior distribution
constr	A boolean variable; shall the 'model' be constrained to sum to zero?
diagonal	An extra constant added to the diagonal of the precision matrix
n.intervals	Number of intervals in the baseline hazard. (Default 15)
cutpoints	The cutpoints to use. If not specified the they are compute from 'n.intervals' and the maximum length of the interval. (Default NULL)
strata.name	The name of the stratefication variable for the baseline hazard in the data.frame
scale.model	Scale the baseline hazard model (RW1, RW2) so the generalized variance is 1. (Default inla.getOption("scale.model.default").)

Value

The function `control.hazard` is used to TAB-complete arguments and returns a list of given arguments. The function `inla.set.control.hazard.default` returns a list with all the default values of all parameters within this control statement.

See Also

`control.update`, `control.lincomb`, `control.group`, `control.mix`, `control.link`, `control.expert`, `control.compute`, `control.family`, `control.fixed`, `control.inla`, `control.predictor`, `control.results`, `control.mode`, `control.hazard`, `inla`

`control.inla` *Control variables in control.inla*

Description

Control variables in `control.inla` for use in `inla`

Usage

```
inla.set.control.inla.default(...)
control.inla(adapt.hessian.max.trials, adapt.hessian.mode, adapt.hessian.scale, adjust.weights, c
```

Arguments

`...`	Possible arguments
`strategy`	Character The strategy to use for the approximations; one of 'gaussian', 'simplified.laplace' (default) or 'laplace'
`int.strategy`	Character The integration strategy to use; one of 'auto' (default), 'ccd', 'grid', 'eb' (empirical bayes), 'user' or 'user.std'
`int.design`	Matrix Matrix of user-defined integration points and weights. Each row consists theta values and the integration weight. (EXPERIMENTAL!)
`interpolator`	Character The interpolator used to compute the marginals for the hyperparameters. One of 'auto', 'nearest', 'quadratic', 'weighted.distance', 'ccd', 'ccdintegrate', 'gridsum', 'gaussian'. Default is 'auto'.
`fast`	Logical If TRUE, then replace conditional modes in the Laplace approximation with conditional expectation (default TRUE)
`linear.correction`	Logical Default TRUE for the 'strategy = laplace' option.
`h`	Numerical The step-length for the gradient calculations for the hyperparameters. Default 0.01.
`dz`	Numerical The step-length in the standarised scale for the integration of the hyperparameters. Default 0.75.
`diff.logdens`	Numerical The difference of the log.density for the hyperpameters to stop numerical integration using int.strategy='grid'. Default 6.
`print.joint.hyper`	Logical If TRUE, the store also the joint distribution of the hyperparameters (without any costs). Default TRUE.

`force.diagonal` Logical If TRUE, then force the Hessian to be diagonal. (Default FALSE.)

`skip.configurations` Logical Skip configurations if the values at the main axis are to small. (Default TRUE.)

`mode.known` Logical If TRUE then no optimisation is done. (Default FALSE.)

`adjust.weights` Logical If TRUE then just more accurate integration weights. (Default TRUE.)

`tolerance` Numerical The tolerance for the optimisation of the hyperparameters. If set, this is the default value for for 'tolerance.f^(2/3)', 'tolerance.g' and 'tolerance.x'; see below.

`tolerance.f` Numerical The tolerance for the absolute change in the log posterior in the optimisation of the hyperparameters.

`tolerance.g` Numerical The tolerance for the absolute change in the gradient of the log posterior in the optimisation of the hyperparameters.

`tolerance.x` Numerical The tolerance for the change in the hyperparameters (root-mean-square) in the optimisation of the hyperparameters.

`restart` Numerical To improve the optimisation, the optimiser is restarted at the found optimum 'restart' number of times.

`optimiser` Character The optimiser to use; one of 'gsl', 'domin' or 'default'.

`verbose` Logical Run in verbose mode? (Default FALSE)

`reordering` Character Type of reordering to use. (EXPERT OPTION; one of "AUTO", "DEFAULT", "IDENTITY", "REVERSEIDENTITY", "BAND", "METIS", "GENMMD", "AMD", "MD", "MMD", "AMDBAR", "AMDC", "AMDBARC", or the output from `inla.qreordering`.)

`cpo.diff` Numerical Threshold to define when the cpo-calculations are inaccurate. (EXPERT OPTION.)

`npoints` Numerical Number of points to use in the 'stratey=laplace' approximation

`cutoff` Numerical The cutoff used in the 'stratey=laplace' approximation. (Smaller value is more accurate and more slow.)

`adapt.hessian.mode` Logical Should optimisation be continued if the Hessian estimate is void? (Default TRUE)

`adapt.hessian.max.trials` Numerical Number of steps in the adaptive Hessian optimisation

`adapt.hessian.scale` Numerical The scaling of the 'h' after each trial.

`huge` Logical If TRUE then try to do some of the internal parallisations differently. Hopefully this will be of benefite for 'HUGE' models. (Default FALSE.) [THIS OPTION IS OBSOLETE AND NOT USED!]

`step.len` Numerical The step-length used to compute numerical derivaties of the log-likelihood

`stencil` Numerical Number of points in the stencil used to compute the numerical derivaties of the log-likelihood (3, 5, 7 or 9).

`lincomb.derived.only` Logical If TRUE the only compute the marginals for the derived linear combinations and if FALSE, the and also the linear combinations to the graph (Default TRUE)

`lincomb.derived.correlation.matrix`
: Logical If TRUE compute also the correlations for the derived linear combinations, if FALSE do not (Default FALSE)

`diagonal`
: Numerical Expert use only! Add a this value on the diagonal of the joint precision matrix.

`numint.maxfeval`
: Numerical Maximum number of function evaluations in the the numerical integration for the hyperparameters. (Default 10000.)

`numint.relerr`
: Numerical Relative error requirement in the the numerical integration for the hyperparameters. (Default 1e-5)

`numint.abserr`
: Numerical Absolute error requirement in the the numerical integration for the hyperparameters. (Default 1e-6)

`cmin`
: Numerical The minimum value for the negative Hessian from the likelihood. Increasing this value will stabalise the optimisation but can introduce bias in some estimates unless `-Inf` is used. (Default -Inf)

`step.factor`
: Numerical The step factor in the Newton-Raphson algorithm saying how large step to take (Default 1.0)

`global.node.factor`
: Numerical The factor which defines the degree required (how many neighbors), as a fraction of n-1, that is required to be classified as a global node and numbered last (whatever the reordering routine says). Here, n, is the size of the graph. (Disabled if larger than 1.)

`global.node.degree`
: Numerical The degree required (number of neighbors) to be classified as a global node and numbered last (whatever the reordering routine says).

`stupid.search`
: Logical Enable or disable the stupid-search-algorithm, if the Hessian calculations reveals that the mode is not found. (Default `TRUE`.)

`stupid.search.max.iter`
: Numerical Maximum number of iterations allowed for the stupid-search-algorithm.

`stupid.search.factor`
: Numerical Factor (>=1) to increase the step-length with after each new interation.

`correct`
: Logical Add correction for the Laplace approximation.

`correct.factor`
: Numerical Factor used in adjusting the correction factor (default=10) if correct=TRUE

`correct.strategy`
: Character The strategy used to compute the correction; one of 'simplified.laplace' (default) or 'laplace'

`correct.verbose`
: Logical Be verbose when computing the correction?

Value

The function `control.inla` is used to TAB-complete arguments and returns a list of given arguments. The function `inla.set.control.inla.default` returns a list with all the default values of all parameters within this control statement.

See Also

control.update, control.lincomb, control.group, control.mix, control.link, control.expert, control.compute, control.family, control.fixed, control.inla, control.predictor, control.results, control.mode, control.hazard, inla

---

control.lincomb *Control variables in control.lincomb*

Description

Control variables in control.lincomb for use in inla

Usage

```
inla.set.control.lincomb.default(...)
control.lincomb(precision, verbose)
```

Arguments

...	Possible arguments
precision	The precision for the artificial tiny noise. Default 1e09.
verbose	Use verbose mode for linear combinations if verbose model is set globally. (Default TRUE)

Value

The function control.lincomb is used to TAB-complete arguments and returns a list of given arguments. The function inla.set.control.lincomb.default returns a list with all the default values of all parameters within this control statement.

See Also

control.update, control.lincomb, control.group, control.mix, control.link, control.expert, control.compute, control.family, control.fixed, control.inla, control.predictor, control.results, control.mode, control.hazard, inla

---

control.link *Control variables in control.link*

Description

Control variables in control.link for use in inla

Usage

```
inla.set.control.link.default(...)
control.link(fixed, hyper, initial, model, nq, order, param, prior, variant)
```

Arguments

`...`	Possible arguments
`model`	The name of the link function/model
`order`	The `order` of the link function, where the interpretation of `order` is model-dependent.
`variant`	The `variant` of the link function, where the interpretation of `variant` is model-dependent.
`nq`	Number of quadrature-points used to do the numerical integration
`hyper`	Definition of the hyperparameter(s) for the link model chosen
`initial`	(OBSOLETE!) The initial value(s) for the hyperparameter(s)
`fixed`	(OBSOLETE!) A boolean variable if hyperparmater(s) is/are fixed or random
`prior`	(OBSOLETE!) The name of the prior distribution(s) for the hyperparmater(s)
`param`	(OBSOLETE!) The parameters for the prior distribution(s) for the hyperparmater(s)

Value

The `control.link`-list is set within the corresponding `control.family`-list as the link is likelihood-familiy spesific. The function `control.link` is used to TAB-complete arguments and returns a list of given arguments. The function `inla.set.control.link.default` returns a list with all the default values of all parameters within this control statement.

See Also

`control.update`, `control.lincomb`, `control.group`, `control.mix`, `control.link`, `control.expert`, `control.compute`, `control.family`, `control.fixed`, `control.inla`, `control.predictor`, `control.results`, `control.mode`, `control.hazard`, `inla`

`control.mix` *Control variables in control.mix*

Description

Control variables in `control.mix` for use in `inla`

Usage

```
inla.set.control.mix.default(...)
control.mix(fixed, hyper, initial, model, param, prior)
```

Arguments

`...`	Possible arguments
`model`	The model for the random effect. Currently, only `model='gaussian'` is implemented
`hyper`	Definition of the hyperparameter(s) for the random effect model chosen
`initial`	(OBSOLETE!) The initial value(s) for the hyperparameter(s)

`fixed`	(OBSOLETE!) A boolean variable if hyperparmater(s) is/are fixed or random
`prior`	(OBSOLETE!) The name of the prior distribution(s) for the hyperparmater(s)
`param`	(OBSOLETE!) The parameters for the prior distribution(s) for the hyperparmater(s)

Value

The `control.mix` -list is set within the corresponding `control.family`-list a the mixture of the likelihood is likelihood spesific. (This option is EXPERIMENTAL.) The function `control.mix` is used to TAB-complete arguments and returns a list of given arguments. The function `inla.set.control.mix.default` returns a list with all the default values of all parameters within this control statement.

See Also

`control.update`, `control.lincomb`, `control.group`, `control.mix`, `control.link`, `control.expert`, `control.compute`, `control.family`, `control.fixed`, `control.inla`, `control.predictor`, `control.results`, `control.mode`, `control.hazard`, `inla`

`control.mode` *Control variables in control.mode*

Description

Control variables in `control.mode` for use in `inla`

Usage

```
inla.set.control.mode.default(...)
control.mode(fixed, restart, result, theta, x)
```

Arguments

`...`	Possible arguments
`result`	Prevous result from inla(). Use the theta- and x-mode from this run.
`theta`	The theta-mode/initial values for theta. This option has preference over result$mode$theta.
`x`	The x-mode/intitial values for x. This option has preference over result$mode$x.
`restart`	A boolean variable; should we restart the optimisation from this configuration or fix the mode at this configuration? (Default FALSE.)
`fixed`	A boolean variable. If TRUE then treat all thetas as known and fixed, and if FALSE then treat all thetas as unknown and random (default).

Value

The function `control.mode` is used to TAB-complete arguments and returns a list of given arguments. The function `inla.set.control.mode.default` returns a list with all the default values of all parameters within this control statement.

See Also

control.update, control.lincomb, control.group, control.mix, control.link, control.expert, control.compute, control.family, control.fixed, control.inla, control.predictor, control.results, control.mode, control.hazard, inla

---

control.predictor *Control variables in control.predictor*

Description

Control variables in `control.predictor` for use in `inla`

Usage

```
inla.set.control.predictor.default(...)
control.predictor(A, cdf, compute, cross, fixed, hyper, initial, link, param, precision, prior, qu
```

Arguments

`...`	Possible arguments
`hyper`	Definition of the hyperparameters.
`fixed`	(OBSOLETE!) If the precision for the artificial noise is fixed or not (defualt TRUE)
`prior`	(OBSOLETE!) The prior for the artificial noise
`param`	(OBSOLETE!) Prior parameters for the artificial noise
`initial`	(OBSOLETE!) The value of the log precision of the artificial noise
`compute`	A boolean variable; should the marginals for the linear predictor be computed? (Default FALSE.)
`cdf`	A list of values to compute the CDF for the linear predictor
`quantiles`	A list of quantiles to compute for the linear predictor
`cross`	Cross-sum-to-zero constraints with the linear predictor. All linear predictors with the same level of 'cross' are constrained to have sum zero. Use 'NA' for no contribution. 'Cross' has the same length as the linear predictor (including the 'A' matrix extention). (THIS IS AN EXPERIMENTAL OPTION, CHANGES MAY APPEAR.)
`A`	The observation matrix (matrix or Matrix::sparseMatrix).
`precision`	The precision for eta* - A*eta,
`link`	Define the family-connection for unobserved observations (`NA`). `link` is integer values which defines the family connection; `family[link[idx]]` unless `is.na(link[idx])` for which the identity-link is used. The `link`-argument only influence the `fitted.values` in the `result`-object. If `is.null(link)` (default) then the identity-link is used for all missing observations. If the length of `link` is 1, then this value is replicated with the length of the responce vector. If an element of the responce vector is `!NA` then the corresponding entry in `link` is not used (but must still be a legal value). Setting this variable implies `compute=TRUE`.

Value

The function `control.predictor` is used to TAB-complete arguments and returns a list of given arguments. The function `inla.set.control.predictor.default` returns a list with all the default values of all parameters within this control statement.

See Also

`control.update`, `control.lincomb`, `control.group`, `control.mix`, `control.link`, `control.expert`, `control.compute`, `control.family`, `control.fixed`, `control.inla`, `control.predictor`, `control.results`, `control.mode`, `control.hazard`, `inla`

`control.results` *Control variables in control.results*

Description

Control variables in `control.results` for use in `inla`

Usage

```
inla.set.control.results.default(...)
control.results(return.marginals.predictor, return.marginals.random)
```

Arguments

`...` Possible arguments

`return.marginals.random`
A boolean variable; read the marginals for the fterms? (Default TRUE)

`return.marginals.predictor`
A boolean variable; read the marginals for the linear predictor? (Default TRUE)

Value

The function `control.results` is used to TAB-complete arguments and returns a list of given arguments. The function `inla.set.control.results.default` returns a list with all the default values of all parameters within this control statement.

See Also

`control.update`, `control.lincomb`, `control.group`, `control.mix`, `control.link`, `control.expert`, `control.compute`, `control.family`, `control.fixed`, `control.inla`, `control.predictor`, `control.results`, `control.mode`, `control.hazard`, `inla` control.update *Control variables in control.update*

Description

Control variables in `control.update` for use in `inla`

Usage

```
inla.set.control.update.default(...)
control.update(result)
```

Arguments

`...`	Possible arguments
`result`	Update the joint posterior for the hyperparameters from result

Value

The function `control.update` is used to TAB-complete arguments and returns a list of given arguments. The function `inla.set.control.update.default` returns a list with all the default values of all parameters within this control statement.

See Also

`control.update`, `control.lincomb`, `control.group`, `control.mix`, `control.link`, `control.expert`, `control.compute`, `control.family`, `control.fixed`, `control.inla`, `control.predictor`, `control.results`, `control.mode`, `control.hazard`, `inla`

cut *Group-wise model criticism using node-splitting*

Description

This function performs group-wise, cross-validatory model assessment for an INLA model using so-called node-splitting (Marshall and Spiegelhalter, 2007; Presanis et al, 2013). The user inputs an object of class `inla` (i.e. a result of a call to `inla()`) as well as a variable name (`split.by`) specifying a grouping: Data points that share the same value of `split.by` are in the same group. The function then checks whether each group is an "outlier", or in conflict with the remaining groups, using the methodology described in Ferkingstad et al (2017). The result is a vector containing a p-value for each group, corresponding to a test for each group `i`, where the null hypothesis is that group `i` is consistent with the other groups except `i` (so a small p-value is evidence that the group is an "outlier"). See Ferkingstad et al (2017) for further details.

Usage

```
inla.cut(result, split.by, debug=FALSE)
```

Arguments

`result`	An object of class `inla`, i.e. a result of a call to `inla()`
`split.by`	The name of the variable to group by. Data points that have the same value of `split.by` are in the same group.
`debug`	Print debugging information if `TRUE`, default is `FALSE`

Value

A numeric vector of p-values, corresponding to a test for each group `i` where the null hypothesis is that group `i` is consistent with the other groups except `i`. A small p-value for a group indicates that the group is an "outlier" (in conflict with remaining groups).

This function is EXPERIMENTAL!!!

Author(s)

Egil Ferkingstad <egil@hi.is> and Havard Rue <hrue@r-inla.org>

References

Ferkingstad, E., Held, L. and Rue, H. (2017). Fast and accurate Bayesian model criticism and conflict diagnostics using R-INLA. Unpublished manuscript, to appear on arXiv

Marshall, E. C. and Spiegelhalter, D. J. (2007). Identifying outliers in Bayesian hierarchical models: a simulation-based approach. Bayesian Analysis, 2(2):409– 444.

Presanis, A. M., Ohlssen, D., Spiegelhalter, D. J., De Angelis, D., et al. (2013). Conflict diagnostics in directed acyclic graphs, with applications in Bayesian evidence synthesis. Statistical Science, 28(3):376–397.

Examples

```
## See Ferkingstad et al (2017).
```

---

debug.graph *Debug a graph-file*

Description

Debug a graph specification on file (ascii-mode only), by checking the specification along the way.

Usage

```
inla.debug.graph(graph.file)
```

Arguments

`graph.file`	The filename of the graph (ascii-mode)

Value

If an error is found, then an error message is shows, otherwise the graph-object returned by `inla.read.graph()` is returned.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla.read.graph

Examples

```
## Not run:
## cat("3\n 1 1 2n\ 2 1 1\n 3 4\n", file="g.dat")
## g = inla.debug.graph("g.dat")
## End(Not run)
```

Drivers *Time series with seasonal effect*

Description

Montly total of car drivers killed or several injuried in England from January 1969 to December 1984

NB: The last 12 lines of the data set have the first column set to `NULL` since these data where not observed but we want to predict them.

Usage

```
data(Drivers)
```

Format

A data frame with 204 observations on the following 4 variables.

`y` Number of deaths

`belt` Indicator of weather the belt was compulsory to use (1) or not (0)

`trend` time (in months)

`seasonal` time (in months)

References

Rue, H and Held, L. (2005) *Gaussian Markov Random Fields - Theory and Applications* Chapman and Hall

Examples

```
data(Drivers)
```

`Epil` *Repeated measures on Poisson counts*

Description

Seizure counts in a randomised trial of anti-convulsant therpay in epilepsy for 59 patients.

Usage

```
data(Epil)
```

Format

A data frame with 236 observations on the following 7 variables.

- `y` Number of seizures
- `Trt` indicator for the presence of treatment
- `Base` 8-week baseline seizure counts
- `Age` Age of the patient
- `V4` indicator variable for the 4th visit.
- `rand` a numeric vector
- `Ind` indicator for the specific patient

Source

WinBUGS/OpenBUGS Manual Examples Vol I

Examples

```
data(Epil)
```

`extract.groups` *Extract tagged boundary/internal segments.*

Description

Extract boundary or internal segments tagged by group id:s.

Usage

```
extract.groups(...)

## S3 method for class 'inla.mesh.segment'
extract.groups(
  segm, groups, groups.new = groups, ...)
```

Arguments

- `segm` — An `inla.mesh.segment` object.
- `groups` — The segment groups id:s to extract.
- `groups.new` — Optional vector of group id remapping; `groups[k]` in the input will be replaced by `groups.new[k]` in the output.
- `...` — Additional arguments, passed on to other methods.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

`inla.mesh.segment`

---

f — *Define general Gaussian models in the INLA formula*

Description

Function used for defining of smooth and spatial terms within `inla` model formulae. The function does not evaluate anything - it exists purely to help set up a model. The function specifies one smooth function in the linear predictor (see `inla.models`) as

$$w\ f(x)$$

Usage

```
f(...,
     model = "iid",
     copy=NULL,
     same.as = NULL,
     n=NULL,
     nrep = NULL,
     replicate = NULL,
     ngroup = NULL,
     group = NULL,
     control.group = inla.set.control.group.default(),
     hyper = NULL,
     initial=NULL,
     prior=NULL,
     param = NULL,
     fixed = NULL,
     season.length=NULL,
     constr = NULL,
     extraconstr=list(A=NULL, e=NULL),
     values=NULL,
     cyclic = NULL,
     diagonal = NULL,
     graph=NULL,
```

```
          graph.file=NULL,
          cdf=NULL,
          quantiles=NULL,
          Cmatrix=NULL,
          rankdef=NULL,
          Z = NULL,
          nrow = NULL,
          ncol = NULL,
          nu = NULL,
          bvalue = NULL,
          spde.prefix = NULL,
          spde2.prefix = NULL,
          spde2.transform = c("logit", "log", "identity"),
          spde3.prefix = NULL,
          spde3.transform = c("logit", "log", "identity"),
          mean.linear = inla.set.control.fixed.default()$mean,
          prec.linear = inla.set.control.fixed.default()$prec,
          compute = TRUE,
          of=NULL,
          precision = exp(14),
          range = NULL,
          adjust.for.con.comp = TRUE,
          order = NULL,
          scale = NULL,
          strata = NULL,
          rgeneric = NULL,
          scale.model = NULL,
        args.slm = list(rho.min = NULL, rho.max = NULL, X = NULL, W = NULL, Q.beta = NULL),
          correct = NULL,
          debug = FALSE)
```

Arguments

`...`	Name of the covariate and, possibly of the weights vector. NB: order counts!!!! The first specified term is the covariate and the second one is the vector of weights (which can be negative).
`model`	A string indicating the choosen model. The default is `iid`. See `names(inla.models()$latent)` for a list of possible alternatives.
`copy`	TODO
`same.as`	TODO
`n`	An optional argument which defines the dimension of the model if this is different from `length(sort(unique(covariate)))`
`nrep`	TODO
`replicate`	We need to write documentation here
`ngroup`	TODO
`group`	TODO
`control.group`	TODO
`hyper`	Spesification of the hyperparameter, fixed or random, initial values, priors and its parameters. See `?inla.models` for the list of hyparameters for each model and its default options.

`initial`	THIS OPTION IS OBSOLETE; use `hyper`!!! Vector indicating the starting values for the optimization algorithm. The length of the vector depends on the number of hyperparamters in the choosen `model`. If `fixed=T` the value at which the parameters are fixed is determines through `initial`. See `inla.models()$latent$'model name'` to have info about the choosen model.
`prior`	THIS OPTION IS OBSOLETE; use `hyper`!!! Prior distribution(s) for the hyperparameters of the !random model. The default value depends on the type of model, see !`www.r-inla.org` for a detailed description of the models. See `names(inla.models()$priors)` for possible prior choices
`param`	THIS OPTION IS OBSOLETE; use `hyper`!!! Vector indicating the parameters a and b of the prior distribution for the hyperparameters. The length of the vector depends on the choosen `model`. See `inla.models()$latent$'model name'` to have info about the choosen model.
`fixed`	THIS OPTION IS OBSOLETE; use `hyper`!!! Vector of boolean variables indicating wheater the hyperparameters of the model are fixed or random. The length of the vector depends on the choosen `model` See `inla.models()$latent$'model name'` to have info about the choosen model.
`season.length`	Lenght of the seasonal compoment (ONLY if `model="seasonal"`)
`constr`	A boolean variable indicating whater to set a sum to 0 constraint on the term. By default the sum to 0 constraint is imposed on all intrinsic models ("iid","rw1","rw1","besag", etc..).
`extraconstr`	This argument defines extra linear constraints. The argument is a list with two elements, a matrix `A` and a vector `e`, which defines the extra constraint `Ax = e`; for example `extraconstr = list(A = A, e=e)`. The number of columns of `A` must correspond to the length of this `f`-model. Note that this constraint comes additional to the sum-to-zero constraint defined if `constr = TRUE`.
`values`	An optional vector giving all values assumed by the covariate for which we want estimated the effect. It must be a numeric vector, a vector of factors or `NULL`.
`cyclic`	A boolean specifying wheather the model is cyclical. Only valid for "rw1" and "rw2" models, is cyclic=T then the sum to 0 constraint is removed. For the correct form of the grah file see *Martino and Rue (2008)*.
`diagonal`	An extra constant added to the diagonal of the precision matrix.
`graph`	Defines the graph-object either as a file with a graph-description, an `inla.graph`-object, or as a (sparse) symmetric matrix.
`graph.file`	THIS OPTION IS OBSOLETE AND REPLACED BY THE MORE GENERAL ARGUMENT `graph`. PLEASE CHANGE YOUR CODE. Name of the file containing the graph of the model; see `www.r-inla.org/faq`.
`cdf`	A vector of maximum 10 values between 0 and 1 $x(0), x(1), \ldots$. The function returns, for each posterior marginal the probabilities $$\text{Prob}(X < x(p))$$
`quantiles`	A vector of maximum 10 quantiles, $p(0), p(1), \ldots$ to compute for each posterior marginal. The function returns, for each posterior marginal, the values $x(0), x(1), \ldots$ such that $$\text{Prob}(X < x(p)) = p$$
`Cmatrix`	The specification of the precision matrix for the generic, generic3 or z models (up to a scaling constant). `Cmatrix` is either a (dense) matrix, a matrix created

using `Matrix::sparseMatrix()`, or a filename which stores the non-zero elements of `Cmatrix`, in three columns: `i`, `j` and `Qij`. In case of the generic3 model, it is a list of such specifications.

`rankdef` A number **defining** the rank deficiency of the model, with sum-to-zero constraint and possible extra-constraints taken into account. See details.

`Z` The matrix for the z-model

`nrow` Number of rows for 2d-models

`ncol` Number of columns for 2d-models

`nu` Smoothing parameter for the Matern2d-model, possible values are `c(0, 1, 2, 3)`

`bvalue` TODO

`spde.prefix` TODO

`spde2.prefix` TODO

`spde2.transform` TODO

`spde3.prefix` TODO

`spde3.transform` TODO

`mean.linear` Prior mean for the linear component, only used if `model="linear"`

`prec.linear` Prior precision for the linear component, only used if `model="linear"`

`compute` A boolean variable indicating wheather the marginal posterior distribution for the nodes in the `f()` model should be computed or not. This is usefull for large models where we are only interested in some posterior marginals.

`of` TODO

`precision` The precision for the artifical noise added when creating a copy of a model and others.

`range` A vector of size two giving the lower and upper range for the scaling parameter `beta` in the model `COPY`, `CLINEAR`, `MEC` and `MEB`. If `low = high` then the identity mapping is used.

`adjust.for.con.comp` If TRUE (default), adjust some of the models (currently: besag, bym, bym2 and besag2) if the number of connected components in graph is larger than 1. If FALSE, do nothing.

`order` Defines the `order` of the model: for model `ar` this defines the order p, in AR(p). Not used for other models at the time being.

`scale` A scaling vector. Its meaning depends on the model.

`strata` A stratum vector. It meaning depends on the model.

`rgeneric` A object of class `inla.rgeneric` which defines the model. (EXPERIMENTAL!)

`scale.model` Logical. If `TRUE` then scale the RW1 and RW2 and BESAG and BYM and BESAG2 and RW2D models so the their (generlized) variance is 1. Default value is `inla.getOption("scale.model.default")`

`args.slm` Required arguments to the model="slm"; see the documentation for further details.,

`correct` Add this model component to the list of variables to be used in the corrected Laplace approximation? If `NULL` use default choice, otherwise correct if `TRUE` and do not if `FALSE`. (This option is currently experimental.),

`debug` Enable local debug output

Details

There is no default value for `rankdef`, if it is not defined by the user then it is computed by the rank deficiency of the prior model (for the generic model, the default is zero), plus 1 for the sum-to-zero constraint if the prior model is proper, plus the number of extra constraints. **Oops:** This can be wrong, and then the user must define the `rankdef` explicitely.

Value

TODO

Author(s)

Havard Rue `<hrue@r-inla.org>`

See Also

`inla`, `hyperpar.inla`

`fgn` *Lookup coefficients in the 3-component AR(1) mixture representing FGN(H)*

Description

This function will lookup the coefficients in the 3-component AR(1) mixture representing FGN(H)

Usage

```
inla.fgn(H, K=3L)
```

Arguments

`H`	The Hurst coeffcient (0<H<1), or a vector of those
`K`	The number of components in representation, must be 3L or 4L

Value

A named matrix with 7 columns, where the first column is `H`, column 2, 3 and 4 are lag one correlations (or phi's), and column 5, 6 and 7 are the weights.

This function is EXPERIMENTAL!!!

Author(s)

Havard Rue `<hrue@r-inla.org>`

Examples

```
r = c(inla.fgn(0.7))
r_m = inla.fgn(seq(0.6, 0.8, by=0.01))
```

geobugs2inla *INLA utility functions*

Description

Various utility functions for INLA

Usage

```
inla.geobugs2inla(adj, num, graph.file="graph.dat")
```

Arguments

adj	A vector listing the ID numbers of the adjacent areas for each area. This is a sparse representation of the full adjacency matrix for the study region, and can be generated using the Adjacency Tool from the Map menu in GeoBUGS.
num	A vector of length N (the total number of areas) giving the number of neighbours n.i for each area.
graph.file	Name of the file of the new graph in the INLA format.

Value

The return value is the name of the graph-file created.

Note

These are all the same function, and the two different names are due to backward-compatibility

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla, inla.surv, hyperpar.inla

Germany *Disease Mapping*

Description

Cases of Oral cavity cancer in Germany from 1986-1990

Usage

```
data(Germany)
```

Format

A data frame with 544 observations on the following 4 variables.

`region` Region of Germany

`E` Fixed quantity which accounts for number of people in the district (offset)

`Y` Number of cases

`x` covariate measuring smoking consumption

References

Rue, H and Held, L. (2005) *Gaussian Markov Random Fields - Theory and Applications* Chapman and Hall

Examples

```
data(Germany)
```

---

graph2matrix — *Construct a neighbour-matrix from a* `graph`

Description

Construct a neighbour-matrix from a `graph` and disaply it

Usage

```
inla.graph2matrix(graph, ...)
inla.spy(graph, ..., reordering = NULL, factor = 1.0, max.dim = NULL)
```

Arguments

`graph`	An `inla.graph`-object, a (sparse) symmetric matrix, a filename containing the graph, or a list or collection of characters and/or numbers defining the graph.
`reordering`	A possible reordering. Typical the one obtained from a `inla`-call, `result$misc$reordering`, or the result of `inla.qreordering`.
`factor`	A scaling of the `inla.graph`-object to reduce the size.
`max.dim`	Maximum dimension of the `inla.graph`-object plotted; if `missing(factor)` and `max.dim` is set, then `factor` is computed automatically to give the given `max.dim`.
`...`	Additional arguments to `inla.read.graph()`

Value

`inla.graph2matrix` returns a sparse symmetric matrix where the non-zero pattern is defined by the `graph`. The `inla.spy` function, plots a binary image of a `graph`. The `reordering` argument is typically the reordering used by `inla`, found in `result$misc$reordering`.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla.read.graph, inla.qreordering

Examples

```
n = 50
Q = matrix(0, n, n)
idx = sample(1:n, 2*n, replace=TRUE)
Q[idx, idx] = 1
diag(Q) = 1
g = inla.read.graph(Q)
QQ = inla.graph2matrix(g)
inla.spy(QQ)
print(all.equal(as.matrix(Q), as.matrix(QQ)))

g.file = inla.write.graph(g)
inla.dev.new()
inla.spy(g.file)
inla.spy(g.file,  reordering = inla.qreordering(g))

g = inla.read.graph(g.file)
inla.dev.new()
inla.spy(g)

inla.dev.new()
inla.spy(3, 1, "1 2 2 1 1 3 0")
inla.dev.new()
inla.spy(3, 1, "1 2 2 1 1 3 0", reordering = 3:1)
```

---

idx *Convert indexes*

Description

Convert indexes given by to triplet '(idx, group, replicate)' to the (one-dimensional) index used in the grouped and replicated model

Usage

```
inla.idx(idx, n = max(idx),
         group = rep(1, length(idx)), ngroup = max(group),
         replicate = rep(1, length(idx)),  nrep = max(replicate))
```

Arguments

idx	The index within the basic model. (Legal values from '1' to 'n'.)
n	The length 'n' of the basic model.
group	The index within group. (Legal values from '1' to 'ngroup'.)
ngroup	Number of groups.
replicate	The index within replication. (Legal values from '1' to 'nrep'.)
nrep	Number of replications.

Value

inla.idx returns indexes in the range '1' to 'n*ngroup*nrep' representing where the triplet '(idx,group,replicate)' is stored internally in the full grouped and replicated model.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
##TODO
```

---

inla — *Bayesian analysis of structured additive models*

Description

inla performs a full Bayesian analysis of additive models using Integrated Nested Laplace approximation

Usage

```
inla(
    formula,
    family = "gaussian",
    contrasts = NULL,
    data,
    quantiles=c(0.025, 0.5, 0.975),
    E = NULL,
    offset=NULL,
    scale = NULL,
    weights = NULL,
    Ntrials = NULL,
    strata = NULL,
    link.covariates = NULL,
    verbose = FALSE,
    lincomb = NULL,
    control.compute = list(),
    control.predictor = list(),
    control.family = list(),
    control.inla = list(),
    control.results = list(),
    control.fixed = list(),
    control.mode = list(),
    control.expert = list(),
    control.hazard = list(),
    control.lincomb = list(),
    control.update = list(),
    only.hyperparam = FALSE,
    inla.call = inla.getOption("inla.call"),
    inla.arg = inla.getOption("inla.arg"),
```

```
num.threads = inla.getOption("num.threads"),
keep = inla.getOption("keep"),
working.directory = inla.getOption("working.directory"),
silent = inla.getOption("silent"),
debug = inla.getOption("debug"),
.parent.frame = parent.frame()
)
```

Arguments

`formula` A `inla` formula like `y ~1 + z + f(ind, model="iid")` + f(ind2, weights, model="ar1") This is much like the formula for a `glm` except that smooth or spatial terms can be added to the right hand side of the formula. See `f` for full details and the web site `www.r-inla.org` for several worked out examples. Each smooth or spatial term specified through `f` should correspond to separate column of the data frame `data`. The response variable, `y` can be a univariate response variable, a list or the output of the function `inla.surf` for survival analysis models.

`family` A string indicating the likelihood family. The default is `gaussian` with identity link. See `names(inla.models()$likelihood)` for a list of possible alternatives.

`contrasts` Optional contrasts for the fixed effects; see `?lm` or `?glm` for details.

`data` A data frame or list containing the variables in the model. The data frame MUST be provided

`quantiles` A vector of quantiles, $p(0), p(1), \dots$ to compute for each posterior marginal. The function returns, for each posterior marginal, the values $x(0), x(1), \dots$ such that

$$\text{Prob}(X < x(p)) = p$$

`E` Known component in the mean for the Poisson likelihoods defined as

$$E_i \exp(\eta_i)$$

where

$$\eta_i$$

is the linear predictor. If not provided it is set to `rep(1, n.data)`.

`offset` This argument is used to specify an a-priori known and fixed component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length either one or equal to the number of cases. One or more `offset()` terms can be included in the formula instead or as well, and if both are used, they are combined into a common offset. If the `A`-matrix is used in the linear predictor statement `control.predictor`, then the `offset` given in this argument is added to `eta*`, the linear predictor related to the observations, as `eta* = A eta + offset`, whereas an offset in the formula is added to `eta`, the linear predictor related to the formula, as `eta = ... + offset.formula`. So in this case, the offset defined here and in the formula has a different meaning and usage.

`scale` Fixed (optional) scale parameters of the precision for Gaussian and Student-T response models. Default value is rep(1, n.data).

`weights` Fixed (optional) weights parameters of the likelihood, so the log-likelihood[i] is changed into weights[i]*log-likelihood[i]. Default value is rep(1, n.data). Due to the danger of mis-interpreting the results (see below), this option is DISABLED by default. You can enable this option for the rest of your `R` session, doing `inla.setOption(enable.inla.argument.weights=TRUE)`. WARNING: The normalizing constant for the likelihood is NOT recomputed, so ALL marginals (and the marginal likelihood) must be interpreted with great care. Possibly, you may want to set the prior for the hyperparameters to `"uniform"` and the integration strategy to `"eb"` to mimic a maximum-likelihood approach.

`Ntrials` A vector containing the number of trials for the `binomial` likelihood. Default value is `rep(1, n.data)`.

`strata` Fixed (optional) strata indicators for tstrata likelihood model.

`link.covariates` A vector or matrix with covariates for link functions

`verbose` Boolean indicating if the `inla`-program should run in a verbose mode (default `FALSE`).

`lincomb` Used to define linear combination of nodes in the latent field. The posterior distribution of such linear combination is computed by the `inla` function. See `www.r-inla.org/faq` for examples of how to define such linear combinations.

`control.compute` See `?control.compute`

`control.predictor` See `?control.predictor`

`control.family` See `?control.family`

`control.inla` See `?control.inla`

`control.results` See `?control.result`

`control.fixed` See `?control.fixed`

`control.mode` See `?control.mode`

`control.expert` See `?control.expert`

`control.hazard` See `?control.hazard`

`control.lincomb` See `?control.lincomb`

`control.update` See `?control.update`

`only.hyperparam` A boolean variable saying if only the hyperparameters should be computed. This option is mainly used internally. (TODO: This option should not be located here, change it!)

`inla.call` The path to, or the name of, the `inla`-program. This is program is installed together with the `R`-package, but, for example, a native compiled version can be used instead to improve the performance.

`inla.arg` A string indicating ALL arguments to the 'inla' program and do not include default arguments. (OOPS: This is an expert option!)

`num.threads` Maximum number of threads the `inla`-program will use. xFor Windows this defaults to 1, otherwise its defaults to `NULL` (for which the system takes over control).

`keep` A boolean variable indicating that the working files (ini file, data files and results files) should be kept. If TRUE and no `working.directory` is specified the working files are stored in a directory called "inla".

`working.directory` A string giving the name of an non-existing directory where to store the working files.

`silent` If equal to 1L or TRUE, then the `inla`-program would be "silent". If equal to 2L, then supress also error messages from the `inla`-program.

`debug` If `TRUE`, then enable some debug output.

`.parent.frame` Internal use only

Value

`inla` returns an object of class `"inla"`. This is a list containing at least the following arguments:

`summary.fixed` Matrix containing the mean and standard deviation (plus, possibly quantiles and cdf) of the the fixed effects of the model.

`marginals.fixed` A list containing the posterior marginal densities of the fixed effects of the model.

`summary.random` List of matrices containing the mean and standard deviation (plus, possibly quantiles and cdf) of the the smooth or spatial effects defined through `f()`.

`marginals.random` If `return.marginals.random=TRUE` in `control.results` (default), a list containing the posterior marginal densities of the random effects defined through `f`.

`summary.hyperpar` A matrix containing the mean and sd (plus, possibly quantiles and cdf) of the hyperparameters of the model

`marginals.hyperpar` A list containing the posterior marginal densities of the hyperparameters of the model.

`summary.linear.predictor` A matrix containing the mean and sd (plus, possibly quantiles and cdf) of the linear predictors η in the model

`marginals.linear.predictor` If `compute=TRUE` in `control.predictor`, a list containing the posterior marginals of the linear predictors η in the model.

`summary.fitted.values` A matrix containing the mean and sd (plus, possibly quantiles and cdf) of the fitted values $g^{-1}(\eta)$ obtained by transforming the linear predictors by the inverse of the link function. This quantity is only computed if `marginals.fitted.values` is computed. Note that if an observation is `NA` then the identity link is used. You can manually transform a marginal using `inla.marginal.transform()` or set the argument `link` in the `control.predictor`-list; see `?control.predictor`

`marginals.fitted.values` If `compute=TRUE` in `control.predictor`, a list containing the posterior marginals of the fitted values $g^{-1}(\eta)$ obtained by transforming the linear predictors by the inverse of the link function. Note that if an observation is `NA` then the identity link is used. You can manually transform a marginal using `inla.marginal.transform()` or set the argument `link` in the `control.predictor`-list; see `?control.predictor`

summary.lincomb
: If lincomb != NULL a list of matrices containing the mean and sd (plus, possibly quantiles and cdf) of all linear combinations defined.

marginals.lincomb
: If lincomb != NULL a list of posterior marginals of all linear combinations defined.

joint.hyper
: A matrix containing the joint density of the hyperparameters (in the internal scale)

dic
: If dic=TRUE in control.compute, the deviance information criteria and effective number of parameters, otherwise NULL

cpo
: If cpo=TRUE in control.compute, a list of three elements: cpo$cpo are the values of the conditional predictive ordinate (CPO), cpo$pit are the values of the probability integral transform (PIT) and cpo$failure indicates whether some assumptions are violated. In short, if cpo$failure[i] > 0 then some assumption is violated, the higher the value (maximum 1) the more seriously.

po
: If po=TRUE in control.compute, a list of one elements: po$po are the values of the predictive ordinate (CPO) (pi(yi|y))

waic
: If waic=TRUE in control.compute, a list of two elements: waic$waic is the Watanabe-Akaike information criteria, and waic$p.eff is the estimated effective number of parameters

mlik
: If mlik=TRUE in control.compute, the log marginal likelihood of the model (using two different estimates), otherwise NULL

neffp
: Expected effective number of parameters in the model. The standard deviation of the expected number of parameters and the number of replicas for parameter are also returned

mode
: A list of two elements: mode$theta is the computed mode of the hyperparameters and mode$x is the mode of the latent field given the modal value of the hyperparamters.

call
: The matched call.

formula
: The formula supplied

nhyper
: The number of hyperparameters in the model

cpu.used
: The cpu time used by the inla function

Author(s)

Havard Rue <hrue@r-inla.org> and Sara Martino

References

Rue, H. and Martino, S. and Chopin, N. (2009) *Approximate Bayesian Inference for latent Gaussian models using Integrated Nested Laplace Approximations, JRSS-series B (with discussion)*, vol 71, no 2, pp 319-392. Rue, H and Held, L. (2005) *Gaussian Markov Random Fields - Theory and Applications* Chapman and Hall

See Also

f, inla.hyperpar

Examples

```
## Not run:
##See the web page \url{www.r-inla.org} for a series of worked out examples

## End(Not run)
```

---

inla.ar *Convert between parameterizations for the AR(p) model*

Description

These functions convert between the AR(p) coefficients phi, the partial autorcorrelation coefficients pacf and the autocorrelation function acf. The phi-parameterization is the same as used for arima-models in R; see ?arima and the parameter-vector a in Details.

Usage

```
inla.ar.pacf2phi(pac)
inla.ar.phi2pacf(phi)
inla.ar.pacf2acf(pac, lag.max = length(pac))
inla.ar.phi2acf(phi, lag.max = length(phi))
```

Arguments

pac	The partial autorcorrelation coefficients
phi	The AR(p) parameters phi
lag.max	The maximum lag to compute the ACF for

Value

inla.ar.pacf2phi returns phi for given pacf. inla.ar.phi2pacf returns pac for given phi. inla.ar.phi2acf returns acf for given phi. inla.ar.pacf2acf returns acf for given pacf.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
pac = runif(5)
phi = inla.ar.pacf2phi(pac)
pac2 = inla.ar.phi2pacf(phi)
print(paste("Error:", max(abs(pac2-pac))))
print("Correlation matrix (from pac)")
print(toeplitz(inla.ar.pacf2acf(pac)))
print("Correlation matrix (from phi)")
print(toeplitz(inla.ar.phi2acf(phi)))
```

inla.as.sparse *Convert a matrix or sparse matrix into the sparse formate used by INLA*

Description

Convert a matrix or sparse matrix into the sparse format used by INLA (dgTMatrix)

Usage

```
inla.as.sparse(...)
inla.as.dgTMatrix(A, unique = TRUE, na.rm = FALSE, zeros.rm = FALSE)
```

Arguments

`...`	The arguments. The matrix or sparse matrix, and the additonal arguments
`A`	The matrix
`unique`	Logical. If `TRUE`, then ensure that the internal representation is unique and there are no duplicated entries. (Do not change this unless you know what you are doing.)
`na.rm`	Replace `NA`'s in the matrix with zeros.
`zeros.rm`	Remove zeros in the matrix.

Value

`inla.as.sparse` and `inla.as.dgTMatrix` is the same function. The returned value is a sparse matrix in the `dgTMatrix`-format.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
A = matrix(1:9, 3, 3)
inla.as.sparse(A)
```

---

inla.changelog *inla.changelog*

Description

List the recent changes in the inla-program and its R-interface

Usage

```
inla.changelog()
```

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla

---

inla.collect.results *Collect results from a inla-call*

Description

inla.collect.results collect results from a inla-call

Usage

```
inla.collect.results(
             results.dir,
             control.results = inla.set.control.results.default(),
             debug=FALSE,
             only.hyperparam=FALSE,
             file.log = NULL,
             file.log2 = NULL)
```

Arguments

results.dir	The directory where the results of the inla run are stored
control.results	a list of parameters controlling the output of the function; see ?control.results
debug	Logical. If TRUE some debugging information are printed
only.hyperparam	Binary variable indicating wheather only the results for the hyperparameters should be collected
file.log	Character. The filename, if any, of the logfile for the internal calculations
file.log2	Character. The filename, if any, of the logfile2 for the internal calculations

Details

This function is mainly used inside inla to collect results after running the inla function. It can also be used to collect results into R after having runned a inla section outside R.

Value

The function returns an object of class "inla", see the help file for inla for details.

inla.compare.results *Compare INLA and MCMC results*

Description

A small utility to compare INLA and MCMC results (OBSOLETE)

Usage

```
inla.compare.results(dir.inla = NULL, dir.mcmc = NULL)
```

Arguments

dir.inla	The directory with the INLA results
dir.mcmc	The directory with the MCMC results

Value

Return nothing. This is an interactive function.

This function is OBSOLETE

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
## See demo("Tokyo-compare")
```

inla.coxph *Convert a Cox proportional hazard model into Poisson regression*

Description

Tools to convert a Cox proportional hazard model into Poisson regression

Usage

```
inla.coxph(formula, data, control.hazard = list(), debug=FALSE)
inla.rbind.data.frames(...)
```

Arguments

formula	The formula for the coxph model where the reponse must be a inla.surv-object.
data	All the data used in the formula, as a list.
control.hazard	Control the model for the baseline-hazard; see ?control.hazard.
debug	Print debug-information
...	Data.frames to be cbind-ed, padding with NA.

Value

`inla.coxph` returns a list of new expanded variables to be used in the `inla`-call. Note that element `data` and `data.list` needs to be merged into a `list` to be passed as the `data` argument. See the example for details. `inla.rbind.data.frames` returns the new data.frame.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
## How the cbind.data.frames works:
df1 = data.frame(x=1:2, y=2:3, z=3:4)
df2 = data.frame(x=3:4, yy=4:5, zz=5:6)
inla.rbind.data.frames(df1, df2)

## Standard example of how to convert a coxph into a Poisson regression
n = 1000
x = runif(n)
lambda = exp(1+x)
y = rexp(n, rate=lambda)
event = rep(1,n)
data = list(y=y, event=event, x=x)
y.surv = inla.surv(y, event)
intercept1 = rep(1, n)
p = inla.coxph(y.surv ~ -1 + intercept1 + x,
               list(y.surv = y.surv,  x=x, intercept1 = intercept1))

r = inla(p$formula,
        family = p$family,
        data=c(as.list(p$data), p$data.list),
        E = p$E)
summary(r)

## How to use this in a joint model
intercept2 = rep(1, n)
y = 1 + x + rnorm(n, sd=0.1)
df = data.frame(intercept2, x, y)

## new need to cbind the data.frames, and then add the list-part of
## the data
df.joint = c(as.list(inla.rbind.data.frames(p$data, df)), p$data.list)
df.joint$Y = cbind(df.joint$y..coxph, df.joint$y)

## merge the formulas, recall to add '-1' and to use the new joint
## reponse 'Y'
formula = update(p$formula, Y ~ intercept2 -1 + .)

rr = inla(formula,
        family = c(p$family, "gaussian"),
        data = df.joint,
        E = df.joint$E)
```

`inla.cpo` *Improved estimates for the CPO/PIT-values*

Description

Improve the estimates of the CPO/PIT-values be recomputing the model-fit by removing data-points.

Usage

```
inla.cpo(result,
         force = FALSE,
         verbose = TRUE,
         recompute.mode = TRUE)
```

Arguments

`result`	An object of class `inla`, ie a result of a call to `inla()`
`force`	If TRUE, then recompute all CPO/PIT values and not just those with `result$cpo$failure > 0`.
`verbose`	Run in verbose mode?
`recompute.mode`	Should be mode (and the integration points) be recomputed when a data-point is removed or not?

Value

The object returned is the same as `result` but the new improved estimates of the CPO/PIT values replaced.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

`inla`

Examples

```
n = 10
y = rnorm(n)
r = inla(y ~ 1, data = data.frame(y), control.compute = list(cpo=TRUE))

rr = inla.cpo(r, force=TRUE)
```

inla.CRS *Create a coordinate reference system object*

Description

Creates either a CRS object or an inla.CRS object, describing a coordinate reference system

Usage

```
inla.CRS(projargs = NA_character_, doCheckCRSArgs = TRUE,
         args = NULL, oblique = NULL, ...)
```

Arguments

projargs	Either 1) a projection argument string suitable as input to sp::CRS, or 2) an existing CRS object, or 3) a shortcut reference string to a predefined projection (longlat, lambert, mollweide, hammer, and sphere).
doCheckCRSArgs	default TRUE, must be set to FALSE by package developers including CRS in an S4 class definition to avoid uncontrolable loading of the rgdal namespace.
args	An optional list of name/value pairs to add to and/or override the PROJ4 arguments in projargs. name=value is converted to "+name=value", and name=NA is converted to "+name".
oblique	Vector of length at most 4 of rotation angles (in degrees) for an oblique projection, all values defaulting to zero. The values indicate (longitude, latitude, orientation, orbit), as explained in the Details section below.
...	Additional parameters. Not currently in use.

Details

The first two elements of the oblique vector are the (longitude, latitude) coordinates for the oblique centre point. The third value (orientation) is a counterclockwise rotation angle for an observer looking at the centre point from outside the sphere. The fourth value is the quasi-longitude (orbit angle) for a rotation along the oblique observers equator.

Simple oblique: oblique=c(0, 45)

Polar: oblique=c(0, 90)

Quasi-transversal: oblique=c(0, 0, 90)

Satellite orbit viewpoint: oblique=c(lon0-time*v1, 0, orbitangle, orbit0+time*v2), where lon0 is the longitude at which a satellite orbit crosses the equator at time=0, when the satellite is at an angle orbit0 further along in its orbit. The orbital angle relative to the equatorial plane is orbitangle, and v1 and v2 are the angular velocities of the planet and the satellite, respectively. Note that "forward" from the satellite's point of view is "to the right" in the projection.

When oblique[2] or oblique[3] are non-zero, the resulting projection is only correct for perfect spheres.

Value

Either an sp::CRS object or an inla.CRS object, depending on if the coordinate reference system described by the parameters can be expressed with a pure sp::CRS object or not.

An S3 inla.CRS object is a list, usually (but not necessarily) containing at least one element:

crs The basic sp::CRS object

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

CRS, inla.CRSargs, plot.CRS, inla.identical.CRS

Examples

```
if (require(rgdal)) {
  halfroot <- "+a=0.7071067811865476 +b=0.7071067811865476"
  crs1 <- inla.CRS("+proj=longlat +ellps=sphere +a=1 +b=1")
  crs2 <- inla.CRS("+proj=cea +ellps=sphere +lat_ts=0 +units=m +a=1 +b=1")
  crs3 <- inla.CRS(paste("+proj=moll +ellps=sphere +units=m", halfroot))
  crs4 <- inla.CRS(paste("+proj=hammer +ellps=sphere +units=m", halfroot))
  crs5 <- inla.CRS("+proj=geocent +ellps=sphere +a=1 +b=1 +units=m")
  ## Shortcuts:
  crs1 <- inla.CRS("longlat")
  crs2 <- inla.CRS("lambert")
  crs3 <- inla.CRS("mollweide")
  crs4 <- inla.CRS("hammer")
  crs5 <- inla.CRS("sphere")
}
```

---

inla.CRSargs *Show expanded CRS arguments*

Description

Wrapper for sp::CRS and inla.CRS objects to extract the coordinate reference system argument string.

Usage

```
inla.CRSargs(x, ...)
inla.as.list.CRS(x, ...)
inla.as.list.CRSargs(x, ...)
inla.as.CRS.list(x, ...)
inla.as.CRSargs.list(x, ...)
```

Arguments

x An sp::CRS or inla.CRS object (for inla.CRSargs and inla.as.list.CRS), a character string (for inla.as.list.CRSargs), or a list (for inla.as.CRS.list and inla.as.CRSargs.list).

... Additional arguments passed on to other methods.

Value

For `inla.CRSargs` and `inla.as.CRSargs.list`, a character string with PROJ.4 arguments.

For `inla.as.list.CRS` and `inla.as.list.CRSargs`, a list of name/value pairs.

For `inla.as.CRS.list`, a `CRS` or `inla.CRS` object.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

`CRSargs`, `inla.CRS`

Examples

```
if (require(rgdal)) {
  crs0 <- inla.CRS("longlat")
  p4s <- inla.CRSargs(crs0)
  lst <- inla.as.list.CRSargs(p4s)
  crs1 <- inla.as.CRS.list(lst)
  lst$a <- 2
  crs2 <- inla.CRS(p4s, args=lst)
  print(inla.CRSargs(crs0))
  print(inla.CRSargs(crs1))
  print(inla.CRSargs(crs2))
}
```

`inla.dev.new` *Opens a new device*

Description

Open a new device using `dev.new` unless using RStudio

Usage

```
inla.dev.new(...)
```

Arguments

`...`	Optional arguments to `dev.new`

Value

The value of `dev.new` if not running RStudio, otherwise `NULL`

Author(s)

Havard Rue <hrue@r-inla.org>

inla.diameter	*Diameter of a point set*

Description

Find an upper bound to the convex hull of a point set

Usage

```
inla.diameter(x, ...)

## Default S3 method:
inla.diameter(x, manifold="", ...)
## S3 method for class 'inla.mesh'
inla.diameter(x, ...)
## S3 method for class 'inla.mesh.segment'
inla.diameter(x, ...)
## S3 method for class 'inla.mesh.lattice'
inla.diameter(x, ...)
## S3 method for class 'inla.mesh.1d'
inla.diameter(x, ...)
```

Arguments

x	A point set as an $n \times d$ matrix, or an `inla.mesh` related object.
manifold	Character string specifying the manifold type. Default is to treat the point set with Euclidean R^d metrics. Use `manifold="S2"` for great circle distances on the unit sphere (this is set automatically for `inla.mesh` objects).
...	Additional parameters passed on to other methods.

Value

A scalar, upper bound for the diameter of the convex hull of the point set.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

Examples

```
inla.diameter(matrix(c(0,1,1,0, 0,0,1,1), 4, 2))
```

inla.doc — *View documentation*

Description

View documentation of latent, prior and likelihood models.

Usage

```
inla.doc(what, sec, verbose=FALSE)
```

Arguments

what	What to view documentation about; name of latent model, name of prior, etc. (A regular expression.)
sec	An optional section to look for the documentation. If missing, all sections are used.
verbose	Logical if TRUE then run in verbose mode

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

www.r-inla.org

Examples

```
## Not run: inla.doc("rw2")
## Not run: inla.doc("gaussian")
```

inla.extract.el — *Extract elements by matching name from container objects.*

Description

Extract elements by wildcard name matching from a data.frame, list, or matrix.

Usage

```
inla.extract.el(M, ...)

## S3 method for class 'data.frame'
inla.extract.el(M, match, by.row = TRUE, ...)
## S3 method for class 'list'
inla.extract.el(M, match, ...)
## S3 method for class 'matrix'
inla.extract.el(M, match, by.row = TRUE, ...)
```

Arguments

`M`	A container object.
`match`	A regex defining the matching criterion.
`by.row`	If `TRUE`, extract data by row, otherwise by column.
`...`	Additional arguments, not used.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

`inla.fmesher.smorg` *Compute various mesh related quantities.*

Description

Low level function for computing finite element matrices, spherical harmonics, B-splines, and point mappings with barycentric triangle coordinates.

Usage

```
inla.fmesher.smorg(loc, tv,
                   fem = NULL, aniso = NULL,
                   gradients = FALSE,
                   sph0 = NULL, sph = NULL, bspline = NULL,
                   points2mesh = NULL,
                   splitlines = NULL,
                   output = NULL,
                   keep = FALSE)
```

Arguments

`loc`	3-column triangle vertex coordinate matrix.
`tv`	3-column triangle vertex index matrix.
`fem`	Maximum finite element matrix order to be computed.
`aniso`	A two-element list with γ and v for an anisotropic operator $\nabla \cdot H\nabla$, where $H = \gamma I + vv^\top$
`gradients`	When `TRUE`, calculate derivative operator matrices `dx`, dy, and dz.
`sph0`	Maximal order of rotationally invariant spherical harmonics.
`sph`	Maximal order of general spherical harmonics.
`bspline`	Rotationally invariant B-splines on a sphere. 3-vector with number of basis functions `n`, basis degree `degree`, and a logical; `TRUE` uniform knot angles, `FALSE` for uniform spacing in $\sin(latitude)$.
`points2mesh`	3-column matrix with points to be located in the mesh.
`splitlines`	A list with elements `loc` (3-column coordinate matrix) and `idx` (2-column index matrix) describing line segments that are to be split into sub-segments at triangle boundaries.
`output`	Names of objects to be included in the output, if different from defaults.
`keep`	When `TRUE`, for debugging purposes keep the `fmesher` I/O files on disk.

Value

A list of generated named quantities.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

---

inla.generate.colors *Generate text RGB color specifications.*

Description

Generates a tex RGB color specification matrix based on a color palette.

Usage

```
inla.generate.colors(color,
                     color.axis = NULL,
                     color.n = 512,
                     color.palette = cm.colors,
                     color.truncate = FALSE,
                     alpha = NULL)
```

Arguments

color	character, matrix or vector
color.axis	The min/max limit values for the color mapping.
color.n	The number of colors to use in the color palette.
color.palette	A color palette function.
color.truncate	If TRUE, truncate the colors at the color axis limits.
alpha	Transparency/opaqueness values.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

---

inla.group *Group or cluster covariates*

Description

inla.group group or cluster covariates so to reduce the number of unique values

Usage

```
inla.group(x, n = 25, method = c("cut", "quantile"), idx.only = FALSE)
```

Arguments

x	The vector of covariates to group.
n	Number of classes or bins to group into.
method	Group either using bins with equal length intervals (method = "cut"), or equal distance in the 'probability' scale using the quantiles (method = "quantile").
idx.only	Option to return the index only and not the method.

Value

inla.group return the new grouped covariates where the classes are set to the median of all the covariates belonging to that group.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

f

Examples

```
## this gives groups 3 and 8
x = 1:10
x.group = inla.group(x, n = 2)

## this is the intended use, to reduce the number of unique values in
## the of first argument of f()
n = 100
x = rnorm(n)
y = x + rnorm(n)
result = inla(y ~ f(inla.group(x, n = 20), model = "iid"), data=data.frame(y=y,x=x))
```

---

inla.hyperpar — *Improved estimates for the hyperparameters*

Description

Improve the estimates of the posterior marginals for the hyperparameters of the model using the grid integration strategy.

Usage

```
inla.hyperpar(
        result,
        skip.configurations = TRUE,
        verbose = FALSE,
        dz = 0.75,
        diff.logdens = 15,
        h = NULL,
        restart = FALSE,
        quantiles = NULL,
        keep = FALSE)
```

Arguments

- `result` — An object of class `inla`, ie a result of a call to `inla()`
- `skip.configurations` — A boolean variable; skip configurations if the values at the main axis are to small. (Default TRUE)
- `verbose` — Boolean indicating wheather the inla program should run in a verbose mode.
- `dz` — Step length in the standardized scale used in the construction of the grid, default 0.75.
- `diff.logdens` — The difference of the log.density for the hyperpameters to stop numerical integration using int.strategy='grid'. Default 15
- `h` — The step-length for the gradient calculations for the hyperparameters. Default 0.01.
- `restart` — A boolean defining wheather the optimizer should start again to ind the mode or if it should use the mode contained in the `object`
- `quantiles` — A vector of quantiles, to compute for each posterior marginal.
- `keep` — A boolean variable indicating the working files (ini file, data files and results files) should be kept

Value

The object returned is the same as `object` but the estimates of the hyperparameters are replaced by improved estimates.

Note

This function might take a long time or if the number of hyperparameters in the model is large. If it complains and says `I cannot get enough memory`, try to increase the value of the argument `dz` or decrease `diff.logdens`.

Author(s)

Havard Rue `<hrue@r-inla.org>`

References

See the references in `inla`

See Also

`inla`

inla.hyperpar.sample *Produce samples from the approximated joint posterior for the hyperparameters*

Description

Produce samples from the approximated joint posterior for the hyperparameters

Usage

```
inla.hyperpar.sample(n, result, intern=FALSE, improve.marginals = FALSE)
```

Arguments

- `n` Integer. Number of samples required.
- `result` An `inla`-object, f.ex the output from an `inla`-call.
- `intern` Logical. If `TRUE` then produce samples in the internal scale for the hyperparmater, if `FALSE` then produce samples in the user-scale. (For example log-precision (intern) and precision (user-scale))
- `improve.marginals` Logical. If `TRUE`, then improve the samples taking into account possible better marginal estimates for the hyperparameters in `result`.

Value

A matrix where each sample is a row. The contents of the column is described in the rownames.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
n = 100
r = inla(y ~ 1 + f(idx), data = data.frame(y=rnorm(n), idx = 1:n))
ns = 500
x = inla.hyperpar.sample(ns, r)

rr = inla.hyperpar(r)
xx = inla.hyperpar.sample(ns, rr, improve.marginals=TRUE)
```

`inla.identical.CRS` *Test CRS and inla.CRS for equality*

Description

Wrapper for identical, optionally testing only the CRS part of two objects

Usage

```
inla.identical.CRS(crs0, crs1, crsonly = FALSE)
```

Arguments

`crs0`	A `CRS` or `inla.CRS` object.
`crs1`	A `CRS` or `inla.CRS` object.
`crsonly`	Logical. If `TRUE`, only the `CRS` part of a `inla.CRS` object is compared.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

`inla.CRS`

Examples

```
crs0 <- inla.CRS("longlat")
crs1 <- inla.CRS("longlat", oblique=c(0,90))
print(c(inla.identical.CRS(crs0, crs0),
        inla.identical.CRS(crs0, crs1),
        inla.identical.CRS(crs0, crs1, crsonly=TRUE)))
```

`inla.ks.plot` *Kolmogorov-Smirnov Test Plots*

Description

Illustrate a one-sample Kolmogorov-Smirnov test by plotting the empirical distribution deviation.

Usage

```
inla.ks.plot(x, y, diff=TRUE, ...)
```

Arguments

`x`	a numeric vector of data values.
`y`	a cumulative distribution function such as 'pnorm'.
`diff`	logical, indicating if the normalised difference should be plotted. If `FALSE`, the absolute distribution functions are plotted.
`...`	additional arguments for `ks.test`, ignored in the plotting. In particular, only two-sided tests are illustrated.

Details

In addition to the (normalised) empirical distribution deviation, lines for the K-S test statistic are drawn, as well as $\pm$ two standard deviations around the expectation under the null hypothesis.

Value

A list with class "htest", as generated by ks.test

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

ks.test

Examples

```
## Check for N(0,1) data
data = rowSums(matrix(runif(100*12)*2-1,100,12))/2
inla.ks.plot(data, pnorm)

## Not run:
## Check the goodness-of-fit of cross-validated predictions
result = inla(..., control.predictor=list(cpo=TRUE))
inla.ks.plot(result$pit, punif)

## End(Not run)
```

---

inla.list.models *List available model components, likelihoods, priors, etc*

Description

List available model components, likelihoods, priors, etc.

Usage

```
inla.list.models(section = names(inla.models()), ...)
```

Arguments

section	The section(s) to list, missing section will list all sections. names(inla.models()) lists available sections.
...	Additional argument to cat

Details

The list is cat'ed with ... arguments.

This function is EXPERIMENTAL.

Value

Nothing is returned

Author(s)

Havard Rue

Examples

```
## Not run:
inla.list.models("likelihood")
inla.list.models(c("prior", "group"))
inla.list.models(file=file("everything.txt"))

## End(Not run)
```

`inla.load` *Load or source a file*

Description

Load or source a file: (internal use)

Usage

```
inla.load(filename, debug = TRUE)
```

Arguments

`filename`	The name of the file to be loaded, alternatively, sourced.
`debug`	Logical. Turn on/off debug information.

Details

Try to `load` the file into the global environment, if that fail, try to `source` the file into the global environment.

Value

None

Author(s)

Havard Rue <hrue@r-inla.org>

inla.matern.cov *Numerical evaluation of Matern and related covariance functions.*

Description

Calculates covariance and correlation functions for Matern models and related oscillating SPDE models, on R^d and on the sphere, S^2.

Usage

```
inla.matern.cov(nu, kappa, x,
                d = 1,
                corr = FALSE,
                norm.corr = FALSE,
                theta,
                epsilon = 1e-08)

inla.matern.cov.s2(nu, kappa, x,
                   norm.corr = FALSE,
                   theta = 0)
```

Arguments

nu	The Matern smoothness parameter.
kappa	The spatial scale parameter.
x	Distance values.
d	Space dimension; the domain is R^d.
corr	If TRUE, calculate correlations, otherwise calculate covariances. Only used for pure Matern models (i.e. with $\theta = 0$).
norm.corr	If TRUE, normalise by the estimated variance, giving approximate correlations.
theta	Oscillation strength parameter.
epsilon	Tolerance for detecting points close to distance zero.

Details

On R^d, the models are *defined* by the spectral density given by

$$S(w) = \frac{1}{(2\pi)^d(\kappa^4 + 2\kappa^2\cos(\pi\theta)|w|^2 + |w|^4)^{(\nu+d/2)/2}}$$

On S^2, the models are *defined* by the spectral coefficients

$$S(k) = \frac{2k+1}{4\pi(\kappa^4 + 2\kappa^2\cos(\pi\theta)k(k+1) + k^2(k+1)^2)^{(\nu+1)/2}}$$

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

inla.mdata *Create an mdata-object for INLA*

Description

This defines an mdata-object for matrix valued response-families

Usage

```
inla.mdata(y, ...)
## S3 method for class 'inla.mdata'
print(x, ...)
is.inla.mdata(object)
as.inla.mdata(object)
```

Arguments

y	The response vector/matrix
...	Additional vectors/matrics of same length as y
object	Any R-object
x	An mdata object

Value

An object of class `inla.mdata`. There is method for `print`.

`is.inla.mdata` returns `TRUE` if `object` inherits from class `inla.mdata`, otherwise `FALSE`.

`as.inla.mdata` returns an object of class `inla.mdata`

Author(s)

Havard Rue

See Also

`inla`

inla.mesh.1d *Function space definition objects for 1D SPDE models.*

Description

Create a 1D mesh specification `inla.mesh.1d` object, that defines a function space for 1D SPDE models.

Usage

```
inla.mesh.1d(loc,
             interval = range(loc),
             boundary = NULL,
             degree = 1,
             free.clamped = FALSE,
             ...)
```

Arguments

`loc`	B-spline knot locations.
`interval`	Interval domain endpoints.
`boundary`	Boundary condition specification. Valid conditions are `c('neumann', 'dirichlet', 'free', 'cyc` Two separate values can be specified, one applied to each endpoint.
`degree`	The B-spline basis degree. Supported values are 0, 1, and 2.
`free.clamped`	If `TRUE`, for `'free'` boundaries, clamp the basis functions to the interval endpoints.
`...`	Additional option, currently unused.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

---

inla.mesh.1d.A *Mapping matrix for 1D meshes*

Description

Calculates barycentric coordinates and weight matrices for `inla.mesh.1d` objects.

Usage

```
inla.mesh.1d.A(mesh, loc, weights = NULL, derivatives = NULL,
               method = c("linear", "nearest", "quadratic"))

inla.mesh.1d.bary(mesh, loc, method = c("linear", "nearest"))
```

Arguments

`mesh`	An `inla.mesh.1d` object.
`loc`	Coordinate values.
`weights`	Weights to be applied to the `A` matrix rows.
`derivatives`	If `TRUE`, also compute derivative weight matrices `dA` and `d2A`.
`method`	Interpolation method. If not specified for `inla.mesh.1d.A` (recommended), it is determined by the `mesh` basis function properties.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

inla.mesh.2d *High-quality triangulations*

Description

Create a triangle mesh based on initial point locations, specified or automatic boundaries, and mesh quality parameters.

Usage

```
inla.mesh.2d(loc = NULL,
             loc.domain = NULL,
             offset = NULL,
             n = NULL,
             boundary = NULL,
             interior = NULL,
             max.edge,
             min.angle = NULL,
             cutoff = 1e-12,
             max.n.strict = NULL,
             max.n = NULL,
             plot.delay = NULL,
             crs = NULL)
```

Arguments

`loc`	Matrix of point locations to be used as initial triangulation nodes. Can alternatively be a `SpatialPoints` or `SpatialPointsDataFrame` object.
`loc.domain`	Matrix of point locations used to determine the domain extent. Can alternatively be a `SpatialPoints` or `SpatialPointsDataFrame` object.
`offset`	The automatic extension distance. One or two values, for an inner and an optional outer extension. If negative, interpreted as a factor relative to the approximate data diameter (default=-0.10???)
`n`	The number of initial nodes in the automatic extensions (default=16)
`boundary`	A list of one or two `inla.mesh.segment` objects describing domain boundaries.
`interior`	An `inla.mesh.segment` object describing desired interior edges.
`max.edge`	The largest allowed triangle edge length. One or two values.
`min.angle`	The smallest allowed triangle angle. One or two values. (Default=21)
`cutoff`	The minimum allowed distance between points. Point at most as far apart as this are replaced by a single vertex prior to the mesh refinement step.
`max.n.strict`	The maximum number of vertices allowed, overriding `min.angle` and `max.edge` (default=-1, meaning no limit). One or two values, where the second value gives the number of additional vertices allowed for the extension.
`max.n`	The maximum number of vertices allowed, overriding `max.edge` only (default=-1, meaning no limit). One or two values, where the second value gives the number of additional vertices allowed for the extension.

`plot.delay` On Linux (and Mac if appropriate X11 libraries are installed), specifying a non-negative numeric value activates a rudimentary plotting system in the underlying `fmesher` program, showing the triangulation algorithm at work, with waiting time factor `plot.delay` between each step.

On all systems, specifying any negative value activates displaying the result after each step of the multi-step domain extension algorithm.

`crs` An optional `CRS` or `inla.CRS` object

Value

An `inla.mesh` object.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

`inla.mesh.create`, `inla.delaunay`

Examples

```
loc = matrix(runif(10*2),10,2)

mesh = inla.mesh.2d(loc,
                    boundary=list(inla.nonconvex.hull(loc, 0.1, 0.15),
                                  inla.nonconvex.hull(loc, 0.2, 0.2)),
                    max.edge=c(0.05, 0.1))
plot(mesh)
```

`inla.mesh.basis` *Basis functions for inla.mesh*

Description

Calculate basis functions on a 1d or 2d `inla.mesh`

Usage

```
inla.mesh.basis(mesh,
                type="b.spline",
                n=3,
                degree=2,
                knot.placement="uniform.area",
                rot.inv=TRUE,
                boundary="free",
                free.clamped=TRUE,
                ...)
```

Arguments

mesh	An inla.mesh.1d or inla.mesh object.
type	b.spline (default) for B-spline basis functions, sph.harm for spherical harmonics (available opnly for meshes on the sphere)
n	For B-splines, the number of basis functions in each direction (for 1d meshes n must be a scalar, and for planar 2d meshes a 2-vector). For spherical harmonics, n is the maximal harmonic order.
degree	Degree of B-spline polynomials. See inla.mesh.1d.
knot.placement	For B-splines on the sphere, controls the latitudinal placements of knots. "uniform.area" (default) gives uniform spacing in sin(latitude), "uniform.latitude" gives uniform spacing in latitudes.
rot.inv	For spherical harmonics on a sphere, rot.inv=TRUE gives the rotationally invariant subset of basis functions.
boundary	Boundary specification, default is free boundaries. See inla.mesh.1d for more information.
free.clamped	If TRUE and boundary is "free", the boundary basis functions are clamped to 0/1 at the interval boundary by repeating the boundary knots.
...	

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.mesh.1d inla.mesh.2d

Examples

```
n = 100
loc = matrix(runif(n*2), n, 2)
mesh = inla.mesh.2d(loc, max.edge=0.05)
basis = inla.mesh.basis(mesh, n=c(4,5))

proj = inla.mesh.projector(mesh)
image(proj$x, proj$y, inla.mesh.project(proj, basis[,7]))

if (require(rgl)) {
  plot(mesh, rgl=TRUE, col=basis[,7], draw.edges=FALSE, draw.vertices=FALSE)
}
```

---

inla.mesh.boundary *Constraint segment extraction for inla.mesh*

Description

Constructs an list of inla.mesh.segment object from boundary or interior constraint information in an inla.mesh object.

Usage

```
inla.mesh.boundary(mesh, grp = NULL)
inla.mesh.interior(mesh, grp = NULL)
```

Arguments

mesh	An inla.mesh object.
grp	Group indices to extract. If NULL, all boundary/interior constrain groups are extracted.

Value

A list of inla.mesh.segment objects.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.mesh.segment, inla.mesh.create, inla.mesh.create.helper

Examples

```
loc = matrix(runif(100*2)*1000,100,2)
mesh = inla.mesh.create.helper(points.domain=loc, max.edge=c(50,500))
boundary = inla.mesh.boundary(mesh)
interior = inla.mesh.interior(mesh)
```

---

inla.mesh.create *Low level function for high-quality triangulations*

Description

Create a constrained refined Delaunay triangulation (CRDT) for a set of spatial locations.

Usage

```
inla.mesh.create(loc = NULL,
                 tv = NULL,
                 boundary = NULL, interior = NULL,
                 extend = (missing(tv) || is.null(tv)),
                 refine = FALSE,
                 lattice = NULL,
                 globe = NULL,
                 cutoff = 1e-12,
                 plot.delay = NULL,
                 data.dir,
                 keep = (!missing(data.dir) && !is.null(data.dir)),
                 timings = FALSE,
                 quality.spec = NULL,
```

```
                crs=NULL)

inla.delaunay(loc, ...)
```

Arguments

`loc`	Matrix of point locations. Can alternatively be a `SpatialPoints` or `SpatialPointsDataFrame` object.
`tv`	A triangle-vertex index matrix, specifying an existing triangulation.
`boundary`	A list of `inla.mesh.segment` objects, generated by `inla.mesh.segment`, specifying boundary constraint segments.
`interior`	A list of `inla.mesh.segment` objects, generated by `inla.mesh.segment`, specifying interior constraint segments.
`extend`	`logical` or `list` specifying whether to extend the data region, with parameters `n` the number of edges in the extended boundary (default=8) `offset` the extension distance. If negative, interpreted as a factor relative to the approximate data diameter (default=-0.10) Setting to `FALSE` is only useful in combination `lattice` or `boundary`.
`refine`	`logical` or `list` specifying whether to refine the triangulation, with parameters `min.angle` the minimum allowed interior angle in any triangle. The algorithm is guaranteed to converge for `min.angle` at most 21 (default=`21`) `max.edge` the maximum allowed edge length in any triangle. If negative, interpreted as a relative factor in an ad hoc formula depending on the data density (default=`Inf`) `max.n.strict` the maximum number of vertices allowed, overriding `min.angle` and `max.edge` (default=-1, meaning no limit) `max.n` the maximum number of vertices allowed, overriding `max.edge` only (default=-1, meaning no limit)
`lattice`	An `inla.mesh.lattice` object, generated by `inla.mesh.lattice`, specifying points on a regular lattice.
`globe`	Subdivision resolution for a semi-regular spherical triangulation with equidistant points along equidistant latitude bands.
`cutoff`	The minimum allowed distance between points. Point at most as far apart as this are replaced by a single vertex prior to the mesh refinement step.
`plot.delay`	On Linux (and Mac if appropriate X11 libraries are installed), specifying a numeric value activates a rudimentary plotting system in the underlying `fmesher` program, showing the triangulation algorithm at work.
`data.dir`	Where to store the `fmesher` data files. Defaults to `tempdir()` if `keep` is `FALSE`, otherwise `"inla.mesh.data"`.
`keep`	`TRUE` if the data files should be kept in `data.dir` or deleted afterwards. Defaults to true if `data.dir` is specified, otherwise false. Warning: If keep is false, `data.dir` and its contents will be deleted (unless set to `tempdir()`).
`timings`	If `TRUE`, obtain timings for the mesh construction.
`quality.spec`	List of vectors of per vertex `max.edge` target specification for each location in `loc`, `boundary/interior` (segm), and `lattice`. Only used if refining the mesh.
`crs`	An optional `CRS` or `inla.CRS` object
`...`	Optional parameters passed on to `inla.mesh.create`.

Details

inla.mesh.create generates triangular meshes on subsets of R^2 and S^2. Use the higher level wrapper function inla.mesh.2d for greater control over mesh resolution and coarser domain extensions.

inla.delaunay is a wrapper function for obtaining the convex hull of a point set and calling inla.mesh.create to generate the classical Delaunay tringulation.

Value

An inla.mesh object.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.mesh.2d, inla.mesh.1d, inla.mesh.segment, inla.mesh.lattice, inla.mesh.query

Examples

```
loc = matrix(runif(10*2),10,2)

mesh = inla.delaunay(loc)
plot(mesh)

mesh = inla.mesh.create(loc,
                        interior=inla.mesh.segment(idx=1:2),
                        extend=TRUE,
                        refine=list(max.edge=0.1))
plot(mesh)

loc2 = matrix(c(0,1,1,0, 0,0,1,1), 4, 2);
mesh2 = inla.mesh.create(loc=loc,
                         boundary=inla.mesh.segment(loc2),
                         interior=inla.mesh.segment(idx=1:2),
                         quality.spec=list(segm=0.2, loc=0.05),
                         refine=list(min.angle=26))
plot(mesh2)
```

---

inla.mesh.deriv *Directional derivative matrices for functions on meshes.*

Description

Calculates directional derivative matrices for functions on inla.mesh objects.

Usage

```
inla.mesh.deriv(mesh, loc)
```

Arguments

`mesh`	An `inla.mesh` object.
`loc`	Coordinates where the derivatives should be evaluated.

Value

`A`	The projection matrix, `u(loc_i)=sum_j A_ij w_i`
`dx, dy, dz`	Derivative weight matrices, `du/dx(loc_i)=sum_j dx_ij w_i`, etc.

Author(s)

Finn Lindgren <`finn.lindgren@gmail.com`>

`inla.mesh.fem` *Finite element matrices*

Description

Constructs finite element matrices for `inla.mesh` and `inla.mesh.1d` objects.

Usage

```
## 2D and 1D meshes
inla.mesh.fem(mesh, order = 2)

## 1D meshes, order 2 models only
inla.mesh.1d.fem(mesh)
```

Arguments

`mesh`	An `inla.mesh` or `inla.mesh.1d` object.
`order`	The model order.

Value

A list of sparse matrices based on basis functions `psi_i`:

`c0`	`c0[i,j] = < psi_i, 1 >`
`c1`	`c1[i,j] = < psi_i, psi_j >`
`g1`	`g1[i,j] = < grad psi_i, grad psi_j >`
`g2`	`g2 = g1 * c0^-1 * g1`
`gk`	`gk = g1 * (c0^-1 * g1)^(k-1)`, up to and including `k=order`

Author(s)

Finn Lindgren <`finn.lindgren@gmail.com`>

inla.mesh.lattice *Lattice grids for inla.mesh*

Description

Construct a lattice grid for inla.mesh

Usage

```
inla.mesh.lattice(x = seq(0, 1, length.out=2),
          y = seq(0, 1, length.out=2),
          z = NULL,
          dims = if (is.matrix(x)) {
                   dim(x)
                 } else {
                   c(length(x), length(y))
                 },
          units = NULL,
          crs = NULL)
```

Arguments

x	
y	
z	
dims	
units	One of c("default", "longlat", "longsinlat").
crs	An optional CRS or inla.CRS object

Value

An inla.mesh.lattice object.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.mesh

Examples

```
lattice = inla.mesh.lattice(seq(0, 1, length.out=17), seq(0, 1, length.out=10))

## Use the lattice "as-is", without refinement:
mesh = inla.mesh.create(lattice=lattice, boundary=lattice$segm)
mesh = inla.mesh.create(lattice=lattice, extend=FALSE)
plot(mesh)

## Refine the triangulation, with limits on triangle angles and edges:
```

```
mesh = inla.mesh.create(lattice=lattice,
                        refine=list(max.edge=0.08),
                        extend=FALSE)
plot(mesh)

## Add an extension around the lattice, but maintain the lattice edges:
mesh = inla.mesh.create(lattice=lattice,
                        refine=list(max.edge=0.08),
                        interior=lattice$segm)
plot(mesh)

## Only add extension:
mesh = inla.mesh.create(lattice=lattice, refine=list(max.edge=0.08))
plot(mesh)
```

`inla.mesh.map` *Coordinate mappings for* `inla.mesh` *projections.*

Description

Calculates coordinate mappings for `inla.mesh` projections.

Usage

```
inla.mesh.map(loc,
              projection = c("default", "longlat",
                             "longsinlat", "mollweide"),
              inverse = TRUE)

## Compute sensible default map axis limits
inla.mesh.map.lim(loc = NULL,
                  projection = c("default", "longlat",
                                 "longsinlat", "mollweide"))
```

Arguments

`loc`	Coordinates to be mapped.
`projection`	The projection type.
`inverse`	If `TRUE`, `loc` are map coordinates and coordinates in the mesh domain are calculated. If `FALSE`, `loc` are coordinates in the mesh domain and the forward map projection is calculated.

Value

For `inla.mesh.map.lim`, a list:

`xlim`	X axis limits in the map domain
`ylim`	Y axis limits in the map domain

No attempt is made to find minimal limits for partial spherical domains.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.mesh.project

---

inla.mesh.project *Methods for projecting to/from an inla.mesh*

Description

Calculate a lattice projection to/from an inla.mesh

Usage

```
inla.mesh.project(...)
inla.mesh.projector(...)

## S3 method for class 'inla.mesh'
inla.mesh.projector(mesh,
             loc = NULL,
             lattice = NULL,
             xlim = NULL,
             ylim = NULL,
             dims = c(100,100),
             projection = NULL,
             crs = NULL,
             ...)

## S3 method for class 'inla.mesh.1d'
inla.mesh.projector(mesh,
             loc = NULL,
             xlim = mesh$interval,
             dims = 100, ...)

## S3 method for class 'inla.mesh.projector'
inla.mesh.project(projector, field, ...)

## S3 method for class 'inla.mesh'
inla.mesh.project(mesh, loc, field = NULL,
                  crs=NULL,
...)
## S3 method for class 'inla.mesh.1d'
inla.mesh.project(mesh, loc, field = NULL, ...)
```

Arguments

mesh	An inla.mesh or inla.mesh.1d object.
loc	Projection locations. Can be a matrix or a SpatialPoints or a SpatialPointsDataFrame object.

lattice	An inla.mesh.lattice object.
xlim	X-axis limits for a lattice. For R2 meshes, defaults to covering the domain.
ylim	Y-axis limits for a lattice. For R2 meshes, defaults to covering the domain.
dims	Lattice dimensions.
projector	An inla.mesh.projector object.
field	Basis function weights, one per mesh basis function, describing the function to be avaluated at the projection locationssFunction values for on the mesh
projection	One of c("default", "longlat", "longsinlat", "mollweide").
crs	An optional CRS or inla.CRS object associated with loc and/or lattice.
...	Additional arguments passed on to methods.

Details

The call inla.mesh.project(mesh, loc, field=..., ...), is a shortcut to inla.mesh.project(inla.mesh.projector(me loc), field).

Value

For inla.mesh.project(mesh, ...), a list with projection information. For inla.mesh.projector(mesh, ...), an inla.mesh.projector object. For inla.mesh.project(projector, field, ...), a field projected from the mesh onto the locations given by the projector object.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.mesh, inla.mesh.1d, inla.mesh.lattice

Examples

```
n = 20
loc = matrix(runif(n*2), n, 2)
mesh = inla.mesh.create(loc, refine=list(max.edge=0.05))
proj = inla.mesh.projector(mesh)
field = cos(mesh$loc[,1]*2*pi*3)*sin(mesh$loc[,2]*2*pi*7)
image(proj$x, proj$y, inla.mesh.project(proj, field))

if (require(rgl)) {
  plot(mesh, rgl=TRUE, col=field, draw.edges=FALSE, draw.vertices=FALSE)
}
```

`inla.mesh.query` *High-quality triangulations*

Description

Query information about an inla.mesh object.

Usage

```
inla.mesh.query(mesh, ...)
```

Arguments

`mesh`	An `inla.mesh` object.
`...`	Query arguments. • tt.neighbours Compute neighbour triangles for triangles; list of vectors: list(triangles, orders) • vt.neighbours Compute neighbour triangles for vertices; list of vectors: list(vertices, orders)

Value

A list of query results.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.mesh.create, inla.mesh.segment, inla.mesh.lattice

Examples

```
loc = matrix(c(0.1,0.15),1,2)
lattice = inla.mesh.lattice(dims=c(10,10))
mesh = inla.mesh.create(loc=loc, lattice=lattice, extend=FALSE)

vt = which(inla.mesh.query(mesh,
                           vt.neighbours=list(mesh$idx$loc,
                                              4:6))$vt.neighbours)

mesh2 = inla.mesh.create(mesh$loc, tv=mesh$graph$tv[vt,,drop=FALSE],
                         refine=FALSE, extend=FALSE)
```

inla.mesh.segment *Constraint segments for inla.mesh*

Description

Constructs inla.mesh.segment objects that can be used to specify boundary and interior constraint edges in calls to inla.mesh.

Usage

```
## Create or join inla.mesh.segment objects.
inla.mesh.segment(...)
## Default S3 method:
inla.mesh.segment(loc = NULL, idx = NULL, grp = NULL,
  is.bnd = TRUE, crs = NULL, ...)
## S3 method for class 'inla.mesh.segment'
inla.mesh.segment(..., grp.default = 0)

inla.contour.segment(x = seq(0, 1, length.out = nrow(z)),
                     y = seq(0, 1, length.out = ncol(z)),
                     z,
                     nlevels = 10,
                     levels = pretty(range(z, na.rm = TRUE), nlevels),
                     groups = seq_len(length(levels)),
                     positive = TRUE,
                     eps = NULL,
                     crs = NULL)
```

Arguments

loc	Matrix of point locations.
idx	Segment index sequence vector or index pair matrix. The indices refer to the rows of loc. If loc==NULL, the indices will be interpreted as indices into the point specification supplied to inla.mesh.create. If is.bnd==TRUE, defaults to linking all the points in loc, as c(1:nrow(loc),1L), otherwise 1:nrow(loc).
grp	Vector of group labels for each segment. Set to NULL to let the labels be chosen automatically in a call to inla.mesh.create.
is.bnd	TRUE if the segments are boundary segments, otherwise FALSE.
grp.default	When joining segments, use this group label for segments that have grp=NULL.
x, y, z, nlevels, levels	Parameters specifying a set of surface contours, with syntax described in contour.
groups	Vector of group ID:s, one for each contour level.
positive	TRUE if the contours should encircle positive level excursions in a counter clockwise direction.
eps	Tolerance for inla.simplify.curve.
crs	An optional CRS or inla.CRS object
...	Additional parameters. When joining segments, a list of inla.mesh.segment objects.

Value

An `inla.mesh.segment` object.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

`inla.mesh.create`, `inla.mesh.2d`

Examples

```
## Create a square boundary and a diagonal interior segment
loc.bnd = matrix(c(0,0, 1,0, 1,1, 0,1), 4, 2, byrow=TRUE)
loc.int = matrix(c(0.9,0.1, 0.1,0.6), 2, 2, byrow=TRUE)
segm.bnd = inla.mesh.segment(loc.bnd)
segm.int = inla.mesh.segment(loc.int, is.bnd=FALSE)

## Points to be meshed
loc = matrix(runif(10*2),10,2)*0.9+0.05
mesh = inla.mesh.create(loc,
                        boundary=segm.bnd,
                        interior=segm.int,
                        refine=list())
plot(mesh)

## Not run:
mesh = inla.mesh.create(loc, interior=list(segm.bnd, segm.int))
plot(mesh)

## End(Not run)
```

`inla.models` *Valid models in INLA*

Description

This page describe the models implemented in `inla`, divided into sections: latent, group, mix, link, predictor, hazard, likelihood, prior, wrapper .

Usage

```
inla.models()
```

Value

Valid sections are: latent, group, mix, link, predictor, hazard, likelihood, prior, wrapper

Section 'latent'. Valid models in this section are:

Model 'linear'. Number of hyperparmeters are 0.

Model 'iid'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '1001'

name = 'log precision'
short.name = 'prec'
prior = 'loggamma'
param = '1 5e-05'
initial = '4'
fixed = 'FALSE'
to.theta = '`function(x) log(x)`'
from.theta = '`function(x) exp(x)`'

Properties: doc = 'Gaussian random effects in dim=1'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'FALSE'
pdf = 'indep'

Model 'mec'. Number of hyperparmeters are 4.

Hyperparameter 'theta1' hyperid = '2001'
name = 'beta'
short.name = 'b'
prior = 'gaussian'
param = '1 0.001'
initial = '1'
fixed = 'FALSE'
to.theta = '`function(x) x`'
from.theta = '`function(x) x`'

Hyperparameter 'theta2' hyperid = '2002'
name = 'prec.u'
short.name = 'prec'
prior = 'loggamma'
param = '1 1e-04'
initial = '9.21034037197618'
fixed = 'TRUE'
to.theta = '`function(x) log(x)`'
from.theta = '`function(x) exp(x)`'

Hyperparameter 'theta3' hyperid = '2003'
name = 'mean.x'
short.name = 'mu.x'
prior = 'gaussian'
param = '0 1e-04'
initial = '0'
fixed = 'TRUE'
to.theta = '`function(x) x`'

from.theta = ‘`function(x) x`’

Hyperparameter ‘theta4’ hyperid = ‘2004’

name = ‘prec.x’

short.name = ‘prec.x’

prior = ‘loggamma’

param = ‘1 10000’

initial = ‘-9.21034037197618’

fixed = ‘TRUE’

to.theta = ‘`function(x) log(x)`’

from.theta = ‘`function(x) exp(x)`’

Properties: doc = ‘Classical measurement error model’

constr = ‘FALSE’

nrow.ncol = ‘FALSE’

augmented = ‘FALSE’

aug.factor = ‘1’

aug.constr = ‘NULL’

n.div.by = ‘NULL’

n.required = ‘FALSE’

set.default.values = ‘FALSE’

status = ‘experimental’

pdf = ‘mec’

Model ‘meb’. Number of hyperparmeters are 2.

Hyperparameter ‘theta1’ hyperid = ‘3001’

name = ‘beta’

short.name = ‘b’

prior = ‘gaussian’

param = ‘1 0.001’

initial = ‘1’

fixed = ‘FALSE’

to.theta = ‘`function(x) x`’

from.theta = ‘`function(x) x`’

Hyperparameter ‘theta2’ hyperid = ‘3002’

name = ‘prec.u’

short.name = ‘prec’

prior = ‘loggamma’

param = ‘1 1e-04’

initial = ‘6.90775527898214’

fixed = ‘FALSE’

to.theta = ‘`function(x) log(x)`’

from.theta = ‘`function(x) exp(x)`’

Properties: doc = ‘Berkson measurement error model’

constr = ‘FALSE’

nrow.ncol = ‘FALSE’

augmented = ‘FALSE’

aug.factor = ‘1’

aug.constr = ‘NULL’
n.div.by = ‘NULL’
n.required = ‘FALSE’
set.default.values = ‘FALSE’
status = ‘experimental’
pdf = ‘meb’

Model ‘rgeneric’. Number of hyperparmeters are 0.

Model ‘rw1’. Number of hyperparmeters are 1.

Hyperparameter ‘theta’ hyperid = ‘4001’
name = ‘log precision’
short.name = ‘prec’
prior = ‘loggamma’
param = ‘1 5e-05’
initial = ‘4’
fixed = ‘FALSE’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’

Properties: doc = ‘Random walk of order 1’
constr = ‘TRUE’
nrow.ncol = ‘FALSE’
augmented = ‘FALSE’
aug.factor = ‘1’
aug.constr = ‘NULL’
n.div.by = ‘NULL’
n.required = ‘FALSE’
set.default.values = ‘FALSE’
min.diff = ‘1e-05’
pdf = ‘rw1’

Model ‘rw2’. Number of hyperparmeters are 1.

Hyperparameter ‘theta’ hyperid = ‘5001’
name = ‘log precision’
short.name = ‘prec’
prior = ‘loggamma’
param = ‘1 5e-05’
initial = ‘4’
fixed = ‘FALSE’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’

Properties: doc = ‘Random walk of order 2’
constr = ‘TRUE’
nrow.ncol = ‘FALSE’
augmented = ‘FALSE’
aug.factor = ‘1’
aug.constr = ‘NULL’
n.div.by = ‘NULL’

n.required = 'FALSE'
set.default.values = 'FALSE'
min.diff = '0.001'
pdf = 'rw2'

Model 'crw2'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '6001'
name = 'log precision'
short.name = 'prec'
prior = 'loggamma'
param = '1 5e-05'
initial = '4'
fixed = 'FALSE'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Properties: doc = 'Exact solution to the random walk of order 2'
constr = 'TRUE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '2'
aug.constr = '1'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'FALSE'
min.diff = '0.001'
pdf = 'crw2'

Model 'seasonal'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '7001'
name = 'log precision'
short.name = 'prec'
prior = 'loggamma'
param = '1 5e-05'
initial = '4'
fixed = 'FALSE'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Properties: doc = 'Seasonal model for time series'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'FALSE'
pdf = 'seasonal'

Model 'besag'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '8001'

name = 'log precision'

short.name = 'prec'

prior = 'loggamma'

param = '1 5e-05'

initial = '4'

fixed = 'FALSE'

to.theta = '`function(x) log(x)`'

from.theta = '`function(x) exp(x)`'

Properties: doc = 'The Besag area model (CAR-model)'

constr = 'TRUE'

nrow.ncol = 'FALSE'

augmented = 'FALSE'

aug.factor = '1'

aug.constr = 'NULL'

n.div.by = 'NULL'

n.required = 'TRUE'

set.default.values = 'TRUE'

pdf = 'besag'

Model 'besag2'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '9001'

name = 'log precision'

short.name = 'prec'

prior = 'loggamma'

param = '1 5e-05'

initial = '4'

fixed = 'FALSE'

to.theta = '`function(x) log(x)`'

from.theta = '`function(x) exp(x)`'

Hyperparameter 'theta2' hyperid = '9002'

name = 'scaling parameter'

short.name = 'a'

prior = 'loggamma'

param = '10 10'

initial = '0'

fixed = 'FALSE'

to.theta = '`function(x) log(x)`'

from.theta = '`function(x) exp(x)`'

Properties: doc = 'The shared Besag model'

constr = 'FALSE'

nrow.ncol = 'FALSE'

augmented = 'FALSE'

aug.factor = '1'

aug.constr = '1 2'

n.div.by = '2'
n.required = 'TRUE'
set.default.values = 'TRUE'
pdf = 'besag2'

Model 'bym'. Number of hyperparmeters are 2.
Hyperparameter 'theta1' hyperid = '10001'
name = 'log unstructured precision'
short.name = 'prec.unstruct'
prior = 'loggamma'
param = '1 5e-04'
initial = '4'
fixed = 'FALSE'
to.theta = '`function(x) log(x)`'
from.theta = '`function(x) exp(x)`'
Hyperparameter 'theta2' hyperid = '10002'
name = 'log spatial precision'
short.name = 'prec.spatial'
prior = 'loggamma'
param = '1 5e-04'
initial = '4'
fixed = 'FALSE'
to.theta = '`function(x) log(x)`'
from.theta = '`function(x) exp(x)`'
Properties: doc = 'The BYM-model (Besag-York-Mollier model)'
constr = 'TRUE'
nrow.ncol = 'FALSE'
augmented = 'TRUE'
aug.factor = '2'
aug.constr = '2'
n.div.by = 'NULL'
n.required = 'TRUE'
set.default.values = 'TRUE'
pdf = 'bym'

Model 'bym2'. Number of hyperparmeters are 2.
Hyperparameter 'theta1' hyperid = '11001'
name = 'log precision'
short.name = 'prec'
prior = 'pc.prec'
param = '1 0.01'
initial = '4'
fixed = 'FALSE'
to.theta = '`function(x) log(x)`'
from.theta = '`function(x) exp(x)`'
Hyperparameter 'theta2' hyperid = '11002'
name = 'logit phi'

short.name = 'phi'
prior = 'pc'
param = '0.5 0.5'
initial = '-3'
fixed = 'FALSE'
to.theta = 'function(x) log(x/(1-x))'
from.theta = 'function(x) exp(x)/(1+exp(x))'

Properties: doc = 'The BYM-model with the PC priors'
constr = 'TRUE'
nrow.ncol = 'FALSE'
augmented = 'TRUE'
aug.factor = '2'
aug.constr = '2'
n.div.by = 'NULL'
n.required = 'TRUE'
set.default.values = 'TRUE'
status = 'experimental'
pdf = 'bym2'

Model 'besagproper'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '12001'
name = 'log precision'
short.name = 'prec'
prior = 'loggamma'
param = '1 5e-04'
initial = '2'
fixed = 'FALSE'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta2' hyperid = '12002'
name = 'log diagonal'
short.name = 'diag'
prior = 'loggamma'
param = '1 1'
initial = '1'
fixed = 'FALSE'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Properties: doc = 'A proper version of the Besag model'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'TRUE'

set.default.values = 'TRUE'
status = 'experimental'
pdf = 'besagproper'

Model 'besagproper2'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '13001'
name = 'log precision'
short.name = 'prec'
prior = 'loggamma'
param = '1 5e-04'
initial = '2'
fixed = 'FALSE'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta2' hyperid = '13002'
name = 'logit lambda'
short.name = 'lambda'
prior = 'gaussian'
param = '0 0.45'
initial = '3'
fixed = 'FALSE'
to.theta = `'function(x) log(x/(1-x))'`
from.theta = `'function(x) exp(x)/(1+exp(x))'`

Properties: doc = 'An alternative proper version of the Besag model'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'TRUE'
set.default.values = 'TRUE'
status = 'experimental'
pdf = 'besagproper2'

Model 'fgn'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '13101'
name = 'log precision'
short.name = 'prec'
prior = 'pc.prec'
param = '3 0.01'
initial = '1'
fixed = 'FALSE'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta2' hyperid = '13102'
name = 'logit H'

short.name = 'H'
prior = 'pcfgnh'
param = '0.9 0.1'
initial = '2'
fixed = 'FALSE'
to.theta = `'function(x) log((2*x-1)/(2*(1-x)))'`
from.theta = `'function(x) 0.5 + 0.5*exp(x)/(1+exp(x))'`

Properties: doc = 'Fractional Gaussian noise model'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'TRUE'
aug.factor = '4'
aug.constr = '1'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'TRUE'
order.default = '3'
order.defined = '3 4'
pdf = 'fgn'

Model 'ar1'. Number of hyperparmeters are 3.

Hyperparameter 'theta1' hyperid = '14001'
name = 'log precision'
short.name = 'prec'
prior = 'loggamma'
param = '1 5e-05'
initial = '4'
fixed = 'FALSE'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta2' hyperid = '14002'
name = 'logit lag one correlation'
short.name = 'rho'
prior = 'normal'
param = '0 0.15'
initial = '2'
fixed = 'FALSE'
to.theta = `'function(x) log((1+x)/(1-x))'`
from.theta = `'function(x) 2*exp(x)/(1+exp(x))-1'`

Hyperparameter 'theta3' hyperid = '14003'
name = 'mean'
short.name = 'mean'
prior = 'normal'
param = '0 1'
initial = '0'
fixed = 'TRUE'

to.theta = ‘function(x) x’
from.theta = ‘function(x) x’
Properties: doc = ‘Auto-regressive model of order 1 (AR(1))’
constr = ‘FALSE’
nrow.ncol = ‘FALSE’
augmented = ‘FALSE’
aug.factor = ‘1’
aug.constr = ‘NULL’
n.div.by = ‘NULL’
n.required = ‘FALSE’
set.default.values = ‘FALSE’
pdf = ‘ar1’

Model ‘ar’. Number of hyperparmeters are 11.
Hyperparameter ‘theta1’ hyperid = ‘15001’
name = ‘log precision’
short.name = ‘prec’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘pc.prec’
param = ‘3 0.01’
to.theta = ‘function(x) log(x)’
from.theta = ‘function(x) exp(x)’
Hyperparameter ‘theta2’ hyperid = ‘15002’
name = ‘pacf1’
short.name = ‘pacf1’
initial = ‘1’
fixed = ‘FALSE’
prior = ‘pc.cor0’
param = ‘0.5 0.5’
to.theta = ‘function(x) log((1+x)/(1-x))’
from.theta = ‘function(x) 2*exp(x)/(1+exp(x))-1’
Hyperparameter ‘theta3’ hyperid = ‘15003’
name = ‘pacf2’
short.name = ‘pacf2’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘pc.cor0’
param = ‘0.5 0.4’
to.theta = ‘function(x) log((1+x)/(1-x))’
from.theta = ‘function(x) 2*exp(x)/(1+exp(x))-1’
Hyperparameter ‘theta4’ hyperid = ‘15004’
name = ‘pacf3’
short.name = ‘pacf3’
initial = ‘0’
fixed = ‘FALSE’

prior = 'pc.cor0'
param = '0.5 0.3'
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'

Hyperparameter 'theta5' hyperid = '15005'
name = 'pacf4'
short.name = 'pacf4'
initial = '0'
fixed = 'FALSE'
prior = 'pc.cor0'
param = '0.5 0.2'
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'

Hyperparameter 'theta6' hyperid = '15006'
name = 'pacf5'
short.name = 'pacf5'
initial = '0'
fixed = 'FALSE'
prior = 'pc.cor0'
param = '0.5 0.1'
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'

Hyperparameter 'theta7' hyperid = '15007'
name = 'pacf6'
short.name = 'pacf6'
initial = '0'
fixed = 'FALSE'
prior = 'pc.cor0'
param = '0.5 0.1'
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'

Hyperparameter 'theta8' hyperid = '15008'
name = 'pacf7'
short.name = 'pacf7'
initial = '0'
fixed = 'FALSE'
prior = 'pc.cor0'
param = '0.5 0.1'
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'

Hyperparameter 'theta9' hyperid = '15009'
name = 'pacf8'
short.name = 'pacf8'
initial = '0'
fixed = 'FALSE'

prior = 'pc.cor0'
param = '0.5 0.1'
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'

Hyperparameter 'theta10' hyperid = '15010'
name = 'pacf9'
short.name = 'pacf9'
initial = '0'
fixed = 'FALSE'
prior = 'pc.cor0'
param = '0.5 0.1'
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'

Hyperparameter 'theta11' hyperid = '15011'
name = 'pacf10'
short.name = 'pacf10'
initial = '0'
fixed = 'FALSE'
prior = 'pc.cor0'
param = '0.5 0.1'
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'

Properties: doc = 'Auto-regressive model of order p (AR(p))'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'FALSE'
pdf = 'ar'

Model 'ou'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '16001'
name = 'log precision'
short.name = 'prec'
prior = 'loggamma'
param = '1 5e-05'
initial = '4'
fixed = 'FALSE'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta2' hyperid = '16002'
name = 'log phi'
short.name = 'phi'

prior = 'normal'
param = '0 0.2'
initial = '-1'
fixed = 'FALSE'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Properties: doc = 'The Ornstein-Uhlenbeck process'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'FALSE'
pdf = 'ou'

Model 'generic'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '17001'
name = 'log precision'
short.name = 'prec'
prior = 'loggamma'
param = '1 5e-05'
initial = '4'
fixed = 'FALSE'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Properties: doc = 'A generic model'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'TRUE'
set.default.values = 'TRUE'
pdf = 'generic0'

Model 'generic0'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '18001'
name = 'log precision'
short.name = 'prec'
prior = 'loggamma'
param = '1 5e-05'
initial = '4'
fixed = 'FALSE'
to.theta = `'function(x) log(x)'`

from.theta = ‘`function(x) exp(x)`’

Properties: doc = ‘A generic model (type 0)’

constr = ‘FALSE’

nrow.ncol = ‘FALSE’

augmented = ‘FALSE’

aug.factor = ‘1’

aug.constr = ‘NULL’

n.div.by = ‘NULL’

n.required = ‘TRUE’

set.default.values = ‘TRUE’

pdf = ‘generic0’

Model ‘generic1’. Number of hyperparmeters are 2.

Hyperparameter ‘theta1’ hyperid = ‘19001’

name = ‘log precision’

short.name = ‘prec’

prior = ‘loggamma’

param = ‘1 5e-05’

initial = ‘4’

fixed = ‘FALSE’

to.theta = ‘`function(x) log(x)`’

from.theta = ‘`function(x) exp(x)`’

Hyperparameter ‘theta2’ hyperid = ‘19002’

name = ‘beta’

short.name = ‘beta’

initial = ‘2’

fixed = ‘FALSE’

prior = ‘gaussian’

param = ‘0 0.1’

to.theta = ‘`function(x) log(x/(1-x))`’

from.theta = ‘`function(x) exp(x)/(1+exp(x))`’

Properties: doc = ‘A generic model (type 1)’

constr = ‘FALSE’

nrow.ncol = ‘FALSE’

augmented = ‘FALSE’

aug.factor = ‘1’

aug.constr = ‘NULL’

n.div.by = ‘NULL’

n.required = ‘TRUE’

set.default.values = ‘TRUE’

pdf = ‘generic1’

Model ‘generic2’. Number of hyperparmeters are 2.

Hyperparameter ‘theta1’ hyperid = ‘20001’

name = ‘log precision cmatrix’

short.name = ‘prec’

initial = ‘4’

fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta2' hyperid = '20002'
name = 'log precision random'
short.name = 'prec.random'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 0.001'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Properties: doc = 'A generic model (type 2)'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '2'
aug.constr = '2'
n.div.by = 'NULL'
n.required = 'TRUE'
set.default.values = 'TRUE'
pdf = 'generic2'

Model 'generic3'. Number of hyperparmeters are 11.

Hyperparameter 'theta1' hyperid = '21001'
name = 'log precision1'
short.name = 'prec1'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta2' hyperid = '21002'
name = 'log precision2'
short.name = 'prec2'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta3' hyperid = '21003'
name = 'log precision3'

short.name = ‘prec3’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘function(x) log(x)’
from.theta = ‘function(x) exp(x)’

Hyperparameter ‘theta4’ hyperid = ‘21004’
name = ‘log precision4’
short.name = ‘prec4’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘function(x) log(x)’
from.theta = ‘function(x) exp(x)’

Hyperparameter ‘theta5’ hyperid = ‘21005’
name = ‘log precision5’
short.name = ‘prec5’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘function(x) log(x)’
from.theta = ‘function(x) exp(x)’

Hyperparameter ‘theta6’ hyperid = ‘21006’
name = ‘log precision6’
short.name = ‘prec6’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘function(x) log(x)’
from.theta = ‘function(x) exp(x)’

Hyperparameter ‘theta7’ hyperid = ‘21007’
name = ‘log precision7’
short.name = ‘prec7’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘function(x) log(x)’
from.theta = ‘function(x) exp(x)’

Hyperparameter ‘theta8’ hyperid = ‘21008’
name = ‘log precision8’

short.name = 'prec8'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta9' hyperid = '21009'
name = 'log precision9'
short.name = 'prec9'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta10' hyperid = '21010'
name = 'log precision10'
short.name = 'prec10'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta11' hyperid = '21011'
name = 'log precision common'
short.name = 'prec.common'
initial = '0'
fixed = 'TRUE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Properties: doc = 'A generic model (type 3)'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'TRUE'
set.default.values = 'TRUE'
status = 'experimental'
pdf = 'generic3'

Model 'spde'. Number of hyperparmeters are 4.

Hyperparameter 'theta1' hyperid = '22001'

name = 'theta.T'

short.name = 'T'

initial = '2'

fixed = 'FALSE'

prior = 'normal'

param = '0 1'

to.theta = '`function(x) x`'

from.theta = '`function(x) x`'

Hyperparameter 'theta2' hyperid = '22002'

name = 'theta.K'

short.name = 'K'

initial = '-2'

fixed = 'FALSE'

prior = 'normal'

param = '0 1'

to.theta = '`function(x) x`'

from.theta = '`function(x) x`'

Hyperparameter 'theta3' hyperid = '22003'

name = 'theta.KT'

short.name = 'KT'

initial = '0'

fixed = 'FALSE'

prior = 'normal'

param = '0 1'

to.theta = '`function(x) x`'

from.theta = '`function(x) x`'

Hyperparameter 'theta4' hyperid = '22004'

name = 'theta.OC'

short.name = 'OC'

initial = '-20'

fixed = 'TRUE'

prior = 'normal'

param = '0 0.2'

to.theta = '`function(x) log(x/(1-x))`'

from.theta = '`function(x) exp(x)/(1+exp(x))`'

Properties: doc = 'A SPDE model'

constr = 'FALSE'

nrow.ncol = 'FALSE'

augmented = 'FALSE'

aug.factor = '1'

aug.constr = 'NULL'

n.div.by = 'NULL'

n.required = 'TRUE'

set.default.values = 'TRUE'
pdf = 'spde'

Model 'spde2'. Number of hyperparmeters are 100.

Hyperparameter 'theta1' hyperid = '23001'
name = 'theta1'
short.name = 't1'
initial = '0'
fixed = 'FALSE'
prior = 'mvnorm'
param = '1 1'
to.theta = '`function(x) x`'
from.theta = '`function(x) x`'

Hyperparameter 'theta2' hyperid = '23002'
name = 'theta2'
short.name = 't2'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = '`function(x) x`'
from.theta = '`function(x) x`'

Hyperparameter 'theta3' hyperid = '23003'
name = 'theta3'
short.name = 't3'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = '`function(x) x`'
from.theta = '`function(x) x`'

Hyperparameter 'theta4' hyperid = '23004'
name = 'theta4'
short.name = 't4'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = '`function(x) x`'
from.theta = '`function(x) x`'

Hyperparameter 'theta5' hyperid = '23005'
name = 'theta5'
short.name = 't5'
initial = '0'
fixed = 'FALSE'
prior = 'none'

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta6’ hyperid = ‘23006’
name = ‘theta6’
short.name = ‘t6’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta7’ hyperid = ‘23007’
name = ‘theta7’
short.name = ‘t7’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta8’ hyperid = ‘23008’
name = ‘theta8’
short.name = ‘t8’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta9’ hyperid = ‘23009’
name = ‘theta9’
short.name = ‘t9’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta10’ hyperid = ‘23010’
name = ‘theta10’
short.name = ‘t10’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta11’ hyperid = ‘23011’
name = ‘theta11’
short.name = ‘t11’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta12’ hyperid = ‘23012’
name = ‘theta12’
short.name = ‘t12’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta13’ hyperid = ‘23013’
name = ‘theta13’
short.name = ‘t13’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta14’ hyperid = ‘23014’
name = ‘theta14’
short.name = ‘t14’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta15’ hyperid = ‘23015’
name = ‘theta15’
short.name = ‘t15’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta16’ hyperid = ‘23016’
name = ‘theta16’
short.name = ‘t16’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta17’ hyperid = ‘23017’
name = ‘theta17’
short.name = ‘t17’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta18’ hyperid = ‘23018’
name = ‘theta18’
short.name = ‘t18’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta19’ hyperid = ‘23019’
name = ‘theta19’
short.name = ‘t19’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta20’ hyperid = ‘23020’
name = ‘theta20’
short.name = ‘t20’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta21’ hyperid = ‘23021’
name = ‘theta21’
short.name = ‘t21’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta22’ hyperid = ‘23022’
name = ‘theta22’
short.name = ‘t22’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta23’ hyperid = ‘23023’
name = ‘theta23’
short.name = ‘t23’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta24’ hyperid = ‘23024’
name = ‘theta24’
short.name = ‘t24’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta25’ hyperid = ‘23025’
name = ‘theta25’
short.name = ‘t25’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta26’ hyperid = ‘23026’
name = ‘theta26’
short.name = ‘t26’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta27’ hyperid = ‘23027’
name = ‘theta27’
short.name = ‘t27’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta28’ hyperid = ‘23028’
name = ‘theta28’
short.name = ‘t28’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta29’ hyperid = ‘23029’
name = ‘theta29’
short.name = ‘t29’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta30’ hyperid = ‘23030’
name = ‘theta30’
short.name = ‘t30’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta31’ hyperid = ‘23031’
name = ‘theta31’
short.name = ‘t31’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta32’ hyperid = ‘23032’
name = ‘theta32’
short.name = ‘t32’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta33’ hyperid = ‘23033’
name = ‘theta33’
short.name = ‘t33’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta34’ hyperid = ‘23034’
name = ‘theta34’
short.name = ‘t34’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta35’ hyperid = ‘23035’
name = ‘theta35’
short.name = ‘t35’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta36’ hyperid = ‘23036’
name = ‘theta36’
short.name = ‘t36’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta37’ hyperid = ‘23037’
name = ‘theta37’
short.name = ‘t37’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta38’ hyperid = ‘23038’
name = ‘theta38’
short.name = ‘t38’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta39’ hyperid = ‘23039’
name = ‘theta39’
short.name = ‘t39’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta40’ hyperid = ‘23040’
name = ‘theta40’
short.name = ‘t40’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta41’ hyperid = ‘23041’
name = ‘theta41’
short.name = ‘t41’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta42’ hyperid = ‘23042’
name = ‘theta42’
short.name = ‘t42’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta43’ hyperid = ‘23043’
name = ‘theta43’
short.name = ‘t43’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta44’ hyperid = ‘23044’
name = ‘theta44’
short.name = ‘t44’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta45’ hyperid = ‘23045’
name = ‘theta45’
short.name = ‘t45’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta46’ hyperid = ‘23046’
name = ‘theta46’
short.name = ‘t46’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta47’ hyperid = ‘23047’
name = ‘theta47’
short.name = ‘t47’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta48’ hyperid = ‘23048’
name = ‘theta48’
short.name = ‘t48’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta49’ hyperid = ‘23049’
name = ‘theta49’
short.name = ‘t49’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta50’ hyperid = ‘23050’
name = ‘theta50’
short.name = ‘t50’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta51’ hyperid = ‘23051’
name = ‘theta51’
short.name = ‘t51’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta52’ hyperid = ‘23052’
name = ‘theta52’
short.name = ‘t52’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta53’ hyperid = ‘23053’
name = ‘theta53’
short.name = ‘t53’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta54’ hyperid = ‘23054’
name = ‘theta54’
short.name = ‘t54’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta55’ hyperid = ‘23055’
name = ‘theta55’
short.name = ‘t55’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta56’ hyperid = ‘23056’
name = ‘theta56’
short.name = ‘t56’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta57’ hyperid = ‘23057’
name = ‘theta57’
short.name = ‘t57’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta58’ hyperid = ‘23058’
name = ‘theta58’
short.name = ‘t58’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta59’ hyperid = ‘23059’
name = ‘theta59’
short.name = ‘t59’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta60’ hyperid = ‘23060’
name = ‘theta60’
short.name = ‘t60’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta61’ hyperid = ‘23061’
name = ‘theta61’
short.name = ‘t61’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta62’ hyperid = ‘23062’
name = ‘theta62’
short.name = ‘t62’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta63’ hyperid = ‘23063’
name = ‘theta63’
short.name = ‘t63’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta64’ hyperid = ‘23064’
name = ‘theta64’
short.name = ‘t64’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta65’ hyperid = ‘23065’
name = ‘theta65’
short.name = ‘t65’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta66’ hyperid = ‘23066’
name = ‘theta66’
short.name = ‘t66’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta67’ hyperid = ‘23067’
name = ‘theta67’
short.name = ‘t67’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta68’ hyperid = ‘23068’
name = ‘theta68’
short.name = ‘t68’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta69’ hyperid = ‘23069’
name = ‘theta69’
short.name = ‘t69’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta70’ hyperid = ‘23070’
name = ‘theta70’
short.name = ‘t70’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta71’ hyperid = ‘23071’
name = ‘theta71’
short.name = ‘t71’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta72’ hyperid = ‘23072’
name = ‘theta72’
short.name = ‘t72’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta73’ hyperid = ‘23073’
name = ‘theta73’
short.name = ‘t73’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta74’ hyperid = ‘23074’
name = ‘theta74’
short.name = ‘t74’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta75’ hyperid = ‘23075’
name = ‘theta75’
short.name = ‘t75’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta76’ hyperid = ‘23076’
name = ‘theta76’
short.name = ‘t76’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta77’ hyperid = ‘23077’
name = ‘theta77’
short.name = ‘t77’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta78’ hyperid = ‘23078’
name = ‘theta78’
short.name = ‘t78’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta79’ hyperid = ‘23079’
name = ‘theta79’
short.name = ‘t79’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta80’ hyperid = ‘23080’
name = ‘theta80’
short.name = ‘t80’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta81’ hyperid = ‘23081’
name = ‘theta81’
short.name = ‘t81’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta82’ hyperid = ‘23082’
name = ‘theta82’
short.name = ‘t82’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta83’ hyperid = ‘23083’
name = ‘theta83’
short.name = ‘t83’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta84’ hyperid = ‘23084’
name = ‘theta84’
short.name = ‘t84’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta85’ hyperid = ‘23085’
name = ‘theta85’
short.name = ‘t85’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta86’ hyperid = ‘23086’
name = ‘theta86’
short.name = ‘t86’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta87’ hyperid = ‘23087’
name = ‘theta87’
short.name = ‘t87’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta88’ hyperid = ‘23088’
name = ‘theta88’
short.name = ‘t88’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta89’ hyperid = ‘23089’
name = ‘theta89’
short.name = ‘t89’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta90’ hyperid = ‘23090’
name = ‘theta90’
short.name = ‘t90’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta91’ hyperid = ‘23091’
name = ‘theta91’
short.name = ‘t91’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta92’ hyperid = ‘23092’
name = ‘theta92’
short.name = ‘t92’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta93’ hyperid = ‘23093’
name = ‘theta93’
short.name = ‘t93’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta94’ hyperid = ‘23094’
name = ‘theta94’
short.name = ‘t94’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta95’ hyperid = ‘23095’
name = ‘theta95’
short.name = ‘t95’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta96’ hyperid = ‘23096’
name = ‘theta96’
short.name = ‘t96’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta97’ hyperid = ‘23097’
name = ‘theta97’
short.name = ‘t97’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta98’ hyperid = ‘23098’
name = ‘theta98’
short.name = ‘t98’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta99’ hyperid = ‘23099’
name = ‘theta99’
short.name = ‘t99’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta100’ hyperid = ‘23100’
name = ‘theta100’
short.name = ‘t100’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’

param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Properties: doc = ‘A SPDE2 model’
constr = ‘FALSE’
nrow.ncol = ‘FALSE’
augmented = ‘FALSE’
aug.factor = ‘1’
aug.constr = ‘NULL’
n.div.by = ‘NULL’
n.required = ‘TRUE’
set.default.values = ‘TRUE’
pdf = ‘spde2’

Model ‘spde3’. Number of hyperparmeters are 100.
Hyperparameter ‘theta1’ hyperid = ‘24001’
name = ‘theta1’
short.name = ‘t1’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘mvnorm’
param = ‘1 1’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta2’ hyperid = ‘24002’
name = ‘theta2’
short.name = ‘t2’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta3’ hyperid = ‘24003’
name = ‘theta3’
short.name = ‘t3’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta4’ hyperid = ‘24004’
name = ‘theta4’
short.name = ‘t4’
initial = ‘0’

fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta5’ hyperid = ‘24005’
name = ‘theta5’
short.name = ‘t5’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta6’ hyperid = ‘24006’
name = ‘theta6’
short.name = ‘t6’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta7’ hyperid = ‘24007’
name = ‘theta7’
short.name = ‘t7’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta8’ hyperid = ‘24008’
name = ‘theta8’
short.name = ‘t8’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta9’ hyperid = ‘24009’
name = ‘theta9’
short.name = ‘t9’
initial = ‘0’

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = '`function(x) x`'
from.theta = '`function(x) x`'

Hyperparameter 'theta10' hyperid = '24010'
name = 'theta10'
short.name = 't10'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = '`function(x) x`'
from.theta = '`function(x) x`'

Hyperparameter 'theta11' hyperid = '24011'
name = 'theta11'
short.name = 't11'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = '`function(x) x`'
from.theta = '`function(x) x`'

Hyperparameter 'theta12' hyperid = '24012'
name = 'theta12'
short.name = 't12'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = '`function(x) x`'
from.theta = '`function(x) x`'

Hyperparameter 'theta13' hyperid = '24013'
name = 'theta13'
short.name = 't13'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = '`function(x) x`'
from.theta = '`function(x) x`'

Hyperparameter 'theta14' hyperid = '24014'
name = 'theta14'
short.name = 't14'
initial = '0'

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) x'`
from.theta = `'function(x) x'`

Hyperparameter 'theta15' hyperid = '24015'
name = 'theta15'
short.name = 't15'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) x'`
from.theta = `'function(x) x'`

Hyperparameter 'theta16' hyperid = '24016'
name = 'theta16'
short.name = 't16'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) x'`
from.theta = `'function(x) x'`

Hyperparameter 'theta17' hyperid = '24017'
name = 'theta17'
short.name = 't17'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) x'`
from.theta = `'function(x) x'`

Hyperparameter 'theta18' hyperid = '24018'
name = 'theta18'
short.name = 't18'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) x'`
from.theta = `'function(x) x'`

Hyperparameter 'theta19' hyperid = '24019'
name = 'theta19'
short.name = 't19'
initial = '0'

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) x'`
from.theta = `'function(x) x'`

Hyperparameter 'theta20' hyperid = '24020'
name = 'theta20'
short.name = 't20'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) x'`
from.theta = `'function(x) x'`

Hyperparameter 'theta21' hyperid = '24021'
name = 'theta21'
short.name = 't21'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) x'`
from.theta = `'function(x) x'`

Hyperparameter 'theta22' hyperid = '24022'
name = 'theta22'
short.name = 't22'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) x'`
from.theta = `'function(x) x'`

Hyperparameter 'theta23' hyperid = '24023'
name = 'theta23'
short.name = 't23'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) x'`
from.theta = `'function(x) x'`

Hyperparameter 'theta24' hyperid = '24024'
name = 'theta24'
short.name = 't24'
initial = '0'

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta25' hyperid = '24025'
name = 'theta25'
short.name = 't25'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta26' hyperid = '24026'
name = 'theta26'
short.name = 't26'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta27' hyperid = '24027'
name = 'theta27'
short.name = 't27'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta28' hyperid = '24028'
name = 'theta28'
short.name = 't28'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta29' hyperid = '24029'
name = 'theta29'
short.name = 't29'
initial = '0'

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta30' hyperid = '24030'
name = 'theta30'
short.name = 't30'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta31' hyperid = '24031'
name = 'theta31'
short.name = 't31'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta32' hyperid = '24032'
name = 'theta32'
short.name = 't32'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta33' hyperid = '24033'
name = 'theta33'
short.name = 't33'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta34' hyperid = '24034'
name = 'theta34'
short.name = 't34'
initial = '0'

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta35' hyperid = '24035'
name = 'theta35'
short.name = 't35'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta36' hyperid = '24036'
name = 'theta36'
short.name = 't36'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta37' hyperid = '24037'
name = 'theta37'
short.name = 't37'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta38' hyperid = '24038'
name = 'theta38'
short.name = 't38'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta39' hyperid = '24039'
name = 'theta39'
short.name = 't39'
initial = '0'

fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta40’ hyperid = ‘24040’
name = ‘theta40’
short.name = ‘t40’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta41’ hyperid = ‘24041’
name = ‘theta41’
short.name = ‘t41’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta42’ hyperid = ‘24042’
name = ‘theta42’
short.name = ‘t42’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta43’ hyperid = ‘24043’
name = ‘theta43’
short.name = ‘t43’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) x’
from.theta = ‘function(x) x’

Hyperparameter ‘theta44’ hyperid = ‘24044’
name = ‘theta44’
short.name = ‘t44’
initial = ‘0’

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta45' hyperid = '24045'
name = 'theta45'
short.name = 't45'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta46' hyperid = '24046'
name = 'theta46'
short.name = 't46'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta47' hyperid = '24047'
name = 'theta47'
short.name = 't47'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta48' hyperid = '24048'
name = 'theta48'
short.name = 't48'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta49' hyperid = '24049'
name = 'theta49'
short.name = 't49'
initial = '0'

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta50' hyperid = '24050'
name = 'theta50'
short.name = 't50'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta51' hyperid = '24051'
name = 'theta51'
short.name = 't51'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta52' hyperid = '24052'
name = 'theta52'
short.name = 't52'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta53' hyperid = '24053'
name = 'theta53'
short.name = 't53'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta54' hyperid = '24054'
name = 'theta54'
short.name = 't54'
initial = '0'

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta55' hyperid = '24055'
name = 'theta55'
short.name = 't55'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta56' hyperid = '24056'
name = 'theta56'
short.name = 't56'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta57' hyperid = '24057'
name = 'theta57'
short.name = 't57'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta58' hyperid = '24058'
name = 'theta58'
short.name = 't58'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta59' hyperid = '24059'
name = 'theta59'
short.name = 't59'
initial = '0'

fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta60’ hyperid = ‘24060’
name = ‘theta60’
short.name = ‘t60’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta61’ hyperid = ‘24061’
name = ‘theta61’
short.name = ‘t61’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta62’ hyperid = ‘24062’
name = ‘theta62’
short.name = ‘t62’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta63’ hyperid = ‘24063’
name = ‘theta63’
short.name = ‘t63’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta64’ hyperid = ‘24064’
name = ‘theta64’
short.name = ‘t64’
initial = ‘0’

fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta65’ hyperid = ‘24065’
name = ‘theta65’
short.name = ‘t65’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta66’ hyperid = ‘24066’
name = ‘theta66’
short.name = ‘t66’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta67’ hyperid = ‘24067’
name = ‘theta67’
short.name = ‘t67’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta68’ hyperid = ‘24068’
name = ‘theta68’
short.name = ‘t68’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta69’ hyperid = ‘24069’
name = ‘theta69’
short.name = ‘t69’
initial = ‘0’

fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta70’ hyperid = ‘24070’
name = ‘theta70’
short.name = ‘t70’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta71’ hyperid = ‘24071’
name = ‘theta71’
short.name = ‘t71’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta72’ hyperid = ‘24072’
name = ‘theta72’
short.name = ‘t72’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta73’ hyperid = ‘24073’
name = ‘theta73’
short.name = ‘t73’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta74’ hyperid = ‘24074’
name = ‘theta74’
short.name = ‘t74’
initial = ‘0’

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta75' hyperid = '24075'
name = 'theta75'
short.name = 't75'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta76' hyperid = '24076'
name = 'theta76'
short.name = 't76'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta77' hyperid = '24077'
name = 'theta77'
short.name = 't77'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta78' hyperid = '24078'
name = 'theta78'
short.name = 't78'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta79' hyperid = '24079'
name = 'theta79'
short.name = 't79'
initial = '0'

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta80' hyperid = '24080'
name = 'theta80'
short.name = 't80'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta81' hyperid = '24081'
name = 'theta81'
short.name = 't81'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta82' hyperid = '24082'
name = 'theta82'
short.name = 't82'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta83' hyperid = '24083'
name = 'theta83'
short.name = 't83'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta84' hyperid = '24084'
name = 'theta84'
short.name = 't84'
initial = '0'

fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta85’ hyperid = ‘24085’
name = ‘theta85’
short.name = ‘t85’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta86’ hyperid = ‘24086’
name = ‘theta86’
short.name = ‘t86’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta87’ hyperid = ‘24087’
name = ‘theta87’
short.name = ‘t87’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta88’ hyperid = ‘24088’
name = ‘theta88’
short.name = ‘t88’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta89’ hyperid = ‘24089’
name = ‘theta89’
short.name = ‘t89’
initial = ‘0’

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta90' hyperid = '24090'
name = 'theta90'
short.name = 't90'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta91' hyperid = '24091'
name = 'theta91'
short.name = 't91'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta92' hyperid = '24092'
name = 'theta92'
short.name = 't92'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta93' hyperid = '24093'
name = 'theta93'
short.name = 't93'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta94' hyperid = '24094'
name = 'theta94'
short.name = 't94'
initial = '0'

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta95' hyperid = '24095'
name = 'theta95'
short.name = 't95'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta96' hyperid = '24096'
name = 'theta96'
short.name = 't96'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta97' hyperid = '24097'
name = 'theta97'
short.name = 't97'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta98' hyperid = '24098'
name = 'theta98'
short.name = 't98'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta99' hyperid = '24099'
name = 'theta99'
short.name = 't99'
initial = '0'

fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta100' hyperid = '24100'
name = 'theta100'
short.name = 't100'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Properties: doc = 'A SPDE3 model'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'TRUE'
set.default.values = 'TRUE'
pdf = 'spde3'

Model 'iid1d'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '25001'
name = 'precision'
short.name = 'prec'
initial = '4'
fixed = 'FALSE'
prior = 'wishart1d'
param = '2 1e-04'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Properties: doc = 'Gaussian random effect in dim=1 with Wishart prior'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'TRUE'
pdf = 'iid123d'

Model 'iid2d'. Number of hyperparmeters are 3.

Hyperparameter 'theta1' hyperid = '26001'
name = 'log precision1'
short.name = 'prec1'
initial = '4'
fixed = 'FALSE'
prior = 'wishart2d'
param = '4 1 1 0'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'
Hyperparameter 'theta2' hyperid = '26002'
name = 'log precision2'
short.name = 'prec2'
initial = '4'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'
Hyperparameter 'theta3' hyperid = '26003'
name = 'logit correlation'
short.name = 'cor'
initial = '4'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'
Properties: doc = 'Gaussian random effect in dim=2 with Wishart prior'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'TRUE'
aug.factor = '1'
aug.constr = '1 2'
n.div.by = '2'
n.required = 'TRUE'
set.default.values = 'TRUE'
pdf = 'iid123d'

Model 'iid3d'. Number of hyperparmeters are 6.
Hyperparameter 'theta1' hyperid = '27001'
name = 'log precision1'
short.name = 'prec1'
initial = '4'
fixed = 'FALSE'
prior = 'wishart3d'
param = '7 1 1 1 0 0 0'

to.theta = ‘function(x) log(x)’
from.theta = ‘function(x) exp(x)’
Hyperparameter ‘theta2’ hyperid = ‘27002’
name = ‘log precision2’
short.name = ‘prec2’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) log(x)’
from.theta = ‘function(x) exp(x)’
Hyperparameter ‘theta3’ hyperid = ‘27003’
name = ‘log precision3’
short.name = ‘prec3’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) log(x)’
from.theta = ‘function(x) exp(x)’
Hyperparameter ‘theta4’ hyperid = ‘27004’
name = ‘logit correlation12’
short.name = ‘cor12’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) log((1+x)/(1-x))’
from.theta = ‘function(x) 2*exp(x)/(1+exp(x))-1’
Hyperparameter ‘theta5’ hyperid = ‘27005’
name = ‘logit correlation13’
short.name = ‘cor13’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) log((1+x)/(1-x))’
from.theta = ‘function(x) 2*exp(x)/(1+exp(x))-1’
Hyperparameter ‘theta6’ hyperid = ‘27006’
name = ‘logit correlation23’
short.name = ‘cor23’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’

to.theta = ‘`function(x) log((1+x)/(1-x))`’
from.theta = ‘`function(x) 2*exp(x)/(1+exp(x))-1`’

Properties: doc = ‘Gaussian random effect in dim=3 with Wishart prior’
constr = ‘FALSE’
nrow.ncol = ‘FALSE’
augmented = ‘TRUE’
aug.factor = ‘1’
aug.constr = ‘1 2 3’
n.div.by = ‘3’
n.required = ‘TRUE’
set.default.values = ‘TRUE’
pdf = ‘iid123d’

Model ‘iid4d’. Number of hyperparmeters are 10.

Hyperparameter ‘theta1’ hyperid = ‘28001’
name = ‘log precision1’
short.name = ‘prec1’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘wishart4d’
param = ‘11 1 1 1 1 0 0 0 0 0 0’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’

Hyperparameter ‘theta2’ hyperid = ‘28002’
name = ‘log precision2’
short.name = ‘prec2’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’

Hyperparameter ‘theta3’ hyperid = ‘28003’
name = ‘log precision3’
short.name = ‘prec3’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’

Hyperparameter ‘theta4’ hyperid = ‘28004’
name = ‘log precision4’
short.name = ‘prec4’
initial = ‘4’
fixed = ‘FALSE’

prior = ‘none’
param = ‘’
to.theta = ‘function(x) log(x)’
from.theta = ‘function(x) exp(x)’

Hyperparameter ‘theta5’ hyperid = ‘28005’
name = ‘logit correlation12’
short.name = ‘cor12’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) log((1+x)/(1-x))’
from.theta = ‘function(x) 2*exp(x)/(1+exp(x))-1’

Hyperparameter ‘theta6’ hyperid = ‘28006’
name = ‘logit correlation13’
short.name = ‘cor13’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) log((1+x)/(1-x))’
from.theta = ‘function(x) 2*exp(x)/(1+exp(x))-1’

Hyperparameter ‘theta7’ hyperid = ‘28007’
name = ‘logit correlation14’
short.name = ‘cor14’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) log((1+x)/(1-x))’
from.theta = ‘function(x) 2*exp(x)/(1+exp(x))-1’

Hyperparameter ‘theta8’ hyperid = ‘28008’
name = ‘logit correlation23’
short.name = ‘cor23’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘function(x) log((1+x)/(1-x))’
from.theta = ‘function(x) 2*exp(x)/(1+exp(x))-1’

Hyperparameter ‘theta9’ hyperid = ‘28009’
name = ‘logit correlation24’
short.name = ‘cor24’
initial = ‘0’
fixed = ‘FALSE’

prior = 'none'
param = ''
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'
Hyperparameter 'theta10' hyperid = '28010'
name = 'logit correlation34'
short.name = 'cor34'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'
Properties: doc = 'Gaussian random effect in dim=4 with Wishart prior'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'TRUE'
aug.factor = '1'
aug.constr = '1 2 3 4'
n.div.by = '4'
n.required = 'TRUE'
set.default.values = 'TRUE'
pdf = 'iid123d'
Model 'iid5d'. Number of hyperparmeters are 15.
Hyperparameter 'theta1' hyperid = '29001'
name = 'log precision1'
short.name = 'prec1'
initial = '4'
fixed = 'FALSE'
prior = 'wishart5d'
param = '16 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'
Hyperparameter 'theta2' hyperid = '29002'
name = 'log precision2'
short.name = 'prec2'
initial = '4'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'
Hyperparameter 'theta3' hyperid = '29003'
name = 'log precision3'
short.name = 'prec3'

initial = '4'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta4' hyperid = '29004'
name = 'log precision4'
short.name = 'prec4'
initial = '4'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta5' hyperid = '29005'
name = 'log precision5'
short.name = 'prec5'
initial = '4'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta6' hyperid = '29006'
name = 'logit correlation12'
short.name = 'cor12'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'

Hyperparameter 'theta7' hyperid = '29007'
name = 'logit correlation13'
short.name = 'cor13'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'

Hyperparameter 'theta8' hyperid = '29008'
name = 'logit correlation14'
short.name = 'cor14'

initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) log((1+x)/(1-x))'`
from.theta = `'function(x) 2*exp(x)/(1+exp(x))-1'`

Hyperparameter 'theta9' hyperid = '29009'
name = 'logit correlation15'
short.name = 'cor15'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) log((1+x)/(1-x))'`
from.theta = `'function(x) 2*exp(x)/(1+exp(x))-1'`

Hyperparameter 'theta10' hyperid = '29010'
name = 'logit correlation23'
short.name = 'cor23'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) log((1+x)/(1-x))'`
from.theta = `'function(x) 2*exp(x)/(1+exp(x))-1'`

Hyperparameter 'theta11' hyperid = '29011'
name = 'logit correlation24'
short.name = 'cor24'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) log((1+x)/(1-x))'`
from.theta = `'function(x) 2*exp(x)/(1+exp(x))-1'`

Hyperparameter 'theta12' hyperid = '29012'
name = 'logit correlation25'
short.name = 'cor25'
initial = '0'
fixed = 'FALSE'
prior = 'none'
param = ''
to.theta = `'function(x) log((1+x)/(1-x))'`
from.theta = `'function(x) 2*exp(x)/(1+exp(x))-1'`

Hyperparameter 'theta13' hyperid = '29013'
name = 'logit correlation34'
short.name = 'cor34'

initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) log((1+x)/(1-x))`’
from.theta = ‘`function(x) 2*exp(x)/(1+exp(x))-1`’

Hyperparameter ‘theta14’ hyperid = ‘29014’
name = ‘logit correlation35’
short.name = ‘cor35’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) log((1+x)/(1-x))`’
from.theta = ‘`function(x) 2*exp(x)/(1+exp(x))-1`’

Hyperparameter ‘theta15’ hyperid = ‘29015’
name = ‘logit correlation45’
short.name = ‘cor45’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) log((1+x)/(1-x))`’
from.theta = ‘`function(x) 2*exp(x)/(1+exp(x))-1`’

Properties: doc = ‘Gaussian random effect in dim=5 with Wishart prior’
constr = ‘FALSE’
nrow.ncol = ‘FALSE’
augmented = ‘TRUE’
aug.factor = ‘1’
aug.constr = ‘1 2 3 4 5’
n.div.by = ‘5’
n.required = ‘TRUE’
set.default.values = ‘TRUE’
pdf = ‘iid123d’

Model ‘2diid’. Number of hyperparmeters are 3.

Hyperparameter ‘theta1’ hyperid = ‘30001’
name = ‘log precision1’
short.name = ‘prec1’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’

Hyperparameter ‘theta2’ hyperid = ‘30002’

name = 'log precision2'
short.name = 'prec2'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta3' hyperid = '30003'
name = 'correlation'
short.name = 'cor'
initial = '4'
fixed = 'FALSE'
prior = 'normal'
param = '0 0.15'
to.theta = 'function(x) log((1+x)/(1-x))'
from.theta = 'function(x) 2*exp(x)/(1+exp(x))-1'

Properties: doc = '(This model is obsolute)'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = '1 2'
n.div.by = '2'
n.required = 'TRUE'
set.default.values = 'TRUE'
pdf = 'iid123d'

Model 'z'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '31001'
name = 'log precision'
short.name = 'prec'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Properties: doc = 'The z-model in a classical mixed model formulation'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'TRUE'

set.default.values = 'TRUE'
pdf = 'z'
status = 'experimental'

Model 'rw2d'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '32001'
name = 'log precision'
short.name = 'prec'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = '`function(x) log(x)`'
from.theta = '`function(x) exp(x)`'

Properties: doc = 'Thin-plate spline model'
constr = 'TRUE'
nrow.ncol = 'TRUE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'TRUE'
pdf = 'rw2d'

Model 'rw2diid'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '33001'
name = 'log precision'
short.name = 'prec'
prior = 'pc.prec'
param = '1 0.01'
initial = '4'
fixed = 'FALSE'
to.theta = '`function(x) log(x)`'
from.theta = '`function(x) exp(x)`'

Hyperparameter 'theta2' hyperid = '33002'
name = 'logit phi'
short.name = 'phi'
prior = 'pc'
param = '0.5 0.5'
initial = '3'
fixed = 'FALSE'
to.theta = '`function(x) log(x/(1-x))`'
from.theta = '`function(x) exp(x)/(1+exp(x))`'

Properties: doc = 'Thin-plate spline with iid noise'
constr = 'TRUE'
nrow.ncol = 'TRUE'

augmented = 'TRUE'
aug.factor = '2'
aug.constr = '2'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'TRUE'
status = 'experimental'
pdf = 'rw2diid'

Model 'slm'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '34001'
name = 'log precision'
short.name = 'prec'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = '`function(x) log(x)`'
from.theta = '`function(x) exp(x)`'

Hyperparameter 'theta2' hyperid = '34002'
name = 'rho'
short.name = 'rho'
initial = '0'
fixed = 'FALSE'
prior = 'normal'
param = '0 10'
to.theta = '`function(x) log(x/(1-x))`'
from.theta = '`function(x) 1/(1+exp(-x))`'

Properties: doc = 'Spatial lag model'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'TRUE'
set.default.values = 'TRUE'
pdf = 'slm'
status = 'experimental'

Model 'matern2d'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '35001'
name = 'log precision'
short.name = 'prec'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'

param = '1 5e-05'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta2' hyperid = '35002'
name = 'log range'
short.name = 'range'
initial = '2'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 0.01'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Properties: doc = 'Matern covariance function on a regular grid'
constr = 'FALSE'
nrow.ncol = 'TRUE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'TRUE'
pdf = 'matern2d'

Model 'copy'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '36001'
name = 'beta'
short.name = 'b'
initial = '1'
fixed = 'TRUE'
prior = 'normal'
param = '1 10'
to.theta = `'function(x, REPLACE.ME.low, REPLACE.ME.high) {}`
from.theta = `'function(x, REPLACE.ME.low, REPLACE.ME.high) {}`

Properties: doc = 'Create a copy of a model component'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'FALSE'
pdf = 'NA'

Model 'clinear'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '37001'
name = 'beta'

short.name = ‘b’
initial = ‘1’
fixed = ‘FALSE’
prior = ‘normal’
param = ‘1 10’
to.theta = ‘`function(x, REPLACE.ME.low, REPLACE.ME.high) {}`
from.theta = ‘`function(x, REPLACE.ME.low, REPLACE.ME.high) {}`

Properties: doc = ‘Constrained linear effect’
constr = ‘FALSE’
nrow.ncol = ‘FALSE’
augmented = ‘FALSE’
aug.factor = ‘1’
aug.constr = ‘NULL’
n.div.by = ‘NULL’
n.required = ‘FALSE’
set.default.values = ‘FALSE’
pdf = ‘clinear’

Model ‘sigm’. Number of hyperparmeters are 3.

Hyperparameter ‘theta1’ hyperid = ‘38001’
name = ‘beta’
short.name = ‘b’
initial = ‘1’
fixed = ‘FALSE’
prior = ‘normal’
param = ‘1 10’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta2’ hyperid = ‘38002’
name = ‘loghalflife’
short.name = ‘halflife’
initial = ‘3’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘3 1’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’

Hyperparameter ‘theta3’ hyperid = ‘38003’
name = ‘logshape’
short.name = ‘shape’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘10 10’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’

Properties: **doc =** 'Sigmoidal effect of a covariate'

constr = 'FALSE'

nrow.ncol = 'FALSE'

augmented = 'FALSE'

aug.factor = '1'

aug.constr = 'NULL'

n.div.by = 'NULL'

n.required = 'FALSE'

set.default.values = 'FALSE'

status = 'experimental'

pdf = 'sigm'

Model 'revsigm'. Number of hyperparmeters are 3.

Hyperparameter 'theta1' hyperid = '39001'

name = 'beta'

short.name = 'b'

initial = '1'

fixed = 'FALSE'

prior = 'normal'

param = '1 10'

to.theta = '`function(x) x`'

from.theta = '`function(x) x`'

Hyperparameter 'theta2' hyperid = '39002'

name = 'loghalflife'

short.name = 'halflife'

initial = '3'

fixed = 'FALSE'

prior = 'loggamma'

param = '3 1'

to.theta = '`function(x) log(x)`'

from.theta = '`function(x) exp(x)`'

Hyperparameter 'theta3' hyperid = '39003'

name = 'logshape'

short.name = 'shape'

initial = '0'

fixed = 'FALSE'

prior = 'loggamma'

param = '10 10'

to.theta = '`function(x) log(x)`'

from.theta = '`function(x) exp(x)`'

Properties: **doc =** 'Reverse sigmoidal effect of a covariate'

constr = 'FALSE'

nrow.ncol = 'FALSE'

augmented = 'FALSE'

aug.factor = '1'

aug.constr = 'NULL'

n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'FALSE'
status = 'experimental'
pdf = 'sigm'

Model 'log1exp'. Number of hyperparmeters are 3.

Hyperparameter 'theta1' hyperid = '39011'
name = 'beta'
short.name = 'b'
initial = '1'
fixed = 'FALSE'
prior = 'normal'
param = '0 1'
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta2' hyperid = '39012'
name = 'alpha'
short.name = 'a'
initial = '0'
fixed = 'FALSE'
prior = 'normal'
param = '0 1'
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Hyperparameter 'theta3' hyperid = '39013'
name = 'gamma'
short.name = 'g'
initial = '0'
fixed = 'FALSE'
prior = 'normal'
param = '0 1'
to.theta = 'function(x) x'
from.theta = 'function(x) x'

Properties: doc = 'A nonlinear model of a covariate'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'FALSE'
status = 'experimental'
pdf = 'log1exp'

Model 'logdist'. Number of hyperparmeters are 3.

Hyperparameter 'theta1' hyperid = '39021'

name = 'beta'

short.name = 'b'

initial = '1'

fixed = 'FALSE'

prior = 'normal'

param = '0 1'

to.theta = `'function(x) x'`

from.theta = `'function(x) x'`

Hyperparameter 'theta2' hyperid = '39022'

name = 'alpha1'

short.name = 'a1'

initial = '0'

fixed = 'FALSE'

prior = 'loggamma'

param = '0.1 1'

to.theta = `'function(x) log(x)'`

from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta3' hyperid = '39023'

name = 'alpha2'

short.name = 'a2'

initial = '0'

fixed = 'FALSE'

prior = 'loggamma'

param = '0.1 1'

to.theta = `'function(x) log(x)'`

from.theta = `'function(x) exp(x)'`

Properties: doc = 'A nonlinear model of a covariate'

constr = 'FALSE'

nrow.ncol = 'FALSE'

augmented = 'FALSE'

aug.factor = '1'

aug.constr = 'NULL'

n.div.by = 'NULL'

n.required = 'FALSE'

set.default.values = 'FALSE'

status = 'experimental'

pdf = 'logdist'

Section 'group'. Valid models in this section are:

Model 'exchangeable'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '40001'

name = 'logit correlation'

short.name = 'rho'

initial = '1'

fixed = 'FALSE'

prior = 'normal'
param = '0 0.2'
to.theta = `'function(x, REPLACE.ME.ngroup) log((1+x*(ngroup-1))/(1-x))'`
from.theta = `'function(x, REPLACE.ME.ngroup) (exp(x)-1)/(exp(x) + ngroup -1)'`

Properties: doc = 'Exchangeable correlations'

Model 'exchangeablepos'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '40101'
name = 'logit correlation'
short.name = 'rho'
initial = '1'
fixed = 'FALSE'
prior = 'pc.cor0'
param = '0.5 0.5'
to.theta = `'function(x) log(x/(1-x))'`
from.theta = `'function(x) exp(x)/(1+exp(x))'`

Properties: doc = 'Exchangeable positive correlations'

Model 'ar1'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '41001'
name = 'logit correlation'
short.name = 'rho'
initial = '2'
fixed = 'FALSE'
prior = 'normal'
param = '0 0.15'
to.theta = `'function(x) log((1+x)/(1-x))'`
from.theta = `'function(x) 2*exp(x)/(1+exp(x))-1'`

Properties: doc = 'AR(1) correlations'

Model 'ar'. Number of hyperparmeters are 11.

Hyperparameter 'theta1' hyperid = '42001'
name = 'log precision'
short.name = 'prec'
initial = '0'
fixed = 'TRUE'
prior = 'pc.prec'
param = '3 0.01'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta2' hyperid = '42002'
name = 'pacf1'
short.name = 'pacf1'
initial = '2'
fixed = 'FALSE'
prior = 'pc.cor0'
param = '0.5 0.5'
to.theta = `'function(x) log((1+x)/(1-x))'`

from.theta = ‘`function(x) 2*exp(x)/(1+exp(x))-1`’

Hyperparameter ‘theta3’ hyperid = ‘42003’

name = ‘pacf2’

short.name = ‘pacf2’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘pc.cor0’

param = ‘0.5 0.4’

to.theta = ‘`function(x) log((1+x)/(1-x))`’

from.theta = ‘`function(x) 2*exp(x)/(1+exp(x))-1`’

Hyperparameter ‘theta4’ hyperid = ‘42004’

name = ‘pacf3’

short.name = ‘pacf3’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘pc.cor0’

param = ‘0.5 0.3’

to.theta = ‘`function(x) log((1+x)/(1-x))`’

from.theta = ‘`function(x) 2*exp(x)/(1+exp(x))-1`’

Hyperparameter ‘theta5’ hyperid = ‘42005’

name = ‘pacf4’

short.name = ‘pacf4’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘pc.cor0’

param = ‘0.5 0.2’

to.theta = ‘`function(x) log((1+x)/(1-x))`’

from.theta = ‘`function(x) 2*exp(x)/(1+exp(x))-1`’

Hyperparameter ‘theta6’ hyperid = ‘42006’

name = ‘pacf5’

short.name = ‘pacf5’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘pc.cor0’

param = ‘0.5 0.1’

to.theta = ‘`function(x) log((1+x)/(1-x))`’

from.theta = ‘`function(x) 2*exp(x)/(1+exp(x))-1`’

Hyperparameter ‘theta7’ hyperid = ‘42007’

name = ‘pacf6’

short.name = ‘pacf6’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘pc.cor0’

param = ‘0.5 0.1’

to.theta = ‘`function(x) log((1+x)/(1-x))`’

from.theta = ‘function(x) 2*exp(x)/(1+exp(x))-1’

Hyperparameter ‘theta8’ hyperid = ‘42008’

name = ‘pacf7’

short.name = ‘pacf7’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘pc.cor0’

param = ‘0.5 0.1’

to.theta = ‘function(x) log((1+x)/(1-x))’

from.theta = ‘function(x) 2*exp(x)/(1+exp(x))-1’

Hyperparameter ‘theta9’ hyperid = ‘42009’

name = ‘pacf8’

short.name = ‘pacf8’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘pc.cor0’

param = ‘0.5 0.1’

to.theta = ‘function(x) log((1+x)/(1-x))’

from.theta = ‘function(x) 2*exp(x)/(1+exp(x))-1’

Hyperparameter ‘theta10’ hyperid = ‘42010’

name = ‘pacf9’

short.name = ‘pacf9’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘pc.cor0’

param = ‘0.5 0.1’

to.theta = ‘function(x) log((1+x)/(1-x))’

from.theta = ‘function(x) 2*exp(x)/(1+exp(x))-1’

Hyperparameter ‘theta11’ hyperid = ‘42011’

name = ‘pacf10’

short.name = ‘pacf10’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘pc.cor0’

param = ‘0.5 0.1’

to.theta = ‘function(x) log((1+x)/(1-x))’

from.theta = ‘function(x) 2*exp(x)/(1+exp(x))-1’

Properties: doc = ‘AR(p) correlations’

Model ‘rw1’. Number of hyperparmeters are 1.

Hyperparameter ‘theta’ hyperid = ‘43001’

name = ‘log precision’

short.name = ‘prec’

prior = ‘loggamma’

param = ‘1 5e-05’

initial = ‘0’

fixed = 'TRUE'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`
Properties: doc = 'Random walk of order 1'
Model 'rw2'. Number of hyperparmeters are 1.
Hyperparameter 'theta' hyperid = '44001'
name = 'log precision'
short.name = 'prec'
prior = 'loggamma'
param = '1 5e-05'
initial = '0'
fixed = 'TRUE'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`
Properties: doc = 'Random walk of order 2'
Model 'besag'. Number of hyperparmeters are 1.
Hyperparameter 'theta' hyperid = '45001'
name = 'log precision'
short.name = 'prec'
prior = 'loggamma'
param = '1 5e-05'
initial = '0'
fixed = 'TRUE'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`
Properties: doc = 'Besag model'
Model 'iid'. Number of hyperparmeters are 1.
Hyperparameter 'theta' hyperid = '46001'
name = 'log precision'
short.name = 'prec'
prior = 'loggamma'
param = '1 5e-05'
initial = '0'
fixed = 'TRUE'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`
Properties: doc = 'Independent model'
Section 'mix'. Valid models in this section are:
Model 'gaussian'. Number of hyperparmeters are 1.
Hyperparameter 'theta' hyperid = '47001'
name = 'log precision'
short.name = 'prec'
prior = 'loggamma'
param = '1 0.01'
initial = '0'

fixed = 'FALSE'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`
Properties: doc = 'Gaussian mixture'

Section 'link'. Valid models in this section are:

Model 'default'. Number of hyperparmeters are 0.
Model 'cloglog'. Number of hyperparmeters are 0.
Model 'loglog'. Number of hyperparmeters are 0.
Model 'identity'. Number of hyperparmeters are 0.
Model 'log'. Number of hyperparmeters are 0.
Model 'neglog'. Number of hyperparmeters are 0.
Model 'logit'. Number of hyperparmeters are 0.
Model 'probit'. Number of hyperparmeters are 0.
Model 'cauchit'. Number of hyperparmeters are 0.
Model 'tan'. Number of hyperparmeters are 0.
Model 'sslogit'. Number of hyperparmeters are 2.
Hyperparameter 'theta1' hyperid = '48001'
name = 'sensitivity'
short.name = 'sens'
prior = 'logitbeta'
param = '10 5'
initial = '1'
fixed = 'FALSE'
to.theta = `'function(x) log(x/(1-x))'`
from.theta = `'function(x) exp(x)/(1+exp(x))'`
Hyperparameter 'theta2' hyperid = '48002'
name = 'specificity'
short.name = 'spec'
prior = 'logitbeta'
param = '10 5'
initial = '1'
fixed = 'FALSE'
to.theta = `'function(x) log(x/(1-x))'`
from.theta = `'function(x) exp(x)/(1+exp(x))'`
Properties: doc = 'Logit link with sensitivity and specificity'
pdf = 'NA'
Model 'logoffset'. Number of hyperparmeters are 1.
Hyperparameter 'theta' hyperid = '49001'
name = 'beta'
short.name = 'b'
prior = 'normal'
param = '0 100'
initial = '0'
fixed = 'TRUE'
to.theta = `'function(x) log(x)'`

from.theta = ‘`function(x) exp(x)`’

Properties: doc = ‘Log-link with an offset’

pdf = ‘logoffset’

Model ‘logitoffset’. Number of hyperparmeters are 1.

Hyperparameter ‘theta’ hyperid = ‘49011’

name = ‘prob’

short.name = ‘p’

prior = ‘normal’

param = ‘-1 100’

initial = ‘-1’

fixed = ‘FALSE’

to.theta = ‘`function(x) log(x/(1-x))`’

from.theta = ‘`function(x) exp(x)/(1+exp(x))`’

Properties: doc = ‘Logit-link with an offset’

status = ‘experimental’

pdf = ‘logitoffset’

Model ‘test1’. Number of hyperparmeters are 1.

Hyperparameter ‘theta’ hyperid = ‘50001’

name = ‘beta’

short.name = ‘b’

prior = ‘normal’

param = ‘0 100’

initial = ‘0’

fixed = ‘FALSE’

to.theta = ‘`function(x) x`’

from.theta = ‘`function(x) x`’

Properties: doc = ‘A test1-link function (experimental)’

pdf = ‘NA’

Model ‘special1’. Number of hyperparmeters are 11.

Hyperparameter ‘theta1’ hyperid = ‘51001’

name = ‘log precision’

short.name = ‘prec’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘loggamma’

param = ‘1 1’

to.theta = ‘`function(x) x`’

from.theta = ‘`function(x) x`’

Hyperparameter ‘theta2’ hyperid = ‘51002’

name = ‘beta1’

short.name = ‘beta1’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘mvnorm’

param = ‘0 100’

to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta3’ hyperid = ‘51003’
name = ‘beta2’
short.name = ‘beta2’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta4’ hyperid = ‘51004’
name = ‘beta3’
short.name = ‘beta3’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta5’ hyperid = ‘51005’
name = ‘beta4’
short.name = ‘beta4’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta6’ hyperid = ‘51006’
name = ‘beta5’
short.name = ‘beta5’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta7’ hyperid = ‘51007’
name = ‘beta6’
short.name = ‘beta6’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’

to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta8’ hyperid = ‘51008’
name = ‘beta7’
short.name = ‘beta7’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta9’ hyperid = ‘51009’
name = ‘beta8’
short.name = ‘beta8’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta10’ hyperid = ‘51010’
name = ‘beta9’
short.name = ‘beta9’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta11’ hyperid = ‘51011’
name = ‘beta10’
short.name = ‘beta10’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘none’
param = ‘’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Properties: doc = ‘A special1-link function (experimental)’
pdf = ‘NA’
Model ‘special2’. Number of hyperparmeters are 1.
Hyperparameter ‘theta’ hyperid = ‘52001’
name = ‘beta’
short.name = ‘b’
prior = ‘normal’

param = ‘0 10’
initial = ‘0’
fixed = ‘FALSE’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Properties: doc = ‘A special2-link function (experimental)’
pdf = ‘NA’

Section ‘predictor’. Valid models in this section are:

Model ‘predictor’. Number of hyperparmeters are 1.

Hyperparameter ‘theta’ hyperid = ‘53001’
name = ‘log precision’
short.name = ‘prec’
initial = ‘12’
fixed = ‘TRUE’
prior = ‘loggamma’
param = ‘1 1e-05’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’

Properties: doc = ‘(not used)’

Section ‘hazard’. Valid models in this section are:

Model ‘rw1’. Number of hyperparmeters are 1.

Hyperparameter ‘theta’ hyperid = ‘54001’
name = ‘log precision’
short.name = ‘prec’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’

Properties: doc = ‘A random walk of order 1 for the log-hazard’

Model ‘rw2’. Number of hyperparmeters are 1.

Hyperparameter ‘theta’ hyperid = ‘55001’
name = ‘log precision’
short.name = ‘prec’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’

Properties: doc = ‘A random walk of order 2 for the log-hazard’

Section ‘likelihood’. Valid models in this section are:

Model ‘poisson’. Number of hyperparmeters are 0.

Model 'qpoisson'. Number of hyperparmeters are 0.

Model 'cenpoisson'. Number of hyperparmeters are 0.

Model 'gpoisson'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '56001'

name = 'overdispersion'

short.name = 'phi'

initial = '0'

fixed = 'FALSE'

prior = 'loggamma'

param = '1 1'

to.theta = '`function(x) log(x)`'

from.theta = '`function(x) exp(x)`'

Hyperparameter 'theta2' hyperid = '56002'

name = 'p'

short.name = 'p'

initial = '1'

fixed = 'TRUE'

prior = 'normal'

param = '1 100'

to.theta = '`function(x) x`'

from.theta = '`function(x) x`'

Properties: doc = 'The generalized Poisson likelihood'

survival = 'FALSE'

discrete = 'TRUE'

link = 'default log logoffset'

pdf = 'gpoisson'

status = 'experimental'

Model 'binomial'. Number of hyperparmeters are 0.

Model 'testbinomial1'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '57001'

name = 'sensitivity'

short.name = 's'

initial = '3'

fixed = 'FALSE'

prior = 'logitbeta'

param = '2 1'

to.theta = '`function(x) log(x/(1-x))`'

from.theta = '`function(x) exp(x)/(1+exp(x))`'

Hyperparameter 'theta2' hyperid = '57002'

name = 'specificity'

short.name = 'e'

initial = '3'

fixed = 'FALSE'

prior = 'logitbeta'

param = '2 1'

to.theta = ‘`function(x) log(x/(1-x))`’
from.theta = ‘`function(x) exp(x)/(1+exp(x))`’
Properties: doc = ‘(experimental)’
status = ‘experimental’
survival = ‘FALSE’
discrete = ‘TRUE’
link = ‘default logit cauchit probit cloglog loglog log’
pdf = ‘testbinomial1’
Model ‘gamma’. Number of hyperparmeters are 1.
Hyperparameter ‘theta’ hyperid = ‘58001’
name = ‘precision parameter’
short.name = ‘prec’
initial = ‘4.60517018598809’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 0.01’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’
Properties: doc = ‘The Gamma likelihood’
survival = ‘FALSE’
discrete = ‘FALSE’
link = ‘default log’
pdf = ‘gamma’
Model ‘gammacount’. Number of hyperparmeters are 1.
Hyperparameter ‘theta’ hyperid = ‘59001’
name = ‘log alpha’
short.name = ‘alpha’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘pc.gammacount’
param = ‘3’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’
Properties: doc = ‘A Gamma generalisation of the Poisson likelihood’
survival = ‘FALSE’
discrete = ‘FALSE’
link = ‘default log’
status = ‘experimental’
pdf = ‘gammacount’
Model ‘qkumar’. Number of hyperparmeters are 1.
Hyperparameter ‘theta’ hyperid = ‘60001’
name = ‘precision parameter’
short.name = ‘prec’
initial = ‘0’
fixed = ‘FALSE’

prior = 'loggamma'
param = '1 0.001'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'
Properties: doc = 'A quantile version of the Kumar likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit cauchit'
pdf = 'qkumar'
Model 'qloglogistic'. Number of hyperparmeters are 1.
Hyperparameter 'theta' hyperid = '60011'
name = 'precision parameter'
short.name = 'prec'
initial = '2'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 0.001'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'
Properties: doc = 'A quantile version of the logistic likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'qloglogistic'
Model 'beta'. Number of hyperparmeters are 1.
Hyperparameter 'theta' hyperid = '61001'
name = 'precision parameter'
short.name = 'phi'
initial = '2.30258509299405'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 0.1'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'
Properties: doc = 'The Beta likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit cauchit probit cloglog loglog'
pdf = 'beta'
Model 'betabinomial'. Number of hyperparmeters are 1.
Hyperparameter 'theta' hyperid = '62001'
name = 'overdispersion'
short.name = 'rho'
initial = '0'
fixed = 'FALSE'

prior = 'gaussian'
param = '0 0.4'
to.theta = 'function(x) log(x/(1-x))'
from.theta = 'function(x) exp(x)/(1+exp(x))'

Properties: **doc =** 'The Beta-Binomial likelihood'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit cauchit probit cloglog loglog'
pdf = 'betabinomial'

Model 'cbinomial'. Number of hyperparmeters are 0.

Model 'nbinomial'. Number of hyperparmeters are 1.

Hyperparameter 'theta' **hyperid =** '63001'
name = 'size'
short.name = 'size'
initial = '2.30258509299405'
fixed = 'FALSE'
prior = 'pc.mgamma'
param = '7'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Properties: **doc =** 'The negBinomial likelihood'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default log logoffset'
pdf = 'nbinomial'

Model 'simplex'. Number of hyperparmeters are 1.

Hyperparameter 'theta' **hyperid =** '64001'
name = 'log precision'
short.name = 'prec'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Properties: **doc =** 'The simplex likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit cauchit probit cloglog loglog'
pdf = 'simplex'

Model 'gaussian'. Number of hyperparmeters are 1.

Hyperparameter 'theta' **hyperid =** '65001'
name = 'log precision'
short.name = 'prec'
initial = '4'

fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'
Properties: doc = 'The Gaussian likelihoood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity logit cauchit log logoffset'
pdf = 'gaussian'
Model 'normal'. Number of hyperparmeters are 1.
Hyperparameter 'theta' hyperid = '66001'
name = 'log precision'
short.name = 'prec'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'
Properties: doc = 'The Gaussian likelihoood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity'
pdf = 'gaussian'
Model 'circularnormal'. Number of hyperparmeters are 1.
Hyperparameter 'theta' hyperid = '67001'
name = 'log precision parameter'
short.name = 'prec'
initial = '2'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 0.01'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'
Properties: doc = 'The circular Gaussian likelihoood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default tan'
pdf = 'circular-normal'
status = 'experimental'
Model 'wrappedcauchy'. Number of hyperparmeters are 1.
Hyperparameter 'theta' hyperid = '68001'
name = 'log precision parameter'
short.name = 'prec'

initial = '2'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 0.005'
to.theta = `'function(x) log(x/(1-x))'`
from.theta = `'function(x) exp(x)/(1+exp(x))'`

Properties: doc = 'The wrapped Cauchy likelihoood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default tan'
pdf = 'wrapped-cauchy'
status = 'disabled'

Model 'iidgamma'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '69001'
name = 'logshape'
short.name = 'shape'
initial = '0'
fixed = 'FALSE'
prior = 'loggamma'
param = '100 100'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta2' hyperid = '69002'
name = 'lograte'
short.name = 'rate'
initial = '0'
fixed = 'FALSE'
prior = 'loggamma'
param = '100 100'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Properties: doc = '(experimental)'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity'
pdf = 'iidgamma'
status = 'experimental'

Model 'iidlogitbeta'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '70001'
name = 'log.a'
short.name = 'a'
initial = '1'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 1'

to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’
Hyperparameter ‘theta2’ hyperid = ‘70002’
name = ‘log.b’
short.name = ‘b’
initial = ‘1’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 1’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’
Properties: doc = ‘(experimental)’
survival = ‘FALSE’
discrete = ‘FALSE’
link = ‘default logit’
pdf = ‘iidlogitbeta’
status = ‘experimental’

Model ‘loggammafrailty’. Number of hyperparmeters are 1.
Hyperparameter ‘theta’ hyperid = ‘71001’
name = ‘log precision’
short.name = ‘prec’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’
Properties: doc = ‘(experimental)’
survival = ‘FALSE’
discrete = ‘FALSE’
link = ‘default identity’
pdf = ‘loggammafrailty’
status = ‘experimental’

Model ‘logistic’. Number of hyperparmeters are 1.
Hyperparameter ‘theta’ hyperid = ‘72001’
name = ‘log precision’
short.name = ‘prec’
initial = ‘1’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’
Properties: doc = ‘The Logistic likelihoood’
survival = ‘FALSE’

discrete = 'FALSE'
link = 'default identity'
pdf = 'logistic'

Model 'skewnormal'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '73001'
name = 'inverse.scale'
short.name = 'iscale'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'

Hyperparameter 'theta2' hyperid = '73002'
name = 'skewness'
short.name = 'skew'
initial = '4'
fixed = 'FALSE'
prior = 'gaussian'
param = '0 10'

Properties: doc = 'The Skew-Normal likelihoood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity'
pdf = 'sn'

Model 'sn'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '74001'
name = 'log inverse scale'
short.name = 'iscale'
initial = '4'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'

Hyperparameter 'theta2' hyperid = '74002'
name = 'logit skewness'
short.name = 'skew'
initial = '0'
fixed = 'FALSE'
prior = 'gaussian'
param = '0 10'
to.theta = '`function(x, shape.max = 1) log((1+x/shape.max)/(1-x/shape.max))`'
from.theta = '`function(x, shape.max = 1) shape.max*(2*exp(x)/(1+exp(x))-1)`'

Properties: doc = 'The Skew-Normal likelihoood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity'
pdf = 'sn'

Model 'sn2'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '75001'

name = 'log precision'

short.name = 'prec'

initial = '1'

fixed = 'FALSE'

prior = 'loggamma'

param = '1 5e-05'

Hyperparameter 'theta2' hyperid = '75002'

name = 'logit skewness'

short.name = 'skew'

initial = '0'

fixed = 'FALSE'

prior = 'gaussian'

param = '0 10'

to.theta = '`function(x) log((1+x)/(1-x))`'

from.theta = '`function(x) (2*exp(x)/(1+exp(x))-1)`'

Properties: doc = 'The Skew-Normal likelihoood (alt param)'

survival = 'FALSE'

discrete = 'FALSE'

link = 'default identity'

status = 'experimental'

pdf = 'sn2'

Model 'gev'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '76001'

name = 'log precision'

short.name = 'prec'

initial = '4'

fixed = 'FALSE'

prior = 'loggamma'

param = '1 5e-05'

to.theta = '`function(x) log(x)`'

from.theta = '`function(x) exp(x)`'

Hyperparameter 'theta2' hyperid = '76002'

name = 'gev parameter'

short.name = 'gev'

initial = '0'

fixed = 'FALSE'

prior = 'gaussian'

param = '0 25'

to.theta = '`function(x) x`'

from.theta = '`function(x) x`'

Properties: doc = 'The Generalized Extreme Value likelihood'

survival = 'FALSE'

discrete = 'FALSE'

link = ‘default identity’
status = ‘experimental’
pdf = ‘gev’

Model ‘lognormal’. Number of hyperparmeters are 1.
Hyperparameter ‘theta’ hyperid = ‘77101’
name = ‘log precision’
short.name = ‘prec’
initial = ‘4’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’
Properties: doc = ‘The log-Normal likelihood’
survival = ‘FALSE’
discrete = ‘FALSE’
link = ‘default identity’
pdf = ‘lognormal’

Model ‘lognormalsurv’. Number of hyperparmeters are 1.
Hyperparameter ‘theta’ hyperid = ‘78001’
name = ‘log precision’
short.name = ‘prec’
initial = ‘2’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’
Properties: doc = ‘The log-Normal likelihood (survival version)’
survival = ‘TRUE’
discrete = ‘FALSE’
link = ‘default identity’
pdf = ‘lognormal’

Model ‘exponential’. Number of hyperparmeters are 0.

Model ‘exponentialsurv’. Number of hyperparmeters are 0.

Model ‘coxph’. Number of hyperparmeters are 0.

Model ‘weibull’. Number of hyperparmeters are 1.
Hyperparameter ‘theta’ hyperid = ‘79001’
name = ‘log alpha’
short.name = ‘alpha’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘25 25’
to.theta = ‘`function(x) log(x)`’

from.theta = ‘`function(x) exp(x)`’

Properties: doc = ‘The Weibull likelihood’

survival = ‘FALSE’

discrete = ‘FALSE’

link = ‘default log’

pdf = ‘weibull’

Model ‘weibullsurv’. Number of hyperparmeters are 1.

Hyperparameter ‘theta’ hyperid = ‘79101’

name = ‘log alpha’

short.name = ‘alpha’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘loggamma’

param = ‘25 25’

to.theta = ‘`function(x) log(x)`’

from.theta = ‘`function(x) exp(x)`’

Properties: doc = ‘The Weibull likelihood (survival version)’

survival = ‘TRUE’

discrete = ‘FALSE’

link = ‘default log neglog’

pdf = ‘weibull’

Model ‘loglogistic’. Number of hyperparmeters are 1.

Hyperparameter ‘theta’ hyperid = ‘80001’

name = ‘log alpha’

short.name = ‘alpha’

initial = ‘1’

fixed = ‘FALSE’

prior = ‘loggamma’

param = ‘25 25’

to.theta = ‘`function(x) log(x)`’

from.theta = ‘`function(x) exp(x)`’

Properties: doc = ‘The log-logistic likelihood (survival version)’

survival = ‘TRUE’

discrete = ‘FALSE’

link = ‘default log neglog’

pdf = ‘loglogistic’

Model ‘weibullcure’. Number of hyperparmeters are 2.

Hyperparameter ‘theta1’ hyperid = ‘81001’

name = ‘log alpha’

short.name = ‘a’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘loggamma’

param = ‘25 25’

to.theta = ‘`function(x) log(x)`’

from.theta = ‘`function(x) exp(x)`’

Hyperparameter ‘theta2’ hyperid = ‘81002’

name = ‘logit probability’

short.name = ‘prob’

initial = ‘-1’

fixed = ‘FALSE’

prior = ‘gaussian’

param = ‘-1 0.2’

to.theta = ‘`function(x) log(x/(1-x))`’

from.theta = ‘`function(x) exp(x)/(1+exp(x))`’

Properties: doc = ‘The Weibull-cure likelihood (survival version)’

survival = ‘TRUE’

discrete = ‘FALSE’

link = ‘default log neglog’

pdf = ‘weibullcure’

Model ‘stochvol’. Number of hyperparmeters are 1.

Hyperparameter ‘theta’ hyperid = ‘82001’

name = ‘log precision’

short.name = ‘prec’

initial = ‘500’

fixed = ‘TRUE’

prior = ‘loggamma’

param = ‘1 0.005’

to.theta = ‘`function(x) log(x)`’

from.theta = ‘`function(x) exp(x)`’

Properties: doc = ‘The Gaussian stochvol likelihood’

survival = ‘FALSE’

discrete = ‘FALSE’

link = ‘default log’

pdf = ‘stochvolgaussian’

Model ‘stochvolt’. Number of hyperparmeters are 1.

Hyperparameter ‘theta’ hyperid = ‘83001’

name = ‘log degrees of freedom’

short.name = ‘dof’

initial = ‘4’

fixed = ‘FALSE’

prior = ‘pc.dof’

param = ‘15 0.5’

to.theta = ‘`function(x) log(x-2)`’

from.theta = ‘`function(x) 2+exp(x)`’

Properties: doc = ‘The Student-t stochvol likelihood’

survival = ‘FALSE’

discrete = ‘FALSE’

link = ‘default log’

pdf = ‘stochvolt’

Model 'stochvolnig'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '84001'

name = 'skewness'

short.name = 'skew'

initial = '0'

fixed = 'FALSE'

prior = 'gaussian'

param = '0 10'

to.theta = `'function(x) x'`

from.theta = `'function(x) x'`

Hyperparameter 'theta2' hyperid = '84002'

name = 'shape'

short.name = 'shape'

initial = '0'

fixed = 'FALSE'

prior = 'loggamma'

param = '1 0.5'

to.theta = `'function(x) log(x-1)'`

from.theta = `'function(x) 1+exp(x)'`

Properties: doc = 'The Normal inverse Gaussian stochvol likelihood'

survival = 'FALSE'

discrete = 'FALSE'

link = 'default log'

pdf = 'stochvolnig'

Model 'zeroinflatedpoisson0'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '85001'

name = 'logit probability'

short.name = 'prob'

initial = '-1'

fixed = 'FALSE'

prior = 'gaussian'

param = '-1 0.2'

to.theta = `'function(x) log(x/(1-x))'`

from.theta = `'function(x) exp(x)/(1+exp(x))'`

Properties: doc = 'Zero-inflated Poisson, type 0'

survival = 'FALSE'

discrete = 'FALSE'

link = 'default log'

pdf = 'zeroinflated'

Model 'zeroinflatedpoisson1'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '86001'

name = 'logit probability'

short.name = 'prob'

initial = '-1'

fixed = 'FALSE'

prior = 'gaussian'
param = '-1 0.2'
to.theta = `'function(x) log(x/(1-x))'`
from.theta = `'function(x) exp(x)/(1+exp(x))'`
Properties: doc = 'Zero-inflated Poisson, type 1'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'

Model 'zeroinflatedpoisson2'. Number of hyperparmeters are 1.
Hyperparameter 'theta' hyperid = '87001'
name = 'log alpha'
short.name = 'a'
initial = '0.693147180559945'
fixed = 'FALSE'
prior = 'gaussian'
param = '0.693147180559945 1'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`
Properties: doc = 'Zero-inflated Poisson, type 2'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'

Model 'zeroinflatedbetabinomial0'. Number of hyperparmeters are 2.
Hyperparameter 'theta1' hyperid = '88001'
name = 'overdispersion'
short.name = 'rho'
initial = '0'
fixed = 'FALSE'
prior = 'gaussian'
param = '0 0.4'
to.theta = `'function(x) log(x/(1-x))'`
from.theta = `'function(x) exp(x)/(1+exp(x))'`
Hyperparameter 'theta2' hyperid = '88002'
name = 'logit probability'
short.name = 'prob'
initial = '-1'
fixed = 'FALSE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = `'function(x) log(x/(1-x))'`
from.theta = `'function(x) exp(x)/(1+exp(x))'`
Properties: doc = 'Zero-inflated Beta-Binomial, type 0'
survival = 'FALSE'

discrete = 'TRUE'
link = 'default logit cauchit probit cloglog loglog'
pdf = 'zeroinflated'

Model 'zeroinflatedbetabinomial1'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '89001'
name = 'overdispersion'
short.name = 'rho'
initial = '0'
fixed = 'FALSE'
prior = 'gaussian'
param = '0 0.4'
to.theta = '`function(x) log(x/(1-x))`'
from.theta = '`function(x) exp(x)/(1+exp(x))`'

Hyperparameter 'theta2' hyperid = '89002'
name = 'logit probability'
short.name = 'prob'
initial = '-1'
fixed = 'FALSE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = '`function(x) log(x/(1-x))`'
from.theta = '`function(x) exp(x)/(1+exp(x))`'

Properties: doc = 'Zero-inflated Beta-Binomial, type 1'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit cauchit probit cloglog loglog'
pdf = 'zeroinflated'

Model 'zeroinflatedbinomial0'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '90001'
name = 'logit probability'
short.name = 'prob'
initial = '-1'
fixed = 'FALSE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = '`function(x) log(x/(1-x))`'
from.theta = '`function(x) exp(x)/(1+exp(x))`'

Properties: doc = 'Zero-inflated Binomial, type 0'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit cauchit probit cloglog loglog'
pdf = 'zeroinflated'

Model 'zeroinflatedbinomial1'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '91001'
name = 'logit probability'

short.name = 'prob'
initial = '-1'
fixed = 'FALSE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = `'function(x) log(x/(1-x))'`
from.theta = `'function(x) exp(x)/(1+exp(x))'`

Properties: doc = 'Zero-inflated Binomial, type 1'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit cauchit probit cloglog loglog'
pdf = 'zeroinflated'

Model 'zeroinflatedbinomial2'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '92001'
name = 'alpha'
short.name = 'alpha'
initial = '-1'
fixed = 'FALSE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Properties: doc = 'Zero-inflated Binomial, type 2'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit cauchit probit cloglog loglog'
pdf = 'zeroinflated'

Model 'zeroninflatedbinomial2'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '93001'
name = 'alpha1'
short.name = 'alpha1'
initial = '-1'
fixed = 'FALSE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta2' hyperid = '93002'
name = 'alpha2'
short.name = 'alpha2'
initial = '-1'
fixed = 'FALSE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = `'function(x) log(x)'`

from.theta = ‘function(x) exp(x)’

Properties: doc = ‘Zero and N inflated binomial, type 2’

survival = ‘FALSE’

discrete = ‘FALSE’

link = ‘default logit cauchit probit cloglog loglog’

pdf = ‘NA’

Model ‘zeroninflatedbinomial3’. Number of hyperparmeters are 2.

Hyperparameter ‘theta1’ hyperid = ‘93101’

name = ‘alpha0’

short.name = ‘alpha0’

initial = ‘1’

fixed = ‘FALSE’

prior = ‘loggamma’

param = ‘1 1’

to.theta = ‘function(x) log(x)’

from.theta = ‘function(x) exp(x)’

Hyperparameter ‘theta2’ hyperid = ‘93102’

name = ‘alphaN’

short.name = ‘alphaN’

initial = ‘1’

fixed = ‘FALSE’

prior = ‘loggamma’

param = ‘1 1’

to.theta = ‘function(x) log(x)’

from.theta = ‘function(x) exp(x)’

Properties: doc = ‘Zero and N inflated binomial, type 3’

status = ‘experimental’

survival = ‘FALSE’

discrete = ‘FALSE’

link = ‘default logit cauchit probit cloglog loglog’

pdf = ‘zeroinflated’

Model ‘zeroinflatedbetabinomial2’. Number of hyperparmeters are 2.

Hyperparameter ‘theta1’ hyperid = ‘94001’

name = ‘log alpha’

short.name = ‘a’

initial = ‘0.693147180559945’

fixed = ‘FALSE’

prior = ‘gaussian’

param = ‘0.693147180559945 1’

to.theta = ‘function(x) log(x)’

from.theta = ‘function(x) exp(x)’

Hyperparameter ‘theta2’ hyperid = ‘94002’

name = ‘beta’

short.name = ‘b’

initial = ‘0’

fixed = 'FALSE'
prior = 'gaussian'
param = '0 1'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Properties: **doc =** 'Zero inflated Beta-Binomial, type 2'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit cauchit probit cloglog loglog'
pdf = 'zeroinflated'

Model 'zeroinflatednbinomial0'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '95001'
name = 'log size'
short.name = 'size'
initial = '2.30258509299405'
fixed = 'FALSE'
prior = 'pc.mgamma'
param = '7'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta2' hyperid = '95002'
name = 'logit probability'
short.name = 'prob'
initial = '-1'
fixed = 'FALSE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = 'function(x) log(x/(1-x))'
from.theta = 'function(x) exp(x)/(1+exp(x))'

Properties: **doc =** 'Zero inflated negBinomial, type 0'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'

Model 'zeroinflatednbinomial1'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '96001'
name = 'log size'
short.name = 'size'
initial = '2.30258509299405'
fixed = 'FALSE'
prior = 'pc.mgamma'
param = '7'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta2' hyperid = '96002'

name = 'logit probability'
short.name = 'prob'
initial = '-1'
fixed = 'FALSE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = `'function(x) log(x/(1-x))'`
from.theta = `'function(x) exp(x)/(1+exp(x))'`

Properties: doc = 'Zero inflated negBinomial, type 1'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'

Model 'zeroinflatednbinomial1strata2'. Number of hyperparmeters are 11.

Hyperparameter 'theta1' hyperid = '97001'
name = 'log size'
short.name = 'size'
initial = '2.30258509299405'
fixed = 'FALSE'
prior = 'pc.mgamma'
param = '7'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta2' hyperid = '97002'
name = 'logit probability 1'
short.name = 'prob1'
initial = '-1'
fixed = 'FALSE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = `'function(x) log(x/(1-x))'`
from.theta = `'function(x) exp(x)/(1+exp(x))'`

Hyperparameter 'theta3' hyperid = '97003'
name = 'logit probability 2'
short.name = 'prob2'
initial = '-1'
fixed = 'FALSE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = `'function(x) log(x/(1-x))'`
from.theta = `'function(x) exp(x)/(1+exp(x))'`

Hyperparameter 'theta4' hyperid = '97004'
name = 'logit probability 3'
short.name = 'prob3'
initial = '-1'

fixed = ‘TRUE’
prior = ‘gaussian’
param = ‘-1 0.2’
to.theta = ‘function(x) log(x/(1-x))’
from.theta = ‘function(x) exp(x)/(1+exp(x))’
Hyperparameter ‘theta5’ hyperid = ‘97005’
name = ‘logit probability 4’
short.name = ‘prob4’
initial = ‘-1’
fixed = ‘TRUE’
prior = ‘gaussian’
param = ‘-1 0.2’
to.theta = ‘function(x) log(x/(1-x))’
from.theta = ‘function(x) exp(x)/(1+exp(x))’
Hyperparameter ‘theta6’ hyperid = ‘97006’
name = ‘logit probability 5’
short.name = ‘prob5’
initial = ‘-1’
fixed = ‘TRUE’
prior = ‘gaussian’
param = ‘-1 0.2’
to.theta = ‘function(x) log(x/(1-x))’
from.theta = ‘function(x) exp(x)/(1+exp(x))’
Hyperparameter ‘theta7’ hyperid = ‘97007’
name = ‘logit probability 6’
short.name = ‘prob6’
initial = ‘-1’
fixed = ‘TRUE’
prior = ‘gaussian’
param = ‘-1 0.2’
to.theta = ‘function(x) log(x/(1-x))’
from.theta = ‘function(x) exp(x)/(1+exp(x))’
Hyperparameter ‘theta8’ hyperid = ‘97008’
name = ‘logit probability 7’
short.name = ‘prob7’
initial = ‘-1’
fixed = ‘TRUE’
prior = ‘gaussian’
param = ‘-1 0.2’
to.theta = ‘function(x) log(x/(1-x))’
from.theta = ‘function(x) exp(x)/(1+exp(x))’
Hyperparameter ‘theta9’ hyperid = ‘97009’
name = ‘logit probability 8’
short.name = ‘prob8’
initial = ‘-1’

fixed = 'TRUE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = 'function(x) log(x/(1-x))'
from.theta = 'function(x) exp(x)/(1+exp(x))'
Hyperparameter 'theta10' hyperid = '97010'
name = 'logit probability 9'
short.name = 'prob9'
initial = '-1'
fixed = 'TRUE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = 'function(x) log(x/(1-x))'
from.theta = 'function(x) exp(x)/(1+exp(x))'
Hyperparameter 'theta11' hyperid = '97011'
name = 'logit probability 10'
short.name = 'prob10'
initial = '-1'
fixed = 'TRUE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = 'function(x) log(x/(1-x))'
from.theta = 'function(x) exp(x)/(1+exp(x))'
Properties: doc = 'Zero inflated negBinomial, type 1, strata 2'
status = 'experimental'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'
Model 'zeroinflatednbinomial1strata3'. Number of hyperparmeters are 11.
Hyperparameter 'theta1' hyperid = '98001'
name = 'logit probability'
short.name = 'prob'
initial = '-1'
fixed = 'FALSE'
prior = 'gaussian'
param = '-1 0.2'
to.theta = 'function(x) log(x/(1-x))'
from.theta = 'function(x) exp(x)/(1+exp(x))'
Hyperparameter 'theta2' hyperid = '98002'
name = 'log size 1'
short.name = 'size1'
initial = '2.30258509299405'
fixed = 'FALSE'
prior = 'pc.mgamma'

param = '7'
to.theta = '`function(x) log(x)`'
from.theta = '`function(x) exp(x)`'

Hyperparameter 'theta3' hyperid = '98003'
name = 'log size 2'
short.name = 'size2'
initial = '2.30258509299405'
fixed = 'FALSE'
prior = 'pc.mgamma'
param = '7'
to.theta = '`function(x) log(x)`'
from.theta = '`function(x) exp(x)`'

Hyperparameter 'theta4' hyperid = '98004'
name = 'log size 3'
short.name = 'size3'
initial = '2.30258509299405'
fixed = 'TRUE'
prior = 'pc.mgamma'
param = '7'
to.theta = '`function(x) log(x)`'
from.theta = '`function(x) exp(x)`'

Hyperparameter 'theta5' hyperid = '98005'
name = 'log size 4'
short.name = 'size4'
initial = '2.30258509299405'
fixed = 'TRUE'
prior = 'pc.mgamma'
param = '7'
to.theta = '`function(x) log(x)`'
from.theta = '`function(x) exp(x)`'

Hyperparameter 'theta6' hyperid = '98006'
name = 'log size 5'
short.name = 'size5'
initial = '2.30258509299405'
fixed = 'TRUE'
prior = 'pc.mgamma'
param = '7'
to.theta = '`function(x) log(x)`'
from.theta = '`function(x) exp(x)`'

Hyperparameter 'theta7' hyperid = '98007'
name = 'log size 6'
short.name = 'size6'
initial = '2.30258509299405'
fixed = 'TRUE'
prior = 'pc.mgamma'

param = '7'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta8' hyperid = '98008'
name = 'log size 7'
short.name = 'size7'
initial = '2.30258509299405'
fixed = 'TRUE'
prior = 'pc.mgamma'
param = '7'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta9' hyperid = '98009'
name = 'log size 8'
short.name = 'size8'
initial = '2.30258509299405'
fixed = 'TRUE'
prior = 'pc.mgamma'
param = '7'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta10' hyperid = '98010'
name = 'log size 9'
short.name = 'size9'
initial = '2.30258509299405'
fixed = 'TRUE'
prior = 'pc.mgamma'
param = '7'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta11' hyperid = '98011'
name = 'log size 10'
short.name = 'size10'
initial = '2.30258509299405'
fixed = 'TRUE'
prior = 'pc.mgamma'
param = '7'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Properties: doc = 'Zero inflated negBinomial, type 1, strata 3'
status = 'experimental'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'

Model 'zeroinflatednbinomial2'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '99001'

name = 'log size'

short.name = 'size'

initial = '2.30258509299405'

fixed = 'FALSE'

prior = 'pc.mgamma'

param = '7'

to.theta = `'function(x) log(x)'`

from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta2' hyperid = '99002'

name = 'log alpha'

short.name = 'a'

initial = '0.693147180559945'

fixed = 'FALSE'

prior = 'gaussian'

param = '2 1'

to.theta = `'function(x) log(x)'`

from.theta = `'function(x) exp(x)'`

Properties: doc = 'Zero inflated negBinomial, type 2'

survival = 'FALSE'

discrete = 'FALSE'

link = 'default log'

pdf = 'zeroinflated'

Model 't'. Number of hyperparmeters are 2.

Hyperparameter 'theta1' hyperid = '100001'

name = 'log precision'

short.name = 'prec'

initial = '0'

fixed = 'FALSE'

prior = 'loggamma'

param = '1 5e-05'

to.theta = `'function(x) log(x)'`

from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta2' hyperid = '100002'

name = 'log degrees of freedom'

short.name = 'dof'

initial = '5'

fixed = 'FALSE'

prior = 'pc.dof'

param = '15 0.5'

to.theta = `'function(x) log(x-2)'`

from.theta = `'function(x) 2+exp(x)'`

Properties: doc = 'Student-t likelihood'

survival = 'FALSE'

discrete = 'FALSE'
link = 'default identity'
pdf = 'student-t'

Model 'tstrata'. Number of hyperparmeters are 11.

Hyperparameter 'theta1' hyperid = '101001'
name = 'log degrees of freedom'
short.name = 'dof'
initial = '4'
fixed = 'FALSE'
prior = 'pc.dof'
param = '15 0.5'
to.theta = `'function(x) log(x-5)'`
from.theta = `'function(x) 5+exp(x)'`

Hyperparameter 'theta2' hyperid = '101002'
name = 'log precision1'
short.name = 'prec1'
initial = '2'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta3' hyperid = '101003'
name = 'log precision2'
short.name = 'prec2'
initial = '2'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta4' hyperid = '101004'
name = 'log precision3'
short.name = 'prec3'
initial = '2'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Hyperparameter 'theta5' hyperid = '101005'
name = 'log precision4'
short.name = 'prec4'
initial = '2'
fixed = 'FALSE'

prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta6' hyperid = '101006'
name = 'log precision5'
short.name = 'prec5'
initial = '2'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta7' hyperid = '101007'
name = 'log precision6'
short.name = 'prec6'
initial = '2'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta8' hyperid = '101008'
name = 'log precision7'
short.name = 'prec7'
initial = '2'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta9' hyperid = '101009'
name = 'log precision8'
short.name = 'prec8'
initial = '2'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 5e-05'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'

Hyperparameter 'theta10' hyperid = '101010'
name = 'log precision9'
short.name = 'prec9'
initial = '2'
fixed = 'FALSE'

prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’
Hyperparameter ‘theta11’ hyperid = ‘101011’
name = ‘log precision10’
short.name = ‘prec10’
initial = ‘2’
fixed = ‘FALSE’
prior = ‘loggamma’
param = ‘1 5e-05’
to.theta = ‘`function(x) log(x)`’
from.theta = ‘`function(x) exp(x)`’
Properties: doc = ‘A stratified version of the Student-t likelihood’
survival = ‘FALSE’
discrete = ‘FALSE’
link = ‘default identity’
pdf = ‘tstrata’
Model ‘nmix’. Number of hyperparmeters are 10.
Hyperparameter ‘theta1’ hyperid = ‘101101’
name = ‘beta1’
short.name = ‘beta1’
initial = ‘2.30258509299405’
fixed = ‘FALSE’
prior = ‘normal’
param = ‘0 0.5’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta2’ hyperid = ‘101102’
name = ‘beta2’
short.name = ‘beta2’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘normal’
param = ‘0 1’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’
Hyperparameter ‘theta3’ hyperid = ‘101103’
name = ‘beta3’
short.name = ‘beta3’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘normal’
param = ‘0 1’
to.theta = ‘`function(x) x`’

from.theta = ‘function(x) x’

Hyperparameter ‘theta4’ hyperid = ‘101104’

name = ‘beta4’

short.name = ‘beta4’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘normal’

param = ‘0 1’

to.theta = ‘function(x) x’

from.theta = ‘function(x) x’

Hyperparameter ‘theta5’ hyperid = ‘101105’

name = ‘beta5’

short.name = ‘beta5’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘normal’

param = ‘0 1’

to.theta = ‘function(x) x’

from.theta = ‘function(x) x’

Hyperparameter ‘theta6’ hyperid = ‘101106’

name = ‘beta6’

short.name = ‘beta6’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘normal’

param = ‘0 1’

to.theta = ‘function(x) x’

from.theta = ‘function(x) x’

Hyperparameter ‘theta7’ hyperid = ‘101107’

name = ‘beta7’

short.name = ‘beta7’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘normal’

param = ‘0 1’

to.theta = ‘function(x) x’

from.theta = ‘function(x) x’

Hyperparameter ‘theta8’ hyperid = ‘101108’

name = ‘beta8’

short.name = ‘beta8’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘normal’

param = ‘0 1’

to.theta = ‘function(x) x’

from.theta = ‘`function(x) x`’

Hyperparameter ‘theta9’ hyperid = ‘101109’

name = ‘beta9’

short.name = ‘beta9’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘normal’

param = ‘0 1’

to.theta = ‘`function(x) x`’

from.theta = ‘`function(x) x`’

Hyperparameter ‘theta10’ hyperid = ‘101110’

name = ‘beta10’

short.name = ‘beta10’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘normal’

param = ‘0 1’

to.theta = ‘`function(x) x`’

from.theta = ‘`function(x) x`’

Properties: doc = ‘Binomial-Poisson mixture’

status = ‘experimental’

survival = ‘FALSE’

discrete = ‘TRUE’

link = ‘default logit probit’

pdf = ‘nmix’

Model ‘nmixnb’. Number of hyperparmeters are 11.

Hyperparameter ‘theta1’ hyperid = ‘101121’

name = ‘beta1’

short.name = ‘beta1’

initial = ‘2.30258509299405’

fixed = ‘FALSE’

prior = ‘normal’

param = ‘0 0.5’

to.theta = ‘`function(x) x`’

from.theta = ‘`function(x) x`’

Hyperparameter ‘theta2’ hyperid = ‘101122’

name = ‘beta2’

short.name = ‘beta2’

initial = ‘0’

fixed = ‘FALSE’

prior = ‘normal’

param = ‘0 1’

to.theta = ‘`function(x) x`’

from.theta = ‘`function(x) x`’

Hyperparameter ‘theta3’ hyperid = ‘101123’

name = ‘beta3’
short.name = ‘beta3’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘normal’
param = ‘0 1’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta4’ hyperid = ‘101124’
name = ‘beta4’
short.name = ‘beta4’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘normal’
param = ‘0 1’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta5’ hyperid = ‘101125’
name = ‘beta5’
short.name = ‘beta5’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘normal’
param = ‘0 1’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta6’ hyperid = ‘101126’
name = ‘beta6’
short.name = ‘beta6’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘normal’
param = ‘0 1’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta7’ hyperid = ‘101127’
name = ‘beta7’
short.name = ‘beta7’
initial = ‘0’
fixed = ‘FALSE’
prior = ‘normal’
param = ‘0 1’
to.theta = ‘`function(x) x`’
from.theta = ‘`function(x) x`’

Hyperparameter ‘theta8’ hyperid = ‘101128’

name = 'beta8'
short.name = 'beta8'
initial = '0'
fixed = 'FALSE'
prior = 'normal'
param = '0 1'
to.theta = 'function(x) x'
from.theta = 'function(x) x'
Hyperparameter 'theta9' hyperid = '101129'
name = 'beta9'
short.name = 'beta9'
initial = '0'
fixed = 'FALSE'
prior = 'normal'
param = '0 1'
to.theta = 'function(x) x'
from.theta = 'function(x) x'
Hyperparameter 'theta10' hyperid = '101130'
name = 'beta10'
short.name = 'beta10'
initial = '0'
fixed = 'FALSE'
prior = 'normal'
param = '0 1'
to.theta = 'function(x) x'
from.theta = 'function(x) x'
Hyperparameter 'theta11' hyperid = '101131'
name = 'overdispersion'
short.name = 'overdispersion'
initial = '0'
fixed = 'FALSE'
prior = 'pc.gamma'
param = '7'
to.theta = 'function(x) log(x)'
from.theta = 'function(x) exp(x)'
Properties: doc = 'NegBinomial-Poisson mixture'
status = 'experimental'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit probit'
pdf = 'nmixnb'
Model 'gp'. Number of hyperparmeters are 1.
Hyperparameter 'theta' hyperid = '101201'
name = 'shape'
short.name = 'xi'

initial = '-2.30258509299405'
fixed = 'FALSE'
prior = 'loggamma'
param = '1 15'
to.theta = `'function(x) log(x)'`
from.theta = `'function(x) exp(x)'`

Properties: **doc =** 'Generalized Pareto likelihood'
status = 'experimental'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default log'
pdf = 'genPareto'

Model 'logperiodogram'. Number of hyperparmeters are 0.

Section 'prior'. Valid models in this section are:

Model 'normal'. Number of parameters in the prior = 2
Model 'gaussian'. Number of parameters in the prior = 2
Model 'wishart1d'. Number of parameters in the prior = 2
Model 'wishart2d'. Number of parameters in the prior = 4
Model 'wishart3d'. Number of parameters in the prior = 7
Model 'wishart4d'. Number of parameters in the prior = 11
Model 'wishart5d'. Number of parameters in the prior = 16
Model 'loggamma'. Number of parameters in the prior = 2
Model 'minuslogsqrtruncnormal'. Number of parameters in the prior = 2
Model 'logtnormal'. Number of parameters in the prior = 2
Model 'logtgaussian'. Number of parameters in the prior = 2
Model 'flat'. Number of parameters in the prior = 0
Model 'logflat'. Number of parameters in the prior = 0
Model 'logiflat'. Number of parameters in the prior = 0
Model 'mvnorm'. Number of parameters in the prior = -1
Model 'pc.ar'. Number of parameters in the prior = 1
Model 'none'. Number of parameters in the prior = 0
Model 'invalid'. Number of parameters in the prior = 0
Model 'betacorrelation'. Number of parameters in the prior = 2
Model 'logitbeta'. Number of parameters in the prior = 2
Model 'pc.prec'. Number of parameters in the prior = 2
Model 'pc.dof'. Number of parameters in the prior = 2
Model 'pc.cor0'. Number of parameters in the prior = 2
Model 'pc.cor1'. Number of parameters in the prior = 2
Model 'pc.fgnh'. Number of parameters in the prior = 2
Model 'pc.spde.GA'. Number of parameters in the prior = 4
Model 'pc.matern'. Number of parameters in the prior = 3
Model 'pc.range'. Number of parameters in the prior = 2
Model 'pc.gamma'. Number of parameters in the prior = 1
Model 'pc.mgamma'. Number of parameters in the prior = 1
Model 'pc.gammacount'. Number of parameters in the prior = 1

Model 'pc'. Number of parameters in the prior = 2

Model 'ref.ar'. Number of parameters in the prior = 0

Model 'jeffreystdf'. Number of parameters in the prior = 0

Model 'expression:'. Number of parameters in the prior = -1

Model 'table:'. Number of parameters in the prior = -1

Section 'wrapper'. Valid models in this section are:

Model 'joint'. Number of hyperparmeters are 1.

Hyperparameter 'theta' hyperid = '102001'

name = 'log precision'

short.name = 'prec'

initial = '0'

fixed = 'TRUE'

prior = 'loggamma'

param = '1 5e-05'

to.theta = '`function(x) log(x)`'

from.theta = '`function(x) exp(x)`'

Properties: doc = '(experimental)'

constr = 'FALSE'

nrow.ncol = 'FALSE'

augmented = 'FALSE'

aug.factor = '1'

aug.constr = 'NULL'

n.div.by = 'NULL'

n.required = 'FALSE'

set.default.values = 'FALSE'

pdf = 'NA'

Examples

```
## How to set hyperparameters to pass as the argument 'hyper'. This
## format is compatible with the old style (using 'initial', 'fixed',
## 'prior', 'param'), but the new style using 'hyper' take preceedence
## over the old style. The two styles can also be mixed. The old style
## might be removed from the code in the future...

## Only a subset need to be given
   hyper = list(theta = list(initial = 2))
## The `name' can be used instead of 'theta', or 'theta1', 'theta2',...
   hyper = list(precision = list(initial = 2))
   hyper = list(precision = list(prior = "flat", param = numeric(0)))
   hyper = list(theta2 = list(initial=3), theta1 = list(prior = "gaussian"))
## The 'short.name' can be used instead of 'name'
   hyper = list(rho = list(param = c(0,1)))
```

inla.nonconvex.hull *Nonconvex set extensions.*

Description

Constructs a nonconvex boundary for a point set using morphological operations.

Usage

```
inla.nonconvex.hull(points,
                    convex = -0.15,
                    concave = convex,
                    resolution = 40,
                    eps = NULL,
                    crs = NULL)

inla.nonconvex.hull.basic(points,
                          convex = -0.15,
                          resolution = 40,
                          eps = NULL,
                          crs = NULL)
```

Arguments

points	2D point coordinates (2-column matrix). Can alternatively be a SpatialPoints or SpatialPointsDataFrame object.
convex	The desired extension radius. Also determines the smallest allowed convex curvature radius. Negative values are interpreted as fractions of the approximate initial set diameter.
concave	The desired minimal concave curvature radius. Default is concave=convex.
resolution	The internal computation resolution. A warning will be issued when this needs to be increased for higher accuracy, with the required resolution stated.
eps	The polygonal curve simplification tolerance used for simplifying the resulting boundary curve. See inla.simplify.curve for details.
crs	An optional CRS or inla.CRS object

Details

Morphological dilation by convex, followed by closing by concave, with minimum concave curvature radius concave. If the dilated set has no gaps of width between

$$2convex(\sqrt{1 + 2concave/convex} - 1)$$

and $2concave$, then the minimum convex curvature radius is convex. Special case concave=0 delegates to inla.nonconvex.hull.basic

The implementation is based on the identity

$$dilation(a)\&closing(b) = dilation(a + b)\&erosion(b)$$

where all operations are with respect to disks with the specified radii.

Value

An `inla.mesh.segment` object.

Note

Requires `nndistF` from the `splancs` package.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

Examples

```
if (require(splancs)) {
  loc = matrix(runif(20), 10, 2)
  boundary = inla.nonconvex.hull(loc, convex=0.2)
  lines(boundary, add=FALSE)
  points(loc)
}
```

---

inla.option *Set and get global options for INLA*

Description

Set and get global options for INLA

Usage

```
inla.setOption(...)
inla.getOption(option)
```

Arguments

`...`	Option and value, like `option=value` or `option, value`; see the Examples
`option`	The option to get. If `option = NULL` then `inla.getOption` then `inla.getOption` will display the current defaults, otherwise, `option` must be one of inla.call: The path to the inla-program. inla.arg: Additional arguments to `inla.call` fmesher.call: The path to the fmesher-program fmesher.arg: Additional arguments to `fmesher.call` num.threads: Number of threads to use. keep: Keep temporary files? working.directory: The name of the working directory. silent: Run the inla-program in a silent mode? debug : Run the inla-program in a debug mode? internal.binary.mode : if `FALSE` the (some) output are in ascii format instead of binary format. Using this option, then `inla.collect.results` will fail (Expert mode)

internal.experimental.mode : Expert option

cygwin : The home of the Cygwin installation (default "C:/cygwin") [Remote computing for Windows only]

ssh.auth.sock: The ssh bind-adress (value of $SSH_AUTH_SOCK int the Cygwin-shell). [Remote computing for Windows only]

enable.inla.argument.weights : if `TRUE` the `inla` accepts argument `weights`

show.warning.graph.file : Give a warning for using the obsolete argument `graph.file` instead of `graph`

scale.model.default : The default value of argument `scale.model` which optionally scale intrinisic models to have generalized unit average variance

The options are stored in the variable `inla.options` in the `.GlobalEnv`-environment.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
## set number of threads
inla.setOption("num.threads", 2)
## alternative format
inla.setOption(num.threads=2)
## check it
inla.getOption("num.threads")
```

`inla.qstat` *Control and view a remote inla-queue*

Description

Control and view a remote inla-queue of submitted jobs

Usage

```
inla.qget(id, remove = TRUE)
inla.qdel(id)
inla.qstat(id)
inla.qlog(id)
inla.qnuke()
## S3 method for class 'inla.q'
summary(object,...)
## S3 method for class 'inla.q'
print(x,...)
```

Arguments

`id`	The job-id which is the output from `inla` when the job is submitted, the job-number or job-name. For `inla.qstat`, `id` is optional and if omitted all the jobs will be listed.
`remove`	Logical If FALSE, leave the job on the server after retrival, otherwise remove it (default).

`x`	An `inla.q`-object which is the output from `inla.qstat`
`object`	An `inla.q`-object which is the output from `inla.qstat`
`...`	other arguments.

Details

`inla.qstat` show job(s) on the server, `inla.qget` fetch the results (and by default remove the files on the server), `inla.qdel` removes a job on the server and `inla.qnuke` remove all jobs on the server. `inla.qlog` fetches the logfile only.

The recommended procedure is to use `r=inla(...,inla.call="submit")` and then do `r=inla.qget(r)` at a later stage. If the job is not finished, then `r` will not be overwritten and this step can be repeated. The reason for this procedure, is that some information usually stored in the result object does not go through the remote server, hence have to be appended to the results that are retrieved from the server. Hence doing `r=inla(..., inla.call="submit")` and then later retrive it using `r=inla.qget(1)`, say, then `r` does not contain all the usual information. All the main results are there, but administrative information which is required to call `inla.hyperpar` or `inla.rerun` are not there.

Value

`inla.qstat` returns an `inla.q`-object with information about current jobs.

Author(s)

Havard Rue

See Also

`inla`

Examples

```
## Not run:
r = inla(y~1, data = data.frame(y=rnorm(10)), inla.call="submit")
inla.qstat()
r = inla.qget(r, remove=FALSE)
inla.qdel(1)
inla.qnuke()

## End(Not run)
```

`inla.reorderings` *Reorderings methods for sparse matrices*

Description

Provide the names of all implemented reordering schemes

Usage

```
inla.reorderings()
```

Arguments

None

Value

The names of all available reorderings

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
inla.reorderings()
```

---

inla.rerun *Rerun an analysis*

Description

Rerun `inla` on an inla-object (output from `link{inla}`)

Usage

```
inla.rerun(object, plain=FALSE)
```

Arguments

object	An `inla`-object, ie the output from an `inla`-call
plain	Logical. If `FALSE` (default), then make changes in `object` to improve the performance

Value

This function will take the result in `object`, and rerun `inla` again. If `plain` is `FALSE`, start the optimization from the mode in `object` so that we can obtain an improvement the mode for the hyperparameters. Otherwise, start from the same configuration as for `object`. The returned value is an `inla`-object.

See Also

`inla`

Examples

```
r = inla(y ~ 1,  data = data.frame(y=1:10))
r = inla.rerun(r)
```

inla.row.kron *Row-wise Kronecker products*

Description

Takes two Matrices and computes the row-wise Kronecker product. Optionally applies row-wise weights and/or applies an additional 0/1 row-wise Kronecker matrix product, as needed by inla.spde.make.A.

Usage

```
inla.row.kron(M1, M2, repl = NULL, n.repl = NULL, weights = NULL)
```

Arguments

M1	A matrix that can be transformed into a sparse Matrix.
M2	A matrix that can be transformed into a sparse Matrix.
repl	An optional index vector. For each entry, specifies which replicate the row belongs to, in the sense used in inla.spde.make.A.
n.repl	The maximum replicate index, in the sense used in inla.spde.make.A.
weights	Optional scaling weights to be applied row-wise to the resulting matrix.

Value

A sparseMatrix object.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.spde.make.A

inla.sample *Generate samples from an approximated posterior of a fitted model*

Description

This function generate samples from an approximated posterior of a fitted model (an inla-object

Usage

```
inla.posterior.sample(n = 1L, result, intern = FALSE, use.improved.mean = TRUE,
                      add.names = TRUE, seed = 0L)
```

Arguments

n	Number of samples.
result	The inla-object, ie the output from an inla-call. The inla-object must be created with control.compute=list(config=TRUE).
use.improved.mean	Logical. If TRUE then use the marginal mean values when constructing samples. If FALSE then use the mean in the Gaussian approximations.
intern	Logical. If TRUE then produce samples in the internal scale for the hyperparmater, if FALSE then produce samples in the user-scale. (For example log-precision (intern) and precision (user-scale))
add.names	Logical. If TRUE then add name for each elements of each sample. If FALSE, only add name for the first sample. (This save space.)
seed	Control the RNG of inla.qsample, see ?inla.qsample for further information. If seed=0L then GMRFLib will set the seed intelligently/at 'random'. If seed < 0L then the saved state of the RNG will be reused if possible, otherwise, GMRFLib will set the seed intelligently/at 'random'. If seed > 0L then this value is used as the seed for the RNG. If you want reproducible results, you ALSO need to control the seed for the RNG in R by controlling the variable .Random.seed or using the function set.seed, the example for how this can be done.

Details

The hyperparameters are sampled from the configurations used to do the numerical integration, hence if you want a higher resolution, you need to to change the int.stratey variable and friends. The latent field is sampled from the Gaussian approximation conditioned on the hyperparameters, but with a correction for the mean (default).

Value

A list of the samples, where each sample is a list with names hyperpar and latent, and with their marginal densities in logdens$hyperpar and logdens$latent and the joint density is in logdens$joint.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
r = inla(y ~ 1 ,data = data.frame(y=rnorm(1)), control.compute = list(config=TRUE))
samples = inla.posterior.sample(2,r)

## reproducible results:
set.seed(1234)
inla.seed = as.integer(runif(1)*.Machine$integer.max)
x = inla.posterior.sample(100, r, seed = inla.seed)
set.seed(1234)
xx = inla.posterior.sample(100, r, seed = inla.seed)
all.equal(x, xx)
```

inla.sens *Testing code for sensitivity*

Description

TODO: Write a description

Usage

```
inla.sens(inlaObj)
```

Arguments

inlaObj	The result from a run of `inla`.

Value

TODO: This is an EXPERIMENTAL function!

Author(s)

Geir-Arne Fuglstad <geirarne.fuglstad@gmail.com>

See Also

inla

Examples

```
## Case 1: Simple linear regression on simulated data
  # Number of observations
  nObs = 100

  # Measurement noise
  sdNoise = 0.1

  # Coefficients
  mu = 2
  beta = 1

  # Covariate
  x = runif(nObs)

  # Generate data
  y = mu + beta*x + rnorm(nObs)*sdNoise

  # Make some data unobserved
  nUnObs = 20
  y[(nObs-nUnObs+1):nObs] = NA

  # Fit the model
  mod = inla(y ~ x,
           data = list(x = x, y = y))
```

```
  # Calculate sensitivites
  inla.sens(mod)

## Case 2: Time series
  # Length of time-series
  nObs = 100

  # Measurement noise
  sdNoise = 0.1

  # Autoregressive process
  rho = 0.6
  sdProc = 0.1
  arP = matrix(0, nrow = nObs, ncol = 1)
  for(i in 2:nObs)
      arP[i] = rho*arP[i-1] + rnorm(1)*sdProc
  tIdx = 1:nObs

  # Coefficients
  mu = 2

  # Generate data
  y = mu + arP + rnorm(nObs)*sdNoise

  # Make some data unobserved
  nUnObs = 20
  y[(nObs-nUnObs+1):nObs] = NA
  idx = 1:nObs

  # Run INLA
  mod = inla(y ~ f(tIdx, model = "ar1"),
             data = list(y = y, tIdx = tIdx),
             control.inla = list(reordering = "metis"))

  # Calculate sensitivities
  inla.sens(mod)

## Case 3: Epil dataset
  data(Epil)
  my.center = function(x) (x - mean(x))

  # make centered covariates
  Epil$CTrt    = my.center(Epil$Trt)
  Epil$ClBase4 = my.center(log(Epil$Base/4))
  Epil$CV4     = my.center(Epil$V4)
  Epil$ClAge   = my.center(log(Epil$Age))
  Epil$CBT     = my.center(Epil$Trt*Epil$ClBase4)

  # Define the model
  formula = y ~ ClBase4 + CTrt + CBT+ ClAge + CV4 +
            f(Ind, model="iid") + f(rand,model="iid")

  mod = inla(formula,family="poisson", data = Epil)

  # Calculate sensitivities
  inla.sens(mod)
```

```
## Case 4: Spatial data
  # Number of observations
  nObs = 100

  # Measurement noise
  sdNoise = 0.2

  # Spatial process
  sdProc = 1.0
  rho0 = 0.2

  # Coefficients
  beta0 = 1
  beta1 = 2

  # Generate spatial data + measurement noise
  loc = cbind(runif(nObs), runif(nObs))
  dd = as.matrix(dist(loc))
  Sig = sdProc^2*inla.matern.cov(nu = 1, kappa = sqrt(8)/rho0, dd, corr = TRUE)
  L = t(chol(Sig))
  u = L

  # Generate Covariate
  x = runif(nObs)-0.5

  # Combine to observations
  y = beta0 + beta1*x + u

  # Number of unobserved
  nUnObs = 2
  y[1:nUnObs] = NA

  # Mesh
  mesh = inla.mesh.2d(loc, max.edge = 0.05, cutoff = 0.05)

  # Make SPDE object
  spde  = inla.spde2.matern(mesh)
  spde2 = inla.spde2.matern(mesh, constr = TRUE)

  # Make A matrix
  A = inla.spde.make.A(mesh, loc)

  # Stack
  X = cbind(1, x)
  stk = inla.stack(data = list(y = y), A = list(A, 1),
                   effects = list(field = 1:spde$n.spde,
                                  X = X))

  # Run INLA
  mod1 = inla(y ~ -1 + X + f(field, model = spde),
              data = inla.stack.data(stk),
              control.predictor = list(A = inla.stack.A(stk)),
              control.family = list(prior = "pcprec",
                                    param = c(3, 0.05)))
  mod2 = inla(y ~ -1 + X + f(field, model = spde2),
              data = inla.stack.data(stk),
              control.predictor = list(A = inla.stack.A(stk)),
```

```
                  control.family = list(prior = "pcprec",
                                        param = c(3, 0.05)))

# Calculate sensitivities
res1 = inla.sens(mod1)
res2 = inla.sens(mod2)
```

`inla.sens` *Calculate sensitivity measurements*

Description

TODO

Usage

```
inla.sens(inlaRes)
```

Arguments

`inlaRes`	Object returned by `inla` function.

Value

`inla.sens` plots robustness and returns object with different robustnesses

Author(s)

Geir-Arne Fuglstad <geirarne.fuglstad@gmail.com>

Examples

```
TODO
```

`inla.simplify.curve` *Recursive curve simplification.*

Description

Attempts to simplify a polygonal curve by joining nearly colinear segments.

Usage

```
inla.simplify.curve(loc, idx, eps)
```

Arguments

`loc`	Coordinate matrix.
`idx`	Index vector into `loc` specifying a polygonal curve.
`eps`	Straightness tolerance.

Details

Uses a variation of the binary splitting Ramer-Douglas-Peucker algorithm, with a width eps ellipse instead of a rectangle, motivated by prediction ellipse for Brownian bridge.

Value

An index vector into loc specifying the simplified polygonal curve.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

Examples

```
theta = seq(0, 2*pi, length=1000)
loc = cbind(cos(theta), sin(theta))
idx = inla.simplify.curve(loc=loc, idx=1:nrow(loc), eps=0.01)
print(c(nrow(loc), length(idx)))
plot(loc, type="l")
lines(loc[idx,], col="red")
```

---

inla.spde.make.A *Observation/prediction matrices for mesh models.*

Description

Constructs observation/prediction weight matrices for models based on inla.mesh and inla.mesh.1d objects.

Usage

```
inla.spde.make.A(mesh = NULL, loc = NULL, index = NULL,
                 group = NULL, repl = 1L,
                 n.spde = NULL, n.group = NULL, n.repl = NULL,
                 group.mesh = NULL,
                 weights = NULL,
                 A.loc = NULL, A.group = NULL, group.index = NULL,
                 block = NULL, n.block = NULL,
                 block.rescale = c("none", "count", "weights", "sum"),
                 ...)
```

Arguments

mesh	An inla.mesh or inla.mesh.1d object specifying a function basis on a mesh domain. Alternatively, an inla.spde object that includes a mesh (e.g. from inla.spde2.matern).
loc	Observation/prediction coordinates. mesh and loc defines a matrix A.loc of mapping weights between basis function weights and field values. If loc is NULL, A.loc is defined as Diagonal(n.spde, 1).
index	For each observation/prediction value, an index into loc. Default is seq_len(nrow(A.loc)).
group	For each observation/prediction value, an index into the group model.

repl	For each observation/prediction value, the replicate index.
n.spde	The number of basis functions in the mesh model. (Note: may be different than the number of mesh vertices/nodes/knots.)
n.group	The size of the group model.
n.repl	The total number of replicates.
group.mesh	An optional inla.mesh.1d object for the group model.
weights	Optional scaling weights to be applied row-wise to the resulting matrix.
A.loc	Optional precomputed observation/prediction matrix. A.loc can be specified instead of mesh+loc, optionally with index supplied.
A.group	Optional precomputed observation/prediction matrix for the group model. A.group can be specified instead of group and/or group.mesh, optionally with group.index supplied.
group.index	For each observation/prediction value, an index into the rows of A.group.
block	Optional indices specifying block groupings: Entries with the same block value are joined into a single row in the resulting matrix, and the block values are the row indices. This is intended for construction of approximate integration schemes for regional data problems. See inla.spde.make.block.A for details.
n.block	The number of blocks.
block.rescale	Specifies what scaling method should be used when joining entries as grouped by a block specification. See inla.spde.make.block.A for details.
...	Additional parameters. Currently unused.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.spde.make.index

Examples

```
loc = matrix(runif(10000*2)*1000,10000,2)
mesh = inla.mesh.2d(loc=loc,
                    cutoff=50,
                    max.edge=c(50,500))
A = inla.spde.make.A(mesh, loc=loc)
```

---

inla.spde.make.block.A *Observation matrices for mesh models.*

Description

Constructs observation/prediction weight matrices for numerical integration schemes for regional data problems. Primarily intended for internal use by inla.spde.make.A.

Usage

```
inla.spde.make.block.A(A, block, n.block = max(block),
                       weights = NULL,
                       rescale = c("none", "count", "weights", "sum"))
```

Arguments

A	A precomputed observation/prediction matrix for locations that are to be joined.
block	Indices specifying block groupings: Entries with the same block value are joined into a single row in the resulting matrix, and the block values are the row indices.
n.block	The number of blocks.
weights	Optional scaling weights to be applied row-wise to the input A matrix.
rescale	Specifies what scaling method should be used when joining the rows of the A matrix as grouped by the block specification. • 'none': Straight sum, no rescaling. • 'count': Divide by the number of entries in the block. • 'weights': Divide by the sum of the weight values within each block. • 'sum': Divide by the resulting row sums.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.spde.make.A

---

inla.spde.make.index *SPDE model index vector generation*

Description

Generates a list of named index vectors for an SPDE model.

Usage

```
inla.spde.make.index(name,
                     n.spde,
                     n.group = 1,
                     n.repl = 1,
                     ...)
```

Arguments

name	A character string with the base name of the effect.
n.spde	The size of the model, typically from spde$n.spde.
n.group	The size of the group model.
n.repl	The number of model replicates.
...	Additional parameters. Currently unused.

Value

A list of named index vectors.

name	Indices into the vector of latent variables
name.group	'group' indices
name.repl	Indices for replicates

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.spde.make.A, inla.spde2.result

Examples

```
loc = matrix(runif(100*2),100,2)
mesh = inla.mesh.create.helper(points.domain=loc, max.edge=c(0.1,0.5))
spde = inla.spde2.matern(mesh)
index = inla.spde.make.index("spatial", spde$n.spde, n.repl=2)
spatial.A = inla.spde.make.A(mesh, loc,
                             index=rep(1:nrow(loc), 2),
                             repl=rep(1:2, each=nrow(loc)))
y = 10+rnorm(100*2)
stack = inla.stack(data=list(y=y),
                   A=list(spatial.A),
                   effects=list(c(index, list(intercept=1))),
                   tag="tag")
data = inla.stack.data(stack, spde=spde)
formula = y ~ -1 + intercept + f(spatial, model=spde,
                                 replicate=spatial.repl)
result = inla(formula, family="gaussian", data=data,
              control.predictor=list(A=inla.stack.A(stack)))
spde.result = inla.spde2.result(result, "spatial", spde)
```

---

inla.spde.models *List SPDE models supported by inla.spde objects*

Description

List SPDE models supported by inla.spde objects

Usage

```
inla.spde.models(function.names=FALSE)
inla.spde1.models()
inla.spde2.models()
```

Arguments

function.names	If FALSE, return list model name lists. If TRUE, return list of model object constructor function names.

Details

Returns a list of available SPDE model type name lists, one for each inla.spde model class (currently inla.spde1 and inla.spde2).

Value

List of available SPDE model type name lists.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

Examples

```
## Display help for each supported inla.spde2 model:
for (model in inla.spde2.models())
  print(help(paste("inla.spde2.", model, sep="")))

## Display help for each supported inla.spde* model:
models = inla.spde.models()
for (type in names(models))
  for (model in models[[type]])
    print(help(paste("inla.", type, ".", model, sep="")))

## Display help for each supported inla.spde* model (equivalent to above):
for (model in inla.spde.models(function.names=TRUE))
  print(help(model))
```

---

inla.spde.precision *Precision matrices for SPDE models*

Description

Calculates the precision matrix for given parameter values based on an inla.spde model object.

Usage

```
inla.spde.precision(...)

## S3 method for class 'inla.spde2'
inla.spde.precision(spde,
                    theta = NULL,
                    phi0 = inla.spde2.theta2phi0(spde, theta),
                    phi1 = inla.spde2.theta2phi1(spde, theta),
                    phi2 = inla.spde2.theta2phi2(spde, theta), ...)
inla.spde2.precision(spde,
                     theta = NULL,
                     phi0 = inla.spde2.theta2phi0(spde, theta),
                     phi1 = inla.spde2.theta2phi1(spde, theta),
```

```
                     phi2 = inla.spde2.theta2phi2(spde, theta), ...)

## For deprecated inla.spde1 models:
## S3 method for class 'inla.spde1'
inla.spde.precision(spde, ...)
inla.spde1.precision(spde, ...)
```

Arguments

spde	An inla.spde object.
theta	The parameter vector.
phi0	Internal parameter for a generic model. Expert option only.
phi1	Internal parameter for a generic model. Expert option only.
phi2	Internal parameter for a generic model. Expert option only.
...	Additional parameters passed on to other methods.

Value

A sparse precision matrix.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.spde.models, inla.spde2.generic, inla.spde2.theta2phi0, inla.spde2.theta2phi1, inla.spde2.theta2phi2

---

inla.spde.result *SPDE result extraction from INLA estimation results*

Description

Exctract field and parameter values and distributions for an inla.spde SPDE effect from an inla result object.

Usage

```
inla.spde.result(...)

## S3 method for class 'inla.spde1'
inla.spde.result(inla, name, spde, do.transform = TRUE, ...)
## S3 method for class 'inla.spde2'
inla.spde.result(inla, name, spde, do.transform = TRUE, ...)

## Direct function call for class 'inla.spde1':
inla.spde1.result(inla, name, spde, do.transform = TRUE, ...)
## Direct function call for class 'inla.spde2':
inla.spde2.result(inla, name, spde, do.transform = TRUE, ...)
```

Arguments

- `inla` — An `inla` object obtained from a call to `inla`
- `name` — A character string with the name of the SPDE effect in the inla formula.
- `spde` — The `inla.spde` object used for the effect in the inla formula. (Note: this could have been stored in the inla output, but isn't.) Usually the result of a call to `inla.spde2.matern`.
- `do.transform` — If `TRUE`, also calculate marginals transformed to user-scale. Setting to `FALSE` is useful for large non-stationary models, as transforming many marginal densities is time-consuming.
- `...` — Further arguments passed to and from other methods.

Value

For `inla.spde2` models, a list, where the nominal range and variance are defined as the values that would have been obtained with a stationary model and no boundary effects:

- `marginals.kappa` — Marginal densities for kappa
- `marginals.log.kappa` — Marginal densities for log(kappa)
- `marginals.log.range.nominal` — Marginal densities for log(range)
- `marginals.log.tau` — Marginal densities for log(tau)
- `marginals.log.variance.nominal` — Marginal densities for log(variance)
- `marginals.range.nominal` — Marginal densities for range
- `marginals.tau` — Marginal densities for tau
- `marginals.theta` — Marginal densities for the theta parameters
- `marginals.values` — Marginal densities for the field values
- `marginals.variance.nominal` — Marginal densities for variance
- `summary.hyperpar` — The SPDE related part of the inla hyperpar output summary
- `summary.log.kappa` — Summary statistics for log(kappa)
- `summary.log.range.nominal` — Summary statistics for log(range)
- `summary.log.tau` — Summary statistics for log(tau)
- `summary.log.variance.nominal` — Summary statistics for log(kappa)
- `summary.theta` — Summary statistics for the theta parameters
- `summary.values` — Summary statistics for the field values

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.spde.models, inla.spde2.matern

Examples

```
loc = matrix(runif(100*2),100,2)
mesh = inla.mesh.create.helper(points.domain=loc, max.edge=c(0.1,0.5))
spde = inla.spde2.matern(mesh)
index = inla.spde.make.index("spatial", mesh$n, n.repl=2)
spatial.A = inla.spde.make.A(mesh, loc,
                             index=rep(1:nrow(loc), 2),
                             repl=rep(1:2, each=nrow(loc)))
## Toy example with no spatial correlation (range=zero)
y = 10+rnorm(100*2)
stack = inla.stack(data=list(y=y),
                   A=list(spatial.A),
                   effects=list(c(index, list(intercept=1))),
                   tag="tag")
data = inla.stack.data(stack, spde=spde)
formula = y ~ -1 + intercept + f(spatial, model=spde,
                                 replicate=spatial.repl)
result = inla(formula, family="gaussian", data=data,
              control.predictor=list(A=inla.stack.A(stack)))
spde.result = inla.spde.result(result, "spatial", spde)
plot(spde.result$marginals.range.nominal[[1]], type="l")
```

---

inla.spde.sample — *Sample from SPDE models*

Description

Old methods fo sampling from a SPDE model. For new code, use inla.spde.precision and inla.qsample instead.

Usage

```
inla.spde.sample(...)

## Default S3 method:
inla.spde.sample(precision, seed=NULL, ...)
## S3 method for class 'inla.spde'
inla.spde.sample(spde, seed=NULL, ...)
```

Arguments

precision	A precision matrix.
seed	The seed for the pseudo-random generator.
spde	An inla.spde object.
...	Parameters passed on to other methods.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.spde.precision, inla.qsample

---

inla.spde1.create *Old SPDE model objects for INLA*

Description

Create an inla.spde1 model object.

Usage

```
## Create an SPDE model object:
inla.spde1.create(mesh,
                  model = c("matern", "imatern", "matern.osc"),
                  param = NULL,
                  ...)

## Shortcuts to the matern, imatern and matern.osc models:
inla.spde1.matern(mesh, ...)
inla.spde1.imatern(mesh, ...)
inla.spde1.matern.osc(mesh, ...)
```

Arguments

mesh	The mesh to build the model on, as an inla.mesh object.
model	The name of the model.
param	Model specific parameters.
...	Additional parameters passed on to other methods.

Details

Note: This is an old spde object format retained for backwards compatibility. Please use inla.spde2 models for new code.

This method constructs an object for SPDE models. Currently implemented:

model="matern"

$$(\kappa^2(u) - \Delta)^{\alpha/2}(\tau(u)x(u)) = W(u)$$

param:

- alpha = 1 or 2
- basis.T = Matrix of basis functions for $\log\tau(u)$
- basis.K = Matrix of basis functions for $\log\kappa^2(u)$

model="imatern"

$$(-\Delta)^{\alpha/2}(\tau(u)x(u)) = W(u)$$

param:

- alpha = 1 or 2
- basis.T = Matrix of basis functions for $\log \tau(u)$

Value

An inla.spde1 object.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.spde2.matern, inla.mesh.2d, inla.mesh.basis

Examples

```
n = 100
field.fcn = function(loc) (10*cos(2*pi*2*(loc[,1]+loc[,2])))
loc = matrix(runif(n*2),n,2)
## One field, 2 observations per location
idx.y = rep(1:n,2)
y = field.fcn(loc[idx.y,]) + rnorm(length(idx.y))

mesh = inla.mesh.create(loc, refine=list(max.edge=0.05))
spde = inla.spde.create(mesh, model="matern")
data = list(y=y, field=mesh$idx$loc[idx.y])
formula = y ~ -1 + f(field, model=spde)
result = inla(formula, data=data, family="normal")

## Plot the mesh structure:
plot(mesh)

if (require(rgl)) {
  ## Plot the posterior mean:
  plot(mesh, rgl=TRUE,
       result$summary.random$field[,"mean"],
       color.palette = colorRampPalette(c("blue","green","red")))
  ## Plot residual field:
  plot(mesh, rgl=TRUE,
       result$summary.random$field[,"mean"]-field.fcn(mesh$loc),
       color.palette = colorRampPalette(c("blue","green","red")))
}
```

`inla.spde2.generic` *Generic spde2 model creation.*

Description

Creates and inla.spde2 object describing the internal structure of an `'spde2'` model.

Usage

```
inla.spde2.generic(M0, M1, M2, B0, B1, B2, theta.mu, theta.Q,
                   transform = c("logit", "log", "identity"),
                   theta.initial = theta.mu,
                   fixed = rep(FALSE, length(theta.mu)),
                   theta.fixed = theta.initial[fixed],
                   BLC = cbind(0, diag(nrow = length(theta.mu))),
                   ...)

## Map theta values to internal phi values
inla.spde2.theta2phi0(spde, theta)
inla.spde2.theta2phi1(spde, theta)
inla.spde2.theta2phi2(spde, theta)
```

Arguments

`M0`	The symmetric `M0` matrix.
`M1`	The square `M1` matrix.
`M2`	The symmetric `M2` matrix.
`B0`	Basis definition matrix for ϕ_0.
`B1`	Basis definition matrix for ϕ_2.
`B2`	Basis definition matrix for ϕ_2.
`theta.mu`	Prior expectation for the θ vector
`theta.Q`	Prior precision for the θ vector
`transform`	Transformation link for ϕ_2. Valid settings are `"logit"`, `"log"`, and `"identity"`
`theta.initial`	Initial value for the θ vector. Default `theta.mu`
`fixed`	Logical vector. For every `TRUE` value, treat the corresponding `theta` value as known.
`theta.fixed`	Vector holding the values of fixed `theta` values. Default =`theta.initial[fixed]`
`BLC`	Basis definition matrix for linear combinations of `theta`.
`...`	Additional parameters, currently unused.
`spde`	An `inla.sdpe2` object.
`theta`	parameter values to be mapped.

Value

For `inla.spde2.generic`, an `inla.spde2` object.

For `inla.spde2.theta2phi0/1/2`, a vector of ϕ values.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.spde2.models, inla.spde2.matern

---

inla.spde2.matern *Matern SPDE model object for INLA*

Description

Create an inla.spde2 model object for a Matern model. Use inla.spde2.pcmatern instead for a PC prior for the parameters.

Usage

```
inla.spde2.matern(mesh,
                  alpha = 2,
                  param = NULL,
                  constr = FALSE,
                  extraconstr.int = NULL,
                  extraconstr = NULL,
                  fractional.method = c("parsimonious", "null"),
                  B.tau = matrix(c(0,1,0),1,3),
                  B.kappa = matrix(c(0,0,1),1,3),
                  prior.variance.nominal = 1,
                  prior.range.nominal = NULL,
                  prior.tau = NULL,
                  prior.kappa = NULL,
                  theta.prior.mean = NULL,
                  theta.prior.prec = 0.1,
                  n.iid.group = 1,
                  ...)
```

Arguments

mesh	The mesh to build the model on, as an inla.mesh or inla.mesh.1d object.
alpha	Fractional operator order, $0 < \alpha \leq 2$ supported. ($\nu = \alpha - d/2$)
param	Parameter, e.g. generated by param2.matern.orig
constr	If TRUE, apply an integrate-to-zero constraint. Default FALSE.
extraconstr.int	Field integral constraints.
extraconstr	Direct linear combination constraints on the basis weights.
fractional.method	Specifies the approximation method to use for fractional (non-integer) alpha values. 'parsimonious' gives an overall approximate minimal covariance error, 'null' uses approximates low-order properties.
B.tau	Matrix with specification of log-linear model for τ.

`B.kappa` Matrix with specification of log-linear model for κ.

`prior.variance.nominal`
Nominal prior mean for the field variance

`prior.range.nominal`
Nominal prior mean for the spatial range

`prior.tau` Prior mean for tau (overrides `prior.variance.nominal`)

`prior.kappa` Prior mean for kappa (overrides `prior.range.nominal`)

`theta.prior.mean`
(overrides `prior.*`)

`theta.prior.prec`
Scalar, vector or matrix, specifying the joint prior precision for $theta$.

`n.iid.group` If greater than 1, build an explicitly iid replicated model, to support constraints applied to the combined replicates, for example in a time-replicated spatial model. Constraints can either be specified for a single mesh, in which case it's applied to the average of the replicates (`ncol(A)` should be `mesh$n` for 2D meshes, `mesh$m` for 1D), or as general constraints on the collection of replicates (`ncol(A)` should be `mesh$n * n.iid.group` for 2D meshes, `mesh$m * n.iid.group` for 1D).

`...` Additional parameters for special uses.

Details

This method constructs a Matern SPDE model, with spatial scale parameter $\kappa(u)$ and variance rescaling parameter $\tau(u)$.

$$(\kappa^2(u) - \Delta)^{\alpha/2}(\tau(u)x(u)) = W(u)$$

Stationary models are supported for $0 < \alpha \leq 2$, with spectral approximation methods used for non-integer α, with approximation method determined by `fractional.method`.

Non-stationary models are supported for $\alpha = 2$ only, with

- $\log\tau(u) = B_0^\tau(u) + \sum_{k=1}^p B_k^\tau(u)\theta_k$
- $\log\kappa(u) = B_0^\kappa(u) + \sum_{k=1}^p B_k^\kappa(u)\theta_k$

The same parameterisation is used in the stationary cases, but with B_0^τ, B_k^τ, B_0^κ, and B_k^τ constant across u.

Integration and other general linear constraints are supported via the `constr`, `extraconstr.int`, and `extraconstr` parameters, which also interact with `n.iid.group`.

Value

An `inla.spde2` object.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

`inla.mesh.2d`, `inla.mesh.create`, `inla.mesh.1d`, `inla.mesh.basis`, `inla.spde2.pcmatern`, `inla.spde2.generic`

Examples

```
n = 100
field.fcn = function(loc) (10*cos(2*pi*2*(loc[,1]+loc[,2])))
loc = matrix(runif(n*2),n,2)
## One field, 2 observations per location
idx.y = rep(1:n,2)
y = field.fcn(loc[idx.y,]) + rnorm(length(idx.y))

mesh = inla.mesh.create(loc, refine=list(max.edge=0.05))
spde = inla.spde2.matern(mesh)
data = list(y=y, field=mesh$idx$loc[idx.y])
formula = y ~ -1 + f(field, model=spde)
result = inla(formula, data=data, family="normal")

## Plot the mesh structure:
plot(mesh)

if (require(rgl)) {
  col.pal = colorRampPalette(c("blue","cyan","green","yellow","red"))
  ## Plot the posterior mean:
  plot(mesh, rgl=TRUE,
       result$summary.random$field[,"mean"],
       color.palette = col.pal)
  ## Plot residual field:
  plot(mesh, rgl=TRUE,
       result$summary.random$field[,"mean"]-field.fcn(mesh$loc),
       color.palette = col.pal)
}

result.field = inla.spde.result(result, "field", spde)
plot(result.field$marginals.range.nominal[[1]])
```

---

inla.spde2.matern.sd.basis

Approximate variance-compensating basis functions

Description

Calculates an approximate basis for `tau` and `kappa` for an `inla.spde2.matern` model where `tau` is a rescaling parameter.

Usage

```
inla.spde2.matern.sd.basis(mesh, B.sd, B.range, method = 1,
                           local.offset.compensation = FALSE,
                           alpha = 2, ...)
```

Arguments

`mesh`	An `inla.mesh` object.
`B.sd`	Desired basis for log-standard deviations.
`B.range`	Desired basis for spatial range.

method	Construction method selector. Expert option only.
local.offset.compensation	If FALSE, only compensate in the average for the tau offset.
alpha	The model alpha parameter.
...	Additional parameters passed on to internal inla.spde2.matern calls.

Value

List of basis specifications

B.tau	Basis for log(tau)
B.kappa	Basis for log(kappa)

Intended for passing on to inla.spde2.matern.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.spde2.matern

---

inla.spde2.pcmatern *Matern SPDE model object with PC prior for INLA*

Description

Create an inla.spde2 model object for a Matern model, using a PC prior for the parameters.

Usage

```
inla.spde2.pcmatern(mesh,
                    alpha = 2,
                    param = NULL,
                    constr = FALSE,
                    extraconstr.int = NULL,
                    extraconstr = NULL,
                    fractional.method = c("parsimonious", "null"),
                    n.iid.group = 1,
                    prior.range = NULL,
                    prior.sigma = NULL)
```

Arguments

mesh	The mesh to build the model on, as an inla.mesh or inla.mesh.1d object.
alpha	Fractional operator order, $0 < \alpha \leq 2$ supported, for $\nu = \alpha - d/2 > 0$.
param	Further model parameters. Not currently used.
constr	If TRUE, apply an integrate-to-zero constraint. Default FALSE.
extraconstr.int	Field integral constraints.

extraconstr — Direct linear combination constraints on the basis weights.

fractional.method — Specifies the approximation method to use for fractional (non-integer) alpha values. 'parsimonious' gives an overall approximate minimal covariance error, 'null' uses approximates low-order properties.

n.iid.group — If greater than 1, build an explicitly iid replicated model, to support constraints applied to the combined replicates, for example in a time-replicated spatial model. Constraints can either be specified for a single mesh, in which case it's applied to the average of the replicates (ncol(A) should be mesh$n for 2D meshes, mesh$m for 1D), or as general constraints on the collection of replicates (ncol(A) should be mesh$n * n.iid.group for 2D meshes, mesh$m * n.iid.group for 1D).

prior.range — A length 2 vector, with (range0,Prange) specifying that $P(\rho < \rho_0) = p_\rho$, where ρ is the spatial range of the random field. If Prange is NA, then range0 is used as a fixed range value.

prior.sigma — A length 2 vector, with (sigma0,Psigma) specifying that $P(\sigma > \sigma_0) = p_\sigma$, where σ is the marginal standard deviation of the field. If Psigma is NA, then sigma0 is used as a fixed range value.

Details

This method constructs a Matern SPDE model, with spatial range ρ and standard deviation parameter σ. In the parameterisation

$$(\kappa^2 - \Delta)^{\alpha/2}(\tau x(u)) = W(u)$$

the spatial scale parameter $\kappa = \sqrt{8\nu}/\rho$, where $\nu = \alpha - d/2$, and τ is proportional to $1/\sigma$.

Stationary models are supported for $0 < \alpha \leq 2$, with spectral approximation methods used for non-integer α, with approximation method determined by fractional.method.

Integration and other general linear constraints are supported via the constr, extraconstr.int, and extraconstr parameters, which also interact with n.iid.group.

The joint PC prior density for the spatial range, ρ, and the marginal standard deviation, σ, and is

$$\pi(\rho, \sigma) = \frac{d\lambda_\rho}{2}\rho^{-1-d/2}\exp(-\lambda_\rho \rho^{-d/2})\lambda_\sigma \exp(-\lambda_\sigma \sigma)$$

where λ_ρ and λ_σ are hyperparameters that must be determined by the analyst. The practical approach for this in INLA is to require the user to indirectly specify these hyperparameters through

$$P(\rho < \rho_0) = p_\rho$$

and

$$P(\sigma > \sigma_0) = p_\sigma$$

where the user specifies the lower tail quantile and probability for the range (ρ_0 and p_ρ) and the upper tail quantile and probability for the standard deviation (σ_0 and α_σ).

This allows the user to control the priors of the parameters by supplying knowledge of the scale of the problem. What is a reasonable upper magnitude for the spatial effect and what is a reasonable lower scale at which the spatial effect can operate? The shape of the prior was derived through a construction that shrinks the spatial effect towards a base model of no spatial effect in the sense of distance measured by Kullback-Leibler divergence.

The prior is constructed in two steps, under the idea that having a spatial field is an extension of not having a spatial field. First, a spatially constant random effect ($\rho = \infty$) with finite variance is more complex than not having a random effect ($\sigma = 0$). Second, a spatial field with spatial variation ($\rho < \infty$) is more complex than the random effect with no spatial variation. Each of these extensions are shrunk towards the simpler model and, as a result, we shrink the spatial field towards the base model of no spatial variation and zero variance ($\rho = \infty$ and $\sigma = 0$).

The details behind the construction of the prior is presented in Fuglstad, et al. (2016) and is based on the PC prior framework (Simpson, et al., 2015).

Value

An `inla.spde2` object.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

References

Fuglstad, G.-A., Simpson, D., Lindgren, F., and Rue, H. (2016) Constructing Priors that Penalize the Complexity of Gaussian Random Fields. arXiv:1503.00256

Simpson, D., Rue, H., Martins, T., Riebler, A., and Sørbye, S. (2015) Penalising model component complexity: A principled, practical approach to constructing priors. arXiv:1403.4630

See Also

inla.mesh.2d, inla.mesh.create, inla.mesh.1d, inla.mesh.basis, inla.spde2.matern, inla.spde2.generic

Examples

```
## Spatial interpolation
  n = 100
  field.fcn = function(loc) (10*cos(2*pi*2*(loc[,1]+loc[,2])))
  loc = matrix(runif(n*2),n,2)
  ## One field, 2 observations per location
  idx.y = rep(1:n,2)
  y = field.fcn(loc[idx.y,]) + rnorm(length(idx.y))

  mesh = inla.mesh.2d(loc, max.edge=0.05, cutoff=0.01)
  spde = inla.spde2.pcmatern(mesh,
           prior.range=c(0.01,0.1), prior.sigma=c(100,0.1))
  data = list(y=y, field=mesh$idx$loc[idx.y])
  formula = y ~ -1 + f(field, model=spde)
  result = inla(formula, data=data, family="normal")

  ## Plot the mesh structure:
  plot(mesh)

  if (require(rgl)) {
    col.pal = colorRampPalette(c("blue","cyan","green","yellow","red"))
    ## Plot the posterior mean:
    plot(mesh, rgl=TRUE,
         result$summary.random$field[,"mean"],
         color.palette = col.pal)
    ## Plot residual field:
```

```
    plot(mesh, rgl=TRUE,
         result$summary.random$field[,"mean"]-field.fcn(mesh$loc),
         color.palette = col.pal)
  }


  result.field = inla.spde.result(result, "field", spde)
  par(mfrow=c(2,1))
  plot(result.field$marginals.range.nominal[[1]],
       type="l", main="Posterior density for range")
  plot(inla.tmarginal(sqrt, result.field$marginals.variance.nominal[[1]]),
       type="l", main="Posterior density for std.dev.")
  par(mfrow=c(1,1))

## Spatial model
  set.seed(1234234)

  ## Generate spatial locations
  nObs = 200
  loc = matrix(runif(nObs*2), nrow = nObs, ncol = 2)

  ## Generate observation of spatial field
  nu = 1.0
  rhoT = 0.2
  kappaT = sqrt(8*nu)/rhoT
  sigT = 1.0
  Sig = sigT^2*inla.matern.cov(nu = nu,
                               kappa = kappaT,
                               x = as.matrix(dist(loc)),
                               d = 2,
                               corr = TRUE)
  L = t(chol(Sig))
  u = L %*% rnorm(nObs)

  ## Construct observation with nugget
  sigN = 0.1
  y = u + sigN*rnorm(nObs)

  ## Create the mesh and spde object
  mesh = inla.mesh.2d(loc,
                      max.edge = 0.05,
                      cutoff = 0.01)
  spde = inla.spde2.pcmatern(mesh,
                             prior.range = c(0.01, 0.05),
                             prior.sigma = c(10, 0.05))

  ## Create projection matrix for observations
  A = inla.spde.make.A(mesh = mesh,
                       loc = loc)

  ## Run model without any covariates
  idx = 1:spde$n.spde
  res = inla(y ~ f(idx, model = spde) - 1,
             data = list(y = y, idx = idx, spde = spde),
             control.predictor = list(A = A))

## Re-run model with fixed range
```

```
spde.fixed = inla.spde2.pcmatern(mesh,
                                 prior.range = c(0.2, NA),
                                 prior.sigma = c(10, 0.05))

res.fixed = inla(y ~ f(idx, model = spde) - 1,
                 data = list(y = y, idx = idx, spde = spde.fixed),
                 control.predictor = list(A = A))
```

---

inla.spTransform *Wrapper method for* sp::spTransform

Description

Handles transformation of various inla objects accorting to coordinate reference systems of sp::CRS or inla.CRS class.

Usage

```
inla.spTransform(x, ...)
## Default S3 method:
inla.spTransform(x, crs0, crs1,
                 passthrough=FALSE, ...)
## S3 method for class 'SpatialPoints'
inla.spTransform(x, CRSobj,
                 passthrough=FALSE, ...)
## S3 method for class 'inla.mesh.lattice'
inla.spTransform(x, CRSobj,
                 passthrough=FALSE, ...)
## S3 method for class 'inla.mesh.segment'
inla.spTransform(x, CRSobj,
                 passthrough=FALSE, ...)
## S3 method for class 'inla.mesh'
inla.spTransform(x, CRSobj,
                 passthrough=FALSE, ...)
```

Arguments

x	The object that should be transformed from it's current CRS to a new CRS
crs0	The source sp::CRS or inla.CRS object
crs1	The target sp::CRS or inla.CRS object
CRSobj	The target sp::CRS or inla.CRS object
passthrough	default FALSE. Setting to TRUE allows objects with no CRS information to be passed through without transformation.
...	Potential additional arguments

Value

The object is returned with its coordinates transformed

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.CRS

Examples

```
if (require(rgdal)) {
  latt <- inla.mesh.lattice(-10:10, 40:60)
  mesh1 <- inla.mesh.create(lattice=latt, extend=FALSE, refine=FALSE,
                            crs=inla.CRS("longlat"))
  mesh2 <- inla.spTransform(mesh1, inla.CRS("lambert"))
  summary(mesh1)
  summary(mesh2)
}
```

---

inla.ssh.copy.id *Setup remote computing*

Description

Initialize the definition file and print the path to the internal script to transfer ssh-keys

Usage

```
inla.remote()
inla.ssh.copy.id()
```

Arguments

None

Value

`inla.remote` is used once to setup the remote host information file (definition file) in the users home directory; see the FAQ entry on this issue for more information. `inla.ssh.copy.id` will return the path to the internal script to transfer ssh-keys.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
##See the FAQ entry on this issue on r-inla.org.
```

inla.stack *Data stacking for advanced INLA models*

Description

Functions for combining data, effects and observation matrices into inla.stack object, and extracting information from such objects.

Usage

```
## Create data stack as a sum of predictors:
inla.stack.sum(data, A, effects, tag="", compress=TRUE, remove.unused=TRUE)

## Join two or more data stacks:
inla.stack.join(..., compress=TRUE, remove.unused=TRUE)

## Shorthand for inla.stack.join and inla.stack.sum:
inla.stack(..., compress=TRUE, remove.unused=TRUE)

## Compress an existing stack:
inla.stack.compress(stack, remove.unused=TRUE)

## Remove unused entries from an existing stack:
inla.stack.remove.unused(stack)

## Extract tagged indices:
inla.stack.index(stack, tag)

## Extract data for inla call, and optionally join with other variables:
inla.stack.data(stack, ...)

## Extract "A matrix" for control.predictor:
inla.stack.A(stack)

## Extract data associated with the "left hand side" of the model:
## (e.g. the data itself, Ntrials, link, E, ...)
inla.stack.LHS(stack)

## Extract data associated with the "right hand side" of the model
## (all the covariates/predictors)
inla.stack.RHS(stack)
```

Arguments

data	A list of data vectors.
A	A list of observation matrices.
effects	A collection of effects/predictors. Each list element corresponds to an observation matrix, and must either be a single vector or a list of vectors.
tag	A string specifying a tag for later identification.

`compress`	If `TRUE`, compress the model by removing duplicated rows of effects, replacing the corresponding A-matrix columns with a single column containing the sum.
`remove.unused`	If `TRUE`, compress the model by removing rows of effects corresponding to all-zero columns in the A matrix (and removing those columns).
`stack`	An `inla.data.stack` object, created by a call to `inla.stack`, `inla.stack.sum`, or `inla.stack.join`.
`...`	For `inla.stack.join`, two or more data stacks of class `inla.data.stack`, created by a call to `inla.stack`, `inla.stack.sum`, or `inla.stack.join`. For `inla.stack.data`, a list of variables to be joined with the data list.

Details

n_l effect blocks

n_k effects

n_i data values

$n_{j,l}$ effect size for block l

n_j $= \sum_{l=1}^{n_l} n_{j,l}$ total effect size

Input:

`data` $(y^1, \ldots, y^p)$ p vectors, each of length n_i

`A` $(A^1, \ldots, A^{n_l})$ matrices of size $n_i \times n_{j,l}$

`effects` $\left((x^{1,1}, \ldots, x^{n_k,1}), \ldots, (x^{1,n_l}, \ldots, x^{n_k,n_l})\right)$ collections of effect vectors of length $n_{j,l}$

$$\text{predictor}(y^1, \ldots, y^p) \sim \sum_{l=1}^{n_l} A^l \sum_{k=1}^{n_k} g(k, x^{k,l}) = \tilde{A} \sum_{k=1}^{n_k} g(k, \tilde{x}^k)$$

where

$$\tilde{A} = \text{cbind}\left(A^1, \ldots, A^{n_l}\right)$$

$$\tilde{x}^k = \text{rbind}\left(x^{k,1}, \ldots, x^{k,n_l}\right)$$

and for each block l, any missing $x^{k,l}$ is replaced by an `NA` vector.

Value

A data stack of class `inla.data.stack`. Elements:

- `data` $= (y^1, \ldots, y^p, \tilde{x}^1, \ldots, \tilde{x}^{n_k})$
- `A` $= \tilde{A}$
- `data.names` List of data names, length p
- `effect.names` List of effect names, length n_k
- `n.data` Data length, n_i
- `index` List indexed by `tag`s, each element indexing into $i = 1, \ldots, n_i$

See Also

`inla.spde.make.A`, `inla.spde.make.index`

Examples

```
n = 200
loc = matrix(runif(n*2),n,2)
mesh = inla.mesh.create.helper(points.domain=loc,
                               max.edge=c(0.05, 0.2))
proj.obs = inla.mesh.projector(mesh, loc=loc)
proj.pred = inla.mesh.projector(mesh, loc=mesh$loc)
spde = inla.spde2.matern(mesh,
                         B.tau=cbind(log(1), 1, 0),
                         B.kappa=matrix(c(log(sqrt(8)/0.2), 0, 1), 1, 3))

covar = rnorm(n)
field = inla.qsample(n=1, Q=inla.spde2.precision(spde, theta=c(0,0)))[,1]
y = 2*covar + inla.mesh.project(proj.obs, field)

A.obs = inla.spde.make.A(mesh, loc=loc)
A.pred = inla.spde.make.A(mesh, loc=proj.pred$loc)
stack.obs =
    inla.stack(data=list(y=y),
               A=list(A.obs, 1),
               effects=list(c(list(intercept=rep(1, mesh$n)),
                              inla.spde.make.index("spatial", spde$n.spde)),
                            covar=covar),
               tag="obs")
stack.pred =
    inla.stack(data=list(y=NA),
               A=list(A.pred),
               effects=list(c(list(intercept=rep(1, mesh$n)),
                              inla.spde.make.index("spatial", mesh$n))),
               tag="pred")
stack = inla.stack(stack.obs, stack.pred)

formula = y ~ -1 + intercept + covar + f(spatial, model=spde)
result1 = inla(formula,
               data=inla.stack.data(stack.obs, spde=spde),
               family="gaussian",
               control.predictor=list(A=inla.stack.A(stack.obs), compute=TRUE),
               verbose=TRUE)

plot(y,result1$summary.fitted.values[inla.stack.index(stack.obs,"obs")$data,"mean"])

result2 = inla(formula,
               data=inla.stack.data(stack, spde=spde),
               family="gaussian",
               control.predictor=list(A=inla.stack.A(stack), compute=TRUE),
               verbose=TRUE)

field.pred = inla.mesh.project(proj.pred,
    result2$summary.fitted.values[inla.stack.index(stack,"pred")$data, "mean"])

dev.new()
plot(field, field.pred)
dev.new()
image(inla.mesh.project(mesh,
    field=field,
    dims=c(200,200)))
```

```
dev.new()
image(inla.mesh.project(mesh,
    field=field.pred,
    dims=c(200,200)))
```

---

inla.surv *Create a Survival Object for INLA*

Description

Create a survival object, to be used as a response variable in a model formula for the inla function for survival models.

Usage

```
inla.surv(time, event, time2, truncation,subject)
## S3 method for class 'inla.surv'
plot(x, y, ...)
## S3 method for class 'inla.surv'
print(x, ...)
is.inla.surv(object)
as.inla.surv(object, ...)
```

Arguments

time	For right censored data, this is the follow up time. For interval data, this is the starting time for the interval.
event	The status indicator, 1=observed event, 0=right censored event, 2=left censored event, 3=interval censored event. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.
time2	Ending time for the interval censured data.
truncation	Left truncation. If missing it is assumed to be 0.
subject	Patient number in multiple event data, not needed otherwise.
object	Any R-object
x	Object to plot or print
y	Object to plot (not in use)
...	Additional argument

Value

An object of class `inla.surv`. There are methods for `print`, `plot` for `inla.surv` objects.

`is.inla.surv` returns `TRUE` if `object` inherits from class `inla.surv`, otherwise `FALSE`.

`as.inla.surv` returns an object of class `inla.surv`

Author(s)

Sara Martino and Rupali Akerkar

See Also

inla

Examples

```
## First example
trt = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
        1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)
time = c(17,42,44,48,60,72,74,95,103, 108, 122, 144, 167, 170, 183, 185,
        193, 195, 197, 208, 234, 235, 254, 307, 315, 401, 445, 464, 484,  528, 542, 567,
        577, 580, 795, 855, 1174, 1214, 1232, 1366, 1455, 1585, 1622, 1626, 1736, 1,63,
        105, 125, 182, 216, 250, 262, 301, 301, 342, 354, 356, 358, 380, 383, 383, 388,
        394, 408, 460, 489, 499, 523, 524, 535, 562, 569, 675, 676, 748, 778, 786, 797,
         955, 968, 977, 1245, 1271, 1420, 1460, 1516, 1551, 1690, 1694)
event = c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
          1,1,1,1,0,1,0,1,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
          1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,0,1)
y = inla.surv(time, event)

## Second example
time = c(182,182,63,68,182,152,182,130,134,145,152,182,98,152,182,88,95,105,130,137,167,182,
         152,182,81,182,71,84,126,134,152,182)
event = c(1,0,1,1,0,1,0,1,1,1,1,0,1,1,1,1,1,1,1,1,1,0,1,0,1,0,1,1,1,1,1,0)
subject = c(1,2,3,3,3,4,4,5,5,5,5,5,6,6,6,7,7,7,7,7,7,7,8,8,9,9,10,10,10,10,10,10)
y = inla.surv(time, event, subject=subject)
```

---

inla.upgrade *Upgrade the INLA-package*

Description

Functions to upgrade the INLA-package to the current version.

Usage

```
inla.upgrade(lib = NULL, testing=FALSE, ask = TRUE)
inla.update(lib = NULL, testing=FALSE, ask = TRUE)
```

Arguments

lib	Location to install the library.
testing	If TRUE, then look for a test-version if the INLA-package.
ask	same argument as in update.packages

Value

inla.upgrade will update the INLA package to the current version, and inla.update do the same for backward compatibility. This function is simple wrapper for update.packages using the INLA repository.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

update.packages

---

inla.version *Show the version of the INLA-package*

Description

Show the version of the INLA-package

Usage

```
inla.version(what = c("default",
                       "version",
                       "info",
                       "hgid",
                       "rinla",
                       "inla",
                       "date",
                       "bdate"))
```

Arguments

what	What to show version of

Value

inla.version either display the current version information using cat with default or info, or return the version number/information for other spesific requests through the call.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
## Summary of all
inla.version()
## The building date
inla.version("bdate")
```

`Kidney` *Kidney infection data*

Description

Times of infection from the time to insertion of the catheter for 38 kindey patients using portable dialysis equipment

Usage

```
data(Kidney)
```

Format

A data frame with 76 observations on the following 9 variables.

`time` a numeric vector. Time to infection from the insertion of catheter

`event` a numeric vector. 1: time of infection 0: time of censuring

`age` a numeric vector. Age of the patient at the time of infection

`sex` a numeric vector. Sex of the patient 0: male 1:female

`disease` a numeric vector. Type of disease

`dis1` a numeric vector. Dummy variable to codify the disease type.

`dis2` a numeric vector. Dummy variable to codify the disease type.

`dis3` a numeric vector. Dummy variable to codify the disease type.

`ID` a numeric vector. Patient code.

References

McGilchrist and C.W. Aisbett (1991), Regression with frailty in survival analysis, Biometrics,vol.47,pages 461–166.

D.J. Spiegelhalter and A. Thomas and N.G. Best and W.R. Gilks (1995) BUGS: Bayesian Inference Using Gibbs sampling, Version 0.50., MRC Biostatistics Unit, Cambridre, England.

`lattice2node` *Functions to define mapping between a lattice and nodes*

Description

These functions define mapping in between two-dimensional indices on a lattice and the one-dimensional node representation used in `inla`.

The mapping from node to lattice follows the default `R` behaviour (which is column based storage), and `as.vector(A)` and `matrix(a, nrow, ncol)` can be used instead of `inla.matrix2vector` and `inla.vector2matrix`.

Usage

```
inla.lattice2node.mapping(nrow, ncol)
inla.node2lattice.mapping(nrow, ncol)
inla.lattice2node(irow, icol, nrow, ncol)
inla.node2lattice(node, nrow, ncol)
inla.matrix2vector(a.matrix)
inla.vector2matrix(a.vector, nrow, ncol)
```

Arguments

nrow	Number of rows in the lattice.
ncol	Number of columns in the lattice.
irow	Lattice row index, between 1 and nrow
icol	Lattice column index, between 1 and ncol
node	The node index, between 1 and ncol*nrow
a.matrix	is a matrix to be mapped to a vector using internal representation defined by inla.lattice2node
a.vector	is a vector to be mapped into a matrix using the internal representation defined by inla.node2lattice

Value

inla.lattice2node.mapping returns the hole mapping as a matrix, and inla.node2lattice.mapping returns the hole mapping as list(irow=..., icol=...). inla.lattice2node and inla.node2lattice provide the mapping for a given set of lattice indices and nodes. inla.matrix2vector provide the mapped vector from a matrix, and inla.vector2matrix provide the inverse mapped matrix from vector.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla

Examples

```
## write out the mapping using the two alternatives
nrow = 2
ncol = 3
mapping = inla.lattice2node.mapping(nrow,ncol)

for (i in 1:nrow){
    for(j in 1:ncol){
        print(paste("Alt.1: lattice index [", i,",", j,"] corresponds",
                    "to node [", mapping[i,j],"]", sep=""))
    }
}

for (i in 1:nrow){
    for(j in 1:ncol){
```

```
        print(paste("Alt.2: lattice index [", i,",", j,"] corresponds to node [",
                    inla.lattice2node(i,j,nrow,ncol), "]", sep=""))
    }
}

inv.mapping = inla.node2lattice.mapping(nrow,ncol)
for(node in 1:(nrow*ncol))
   print(paste("Alt.1: node [", node, "] corresponds to lattice index [",
               inv.mapping$irow[node], ",",
               inv.mapping$icol[node],"]", sep=""))

for(node in 1:(nrow*ncol))
   print(paste("Alt.2: node [", node, "] corresponds to lattice index [",
               inla.node2lattice(node,nrow,ncol)$irow[1], ",",
               inla.node2lattice(node,nrow,ncol)$icol[1],"]", sep=""))

## apply the mapping from matrix to vector and back
n = nrow*ncol
z = matrix(1:n,nrow,ncol)
z.vector = inla.matrix2vector(z)  # as.vector(z) could also be used
print(mapping)
print(z)
print(z.vector)

## the vector2matrix is the inverse, and should give us the z-matrix
## back. matrix(z.vector, nrow, ncol) could also be used here.
z.matrix = inla.vector2matrix(z.vector, nrow, ncol)
print(z.matrix)
```

Leuk *The Leukemia data*

Description

This the Leukemia data from Henderson et al (2003); see source.

Usage

```
data(Leuk)
```

Format

A data frame with 1043 observations on the following 9 variables.

time TODO

cens TODO

xcoord TODO

ycoord TODO

age TODO

sex TODO

wbc TODO

tpi TODO

district TODO

Source

This is the dataset from

Henderson, R. and Shimakura, S. and Gorst, D., 2002, Modeling spatial variation in leukemia survival data, JASA, 97, 460, 965–972.

Examples

```
data(Leuk)
```

---

lines.inla.mesh.segment

Draw `inla.mesh.segment` *objects.*

Description

Draws a `inla.mesh.segment` object with generic or `rgl` graphics.

Usage

```
## S3 method for class 'inla.mesh.segment'
lines(x, loc = NULL, col = NULL,
     colors = c("black", "blue", "red", "green"),
     add = TRUE, xlim = NULL, ylim = NULL,
     rgl = FALSE, ...)
```

Arguments

`x`	An `inla.mesh.segment` object.
`loc`	Point locations to be used if `x$loc` is `NULL`.
`col`	Segment color specification.
`colors`	Colors to cycle through if `col` is `NULL`.
`add`	If `TRUE`, add to the current plot, otherwise start a new plot.
`xlim`	X axis limits for a new plot.
`ylim`	Y axis limits for a new plot.
`rgl`	If `TRUE`, use `rgl` for plotting.
`...`	Additional parameters, passed on to graphics methods.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

`inla.mesh.segment`

link *Link functions in INLA*

Description

Define link-functions and its inverse

Usage

```
inla.link.log(x, inverse=FALSE)
inla.link.invlog(x, inverse=FALSE)
inla.link.neglog(x, inverse=FALSE)
inla.link.invneglog(x, inverse=FALSE)
inla.link.logit(x, inverse=FALSE)
inla.link.invlogit(x, inverse=FALSE)
inla.link.probit(x, inverse=FALSE)
inla.link.invprobit(x, inverse=FALSE)
inla.link.cloglog(x, inverse=FALSE)
inla.link.invcloglog(x, inverse=FALSE)
inla.link.loglog(x, inverse=FALSE)
inla.link.invloglog(x, inverse=FALSE)
inla.link.tan(x, inverse=FALSE)
inla.link.invtan(x, inverse=FALSE)
inla.link.cauchit(x, inverse=FALSE)
inla.link.invcauchit(x, inverse=FALSE)
inla.link.identity(x, inverse=FALSE)
inla.link.invidentity(x, inverse=FALSE)
inla.link.invalid(x, inverse=FALSE)
inla.link.invinvalid(x, inverse=FALSE)
```

Arguments

`x`	The argument. A numeric vector.
`inverse`	Logical. Use the link (`inverse=FALSE`) or its inverse (`inverse=TRUE`)

Value

Return the values of the link-function or its inverse.

Note

The `inv`-functions are redundant, as `inla.link.invlog(x) = inla.link.log(x, inverse=TRUE)` and so on, but they are simpler to use a arguments to other functions.

Author(s)

Havard Rue <`hrue@r-inla.org`>

make.lincomb *Create linear combinations*

Description

Create a linear combination or several linear combinations, as input to `inla(..., lincomb = <lincomb>)`

Usage

```
inla.make.lincomb(...)
inla.make.lincombs(...)
```

Arguments

`...`	Arguments; see examples

Value

A structure to be passed on to `inla` argument `lincomb`

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

TODO

Examples

```
##See the worked out examples and description in the FAQ
##section on {www.r-inla.org}
```

marginal *Functions which operates on marginals*

Description

Density, distribution function, quantile function, random generation, hpd-interval, interpolation, expectations, mode and transformations of marginals obtained by `inla` or `inla.hyperpar()`. These functions computes the density (inla.dmarginal), the distribution function (inla.pmarginal), the quantile function (inla.qmarginal), random generation (inla.rmarginal), spline smoothing (inla.smarginal), computes expected values (inla.emarginal), computes the mode (inla.mmarginal), transforms the marginal (inla.tmarginal), and provide summary statistics (inla.zmarginal).

Usage

```
inla.dmarginal(x, marginal, log = FALSE)
inla.pmarginal(q, marginal, normalize = TRUE, len = 2048L)
inla.qmarginal(p, marginal, len = 2048L)
inla.rmarginal(n, marginal)
inla.hpdmarginal(p, marginal, len = 2048L)
inla.smarginal(marginal, log = FALSE, extrapolate = 0.0, keep.type = FALSE, factor=15L)
inla.emarginal(fun, marginal, ...)
inla.mmarginal(marginal)
inla.tmarginal(fun, marginal, n=2048L, h.diff = .Machine$double.eps^(1/3),
               method = c("quantile", "linear"))
inla.zmarginal(marginal, silent = FALSE)
```

Arguments

`marginal`	A marginal object from either `inla` or `inla.hyperpar()`, which is either `list(x=c(), y=c())` with density values `y` at locations `x`, or a `matrix(,n,2)` for which the density values are the second column and the locations in the first column. The`inla.hpdmarginal()`-function assumes a unimodal density.
`fun`	A (vectorised) function like `function(x) exp(x)` to compute the expectation against, or which define the transformation new = fun(old)
`x`	Evaluation points
`q`	Quantiles
`p`	Probabilities
`n`	The number of observations. If `length(n) > 1`, the length is taken to be the number required. For `inla.marginal.transform`, its the number of points to use in the new density.
`h.diff`	The step-length for the numerical differeniation inside `inla.marginal.transform`
`...`	Further arguments to be passed to function which expectation is to be computed.
`log`	Return density or interpolated density in log-scale?
`normalize`	Renormalise the density after interpolation?
`len`	Number of locations used to interpolate the distribution function.
`keep.type`	If `FALSE` then return a `list(x=, y=)`, otherwise if `TRUE`, then return a matrix if the input is a matrix
`extrapolate`	How much to extrapolate on each side when computing the interpolation. In fraction of the range.
`factor`	The number of points after interpolation is `factor` times the original number of points; which is argument `n` in `spline`
`method`	Which method should be used to layout points for where the transformation is computed.
`silent`	Output the result visually (TRUE) or just through the call.

Value

`inla.smarginal` returns `list=c(x=c(), y=c())` of interpolated values do extrapolation using the factor given, and the remaining function returns what they say they should do.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla, inla.hyperpar

Examples

```
## a simple linear regression example
n = 10
x = rnorm(n)
sd = 0.1
y = 1+x + rnorm(n,sd=sd)
res = inla(y ~ 1 + x, data = data.frame(x,y),
           control.family=list(initial = log(1/sd^2),fixed=TRUE))

## chose a marginal and compare the with the results computed by the
## inla-program
r = res$summary.fixed["x",]
m = res$marginals.fixed$x

## compute the 95% HPD interval
inla.hpdmarginal(0.95, m)

x = seq(-6, 6, len = 1000)
y = dnorm(x)
inla.hpdmarginal(0.95, list(x=x, y=y))

## compute the the density for exp(r), version 1
r.exp = inla.tmarginal(exp, m)
## or version 2
r.exp = inla.tmarginal(function(x) exp(x), m)

## to plot the marginal, we use the inla.smarginal, which interpolates (in
## log-scale). Compare with some samples.
plot(inla.smarginal(m), type="l")
s = inla.rmarginal(1000, m)
hist(inla.rmarginal(1000, m), add=TRUE, prob=TRUE)
lines(density(s), lty=2)

m1 = inla.emarginal(function(x) x^1, m)
m2 = inla.emarginal(function(x) x^2, m)
stdev = sqrt(m2 - m1^2)
q = inla.qmarginal(c(0.025,0.975), m)

## inla-program results
print(r)

## inla.marginal-results (they shouldn't be perfect!)
print(c(mean=m1, sd=stdev, "0.025quant" = q[1], "0.975quant" = q[2]))
## using the buildt-in function
inla.zmarginal(m)
```

Munich *The Munich rent data*

Description

The Munich rent data

Usage

```
data(Munich)
```

Format

A data frame with 2035 observations on the following 17 variables.

`rent` Net rent per square meter.

`floor.size` Size of the flat in square meters.

`year` Year of construction of the building in which the flat is located.

`location` Location index (in terms of subquarters).

`Gute.Wohnlage` Dummy variable for good locations / good neighborhoods.

`Beste.Wohnlage` Dummy variable for very good locations / very good neighborhoods.

`Keine.Wwv` Dummy for absence of warm water supply.

`Keine.Zh` Dummy for absence of central heating system.

`Kein.Badkach` Dummy for absence of flagging in the bathroom.

`Besond.Bad` Dummy for special features of the bathroom.

`Gehobene.Kueche` Dummy for more refined kitchen equipment.

`zim1` Dummy for a flat with 1 room.

`zim2` Dummy for a flat with 2 rooms.

`zim3` Dummy for a flat with 3 rooms.

`zim4` Dummy for a flat with 4 rooms.

`zim5` Dummy for a flat with 5 rooms.

`zim6` Dummy for a flat with 6 rooms.

Source

See Rue and Held (2005), Chapter 4.

References

Rue, H and Held, L. (2005) *Gaussian Markov Random Fields - Theory and Applications* Chapman and Hall

`nwEngland` *The New England map*

Description

This map is used in association to the Leukemia data from Henderson et al (2003); see source.

Usage

```
data(Leuk)
```

Format

A SpatialPolygons object.

Source

This map are used to analyse the Leukaemia dataset from

Henderson, R. and Shimakura, S. and Gorst, D., 2002, Modeling spatial variation in leukemia survival data, JASA, 97, 460, 965–972.

Examples

```
data(Leuk)
plot(nwEngland)
```

`Oral` *~~ data name/kind ... ~~*

Description

~~ A concise (1-5 lines) description of the dataset. ~~

Usage

```
data(Oral)
```

Format

A data frame with 544 observations on the following 3 variables.

`region` a numeric vector

`E` a numeric vector

`Y` a numeric vector

References

Rue, H and Held, L. (2005) *Gaussian Markov Random Fields - Theory and Applications* Chapman and Hall

param2.matern.orig *Parameter settings for* inla.spde2.matern *models.*

Description

Construct parameter settings for inla.spde2.matern models.

Usage

```
param2.matern.orig(mesh, alpha = 2,
                   B.tau = matrix(c(0, 1, 0), 1, 3),
                   B.kappa = matrix(c(0, 0, 1), 1, 3),
                   prior.variance.nominal = 1,
                   prior.range.nominal = NULL,
                   prior.tau = NULL,
                   prior.kappa = NULL,
                   theta.prior.mean = NULL,
                   theta.prior.prec = 0.1)
```

Arguments

mesh The mesh to build the model on, as an inla.mesh object.

alpha Fractional operator order, $0 < \alpha \leq 2$ supported. ($\nu = \alpha - d/2$)

B.tau Matrix with specification of log-linear model for τ.

B.kappa Matrix with specification of log-linear model for κ.

prior.variance.nominal
Nominal prior mean for the field variance

prior.range.nominal
Nominal prior mean for the spatial range

prior.tau Prior mean for tau (overrides prior.variance.nominal)

prior.kappa Prior mean for kappa (overrides prior.range.nominal)

theta.prior.mean
(overrides prior.*)

theta.prior.prec
Scalar, vector or matrix, specifying the joint prior precision for $theta$.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.spde2.matern

pc.ar	*Utility functions for the PC prior for a an AR(p) model*

Description

Functions to evaluate and sample from the PC prior for an AR(p) model

Usage

```
inla.pc.ar.rpacf(n=1, p, lambda = 1)
inla.pc.ar.dpacf(pac, lambda = 1, log = TRUE)
```

Arguments

`p`	The order of the AR-model
`pac`	A vector of partial autocorrelation coefficients
`n`	Number of observations
`lambda`	The rate parameter in the prior
`log`	Logical. Return the density in natural or log-scale.

Value

`inla.pc.ar.rpac` generate samples from the prior, returning a matrix where each row is a sample of `theta`. `inla.pc.ar.dpac` evaluates the density of `pac`. Use `inla.ar.pacf2phi`, `inla.ar.phi2pacf`, `inla.ar.pacf2acf` and `inla.ar.acf2pacf` to convert between various parameterisations.

Author(s)

Havard Rue `<hrue@r-inla.org>`

pc.cor0	*Utility functions for the PC prior for correlation in AR(1)*

Description

Functions to evaluate, sample, compute quantiles and percentiles of the PC prior for the correlation in the Gaussian AR(1) model where the base-model is zero correlation.

Usage

```
inla.pc.rcor0(n, u, alpha, lambda)
inla.pc.dcor0(cor, u, alpha, lambda, log = FALSE)
inla.pc.qcor0(p, u, alpha, lambda)
inla.pc.pcor0(q, u, alpha, lambda)
```

Arguments

n	Number of observations
u	The upper limit (see Details)
alpha	The probability going above the upper limit (see Details)
lambda	The rate parameter (see Details)
cor	Vector of correlations
log	Logical. Return the density in natural or log-scale.
p	Vector of probabilities
q	Vector of quantiles

Details

The statement `Prob(|cor| > u) = alpha` is used to determine `lambda` unless `lambda` is given. Either `lambda` must be given, or `u` AND `alpha`. The density is symmetric around zero.

Value

`inla.pc.dcor0` gives the density, `inla.pc.pcor0` gives the distribution function, `inla.pc.qcor0` gives the quantile function, and `inla.pc.rcor0` generates random deviates.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla.doc("pc.rho0")

Examples

```
cor = inla.pc.rcor0(100,  lambda = 1)
d = inla.pc.dcor0(cor, lambda = 1)
cor = inla.pc.qcor0(c(0.3, 0.7), u = 0.5, alpha=0.01)
inla.pc.pcor0(cor, u = 0.5, alpha=0.01)
```

---

pc.cor1 *Utility functions for the PC prior for correlation in AR(1)*

Description

Functions to evaluate, sample, compute quantiles and percentiles of the PC prior for the correlation in the Gaussian AR(1) model where the base-model is correlation one.

Usage

```
inla.pc.rcor1(n, u, alpha, lambda)
inla.pc.dcor1(cor, u, alpha, lambda, log = FALSE)
inla.pc.qcor1(p, u, alpha, lambda)
inla.pc.pcor1(q, u, alpha, lambda)
```

Arguments

n	Number of observations
u	The upper limit (see Details)
alpha	The probability going above the upper limit (see Details)
lambda	The rate parameter (see Details)
cor	Vector of correlations
log	Logical. Return the density in natural or log-scale.
p	Vector of probabilities
q	Vector of quantiles

Details

The statement `Prob(cor > u) = alpha` is used to determine `lambda` unless `lambda` is given. Either `lambda` must be given, or `u` AND `alpha`.

Value

`inla.pc.dcor1` gives the density, `inla.pc.pcor1` gives the distribution function, `inla.pc.qcor1` gives the quantile function, and `inla.pc.rcor1` generates random deviates.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla.doc("pc.rho1")

Examples

```
cor = inla.pc.rcor1(100,  lambda = 1)
d = inla.pc.dcor1(cor, lambda = 1)
cor = inla.pc.qcor1(c(0.3, 0.7), u = 0.5, alpha=0.75)
inla.pc.pcor1(cor, u = 0.5, alpha=0.75)
```

---

pc.cormat — *Utility functions for the PC prior for a correlation matrix*

Description

Functions to evaluate and sample from the PC prior for a correlation matrix.

Usage

```
inla.pc.cormat.dim2p(dim)
inla.pc.cormat.p2dim(p)
inla.pc.cormat.theta2R(theta)
inla.pc.cormat.R2theta(R)
inla.pc.cormat.r2R(r)
inla.pc.cormat.R2r(R)
inla.pc.cormat.r2theta(r)
inla.pc.cormat.theta2r(theta)
inla.pc.cormat.permute(R)
inla.pc.cormat.rtheta(n=1, p, lambda = 1)
inla.pc.cormat.dtheta(theta, lambda = 1, log = FALSE)
```

Arguments

`dim`	The dimension of `theta`, the parameterisatin of the correlation matrix
`p`	The dimension the correlation matrix
`theta`	A vector of parameters for the correlation matrix
`r`	The off diagonal elements of a correlation matrix
`R`	A correlation matrix
`n`	Number of observations
`lambda`	The rate parameter in the prior
`log`	Logical. Return the density in natural or log-scale.

Details

The parameterisation of a correlation matrix of dimension `p` has `dim` parameters: `theta` which are in the interval -pi to pi. The alternative parameterisation is through the off-diagonal elements `r` of the correlation matrix `R`. The functions `inla.pc.cormat.<A>2<B>` convert between parameterisations <A> to parameterisations <B>, where both <A> and <B> are one of `theta`, `r` and `R`, and `p` and `dim`.

Value

`inla.pc.cormat.rtheta` generate samples from the prior, returning a matrix where each row is a sample of `theta`. `inla.pc.cormat.dtheta` evaluates the density of `theta`. `inla.pc.cormat.permute` randomly permutes a correlation matrix, which is useful if an exchangable sample of a correlation matrix is required.

Author(s)

Havard Rue `<hrue@r-inla.org>`

Examples

```
p = 4
print(paste("theta has length", inla.pc.cormat.p2dim(p)))
theta = inla.pc.cormat.rtheta(n=1, p=4, lambda = 1)
print("sample theta:")
print(theta)
print(paste("log.dens", inla.pc.cormat.dtheta(theta, log=TRUE)))
```

```
print("r:")
r = inla.pc.cormat.theta2r(theta)
print(r)
print("A sample from the non-exchangable prior, R:")
R = inla.pc.cormat.r2R(r)
print(R)
print("A sample from the exchangable prior, R:")
R = inla.pc.cormat.permute(R)
print(R)
```

---

pc.ddof *PC-prior for dof in a standarized Student-t*

Description

A function to evaluate the PC-prior for the degrees of freedom in a standarized Student-t distribution

Usage

```
inla.pc.ddof(dof, lambda, u, alpha, log=FALSE)
```

Arguments

`dof`	Degrees of freedom
`log`	Logical. Return the density or the log-density
`lambda`	The optional value of `lambda`, instead of defining it implicitely through `u` and `alpha`
`u`	The upper value of dof used to elicitate `lambda`, `Prob(dof < u) = alpha`
`alpha`	The probability `alpha` used to elicitate `lambda`

Details

These functions implements the PC-prior for the dof in a standarized Student-t distribution (ie. with unit variance and `dof` > 2). Either `lambda`, or `u` AND `alpha` must be given. Due the internal tabulation, `dof` must be larger than 2.0025.

Value

`inla.pc.ddof` returns the prior density for given `dof`.

Author(s)

Havard Rue <hrue@r-inla.org>

pc.gamma *Utility functions for the PC prior for* `Gamma(1/a, 1/a)`

Description

Functions to evaluate, sample, compute quantiles and percentiles of the PC prior for `Gamma(1/a, 1/a)`

Usage

```
inla.pc.rgamma(n, lambda = 1)
inla.pc.dgamma(x, lambda = 1, = FALSE)
inla.pc.qgamma(p, lambda = 1)
inla.pc.pgamma(q, lambda = 1)
```

Arguments

`n`	Number of observations
`lambda`	The rate parameter (see Details)
`x`	Evaluation points
`log`	Logical. Return the density in natural or log-scale.
`p`	Vector of probabilities
`q`	Vector of quantiles

Details

This gives the PC prior for the `Gamma(1/a, 1/a)` case, where `a=0` is the base model.

Value

`inla.pc.dgamma` gives the density, `inla.pc.pgamma` gives the distribution function, `inla.pc.qgamma` gives the quantile function, and `inla.pc.rgamma` generates random deviates.

Author(s)

Havard Rue `<hrue@r-inla.org>`

See Also

inla.doc("pc.gamma")

Examples

```
x = inla.pc.rgamma(100,  lambda = 1)
d = inla.pc.dgamma(x, lambda = 1)
x = inla.pc.qgamma(0.5, lambda = 1)
inla.pc.pgamma(x, lambda = 1)
```

`pc.gammacount` *Utility functions for the PC prior for the* `gammacount` *likelihood*

Description

Functions to evaluate, sample, compute quantiles and percentiles of the PC prior for the `gammacount` likelihood

Usage

```
inla.pc.rgammacount(n, lambda = 1)
inla.pc.dgammacount(x, lambda = 1, = FALSE)
inla.pc.qgammacount(p, lambda = 1)
inla.pc.pgammacount(q, lambda = 1)
```

Arguments

`n`	Number of observations
`lambda`	The rate parameter (see Details)
`x`	Evaluation points
`log`	Logical. Return the density in natural or log-scale.
`p`	Vector of probabilities
`q`	Vector of quantiles

Details

This gives the PC prior for the `gammacount` likelihood, which is the PC prior for `a` in `Gamma(a, 1)` where `Gamma(1, 1)` is the base model.

Value

`inla.pc.dgammacount` gives the density, `inla.pc.pgammacount` gives the distribution function, `inla.pc.qgammacount` gives the quantile function, and `inla.pc.rgammacount` generates random deviates.

Author(s)

Havard Rue <`hrue@r-inla.org`>

See Also

inla.doc("pc.gammacount")

Examples

```
x = inla.pc.rgammacount(100,  lambda = 1)
d = inla.pc.dgammacount(x, lambda = 1)
x = inla.pc.qgammacount(0.5, lambda = 1)
inla.pc.pgammacount(x, lambda = 1)
```

pc.multvar *Multivariate PC priors*

Description

Functions to evaluate and simulate from multivariate PC priors: The simplex and sphere case

Usage

```
inla.pc.multvar.h.default(x, inverse = FALSE, derivative = FALSE)
inla.pc.multvar.simplex.r(n = NULL, lambda = 1, h = inla.pc.multvar.h.default, b = NULL)
inla.pc.multvar.simplex.d(x = NULL, lambda = 1, log = FALSE, h = inla.pc.multvar.h.default, b = NU
inla.pc.multvar.sphere.r(n = NULL, lambda = 1, h = inla.pc.multvar.h.default, H = NULL)
inla.pc.multvar.sphere.d(x = NULL, lambda = 1, log = FALSE, h = inla.pc.multvar.h.default, H = NUL
```

Arguments

`x`	Samples to evaluate. If input is a matrix then each row is a sample. If input is a vector then this is the sample.
`inverse`	Compute the inverse of the h()-function.
`derivative`	Compute the derivative of the h()-function. (derivative of the inverse function is not used).
`n`	Number of samples to generate.
`lambda`	The lambda-parameter in the PC-prior.
`log`	Evaluate the density in log-scale or ordinary scale.
`h`	The h()-function, defaults to `inla.pc.multvar.h.default`. See that code for an example of how to write a user-spesific function.
`b`	The b-vector (gradient) in the expression for the simplex option, `d(xi) = h(b^T xi)`
`H`	The H(essian)-matrix in the expression for the sphere option, `d(xi) =` `h(1/2 *xi^T H xi` If `H` is a vector, then it is interpreted as the diagonal of a (sparse) diagonal matrix.

Details

These functions implements multivariate PC-priors of the simplex and sphere type.

Value

`inla.pc.multvar.simplex.r` generate samples from the simplex case, and `inla.pc.multvar.simplex.d` evaluate the density. `inla.pc.multvar.sphere.r` generate samples from the sphere case, and `inla.pc.multvar.sphere.d` evaluate the density. `inla.pc.multvar.h.default` implements the default h()-function and illustrate how to code your own spesific one, if needed.

Author(s)

Havard Rue <hrue@r-inla.org>

pc.prec *Utility functions for the PC prior for the precision*

Description

Functions to evaluate, sample, compute quantiles and percentiles of the PC prior for the precision in the Gaussian distribution.

Usage

```
inla.pc.rprec(n, u, alpha, lambda)
inla.pc.dprec(prec, u, alpha, lambda, log = FALSE)
inla.pc.qprec(p, u, alpha, lambda)
inla.pc.pprec(q, u, alpha, lambda)
```

Arguments

n	Number of observations
u	The upper limit (see Details)
alpha	The probability going above the upper limit (see Details)
lambda	The rate parameter (see Details)
prec	Vector of precisions
log	Logical. Return the density in natural or log-scale.
p	Vector of probabilities
q	Vector of quantiles

Details

The statement `Prob(1/sqrt(prec) > u) = alpha` is used to determine `lambda` unless `lambda` is given. Either `lambda` must be given, or `u` AND `alpha`.

Value

`inla.pc.dprec` gives the density, `inla.pc.pprec` gives the distribution function, `inla.pc.qprec` gives the quantile function, and `inla.pc.rprec` generates random deviates.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla.doc("pc.prec")

Examples

```
prec = inla.pc.rprec(100,  lambda = 1)
d = inla.pc.dprec(prec, lambda = 1)
prec = inla.pc.qprec(0.5, u = 1, alpha=0.01)
inla.pc.pprec(prec, u = 1, alpha=0.01)
```

plot.inla *Default INLA plotting*

Description

Takes am `inla` object produced by `inla` and plot the results

Usage

```
 ## S3 method for class 'inla'
plot(x,
              plot.fixed.effects = TRUE,
              plot.lincomb = TRUE,
              plot.random.effects = TRUE,
              plot.hyperparameters = TRUE,
              plot.predictor = TRUE,
              plot.q = TRUE,
              plot.cpo = TRUE,
              plot.prior = FALSE,
              single = FALSE,
              postscript = FALSE,
              pdf = FALSE,
              prefix = "inla.plots/figure-",
              intern = FALSE,
              debug = FALSE,
              ...)
```

Arguments

- `x` A fitted `inla` object produced by `inla`
- `plot.fixed.effects` Boolean indicating if posterior marginals for the fixed effects in the model should be plotted
- `plot.lincomb` Boolean indicating if posterior marginals for the linear combinations should be plotted
- `plot.random.effects` Boolean indicating if posterior mean and quantiles for the random effects in the model should be plotted
- `plot.hyperparameters` Boolean indicating if posterior marginals for the hyperparameters in the model should be plotted
- `plot.predictor` Boolean indicating if posterior mean and quantiles for the linear predictor in the model should be plotted
- `plot.q` Boolean indicating if precision matrix should be displayed
- `plot.cpo` Boolean indicating if CPO/PIT valuesshould be plotted
- `plot.prior` Plot also the prior density for the hyperparameters
- `single` Boolean indicating if there should be more than one plot per page (FALSE) or just one (TRUE)

postscript	Boolean indicating if postscript files should be produced instead
pdf	Boolean indicating if PDF files should be produced instead
prefix	The prefix for the created files. Additional numbering and suffix is added.
intern	Plot also the hyperparameters in its internal scale.
debug	Write some debug information
...	Additional arguments to postscript(), pdf() or dev.new().

Value

The return value is a list of the files created (if any).

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla

Examples

```
## Not run:
result = inla(...)
plot(result)
plot(result, single=TRUE)
plot(result, single=TRUE, pdf=TRUE, paper = "a4")

## End(Not run)
```

---

plot.inla.CRS *Plot CRS and inla.CRS objects*

Description

Plot the outline of a CRS or inla.CRS projection, with optional graticules (transformed parallels and meridians) and Tissot indicatrices.

Usage

```
## S3 method for class 'inla.CRS'
plot(x,
     xlim = NULL, ylim = NULL,
     outline = TRUE,
     graticule = c(15, 15, 45),
     tissot = c(30, 30, 30),
     asp = 1, add = FALSE,
     eps=0.05, ...)
## S3 method for class 'CRS'
plot(x,
     xlim = NULL, ylim = NULL,
```

```
outline = TRUE,
graticule = c(15, 15, 45),
tissot = c(30, 30, 30),
asp = 1, add = FALSE,
eps=0.05, ...)
```

Arguments

x	A CRS or inla.CRS object.
xlim	Optional x-axis limits.
ylim	Optional y-axis limits.
outline	Logical, if TRUE, draw the outline of the projection.
graticule	Vector of length at most 3, to plot meridians with spacing graticule[1] degrees and parallels with spacing graticule[2] degrees. graticule[3] optionally specifies the spacing above and below the first and last parallel. When graticule[1]==0 no meridians are drawn, and when graticule[2]==0 no parallels are drawn. Use graticule=NULL to skip drawing a graticule.
tissot	Vector of length at most 3, to plot Tissot's indicatrices with spacing tissot[1] degrees and parallels with spacing tissot[2] degrees. tissot[3] specifices a scaling factor. Use tissot=NULL to skip drawing a Tissot's indicatrices.
asp	The aspect ratio for the plot, default 1.
add	If TRUE, add the projecton plot to an existing plot.
eps	Clipping tolerance for rudimentary boundary clipping
...	Additional arguments passed on to the internal calls to plot and lines.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.CRS

Examples

```
if (require(rgdal)) {
  oblique <- c(0,45,45,0)
  par(mfrow=c(4,2))
  for (projtype in c("longlat", "lambert", "mollweide", "hammer")) {
    plot(inla.CRS(projtype), main=projtype)
    plot(inla.CRS(projtype, oblique=oblique), main=paste("oblique", projtype))
  }
  par(mfrow=c(1,1))
}
```

plot.inla.mesh	*Draw a triangulation mesh object*

Description

Plots an inla.mesh object using either standard graphics or with rgl.

Usage

```
## S3 method for class 'inla.mesh'
plot(x,
     col = "white",
     t.sub = 1:nrow(mesh$graph$tv),
     add = FALSE,
     lwd = 1,
     xlim = range(mesh$loc[, 1]),
     ylim = range(mesh$loc[, 2]),
     main = NULL,
     rgl = FALSE,
     size = 2,
     draw.vertices = FALSE,
     vertex.color = "black",
     draw.edges = TRUE,
     edge.color = rgb(0.3, 0.3, 0.3),
     draw.segments = draw.edges, ...)
```

Arguments

x	An inla.mesh object.
col	Color specification. A single named color, a vector of scalar values, or a matrix of RGB values. Requires rgl=TRUE.
t.sub	Optional triangle index subset to be drawn.
add	If TRUE, adds to the current plot instead of starting a new one.
lwd	Line width for triangle edges.
xlim	X-axis limits.
ylim	Y-axis limits.
main	The main plot title. If not specified, a default title is generated based on the mesh type.
rgl	When TRUE, generates an rgl plot instead of a generic graphics plot. Allows 3D plotting and color surface plotting.
size	Size of vertex points in rgl plotting. See rgl.material.
draw.vertices	If TRUE, draw triengle vertices.
vertex.color	Color specification for all vertices.
draw.edges	If TRUE, draw triangle edges.
edge.color	Color specification for all edges.
draw.segments	If TRUE, draw boundary and interior constraint edges more prominently.
...	Further graphics parameters, interpreted by the respective plotting systems.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

plot.inla.trimesh

Examples

```
mesh = inla.mesh.create(globe=10)
plot(mesh)

if (require(rgl)) {
  plot(mesh, rgl=TRUE, col=mesh$loc[,1])
}
```

---

plot.inla.trimesh *Low level triangulation mesh plotting*

Description

Plots a triangulation mesh using rgl.

Usage

```
## S3 method for class 'inla.trimesh'
plot(x, S,
     color = NULL, color.axis = NULL,
     color.n = 512, color.palette = cm.colors, color.truncate = FALSE,
     alpha = NULL, lwd = 1, specular = "black",
     draw.vertices = TRUE,
     draw.edges = TRUE, edge.color = rgb(0.3, 0.3, 0.3),
     ...)
```

Arguments

x	A 3-column triangle-to-vertex index map matrix.
S	A 3-column vertex coordinate matrix.
color	Color specification. A single named color, a vector of scalar values, or a matrix of RGB values.
color.axis	The min/max limit values for the color mapping.
color.n	The number of colors to use in the color palette.
color.palette	A color palette function.
color.truncate	If TRUE, truncate the colors at the color axis limits.
alpha	Transparency/opaqueness values. See rgl.material.
lwd	Line width for edges. See rgl.material.
specular	Specular color. See rgl.material.
draw.vertices	If TRUE, draw triangle vertices.

`draw.edges`	If `TRUE`, draw triangle edges.
`edge.color`	Edge color specification.
`...`	Additional parameters passed to and from other methods.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

`plot.inla.mesh`

PRborder *The PRborder data*

Description

A data matrix with Longitude and Latitude coordinates for the boundary of Parana State.

Usage

```
data(PRborder)
```

Format

Longtiude The Longtiude coordinate

Latitude The Latitude coordinate

See Also

PRprec

---

print.inla *Print a INLA fit*

Description

Print a INLA fit

Usage

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

Arguments

`x`	An inla-object (output from an `inla`-call).
`...`	other arguments.

Details

None

Value

None

Author(s)

Havard Rue

See Also

inla

Examples

```
## None
```

PRprec *The PRprec data*

Description

A data frame with daily rainfall in the Parana State.

Usage

```
data(PRprec)
```

Format

A data frame .... TODO

Altitude TODO

Latitude TODO

Longitude TODO

d0101 Daily rainfall at day "mmdd"

d0102 Daily rainfall at day "mmdd"

d0103 Daily rainfall at day "mmdd"

d0104 Daily rainfall at day "mmdd"

d0105 Daily rainfall at day "mmdd"

d0106 Daily rainfall at day "mmdd"

d0107 Daily rainfall at day "mmdd"

d0108 Daily rainfall at day "mmdd"

d0109 Daily rainfall at day "mmdd"

d0110 Daily rainfall at day "mmdd"

d0111 Daily rainfall at day "mmdd"

d0112 Daily rainfall at day "mmdd"
d0113 Daily rainfall at day "mmdd"
d0114 Daily rainfall at day "mmdd"
d0115 Daily rainfall at day "mmdd"
d0116 Daily rainfall at day "mmdd"
d0117 Daily rainfall at day "mmdd"
d0118 Daily rainfall at day "mmdd"
d0119 Daily rainfall at day "mmdd"
d0120 Daily rainfall at day "mmdd"
d0121 Daily rainfall at day "mmdd"
d0122 Daily rainfall at day "mmdd"
d0123 Daily rainfall at day "mmdd"
d0124 Daily rainfall at day "mmdd"
d0125 Daily rainfall at day "mmdd"
d0126 Daily rainfall at day "mmdd"
d0127 Daily rainfall at day "mmdd"
d0128 Daily rainfall at day "mmdd"
d0129 Daily rainfall at day "mmdd"
d0130 Daily rainfall at day "mmdd"
d0131 Daily rainfall at day "mmdd"
d0201 Daily rainfall at day "mmdd"
d0202 Daily rainfall at day "mmdd"
d0203 Daily rainfall at day "mmdd"
d0204 Daily rainfall at day "mmdd"
d0205 Daily rainfall at day "mmdd"
d0206 Daily rainfall at day "mmdd"
d0207 Daily rainfall at day "mmdd"
d0208 Daily rainfall at day "mmdd"
d0209 Daily rainfall at day "mmdd"
d0210 Daily rainfall at day "mmdd"
d0211 Daily rainfall at day "mmdd"
d0212 Daily rainfall at day "mmdd"
d0213 Daily rainfall at day "mmdd"
d0214 Daily rainfall at day "mmdd"
d0215 Daily rainfall at day "mmdd"
d0216 Daily rainfall at day "mmdd"
d0217 Daily rainfall at day "mmdd"
d0218 Daily rainfall at day "mmdd"
d0219 Daily rainfall at day "mmdd"
d0220 Daily rainfall at day "mmdd"

d0221 Daily rainfall at day "mmdd"

d0222 Daily rainfall at day "mmdd"

d0223 Daily rainfall at day "mmdd"

d0224 Daily rainfall at day "mmdd"

d0225 Daily rainfall at day "mmdd"

d0226 Daily rainfall at day "mmdd"

d0227 Daily rainfall at day "mmdd"

d0228 Daily rainfall at day "mmdd"

d0301 Daily rainfall at day "mmdd"

d0302 Daily rainfall at day "mmdd"

d0303 Daily rainfall at day "mmdd"

d0304 Daily rainfall at day "mmdd"

d0305 Daily rainfall at day "mmdd"

d0306 Daily rainfall at day "mmdd"

d0307 Daily rainfall at day "mmdd"

d0308 Daily rainfall at day "mmdd"

d0309 Daily rainfall at day "mmdd"

d0310 Daily rainfall at day "mmdd"

d0311 Daily rainfall at day "mmdd"

d0312 Daily rainfall at day "mmdd"

d0313 Daily rainfall at day "mmdd"

d0314 Daily rainfall at day "mmdd"

d0315 Daily rainfall at day "mmdd"

d0316 Daily rainfall at day "mmdd"

d0317 Daily rainfall at day "mmdd"

d0318 Daily rainfall at day "mmdd"

d0319 Daily rainfall at day "mmdd"

d0320 Daily rainfall at day "mmdd"

d0321 Daily rainfall at day "mmdd"

d0322 Daily rainfall at day "mmdd"

d0323 Daily rainfall at day "mmdd"

d0324 Daily rainfall at day "mmdd"

d0325 Daily rainfall at day "mmdd"

d0326 Daily rainfall at day "mmdd"

d0327 Daily rainfall at day "mmdd"

d0328 Daily rainfall at day "mmdd"

d0329 Daily rainfall at day "mmdd"

d0330 Daily rainfall at day "mmdd"

d0331 Daily rainfall at day "mmdd"

d0401 Daily rainfall at day "mmdd"

d0402 Daily rainfall at day "mmdd"

d0403 Daily rainfall at day "mmdd"

d0404 Daily rainfall at day "mmdd"

d0405 Daily rainfall at day "mmdd"

d0406 Daily rainfall at day "mmdd"

d0407 Daily rainfall at day "mmdd"

d0408 Daily rainfall at day "mmdd"

d0409 Daily rainfall at day "mmdd"

d0410 Daily rainfall at day "mmdd"

d0411 Daily rainfall at day "mmdd"

d0412 Daily rainfall at day "mmdd"

d0413 Daily rainfall at day "mmdd"

d0414 Daily rainfall at day "mmdd"

d0415 Daily rainfall at day "mmdd"

d0416 Daily rainfall at day "mmdd"

d0417 Daily rainfall at day "mmdd"

d0418 Daily rainfall at day "mmdd"

d0419 Daily rainfall at day "mmdd"

d0420 Daily rainfall at day "mmdd"

d0421 Daily rainfall at day "mmdd"

d0422 Daily rainfall at day "mmdd"

d0423 Daily rainfall at day "mmdd"

d0424 Daily rainfall at day "mmdd"

d0425 Daily rainfall at day "mmdd"

d0426 Daily rainfall at day "mmdd"

d0427 Daily rainfall at day "mmdd"

d0428 Daily rainfall at day "mmdd"

d0429 Daily rainfall at day "mmdd"

d0430 Daily rainfall at day "mmdd"

d0501 Daily rainfall at day "mmdd"

d0502 Daily rainfall at day "mmdd"

d0503 Daily rainfall at day "mmdd"

d0504 Daily rainfall at day "mmdd"

d0505 Daily rainfall at day "mmdd"

d0506 Daily rainfall at day "mmdd"

d0507 Daily rainfall at day "mmdd"

d0508 Daily rainfall at day "mmdd"

d0509 Daily rainfall at day "mmdd"

d0510 Daily rainfall at day "mmdd"

d0511 Daily rainfall at day "mmdd"

d0512 Daily rainfall at day "mmdd"
d0513 Daily rainfall at day "mmdd"
d0514 Daily rainfall at day "mmdd"
d0515 Daily rainfall at day "mmdd"
d0516 Daily rainfall at day "mmdd"
d0517 Daily rainfall at day "mmdd"
d0518 Daily rainfall at day "mmdd"
d0519 Daily rainfall at day "mmdd"
d0520 Daily rainfall at day "mmdd"
d0521 Daily rainfall at day "mmdd"
d0522 Daily rainfall at day "mmdd"
d0523 Daily rainfall at day "mmdd"
d0524 Daily rainfall at day "mmdd"
d0525 Daily rainfall at day "mmdd"
d0526 Daily rainfall at day "mmdd"
d0527 Daily rainfall at day "mmdd"
d0528 Daily rainfall at day "mmdd"
d0529 Daily rainfall at day "mmdd"
d0530 Daily rainfall at day "mmdd"
d0531 Daily rainfall at day "mmdd"
d0601 Daily rainfall at day "mmdd"
d0602 Daily rainfall at day "mmdd"
d0603 Daily rainfall at day "mmdd"
d0604 Daily rainfall at day "mmdd"
d0605 Daily rainfall at day "mmdd"
d0606 Daily rainfall at day "mmdd"
d0607 Daily rainfall at day "mmdd"
d0608 Daily rainfall at day "mmdd"
d0609 Daily rainfall at day "mmdd"
d0610 Daily rainfall at day "mmdd"
d0611 Daily rainfall at day "mmdd"
d0612 Daily rainfall at day "mmdd"
d0613 Daily rainfall at day "mmdd"
d0614 Daily rainfall at day "mmdd"
d0615 Daily rainfall at day "mmdd"
d0616 Daily rainfall at day "mmdd"
d0617 Daily rainfall at day "mmdd"
d0618 Daily rainfall at day "mmdd"
d0619 Daily rainfall at day "mmdd"
d0620 Daily rainfall at day "mmdd"

d0621 Daily rainfall at day "mmdd"
d0622 Daily rainfall at day "mmdd"
d0623 Daily rainfall at day "mmdd"
d0624 Daily rainfall at day "mmdd"
d0625 Daily rainfall at day "mmdd"
d0626 Daily rainfall at day "mmdd"
d0627 Daily rainfall at day "mmdd"
d0628 Daily rainfall at day "mmdd"
d0629 Daily rainfall at day "mmdd"
d0630 Daily rainfall at day "mmdd"
d0701 Daily rainfall at day "mmdd"
d0702 Daily rainfall at day "mmdd"
d0703 Daily rainfall at day "mmdd"
d0704 Daily rainfall at day "mmdd"
d0705 Daily rainfall at day "mmdd"
d0706 Daily rainfall at day "mmdd"
d0707 Daily rainfall at day "mmdd"
d0708 Daily rainfall at day "mmdd"
d0709 Daily rainfall at day "mmdd"
d0710 Daily rainfall at day "mmdd"
d0711 Daily rainfall at day "mmdd"
d0712 Daily rainfall at day "mmdd"
d0713 Daily rainfall at day "mmdd"
d0714 Daily rainfall at day "mmdd"
d0715 Daily rainfall at day "mmdd"
d0716 Daily rainfall at day "mmdd"
d0717 Daily rainfall at day "mmdd"
d0718 Daily rainfall at day "mmdd"
d0719 Daily rainfall at day "mmdd"
d0720 Daily rainfall at day "mmdd"
d0721 Daily rainfall at day "mmdd"
d0722 Daily rainfall at day "mmdd"
d0723 Daily rainfall at day "mmdd"
d0724 Daily rainfall at day "mmdd"
d0725 Daily rainfall at day "mmdd"
d0726 Daily rainfall at day "mmdd"
d0727 Daily rainfall at day "mmdd"
d0728 Daily rainfall at day "mmdd"
d0729 Daily rainfall at day "mmdd"
d0730 Daily rainfall at day "mmdd"

d0731 Daily rainfall at day "mmdd"
d0801 Daily rainfall at day "mmdd"
d0802 Daily rainfall at day "mmdd"
d0803 Daily rainfall at day "mmdd"
d0804 Daily rainfall at day "mmdd"
d0805 Daily rainfall at day "mmdd"
d0806 Daily rainfall at day "mmdd"
d0807 Daily rainfall at day "mmdd"
d0808 Daily rainfall at day "mmdd"
d0809 Daily rainfall at day "mmdd"
d0810 Daily rainfall at day "mmdd"
d0811 Daily rainfall at day "mmdd"
d0812 Daily rainfall at day "mmdd"
d0813 Daily rainfall at day "mmdd"
d0814 Daily rainfall at day "mmdd"
d0815 Daily rainfall at day "mmdd"
d0816 Daily rainfall at day "mmdd"
d0817 Daily rainfall at day "mmdd"
d0818 Daily rainfall at day "mmdd"
d0819 Daily rainfall at day "mmdd"
d0820 Daily rainfall at day "mmdd"
d0821 Daily rainfall at day "mmdd"
d0822 Daily rainfall at day "mmdd"
d0823 Daily rainfall at day "mmdd"
d0824 Daily rainfall at day "mmdd"
d0825 Daily rainfall at day "mmdd"
d0826 Daily rainfall at day "mmdd"
d0827 Daily rainfall at day "mmdd"
d0828 Daily rainfall at day "mmdd"
d0829 Daily rainfall at day "mmdd"
d0830 Daily rainfall at day "mmdd"
d0831 Daily rainfall at day "mmdd"
d0901 Daily rainfall at day "mmdd"
d0902 Daily rainfall at day "mmdd"
d0903 Daily rainfall at day "mmdd"
d0904 Daily rainfall at day "mmdd"
d0905 Daily rainfall at day "mmdd"
d0906 Daily rainfall at day "mmdd"
d0907 Daily rainfall at day "mmdd"
d0908 Daily rainfall at day "mmdd"

d0909 Daily rainfall at day "mmdd"

d0910 Daily rainfall at day "mmdd"

d0911 Daily rainfall at day "mmdd"

d0912 Daily rainfall at day "mmdd"

d0913 Daily rainfall at day "mmdd"

d0914 Daily rainfall at day "mmdd"

d0915 Daily rainfall at day "mmdd"

d0916 Daily rainfall at day "mmdd"

d0917 Daily rainfall at day "mmdd"

d0918 Daily rainfall at day "mmdd"

d0919 Daily rainfall at day "mmdd"

d0920 Daily rainfall at day "mmdd"

d0921 Daily rainfall at day "mmdd"

d0922 Daily rainfall at day "mmdd"

d0923 Daily rainfall at day "mmdd"

d0924 Daily rainfall at day "mmdd"

d0925 Daily rainfall at day "mmdd"

d0926 Daily rainfall at day "mmdd"

d0927 Daily rainfall at day "mmdd"

d0928 Daily rainfall at day "mmdd"

d0929 Daily rainfall at day "mmdd"

d0930 Daily rainfall at day "mmdd"

d1001 Daily rainfall at day "mmdd"

d1002 Daily rainfall at day "mmdd"

d1003 Daily rainfall at day "mmdd"

d1004 Daily rainfall at day "mmdd"

d1005 Daily rainfall at day "mmdd"

d1006 Daily rainfall at day "mmdd"

d1007 Daily rainfall at day "mmdd"

d1008 Daily rainfall at day "mmdd"

d1009 Daily rainfall at day "mmdd"

d1010 Daily rainfall at day "mmdd"

d1011 Daily rainfall at day "mmdd"

d1012 Daily rainfall at day "mmdd"

d1013 Daily rainfall at day "mmdd"

d1014 Daily rainfall at day "mmdd"

d1015 Daily rainfall at day "mmdd"

d1016 Daily rainfall at day "mmdd"

d1017 Daily rainfall at day "mmdd"

d1018 Daily rainfall at day "mmdd"

d1019 Daily rainfall at day "mmdd"
d1020 Daily rainfall at day "mmdd"
d1021 Daily rainfall at day "mmdd"
d1022 Daily rainfall at day "mmdd"
d1023 Daily rainfall at day "mmdd"
d1024 Daily rainfall at day "mmdd"
d1025 Daily rainfall at day "mmdd"
d1026 Daily rainfall at day "mmdd"
d1027 Daily rainfall at day "mmdd"
d1028 Daily rainfall at day "mmdd"
d1029 Daily rainfall at day "mmdd"
d1030 Daily rainfall at day "mmdd"
d1031 Daily rainfall at day "mmdd"
d1101 Daily rainfall at day "mmdd"
d1102 Daily rainfall at day "mmdd"
d1103 Daily rainfall at day "mmdd"
d1104 Daily rainfall at day "mmdd"
d1105 Daily rainfall at day "mmdd"
d1106 Daily rainfall at day "mmdd"
d1107 Daily rainfall at day "mmdd"
d1108 Daily rainfall at day "mmdd"
d1109 Daily rainfall at day "mmdd"
d1110 Daily rainfall at day "mmdd"
d1111 Daily rainfall at day "mmdd"
d1112 Daily rainfall at day "mmdd"
d1113 Daily rainfall at day "mmdd"
d1114 Daily rainfall at day "mmdd"
d1115 Daily rainfall at day "mmdd"
d1116 Daily rainfall at day "mmdd"
d1117 Daily rainfall at day "mmdd"
d1118 Daily rainfall at day "mmdd"
d1119 Daily rainfall at day "mmdd"
d1120 Daily rainfall at day "mmdd"
d1121 Daily rainfall at day "mmdd"
d1122 Daily rainfall at day "mmdd"
d1123 Daily rainfall at day "mmdd"
d1124 Daily rainfall at day "mmdd"
d1125 Daily rainfall at day "mmdd"
d1126 Daily rainfall at day "mmdd"
d1127 Daily rainfall at day "mmdd"

d1128 Daily rainfall at day "mmdd"
d1129 Daily rainfall at day "mmdd"
d1130 Daily rainfall at day "mmdd"
d1201 Daily rainfall at day "mmdd"
d1202 Daily rainfall at day "mmdd"
d1203 Daily rainfall at day "mmdd"
d1204 Daily rainfall at day "mmdd"
d1205 Daily rainfall at day "mmdd"
d1206 Daily rainfall at day "mmdd"
d1207 Daily rainfall at day "mmdd"
d1208 Daily rainfall at day "mmdd"
d1209 Daily rainfall at day "mmdd"
d1210 Daily rainfall at day "mmdd"
d1211 Daily rainfall at day "mmdd"
d1212 Daily rainfall at day "mmdd"
d1213 Daily rainfall at day "mmdd"
d1214 Daily rainfall at day "mmdd"
d1215 Daily rainfall at day "mmdd"
d1216 Daily rainfall at day "mmdd"
d1217 Daily rainfall at day "mmdd"
d1218 Daily rainfall at day "mmdd"
d1219 Daily rainfall at day "mmdd"
d1220 Daily rainfall at day "mmdd"
d1221 Daily rainfall at day "mmdd"
d1222 Daily rainfall at day "mmdd"
d1223 Daily rainfall at day "mmdd"
d1224 Daily rainfall at day "mmdd"
d1225 Daily rainfall at day "mmdd"
d1226 Daily rainfall at day "mmdd"
d1227 Daily rainfall at day "mmdd"
d1228 Daily rainfall at day "mmdd"
d1229 Daily rainfall at day "mmdd"
d1230 Daily rainfall at day "mmdd"
d1231 Daily rainfall at day "mmdd"

See Also

PRborder

qinv *Computes (parts of) the inverse of a SPD sparse matrix*

Description

This routine use the GMRFLib implementation which compute parts of the inverse of a SPD sparse matrix. The diagonal and values for the neighbours in the inverse, are provided.

Usage

```
inla.qinv(Q, constr, reordering = inla.reorderings())
```

Arguments

Q	A SPD matrix, either as a (dense) matrix or sparseMatrix.
constr	Optional linear constraints; see `?INLA::f` and argument `extraconstr`
reordering	The type of reordering algorithm to be used; either one of the names listed in `inla.reorderings()` or the output from `inla.qreordering(Q)`. The default is "auto" which try several reordering algorithm and use the best one for this particular matrix.

Value

`inla.qinv` returns a `sparseMatrix` of type `dgTMatrix` with the diagonal and values for the neigbours in the inverse. Note that the full inverse is NOT provided!

Author(s)

Havard Rue <`hrue@r-inla.org`>

Examples

```
## dense matrix example
n = 10
A = matrix(runif(n^2), n, n)
Q = A %*% t(A)
print(mean(abs(inla.qinv(Q) - solve(Q))))

## sparse matrix example
rho = 0.9
Q = toeplitz(c(1+rho^2, -rho,  rep(0, n-3), -rho)) / (1-rho^2)
Q = inla.as.dgTMatrix(Q)
Q.inv = inla.qinv(Q)

## compute the marginal variances as a vector from a precision matrix
marginal.variances = diag(inla.qinv(Q))

## read the sparse matrix from a file in the 'i, j, value' format
filename = INLA:::inla.tempfile()
write(t(cbind(Q@i+1L,  Q@j+1L,  Q@x)), ncol=3, file=filename)
Qinv = inla.qinv(filename)
```

```
unlink(filename)
```

`qreordering` *Compute the reordering using the GMRFLib implementation*

Description

This function compute the reordering (or find the best reordering) using the GMRFLib implementation

Usage

```
inla.qreordering(graph, reordering)
```

Arguments

`graph`	A `(inla-)graph` object, a filename containing the graph or a matrix/Matrix defining it.
`reordering`	The type of reordering algorithm to be used; either one of the names listed in `inla.reorderings()`. The default is "auto" which try several reordering algorithm and use the best one for this particular matrix.

Value

`inla.qreordering` returns a list with the name of the reordering algorithm used or found, the reordering code for the reordering algorithm, the actual reordering and its inverse.

Author(s)

Havard Rue <`hrue@r-inla.org`>

Examples

```
g = system.file("demodata/germany.graph", package="INLA")
r = inla.qreordering(g)
m = inla.graph2matrix(g)
r = inla.qreordering(m)
m.file = INLA:::inla.write.fmesher.file(m)
r = inla.qreordering(m.file)
unlink(m.file)
```

---

qsample *Generate samples from a GMRF using the GMRFLib implementation*

Description

This function generate samples from a GMRF using the GMRFLib implementation

Usage

```
inla.qsample(
       n = 1L,
       Q,
       b,
       mu,
       sample,
       constr,
       reordering = inla.reorderings(),
       seed = 0L,
       logdens = ifelse(missing(sample), FALSE, TRUE),
       compute.mean = ifelse(missing(sample), FALSE, TRUE))
```

Arguments

`n`	Number of samples. Only used if `missing(sample)`
`Q`	The precision matrix or a filename containing it.
`b`	The linear term
`mu`	The mu term
`sample`	A matrix of optional samples where each column is a sample. If set, then evaluate the log-density for each sample only.
`constr`	Optional linear constraints; see `?INLA::f` and argument `extraconstr`
`reordering`	The type of reordering algorithm to be used; either one of the names listed in `inla.reorderings()` or the output from `inla.qreordering(Q)`. The default is "auto" which try several reordering algorithm and use the best one for this particular matrix.
`seed`	Control the RNG. If `seed=0L` then GMRFLib will set the seed intelligently/at 'random'. If `seed < 0L` then the saved state of the RNG will be reused if possible, otherwise, GMRFLib will set the seed intelligently/at 'random'. If `seed > 0L` then this value is used as the seed for the RNG.
`logdens`	If `TRUE`, compute also the log-density of each sample. Note that the output format then change.
`compute.mean`	If `TRUE`, compute also the (constrained) mean. Note that the output format then change.

Value

The log-density has form -1/2(x-mu)^T Q (x-mu) + b^T x

If `logdens` is `FALSE`, then `inla.qsample` returns the samples in a matrix, where each column is a sample. If `logdens` or `compute.mean` is `TRUE`, then a list with names `sample`, `logdens` and `mean` is returned. The samples are stored in the matrix `sample` where each column is a sample, and the log densities of each sample are stored as the vector `logdens`. The mean (include corrections for the constraints, if any) is store in the vector `mean`.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
g = system.file("demodata/germany.graph", package="INLA")
Q = inla.graph2matrix(g)
diag(Q) = dim(Q)[1L]
x = inla.qsample(10, Q)
## Not run: matplot(x)
x = inla.qsample(10, Q, logdens=TRUE)
## Not run: matplot(x$sample)

n = 3
Q = diag(n)
ns = 2

## sample and evaluate a sample
x = inla.qsample(n, Q=Q, logdens=TRUE)
xx = inla.qsample(Q=Q,  sample = x$sample)
print(x$logdens - xx$logdens)

## the use of a constraint
constr = list(A = matrix(rep(1, n), 1, n), e = 0)
x = inla.qsample(n, Q=Q, constr=constr)
print(constr$A %*% x)

## control the RNG
x = inla.qsample(n, Q=Q, seed = 123)
## restart from same seed,  only sample 1
xx = inla.qsample(n=1, Q=Q, seed = 123)
## continue from the save state, sample the remaining 2
xxx = inla.qsample(n=n-1, Q=Q, seed = -1)
## should be 0
print(x - cbind(xx, xxx))
```

---

qsolve *Solves linear SPD systems*

Description

This routine use the GMRFLib implementation to solve linear systems with a SPD matrix.

Usage

```
inla.qsolve(Q, B, reordering, method = c("solve", "forward", "backward"))
```

Arguments

- `Q` — A SPD matrix, either as a (dense) matrix, sparse-matrix or a filename containing the matrix (in the fmesher-format).
- `B` — The right hand side matrix, either as a (dense) matrix, sparse-matrix or a filename containing the matrix (in the fmesher-format). (Must be a matrix or sparse-matrix even if `ncol(B)` is 1.)
- `reordering` — The type of reordering algorithm to be used; either one of the names listed in `inla.reorderings()` or the output from `inla.qreordering(Q)`. The default is "auto" which try several reordering algorithm and use the best one for this particular matrix.
- `method` — The system to solve, one of `"solve"`, `"forward"` or `"backward"`. Let `Q = L L^T`, where `L` is lower triangular (the Cholesky triangle), then `method="solve"` solves `L L^T X = B` or equivalently `Q X = B`, `method="forward"` solves `L X = B`, and `method="backward"` solves `L^T X  = B`.

Value

`inla.qsolve` returns a matrix `X`, which is the solution of `Q X = B`, `L X = B` or `L^T X = B` depending on the value of `method`.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
n = 10
QQ = matrix(runif(n^2), n, n)
Q = inla.as.dgTMatrix(QQ %*% t(QQ))
B = matrix(runif(n^2-n), n, n-1)

X = inla.qsolve(Q, B, method = "solve")
print(paste("err", sum(abs( Q %*% X - B))))

L = t(chol(Q))
X = inla.qsolve(Q, B, method = "forward")
print(paste("err", sum(abs( L %*% X - B))))

X = inla.qsolve(Q, B, method = "backward")
print(paste("err", sum(abs( t(L) %*% X - B))))

Q.file = INLA:::inla.write.fmesher.file(Q)
B.file = INLA:::inla.write.fmesher.file(B)
X = inla.qsolve(Q.file, B.file, method = "backward")
print(paste("err", sum(abs( t(L) %*% X - B))))
unlink(Q.file)
unlink(B.file)
```

`read.graph` *Read and write a graph-object*

Description

Reads a graph-object to a file and write graph-object to file

Usage

```
inla.read.graph(..., size.only = FALSE)
inla.write.graph(graph, filename = "graph.dat", mode = c("binary", "ascii"), ...)

## S3 method for class 'inla.graph'
summary(object, ...)
## S3 method for class 'inla.graph'
plot(x, y, ...)
## S3 method for class 'inla.graph.summary'
print(x, ...)
```

Arguments

`filename`	The filename of the graph.
`graph`	An `inla.graph`-object, a (sparse) symmetric matrix, a filename containing the graph, a list or collection of characters and/or numbers defining the graph, or a neighbours list with class `nb` (see `spdep::card` and `spdep::poly2nb` for for details of `nb` and an example a function returning an `nb` object
`mode`	The mode of the file; ascii-file or a (gzip-compressed) binary. Default value depends on the inla.option `internal.binary.mode` which is default `TRUE`; see `inla.setOption`.
`object`	An `inla.graph` -object
`x`	An `inla.graph` -object
`y`	Not used
`size.only`	Only read the size of the graph
`...`	Additional arguments. In `inla.read.graph`, then it is the graph definition (object, character, filename), plus extra arguments. In `inla.write.graph` it is extra arguments to `inla.read.graph`.

Value

The output of `inla.read.graph`, is an `inla.graph` object, with elements

`n`	is the size of the graph
`nnbs`	is a vector with the number of neigbours
`nbs`	is a list-list with the neigbours
`cc`	list with connected component information (this entry can be auto-generated; see below) • `id`is a vector with the connected component id for each node (starting from 1)

- nis the number of connected components
- nodesis a list-list of nodes belonging to each connected component

The connected component information, can be generated from the rest of the graph-structure, using `graph = inla.add.graph.cc(graph)` if you manually construct the `inla.graph`-object. Methods implemented for `inla.graph` are `summary` and `plot`. The method `plot` require the libraries `Rgraphviz` and `graph` from the Bioconductor-project, see https://www.bioconductor.org.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

`inla.spy`

Examples

```
## a graph on a file
cat("3 1 1 2 2 1 1 3 0\n", file="g.dat")
g = inla.read.graph("g.dat")
## writing an inla.graph-object to file
g.file = inla.write.graph(g, mode="binary")
## re-reading it from that file
gg = inla.read.graph(g.file)
summary(g)
plot(g)
inla.spy(g)
## when defining the graph directly in the call, we can use a mix of character and numbers
g = inla.read.graph(c(3, 1, "1 2 2 1 1 3", 0))
inla.spy(c(3, 1, "1 2 2 1 1 3 0"))
inla.spy(c(3, 1, "1 2 2 1 1 3 0"),  reordering=3:1)
inla.write.graph(c(3, 1, "1 2 2 1 1 3 0"))
```

`rgeneric.define` *rgeneric models*

Description

A framework for defining latent models in R

Usage

```
inla.rgeneric.define(model = NULL, debug = FALSE, ...)
inla.rgeneric.iid.model(
       cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
         theta = NULL)
inla.rgeneric.ar1.model(
       cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
         theta = NULL)
inla.rgeneric.wrapper(
       cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
         model, theta = NULL)
```

Arguments

`model`	The definition of the model; see `inla.rgeneric.ar1.model`
`debug`	Logical. Turn on/off debugging
`cmd`	An allowed request
`theta`	Values of theta
`...`	Named list of variables that defines the environment of `model`
`debug`	Logical. Enable debug output

Value

This allows a latent model to be defined in `R`. See `inla.rgeneric.ar1.model` and `inla.rgeneric.iid.model` and the documentation for worked out examples of how to define latent models in this way. This will be somewhat slow and is intended for special cases and protyping. The function `inla.rgeneric.wrapper` is for internal use only.

Author(s)

Havard Rue `<hrue@r-inla.org>`

`Salm` *Extra-Poisson variation in dose-response study*

Description

Breslow (1984) analyses some mutagenicity assay data (shown below) on salmonella in which three plates have been processed at each dose i of quinoline and the number of revertant colonies of TA98 Salmonella measured

Usage

```
data(Salm)
```

Format

A data frame with 18 observations on the following 3 variables.

`y` number of salmonella bacteria

`dose` dose of quinoline (mg per plate)

`rand` indicator

Source

WinBUGS/OpenBUGS manual Examples VOl.I

Examples

```
data(Salm)
```

scale.model *Scale an intrinsic GMRF model*

Description

This function scales an intrinsic GMRF model so the geometric mean of the marginal variances is one

Usage

```
inla.scale.model(Q, constr = NULL, eps = sqrt(.Machine$double.eps))
```

Arguments

Q	A SPD matrix, either as a (dense) matrix or `sparseMatrix`
constr	Linear constraints spanning the null-space of `Q`; see `?INLA::f` and argument `extraconstr`
eps	A small constant added to the diagonal of `Q` if `constr`

Value

`inla.scale.model` returns a `sparseMatrix` of type `dgTMatrix` scaled so the geometric mean of the marginal variances (of the possible non-singular part of `Q`) is one, for each connected component of the matrix.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
## Q is singular
data(Germany)
g = system.file("demodata/germany.graph", package="INLA")
Q = -inla.graph2matrix(g)
diag(Q) = 0
diag(Q) = -rowSums(Q)
n = dim(Q)[1]
Q.scaled = inla.scale.model(Q, constr = list(A = matrix(1, 1, n), e=0))
print(diag(INLA:::inla.ginv(Q.scaled)))

## Q is singular with 3 connected components
g = inla.read.graph("6 1 2 2 3 2 2 1 3 3 2 1 2 4 1 5 5 1 4 6 0")
print(paste("Number of connected components", g$cc$n))
Q = -inla.graph2matrix(g)
diag(Q) = 0
diag(Q) = -rowSums(Q)
n = dim(Q)[1]
Q.scaled = inla.scale.model(Q, constr = list(A = matrix(1, 1, n), e=0))
print(diag(INLA:::inla.ginv(Q.scaled)))

## Q is non-singular with 3 connected components. no constraints needed
```

```
diag(Q) = diag(Q) + 1
Q.scaled = inla.scale.model(Q)
print(diag(INLA:::inla.ginv(Q.scaled)))
```

Scotland *Conditional Autoregressive (CAR) model for disease mapping*

Description

The rate of lip cancer in 56 counties in Scotland is recorder. The data set includes the observed and expected cases (based on the population and its age and sex distribution in the country), a covariate measuring the percentage of the population engaged in agricolture, fishing or forestry and the "position" of each county expressed as a list of adjacent counties

Usage

```
data(Scotland)
```

Format

A data frame with 56 observations on the following 4 variables.

`Counts` The number of lip cancer registered

`E` The expected number of lip cancer

`X` The percentage of the population engaged in agricolture, fishing or forestry

`Region` The county

Source

OpenBUGS Example manual, GeoBUGS

References

Clayton and Kaldor (1987) and Breslow and Clayton (1993)

Examples

```
data(Scotland)
```

`Seeds` *Factorial design*

Description

Proportion of seeds that germinated on each of 21 plates arranged according to a 2 by 2 factorial layout by seed and type of root extract

Usage

```
data(Seeds)
```

Format

A data frame with 21 observations on the following 5 variables.

- `r` number of germinated seeds per plate
- `n` number of total seeds per plate
- `x1` seed type
- `x2` root extracted
- `plate` indicator for the plate

Source

WinBUGS/OpenBUGS Manual Example, Vol. I

Examples

```
data(Seeds)
```

`SPDEtoy` *toy simulated data set for the SPDE tutorial*

Description

Simulated data set on 200 location points. The simulation process is made at the introduction of the SPDE tutorial.

Usage

```
data(SPDEtoy)
```

Format

A data frame with 200 observations on the following 3 variables.

- `s1` First element of the coordinates
- `s2` Second element of the coordinates
- `y` data simulated at the locations

Source

SPDE tutorial

Examples

```
data(SPDEtoy)
```

`summary.inla` *Summary for a INLA fit*

Description

Takes a fitted `inla` or `surv.inla` object produced by `inla` or `surv.inla` and produces a summary from it.

Usage

```
## S3 method for class 'inla'
summary(object, ..., digits = 4L, include.lincomb = TRUE)
## S3 method for class 'summary.inla'
print(x, ...)
```

Arguments

`object`	a fitted `inla` object as produced by `inla`.
`x`	a `summary.inla` object produced by `summary.inla`
`digits`	Integer Number of digits
`include.lincomb`	Logcial Include the summary for the the linear combinations or not
`...`	other arguments.

Details

Posterior mean and standard deviation (together with quantiles or cdf) are printed for the fixed effects in the model.

For the random effects the function `summary()` prints the posterior mean and standard deviations for the hyperparameters

Value

`summary.inla` returns an object of call `summaryinla`, a list with components:

`call`	the component from `object`.
`fixed`	the component from `object`.
`random`	the component from `object`.
`neffp`	the component from `object`.
`linear.predictor`	the component from `object`.

`lincomb` the component from `object`.

`lincomb.derived`
the component from `object`.

`family` the component from `object`.

Author(s)

Sara Martino and Havard Rue

See Also

`inla`

`summary.inla.mesh` *Summarizing triangular mesh objects*

Description

Construct and print `inla.mesh` object summaries

Usage

```
## S3 method for class 'inla.mesh'
summary(object, verbose = FALSE, ...)

## S3 method for class 'summary.inla.mesh'
print(x, ...)
```

Arguments

`object`	an object of class `"inla.mesh"`, usually a result of a call to `inla.mesh.create` or `inla.mesh.2d`.
`x`	an object of class `"summary.inla.mesh"`, usually a result of a call to `summary.inla.mesh`.
`verbose`	If `TRUE`, produce a more detailed output.
`...`	further arguments passed to or from other methods.

Author(s)

Finn Lindgren <`finn.lindgren@gmail.com`>

Surg — *Surgical: Institutional ranking*

Description

This example considers mortality rates in 12 hospitals performing cardiac surgery in babies

Usage

```
data(Surg)
```

Format

A data frame with 12 observations on the following 3 variables.

- `n` Number of deaths
- `r` Total number of cases
- `hospital` a factor with levels `A` `B` `C` `D` `E` `F` `G` `H` `I` `J` `K` `L`

Source

WinBUGS/OpenBUGS Manual Examples Vol. I

Examples

```
data(Surg)
```

SurvSim — *Survival data*

Description

Simulated data set for Weibull survival model

Usage

```
data(SurvSim)
```

Format

A data frame with 100 observations on the following 3 variables.

- `y` a numeric vector of survival times
- `cens` a numeric vector of event indicator (0=censured 1=failure)
- `x` a numeric vector of covariate

`Tokyo` *Binomial time series*

Description

Recorded days of rain above 1 mm in Tokyo for 2 years, 1983:84

Usage

```
data(Tokyo)
```

Format

A data frame with 366 observations on the following 3 variables.

`y` number of days with rain

`n` total number of days

`time` day of the year

Source

http://www.math.ntnu.no/~hrue/GMRF-book/tokyo.rainfall.data.dat

References

Rue, H and Held, L. (2005) *Gaussian Markov Random Fields - Theory and Applications* Chapman and Hall

Examples

```
data(Tokyo)
```

`Zambia` *Semiparametric regression*

Description

Undernutrition of children in each region of Zambia is measured through a score computed on the basis of some anthropometric measures. The data set contains also other infomation about each child.

Usage

```
data(Zambia)
```

Format

A data frame with 4847 observations on the following 10 variables.

`hazstd` standardised Z score of stunting

`bmi` body mass index of the mother

`agc` age of the child in months

`district` district where the child lives

`rcw` mother employment status with categories "working" (1) and "not working" (-1)

`edu1` mother's educations status with categories "complete primary but incomplete secondary " (`edu1=1`), "complete secondary or higher" (`edu2=1`) and "no education or incomplete primary" (`edu1=edu2=-1`)

`edu2` see above

`tpr` locality of the domicile with categories "urban" (1) and "rural" (-1)

`sex` gender of the child with categories "male" (1) and "female" (-1)

`edu` DO NOT KNOW; check source

Source

BayesX Manual http://www.stat.uni-muenchen.de/~bayesx/bayesx.html

Examples

```
data(Zambia)
```

Index