Package 'INLA'

August 26, 2023

Type Package

Title Full Bayesian Analysis of Latent Gaussian Models using Integrated Nested Laplace Approximations

Description Full Bayesian analysis of latent Gaussian models using Integrated Nested Laplace Approximaxion. It is a front-end to the inla-program.

Additional_repositories http:://inlabru-org.r-universe.dev/

Depends R (>= 3.5),
Matrix (>= 1.3-0),
sp (>= 1.6-0)

Imports graphics,
grDevices,
fmesher (>= 0.0.9.9026),
lifecycle,
methods,
parallel,
rlang,
splines,
stats,
utils,
withr

Suggests Deriv,
Ecdat,
HKprocess,
MatrixModels,
Rgraphviz,
deldir,
devtools,
doParallel,
dplyr,
evd,
fields,
ggplot2,
gsl,
graph,
gridExtra,
knitr,
markdown,
MASS,

matrixStats,
mlogit,
mvtnorm,
numDeriv,
pixmap,
rgl,
rmarkdown,
sf,
shiny,
sn,
spdep,
splancs,
terra,
tidyterra,
testthat,
tools,
INLAspacetime

VignetteBuilder knitr

BuildVignettes true

LazyData true

License GPL (>= 2) + file LICENSE

RoxygenNote 7.2.3

Encoding UTF-8

StagedInstall no

Collate '00000.R'
'INLA-package.R'
'agaussian.R'
'ar.R'
'barrier.R'
'binary.R'
'binary.install.R'
'cgeneric.R'
'changelog.R'
'collect.results.R'
'compare.results.R'
'coxph.R'
'cpo.R'
'create.data.file.R'
'cut.R'
'debug.graph.R'
'dev.new.R'
'display.matrix.R'
'doc.R'
'dryrun.R'
'expand.dataframe.R'
'export-class.R'
'external-package.R'
'f.R'
'fgn.R'
'finn.R'

'fmesher-io.R'
'fmesher-transition.R'
'fmesher.R'
'graph.convert.R'
'graph.matrix.R'
'group.R'
'group.cv.R'
'hyper.R'
'hyperpar.R'
'idx.R'
'iidkd.R'
'inla.R'
'inla.call.R'
'inlaEnv.R'
'interpret.formula.R'
'jmarginal.R'
'jp.R'
'knmodels.R'
'knmodels.sample.R'
'lattice2node.R'
'likelihood.R'
'lincomb.R'
'link-functions.R'
'list-models.R'
'load.R'
'marginal.R'
'mdata.R'
'merge.R'
'mesh.R'
'mesh.components.R'
'meshassessment.R'
'meshbuilder.R'
'model-wrapper.R'
'models-generate.R'
'models.R'
'models_documentation.R'
'nmix.lambda.fitted.R'
'obsolete.R'
'options.R'
'os.R'
'over_sp_mesh.R'
'pardiso.R'
'pc-alphaw.R'
'pc-ar.R'
'pc-bym.R'
'pc-cor0.R'
'pc-cor1.R'
'pc-cormat.R'
'pc-gamma.R'
'pc-gammacount.R'
'pc-gev.R'
'pc-multvar.R'

'pc-prec.R'
'pc-sn.R'
'pc-t.R'
'plot.R'
'posterior.sample.R'
'print.R'
'priors.used.R'
'prune.R'
'q.R'
'qinv.R'
'qreordering.R'
'qsample.R'
'qsolve.R'
'quantile-regression.R'
'read.graph.R'
'remote.R'
'reorderings.R'
'rerun.R'
'residuals.R'
'rgeneric.R'
'sampler.R'
'sandbox.R'
'scale.model.R'
'sections.R'
'set.default.arguments.R'
'sm.R'
'spde.common.R'
'spde1.R'
'spde2.R'
'spde3.R'
'spmesh.R'
'startup.R'
'summary.R'
'summary.scopy.R'
'surv.R'
'upgrade.R'
'utils.R'
'version.R'

Roxygen list(markdown = TRUE)

Version 23.08.26

Date Sat Aug 26 05:07:23 PM +03 2023 (Version_23.08.26)

R topics documented:

INLA-package 8
as.inla.mesh.segment 8
BivMetaAnalysis 9
Cancer 10
cgeneric 10
control.bgev 11
control.compute 12

control.expert . . . 13
control.family . . . 14
control.fixed . . . 16
control.gcpo . . . 17
control.group . . . 19
control.hazard . . . 20
control.inla . . . 21
control.lincomb . . . 25
control.link . . . 26
control.lp.scale . . . 27
control.mix . . . 27
control.mode . . . 28
control.pardiso . . . 29
control.pom . . . 30
control.predictor . . . 31
control.scopy . . . 32
control.update . . . 33
control.vb . . . 34
crs_wkt . . . 35
cut . . . 37
debug.graph . . . 38
Drivers . . . 39
dryrun . . . 39
Epil . . . 40
extract.groups . . . 40
f . . . 41
fgn . . . 45
Germany . . . 46
graph.convert . . . 47
graph.matrix . . . 47
idx . . . 49
inla . . . 50
inla-class . . . 55
inla.agaussian . . . 55
inla.ar.pacf2phi . . . 56
inla.as.sparse . . . 57
inla.as.wkt_tree.wkt . . . 57
inla.barrier . . . 58
inla.barrier.pcmatern . . . 59
inla.binary.install . . . 60
inla.changelog . . . 61
inla.collect.results . . . 62
inla.coxph . . . 63
inla.cpo . . . 64
inla.CRS . . . 65
inla.CRSargs . . . 66
inla.dev.new . . . 68
inla.diameter . . . 68
inla.doc . . . 69
inla.external.lib . . . 70
inla.extract.el . . . 70
inla.fmesher.smorg . . . 71

inla.generate.colors . . . 72
inla.get.inlaEnv . . . 73
inla.group . . . 73
inla.group.cv . . . 74
inla.has_PROJ6 . . . 75
inla.hyperpar . . . 76
inla.hyperpar.sample . . . 78
inla.identical.CRS . . . 79
inla.iidkd.sample . . . 79
inla.knmodels . . . 80
inla.knmodels.sample . . . 82
inla.ks.plot . . . 83
inla.likelihood . . . 84
inla.list.models . . . 85
inla.matern.cov . . . 86
inla.mdata . . . 87
inla.mesh.1d . . . 88
inla.mesh.1d.bary . . . 89
inla.mesh.2d . . . 89
inla.mesh.assessment . . . 91
inla.mesh.basis . . . 92
inla.mesh.boundary . . . 93
inla.mesh.components . . . 94
inla.mesh.components . . . 95
inla.mesh.create . . . 96
inla.mesh.deriv . . . 98
inla.mesh.fem . . . 99
inla.mesh.lattice . . . 99
inla.mesh.map.lim . . . 101
inla.mesh.project . . . 102
inla.mesh.query . . . 104
inla.mesh.segment . . . 105
inla.models . . . 107
inla.nmix.lambda.fitted . . . 338
inla.nonconvex.hull . . . 340
inla.option . . . 342
inla.over_sp_mesh . . . 344
inla.priors.used . . . 345
inla.prune . . . 346
inla.qstat . . . 346
inla.reorderings . . . 348
inla.rerun . . . 348
inla.row.kron . . . 349
inla.sample . . . 350
inla.simplify.curve . . . 354
inla.spde.make.A . . . 355
inla.spde.make.block.A . . . 356
inla.spde.make.index . . . 357
inla.spde.models . . . 358
inla.spde.precision . . . 360
inla.spde.result . . . 361
inla.spde.sample . . . 363

inla.spde1.create . . . 364
inla.spde2.generic . . . 366
inla.spde2.matern . . . 367
inla.spde2.matern.sd.basis . . . 370
inla.spde2.pcmatern . . . 371
inla.spTransform . . . 375
inla.sp_get_crs . . . 376
inla.ssh.copy.id . . . 377
inla.stack.remove.unused . . . 377
inla.surv . . . 382
inla.update . . . 384
inla.version . . . 384
joint.marginal . . . 385
jp . . . 387
Kidney . . . 388
lattice2node . . . 388
Leuk . . . 390
lines.inla.mesh.segment . . . 391
link . . . 392
make.lincomb . . . 393
marginal . . . 394
merge.inla . . . 397
meshbuilder . . . 398
Munich . . . 399
nwEngland . . . 400
Oral . . . 400
param2.matern.orig . . . 401
pardiso . . . 402
pc.alphaw . . . 402
pc.ar . . . 403
pc.cor0 . . . 404
pc.cor1 . . . 405
pc.cormat . . . 406
pc.ddof . . . 407
pc.gamma . . . 408
pc.gammacount . . . 409
pc.gevtail . . . 410
pc.multvar . . . 411
pc.prec . . . 413
pc.sn . . . 414
plot.inla . . . 415
plot.inla.CRS . . . 417
plot.inla.mesh . . . 418
plot.inla.trimesh . . . 420
PRborder . . . 421
print.inla . . . 421
PRprec . . . 422
qinv . . . 431
qreordering . . . 432
qsample . . . 433
qsolve . . . 435
read.graph . . . 436

rgeneric.define . . . 439
Salm . . . 440
scale.model . . . 441
Scotland . . . 442
Seeds . . . 442
SPDEtoy . . . 443
summary.inla . . . 444
summary.inla.mesh . . . 445
summary.scopy . . . 445
Surg . . . 446
SurvSim . . . 446
Tokyo . . . 447
Zambia . . . 447

Index **449**

`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

Version Currently, this package uses a YY.MM.DD versioning system, and is in heavy development. See https://github.com/hrue/r-inla/ and https://www.r-inla.org

License GPL2

See the web-site https://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` *objects to* `inla.mesh.segment` *objects.*

Description

[Superseded] by `fmesher::fm_as_segm()`

Usage

```
as.inla.mesh.segment(sp, ...)

inla.sp2segment(sp, ...)
```

Arguments

sp	An sp polygon object of class Polygon, Polygons, SpatialPolygons, or SpatialPolygonsDataFra
...	Additional arguments passed on to fmesher::fm_as_segm().

Value

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

Functions

- inla.sp2segment(): **[Superseded]** by fmesher::fm_as_segm()

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

```
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. ~~

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

---

cgeneric *cgeneric models*

Description

A framework for defining latent models in C

Usage

```
inla.cgeneric.define(model = NULL, shlib = NULL, n = 0L, debug = FALSE, ...)

inla.cgeneric.q(cmodel = NULL)
```

Arguments

model	The name of the model function
shlib	Name of the compiled object-file with model
n	The size of the model
debug	Logical. Turn on/off debugging
...	Additional arguments, required by inla.cgeneric.define() to be named arguments
cmodel	The name of a cgeneric model-object (output from inla.cgeneric.define

Author(s)

Havard Rue <hrue@r-inla.org>

`control.bgev` *control.bgev*

Description

Control variables in `control.*` for use with `inla()`. The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.bgev(
  q.location = 0.5,
  q.spread = 0.25,
  q.mix = c(0.1, 0.2),
  beta.ab = 5L
)

inla.set.control.bgev.default(...)
```

Arguments

`q.location`	The quantile level for the location parameter
`q.spread`	The quantile level for the spread parameter (must be < 0.5)
`q.mix`	The lower and upper quantile level for the mixing function
`beta.ab`	The parameters a and b in the Beta mixing function
`...`	Named arguments passed on to the main function

Details

The `control.bgev`-list is set within the corresponding `control.family`-list as control parameters to the `family="bgev"`

See Also

Other control: `control.compute()`, `control.expert()`, `control.family()`, `control.fixed()`, `control.gcpo()`, `control.group()`, `control.hazard()`, `control.inla()`, `control.lincomb()`, `control.link()`, `control.lp.scale()`, `control.mix()`, `control.mode()`, `control.pardiso()`, `control.pom()`, `control.predictor()`, `control.scopy()`, `control.update()`, `control.vb()`

control.compute *control.compute*

Description

Control variables in `control.*` for use with `inla()`. The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.compute(
  openmp.strategy = "default",
  hyperpar = TRUE,
  return.marginals = TRUE,
  return.marginals.predictor = FALSE,
  dic = FALSE,
  mlik = TRUE,
  cpo = FALSE,
  po = FALSE,
  waic = FALSE,
  residuals = FALSE,
  q = FALSE,
  config = FALSE,
  likelihood.info = FALSE,
  smtp = NULL,
  graph = FALSE,
  internal.opt = TRUE,
  save.memory = NULL,
  control.gcpo = INLA::control.gcpo()
)

inla.set.control.compute.default(...)
```

Arguments

`openmp.strategy`
: The computational strategy to use: 'small', 'medium', 'large', 'huge', 'default' and 'pardiso'.

`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

`return.marginals.predictor`
: A boolean variable if the marginals for the linear predictor should be returned (although it is computed). Default FALSE

`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 `TRUE`.

cpo	A boolean variable if the cross-validated predictive measures (cpo, pit) should be computed (default FALSE)
po	A boolean variable if the predictive ordinate should be computed (default FALSE)
waic	A boolean variable if the Watanabe-Akaike information criteria should be computed (default FALSE)
residuals	Provide estimates of residuals (whatever we mean by that). (default FALSE) Currently only residuals base on expected (saturated) deviance are available. The sign of the residuals are only very likely correct. These residuals are not properly justified from a Bayesian point of view, hence must be used with caution. It is provided in the hope they would be useful. This feature is EXPERIMENTAL for the moment, so changes can happen at any time.
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)
likelihood.info	A boolean variable to store likelihood-information or not. This option requires config=TRUE (Default FALSE. EXPERIMENTAL)
smtp	The sparse-matrix solver, one of 'default', 'taucs', 'band' or 'pardiso' (default inla.getOption("smtp")). smtp='pardiso' implies openmp.strategy='pardiso'.
graph	A boolean variable if the graph itself should be returned. (Default FALSE.)
internal.opt	A boolean variable, if to do internal online optimisations or not. (Default TRUE.)
save.memory	A boolean variable, make choices which saves memory over accuracy. (Default 'inla.getOption("save.memory")')
control.gcpo	(For experts only!) Set control variables for the gcpo. The intended use is to use inla.group.cv. Refer to control.gcpo, ?inla.group.cv and the vignette for details.
...	Named arguments passed on to the main function

See Also

Other control: control.bgev(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.lp.scale(), control.mix(), control.mode(), control.pardiso(), control.pom(), control.predictor(), control.scopy(), control.update(), control.vb()

---

control.expert *control.expert*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.expert(
  cpo.manual = FALSE,
  cpo.idx = -1,
  disable.gaussian.check = FALSE,
  jp = NULL,
  dot.product.gain = FALSE,
  globalconstr = list(A = NULL, e = NULL)
)

inla.set.control.expert.default(...)
```

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 (default FALSE)
jp	An object of class inla.jp defining a joint prior
dot.product.gain	Output the gain in optimizing dot-products? (Default FALSE)
globalconstr	Add a global constraint (see ?f and argument extraconstr). Note that a global constraint does NOT correct the normalisation constant.
...	Named arguments passed on to the main function

See Also

Other control: control.bgev(), control.compute(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.lp.scale(), control.mix(), control.mode(), control.pardiso(), control.pom(), control.predictor(), control.scopy(), control.update(), control.vb()

---

control.family *control.family*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.family(
  dummy = 0,
  hyper = NULL,
  initial = NULL,
  prior = NULL,
  param = NULL,
  fixed = NULL,
  link = "default",
  sn.shape.max = 5,
  gev.scale.xi = 0.1,
  control.bgev = NULL,
  cenpoisson.I = c(-1L, -1L),
  beta.censor.value = 0,
  variant = 0L,
  control.mix = NULL,
  control.pom = NULL,
  control.link = NULL,
  link.simple = "default"
)

inla.set.control.family.default(...)
```

Arguments

dummy	A dummy argument that can be used as a workaround
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 (default 5.0)
gev.scale.xi	(Expert option, do not use unless you know what you are doing.) The internal scaling of the shape-parameter for the GEV distribution. (default 0.1)
control.bgev	See `?control.bgev`
cenpoisson.I	The censoring interval for the censored Poisson
beta.censor.value	The censor value for the Beta-likelihood (`0 <= beta.censor.value < 1/2`)
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.pom	See `?control.pom`

control.link	See ?control.link
link.simple	See inla.doc("0inflated")
...	Named arguments passed on to the main function

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.lp.scale(), control.mix(), control.mode(), control.pardiso(), control.pom(), control.predictor(), control.scopy(), control.update(), control.vb()

---

control.fixed *control.fixed*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.fixed(
  cdf = NULL,
  quantiles = NULL,
  expand.factor.strategy = "model.matrix",
  mean = 0,
  mean.intercept = 0,
  prec = 0.001,
  prec.intercept = 0,
  compute = TRUE,
  correlation.matrix = FALSE,
  remove.names = NULL
)

inla.set.control.fixed.default(...)
```

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-specific 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. (default 0.0)
mean.intercept	Prior mean for the intercept (default 0.0)
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. (default 0.001)
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).
remove.names	A vector of names of expanded fixed effects to remove from the model-matrix. This is an expert option, and should only be used if you know what you are doing.
...	Named arguments passed on to the main function

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.lp.scale(), control.mix(), control.mode(), control.pardiso(), control.pom(), control.predictor(), control.scopy(), control.update(), control.vb()

---

control.gcpo *control.gcpo*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.gcpo(
  enable = FALSE,
  num.level.sets = -1,
  size.max = 32,
  strategy = c("posterior", "prior"),
  groups = NULL,
  selection = NULL,
  friends = NULL,
```

```
  verbose = FALSE,
  epsilon = 0.005,
  prior.diagonal = 1e-04,
  correct.hyperpar = TRUE,
  keep = NULL,
  remove = NULL,
  remove.fixed = TRUE
)

inla.set.control.gcpo.default(...)
```

Arguments

enable	TODO
num.level.sets	TODO
size.max	TODO
strategy	TODO
groups	TODO
selection	TODO
friends	TODO
verbose	TODO
epsilon	TODO
prior.diagonal	TODO
correct.hyperpar	TODO
keep	TODO
remove	TODO
remove.fixed	TODO
...	Named arguments passed on to the main function

Details

(For experts only!) Set control variables for the gcpo in control.compute. The intended use is to use `inla.group.cv`. Refer to `?inla.group.cv` and the vignette for details.

See Also

Other control: `control.bgev()`, `control.compute()`, `control.expert()`, `control.family()`, `control.fixed()`, `control.group()`, `control.hazard()`, `control.inla()`, `control.lincomb()`, `control.link()`, `control.lp.scale()`, `control.mix()`, `control.mode()`, `control.pardiso()`, `control.pom()`, `control.predictor()`, `control.scopy()`, `control.update()`, `control.vb()`

control.group *control.group*

Description

Control variables in `control.*` for use with `inla()`. The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.group(
  model = "exchangeable",
  order = NULL,
  cyclic = FALSE,
  graph = NULL,
  scale.model = TRUE,
  adjust.for.con.comp = TRUE,
  hyper = NULL,
  initial = NULL,
  fixed = NULL,
  prior = NULL,
  param = NULL
)

inla.set.control.group.default(...)
```

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 specification (Only applies to model 'besag')
`scale.model`	Scale the intrinsic model (RW1, RW2, BESAG) so the generalized variance is 1. (Default `TRUE`)
`adjust.for.con.comp`	Adjust for connected components when `scale.model=TRUE`? (default `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
`...`	Named arguments passed on to the main function

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.lp.scale(), control.mix(), control.mode(), control.pardiso(), control.pom(), control.predictor(), control.scopy(), control.update(), control.vb()

`control.hazard` *control.hazard*

Description

Control variables in `control.*` for use with `inla()`. The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.hazard(
  model = "rw1",
  hyper = NULL,
  fixed = FALSE,
  initial = NULL,
  prior = NULL,
  param = NULL,
  constr = TRUE,
  diagonal = NULL,
  n.intervals = 15,
  cutpoints = NULL,
  strata.name = NULL,
  scale.model = NULL
)

inla.set.control.hazard.default(...)
```

Arguments

`model`	The model for the baseline hazard model. One of 'rw1', 'rw2' or 'iid'. (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")`.)
...	Named arguments passed on to the main function

See Also

Other control: `control.bgev()`, `control.compute()`, `control.expert()`, `control.family()`, `control.fixed()`, `control.gcpo()`, `control.group()`, `control.inla()`, `control.lincomb()`, `control.link()`, `control.lp.scale()`, `control.mix()`, `control.mode()`, `control.pardiso()`, `control.pom()`, `control.predictor()`, `control.scopy()`, `control.update()`, `control.vb()`

`control.inla` *control.inla*

Description

Control variables in `control.*` for use with `inla()`. The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.inla(
  strategy = "auto",
  int.strategy = "auto",
  int.design = NULL,
  interpolator = "auto",
  fast = TRUE,
  linear.correction = NULL,
  h = 0.005,
  dz = 0.75,
  diff.logdens = 6,
  print.joint.hyper = TRUE,
  force.diagonal = FALSE,
  skip.configurations = TRUE,
  mode.known = FALSE,
  adjust.weights = TRUE,
  tolerance = 0.005,
  tolerance.f = NULL,
  tolerance.g = NULL,
  tolerance.x = NULL,
  tolerance.step = 0.001,
  restart = 0L,
  optimiser = "default",
  verbose = NULL,
  reordering = "auto",
  cpo.diff = NULL,
```

```
  npoints = 9,
  cutoff = 1e-04,
  adapt.hessian.mode = NULL,
  adapt.hessian.max.trials = NULL,
  adapt.hessian.scale = NULL,
  adaptive.max = 25L,
  huge = FALSE,
  step.len = 0,
  stencil = 5L,
  lincomb.derived.correlation.matrix = FALSE,
  diagonal = 0,
  numint.maxfeval = 1e+05,
  numint.relerr = 1e-05,
  numint.abserr = 1e-06,
  cmin = -Inf,
  b.strategy = "skip",
  step.factor = -0.1,
  global.node.factor = 2,
  global.node.degree = .Machine$integer.max,
  stupid.search = TRUE,
  stupid.search.max.iter = 1000L,
  stupid.search.factor = 1.05,
  control.vb = INLA::control.vb(),
  num.gradient = "central",
  num.hessian = "central",
  optimise.strategy = "smart",
  use.directions = TRUE,
  constr.marginal.diagonal = sqrt(.Machine$double.eps),
  improved.simplified.laplace = FALSE,
  parallel.linesearch = FALSE,
  compute.initial.values = TRUE,
  hessian.correct.skewness.only = FALSE
)

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

Arguments

strategy	Character The strategy to use for the approximations; one of 'auto' (default), 'gaussian', 'simplified.laplace', 'laplace' or 'adaptive'.
int.strategy	Character The integration strategy to use; one of 'auto' (default), 'ccd', 'grid', 'eb' (empirical bayes), 'user' or 'user.std'. For the experimental mode, then 'grid' equal 'ccd' for more than two hyperparameters.
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 0.005.

`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 '2.5*tolerance.f', 'tolerance.g' and '5*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.

`tolerance.step` Numerical The tolerance for the change in root-mean_squre in the inner Newton-like optimisation of the latent field.

`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' 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`. Default is 'auto'.)

`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 (default 9)

`cutoff` Numerical The cutoff used in the 'stratey=laplace' approximation. (Smaller value is more accurate and more slow.) (default 1e-4)

`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.

`adaptive.max` Selecting `strategy="adaptive"` will chose the default strategy for all fixed effects and model components with length less or equal to `adaptive.max`, for others, the gaussian strategy will be applied.

`huge` Logical If TRUE then try to do some of the internal parallelisations differently. Hopefully this will be of benefit 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 (0 means `default` which depends on `stencil`)

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

`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. (default 0.0)

`numint.maxfeval`
Numerical Maximum number of function evaluations in the the numerical integration for the hyperparameters. (Default 100000.)

`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. (Default -Inf)

`b.strategy` Character If `cmin` is used, either keep the linear term (with `b.strategy="keep"`) or skip the contribution by setting the linear term to zero (`b.strategy="skip"`). The default value is `"skip"`

`step.factor` Numerical The step factor in the Newton-Raphson algorithm saying how large step to take (Default 1.0) YES! setting this to a negative values means = 1, EXCEPT the first time (for each thread) where |step.factor| is used.

`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, default 2)

`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). (default `.Machine$integer.max`)

`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. (default 1000)

`stupid.search.factor`
Numerical Factor (>=1) to increase the step-length with after each new iteration. (default 1.05)

control.vb — list of arguments for various VB corrections. See control.vb() for details.

num.gradient — Character Set the numerical scheme to compute the gradient, one of "forward" or "central" (default).

num.hessian — Character Set the numerical scheme to compute the Hessian, one of "forward" or "central" (default).

optimise.strategy — Character THIS OPTION IS EXPERIMENTAL. Chose the optimiser strategy, one of "plain" or "smart" (default)

use.directions — THIS OPTION IS EXPERIMENTAL. Unless FALSE or NULL, use directions for computing gradient and Hessian, initialised with use.directions if a matrix.

constr.marginal.diagonal — Add stability to AQ^-1A^T by adding a small diagonal term. (default epsilon^0.5)

improved.simplified.laplace — If TRUE use an experimental improved variant, otherwise, use the standard one.

parallel.linesearch — Use serial (default) or parallel line-search (highly experimental for the moment)

compute.initial.values — Compute initial values for the latent field or not. (experimental-mode only)

hessian.correct.skewness.only — If TRUE then correct only skewness in the Hessian, for the hyperparameters. If FALSE (default), correct also variance (experimental-mode only)

... — Named arguments passed on to the main function

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.lincomb(), control.link(), control.lp.scale(), control.mix(), control.mode(), control.pardiso(), control.pom(), control.predictor(), control.scopy(), control.update(), control.vb()

---

control.lincomb *control.lincomb*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.lincomb(verbose = FALSE)

inla.set.control.lincomb.default(...)
```

Arguments

verbose — Use verbose mode for linear combinations if verbose model is set globally. (Default FALSE). This option is only available for the default inla.mode (inla.mode="compact").

... — Named arguments passed on to the main function

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.link(), control.lp.scale(), control.mix(), control.mode(), control.pardiso(), control.pom(), control.predictor(), control.scopy(), control.update(), control.vb()

---

control.link *control.link*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.link(
  model = "default",
  order = NULL,
  variant = NULL,
  hyper = NULL,
  quantile = NULL,
  a = 1,
  initial = NULL,
  fixed = NULL,
  prior = NULL,
  param = NULL
)

inla.set.control.link.default(...)
```

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.
hyper	Definition of the hyperparameter(s) for the link model chosen
quantile	The quantile for quantile link function
a	The parameter a in the LOGa link
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)
...	Named arguments passed on to the main function

Details

The control.link-list is set within the corresponding control.family-list as the link is likelihood-family specific.

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.lp.scale(), control.mix(), control.mode(), control.pardiso(), control.pom(), control.predictor(), control.scopy(), control.update(), control.vb()

---

control.lp.scale *control.lp.scale*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.lp.scale(hyper = NULL)

inla.set.control.lp.scale.default(...)
```

Arguments

hyper	Definition of the hyperparameter(s)
...	Named arguments passed on to the main function

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.mix(), control.mode(), control.pardiso(), control.pom(), control.predictor(), control.scopy(), control.update(), control.vb()

---

control.mix *control.mix*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.mix(
  model = NULL,
  hyper = NULL,
  initial = NULL,
  fixed = NULL,
  prior = NULL,
  param = NULL,
  npoints = 101,
  integrator = "default"
)

inla.set.control.mix.default(...)
```

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)
npoints	Number of points used to do the numerical integration (default 101)
integrator	The integration scheme to use (default, quadrature, simpson)
...	Named arguments passed on to the main function

Details

The control.mix list is set within the corresponding control.family-list a the mixture of the likelihood is likelihood specific. (This option is EXPERIMENTAL.)

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.lp.scale(), control.mode(), control.pardiso(), control.pom(), control.predictor(), control.scopy(), control.update(), control.vb()

---

control.mode *control.mode*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.mode(
  result = NULL,
  theta = NULL,
  x = NULL,
  restart = FALSE,
  fixed = FALSE
)

inla.set.control.mode.default(...)
```

Arguments

result	Previous 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/initial 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).
...	Named arguments passed on to the main function

Details

For internal use and for algorithms built on to of INLA.

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.lp.scale(), control.mix(), control.pardiso(), control.pom(), control.predictor(), control.scopy(), control.update(), control.vb()

---

control.pardiso *control.pardiso*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.pardiso(
  verbose = FALSE,
  debug = FALSE,
  parallel.reordering = TRUE,
  nrhs = -1
)

inla.set.control.pardiso.default(...)
```

Arguments

verbose	Show detailed output (default FALSE)
debug	Show internal debug output (default FALSE)
parallel.reordering	Do reordering in parallel (default TRUE)
nrhs	Number of right-hand sides to solve for in parallel (-1 will determine this adaptive)
...	Named arguments passed on to the main function

Details

Extra options controlling the PARDISO library

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.lp.scale(), control.mix(), control.mode(), control.pom(), control.predictor(), control.scopy(), control.update(), control.vb()

---

control.pom *control.pom*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.pom(cdf = "logit", fast = FALSE)

inla.set.control.pom.default(...)
```

Arguments

cdf	character The cdf to use, "logit" (default) or "probit"
fast	Logical Use a faster but approximate form for the probit cdf (default FALSE)?
...	Named arguments passed on to the main function

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.lp.scale(), control.mix(), control.mode(), control.pardiso(), control.predictor(), control.scopy(), control.update(), control.vb()

`control.predictor` *control.predictor*

Description

Control variables in `control.*` for use with `inla()`. The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.predictor(
  hyper = NULL,
  fixed = NULL,
  prior = NULL,
  param = NULL,
  initial = NULL,
  compute = FALSE,
  cdf = NULL,
  quantiles = NULL,
  cross = NULL,
  A = NULL,
  precision = exp(15),
  link = NULL
)

inla.set.control.predictor.default(...)
```

Arguments

`hyper`	Definition of the hyperparameters.
`fixed`	(OBSOLETE!) If the precision for the artificial noise is fixed or not (default 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, (default exp(15))
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.
...	Named arguments passed on to the main function

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.lp.scale(), control.mix(), control.mode(), control.pardiso(), control.pom(), control.scopy(), control.update(), control.vb()

`control.scopy` *control.scopy*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.scopy(
  covariate = NULL,
  n = 5,
  model = "rw2",
  mean = 1,
  prec.mean = 1,
  prec.betas = 10
)

inla.set.control.scopy.default(...)
```

Arguments

covariate	The covariate for the scopy function
n	Number of betas
model	scopy model (one of 'rw1' and 'rw2')
mean	The prior mean for mean(betas)
prec.mean	The prior precision for mean(betas)
prec.betas	The prior precision prec(betas-mean(betas))
...	Named arguments passed on to the main function

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.lp.scale(), control.mix(), control.mode(), control.pardiso(), control.pom(), control.predictor(), control.update(), control.vb()

---

control.update *control.update*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.update(result = NULL)

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

Arguments

result	Update the joint posterior for the hyperparameters from result
...	Named arguments passed on to the main function

See Also

inla()

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.lp.scale(), control.mix(), control.mode(), control.pardiso(), control.pom(), control.predictor(), control.scopy(), control.vb()

control.vb *control.fixed*

Description

Control variables in control.* for use with inla(). The functions can be used to TAB-complete arguments, and returns a list of the default control arguments, unless overridden by specific input arguments.

Usage

```
control.vb(
  enable = "auto",
  strategy = c("mean", "variance"),
  verbose = TRUE,
  iter.max = 25,
  emergency = 25,
  f.enable.limit = c(30, 25),
  hessian.update = 2,
  hessian.strategy = c("default", "full", "partial", "diagonal")
)

inla.set.control.vb.default(...)
```

Arguments

enable	Logical/Character Use this feature? If "auto" this will be selected automatically.
strategy	Character What to correct, either "mean" or "variance".
verbose	Logical Be verbose or not.
iter.max	Integer Maximum number of iterations.
emergency	Numeric If the standardized correction for the mean is larger than this value, then call the vb.correction off and issue a warning
f.enable.limit	Vector of length 2. The size limit to correct for a f(). First element is for strategy="mean". Second element is for strategy="mean".
hessian.update	How many times the Hessian is updated for each correction (strategy="variance" only).
hessian.strategy	Select strategy for computing the Hessian matrix for strategy="variance", one of "full", "diagonal", "partial" and "default".
...	Named arguments passed on to the main function

Details

control.vb List of arguments for various VB corrections. Used for control.inla control.vb specifications.

See Also

Other control: control.bgev(), control.compute(), control.expert(), control.family(), control.fixed(), control.gcpo(), control.group(), control.hazard(), control.inla(), control.lincomb(), control.link(), control.lp.scale(), control.mix(), control.mode(), control.pardiso(), control.pom(), control.predictor(), control.scopy(), control.update()

---

crs_wkt *Handling CRS/WKT*

Description

[Deprecated] in favour of fmesher::fm_wkt() and related methods.

Get and set CRS object or WKT string properties.

Usage

```
inla.wkt_is_geocent(wkt)

inla.crs_is_geocent(crs)

inla.wkt_get_ellipsoid_radius(wkt)

inla.crs_get_ellipsoid_radius(crs)

inla.wkt_set_ellipsoid_radius(wkt, radius)

inla.crs_set_ellipsoid_radius(crs, radius)

inla.wkt_unit_params()

inla.wkt_get_lengthunit(wkt)

inla.wkt_set_lengthunit(wkt, unit, params = NULL)

inla.crs_get_wkt(crs)

inla.crs_get_lengthunit(crs)

inla.crs_set_lengthunit(crs, unit, params = NULL)
```

Arguments

wkt	A WKT2 character string
crs	A sp::CRS or inla.CRS object
radius	numeric
unit	character, name of a unit. Supported names are "metre", "kilometre", and the aliases "meter", "m", International metre", "kilometer", and "km", as defined by inla.wkt_unit_params or the params argument. (For legacy PROJ4 use, only "m" and "km" are supported)
params	Length unit definitions, in the list format produced by inla.wkt_unit_params(), Default: NULL, which invokes inla.wkt_unit_params()

Value

For `inla.wkt_unit_params`, a list of named unit definitions

For `inla.wkt_get_lengthunit`, a list of length units used in the wkt string, excluding the ellipsoid radius unit.

For `inla.wkt_set_lengthunit`, a WKT2 string with altered length units. Note that the length unit for the ellipsoid radius is unchanged.

For `inla.crs_get_wkt`, WKT2 string.

For `inla.crs_get_lengthunit`, a list of length units used in the wkt string, excluding the ellipsoid radius unit. (For legacy PROJ4 code, the raw units from the proj4string are returned, if present.)

For `inla.crs_set_lengthunit`, a `sp::CRS` object with altered length units. Note that the length unit for the ellipsoid radius is unchanged.

For `inla.wkt_unit_params`, a list of named unit definitions

For `inla.wkt_get_lengthunit`, a list of length units used in the wkt string, excluding the ellipsoid radius unit.

For `inla.wkt_set_lengthunit`, a WKT2 string with altered length units. Note that the length unit for the ellipsoid radius is unchanged.

For `inla.crs_get_wkt`, WKT2 string.

For `inla.crs_get_lengthunit`, a list of length units used in the wkt string, excluding the ellipsoid radius unit. (For legacy PROJ4 code, the raw units from the proj4string are returned, if present.)

For `inla.crs_set_lengthunit`, a `sp::CRS` object with altered length units. Note that the length unit for the ellipsoid radius is unchanged.

Functions

- `inla.wkt_is_geocent()`: **[Deprecated]** in favour of `fmesher::fm_wkt_is_geocent()`
- `inla.crs_is_geocent()`: **[Deprecated]** in favour of `fmesher::fm_crs_is_geocent()`
- `inla.wkt_get_ellipsoid_radius()`: **[Deprecated]** in favour of `fmesher::fm_ellipsoid_radius()`
- `inla.crs_get_ellipsoid_radius()`: **[Deprecated]** in favour of `fmesher::fm_ellipsoid_radius()`
- `inla.wkt_set_ellipsoid_radius()`: **[Deprecated]** in favour of `fmesher::fm_wkt_set_ellipsoid_radius()`
- `inla.crs_set_ellipsoid_radius()`: **[Deprecated]** in favour of `fmesher::fm_ellipsoid_radius<-()`

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

`inla.sp_get_crs()`

`inla.sp_get_crs()`

Examples

```
## Not run:
c1 <- fmesher::fm_CRS("globe")
inla.crs_get_lengthunit(c1)
c2 <- inla.crs_set_lengthunit(c1, "km")
inla.crs_get_lengthunit(c2)
```

```
## End(Not run)

## Not run:
c1 <- inla.CRS("globe")
inla.crs_get_lengthunit(c1)
c2 <- inla.crs_set_lengthunit(c1, "km")
inla.crs_get_lengthunit(c2)

## End(Not run)
```

---

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, mc.cores = NULL, 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.
`mc.cores`	The number of cores to use in `parallel::mclapply`. If `is.null(mc.cores)`, then check `getOption("mc.cores")` and `inla.getOption("num.threads")` in that order.
`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.ferkingstad@gmail.com> 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. arXiv preprint arXiv:1708.03272, available at http://arxiv.org/abs/1708.03272. Published in Stat, 6:331-344 (2017).

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 http://www.r-inla.org/examples/case-studies/ferkingstad-2017 and 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

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)

Details

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.

References

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

Examples

```
data(Drivers)
```

dryrun — *Do a dryrun to extract some internal information upfront*

Description

Do a dryrun to get information about the internal storage and the list (and ordering) of the hyperparameters

Usage

```
inla.dryrun(...)
```

Arguments

...	Same arguments as inla()

Value

A list of start-index and length for each latent component and a list of the hyperparameters in the model

Author(s)

Havard Rue <hrue@r-inla.org>

Epil *Repeated measures on Poisson counts*

Description

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

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(segm, groups, groups.new = 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.list.models()) as

$$w\, f(x)$$

Usage

```
f(
  ...,
  model = "iid",
  copy = NULL,
  scopy = NULL,
  same.as = NULL,
  n = NULL,
  nrep = NULL,
  replicate = NULL,
  ngroup = NULL,
  group = NULL,
  control.group = inla.set.control.group.default(),
  control.scopy = inla.set.control.scopy.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(13),
  range = NULL,
  adjust.for.con.comp = TRUE,
  order = NULL,
  scale = NULL,
  rgeneric = NULL,
  cgeneric = NULL,
  scale.model = NULL,
 args.slm = list(rho.min = NULL, rho.max = NULL, X = NULL, W = NULL, Q.beta = NULL),
  args.ar1c = list(Z = NULL, Q.beta = NULL),
  args.intslope = list(subject = NULL, strata = NULL, covariates = NULL),
  vb.correct = TRUE,
  locations = 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 chosen model. The default is `iid`. See `names(inla.models()$latent)` for a list of possible alternatives and `inla.doc()` for detailed docs.
copy	The name of the model-component to copy
scopy	The name of the model-component to smooth-copy (where the copy-function is a spline)
same.as	Can be used with `copy="..."`. `same.as="A"` says that this `copy` should use the same scaling parameter as another `copy` "A"
n	An optional argument which defines the dimension of the model if this is different from `length(sort(unique(covariate)))`

nrep	Number of replications, if not given, then nrep=max(replications)
replicate	A vector of which replications to use.
ngroup	Number of groups, if not given, then ngroup=max(group)
group	A vector of which groups to use.
control.group	Controls the use of group
control.scopy	Controls the use of scopy
hyper	Specification 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 or use inla.doc() for detailed info on the family and supported prior distributions.
initial	THIS OPTION IS OBSOLETE, DO NOT USE
prior	THIS OPTION IS OBSOLETE, DO NOT USE
param	THIS OPTION IS OBSOLETE, DO NOT USE
fixed	THIS OPTION IS OBSOLETE; DO NOT USE
season.length	Length of the seasonal component for 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 to prevent numerical issues.
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, DO NOT USE
cdf	THIS OPTION IS OBSOLETE, DO NOT USE
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	The boundary conditions for model rw2d, 0 means use the correct subspace (default), while 1 means condition on 0's outside
spde.prefix	Internal use only
spde2.prefix	Internal use only
spde2.transform	Internal use only
spde3.prefix	Internal use only
spde3.transform	Internal use only
mean.linear	Prior mean for model="linear"
prec.linear	Prior precision for model="linear"
compute	A boolean variable indicating whether 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	Internal use only
precision	The precision for the artificial 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.
rgeneric	A object of class inla.rgeneric which defines the model. (EXPERIMENTAL!)
cgeneric	A object of class inla.cgeneric 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.
args.ar1c	Required arguments to the model="ar1c"; see the documentation for further details.
args.intslope	A list with the subject (factor), strata (factor) and covariates (numeric) for the intslope model; see the documentation for further details,

vb.correct	Add this model component to the list of nodes to be used for the (potential) vb correction? If TRUE do, and do not if FALSE. Can also be a vector of nodes to add in the correction-set.
locations	A matrix with locations for the model dmatern. This also defines n.
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 explicitly.

Value

TODO

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla(), hyperpar.inla()

---

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

Description

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

Usage

```
inla.fgn(H, K = 4L, lag.max = NULL, approx = TRUE)
```

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
lag.max	Integer. If positive integer, return the coeffcients implicitely as the ACF from 0 to lag.max
approx	Logical. If lag.max is an positive integer and approx is FALSE, then return the true ACF instead of the approximated one.

Value

`inla.fgn` returns a named matrix. If `is.null(lag.max)`, then first column is `H`, columns `1+1:K` are lag one correlations (or phi's), and columns `1+K+1:K` are the weights. If `lag.max > 0`, then return the ACFs in columns `2+(0:lag.max)`, for the H in column 1, either the approximated ones or the the true ones.

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))
```

`Germany` *Disease Mapping*

Description

Cases of Oral cavity cancer in Germany from 1986-1990

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)
```

graph.convert *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()

graph.matrix *Construct a neighbour-matrix from a* graph

Description

Construct a neighbour-matrix from a graph and disaply it

Usage

```
inla.matrix2graph(graph, ...)

inla.graph2matrix(graph, ...)

inla.spy(graph, ..., reordering = NULL, factor = 1, 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.
...	Additional arguments to inla.read.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.

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, filename = tempfile())
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 = NULL,
  family = "gaussian",
  contrasts = NULL,
  data = NULL,
  quantiles = c(0.025, 0.5, 0.975),
  E = NULL,
  offset = NULL,
  scale = NULL,
  weights = NULL,
  Ntrials = NULL,
  strata = NULL,
  lp.scale = NULL,
  link.covariates = NULL,
  verbose = inla.getOption("verbose"),
  lincomb = NULL,
  selection = NULL,
  control.compute = list(),
  control.predictor = list(),
  control.family = list(),
  control.inla = list(),
  control.fixed = list(),
  control.mode = list(),
  control.expert = list(),
  control.hazard = list(),
  control.lincomb = list(),
  control.update = list(),
  control.lp.scale = list(),
  control.pardiso = 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"),
  inla.mode = inla.getOption("inla.mode"),
  safe = inla.getOption("safe"),
  debug = inla.getOption("debug"),
  .parent.frame = environment(formula)
)
```

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 and use inla.doc() for detailed docs for individual families.
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, $p(0), p(1), \ldots$ to compute for each posterior marginal. The function returns, for each posterior marginal, the values $x(0), x(1), \ldots$ such that $$\text{Prob}(X < x(p)) = p$$
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). WARNING: The normalizing constant for the likelihood is NOT recomputed, so ALL marginals (and the marginal likelihood) must be interpreted with great care.
Ntrials	A vector containing the number of trials for the binomial likelihood and variantes, or the number of required successes for the nbinomial2 likelihood. Default value is rep(1, n.data).
strata	Fixed (optional) strata indicators for tstrata likelihood model and similar. The documentation for each likelihood will inform if this argument is required.

lp.scale
: A vector with same length as the predictor going into the likelihood with either NA's or indices indexing the scaling coefficients. NA or a index less or equal to 0 means no scaling. The priors and properties of the scaling coefficients are set in control.lp.scale

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 inla.getOption("verbose"))

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 vignette Short tutorials from old www-page for information on how to define such linear combinations.

selection
: This is a similar argument to the one in inla.posterior.sample and follow the same format. This argument allows to define a subset of the latent field for which to compute an approximated joint distribution. It will appear in result$selection. See also ?inla.rjmarginal and the approriate vignette.

control.compute
: See ?control.compute

control.predictor
: See ?control.predictor

control.family
: See ?control.family

control.inla
: See ?control.inla

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

control.lp.scale
: See ?control.lp.scale

control.pardiso
: See ?control.pardiso

only.hyperparam
: If TRUE, then only the hyperparameters are computed.

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. (This is an expert option and not intended for normal usage.)

num.threads
: Maximum number of threads the inla-program will use, or as 'A:B' defining the number threads in the outer (A) and inner (B) layer for nested parallelism. If B is set to -1, then one can force some single function evaluations to be perfored in parallel, so num.threads=4:-1 will locally behave like num.threads=4:1 (if considered to be more efficient). If B > 1 then num.threads=A:B and num.threads=A:-B are equivalent.

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 model-files are stored in the current directory called "inla.model" or "inla.model-NUMBER".

working.directory — A string giving the name of an non-existing directory where to store the model-files. Sometimes this argument is required if the temporary directory returned with tempdir() not writeable or has an encoding that is not supported.

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.

inla.mode — Run inla in compact-mode, or the classic-mode. Default is to use the mode set by inla.getOption("inla.mode") which is default compact-mode.

safe — If TRUE, then enable possible restarts to improve initial values and Hessian if needed.

debug — If TRUE, print 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 — 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.

`selection`
: Provide the approximated joint distribution for the `selection`

`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)`)

`residuals`
: If `residuals=TRUE` in `control.compute`, a list of standardized residuals are provided, see `?control.compute` for details

`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 hyperparameters.

`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

See Also

`f()`

inla-class *inla estimation object class*

Description

The inla class is defined in the INLA package

See Also

inla

inla.agaussian *Aggregate Gaussian into an equivalent observation*

Description

Aggregate Gaussians observed with the same mean and precision, into an equivalent triplet, for use with `family="agaussian"`

Usage

```
inla.agaussian(y, s = NULL)
```

Arguments

y	Repeated observations. If y is a matrix, then each row represents repeated observations. if y is a list, then each element of the list is a vector of repeated observations. If y is a vector, then the whole vector represents repeated observations. The optional scaling s, must have the same format as y, ie `matrix` or `vector`. NA's in y (and s) are removed and not used or counted. If s is given, then the NA-pattern in y and s must be the same.
s	Optional fixed scaling of the precisions. Must be in the same format as y, and have the same NA-pattern. See the documentation for details.

Value

The output is a `inla.mdata`-object ready for use with `family="agaussian"`. See the example in the documentation.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
A = matrix(1:25,5,5)
inla.agaussian(A)

A[1,-1] = NA
A[2,-(2:3)] = NA
inla.agaussian(A)
```

`inla.ar.pacf2phi` *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.phi2acf(phi, lag.max = length(phi))

inla.ar.pacf2acf(pac, lag.max = length(pac))
```

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.as.wkt_tree.wkt *Internal WKT handling*

Description

[Deprecated] in favour of fmesher::fm_wkt_as_wkt_tree(). Conversion between WKT and a tree representation

Usage

```
inla.as.wkt_tree.wkt(x, ...)

inla.as.wkt.wkt_tree(x, pretty = FALSE, ...)

inla.wkt_tree_get_item(x, item, duplicate = 1)

inla.wkt_tree_set_item(x, item_tree, duplicate = 1)
```

Arguments

x	A WKT2 string, or a wkt_tree list structure
...	Unused
pretty	logical
item	character vector with item labels identifying a parameter item entry.
duplicate	For items that have more than one match, duplicate indicates the index number of the desired version. Default: 1
item_tree	An item tree identifying a parameter item entry

---

inla.barrier *Functions for defining the Barrier models*

Description

Functions for defining Barrier models as an inla rgeneric model

Usage

```
inla.barrier.pcmatern(mesh, barrier.triangles, prior.range,
                      prior.sigma, range.fraction=0.2)
inla.barrier.polygon(mesh, barrier.triangles, Omega=NULL)
inla.barrier.q(fem, ranges, sigma=1)
inla.barrier.fem(mesh, barrier.triangles, Omega=NULL)
```

Arguments

mesh	The mesh to build the model on, from inla.mesh.2d
barrier.triangles	The numerical ids of the triangles that make up the barrier area
prior.range	2 parameters (range0,Prange) for the prior spatial range. If Prange is NA, then range0 is used as a fixed range value (not tested).
prior.sigma	2 parameters (sig0,Psig) for the prior marginal standard deviation sigma. If Psig is NA, then sig0 is used as a fixed sigma value (not tested).
range.fraction	The length of the spatial range inside the barrier area, as a fraction of the range parameter.
Omega	Advanced option for creating a set of permeable barriers (not documented)

Details

This model is described in the ArXiv preprint arXiv:1608.03787. For examples, see https://haakonbakka.bitbucket.io/btopic107.html.

Value

`inla.barrier.pcmatern` gives the (rgeneric) model object for fitting the model in INLA, `inla.barrier.polygon` gives the polygon around the barrier (mainly for plotting), `inla.barrier.q` is an internal method producing the Q matrix from a result of inla.barrier.fem, `inla.barrier.fem` is an internal method producing the Finite Element matrices.

Author(s)

Haakon Bakka <bakka@r-inla.org>

See Also

inla.spde2.pcmatern

`inla.barrier.pcmatern` *Functions for defining the Barrier models*

Description

Functions for defining Barrier models as an `inla rgeneric` model

Usage

```
inla.barrier.pcmatern(
  mesh,
  barrier.triangles,
  prior.range,
  prior.sigma,
  range.fraction = 0.2
)

inla.barrier.polygon(mesh, barrier.triangles, Omega = NULL)

inla.barrier.q(fem, ranges, sigma = 1, envir = NULL)

inla.barrier.fem(mesh, barrier.triangles, Omega = NULL)
```

Arguments

`mesh`	The mesh to build the model on, from inla.mesh.2d
`barrier.triangles`	The numerical ids of the triangles that make up the barrier area
`prior.range`	2 parameters `(range0,Prange)` for the prior spatial range. If `Prange` is `NA`, then `range0` is used as a fixed range value (not tested).
`prior.sigma`	2 parameters `(sig0,Psig)` for the prior marginal standard deviation sigma. If `Psig` is `NA`, then `sig0` is used as a fixed sigma value (not tested).

range.fraction	The length of the spatial range inside the barrier area, as a fraction of the range parameter.
Omega	Advanced option for creating a set of permeable barriers (not documented)
fem	represents the Barrier model or the Different Terrains (DT) model, by containing all the needed matrices to solve the SPDE
ranges, sigma	the hyperparameters that determine Q
envir	the environment used for caching (with optimize=TRUE), if any

Details

This model is described in the ArXiv preprint arXiv:1608.03787. For examples, see https://haakonbakkagit.github.io/btopic128.html

- inla.barrier.pcmatern This function creates the model component used in inla(...)
- inla.barrier.polygon This function constructs SpatialPolygons for the different subdomains (areas)
- inla.barrier.q: This function computes a specific precision matrix
- inla.barrier.fem This function computes the Finite Element matrices that are needed to compute the precision matrix Q later

Value

- inla.barrier.pcmatern gives the (rgeneric) model object for fitting the model in INLA
- inla.barrier.polygon gives the polygon around the barrier (mainly for plotting)
- inla.barrier.q is an internal method producing the Q matrix from a result of inla.barrier.fem,
- inla.barrier.fem is an internal method producing the Finite Element matrices.

Author(s)

Haakon Bakka <bakka@r-inla.org>

See Also

inla.spde2.pcmatern

---

inla.binary.install *Install alternative binary builds.*

Description

Install a new binary for os unless missing(os), for which the os is chosen interactively among the available builds.

Usage

```
inla.binary.install(
  os = c("CentOS Linux-6", "CentOS Linux-7", "CentOS Linux-8", "CentOS Stream-8",
   "Rocky Linux-9", "Fedora-33", "Fedora-34", "Fedora Linux-35", "Fedora Linux-36",
    "Fedora Linux-37", "Fedora Linux-38", "Manjaro Linux", "Ubuntu-16.04",
    "Ubuntu-18.04", "Ubuntu-20.04", "Ubuntu-22.04"),
  path = NULL,
  verbose = TRUE,
  md5.check = TRUE,
  secure.http = TRUE
)
```

Arguments

os	If os is given, install binary build for this os. If os is not given, chose os interactively among available builds.
path	character. The install path. If NULL the path is derived from INLA package
verbose	Logical. Verbose output if TRUE
md5.check	Logical. If TRUE, stop if md5-checksum-file is not present or md5-checksum fail. If FALSE, ignore md5-checksum check.
secure.http	Logical. Use secure http (ie https://) or http://

Value

Return TRUE if installation was sucessful and FALSE if not.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
   ## Not run:
     inla.binary.install()
     inla.binary.install(os = "CentOS Linux-7")
     inla.binary.install(os = "CentOS Linux-7",  path = "~/local/bin/inla.binary")

## End(Not run)
```

---

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,
  debug = FALSE,
  only.hyperparam = FALSE,
  file.log = NULL,
  file.log2 = NULL,
  silent = inla.getOption("silent")
)
```

Arguments

results.dir	The directory where the results of the inla run are stored
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
silent	Internal use only

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 run an inla section outside R.

Value

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

`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, tag = "")

inla.rbind.data.frames(...)
```

Arguments

`formula`	The formula for the coxph model where the response 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
`tag`	An optional tag added to the names of the new variables created (to make them unique when combined with several calls of `inla.coxph`. Note that `E..coxph` is not included, as its usually merged into one vector over different expansions.
`...`	Data.frames to be `rbind`-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 rbinded data.frames padded with NAs. There is a better implementation in `dplyr::bind_rows`, which is used if package `dplyr` is installed.

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..coxph)
```

`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,
  mc.cores = NULL,
  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`.

mc.cores	The number of cores to use in parallel::mclapply. If is.null(mc.cores), then check getOption("mc.cores") and inla.getOption("num.threads") in that order.
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

[Deprecated] in favour of fmesher::fm_CRS() Creates either a CRS object or an inla.CRS object, describing a coordinate reference system.

Usage

```
inla.CRS(..., args = NULL)

inla.wkt_predef()
```

Arguments

...	Arguments passed on to fmesher::fm_CRS(...).
args	list of named proj4 arguments.

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

`inla.wkt_predef` returns a WKT2 string defining a projection

`inla.wkt_predef` returns a WKT2 string defining a projection

Functions

- `inla.wkt_predef()`: **[Deprecated]** in favour of `fmesher::fm_wkt_predef()`

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

`sp::CRS()`, `crs_wkt()`, `inla.sp_get_crs()` `plot.CRS()`, `inla.identical.CRS()`

Examples

```
if (require("sf") && require("sp")) {
    crs1 <- fmesher::fm_CRS("longlat_globe")
    crs2 <- fmesher::fm_CRS("lambert_globe")
    crs3 <- fmesher::fm_CRS("mollweide_norm")
    crs4 <- fmesher::fm_CRS("hammer_globe")
    crs5 <- fmesher::fm_CRS("sphere")
    crs6 <- fmesher::fm_CRS("globe")
}
## Not run:
names(inla.wkt_predef())

## End(Not run)

## Not run:
names(inla.wkt_predef())

## End(Not run)
```

---

inla.CRSargs *Show expanded CRS arguments*

Description

Wrapper for `sp::CRS` and `inla.CRS` objects to extract the coordinate reference system argument string. 'r lifecycle::badge("deprecated")‘ in favour of `fmesher::fm_proj4string()`, or `fmesher::fm_wkt()` for WKT2 representations.

Usage

```
inla.CRSargs(x, ...)

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

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

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

inla.as.CRS.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.

Details

- inla.as.CRSargs.list: CRS proj4 string for name=value pair list
- inla.as.list.CRSargs: List of name=value pairs from CRS proj4 string

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

inla.CRS()

Examples

```
if (require("sf") && require("sp") && require("fmesher")) {
    crs0 <- fm_CRS("longlat_norm")
    p4s <- fm_proj4string(crs0)
    lst <- inla.as.list.CRSargs(p4s)
    crs1 <- inla.as.CRS.list(lst)
    lst$a <- 2
    crs2 <- fm_CRS(p4s, args = lst)
    print(fm_proj4string(crs0))
    print(fm_proj4string(crs1))
    print(fm_proj4string(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.1d'
inla.diameter(x, ...)

## 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, ...)
```

Arguments

x	A point set as an $n \times d$ matrix, or an inla.mesh related object.
...	Additional parameters passed on to other methods.
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).

Details

- inla.diameter.default Calculate upper bound for the diameter of a point set, by encapsulating in a circular domain.

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, section, verbose = FALSE)
```

Arguments

what	What to view documentation about; name of latent model, name of prior, etc. (A regular expression.)
section	An optional section, like names(inla.models()), 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", section = "prior")
```

`inla.external.lib` *Return the path to the cgeneric-library for a pre-compiled external package*

Description

Return the path to the cgeneric-library for a pre-compiled external package

Usage

```
inla.external.lib(package)
```

Arguments

package	the name of a package, given as a name or literal character string

Value

This function returns the complete path or `NULL` if file does not exists

Author(s)

Havard Rue <hrue@r-inla.org>

`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 'matrix'
inla.extract.el(M, match, by.row = TRUE, ...)

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

## S3 method for class 'list'
inla.extract.el(M, match, ...)
```

Arguments

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

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

---

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

Description

[Deprecated] Use the methods in the fmesher package instead; see details below.

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 = deprecated(),
  sph = deprecated(),
  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	**[Deprecated]** Use fmesher::fm_fem() instead. Maximum finite element matrix order to be computed.
aniso	**[Deprecated]** Use fmesher::fm_fem() instead. A two-element list with γ and v for an anisotropic operator $\nabla \cdot H \nabla$, where $H = \gamma I + vv^\top$
gradients	**[Deprecated]** Use fmesher::fm_fem() instead. When TRUE, calculate derivative operator matrices dx, dy, and dz.
sph0	**[Deprecated]** Use fmesher::fm_raw_basis() instead.
sph	**[Deprecated]** Use fmesher::fm_raw_basis() instead.

bspline	**[Deprecated]** Use fmesher::fm_raw_basis() instead. 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	**[Deprecated]** Use fmesher::fm_bary() instead. 3-column matrix with points to be located in the mesh.
splitlines	**[Deprecated]** Use fmesher::fm_split_lines() or fmesher::fmesher_split_lines() instead. 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.get.inlaEnv *Return the internal environment used by INLA*

Description

A function which return the internal environment used by INLA

Usage

```
inla.get.inlaEnv()
```

Value

This function returns the internal environment used by INLA to keep internal variables.

Author(s)

Havard Rue <hrue@r-inla.org>

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.group.cv *Compute group.cv-values*

Description

From a fitted model, compute and add the group.cv-values

Usage

```
inla.group.cv(
  result,
  group.cv = NULL,
  num.level.sets = -1,
  strategy = c("posterior", "prior"),
  size.max = 32,
  groups = NULL,
  selection = NULL,
  friends = NULL,
  verbose = FALSE,
  epsilon = 0.005,
  prior.diagonal = 1e-04,
  keep = NULL,
  remove = NULL,
  remove.fixed = TRUE
)
```

Arguments

result	An object of class inla, ie a result of a call to inla().
group.cv	If given, the groups are taken from this argument. group.cv must be the output of previous call to inla.group.cv().
num.level.sets	Number of level.sets to use. The default value -1 corresponds to leave-one-out cross-validation.
strategy	One of "posterior" or "prior". See the vignette for details.

size.max	The maximum size of a group. If the computed group-size is larger, it will be truncated to size.max.
groups	An (optional) predefined list of groups. See the vignette for details.
selection	An optional list of data-indices to use. If not given, then all data are used.
friends	An optional list of lists of indices to use a friends
verbose	Run with verbose output of some of the internals in the calculations. This option will also enable inla(..., verbose=TRUE) if its not enabled already.
epsilon	Two correlations with a difference less than epsilon, will be classified as identical.
prior.diagonal	When strategy="prior", prior.diagonal is added to the diagonal of the prior precision matrix to avoid singularities
keep	For strategy="prior", then this gives a vector of the name of model-components TO USE when computing the groups. See the vignette for details. Not both of keep and remove can be defined.
remove	For strategy="prior", then this gives a vector of the name of model-components NOT TO USE when computing the groups. See the vignette for details. Not both of keep and remove can be defined.
remove.fixed	For strategy="prior", this is the default option which is in effect if both keep and remove are NULL. If TRUE, it will remove (or condition on) all fixed effects when computing the groups. See the vignette for details.

Value

The object returned is list related to leave-group-out cross-validation. See the vignette for details.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

control.compute()

---

inla.has_PROJ6 *PROJ6 detection*

Description

Detect whether PROJ6 is available for INLA. Deprecated and always returns TRUE..

Usage

```
inla.has_PROJ6()

inla.not_for_PROJ6(fun)

inla.not_for_PROJ4(fun)

inla.fallback_PROJ6(fun)

inla.requires_PROJ6(fun)
```

Arguments

fun	The name of the calling function

Details

inla.has_PROJ6 is called to check if PROJ6&GDAL3 are available.

Value

For inla.has_PROJ6, always returns TRUE. Previously: logical; TRUE if PROJ6 is available, FALSE otherwise

Functions

- inla.has_PROJ6(): **[Deprecated]**
- inla.not_for_PROJ6(): **[Deprecated]** Called to warn about using old PROJ4 features even though PROJ6 is available
- inla.not_for_PROJ4(): **[Deprecated]** Called to give an error when calling methods that are only available for PROJ6
- inla.fallback_PROJ6(): **[Deprecated]** Called to warn about falling back to using old PROJ4 methods when a PROJ6 method hasn't been implemented
- inla.requires_PROJ6(): **[Deprecated]** Called to give an error when PROJ6 is required but not available

Examples

```
## Not run:
inla.has_PROJ6()

## End(Not run)
```

---

inla.hyperpar *Improved estimates for the hyperparameters (classic-mode only)*

Description

Improve the estimates of the posterior marginals for the hyperparameters of the model using the grid integration strategy. (classic-mode only)

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() in classic mode
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 whether 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 Deprecated in favour of `fmesher::fm_crs_is_identical()`

Usage

```
inla.identical.CRS(...)
```

Arguments

`...`	Arguments passed on to `fmesher::fm_crs_is_identical()`

inla.iidkd.sample *Provide samples from the* `iidkd` *component (experimental)*

Description

This function provide samples of the `iidkd` component using more interpretable parameters

Usage

```
inla.iidkd.sample(n = 10^4, result, name, return.cov = FALSE)
```

Arguments

`n`	Integer Number of samples to use
`result`	inla-object An object of class `inla`, ie a result of a call to `inla()`
`name`	Character The name of the `iidkd` component
`return.cov`	Logical Return samples of the covariance matrix instead of stdev/correlation matrix described below?

Value

A list of sampled matrices, with (default) correlations on the off-diagonal and standard-deviations on the diagonal

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

```
inla.doc("iidkd")
```

`inla.knmodels` *Spacetime interaction models*

Description

It implements the models in Knorr-Held, L. (2000) with three different constraint approaches: sum-to-zero, contrast or diagonal add.

Usage

```
inla.knmodels(
  formula,
  progress = FALSE,
 control.st = list(time, space, spacetime, graph, type = c(paste(1:4), paste0(2:4, "c"),
    paste0(2:4, "d")), diagonal = 1e-05, timeref = 1, spaceref = 1),
  ...,
  envir = parent.frame()
)
```

Arguments

`formula` The formula specifying the other model components, without the spacetime interaction term. The spacetime interaction term will be added accordly to the specification in the `control.st` argument. See `inla`

`progress` If it is to be shown the model fitting progress. Useful if more than one interaction type is being fitted.

`control.st` Named list of arguments to control the spacetime interaction. It should contain:

time to be used as the index set for the main temporal effect which will be considered for the constraints when it is the case.

space to be used as the index set for the main spatial effect which will be considered for the constraints when it is the case.

spacetime to be the index set for the spacetime interaction effect.

graph to be the graph for the spatial neighbor structure to be used in a `f()` term for the main spatial random effect term or for building the spacetime interaction model.

type to specify the spacetime interaction type. `1` to `4` corresponds to the four interaction types in Knorr-Held, L. (2000) with all the needed sum-to-zero constraints. `2c`, `3c` and `4c` are the contrast version considering the first time or space constrained to be equal to zero. `2d`, `3d` and `4d` are the corresponding versions when considering the diagonal add approach.

diagonal to be the value to be added to the diagonal when using the diagonal add approach.

timeref to specify the time point to be the reference time in the contrast parametrization.

spaceref to specify the area to be the reference for the contrast parametrization.

... where additional arguments can be passed to `f()` function. Specification of the hyperparameter, fixed or random, initial value, prior and its parameters for the spacetime interaction. See `?inla.models` and look for

generic0. By default we scale it and use the PC-prior to set the prior using the pc.prec prior with param = c(0.5, 0.5). See documentation with ?inla.doc("pc.prec").

... Arguments to be passed to the inla() function.

envir Environment in which to evaluate the ... arguments.

Value

inla.knmodels returns an object of class "inla". or a list of objects of this class if it is asked to compute more than one interaction type at once. Note: when the model type is 2c, 3c, 4c, 2d, 3d or 4d, it also includes linear combinations summary.

Author(s)

Elias T. Krainski

See Also

inla.knmodels.sample() to sample from

Examples

```
### define space domain as a grid
grid <- SpatialGrid(GridTopology(c(0,0), c(1, 1), c(4, 5)))
(n <- nrow(xy <- coordinates(grid)))

### build a spatial neighborhood list
jj <- lapply(1:n, function(i)
    which(sqrt((xy[i,1]-xy[,1])^2 + (xy[i,2]-xy[,2])^2)==1))

### build the spatial adjacency matrix
graph <- sparseMatrix(rep(1:n, sapply(jj, length)),
                      unlist(jj), x=1, dims=c(n, n))

### some random data at 10 time point
dat <- inla.knmodels.sample(graph, m=10, tau.t=2, tau.s=2, tau.st=3)
str(dat)
sapply(dat$x, summary)

nd <- length(dat$x$eta)
dat$e <- runif(nd, 0.9, 1.1)*rgamma(n, 40, 2)
dat$y <- rpois(nd, dat$e*exp(dat$x$eta-3))
summary(dat$y)

### fit the type 4 considering three different approaches
tgraph <- sparseMatrix(i=c(2:10, 1:9), j=c(1:9, 2:10), x=1)
res <- inla.knmodels(y ~ f(time, model='bym2', graph=tgraph) +
     f(space, model='bym2', graph=graph),
     data=dat, family='poisson', E=dat$E, progress=TRUE,
     control.st=list(time=time, space=space,
        spacetime=spacetime, graph=graph, type=c(4, '4c')),
     control.compute=list(dic=TRUE, waic=TRUE, cpo=TRUE))
sapply(res, function(x)
       c(dic=x$dic$dic, waic=x$waic$waic, cpo=-sum(log(x$cpo$cpo))))
```

`inla.knmodels.sample` *Spacetime interaction models sampler function*

Description

It implements the sampling method for the models in Knorr-Held, L. (2000) considering the algorithm 3.1 in Rue & Held (2005) book.

Usage

```
inla.knmodels.sample(
  graph,
  m,
  type = 4,
  intercept = 0,
  tau.t = 1,
  phi.t = 0.7,
  tau.s = 1,
  phi.s = 0.7,
  tau.st = 1,
  ev.t = NULL,
  ev.s = NULL
)
```

Arguments

`graph`	Model graph definition
`m`	Time dimention.
`type`	Integer from 1 to 4 to identify one of the four interaction type.
`intercept`	A constant to be added to the linear predictor
`tau.t`	Precision parameter for the main temporal effect.
`phi.t`	Mixing parameter in the `bym2` model assumed for the main temporal effect.
`tau.s`	Precision parameter for the main spatial effect.
`phi.s`	Mixing parameter in the `bym2` model assumed for the main spatial effect.
`tau.st`	Precision parameter for the spacetime effect.
`ev.t`	Eigenvalues and eigenvectors of the temporal precision matrix structure.
`ev.s`	Eigenvalues and eigenvectors of the spatial precision matrix structure.

Value

A list with the following elements

`time`	The time index for each obervation, with length equals `m*n`.
`space`	The spatial index for each observation, with length equals `m*n`.
`spacetime`	The spacetime index for each obervation, with length equals `m*n`.
`x`	A list with the following elements
`t.iid`	The unstructured main temporal effect part.

t.str	The structured main temporal effect part.
t	The main temporal effect with length equals 2m.
s.iid	The unstructured main spatial effect part.
s.str	The structured main spatial effect part.
s	The main spatial effect with length equals 2n.
st	The spacetime interaction effect with length m*n.
eta	The linear predictor with length n*m.

Author(s)

Elias T. Krainski

See Also

inla.knmodels() for model fitting

---

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.likelihood *Providing functions for sampling new data, evaluating pdf, cdf, and quantiles for new data.*

Description

This function return function to compute the pdf,cdf,quantiles, or samples for new data using the likelihood from a inla-object.

Usage

```
inla.likelihood(type = c("d", "p", "r", "q", "s"), args)
```

Arguments

type	The returned function type. The definition is similar to "rnorm","dnorm","pnorm",and "dnorm".
args	It is usually a return value from "inla.likelihood.parser", which specifies parameters, link function and transformation function of hyperparameters.

Value

value goes here

Author(s)

Havard Rue <hrue@r-inla.org>

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

Description

List available model components, likelihoods, priors, etc. To read specific documentation for the individual elements, use `inla.doc()`.

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

This function is EXPERIMENTAL.

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`

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"))

#Show detailed doc for a specific prior/likelihood/latent model
inla.doc("binomial")

## End(Not run)
```

`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, freq.max = NULL)
```

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.
`freq.max`	The maximum allowed harmonic order. Current default 40, to be changed to a dynamic choice based on error bounds.

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, ...)

as.inla.mdata(object)

is.inla.mdata(object)
```

Arguments

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

Note

It is often required to set `Y=inla.mdata(...)` and then define the formula as `Y~...`, especially when used with `inla.stack`.

Author(s)

Havard Rue

See Also

`inla()`

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

Description

[Deprecated] in favour of `fmesher::fm_mesh_1d()`

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,
  ...
)

inla.mesh.1d.fem(mesh)
```

Arguments

`loc`	B-spline knot locations.
`interval`	Interval domain endpoints.
`boundary`	Boundary condition specification. Valid conditions are `c('neumann', 'dirichlet', 'free', 'cyclic')`. 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.
`mesh`	An inla.mesh.1d object

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

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

Description

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

Usage

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

inla.mesh.1d.A(mesh, loc, weights = NULL, derivatives = NULL, method = NULL)
```

Arguments

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

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

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

Description

[Deprecated] in favour of `fmesher::fm_mesh_2d_inla()`.

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 = NULL,
  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(), inla.nonconvex.hull()

Examples

```
loc <- matrix(runif(10 * 2), 10, 2)

if (require("splancs")) {
  boundary <- list(
    inla.nonconvex.hull(loc, 0.1, 0.15),
    inla.nonconvex.hull(loc, 0.2, 0.2)
  )
  offset <- NULL
} else {
  boundary <- NULL
  offset <- c(0.1, 0.2)
}
mesh <- inla.mesh.2d(loc, boundary = boundary, offset = offset, max.edge = c(0.05, 0.1))

plot(mesh)
```

---

inla.mesh.assessment *Interactive mesh building and diagnostics*

Description

Assess the finite element approximation errors in a mesh for interactive R sessions. More detailed assessment tools are in `meshbuilder()`.

Usage

```
inla.mesh.assessment(mesh, spatial.range, alpha = 2, dims = c(500, 500))
```

Arguments

`mesh`	An `inla.mesh`
`spatial.range`	numeric; the spatial range parameter to use for the assessment
`alpha`	numeric; A valid `inla.spde2.pcmatern` `alpha` parameter
`dims`	2-numeric; the grid size

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

inla.mesh.2d, inla.mesh.create, meshbuilder

Examples

```
bnd <- inla.mesh.segment(cbind(
    c(0, 10, 10, 0, 0),
    c(0, 0, 10, 10, 0)
), bnd = TRUE)
mesh <- inla.mesh.2d(boundary = bnd, max.edge = 1)
out <- inla.mesh.assessment(mesh, spatial.range = 3, alpha = 2)
```

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

Description

[Deprecated] Use the `fmesher::fm_raw_basis()` instead.

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 only 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.
`...`	Unused

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.components` *Compute connected mesh subsets*

Description

Compute subsets of vertices and triangles in an inla.mesh object that are connected by edges.

Usage

```
inla.mesh.components(mesh)
```

Arguments

`mesh`	An `inla.mesh` object

Value

A list with elements `vertex` and `triangle`, vectors of integer labels for which connected component they belong, and `info`, a `data.frame` with columns

`component`	Connected component integer label.
`nV`	The number of vertices in the component.
`nT`	The number of triangles in the component.
`area`	The surface area associated with the component. Component lables are not comparable across different meshes, but some ordering stability is guaranteed by initiating each component from the lowest numbered triangle whenever a new component is initiated.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

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

Examples

```
# Construct two simple meshes:
loc <- matrix(c(0, 1, 0, 1), 2, 2)
mesh1 <- inla.mesh.2d(loc = loc, max.edge = 0.1)
bnd <- inla.nonconvex.hull(loc, 0.3)
mesh2 <- inla.mesh.2d(boundary = bnd, max.edge = 0.1)

# Compute connectivity information:
conn1 <- inla.mesh.components(mesh1)
conn2 <- inla.mesh.components(mesh2)
# One component, simply connected mesh
conn1$info
# Two disconnected components
conn2$info
```

```
# Extract the subset mesh for the largest component:
# (Note: some information is lost, such as fixed segments,
# and boundary edge labels.)
maxi <- conn2$info$component[which.max(conn2$info$area)]
mesh3 <- inla.mesh.create(
    loc = mesh2$loc,
    tv = mesh2$graph$tv[conn2$triangle == maxi, , drop = FALSE]
)
```

`inla.mesh.components` *Compute connected mesh subsets*

Description

Compute subsets of vertices and triangles in an inla.mesh object that are connected by edges.

Usage

```
inla.mesh.components(mesh)
```

Value

A list with elements `vertex` and `triangle`, vectors of integer labels for which connected component they belong, and `info`, a `data.frame` with columns

`component`	Connected component integer label.
`nV`	The number of vertices in the component.
`nT`	The number of triangles in the component.
`area`	The surface area associated with the component. Component lables are not comparable across different meshes, but some ordering stability is guaranteed by initiating each component from the lowest numbered triangle whenever a new component is initiated.

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

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

Examples

```
# Construct two simple meshes:
loc <- matrix(c(0,1,0,1), 2, 2)
mesh1 <- inla.mesh.2d(loc = loc, max.edge=0.1)
bnd <- inla.nonconvex.hull(loc, 0.3)
mesh2 <- inla.mesh.2d(boundary = bnd, max.edge=0.1)

# Compute connectivity information:
conn1 <- inla.mesh.components(mesh1)
conn2 <- inla.mesh.components(mesh2)
# One component, simply connected mesh
```

```
conn1$info
# Two disconnected components
conn2$info

# Extract the subset mesh for the largest component:
# (Note: some information is lost, such as fixed segments,
#  and boundary edge labels.)
maxi <- conn2$info$component[which.max(conn2$info$area)]
mesh3 <- inla.mesh.create(loc = mesh2$loc,
                          tv = mesh2$graph$tv[conn2$triangle == maxi,,drop=FALSE])
```

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

Description

[Deprecated] in favour of `fmesher::fm_rcdt_2d_inla()`.

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

`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.

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 = NULL,
  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 **list("n")** the number of edges in the extended boundary (default=8) **list("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 **list("min.angle")** the minimum allowed interior angle in any triangle. The algorithm is guaranteed to converge for min.angle at most 21 (default=21) **list("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) **list("max.n.strict")** the maximum number of vertices allowed, overriding min.angle and max.edge (default=-1, meaning no limit) **list("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.

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

```
inla.mesh.fem(mesh, order = 2)
```

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>

See Also

`inla.mesh.1d.fem()`

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	vector or grid matrix of x-values
y	vector of grid matrix of y-values
z	if x is a matrix, a grid matrix of z-values
dims	the size of the grid, length 2 vector
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.lim` *Coordinate mappings for* `inla.mesh` *projections.*

Description

Calculates coordinate mappings for `inla.mesh` projections.

Usage

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

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

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.

Functions

- `inla.mesh.map.lim()`: Projection extent limit calculations

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(). Deprecated in favour of fmesher::fm_evaluate() and fmesher::fm_evaluator().

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

Usage

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

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

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

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

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, ...)
```

Arguments

...	Additional arguments passed on to methods.
mesh	An inla.mesh() or inla.mesh.1d() object.
loc	Projection locations. Can be a matrix or a SpatialPoints or a SpatialPointsDataFrame 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
crs	An optional CRS or inla.CRS object associated with loc and/or lattice.
projector	An inla.mesh.projector 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.
projection	One of c("default", "longlat", "longsinlat", "mollweide").

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

[Deprecated] in favour of `fmesher::fm_segm()`

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

Usage

```
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

`...`	Additional parameters. When joining segments, a list of `inla.mesh.segment` objects.
`loc`	Matrix of point locations, or `SpatialPoints`, or `sf`/`sfc` point object.
`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.
crs	An optional CRS or inla.CRS object
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().

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, scopy, mix, link, predictor, hazard, likelihood, prior, wrapper, lp.scale.

Usage

```
inla.models()
```

Value

Valid sections are: latent, group, scopy, mix, link, predictor, hazard, likelihood, prior, wrapper, lp.scale.

'latent'

Valid models in this section are:

Model 'linear'. Properties: doc = 'Alternative interface to an fixed 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 = 'linear'
Number of hyperparmeters is 0.

Model 'iid'. 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'
Number of hyperparmeters is 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)`

Model 'mec'. 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'
pdf = 'mec'

Number of hyperparmeters is 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)`

Model 'meb'. 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'
pdf = 'meb'

Number of hyperparmeters is 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)`

Model 'rgeneric'. Properties: doc = 'Generic latent model specified using R'
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 = 'rgeneric'
Number of hyperparmeters is 0.

Model 'cgeneric'. Properties: doc = 'Generic latent model specified using C'
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 = 'rgeneric'
Number of hyperparmeters is 0.

Model 'rw1'. 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'
Number of hyperparmeters is 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)`

Model 'rw2'. 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'
Number of hyperparmeters is 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)`
Model 'crw2'. 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'
Number of hyperparmeters is 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)`
Model 'seasonal'. 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'
Number of hyperparmeters is 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)`

Model 'besag'. 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'

Number of hyperparmeters is 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)`

Model 'besag2'. Properties: doc = 'The shared Besag model'
constr = 'TRUE'
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'

Number of hyperparmeters is 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)`
Model 'bym'. 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'
Number of hyperparmeters is 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)`
Model 'bym2'. 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'

Number of hyperparmeters is 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))`

Model 'besagproper'. 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'

Number of hyperparmeters is 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)`
Model 'besagproper2'. 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'
Number of hyperparmeters is 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))`
Model 'fgn'. Properties: doc = 'Fractional Gaussian noise model'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'TRUE'

aug.factor = '5'
aug.constr = '1'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'TRUE'
order.default = '4'
order.defined = '3 4'
pdf = 'fgn'

Number of hyperparmeters is 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))`

Model 'fgn2'. Properties: doc = 'Fractional Gaussian noise model (alt 2)'
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 = '4'
order.defined = '3 4'
pdf = 'fgn'

Number of hyperparmeters is 2.

Hyperparameter 'theta1' hyperid = 13111
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 = 13112
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))`
Model 'ar1'. 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'
Number of hyperparmeters is 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`

Model 'ar1c'. Properties: doc = 'Auto-regressive model of order 1 w/covariates'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'FALSE'
aug.factor = '1'
aug.constr = 'NULL'
n.div.by = 'NULL'
n.required = 'FALSE'
set.default.values = 'TRUE'
status = 'experimental'
pdf = 'ar1c'

Number of hyperparmeters is 2.

Hyperparameter 'theta1' hyperid = 14101
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 = 14102
name = logit lag one correlation
short.name = rho
prior = pc.cor0
param = 0.5 0.5
initial = 2
fixed = FALSE
to.theta = `function(x) log((1 + x) / (1 - x))`
from.theta = `function(x) 2 * exp(x) / (1 + exp(x)) - 1`

Model 'ar'. 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'

Number of hyperparmeters is 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`
Model 'ou'. 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'
Number of hyperparmeters is 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)`
Model 'intslope'. Properties: doc = 'Intecept-slope model 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'
status = 'experimental'
pdf = 'intslope'

Number of hyperparmeters is 13.

Hyperparameter 'theta1' hyperid = 16101
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 = 16102
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 = 16103
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`

Hyperparameter 'theta4' hyperid = 16104
name = gamma1
short.name = g1
initial = 1
fixed = TRUE
prior = normal
param = 1 36
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta5' hyperid = 16105
name = gamma2

short.name = g2
initial = 1
fixed = TRUE
prior = normal
param = 1 36
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta6' hyperid = 16106
name = gamma3
short.name = g3
initial = 1
fixed = TRUE
prior = normal
param = 1 36
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta7' hyperid = 16107
name = gamma4
short.name = g4
initial = 1
fixed = TRUE
prior = normal
param = 1 36
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta8' hyperid = 16108
name = gamma5
short.name = g5
initial = 1
fixed = TRUE
prior = normal
param = 1 36
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta9' hyperid = 16109
name = gamma6
short.name = g6
initial = 1
fixed = TRUE
prior = normal
param = 1 36
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta10' hyperid = 16110
name = gamma7

short.name = g7
initial = 1
fixed = TRUE
prior = normal
param = 1 36
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta11' hyperid = 16111
name = gamma8
short.name = g8
initial = 1
fixed = TRUE
prior = normal
param = 1 36
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta12' hyperid = 16112
name = gamma9
short.name = g9
initial = 1
fixed = TRUE
prior = normal
param = 1 36
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta13' hyperid = 16113
name = gamma10
short.name = g10
initial = 1
fixed = TRUE
prior = normal
param = 1 36
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'generic'. 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'
Number of hyperparmeters is 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)`

Model 'generic0'. 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'
Number of hyperparmeters is 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)`

Model 'generic1'. 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'
Number of hyperparmeters is 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))`
Model 'generic2'. 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'
Number of hyperparmeters is 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)`
Model 'generic3'. 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'

Number of hyperparmeters is 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)`

Model 'spde'. 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'
Number of hyperparmeters is 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))`
Model 'spde2'. 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'
Number of hyperparmeters is 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`
Model 'spde3'. 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'
Number of hyperparmeters is 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`

Model 'iid1d'. 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'

Number of hyperparmeters is 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)`

Model 'iid2d'. 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'

Number of hyperparmeters is 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`

Model 'iid3d'. 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'

Number of hyperparmeters is 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`

Model 'iid4d'. 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'
Number of hyperparmeters is 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`
Model 'iid5d'. 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'

Number of hyperparmeters is 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`

Model 'iidkd'. Properties: doc = 'Gaussian random effect in dim=k with Wishart prior'
constr = 'FALSE'
nrow.ncol = 'FALSE'
augmented = 'TRUE'
aug.factor = '1'
aug.constr = '1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20'
n.div.by = '-1'
n.required = 'TRUE'
set.default.values = 'TRUE'
pdf = 'iidkd'

Number of hyperparmeters is 210.

Hyperparameter 'theta1' hyperid = 29101
name = theta1
short.name = theta1
initial = 1048576
fixed = FALSE
prior = wishartkd
param = 21 1048576

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta2' hyperid = 29102
name = theta2
short.name = theta2
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta3' hyperid = 29103
name = theta3
short.name = theta3
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta4' hyperid = 29104
name = theta4
short.name = theta4
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta5' hyperid = 29105
name = theta5
short.name = theta5
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 29106
name = theta6
short.name = theta6
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 29107
name = theta7
short.name = theta7
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta8' hyperid = 29108
name = theta8
short.name = theta8
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta9' hyperid = 29109
name = theta9
short.name = theta9
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta10' hyperid = 29110
name = theta10
short.name = theta10
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta11' hyperid = 29111
name = theta11
short.name = theta11
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta12' hyperid = 29112
name = theta12
short.name = theta12
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta13' hyperid = 29113
name = theta13
short.name = theta13
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta14' hyperid = 29114
name = theta14
short.name = theta14
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta15' hyperid = 29115
name = theta15
short.name = theta15
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta16' hyperid = 29116
name = theta16
short.name = theta16
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta17' hyperid = 29117
name = theta17
short.name = theta17
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta18' hyperid = 29118
name = theta18
short.name = theta18
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta19' hyperid = 29119
name = theta19
short.name = theta19
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta20' hyperid = 29120
name = theta20
short.name = theta20
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta21' hyperid = 29121
name = theta21
short.name = theta21
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta22' hyperid = 29122
name = theta22
short.name = theta22
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta23' hyperid = 29123
name = theta23
short.name = theta23
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta24' hyperid = 29124
name = theta24
short.name = theta24
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta25' hyperid = 29125
name = theta25
short.name = theta25
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta26' hyperid = 29126
name = theta26
short.name = theta26
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta27' hyperid = 29127
name = theta27
short.name = theta27
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta28' hyperid = 29128
name = theta28
short.name = theta28
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta29' hyperid = 29129
name = theta29
short.name = theta29
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta30' hyperid = 29130
name = theta30
short.name = theta30
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta31' hyperid = 29131
name = theta31
short.name = theta31
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta32' hyperid = 29132
name = theta32
short.name = theta32
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta33' hyperid = 29133
name = theta33
short.name = theta33
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta34' hyperid = 29134
name = theta34
short.name = theta34
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta35' hyperid = 29135
name = theta35
short.name = theta35
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta36' hyperid = 29136
name = theta36
short.name = theta36
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta37' hyperid = 29137
name = theta37
short.name = theta37
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta38' hyperid = 29138
name = theta38
short.name = theta38
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta39' hyperid = 29139
name = theta39
short.name = theta39
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta40' hyperid = 29140
name = theta40
short.name = theta40
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta41' hyperid = 29141
name = theta41
short.name = theta41
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta42' hyperid = 29142
name = theta42
short.name = theta42
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta43' hyperid = 29143
name = theta43
short.name = theta43
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta44' hyperid = 29144
name = theta44
short.name = theta44
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta45' hyperid = 29145
name = theta45
short.name = theta45
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta46' hyperid = 29146
name = theta46
short.name = theta46
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta47' hyperid = 29147
name = theta47
short.name = theta47
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta48' hyperid = 29148
name = theta48
short.name = theta48
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta49' hyperid = 29149
name = theta49
short.name = theta49
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta50' hyperid = 29150
name = theta50
short.name = theta50
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta51' hyperid = 29151
name = theta51
short.name = theta51
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta52' hyperid = 29152
name = theta52
short.name = theta52
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta53' hyperid = 29153
name = theta53
short.name = theta53
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta54' hyperid = 29154
name = theta54
short.name = theta54
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta55' hyperid = 29155
name = theta55
short.name = theta55
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta56' hyperid = 29156
name = theta56
short.name = theta56
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta57' hyperid = 29157
name = theta57
short.name = theta57
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta58' hyperid = 29158
name = theta58
short.name = theta58
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta59' hyperid = 29159
name = theta59
short.name = theta59
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta60' hyperid = 29160
name = theta60
short.name = theta60
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta61' hyperid = 29161
name = theta61
short.name = theta61
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta62' hyperid = 29162
name = theta62
short.name = theta62
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta63' hyperid = 29163
name = theta63
short.name = theta63
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta64' hyperid = 29164
name = theta64
short.name = theta64
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta65' hyperid = 29165
name = theta65
short.name = theta65
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta66' hyperid = 29166
name = theta66
short.name = theta66
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta67' hyperid = 29167
name = theta67
short.name = theta67
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta68' hyperid = 29168
name = theta68
short.name = theta68
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta69' hyperid = 29169
name = theta69
short.name = theta69
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta70' hyperid = 29170
name = theta70
short.name = theta70
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta71' hyperid = 29171
name = theta71
short.name = theta71
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta72' hyperid = 29172
name = theta72
short.name = theta72
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta73' hyperid = 29173
name = theta73
short.name = theta73
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta74' hyperid = 29174
name = theta74
short.name = theta74
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta75' hyperid = 29175
name = theta75
short.name = theta75
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta76' hyperid = 29176
name = theta76
short.name = theta76
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta77' hyperid = 29177
name = theta77
short.name = theta77
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta78' hyperid = 29178
name = theta78
short.name = theta78
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta79' hyperid = 29179
name = theta79
short.name = theta79
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta80' hyperid = 29180
name = theta80
short.name = theta80
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta81' hyperid = 29181
name = theta81
short.name = theta81
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta82' hyperid = 29182
name = theta82
short.name = theta82
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta83' hyperid = 29183
name = theta83
short.name = theta83
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta84' hyperid = 29184
name = theta84
short.name = theta84
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta85' hyperid = 29185
name = theta85
short.name = theta85
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta86' hyperid = 29186
name = theta86
short.name = theta86
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta87' hyperid = 29187
name = theta87
short.name = theta87
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta88' hyperid = 29188
name = theta88
short.name = theta88
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta89' hyperid = 29189
name = theta89
short.name = theta89
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta90' hyperid = 29190
name = theta90
short.name = theta90
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta91' hyperid = 29191
name = theta91
short.name = theta91
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta92' hyperid = 29192
name = theta92
short.name = theta92
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta93' hyperid = 29193
name = theta93
short.name = theta93
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta94' hyperid = 29194
name = theta94
short.name = theta94
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta95' hyperid = 29195
name = theta95
short.name = theta95
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta96' hyperid = 29196
name = theta96
short.name = theta96
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta97' hyperid = 29197
name = theta97
short.name = theta97
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta98' hyperid = 29198
name = theta98
short.name = theta98
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta99' hyperid = 29199
name = theta99
short.name = theta99
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta100' hyperid = 29200
name = theta100
short.name = theta100
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta101' hyperid = 29201
name = theta101
short.name = theta101
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta102' hyperid = 29202
name = theta102
short.name = theta102
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta103' hyperid = 29203
name = theta103
short.name = theta103
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta104' hyperid = 29204
name = theta104
short.name = theta104
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta105' hyperid = 29205
name = theta105
short.name = theta105
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta106' hyperid = 29206
name = theta106
short.name = theta106
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta107' hyperid = 29207
name = theta107
short.name = theta107
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta108' hyperid = 29208
name = theta108
short.name = theta108
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta109' hyperid = 29209
name = theta109
short.name = theta109
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta110' hyperid = 29210
name = theta110
short.name = theta110
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta111' hyperid = 29211
name = theta111
short.name = theta111
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta112' hyperid = 29212
name = theta112
short.name = theta112
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta113' hyperid = 29213
name = theta113
short.name = theta113
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta114' hyperid = 29214
name = theta114
short.name = theta114
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta115' hyperid = 29215
name = theta115
short.name = theta115
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta116' hyperid = 29216
name = theta116
short.name = theta116
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta117' hyperid = 29217
name = theta117
short.name = theta117
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta118' hyperid = 29218
name = theta118
short.name = theta118
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta119' hyperid = 29219
name = theta119
short.name = theta119
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta120' hyperid = 29220
name = theta120
short.name = theta120
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta121' hyperid = 29221
name = theta121
short.name = theta121
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta122' hyperid = 29222
name = theta122
short.name = theta122
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta123' hyperid = 29223
name = theta123
short.name = theta123
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta124' hyperid = 29224
name = theta124
short.name = theta124
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta125' hyperid = 29225
name = theta125
short.name = theta125
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta126' hyperid = 29226
name = theta126
short.name = theta126
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta127' hyperid = 29227
name = theta127
short.name = theta127
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta128' hyperid = 29228
name = theta128
short.name = theta128
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta129' hyperid = 29229
name = theta129
short.name = theta129
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta130' hyperid = 29230
name = theta130
short.name = theta130
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta131' hyperid = 29231
name = theta131
short.name = theta131
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta132' hyperid = 29232
name = theta132
short.name = theta132
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta133' hyperid = 29233
name = theta133
short.name = theta133
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta134' hyperid = 29234
name = theta134
short.name = theta134
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta135' hyperid = 29235
name = theta135
short.name = theta135
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta136' hyperid = 29236
name = theta136
short.name = theta136
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta137' hyperid = 29237
name = theta137
short.name = theta137
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta138' hyperid = 29238
name = theta138
short.name = theta138
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta139' hyperid = 29239
name = theta139
short.name = theta139
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta140' hyperid = 29240
name = theta140
short.name = theta140
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta141' hyperid = 29241
name = theta141
short.name = theta141
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta142' hyperid = 29242
name = theta142
short.name = theta142
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta143' hyperid = 29243
name = theta143
short.name = theta143
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta144' hyperid = 29244
name = theta144
short.name = theta144
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta145' hyperid = 29245
name = theta145
short.name = theta145
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta146' hyperid = 29246
name = theta146
short.name = theta146
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta147' hyperid = 29247
name = theta147
short.name = theta147
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta148' hyperid = 29248
name = theta148
short.name = theta148
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta149' hyperid = 29249
name = theta149
short.name = theta149
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta150' hyperid = 29250
name = theta150
short.name = theta150
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta151' hyperid = 29251
name = theta151
short.name = theta151
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta152' hyperid = 29252
name = theta152
short.name = theta152
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta153' hyperid = 29253
name = theta153
short.name = theta153
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta154' hyperid = 29254
name = theta154
short.name = theta154
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta155' hyperid = 29255
name = theta155
short.name = theta155
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta156' hyperid = 29256
name = theta156
short.name = theta156
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta157' hyperid = 29257
name = theta157
short.name = theta157
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta158' hyperid = 29258
name = theta158
short.name = theta158
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta159' hyperid = 29259
name = theta159
short.name = theta159
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta160' hyperid = 29260
name = theta160
short.name = theta160
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta161' hyperid = 29261
name = theta161
short.name = theta161
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta162' hyperid = 29262
name = theta162
short.name = theta162
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta163' hyperid = 29263
name = theta163
short.name = theta163
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta164' hyperid = 29264
name = theta164
short.name = theta164
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta165' hyperid = 29265
name = theta165
short.name = theta165
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta166' hyperid = 29266
name = theta166
short.name = theta166
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta167' hyperid = 29267
name = theta167
short.name = theta167
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta168' hyperid = 29268
name = theta168
short.name = theta168
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta169' hyperid = 29269
name = theta169
short.name = theta169
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta170' hyperid = 29270
name = theta170
short.name = theta170
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta171' hyperid = 29271
name = theta171
short.name = theta171
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta172' hyperid = 29272
name = theta172
short.name = theta172
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta173' hyperid = 29273
name = theta173
short.name = theta173
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta174' hyperid = 29274
name = theta174
short.name = theta174
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta175' hyperid = 29275
name = theta175
short.name = theta175
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta176' hyperid = 29276
name = theta176
short.name = theta176
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta177' hyperid = 29277
name = theta177
short.name = theta177
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta178' hyperid = 29278
name = theta178
short.name = theta178
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta179' hyperid = 29279
name = theta179
short.name = theta179
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta180' hyperid = 29280
name = theta180
short.name = theta180
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta181' hyperid = 29281
name = theta181
short.name = theta181
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta182' hyperid = 29282
name = theta182
short.name = theta182
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta183' hyperid = 29283
name = theta183
short.name = theta183
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta184' hyperid = 29284
name = theta184
short.name = theta184
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta185' hyperid = 29285
name = theta185
short.name = theta185
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta186' hyperid = 29286
name = theta186
short.name = theta186
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta187' hyperid = 29287
name = theta187
short.name = theta187
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta188' hyperid = 29288
name = theta188
short.name = theta188
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta189' hyperid = 29289
name = theta189
short.name = theta189
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta190' hyperid = 29290
name = theta190
short.name = theta190
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta191' hyperid = 29291
name = theta191
short.name = theta191
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta192' hyperid = 29292
name = theta192
short.name = theta192
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta193' hyperid = 29293
name = theta193
short.name = theta193
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta194' hyperid = 29294
name = theta194
short.name = theta194
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta195' hyperid = 29295
name = theta195
short.name = theta195
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta196' hyperid = 29296
name = theta196
short.name = theta196
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta197' hyperid = 29297
name = theta197
short.name = theta197
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta198' hyperid = 29298
name = theta198
short.name = theta198
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta199' hyperid = 29299
name = theta199
short.name = theta199
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta200' hyperid = 29300
name = theta200
short.name = theta200
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta201' hyperid = 29301
name = theta201
short.name = theta201
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta202' hyperid = 29302
name = theta202
short.name = theta202
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta203' hyperid = 29303
name = theta203
short.name = theta203
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta204' hyperid = 29304
name = theta204
short.name = theta204
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta205' hyperid = 29305
name = theta205
short.name = theta205
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta206' hyperid = 29306
name = theta206
short.name = theta206
initial = 1048576
fixed = FALSE
prior = none
param =

to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta207' hyperid = 29307
name = theta207
short.name = theta207
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta208' hyperid = 29308
name = theta208
short.name = theta208
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta209' hyperid = 29309
name = theta209
short.name = theta209
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta210' hyperid = 29310
name = theta210
short.name = theta210
initial = 1048576
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`
Model '2diid'. 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'

Number of hyperparmeters is 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`

Model 'z'. 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'

Number of hyperparmeters is 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)`

Model 'rw2d'. 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'

Number of hyperparmeters is 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)`

Model 'rw2diid'. 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'

Number of hyperparmeters is 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))`
Model 'slm'. 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'
Number of hyperparmeters is 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))`
Model 'matern2d'. 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'

Number of hyperparmeters is 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)`

Model 'dmatern'. Properties: doc = 'Dense Matern field'
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 = 'dmatern'

Number of hyperparmeters is 3.

Hyperparameter 'theta1' hyperid = 35101
name = log precision
short.name = prec
initial = 3
fixed = FALSE
prior = pc.prec
param = 1 0.01

to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`

Hyperparameter 'theta2' hyperid = 35102
name = log range
short.name = range
initial = 0
fixed = FALSE
prior = pc.range
param = 1 0.5
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`

Hyperparameter 'theta3' hyperid = 35103
name = log nu
short.name = nu
initial = -0.693147180559945
fixed = TRUE
prior = loggamma
param = 0.5 1
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`

Model 'copy'. 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 = 'copy'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 36001
name = beta
short.name = b
initial = 0
fixed = TRUE
prior = normal
param = 1 10
to.theta =

```
function(x, REPLACE.ME.low, REPLACE.ME.high) {                      if (all(is.i
    return(x)                       } else if (all(is.finite(c(low, high)))) {
    stopifnot(low < high) return(log(-(low - x) / (high - x)))                    } else if
    return(log(x - low))                                          } else {
    stop("Condition not yet implemented")                                 }
    }
```

from.theta = `function(x, REPLACE.ME.low, REPLACE.ME.high) { if (all(i`
`return(x) } else if (all(is.finite(c(low, high)))) {`
`stopifnot(low < high) return(low + exp(x) / (1 + exp(x)) * (high - low))`
`return(low + exp(x)) } else {`
`stop("Condition not yet implemented") }`
`}`

Model 'scopy'. Properties: doc = 'Create a scopy 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'
status = 'experimental'
pdf = 'scopy'

Number of hyperparmeters is 15.

Hyperparameter 'theta1' hyperid = 36101
name = beta1
short.name = b1
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta2' hyperid = 36102
name = beta2
short.name = b2
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta3' hyperid = 36103
name = beta3
short.name = b3
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta4' hyperid = 36104

name = beta4
short.name = b4
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta5' hyperid = 36105
name = beta5
short.name = b5
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta6' hyperid = 36106
name = beta6
short.name = b6
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta7' hyperid = 36107
name = beta7
short.name = b7
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta8' hyperid = 36108
name = beta8
short.name = b8
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta9' hyperid = 36109

name = beta9
short.name = b9
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta10' hyperid = 36110
name = beta10
short.name = b10
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta11' hyperid = 36111
name = beta11
short.name = b11
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta12' hyperid = 36112
name = beta12
short.name = b12
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta13' hyperid = 36113
name = beta13
short.name = b13
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta14' hyperid = 36114

name = beta14
short.name = b14
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta15' hyperid = 36115
name = beta15
short.name = b15
initial = 0.1
fixed = FALSE
prior = none
param =
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'clinear'. 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'

Number of hyperparmeters is 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) {                      if (all(is.i
    stopifnot(low < high) return(x)                              } else if (all(is.finite(c(low, hig
    stopifnot(low < high) return(log(-(low - x) / (high - x)))                    } else if
    return(log(x - low))                                              } else {
    stop("Condition not yet implemented")                                    }
    }
```
from.theta =
```
function(x, REPLACE.ME.low, REPLACE.ME.high) {                      if (all(i
    stopifnot(low < high) return(x)                              } else if (all(is.finite(c(low, hig
    stopifnot(low < high) return(low + exp(x) / (1 + exp(x)) * (high - low))
    return(low + exp(x))                                              } else {
    stop("Condition not yet implemented")                                    }
    }
```

Model 'sigm'. 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'
Number of hyperparmeters is 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)`

Model 'revsigm'. 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'

Number of hyperparmeters is 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)`

Model 'log1exp'. 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'

Number of hyperparmeters is 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`

Model 'logdist'. 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'
Number of hyperparmeters is 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)`

'group'

Valid models in this section are:

Model 'exchangeable'. Properties: doc = 'Exchangeable correlations'

Number of hyperparmeters is 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)`

Model 'exchangeablepos'. Properties: doc = 'Exchangeable positive correlations'

Number of hyperparmeters is 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))`

Model 'ar1'. Properties: doc = 'AR(1) correlations'
Number of hyperparmeters is 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`

Model 'ar'. Properties: doc = 'AR(p) correlations'
Number of hyperparmeters is 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`

Model 'rw1'. Properties: doc = 'Random walk of order 1'
Number of hyperparmeters is 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)`

Model 'rw2'. Properties: doc = 'Random walk of order 2'
Number of hyperparmeters is 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)`

Model 'besag'. Properties: doc = 'Besag model'

Number of hyperparmeters is 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)`

Model 'iid'. Properties: doc = 'Independent model'

Number of hyperparmeters is 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)`

'scopy'

Valid models in this section are:

Model 'rw1'. Properties: doc = 'Random walk of order 1'

Number of hyperparmeters is 0.

Model 'rw2'. Properties: doc = 'Random walk of order 2'

Number of hyperparmeters is 0.

'mix'

Valid models in this section are:

Model 'gaussian'. Properties: doc = 'Gaussian mixture'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 47001

name = log precision
short.name = prec
prior = pc.prec
param = 1 0.01
initial = 0
fixed = FALSE
to.theta = `function(x) log(x)`

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

Model 'loggamma'. Properties: doc = 'LogGamma mixture'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 47101

name = log precision

short.name = prec

prior = pc.mgamma

param = 4.8

initial = 4

fixed = FALSE

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

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

Model 'mloggamma'. Properties: doc = 'Minus-LogGamma mixture'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 47201

name = log precision

short.name = prec

prior = pc.mgamma

param = 4.8

initial = 4

fixed = FALSE

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

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

'link'

Valid models in this section are:

Model 'default'. Properties: doc = 'The default link'

Number of hyperparmeters is 0.

Model 'cloglog'. Properties: doc = 'The complementary log-log link'

Number of hyperparmeters is 0.

Model 'ccloglog'. Properties: doc = 'The complement complementary log-log link'

Number of hyperparmeters is 0.

Model 'loglog'. Properties: doc = 'The log-log link'

Number of hyperparmeters is 0.

Model 'identity'. Properties: doc = 'The identity link'

Number of hyperparmeters is 0.

Model 'inverse'. Properties: doc = 'The inverse link'

Number of hyperparmeters is 0.

Model 'log'. Properties: doc = 'The log-link'

Number of hyperparmeters is 0.

Model 'loga'. Properties: doc = 'The loga-link'

Number of hyperparmeters is 0.

Model 'neglog'. Properties: doc = 'The negative log-link'

Number of hyperparmeters is 0.

Model 'logit'. Properties: doc = 'The logit-link'
Number of hyperparmeters is 0.

Model 'probit'. Properties: doc = 'The probit-link'
Number of hyperparmeters is 0.

Model 'cauchit'. Properties: doc = 'The cauchit-link'
Number of hyperparmeters is 0.

Model 'tan'. Properties: doc = 'The tan-link'
Number of hyperparmeters is 0.

Model 'quantile'. Properties: doc = 'The quantile-link'
Number of hyperparmeters is 0.

Model 'pquantile'. Properties: doc = 'The population quantile-link'
Number of hyperparmeters is 0.

Model 'sslogit'. Properties: doc = 'Logit link with sensitivity and specificity'
status = 'disabled'
pdf = 'NA'
Number of hyperparmeters is 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))`

Model 'logoffset'. Properties: doc = 'Log-link with an offset'
pdf = 'logoffset'
Number of hyperparmeters is 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)`

Model 'logitoffset'. Properties: doc = 'Logit-link with an offset'
status = 'experimental'
pdf = 'logitoffset'
Number of hyperparmeters is 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))`

Model 'robit'. Properties: doc = 'Robit link'
status = 'experimental'
pdf = 'robit'
Number of hyperparmeters is 1.
Hyperparameter 'theta' hyperid = 49021
name = log degrees of freedom
short.name = dof
initial = 1.6094379124341
fixed = TRUE
prior = pc.dof
param = 50 0.5
to.theta = `function(x) log(x - 2)`
from.theta = `function(x) 2 + exp(x)`

Model 'sn'. Properties: doc = 'Skew-normal link'
pdf = 'linksn'
Number of hyperparmeters is 2.
Hyperparameter 'theta1' hyperid = 49031
name = skewness
short.name = skew
initial = 0.00123456789
fixed = FALSE
prior = pc.sn
param = 10
to.theta = `function(x, skew.max = 0.988) log((1 + x / skew.max) / (1 - x / skew.max))`
from.theta = `function(x, skew.max = 0.988) skew.max * (2 * exp(x) / (1 + exp(x)) - 1)`
Hyperparameter 'theta2' hyperid = 49032
name = intercept
short.name = intercept
initial = 0

fixed = FALSE
prior = linksnintercept
param = 0 0
to.theta = `function(x) log(x / (1 - x))`
from.theta = `function(x) exp(x) / (1 + exp(x))`

Model 'powerlogit'. Properties: doc = 'Power logit link'
pdf = 'powerlogit'

Number of hyperparmeters is 2.

Hyperparameter 'theta1' hyperid = 49131
name = power
short.name = power
initial = 0.00123456789
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`

Hyperparameter 'theta2' hyperid = 49132
name = intercept
short.name = intercept
initial = 0
fixed = FALSE
prior = logitbeta
param = 1 1
to.theta = `function(x) log(x / (1 - x))`
from.theta = `function(x) exp(x) / (1 + exp(x))`

Model 'test1'. Properties: doc = 'A test1-link function (experimental)'
pdf = 'NA'

Number of hyperparmeters is 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`

Model 'special1'. Properties: doc = 'A special1-link function (experimental)'
pdf = 'NA'

Number of hyperparmeters is 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`

Model 'special2'. Properties: doc = 'A special2-link function (experimental)'
pdf = 'NA'

Number of hyperparmeters is 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`

'predictor'

Valid models in this section are:

Model 'predictor'. Properties: doc = '(do not use)'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 53001
name = log precision
short.name = prec
initial = 13.8155105579643
fixed = TRUE
prior = loggamma
param = 1 1e-05
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`

'hazard'

Valid models in this section are:

Model 'rw1'. Properties: doc = 'A random walk of order 1 for the log-hazard'

Number of hyperparmeters is 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)`

Model 'rw2'. Properties: doc = 'A random walk of order 2 for the log-hazard'

Number of hyperparmeters is 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)`

Model 'iid'. Properties: doc = 'An iid model for the log-hazard'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 55501

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)`

'likelihood'

Valid models in this section are:

Model 'poisson'. Properties: doc = 'The Poisson likelihood'

survival = 'FALSE'

discrete = 'TRUE'

link = 'default log logoffset quantile test1 special1 special2'

pdf = 'poisson'

Number of hyperparmeters is 0.

Model 'xpoisson'. Properties: doc = 'The Poisson likelihood (expert version)'

survival = 'FALSE'

discrete = 'TRUE'

link = 'default log logoffset quantile test1 special1 special2'

pdf = 'poisson'

Number of hyperparmeters is 0.

Model 'cenpoisson'. Properties: doc = 'Then censored Poisson likelihood'

survival = 'FALSE'

discrete = 'TRUE'

link = 'default log logoffset test1 special1 special2'

pdf = 'cenpoisson'

Number of hyperparmeters is 0.

Model 'cenpoisson2'. Properties: doc = 'Then censored Poisson likelihood (version 2)'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default log logoffset test1 special1 special2'
pdf = 'cenpoisson2'
Number of hyperparmeters is 0.

Model 'gpoisson'. Properties: doc = 'The generalized Poisson likelihood'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default log logoffset'
pdf = 'gpoisson'
status = 'experimental'
Number of hyperparmeters is 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`

Model 'poisson.special1'. Properties: doc = 'The Poisson.special1 likelihood'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default log'
pdf = 'poisson-special'
Number of hyperparmeters is 1.
Hyperparameter 'theta' hyperid = 56100
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))`

Model '0poisson'. Properties: doc = 'New 0-inflated Poisson'

status = 'experimental'

survival = 'FALSE'

discrete = 'TRUE'

link = 'default log quantile'

link.simple = 'default logit cauchit probit cloglog ccloglog'

pdf = '0inflated'

Number of hyperparmeters is 10.

Hyperparameter 'theta1' hyperid = 56201

name = beta1

short.name = beta1

initial = -4

fixed = FALSE

prior = normal

param = -4 10

to.theta = `function(x) x`

from.theta = `function(x) x`

Hyperparameter 'theta2' hyperid = 56202

name = beta2

short.name = beta2

initial = 0

fixed = FALSE

prior = normal

param = 0 10

to.theta = `function(x) x`

from.theta = `function(x) x`

Hyperparameter 'theta3' hyperid = 56203

name = beta3

short.name = beta3

initial = 0

fixed = FALSE

prior = normal

param = 0 10

to.theta = `function(x) x`

from.theta = `function(x) x`

Hyperparameter 'theta4' hyperid = 56204

name = beta4

short.name = beta4

initial = 0

fixed = FALSE

prior = normal

param = 0 10

to.theta = `function(x) x`

from.theta = `function(x) x`

Hyperparameter 'theta5' hyperid = 56205
name = beta5
short.name = beta5
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 56206
name = beta6
short.name = beta6
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 56207
name = beta7
short.name = beta7
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta8' hyperid = 56208
name = beta8
short.name = beta8
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta9' hyperid = 56209
name = beta9
short.name = beta9
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta10' hyperid = 56210

name = beta10

short.name = beta10

initial = 0

fixed = FALSE

prior = normal

param = 0 10

to.theta = `function(x) x`

from.theta = `function(x) x`

Model '0poissonS'. Properties: doc = 'New 0-inflated Poisson Swap'

status = 'experimental'

survival = 'FALSE'

discrete = 'TRUE'

link = 'default logit loga cauchit probit cloglog ccloglog loglog log sslogit logitoffset quantile pquantile robit sn powerlogit'

link.simple = 'default log'

pdf = '0inflated'

Number of hyperparmeters is 10.

Hyperparameter 'theta1' hyperid = 56301

name = beta1

short.name = beta1

initial = -4

fixed = FALSE

prior = normal

param = -4 10

to.theta = `function(x) x`

from.theta = `function(x) x`

Hyperparameter 'theta2' hyperid = 56302

name = beta2

short.name = beta2

initial = 0

fixed = FALSE

prior = normal

param = 0 10

to.theta = `function(x) x`

from.theta = `function(x) x`

Hyperparameter 'theta3' hyperid = 56303

name = beta3

short.name = beta3

initial = 0

fixed = FALSE

prior = normal

param = 0 10

to.theta = `function(x) x`

from.theta = `function(x) x`

Hyperparameter 'theta4' hyperid = 56304
name = beta4
short.name = beta4
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta5' hyperid = 56305
name = beta5
short.name = beta5
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 56306
name = beta6
short.name = beta6
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 56307
name = beta7
short.name = beta7
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta8' hyperid = 56308
name = beta8
short.name = beta8
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta9' hyperid = 56309
name = beta9
short.name = beta9
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta10' hyperid = 56310
name = beta10
short.name = beta10
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'bell'. Properties: doc = 'The Bell likelihood'
status = 'experimental'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default log'
pdf = 'bell'
Number of hyperparmeters is 0.

Model '0binomial'. Properties: doc = 'New 0-inflated Binomial'
status = 'experimental'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog log'
link.simple = 'default logit cauchit probit cloglog ccloglog'
pdf = '0inflated'
Number of hyperparmeters is 10.
Hyperparameter 'theta1' hyperid = 56401
name = beta1
short.name = beta1
initial = -4
fixed = FALSE
prior = normal
param = -4 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta2' hyperid = 56402
name = beta2
short.name = beta2

initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta3' hyperid = 56403
name = beta3
short.name = beta3
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta4' hyperid = 56404
name = beta4
short.name = beta4
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta5' hyperid = 56405
name = beta5
short.name = beta5
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 56406
name = beta6
short.name = beta6
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 56407
name = beta7
short.name = beta7

initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta8' hyperid = 56408
name = beta8
short.name = beta8
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta9' hyperid = 56409
name = beta9
short.name = beta9
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta10' hyperid = 56410
name = beta10
short.name = beta10
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Model '0binomialS'. Properties: doc = 'New 0-inflated Binomial Swap'
status = 'experimental'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog log'
link.simple = 'default logit cauchit probit cloglog ccloglog'
pdf = '0inflated'

Number of hyperparmeters is 10.

Hyperparameter 'theta1' hyperid = 56501
name = beta1
short.name = beta1
initial = -4

fixed = FALSE
prior = normal
param = -4 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta2' hyperid = 56502
name = beta2
short.name = beta2
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta3' hyperid = 56503
name = beta3
short.name = beta3
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta4' hyperid = 56504
name = beta4
short.name = beta4
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta5' hyperid = 56505
name = beta5
short.name = beta5
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 56506
name = beta6
short.name = beta6
initial = 0

fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 56507
name = beta7
short.name = beta7
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta8' hyperid = 56508
name = beta8
short.name = beta8
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta9' hyperid = 56509
name = beta9
short.name = beta9
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta10' hyperid = 56510
name = beta10
short.name = beta10
initial = 0
fixed = FALSE
prior = normal
param = 0 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'binomial'. Properties: doc = 'The Binomial likelihood'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog log sslogit logitoffset quantile pquantile robit sn powerlogit'

pdf = 'binomial'

Number of hyperparmeters is 0.

Model 'xbinomial'. Properties: doc = 'The Binomial likelihood (expert version)'

survival = 'FALSE'

discrete = 'TRUE'

link = 'default logit loga cauchit probit cloglog ccloglog loglog log sslogit logitoffset quantile pquantile robit sn powerlogit'

pdf = 'binomial'

status = 'experimental'

Number of hyperparmeters is 0.

Model 'pom'. Properties: doc = 'Likelihood for the proportional odds model'

status = 'experimental'

survival = 'FALSE'

discrete = 'TRUE'

link = 'default identity'

pdf = 'pom'

Number of hyperparmeters is 10.

Hyperparameter 'theta1' hyperid = 57101

name = theta1

short.name = theta1

initial = NA

fixed = FALSE

prior = dirichlet

param = 3

to.theta = `function(x) x`

from.theta = `function(x) x`

Hyperparameter 'theta2' hyperid = 57102

name = theta2

short.name = theta2

initial = NA

fixed = FALSE

prior = none

param =

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

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

Hyperparameter 'theta3' hyperid = 57103

name = theta3

short.name = theta3

initial = NA

fixed = FALSE

prior = none

param =

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

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

Hyperparameter 'theta4' hyperid = 57104

name = theta4
short.name = theta4
initial = NA
fixed = FALSE
prior = none
param =
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`
Hyperparameter 'theta5' hyperid = 57105
name = theta5
short.name = theta5
initial = NA
fixed = FALSE
prior = none
param =
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`
Hyperparameter 'theta6' hyperid = 57106
name = theta6
short.name = theta6
initial = NA
fixed = FALSE
prior = none
param =
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`
Hyperparameter 'theta7' hyperid = 57107
name = theta7
short.name = theta7
initial = NA
fixed = FALSE
prior = none
param =
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`
Hyperparameter 'theta8' hyperid = 57108
name = theta8
short.name = theta8
initial = NA
fixed = FALSE
prior = none
param =
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`
Hyperparameter 'theta9' hyperid = 57109

name = theta9
short.name = theta9
initial = NA
fixed = FALSE
prior = none
param =
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`

Hyperparameter 'theta10' hyperid = 57110
name = theta10
short.name = theta10
initial = NA
fixed = FALSE
prior = none
param =
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`

Model 'bgev'. Properties: doc = 'The blended Generalized Extreme Value likelihood'
status = 'experimental'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity log'
pdf = 'bgev'

Number of hyperparmeters is 12.

Hyperparameter 'theta1' hyperid = 57201
name = spread
short.name = sd
initial = 0
fixed = FALSE
prior = loggamma
param = 1 3
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`

Hyperparameter 'theta2' hyperid = 57202
name = tail
short.name = xi
initial = -4
fixed = FALSE
prior = pc.gevtail
param = 7 0 0.5
to.theta = `function(x, interval = c(REPLACE.ME.low, REPLACE.ME.high)) log(-(interval[1] - x) / (interval[2] - x))`
from.theta = `function(x, interval = c(REPLACE.ME.low, REPLACE.ME.high)) interval[1] + (interval[2] - interval[1]) * exp(x) / (1.0 + exp(x))`

Hyperparameter 'theta3' hyperid = 57203

name = beta1
short.name = beta1
initial = NA
fixed = FALSE
prior = normal
param = 0 300
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta4' hyperid = 57204
name = beta2
short.name = beta2
initial = NA
fixed = FALSE
prior = normal
param = 0 300
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta5' hyperid = 57205
name = beta3
short.name = beta3
initial = NA
fixed = FALSE
prior = normal
param = 0 300
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta6' hyperid = 57206
name = beta4
short.name = beta4
initial = NA
fixed = FALSE
prior = normal
param = 0 300
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta7' hyperid = 57207
name = beta5
short.name = beta5
initial = NA
fixed = FALSE
prior = normal
param = 0 300
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta8' hyperid = 57208

name = beta6
short.name = beta6
initial = NA
fixed = FALSE
prior = normal
param = 0 300
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta9' hyperid = 57209
name = beta7
short.name = beta7
initial = NA
fixed = FALSE
prior = normal
param = 0 300
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta10' hyperid = 57210
name = beta8
short.name = beta8
initial = NA
fixed = FALSE
prior = normal
param = 0 300
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta11' hyperid = 57211
name = beta9
short.name = beta9
initial = NA
fixed = FALSE
prior = normal
param = 0 300
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta12' hyperid = 57212
name = beta10
short.name = beta
initial = NA
fixed = FALSE
prior = normal
param = 0 300
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'gamma'. Properties: doc = 'The Gamma likelihood'

survival = 'FALSE'
discrete = 'FALSE'
link = 'default log quantile'
pdf = 'gamma'

Number of hyperparmeters is 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)`

Model 'gammasurv'. Properties: doc = 'The Gamma likelihood (survival)'
survival = 'TRUE'
discrete = 'FALSE'
status = 'experimental'
link = 'default log neglog quantile'
pdf = 'gammasurv'

Number of hyperparmeters is 11.

Hyperparameter 'theta1' hyperid = 58101
name = precision parameter
short.name = prec
initial = 0
fixed = FALSE
prior = loggamma
param = 1 0.01
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`

Hyperparameter 'theta2' hyperid = 58102
name = beta1
short.name = beta1
initial = -7
fixed = FALSE
prior = normal
param = -4 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta3' hyperid = 58103
name = beta2
short.name = beta2
initial = 0
fixed = FALSE
prior = normal

param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta4' hyperid = 58104
name = beta3
short.name = beta3
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta5' hyperid = 58105
name = beta4
short.name = beta4
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 58106
name = beta5
short.name = beta5
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 58107
name = beta6
short.name = beta6
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta8' hyperid = 58108
name = beta7
short.name = beta7
initial = 0
fixed = FALSE
prior = normal

param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta9' hyperid = 58109
name = beta8
short.name = beta8
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta10' hyperid = 58110
name = beta9
short.name = beta9
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta11' hyperid = 58111
name = beta10
short.name = beta10
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'gammajw'. Properties: doc = 'A special case of the Gamma likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log neglog'
pdf = 'gammajw'
Number of hyperparmeters is 0.

Model 'gammajwsurv'. Properties: doc = 'A special case of the Gamma likelihood (survival)'
survival = 'TRUE'
discrete = 'FALSE'
link = 'default log'
pdf = 'gammajw'
Number of hyperparmeters is 10.

Hyperparameter 'theta1' hyperid = 58200
name = beta1
short.name = beta1

initial = -7
fixed = FALSE
prior = normal
param = -4 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta2' hyperid = 58201
name = beta2
short.name = beta2
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta3' hyperid = 58202
name = beta3
short.name = beta3
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta4' hyperid = 58203
name = beta4
short.name = beta4
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta5' hyperid = 58204
name = beta5
short.name = beta5
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 58205
name = beta6
short.name = beta6

initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 58206
name = beta7
short.name = beta7
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta8' hyperid = 58207
name = beta8
short.name = beta8
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta9' hyperid = 58208
name = beta9
short.name = beta9
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta10' hyperid = 58209
name = beta10
short.name = beta10
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Model 'gammacount'. Properties: doc = 'A Gamma generalisation of the Poisson likelihood'
survival = 'FALSE'
discrete = 'FALSE'

link = 'default log'
status = 'experimental'
pdf = 'gammacount'

Number of hyperparmeters is 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)`

Model 'qkumar'. Properties: doc = 'A quantile version of the Kumar likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit loga cauchit'
pdf = 'qkumar'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 60001
name = precision parameter
short.name = prec
initial = 1
fixed = FALSE
prior = loggamma
param = 1 0.1
to.theta = `function(x, sc = 0.1) log(x) / sc`
from.theta = `function(x, sc = 0.1) exp(sc * x)`

Model 'qloglogistic'. Properties: doc = 'A quantile loglogistic likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log neglog'
pdf = 'qloglogistic'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 60011
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)`

Model 'qloglogisticsurv'. Properties: doc = 'A quantile loglogistic likelihood (survival)'
survival = 'TRUE'

discrete = 'FALSE'
link = 'default log neglog'
pdf = 'qloglogistic'

Number of hyperparmeters is 11.

Hyperparameter 'theta1' hyperid = 60021
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)`

Hyperparameter 'theta2' hyperid = 60022
name = beta1
short.name = beta1
initial = -5
fixed = FALSE
prior = normal
param = -4 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta3' hyperid = 60023
name = beta2
short.name = beta2
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta4' hyperid = 60024
name = beta3
short.name = beta3
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta5' hyperid = 60025
name = beta4
short.name = beta4
initial = 0
fixed = FALSE

prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 60026
name = beta5
short.name = beta5
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 60027
name = beta6
short.name = beta6
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta8' hyperid = 60028
name = beta7
short.name = beta7
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta9' hyperid = 60029
name = beta8
short.name = beta8
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta10' hyperid = 60030
name = beta9
short.name = beta9
initial = 0
fixed = FALSE

prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta11' hyperid = 60031
name = beta10
short.name = beta10
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'beta'. Properties: doc = 'The Beta likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog'
pdf = 'beta'

Number of hyperparmeters is 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)`

Model 'betabinomial'. Properties: doc = 'The Beta-Binomial likelihood'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog robit sn'
pdf = 'betabinomial'

Number of hyperparmeters is 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))`

Model 'betabinomialna'. Properties: doc = 'The Beta-Binomial Normal approximation likelihood'

survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog robit sn'
pdf = 'betabinomialna'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 62101
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))`

Model 'cbinomial'. Properties: doc = 'The clustered Binomial likelihood'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog robit sn'
status = 'experimental'
pdf = 'cbinomial'

Number of hyperparmeters is 0.

Model 'nbinomial'. Properties: doc = 'The negBinomial likelihood'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default log logoffset quantile'
pdf = 'nbinomial'

Number of hyperparmeters is 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)`

Model 'nbinomial2'. Properties: doc = 'The negBinomial2 likelihood'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog'
pdf = 'nbinomial'

Number of hyperparmeters is 0.

Model 'cennbinomial2'. Properties: doc = 'The CenNegBinomial2 likelihood (similar to cenpoisson2)'
status = 'experimental'

survival = 'FALSE'
discrete = 'TRUE'
link = 'default log logoffset quantile'
pdf = 'cennbinomial2'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 63101
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)`

Model 'simplex'. Properties: doc = 'The simplex likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog'
pdf = 'simplex'

Number of hyperparmeters is 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)`

Model 'gaussian'. Properties: doc = 'The Gaussian likelihoood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity logit loga cauchit log logoffset'
pdf = 'gaussian'

Number of hyperparmeters is 2.

Hyperparameter 'theta1' 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)`

Hyperparameter 'theta2' hyperid = 65002

name = log precision offset
short.name = precoffset
initial = 72.0873067782343
fixed = TRUE
prior = none
param =
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`

Model 'gaussianjw'. Properties: doc = 'The GaussianJW likelihoood'
status = 'experimental'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit probit'
pdf = 'gaussianjw'

Number of hyperparmeters is 3.

Hyperparameter 'theta1' hyperid = 65101
name = beta1
short.name = beta1
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta2' hyperid = 65102
name = beta2
short.name = beta2
initial = 1
fixed = FALSE
prior = normal
param = 1 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta3' hyperid = 65103
name = beta3
short.name = beta3
initial = -1
fixed = FALSE
prior = normal
param = -1 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'agaussian'. Properties: doc = 'The aggregated Gaussian likelihoood'
status = 'experimental'
survival = 'FALSE'

discrete = 'FALSE'
link = 'default identity logit loga cauchit log logoffset'
pdf = 'agaussian'
Number of hyperparmeters is 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)`

Model 'circularnormal'. Properties: doc = 'The circular Gaussian likelihoood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default tan'
pdf = 'circular-normal'
status = 'experimental'
Number of hyperparmeters is 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)`

Model 'wrappedcauchy'. Properties: doc = 'The wrapped Cauchy likelihoood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default tan'
pdf = 'wrapped-cauchy'
status = 'disabled'
Number of hyperparmeters is 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))`

Model 'iidgamma'. Properties: doc = '(experimental)'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity'
pdf = 'iidgamma'
status = 'experimental'
Number of hyperparmeters is 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)`
Model 'iidlogitbeta'. Properties: doc = '(experimental)'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit loga'
pdf = 'iidlogitbeta'
status = 'experimental'
Number of hyperparmeters is 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)`

Model 'loggammafrailty'. Properties: doc = '(experimental)'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity'
pdf = 'loggammafrailty'
status = 'experimental'

Number of hyperparmeters is 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)`

Model 'logistic'. Properties: doc = 'The Logistic likelihoood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity'
pdf = 'logistic'

Number of hyperparmeters is 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)`

Model 'sn'. Properties: doc = 'The Skew-Normal likelihoood'
status = 'experimental'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity'
pdf = 'sn'

Number of hyperparmeters is 2.

Hyperparameter 'theta1' hyperid = 74001
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 = 74002
name = logit skew
short.name = skew
initial = 0.00123456789
fixed = FALSE
prior = pc.sn
param = 10
to.theta = `function(x, skew.max = 0.988) log((1 + x / skew.max) / (1 - x / skew.max))`
from.theta = `function(x, skew.max = 0.988) skew.max * (2 * exp(x) / (1 + exp(x)) - 1)`

Model 'gev'. Properties: doc = 'The Generalized Extreme Value likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity'
status = 'disabled: Use likelihood model 'bgev' instead; see inla.doc('bgev')'
pdf = 'gev'

Number of hyperparmeters is 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 = tail parameter
short.name = tail
initial = 0
fixed = FALSE
prior = gaussian
param = 0 25
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'lognormal'. Properties: doc = 'The log-Normal likelihood'
survival = 'FALSE'
discrete = 'FALSE'

link = 'default identity'
pdf = 'lognormal'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 77101
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)`

Model 'lognormalsurv'. Properties: doc = 'The log-Normal likelihood (survival)'
survival = 'TRUE'
discrete = 'FALSE'
link = 'default identity'
pdf = 'lognormal'

Number of hyperparmeters is 11.

Hyperparameter 'theta' hyperid = 78001
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 = 78002
name = beta1
short.name = beta1
initial = -7
fixed = FALSE
prior = normal
param = -4 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta3' hyperid = 78003
name = beta2
short.name = beta2
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta4' hyperid = 78004
name = beta3
short.name = beta3
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta5' hyperid = 78005
name = beta4
short.name = beta4
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 78006
name = beta5
short.name = beta5
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 78007
name = beta6
short.name = beta6
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta8' hyperid = 78008
name = beta7
short.name = beta7
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta9' hyperid = 78009
name = beta8
short.name = beta8
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta10' hyperid = 78010
name = beta9
short.name = beta9
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta11' hyperid = 78011
name = beta10
short.name = beta10
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'exponential'. Properties: doc = 'The Exponential likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'exponential'
Number of hyperparmeters is 0.

Model 'exponentialsurv'. Properties: doc = 'The Exponential likelihood (survival)'
survival = 'TRUE'
discrete = 'FALSE'
link = 'default log neglog'
pdf = 'exponential'
Number of hyperparmeters is 10.
Hyperparameter 'theta1' hyperid = 78020
name = beta1
short.name = beta1
initial = -4
fixed = FALSE
prior = normal

param = -1 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta2' hyperid = 78021
name = beta2
short.name = beta2
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta3' hyperid = 78022
name = beta3
short.name = beta3
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta4' hyperid = 78023
name = beta4
short.name = beta4
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta5' hyperid = 78024
name = beta5
short.name = beta5
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 78025
name = beta6
short.name = beta6
initial = 0
fixed = FALSE
prior = normal

param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 78026
name = beta7
short.name = beta7
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta8' hyperid = 78027
name = beta8
short.name = beta8
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta9' hyperid = 78028
name = beta9
short.name = beta9
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta10' hyperid = 78029
name = beta10
short.name = beta10
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'coxph'. Properties: doc = 'Cox-proportional hazard likelihood'
survival = 'TRUE'
discrete = 'TRUE'
link = 'default log neglog'
pdf = 'coxph'
Number of hyperparmeters is 0.

Model 'weibull'. Properties: doc = 'The Weibull likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log neglog quantile'
pdf = 'weibull'
Number of hyperparmeters is 1.
Hyperparameter 'theta' hyperid = 79001
name = log alpha
short.name = alpha
initial = -2
fixed = FALSE
prior = pc.alphaw
param = 5
to.theta = `function(x, sc = 0.1) log(x) / sc`
from.theta = `function(x, sc = 0.1) exp(sc * x)`

Model 'weibullsurv'. Properties: doc = 'The Weibull likelihood (survival)'
survival = 'TRUE'
discrete = 'FALSE'
link = 'default log neglog quantile'
pdf = 'weibull'
Number of hyperparmeters is 11.
Hyperparameter 'theta' hyperid = 79101
name = log alpha
short.name = alpha
initial = -2
fixed = FALSE
prior = pc.alphaw
param = 5
to.theta = `function(x, sc = 0.1) log(x) / sc`
from.theta = `function(x, sc = 0.1) exp(sc * x)`
Hyperparameter 'theta2' hyperid = 79102
name = beta1
short.name = beta1
initial = -7
fixed = FALSE
prior = normal
param = -4 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta3' hyperid = 79103
name = beta2
short.name = beta2
initial = 0
fixed = FALSE
prior = normal

param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta4' hyperid = 79104
name = beta3
short.name = beta3
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta5' hyperid = 79105
name = beta4
short.name = beta4
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 79106
name = beta5
short.name = beta5
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 79107
name = beta6
short.name = beta6
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta8' hyperid = 79108
name = beta7
short.name = beta7
initial = 0
fixed = FALSE
prior = normal

param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta9' hyperid = 79109
name = beta8
short.name = beta8
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta10' hyperid = 79110
name = beta9
short.name = beta9
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta11' hyperid = 79111
name = beta10
short.name = beta10
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'loglogistic'. Properties: doc = 'The loglogistic likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log neglog'
pdf = 'loglogistic'

Number of hyperparmeters is 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)`

Model 'loglogisticsurv'. Properties: doc = 'The loglogistic likelihood (survival)'
survival = 'TRUE'
discrete = 'FALSE'
link = 'default log neglog'
pdf = 'loglogistic'
Number of hyperparmeters is 11.
Hyperparameter 'theta1' hyperid = 80011
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)`
Hyperparameter 'theta2' hyperid = 80012
name = beta1
short.name = beta1
initial = -5
fixed = FALSE
prior = normal
param = -4 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta3' hyperid = 80013
name = beta2
short.name = beta2
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta4' hyperid = 80014
name = beta3
short.name = beta3
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta5' hyperid = 80015
name = beta4
short.name = beta4

initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 80016
name = beta5
short.name = beta5
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 80017
name = beta6
short.name = beta6
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta8' hyperid = 80018
name = beta7
short.name = beta7
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta9' hyperid = 80019
name = beta8
short.name = beta8
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta10' hyperid = 80020
name = beta9
short.name = beta9

initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta11' hyperid = 80021
name = beta10
short.name = beta10
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'stochvol'. Properties: doc = 'The Gaussian stochvol likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'stochvolgaussian'

Number of hyperparmeters is 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)`

Model 'stochvolsn'. Properties: doc = 'The SkewNormal stochvol likelihood'
status = 'experimental'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'stochvolsn'

Number of hyperparmeters is 2.

Hyperparameter 'theta1' hyperid = 82101
name = logit skew
short.name = skew
initial = 0.00123456789
fixed = FALSE
prior = pc.sn
param = 10
to.theta = `function(x, skew.max = 0.988) log((1 + x / skew.max) / (1 - x / skew.max))`

from.theta = `function(x, skew.max = 0.988) skew.max * (2 * exp(x) / (1 + exp(x)) - 1)`

Hyperparameter 'theta2' hyperid = 82102

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)`

Model 'stochvolt'. Properties: doc = 'The Student-t stochvol likelihood'

survival = 'FALSE'

discrete = 'FALSE'

link = 'default log'

pdf = 'stochvolt'

Number of hyperparmeters is 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)`

Model 'stochvolnig'. Properties: doc = 'The Normal inverse Gaussian stochvol likelihood'

survival = 'FALSE'

discrete = 'FALSE'

link = 'default log'

pdf = 'stochvolnig'

Number of hyperparmeters is 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)`

Model 'zeroinflatedpoisson0'. Properties: doc = 'Zero-inflated Poisson, type 0'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'

Number of hyperparmeters is 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))`

Model 'zeroinflatedpoisson1'. Properties: doc = 'Zero-inflated Poisson, type 1'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'

Number of hyperparmeters is 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))`

Model 'zeroinflatedpoisson2'. Properties: doc = 'Zero-inflated Poisson, type 2'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'

Number of hyperparmeters is 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)`

Model 'zeroinflatedcenpoisson0'. Properties: doc = 'Zero-inflated censored Poisson, type 0'
status = 'experimental'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 87101
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))`

Model 'zeroinflatedcenpoisson1'. Properties: doc = 'Zero-inflated censored Poisson, type 1'
status = 'experimental'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 87201
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))`

Model 'zeroinflatedbetabinomial0'. Properties: doc = 'Zero-inflated Beta-Binomial, type 0'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog robit sn'
pdf = 'zeroinflated'

Number of hyperparmeters is 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))`

Model 'zeroinflatedbetabinomial1'. Properties: doc = 'Zero-inflated Beta-Binomial, type 1'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog robit sn'
pdf = 'zeroinflated'

Number of hyperparmeters is 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))`

Model 'zeroinflatedbinomial0'. Properties: doc = 'Zero-inflated Binomial, type 0'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog robit sn'
pdf = 'zeroinflated'

Number of hyperparmeters is 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))`

Model 'zeroinflatedbinomial1'. Properties: doc = 'Zero-inflated Binomial, type 1'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog robit sn'
pdf = 'zeroinflated'

Number of hyperparmeters is 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))`

Model 'zeroinflatedbinomial2'. Properties: doc = 'Zero-inflated Binomial, type 2'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog robit sn'
pdf = 'zeroinflated'

Number of hyperparmeters is 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)`

Model 'zeroninflatedbinomial2'. Properties: doc = 'Zero and N inflated binomial, type 2'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog robit sn'
pdf = 'NA'

Number of hyperparmeters is 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)`

Model 'zeroninflatedbinomial3'. Properties: doc = 'Zero and N inflated binomial, type 3'
status = 'experimental'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog robit sn'
pdf = 'zeroinflated'
Number of hyperparmeters is 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)`

Model 'zeroinflatedbetabinomial2'. Properties: doc = 'Zero inflated Beta-Binomial, type 2'
survival = 'FALSE'

discrete = 'FALSE'
link = 'default logit loga cauchit probit cloglog ccloglog loglog robit sn'
pdf = 'zeroinflated'

Number of hyperparmeters is 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)`

Model 'zeroinflatednbinomial0'. Properties: doc = 'Zero inflated negBinomial, type 0'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'

Number of hyperparmeters is 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))`

Model 'zeroinflatednbinomial1'. Properties: doc = 'Zero inflated negBinomial, type 1'

survival = 'FALSE'

discrete = 'FALSE'

link = 'default log'

pdf = 'zeroinflated'

Number of hyperparmeters is 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))`

Model 'zeroinflatednbinomial1strata2'. Properties: doc = 'Zero inflated negBinomial, type 1, strata 2'

status = 'experimental'

survival = 'FALSE'

discrete = 'FALSE'

link = 'default log'

pdf = 'zeroinflated'

Number of hyperparmeters is 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))`

Model 'zeroinflatednbinomial1strata3'. Properties: doc = 'Zero inflated negBinomial, type 1, strata 3'
status = 'experimental'

survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'

Number of hyperparmeters is 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)`

Model 'zeroinflatednbinomial2'. Properties: doc = 'Zero inflated negBinomial, type 2'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'zeroinflated'

Number of hyperparmeters is 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)`

Model 't'. Properties: doc = 'Student-t likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity'
pdf = 'student-t'

Number of hyperparmeters is 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)`
Model 'tstrata'. Properties: doc = 'A stratified version of the Student-t likelihood'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity'
pdf = 'tstrata'
Number of hyperparmeters is 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)`

Model 'nmix'. Properties: doc = 'Binomial-Poisson mixture'
status = 'experimental'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit loga probit'
pdf = 'nmix'

Number of hyperparmeters is 15.

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`
Hyperparameter 'theta11' hyperid = 101111
name = beta11
short.name = beta11
initial = 0
fixed = FALSE
prior = normal

param = 0 1
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta12' hyperid = 101112
name = beta12
short.name = beta12
initial = 0
fixed = FALSE
prior = normal
param = 0 1
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta13' hyperid = 101113
name = beta13
short.name = beta13
initial = 0
fixed = FALSE
prior = normal
param = 0 1
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta14' hyperid = 101114
name = beta14
short.name = beta14
initial = 0
fixed = FALSE
prior = normal
param = 0 1
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta15' hyperid = 101115
name = beta15
short.name = beta15
initial = 0
fixed = FALSE
prior = normal
param = 0 1
to.theta = `function(x) x`
from.theta = `function(x) x`

Model 'nmixnb'. Properties: doc = 'NegBinomial-Poisson mixture'
status = 'experimental'
survival = 'FALSE'
discrete = 'TRUE'
link = 'default logit loga probit'
pdf = 'nmixnb'
Number of hyperparmeters is 16.

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 = beta11
short.name = beta11
initial = 0
fixed = FALSE
prior = normal
param = 0 1
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta12' hyperid = 101132
name = beta12
short.name = beta12
initial = 0
fixed = FALSE
prior = normal
param = 0 1
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta13' hyperid = 101133
name = beta13
short.name = beta13
initial = 0
fixed = FALSE
prior = normal
param = 0 1
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta14' hyperid = 101134
name = beta14
short.name = beta14
initial = 0
fixed = FALSE
prior = normal
param = 0 1
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta15' hyperid = 101135
name = beta15
short.name = beta15
initial = 0
fixed = FALSE
prior = normal
param = 0 1
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta16' hyperid = 101136

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)`

Model 'gp'. Properties: doc = 'Generalized Pareto likelihood'

status = 'experimental'

survival = 'FALSE'

discrete = 'TRUE'

link = 'default quantile'

pdf = 'genPareto'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 101201

name = tail

short.name = xi

initial = -4

fixed = FALSE

prior = pc.gevtail

param = 7 0 0.5

to.theta = `function(x, interval = c(REPLACE.ME.low, REPLACE.ME.high)) log(-(interval[1] - x) / (interval[2] - x))`

from.theta = `function(x, interval = c(REPLACE.ME.low, REPLACE.ME.high)) interval[1] + (interval[2] - interval[1]) * exp(x) / (1.0 + exp(x))`

Model 'dgp'. Properties: doc = 'Discrete generalized Pareto likelihood'

status = 'experimental'

survival = 'FALSE'

discrete = 'TRUE'

link = 'default quantile'

pdf = 'dgp'

Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 101301

name = tail

short.name = xi

initial = 2

fixed = FALSE

prior = pc.gevtail

param = 7 0 0.5

to.theta = `function(x, interval = c(REPLACE.ME.low, REPLACE.ME.high)) log(-(interval[1] - x) / (interval[2] - x))`

from.theta = `function(x, interval = c(REPLACE.ME.low, REPLACE.ME.high)) interval[1] + (interval[2] - interval[1]) * exp(x) / (1.0 + exp(x))`

Model 'logperiodogram'. Properties: doc = 'Likelihood for the log-periodogram'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default identity'
pdf = 'NA'
Number of hyperparmeters is 0.

Model 'tweedie'. Properties: doc = 'Tweedie distribution'
status = 'experimental'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'tweedie'
Number of hyperparmeters is 2.

Hyperparameter 'theta1' hyperid = 102101
name = p
short.name = p
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x, interval = c(1.0, 2.0)) log(-(interval[1] - x) / (interval[2] - x))`
from.theta = `function(x, interval = c(1.0, 2.0)) interval[1] + (interval[2] - interval[1]) * exp(x) / (1.0 + exp(x))`

Hyperparameter 'theta2' hyperid = 102201
name = dispersion
short.name = phi
initial = -4
fixed = FALSE
prior = loggamma
param = 100 100
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`

Model 'fmri'. Properties: doc = 'fmri distribution (special nc-chi)'
status = 'experimental'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log'
pdf = 'fmri'
Number of hyperparmeters is 2.

Hyperparameter 'theta1' hyperid = 103101
name = precision
short.name = prec
initial = 0
fixed = FALSE

prior = loggamma
param = 10 10
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`
Hyperparameter 'theta2' hyperid = 103202
name = dof
short.name = df
initial = 4
fixed = TRUE
prior = normal
param = 0 1
to.theta = `function(x) x`
from.theta = `function(x) x`
Model 'fmrisurv'. Properties: doc = 'fmri distribution (special nc-chi)'
status = 'experimental'
survival = 'TRUE'
discrete = 'FALSE'
link = 'default log'
pdf = 'fmri'
Number of hyperparmeters is 2.
Hyperparameter 'theta1' hyperid = 104101
name = precision
short.name = prec
initial = 0
fixed = FALSE
prior = loggamma
param = 10 10
to.theta = `function(x) log(x)`
from.theta = `function(x) exp(x)`
Hyperparameter 'theta2' hyperid = 104201
name = dof
short.name = df
initial = 4
fixed = TRUE
prior = normal
param = 0 1
to.theta = `function(x) x`
from.theta = `function(x) x`
Model 'gompertz'. Properties: doc = 'gompertz distribution'
status = 'experimental'
survival = 'FALSE'
discrete = 'FALSE'
link = 'default log neglog'
pdf = 'gompertz'
Number of hyperparmeters is 1.

Hyperparameter 'theta' hyperid = 105101
name = shape
short.name = alpha
initial = -1
fixed = FALSE
prior = normal
param = 0 1
to.theta = `function(x, sc = 0.1) log(x) / sc`
from.theta = `function(x, sc = 0.1) exp(sc * x)`

Model 'gompertzsurv'. Properties: doc = 'gompertz distribution'
status = 'experimental'
survival = 'TRUE'
discrete = 'FALSE'
link = 'default log neglog'
pdf = 'gompertz'
Number of hyperparmeters is 11.

Hyperparameter 'theta1' hyperid = 106101
name = shape
short.name = alpha
initial = -10
fixed = FALSE
prior = normal
param = 0 1
to.theta = `function(x, sc = 0.1) log(x) / sc`
from.theta = `function(x, sc = 0.1) exp(sc * x)`

Hyperparameter 'theta2' hyperid = 106102
name = beta1
short.name = beta1
initial = -5
fixed = FALSE
prior = normal
param = -4 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta3' hyperid = 106103
name = beta2
short.name = beta2
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta4' hyperid = 106104
name = beta3

short.name = beta3
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta5' hyperid = 106105
name = beta4
short.name = beta4
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 106106
name = beta5
short.name = beta5
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 106107
name = beta6
short.name = beta6
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta8' hyperid = 106108
name = beta7
short.name = beta7
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta9' hyperid = 106109
name = beta8

short.name = beta8
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta10' hyperid = 106110
name = beta9
short.name = beta9
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta11' hyperid = 106111
name = beta10
short.name = beta10
initial = 0
fixed = FALSE
prior = normal
param = 0 100
to.theta = `function(x) x`
from.theta = `function(x) x`

'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 'linksnintercept'. 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 'gamma'. 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.alphaw'. Number of parameters in the prior = 1

Model 'pc.ar'. Number of parameters in the prior = 1

Model 'dirichlet'. 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.sn'. Number of parameters in the prior = 1

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.gevtail'. Number of parameters in the prior = 3

Model 'pc'. Number of parameters in the prior = 2

Model 'ref.ar'. Number of parameters in the prior = 0

Model 'pom'. Number of parameters in the prior = 0

Model 'jeffreystdf'. Number of parameters in the prior = 0

Model 'wishartkd'. Number of parameters in the prior = 211

Model 'expression:'. Number of parameters in the prior = -1

Model 'table:'. Number of parameters in the prior = -1

'wrapper'

Valid models in this section are:

Model 'joint'. 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'

Number of hyperparmeters is 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)`

'lp.scale'

Valid models in this section are:

Model 'lp.scale'. Properties: pdf = 'lp.scale'

Number of hyperparmeters is 100.

Hyperparameter 'theta1' hyperid = 103001

name = beta1

short.name = b1

initial = 1

fixed = FALSE

prior = normal

param = 1 10

to.theta = `function(x) x`

from.theta = `function(x) x`

Hyperparameter 'theta2' hyperid = 103002

name = beta2

short.name = b2

initial = 1

fixed = FALSE

prior = normal

param = 1 10

to.theta = `function(x) x`

from.theta = `function(x) x`

Hyperparameter 'theta3' hyperid = 103003

name = beta3

short.name = b3

initial = 1

fixed = FALSE

prior = normal

param = 1 10

to.theta = `function(x) x`

from.theta = `function(x) x`

Hyperparameter 'theta4' hyperid = 103004

name = beta4

short.name = b4
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta5' hyperid = 103005
name = beta5
short.name = b5
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta6' hyperid = 103006
name = beta6
short.name = b6
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta7' hyperid = 103007
name = beta7
short.name = b7
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta8' hyperid = 103008
name = beta8
short.name = b8
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta9' hyperid = 103009
name = beta9

short.name = b9
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta10' hyperid = 103010
name = beta10
short.name = b10
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta11' hyperid = 103011
name = beta11
short.name = b11
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta12' hyperid = 103012
name = beta12
short.name = b12
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta13' hyperid = 103013
name = beta13
short.name = b13
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta14' hyperid = 103014
name = beta14

short.name = b14
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta15' hyperid = 103015
name = beta15
short.name = b15
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta16' hyperid = 103016
name = beta16
short.name = b16
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta17' hyperid = 103017
name = beta17
short.name = b17
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta18' hyperid = 103018
name = beta18
short.name = b18
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta19' hyperid = 103019
name = beta19

short.name = b19
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta20' hyperid = 103020
name = beta20
short.name = b20
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta21' hyperid = 103021
name = beta21
short.name = b21
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta22' hyperid = 103022
name = beta22
short.name = b22
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta23' hyperid = 103023
name = beta23
short.name = b23
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta24' hyperid = 103024
name = beta24

short.name = b24
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta25' hyperid = 103025
name = beta25
short.name = b25
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta26' hyperid = 103026
name = beta26
short.name = b26
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta27' hyperid = 103027
name = beta27
short.name = b27
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta28' hyperid = 103028
name = beta28
short.name = b28
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta29' hyperid = 103029
name = beta29

short.name = b29
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta30' hyperid = 103030
name = beta30
short.name = b30
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta31' hyperid = 103031
name = beta31
short.name = b31
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta32' hyperid = 103032
name = beta32
short.name = b32
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta33' hyperid = 103033
name = beta33
short.name = b33
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta34' hyperid = 103034
name = beta34

short.name = b34
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta35' hyperid = 103035
name = beta35
short.name = b35
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta36' hyperid = 103036
name = beta36
short.name = b36
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta37' hyperid = 103037
name = beta37
short.name = b37
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta38' hyperid = 103038
name = beta38
short.name = b38
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta39' hyperid = 103039
name = beta39

short.name = b39
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta40' hyperid = 103040
name = beta40
short.name = b40
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta41' hyperid = 103041
name = beta41
short.name = b41
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta42' hyperid = 103042
name = beta42
short.name = b42
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta43' hyperid = 103043
name = beta43
short.name = b43
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta44' hyperid = 103044
name = beta44

short.name = b44
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta45' hyperid = 103045
name = beta45
short.name = b45
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta46' hyperid = 103046
name = beta46
short.name = b46
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta47' hyperid = 103047
name = beta47
short.name = b47
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta48' hyperid = 103048
name = beta48
short.name = b48
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta49' hyperid = 103049
name = beta49

short.name = b49
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta50' hyperid = 103050
name = beta50
short.name = b50
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta51' hyperid = 103051
name = beta51
short.name = b51
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta52' hyperid = 103052
name = beta52
short.name = b52
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta53' hyperid = 103053
name = beta53
short.name = b53
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta54' hyperid = 103054
name = beta54

short.name = b54
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta55' hyperid = 103055
name = beta55
short.name = b55
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta56' hyperid = 103056
name = beta56
short.name = b56
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta57' hyperid = 103057
name = beta57
short.name = b57
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta58' hyperid = 103058
name = beta58
short.name = b58
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta59' hyperid = 103059
name = beta59

short.name = b59
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta60' hyperid = 103060
name = beta60
short.name = b60
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta61' hyperid = 103061
name = beta61
short.name = b61
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta62' hyperid = 103062
name = beta62
short.name = b62
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta63' hyperid = 103063
name = beta63
short.name = b63
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta64' hyperid = 103064
name = beta64

short.name = b64
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta65' hyperid = 103065
name = beta65
short.name = b65
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta66' hyperid = 103066
name = beta66
short.name = b66
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta67' hyperid = 103067
name = beta67
short.name = b67
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta68' hyperid = 103068
name = beta68
short.name = b68
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta69' hyperid = 103069
name = beta69

short.name = b69
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta70' hyperid = 103070
name = beta70
short.name = b70
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta71' hyperid = 103071
name = beta71
short.name = b71
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta72' hyperid = 103072
name = beta72
short.name = b72
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta73' hyperid = 103073
name = beta73
short.name = b73
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta74' hyperid = 103074
name = beta74

short.name = b74
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta75' hyperid = 103075
name = beta75
short.name = b75
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta76' hyperid = 103076
name = beta76
short.name = b76
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta77' hyperid = 103077
name = beta77
short.name = b77
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta78' hyperid = 103078
name = beta78
short.name = b78
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta79' hyperid = 103079
name = beta79

short.name = b79
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta80' hyperid = 103080
name = beta80
short.name = b80
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta81' hyperid = 103081
name = beta81
short.name = b81
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta82' hyperid = 103082
name = beta82
short.name = b82
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta83' hyperid = 103083
name = beta83
short.name = b83
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta84' hyperid = 103084
name = beta84

short.name = b84
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta85' hyperid = 103085
name = beta85
short.name = b85
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta86' hyperid = 103086
name = beta86
short.name = b86
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta87' hyperid = 103087
name = beta87
short.name = b87
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta88' hyperid = 103088
name = beta88
short.name = b88
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta89' hyperid = 103089
name = beta89

short.name = b89
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta90' hyperid = 103090
name = beta90
short.name = b90
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta91' hyperid = 103091
name = beta91
short.name = b91
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta92' hyperid = 103092
name = beta92
short.name = b92
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta93' hyperid = 103093
name = beta93
short.name = b93
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta94' hyperid = 103094
name = beta94

short.name = b94
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta95' hyperid = 103095
name = beta95
short.name = b95
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta96' hyperid = 103096
name = beta96
short.name = b96
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta97' hyperid = 103097
name = beta97
short.name = b97
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta98' hyperid = 103098
name = beta98
short.name = b98
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`
Hyperparameter 'theta99' hyperid = 103099
name = beta99

short.name = b99
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

Hyperparameter 'theta100' hyperid = 103100
name = beta100
short.name = b100
initial = 1
fixed = FALSE
prior = normal
param = 1 10
to.theta = `function(x) x`
from.theta = `function(x) x`

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' takes precedence
## 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.nmix.lambda.fitted`
Estimate posterior distributions of fitted lambda values

Description

For use with `'nmix'` and `'nmixnb'` models. This function takes the information contained in an object returned by `inla()` and uses the contents to create fitted lambda values using the linear predictor for log(lambda), the input covariate values, and samples from the posteriors of the model hyperparameters. Fitted values from the linear predictor are exponentiated, by default, before being returned.

Usage

```
inla.nmix.lambda.fitted(
  result,
  sample.size = 1000,
  return.posteriors = FALSE,
  scale = "exp"
)
```

Arguments

result	The output object from a call to inla(), where the family argument has been set to 'nmix' or 'nmixnb'. For the function to work, the call to inla() should also include the argument control.compute=list(config = TRUE)).
sample.size	The size of the sample from the posteriors of the model hyperparameters. This sample size ends up being the size of the estimated posterior for a fitted lambda value. Default is 1000. Larger values are recommended.
return.posteriors	A logical value for whether or not to return the full estimated posteriors for each fitted value (TRUE), or just a summary of the posteriors (FALSE). Default is FALSE.
scale	A character string, where the default string, "exp", causes values from the linear predictor to be exponentiated before being returned. The string, "log", causes values to be returned on the log(lambda) scale.

Value

fitted.summary	A data frame with summaries of estimated posteriors of fitted lambda values. The number of rows equals the number of rows in the data used to create the 'nmix' or 'nmixnb' model. There are six columns of summary statistics for each estimated posterior. Columns include an index, mean.lambda, sd.lambda, quant025.lambda, median.lambda, quant975.lambda, and mode.lambda.
fitted.posteriors	A data frame containing samples that comprise the full estimated posteriors of fitted values. The number of rows equals the number of rows in the data used to create the 'nmix' or 'nmixnb' model. The number of columns equals one plus the number of samples specified by the sample.size argument.

Note

This function is experimental.

Author(s)

Tim Meehan tmeehan@audubon.org

References

See documentation for families "nmix" and "nmixmb": inla.doc("nmix")

Examples

```
## an example analysis of an N-mixture model using simulated data
## set parameters
n <- 75                       # number of study sites
nrep.max <- 5                 # number of surveys per site
b0 <- 0.5                     # lambda intercept, expected abundance
b1 <- 2.0                     # effect of x1 on lambda
a0 <- 1.0                     # p intercept, detection probability
a2 <- 0.5                     # effect of x2 on p
size <- 3.0                   # size of theta
overdispersion <- 1 / size    # for negative binomial distribution

## make empty vectors and matrix
x1 <- c(); x2 <- c()
lambdas <- c(); Ns <- c()
y <- matrix(NA, n, nrep.max)

## fill vectors and matrix
for(i in 1:n) {
    x1.i <- runif(1) - 0.5
    lambda <- exp(b0 + b1 * x1.i)
    N <- rnbinom(1, mu = lambda, size = size)
    x2.i <- runif(1) - 0.5
    eta <- a0 + a2 * x2.i
    p <- exp(eta) / (exp(eta) + 1)
    nr <- sample(1:nrep.max, 1)
    y[i, 1:nr] <- rbinom(nr, size = N, prob = p)
    x1 <- c(x1, x1.i); x2 <- c(x2, x2.i)
    lambdas <- c(lambdas, lambda); Ns <- c(Ns, N)
}

## bundle counts, lambda intercept, and lambda covariates
Y <- inla.mdata(y, 1, x1)

## run inla and summarize output
result <- inla(Y ~ 1 + x2,
  data = list(Y=Y, x2=x2),
  family = "nmixnb",
  control.fixed = list(mean = 0, mean.intercept = 0, prec = 0.01,
                       prec.intercept = 0.01),
  control.family = list(hyper = list(theta1 = list(param = c(0, 0.01)),
                                     theta2 = list(param = c(0, 0.01)),
                                     theta3 = list(prior = "flat",
                                                   param = numeric()))),
  control.compute=list(config = TRUE)) # important argument
summary(result)

## get and evaluate fitted values
lam.fits <- inla.nmix.lambda.fitted(result, 5000)$fitted.summary
plot(lam.fits$median.lambda, lambdas)
round(sum(lam.fits$median.lambda), 0); sum(Ns)
```

`inla.nonconvex.hull` *Nonconvex set extensions.*

Description

[Deprecated] in favour of `fmesher::fm_nonconvex_hull_inla()` and `fmesher::fm_nonconvex_hull()`.

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.getOption(
 option = c("inla.call", "inla.arg", "fmesher.call", "fmesher.arg", "num.threads",
    "smtp", "safe", "pardiso.license", "keep", "verbose", "save.memory",
    "working.directory", "silent", "debug", "show.warning.graph.file",
    "scale.model.default", "short.summary", "inla.timeout", "fmesher.timeout",
    "inla.mode", "fmesher.evolution", "fmesher.evolution.warn",
    "fmesher.evolution.verbosity")
)

inla.setOption(...)
```

Arguments

option	The option to get. If option = NULL then inla.getOption then inla.getOption will return a named list of current values, 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** Character string with the number of threads to use as A:B, see ?inla **smtp** Sparse matrix library to use, one of band, taucs (default) or pardiso

safe Run in safe-mode (ie try to automatically fix convergence errors) (default TRUE)

pardiso.license The full path to the PARDISO license file or a newline-separated string with license key(s)

keep Keep temporary files?

verbose Verbose output?

save.memory Save memory at the cost of (minor) accuracy and computing time?

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?

cygwin The home of the Cygwin installation (default "C:/cygwin") (Remote computing for Windows only) (No longer in use!)

ssh.auth.sock The ssh bind-adress (value of $SSH_AUTH_SOCK int the Cygwin-shell). (Remote computing for Windows only)

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

short.summary Use a less verbose output for `summary`. Useful for Markdown documents.

inla.timeout The timeout limit, in whole seconds, for calls to the inla binary. Default is 0, meaning no timeout limit. Set to a positive integer to terminate inla calls if they run to long. Fractional seconds are rounded up to the nearest integer. This feature is EXPERIMENTAL and might change at a later stage.

fmesher.timeout The timeout limit, in whole seconds, for calls to the fmesher binary. Default is 0, meaning no timeout limit. Set to a positive integer to terminate fmesher calls that may enter infinite loops due to special geometry regularity. Fractional seconds are rounded up to the nearest integer.

inla.mode Which mode to use in INLA? Default is `"compact"`. Other options are `"classic"` and `"twostage"`.

fmesher.evolution Control use of fmesher methods during the transition to a separate fmesher package. Levels of `fmesher.evolution`:
`1L` uses the intermediate `fm_*` methods in `fmesher` that were already available via `inlabru` from 2.8.0.
`2L` (current default) uses the full range of `fmesher` package methods.
Further levels may be added as the package development progresses.

fmesher.evolution.warn logical; whether to show warnings about deprecated use of legacy INLA methods with fmesher package replacements. When `TRUE`, shows deprecation messages for many CRS and mesh related methods, pointing to their `fm_*` replacements. Default is currently `FALSE`.

fmesher.evolution.verbosity logical or character; at what minimum severity to show warnings about deprecated use of legacy INLA methods with fmesher package replacements. When set to "default" (default), "soft", "warn", or "stop", indicates the minimum warning level used when `fmesher.evolution.warn` is `TRUE`.

`...` Option and value, like `option=value` or `option, value`; see the Examples

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
## set number of threads
inla.setOption("num.threads", "4:1")
## alternative format
inla.setOption(num.threads="4:1")
## check it
inla.getOption("num.threads")
```

---

inla.over_sp_mesh *Check which mesh triangles are inside a polygon*

Description

Wrapper for the sp::over() method to find triangle centroids or vertices inside sp polygon objects. Deprecated since 23.06.06 in favour of inlabru::fm_contains() when inlabru version >= 2.7.0.9011 is installed, and since 23.08.02 in favour of fmesher::fm_contains() when fmesher.

Usage

```
inla.over_sp_mesh(x, y, type = c("centroid", "vertex"), ignore.CRS = FALSE)
```

Arguments

x	geometry (typically a sp::SpatialPolygons() object) for the queries
y	an inla.mesh() object
type	the query type; either 'centroid' (default, for triangle centroids), or 'vertex' (for mesh vertices)
ignore.CRS	logical; whether to ignore the coordinate system information in x and y (default FALSE)

Value

A vector of triangle indices (when type is 'centroid') or vertex indices (when type is 'vertex')

Author(s)

Haakon Bakka, <bakka@r-inla.org>, and Finn Lindgren <finn.lindgren@gmail.com>

Examples

```
# Create a polygon and a mesh
obj <- sp::SpatialPolygons(
    list(Polygons(
        list(Polygon(rbind(
            c(0, 0),
            c(50, 0),
            c(50, 50),
            c(0, 50)
        ))),
        ID = 1
    )),
    proj4string = fmesher::fm_CRS("longlat_globe")
)
mesh <- inla.mesh.create(globe = 2, crs = fmesher::fm_CRS("sphere"))

## 3 vertices found in the polygon
inla.over_sp_mesh(obj, mesh, type = "vertex")

## 3 triangles found in the polygon
inla.over_sp_mesh(obj, mesh)

## Multiple transformations can lead to slightly different results due to edge cases
## 4 triangles found in the polygon
inla.over_sp_mesh(
    obj,
    fmesher::fm_transform(mesh, crs = fmesher::fm_crs("mollweide_norm"))
)
```

`inla.priors.used` *Print priors used*

Description

Print the priors used for the hyperparameters

Usage

```
inla.priors.used(result, digits = 6L)
```

Arguments

`result`	An `inla`-object, typically the output from an `inla()`-call
`digits`	The `digits` argument to the function `format()`

Details

This function provides a more human-friendly output of `result$all.hyper` of all the priors used for the hyperparameters. Since not all information about the model is encoded in this object, more hyperparameters than actually used, may be printed. In particular, `group.theta1` is printed even though the argument `group` in `f()` is not used. Similarly for spde-models, but the user should know that, for example, only the two first ones are actually used. Hopefully, this issue will be fixed in the future.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
r = inla(y ~ 1 + x, data = data.frame(y = 1:10, x = rep(1:5, 2)))
inla.priors.used(r)
```

---

inla.prune *Prune the INLA-package*

Description

Prune the INLA-package by deleting binary files not supported by the running OS

Usage

```
inla.prune(ask = TRUE)
```

Arguments

ask	Logical. If TRUE, then ask for user confirmation before deleting. If FALSE, then delete without user confirmation.

Value

No value is returned.

Author(s)

Havard Rue <hrue@r-inla.org>

---

inla.qstat *Control and view a remote inla-queue*

Description

Control and view a remote inla-queue of submitted jobs

`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.

Usage

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

## S3 method for class 'inla.q'
print(x, ...)

inla.qget(id, remove = TRUE)

inla.qdel(id)

inla.qstat(id)

inla.qlog(id)

inla.qnuke()
```

Arguments

object	An inla.q-object which is the output from inla.qstat
...	other arguments.
x	An inla.q-object which is the output from inla.qstat
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).

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()
```

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, and functions thereof, from an approximated posterior of a fitted model*

Description

This function generate samples, and functions of those, from an approximated posterior of a fitted model (an inla-object)

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), and optional (and by default) corrected for the estimated skewness.

The log.density report is only correct when there is no constraints. With constraints, it correct the Gaussian part of the sample for the constraints.

After the sample is (optional) skewness corrected, the log.density is is not exact for correcting for constraints, but the error is very small in most cases.

Usage

```
inla.posterior.sample(
  n = 1L,
  result,
  selection = list(),
  intern = FALSE,
  use.improved.mean = TRUE,
  skew.corr = TRUE,
  add.names = TRUE,
  seed = 0L,
  num.threads = NULL,
  parallel.configs = TRUE,
  verbose = FALSE
)

inla.posterior.sample.eval(fun, samples, return.matrix = TRUE, ...)
```

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)`.
`selection`	Select what part of the sample to return. By default, the whole sample is returned. `selection` is a named list with the name of the components of the sample, and what indices of them to return. Names include `APredictor`, `Predictor`, `(Intercept)`, and otherwise names in the formula. The values of the list, is interpreted as indices. If they are negative, they are interpreted as 'not', a zero is interpreted as 'all', and positive indices are interpreted as 'only'. The names of elements of each samples refer to the indices in the full sample. DO NOT USE this feature together with `inla.posterior.sample.eval`.

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))

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.

skew.corr — Logical. If TRUE then correct samples for skewness, if FALSE, do not correct samples for skewness (ie use the Gaussian).

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 — See the same argument in ?inla.qsample for further information. In order to produce reproducible results, you ALSO need to make sure the RNG in R is in the same state, see example below. When seed is non-zero, num.threads is forced to "1:1" and parallel.configs is set to FALSE, since parallel sampling would not produce a reproducible sequence of pseudo-random numbers.

num.threads — The number of threads to use in the format 'A:B' defining the number threads in the outer (A) and inner (B) layer for nested parallelism. A '0' will be replaced intelligently. seed!=0 requires serial comptuations.

parallel.configs — Logical. If TRUE and not on Windows, then try to run each configuration in parallel (not Windows) using A threads (see num.threads), where each of them is using B:0 threads.

verbose — Logical. Run in verbose mode or not.

fun — The function to evaluate for each sample. Upon entry, the variable names defined in the model are defined as the value of the sample. The list of names are defined in result$misc$configs$contents where result is an inla-object. This includes predefined names for for the linear predictor (Predictor and APredictor), and the intercept ((Intercept) or Intercept). The hyperparameters are defined as theta, no matter if they are in the internal scale or not. The function fun can also return a vector. To simplify usage, fun can also be a vector character's. In this case fun it is interpreted as (strict) variable names, and a function is created that return these variables: if argument fun equals c("Intercept", "a[1:2]"), then this is equivalent to pass function() return(c(get('Intercept'), get('a[1:2]'))).

samples — samples is the output from inla.posterior.sample()

return.matrix — Logical. If TRUE, then return the samples of fun as matrix, otherwise, as a list.

... — Additional arguments to fun

Value

inla.posterior.sample returns 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. inla.posterior.sample.eval return a list or a matrix of fun applied to each sample.

Author(s)

Havard Rue <hrue@r-inla.org> and Cristian Chiuchiolo <cristian.chiuchiolo@kaust.edu.sa>

Examples

```
 r = inla(y ~ 1 ,data = data.frame(y=rnorm(1)), control.compute = list(config=TRUE))
 samples = inla.posterior.sample(2,r)

 ## reproducible results:
 inla.seed = as.integer(runif(1)*.Machine$integer.max)
 set.seed(12345)
 x = inla.posterior.sample(10, r, seed = inla.seed, num.threads="1:1")
 set.seed(12345)
 xx = inla.posterior.sample(10, r, seed = inla.seed, num.threads="1.1")
 all.equal(x, xx)

set.seed(1234)
n = 25
xx = rnorm(n)
yy = rev(xx)
z = runif(n)
y = rnorm(n)
r = inla(y ~ 1 + z + f(xx) + f(yy, copy="xx"),
        data = data.frame(y, z, xx, yy),
        control.compute = list(config=TRUE),
        family = "gaussian")
r.samples = inla.posterior.sample(10, r)

fun = function(...) {
    mean(xx) - mean(yy)
}
f1 = inla.posterior.sample.eval(fun, r.samples)

fun = function(...) {
    c(exp(Intercept), exp(Intercept + z))
}
f2 = inla.posterior.sample.eval(fun, r.samples)

fun = function(...) {
    return (theta[1]/(theta[1] + theta[2]))
}
f3 = inla.posterior.sample.eval(fun, r.samples)

## Predicting nz new observations, and
## comparing the estimated one with the true one
set.seed(1234)
n = 100
alpha = beta = s = 1
z = rnorm(n)
y = alpha + beta * z + rnorm(n, sd = s)
r = inla(y ~ 1 + z,
        data = data.frame(y, z),
        control.compute = list(config=TRUE),
        family = "gaussian")
r.samples = inla.posterior.sample(10^3, r)

## just return samples of the intercept
intercepts = inla.posterior.sample.eval("Intercept", r.samples)

nz = 3
```

```
znew = rnorm(nz)
fun = function(zz = NA) {
    ## theta[1] is the precision
    return (Intercept + z * zz +
            rnorm(length(zz), sd = sqrt(1/theta[1])))
}
par(mfrow=c(1, nz))
f1 = inla.posterior.sample.eval(fun, r.samples, zz = znew)
for(i in 1:nz) {
    hist(f1[i, ], n = 100, prob = TRUE)
    m = alpha + beta * znew[i]
    xx = seq(m-4*s, m+4*s, by = s/100)
    lines(xx, dnorm(xx, mean=m, sd = s), lwd=2)
}

##
## Be aware that using non-clean variable names might be a little tricky
##
n <- 100
X <- matrix(rnorm(n^2), n, 2)
x <- X[, 1]
xx <- X[, 2]
xxx <- x*xx

y <- 1 + 2*x + 3*xx + 4*xxx + rnorm(n, sd = 0.01)

r <- inla(y ~ X[, 1]*X[, 2],
          data = list(y = y, X = X),
          control.compute = list(config = TRUE))
print(round(dig = 4, r$summary.fixed[,"mean"]))

sam <- inla.posterior.sample(100, r)
sam.extract <- inla.posterior.sample.eval(
    (function(...) {
        beta.1 <- get("X[, 1]")
        beta.2 <- get("X[, 2]")
        beta.12 <- get("X[, 1]:X[, 2]")
        return(c(Intercept, beta.1, beta.2, beta.12))
    }), sam)
print(round(dig = 4, rowMeans(sam.extract)))

## a simpler form can also be used here, and in the examples below
sam.extract <- inla.posterior.sample.eval(
               c("Intercept", "X[, 1]", "X[, 2]", "X[, 1]:X[, 2]"), sam)
print(round(dig = 4, rowMeans(sam.extract)))

r <- inla(y ~ x + xx + xxx,
          data = list(y = y, x = x, xx = xx, xxx = xxx),
          control.compute = list(config = TRUE))

sam <- inla.posterior.sample(100, r)
sam.extract <- inla.posterior.sample.eval(
    (function(...) {
        return(c(Intercept, x, xx, xxx))
    }), sam)
print(round(dig = 4, rowMeans(sam.extract)))
```

```
sam.extract <- inla.posterior.sample.eval(c("Intercept", "x", "xx", "xxx"), sam)
print(round(dig = 4, rowMeans(sam.extract)))

r <- inla(y ~ x*xx,
          data = list(y = y, x = x, xx = xx),
          control.compute = list(config = TRUE))

sam <- inla.posterior.sample(100, r)
sam.extract <- inla.posterior.sample.eval(
    (function(...) {
        return(c(Intercept, x, xx, get("x:xx")))
    }), sam)
print(round(dig = 4, rowMeans(sam.extract)))

sam.extract <- inla.posterior.sample.eval(c("Intercept", "x", "xx", "x:xx"), sam)
print(round(dig = 4, rowMeans(sam.extract)))
```

`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.out = 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

```
## Not run:
## 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))
}

## End(Not run)
```

`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(...)

inla.spde1.precision(spde, ...)

## S3 method for class 'inla.spde1'
inla.spde.precision(spde, ...)

inla.spde2.precision(
  spde,
  theta = NULL,
  phi0 = inla.spde2.theta2phi0(spde, theta),
  phi1 = inla.spde2.theta2phi1(spde, theta),
  phi2 = inla.spde2.theta2phi2(spde, theta),
  ...
)

## 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),
  ...
)
```

Arguments

`...`	Additional parameters passed on to other methods.
`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.

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(...)

inla.spde1.result(inla, name, spde, do.transform = TRUE, ...)

## S3 method for class 'inla.spde1'
inla.spde.result(inla, name, spde, do.transform = TRUE, ...)

inla.spde2.result(inla, name, spde, do.transform = TRUE, ...)

## S3 method for class 'inla.spde2'
inla.spde.result(inla, name, spde, do.transform = TRUE, ...)
```

Arguments

...	Further arguments passed to and from other methods.
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.

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

`...`	Parameters passed on to other methods.
`precision`	A precision matrix.
`seed`	The seed for the pseudo-random generator.
`spde`	An `inla.spde` object.

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

```
inla.spde1.create(
  mesh,
  model = c("matern", "imatern", "matern.osc"),
  param = NULL,
  ...
)

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.spde1.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))),
  ...
)
```

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.
`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,
  ...
)

inla.spde2.theta2phi0(spde, theta)

inla.spde2.theta2phi1(spde, theta)

inla.spde2.theta2phi2(spde, theta)
```

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.
spde	An spde model object
theta	Parameters in the model's internal scale

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.

Functions

- `inla.spde2.theta2phi0()`: Convert from theta vector to phi0 values in the internal spde2 model representation
- `inla.spde2.theta2phi1()`: Convert from theta vector to phi1 values in the internal spde2 model representation
- `inla.spde2.theta2phi2()`: Convert from theta vector to phi2 values in the internal spde2 model representation

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* `fmesher::fm_transform`

Description

[Deprecated] in favour of `fmesher::fm_transform()`. Handles transformation of various inla objects according to coordinate reference systems of `sf::crs`, `sp::CRS` or `inla.CRS` class.

Usage

```
inla.spTransform(x, CRSobj, ...)
```

Arguments

`x`	The object that should be transformed from it's current CRS to a new CRS
`CRSobj`	passed on as the `crs` argument to `fmesher::fm_transform()`.
`...`	Potential other arguments for `fmesher::fm_transform()`.

Value

The object is returned with its coordinates transformed

Author(s)

Finn Lindgren finn.lindgren@gmail.com

See Also

`inla.CRS()`

Examples

```
if (require("sf") && require("sp") && require("fmesher")) {
    latt <- inla.mesh.lattice(-10:10, 40:60)
    mesh1 <- inla.mesh.create(
        lattice = latt, extend = FALSE, refine = FALSE,
        crs = fm_CRS("longlat_norm")
    )
    mesh2 <- fm_transform(mesh1, fm_crs("lambert_globe"))
    print(summary(mesh1))
    print(summary(mesh2))
}
```

---

inla.sp_get_crs *Extract CRS information*

Description

[Deprecated] in favour of `fmesher::fm_CRS()`. Wrapper for CRS(projargs) (PROJ4) and CRS(wkt) for `sp::Spatial` objects.

This function is a convenience method to workaround PROJ4/PROJ6 differences, and the lack of a crs extraction method for Spatial objects.

Usage

```
inla.sp_get_crs(x)
```

Arguments

`x`	A `sp::Spatial` object

Value

A `CRS` object, or NULL if no valid CRS identified

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

Examples

```
## Not run:
if (interactive()) {
    s <- sp::SpatialPoints(matrix(1:6, 3, 2), proj4string = fmesher::fm_CRS("sphere"))
    inla.sp_get_crs(s)
}

## End(Not run)
```

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.ssh.copy.id()

inla.remote()
```

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.remove.unused *Data stacking for advanced INLA models*

Description

Functions for combining data, effects and observation matrices into `inla.stack` objects, and extracting information from such objects.

Usage

```
inla.stack.remove.unused(stack)

inla.stack.compress(stack, remove.unused = TRUE)

inla.stack(..., compress = TRUE, remove.unused = TRUE)

inla.stack.sum(
  data,
  A,
  effects,
  tag = "",
```

```
  compress = TRUE,
  remove.unused = TRUE
)

inla.stack.join(..., compress = TRUE, remove.unused = TRUE)

inla.stack.index(stack, tag)

inla.stack.LHS(stack)

inla.stack.RHS(stack)

inla.stack.data(stack, ...)

inla.stack.A(stack)
```

Arguments

stack	A inla.data.stack object, created by a call to inla.stack, inla.stack.sum, or inla.stack.join.
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).
...	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.
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.
data	A list or codedata.frame of named data vectors. Scalars are expanded to match the number of rows in the A matrices, or any non-scalar data vectors. An error is given if the input is inconsistent.
A	A list of observation matrices. Scalars are expanded to diagonal matrices matching the effect vector lengths. An error is given if the input is inconsistent or ambiguous.
effects	A collection of effects/predictors. Each list element corresponds to an observation matrix, and must either be a single vector, a list of vectors, or a data.frame. Single-element effect vectors are expanded to vectors matching the number of columns in the corresponding A matrix. An error is given if the input is inconsistent or ombiguous.
tag	A string specifying a tag for later identification.

Details

For models with a single effects collection, the outer list container for A and effects may be omitted.

Component size definitions:

- 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

$$\tilde{A} = \text{cbind}\left(A^1, \ldots, A^{n_l}\right)$$

and

$$\tilde{x}^k = \text{rbind}\left(x^{k,1}, \ldots, x^{k,n_l}\right)$$

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 `tags`, each element indexing into $i = 1, \ldots, n_i$

Functions

- `inla.stack.remove.unused()`: Remove unused entries from an existing stack
- `inla.stack.compress()`: Compress an existing stack by removing duplicates
- `inla.stack.sum()`: Create data stack as a sum of predictors
- `inla.stack.join()`: Join two or more data stacks
- `inla.stack.index()`: Extract tagged indices
- `inla.stack.LHS()`: Extract data associated with the "left hand side" of the model (e.g. the data itself, `Ntrials`, `link`, `E`)
- `inla.stack.RHS()`: Extract data associated with the "right hand side" of the model (all the covariates/predictors)
- `inla.stack.data()`: Extract data for an inla call, and optionally join with other variables
- `inla.stack.A()`: Extract the "A matrix" for control.predictor

Functions

- `inla.stack.remove.unused`: Remove unused entries from an existing stack
- `inla.stack.compress`: Compress an existing stack by removing duplicates
- `inla.stack`: Shorthand for inla.stack.join and inla.stack.sum
- `inla.stack.sum`: Create data stack as a sum of predictors
- `inla.stack.join`: Join two or more data stacks
- `inla.stack.index`: Extract tagged indices
- `inla.stack.LHS`: Extract data associated with the "left hand side" of the model (e.g. the data itself, `Ntrials`, `link`, `E`)
- `inla.stack.RHS`: Extract data associated with the "right hand side" of the model (all the covariates/predictors)
- `inla.stack.data`: Extract data for an inla call, and optionally join with other variables
- `inla.stack.A`: Extract the "A matrix" for control.predictor

See Also

`inla.spde.make.A()`, `inla.spde.make.index()`

Examples

```
n <- 200
loc <- matrix(runif(n * 2), n, 2)
mesh <- inla.mesh.2d(
    loc.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.pcmatern(mesh,
    prior.range = c(0.01, 0.01),
    prior.sigma = c(10, 0.01)
)

covar <- rnorm(n)
field <- inla.qsample(n = 1, Q = inla.spde.precision(spde, theta = log(c(0.5, 1))))[, 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$loc)
stack.obs <-
    inla.stack(
        data = list(y = y),
        A = list(A.obs, 1),
        effects = list(c(
            list(Intercept = 1),
            inla.spde.make.index("spatial", spde$n.spde)
        ),
        covar = covar
        ),
        tag = "obs"
    )
stack.pred <-
```

```
    inla.stack(
        data = list(y = NA),
        A = list(A.pred),
        effects = list(c(
            list(Intercept = 1),
            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
    )
)

plot(y, result1$summary.fitted.values[inla.stack.index(stack.obs, "obs")$data, "mean"],
    main = "Observations vs posterior predicted values at the data locations"
)

result2 <- inla(formula,
    data = inla.stack.data(stack, spde = spde),
    family = "gaussian",
    control.predictor = list(
        A = inla.stack.A(stack),
        compute = TRUE
    )
)

field.pred <- inla.mesh.project(
    proj.pred,
    result2$summary.fitted.values[inla.stack.index(stack, "pred")$data, "mean"]
)
field.pred.sd <- inla.mesh.project(
    proj.pred,
    result2$summary.fitted.values[inla.stack.index(stack, "pred")$data, "sd"]
)

plot(field, field.pred, main = "True vs predicted field")
abline(0, 1)
image(inla.mesh.project(mesh,
    field = field,
    dims = c(200, 200)
),
main = "True field"
)
image(inla.mesh.project(mesh,
    field = field.pred,
    dims = c(200, 200)
),
main = "Posterior field mean"
)
```

```
image(inla.mesh.project(mesh,
    field = field.pred.sd,
    dims = c(200, 200)
),
main = "Prediction standard deviation"
)
plot(field, (field.pred - field) / 1,
    main = "True field vs standardised prediction residuals"
)
```

`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 = NULL,
  cure = NULL,
  .special = NULL
)

## S3 method for class 'inla.surv'
plot(x, y, ...)

## S3 method for class 'inla.surv'
print(x, ...)

as.inla.surv(object, ...)

is.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. For in-interval event, this is the observed time (in the interval) for the event. For left censored data, this the censoring time.
`event`	The status indicator, 1=observed event, 0=right censored event, 2=left censored event, 3=interval censored event, and 4=observed event in an interval (left, right).
`time2`	Ending time for the interval censored data or an in-interval event.
`truncation`	Left truncation. If missing it is assumed to be 0. The lower limit for event=4.
`subject`	Patient number in multiple event data, not needed otherwise.

cure	A matrix of covariates that can be used with a cure-model.
.special	An internal object, not for public use
x	Object to plot or print
y	Object to plot (not in use)
...	Additional argument
object	Any R-object

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, Rupali Akerkar and Haavard Rue

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.update *Upgrade the INLA-package*

Description

Functions to upgrade the INLA-package to the current version.

Usage

```
inla.update(lib = NULL, testing = FALSE, ask = TRUE)

inla.upgrade(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", "date"))
```

Arguments

what	What to show version of

Value

inla.version display the current version information using cat with default or info, or return other specific requests through the call.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
## Summary of all
inla.version()
## The building date
inla.version("date")
```

---

joint.marginal *Sample, transform and evaluate from a joint marginal approximation*

Description

Sample, transform and evaluate from from a joint marginal approximation as returned using argument selection in inla.

Usage

```
inla.rjmarginal(n, jmarginal, constr)

inla.rjmarginal.eval(fun, samples, ...)

## S3 method for class 'inla.jmarginal'
print(x, ...)

## S3 method for class 'inla.jmarginal'
summary(object, ...)

## S3 method for class 'summary.inla.jmarginal'
print(x, ...)

inla.tjmarginal(jmarginal, A)

inla.1djmarginal(jmarginal)
```

Arguments

n	The number of samples
jmarginal	A marginal object given either by a inla object or result$selection
constr	Optional linear constraints; see ?INLA::f and argument extraconstr
fun	A function which is evaluated for each sample, similar to inla.posterior.sample.eval: please see the documentation for this functions for details.

samples	The samples, as in the form of the output from inla.rjmarginal
...	Arguments passed on to other methods (printing and summarising)
x	Object to be printed
object	Object to be summarised
A	A matrix used for the linear combination

Value

THESE FUNCTIONS ARE EXPERIMENTAL FOR THE MOMENT (JULY 2020)

inla.rjmarginal returns a list with the samples in samples (matrix) and the corresponding log-densities in log.density (vector). Each column in samples contains one sample.

inla.rjmarginal.eval returns a matrix, where each row is the (vector) function evaluated at each sample.

inla.tjmarginal returns a inla.jmarginal-object of the linear combination defined by the matrix A.

inla.1djmarginal return the marginal densities from a joint approximation.

Author(s)

Cristian Chiuchiolo and Havard Rue <hrue@r-inla.org>

See Also

inla()

Examples

```
n = 10
x = 1+rnorm(n)
xx = 3 + rnorm(n)
y = 1 + x + xx + rnorm(n)
selection = list(xx=1, x=1)
r = inla(y ~ 1 + x + xx,
         data = data.frame(y, x, xx),
         selection = selection)
ns = 100
xx = inla.rjmarginal(ns, r)

print(cbind(mean = r$selection$mean, sample.mean = rowMeans(xx$samples)))
print("cov matrix")
print(round(r$selection$cov.matrix, dig=3))
print("sample cov matrix")
print(round(cov(t(xx$samples)), dig=3))

skew = function(z) mean((z-mean(z))^3)/var(z)^1.5
print(round(cbind(skew = r$selection$skewness,
                  sample.skew = apply(xx$samples, 1, skew)), digits = 3))

## illustrating the eval function
n = 10
x = rnorm(n)
eta = 1 + x
y = eta + rnorm(n, sd=0.1)
```

```
selection = list(x = 1, '(Intercept)' = 1)
r = inla(y ~ 1 + x,
         data = data.frame(y, x),
         selection = selection)
xx = inla.rjmarginal(100,  r)
xx.eval = inla.rjmarginal.eval(function() c(x, Intercept),  xx)
print(cbind(xx$samples[, 1]))
print(cbind(xx.eval[, 1]))

constr <- list(A = matrix(1, ncol = nrow(xx$samples), nrow = 1), e = 1)
x <- inla.rjmarginal(10, r, constr = constr)

A <- matrix(rnorm(nrow(xx$samples)^2), nrow(xx$samples), nrow(xx$samples))
b <- inla.tjmarginal(r, A)
b.marg <- inla.1djmarginal(b)
```

---

jp *Joint-prior models*

Description

A framework for defining joint priors in R

Usage

```
inla.jp.define(jp = NULL, ...)
```

Arguments

jp	The jp-function which returns the joint log-prior as a function of argument theta. There is an optional second argument that is a vector of theta-names. If second argument is not present, argument .theta.desc will be added.
...	Named list of variables that defines the environment of jp

Value

This allows joint priors to be defined in R.

This function is for internal use only.

Author(s)

Havard Rue <hrue@r-inla.org>

Kidney *Kidney infection data*

Description

Times of infection from the time to insertion of the catheter for 38 kindey patients using portable dialysis equipment

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`.

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

Details

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.

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.

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.cauchit(x, inverse = FALSE)

inla.link.invcauchit(x, inverse = FALSE)

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.robit(x, df = 7, inverse = FALSE)

inla.link.invrobit(x, df = 7, inverse = FALSE)

inla.link.loglog(x, inverse = FALSE)

inla.link.invloglog(x, inverse = FALSE)

inla.link.cloglog(x, inverse = FALSE)

inla.link.invcloglog(x, inverse = FALSE)

inla.link.ccloglog(x, inverse = FALSE)

inla.link.invccloglog(x, inverse = FALSE)

inla.link.tan(x, inverse = FALSE)

inla.link.invtan(x, inverse = FALSE)

inla.link.identity(x, inverse = FALSE)
```

```
inla.link.invidentity(x, inverse = FALSE)

inla.link.inverse(x, inverse = FALSE)

inla.link.invinverse(x, inverse = FALSE)

inla.link.invqpoisson(x, inverse = FALSE, quantile = 0.5)

inla.link.sn(x, intercept = 0.5, skew = 0, a = NULL, inverse = FALSE)

inla.link.invsn(x, intercept = 0.5, skew = 0, a = NULL, 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)
df	The degrees of freedom for the Student-t
quantile	The quantile level for quantile links
intercept	The quantile level for the intercept in the Skew-Normal link
skew	The skewness in the Skew-Normal. Only one of skew and a can be given.
a	The a-parameter in the Skew-Normal. Only one of skew and a can be given.

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.smarginal(
  marginal,
  log = FALSE,
  extrapolate = 0,
  keep.type = FALSE,
  factor = 15L
)

inla.emarginal(fun, marginal, ...)

inla.dmarginal(x, marginal, log = FALSE)

inla.pmarginal(q, marginal, normalize = TRUE, len = 2048L)

inla.qmarginal(p, marginal, len = 2048L)
```

```
inla.hpdmarginal(p, marginal, len = 2048L)

inla.rmarginal(n, marginal)

inla.tmarginal(
  fun,
  marginal,
  n = 2048L,
  h.diff = .Machine[["double.eps"]]^(1/3),
  method = c("quantile", "linear")
)

inla.mmarginal(marginal)

inla.zmarginal(marginal, silent = FALSE)

inla.is.marginal(marginal)
```

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. Theinla.hpdmarginal()-function assumes a unimodal density.
log	Return density or interpolated density in log-scale?
extrapolate	How much to extrapolate on each side when computing the interpolation. In fraction of the range.
keep.type	If FALSE then return a list(x=, y=), otherwise if TRUE, then return a matrix if the input is a matrix
factor	The number of points after interpolation is factor times the original number of points; which is argument n in spline
fun	A (vectorised) function like function(x) exp(x) to compute the expectation against, or which define the transformation new = fun(old)
...	Further arguments to be passed to function which expectation is to be computed.
x	Evaluation points
q	Quantiles
normalize	Renormalise the density after interpolation?
len	Number of locations used to interpolate the distribution function.
p	Probabilities
n	The number of observations. If length(n) > 1, the length is taken to be the number required.
h.diff	The step-length for the numerical differeniation inside inla.tmarginal
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^2L),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, length.out = 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, m)
m2 = inla.emarginal(function(x) x^2L, m)
stdev = sqrt(m2 - m1^2L)
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[2L]))
## using the buildt-in function
inla.zmarginal(m)
```

merge.inla *Merge a mixture of* inla*-objects*

Description

The function merge.inla implements method merge for inla-objects. merge.inla is a wrapper for the function inla.merge. The interface is slightly different, merge.inla is more tailored for interactive use, whereas inla.merge is better in general code.

inla.merge is intented for merging a mixture of inla-objects, each run with the same formula and settings, except for a set of hyperparameters that are fixed to different values. Using this function, we can then integrate over these hyperparameters using (unnormalized) integration weights prob. The main objects to be merged, are the summary statistics and marginal densities (like for hyperparameters, fixed, random, etc). Not all entries in the object can be merged, and by default these are inheritated from the first object in the list, while some are just set to NULL. Those objectes that are merged, will be listed if run with option verbose=TRUE.

Note that merging hyperparameter in the user-scale is prone to discretization error in general, so it is more stable to convert the marginal of the hyperparameter from the merged internal scale to the user-scale. (This is not done by this function.)

Usage

```
## S3 method for class 'inla'
merge(x, y, ..., prob = rep(1, length(list(x, y, ...))), verbose = FALSE)

inla.merge(loo, prob = rep(1, length(loo)), mc.cores = NULL, verbose = FALSE)
```

Arguments

x	An inla-object to be merged
y	An inla-object to be merged
...	Additional inla-objects to be merged
prob	The mixture of (possibly unnormalized) probabilities
verbose	Turn on verbose-output or not
loo	List of inla-objects to be merged
mc.cores	The number of cores to use in parallel::mclapply. If is.null(mc.cores), then check getOption("mc.cores") and inla.getOption("num.threads") in that order.

Value

A merged inla-object.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
set.seed(123)
n = 100
y = rnorm(n)
y[1:10] = NA
x = rnorm(n)
z1 = runif(n)
z2 = runif(n)*n
idx = 1:n
idx2 = 1:n
lc1 = inla.make.lincomb(idx = c(1, 2, 3))
names(lc1) = "lc1"
lc2 = inla.make.lincomb(idx = c(0, 1, 2, 3))
names(lc2) = "lc2"
lc3 = inla.make.lincomb(idx = c(0, 0, 1, 2, 3))
names(lc3) = "lc3"
lc = c(lc1, lc2, lc3)
rr = list()
for (logprec in c(0, 1, 2))
    rr[[length(rr)+1]] = inla(y ~ 1 + x + f(idx, z1) + f(idx2, z2),
             lincomb = lc,
             control.family = list(hyper = list(prec = list(initial = logprec))),
             control.predictor = list(compute = TRUE, link = 1),
             data = data.frame(y, x, idx, idx2, z1, z2))
r = inla.merge(rr, prob = seq_along(rr), verbose=TRUE)
summary(r)
```

`meshbuilder` *Interactive mesh building and diagnostics*

Description

Interactively design and build a triangle mesh for use with SPDE models, and assess the finite element approximation errors. The R code needed to recreate the mesh outside the interactive Shiny app is also generated. Spatial objects can be imported from the global workspace.

Usage

```
meshbuilder()
```

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

See Also

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

Examples

```
## Not run:
meshbuilder()

## End(Not run)
```

`Munich` *The Munich rent data*

Description

The Munich rent data

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.

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. ~~

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()`

pardiso — *Describe and check the PARDISO support in R-INLA*

Description

`inla.pardiso()` describes the `PARDISO` support in R-INLA, how to get the license key and enable it in the `R-INLA` package. `inla.pardiso.check()` check if the `PARDISO` support is working.

Usage

```
inla.pardiso()

inla.pardiso.check()
```

Author(s)

Havard Rue <hrue@r-inla.org>

pc.alphaw — *Utility functions for the PC prior for the* `alpha` *parameter in the Weibull likelihood*

Description

Functions to evaluate, sample, compute quantiles and percentiles of the PC prior for the `alpha` parameter in the Weibull likelihood

Usage

```
inla.pc.ralphaw(n, lambda = 5)

inla.pc.dalphaw(alpha, lambda = 5, log = FALSE)

inla.pc.qalphaw(p, lambda = 5)

inla.pc.palphaw(q, lambda = 5)
```

Arguments

`n`	Number of observations
`lambda`	The rate parameter in the PC-prior
`alpha`	Vector of evaluation points, where `alpha>0`.
`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 alpha parameter for the Weibull likelihood, where alpha=1 is the base model.

Value

inla.pc.dalphaw gives the density, inla.pc.palphaw gives the distribution function, inla.pc.qalphaw gives the quantile function, and inla.pc.ralphaw generates random deviates.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla.doc("pc.alphaw")

Examples

```
x = inla.pc.ralphaw(100,  lambda = 5)
d = inla.pc.dalphaw(x, lambda = 5)
x = inla.pc.qalphaw(0.5, lambda = 5)
inla.pc.palphaw(x, lambda = 5)
```

---

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

n	Number of observations
p	The order of the AR-model
lambda	The rate parameter in the prior
pac	A vector of partial autocorrelation coefficients
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.

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.

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	A correlation matrix
r	The off diagonal elements of a correlation matrix
n	Number of observations
lambda	The rate parameter in the prior
log	Logical. Return the density in natural or log-scale.

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 standardized Student-t*

Description

A function to evaluate the PC-prior for the degrees of freedom in a standardized Student-t distribution

Usage

```
inla.pc.ddof(dof, lambda, u, alpha, log = FALSE)
```

Arguments

dof	Degrees of freedom
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
log	Logical. Return the density or the log-density

Details

These functions implements the PC-prior for the dof in a standardized 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, log = 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, log = 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.gevtail — *Utility functions for the PC prior for the* tail *parameter in the GEV likelihood*

Description

Functions to evaluate, sample, compute quantiles and percentiles of the PC prior for the tail parameter in the GEV likelihood

Usage

```
inla.pc.rgevtail(n, lambda = 7)

inla.pc.dgevtail(xi, lambda = 7, log = FALSE)

inla.pc.qgevtail(p, lambda = 7)

inla.pc.pgevtail(q, lambda = 7)
```

Arguments

n	Number of observations
lambda	The rate parameter in the PC-prior
xi	Vector of evaluation points, where 1>xi>0.
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 `tail` parameter for the GEV likelihood, where `xi=0` is the base model.

Value

`inla.pc.dgevtail` gives the density, `inla.pc.pgevtail` gives the distribution function, `inla.pc.qgevtail` gives the quantile function, and `inla.pc.rgevtail` generates random deviates.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla.doc("pc.gevtail")

Examples

```
xi = inla.pc.rgevtail(100,  lambda = 7)
d = inla.pc.dgevtail(xi, lambda = 7)
xi = inla.pc.qgevtail(0.5, lambda = 7)
inla.pc.pgevtail(xi, lambda = 7)
```

---

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 = NULL
)

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 = NULL
)
```

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.
h	The h()-function, defaults to `inla.pc.multvar.h.default`. See that code for an example of how to write a user-specific function.
b	The b-vector (gradient) in the expression for the simplex option, `d(xi) = h(b^T xi)`
log	Evaluate the density in log-scale or ordinary scale.
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 specific 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)
```

---

pc.sn

Utility functions for the PC prior for skewness *in the skew-normal linkfunction and likelihood*

Description

Functions to evaluate, sample, compute quantiles and percentiles of the PC prior for the skewness in the skew-normal link-function and likelihood

Usage

```
inla.pc.rsn(n, lambda = 40)

inla.pc.dsn(skew, lambda = 40, log = FALSE)

inla.pc.qsn(p, lambda = 40)

inla.pc.psn(q, lambda = 40)
```

Arguments

n	number of observations
lambda	the rate parameter in the PC prior
skew	vector of evaluation points
log	logical. return the density in natural or log-scale.
p	vector of probabilities
q	vector of quantiles

Details

Defines the PC prior for the skewness for the skew-normal linkfunction and likelihood, where skew=0 is the base model. The skewness range from -0.99527... to 0.99527.... ca.

Value

inla.pc.dsn gives the density, inla.pc.psn gives the distribution function, inla.pc.qsn gives the quantile function, and inla.pc.rsn generates random deviates.

Author(s)

Havard Rue <hrue@r-inla.org>

See Also

inla.doc("pc.sn")

Examples

```
x = inla.pc.rsn(100,  lambda = 40)
d = inla.pc.dsn(x, lambda = 40)
x = inla.pc.qsn(0.5, lambda = 40)
inla.pc.psn(x, lambda = 40)
```

`plot.inla` *Default INLA plotting*

Description

Takes an `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,
  cex = 1.75,
  ...
)
```

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

`cex`
: The `cex` parameter in `par()`. If set to `NULL` or `0`, then default values will be used for graphics parameters

`...`
: 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.prior = 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("sf") && require("sp")) {
   for (projtype in c("longlat_norm", "lambert_norm", "mollweide_norm", "hammer_norm")) {
        plot(fmesher::fm_CRS(projtype), main = projtype)
    }
}

if (require("sf") && require("sp")) {
    oblique <- c(0, 45, 45, 0)
   for (projtype in c("longlat_norm", "lambert_norm", "mollweide_norm", "hammer_norm")) {
     INLA:::plot.inla.CRS(fmesher::fm_CRS(projtype, oblique = oblique), main = paste("oblique", projtype))
    }
}
```

---

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.

Format

Longtiude The Longtiude coordinate

Latitude The Latitude coordinate

See Also

PRprec

---

print.inla *Print an INLA fit*

Description

Print an INLA fit

Usage

```
## S3 method for class 'inla'
print(x, digits = 3L, ...)
```

Arguments

x	An inla-object (output from an inla()-call).
digits	Number of digits to print
...	other arguments.

Value

None

Author(s)

Havard Rue

See Also

inla()

PRprec *The PRprec data*

Description

A data frame with daily rainfall in the Parana State.

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::inla.reorderings(), num.threads = NULL)
```

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 for `TAUCS`; 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.
`num.threads`	Maximum number of threads the `inla`-program will use, or as 'A:B' defining the number threads in the outer (A) and inner (B) layer for nested parallelism.

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 = 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 = inla.reorderings())
```

Arguments

graph	A (inla-)graph object
reordering	The name of the 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)
```

---

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::inla.reorderings(),
  seed = 0L,
  logdens = ifelse(missing(sample), FALSE, TRUE),
  compute.mean = ifelse(missing(sample), FALSE, TRUE),
  num.threads = if (seed == 0L) "0:0" else NULL,
  selection = NULL,
  verbose = inla.getOption("verbose"),
  .debug = FALSE
)
```

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 for TAUCS; 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', and this is and should be the default behaviour. 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. PLEASE NOTE1: If seed!=0 then the computations will run in serial mode, over-riding whatever is set in num.threads (a warning might be issued). PLEASE NOTE2: If the PARDISO sparse matrix library is used, continuity of the samples with respect to small changes in the precision matrix, can be expected but is not guaranteed. If this feature is required, please use the TAUCS sparse matrix library.
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.
num.threads	Maximum number of threads the inla-program will use, or as 'A:B' defining the number threads in the outer (A) and inner (B) layer for nested parallelism. seed!=0 requires serial comptuations.
selection	A vector of indices of each sample to return. NULL means return the whole sample. (Note that the log-density retured, is for the whole sample.) The use of selection cannot be combined with the use of sample.
verbose	Logical. Run in verbose mode or not.
.debug	Logical. Internal debug-mode.

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 (require serial mode)
x = inla.qsample(n, Q=Q, seed = 123, num.threads="1:1")
## restart from same seed,  only sample 1
xx = inla.qsample(n=1, Q=Q, seed = 123, num.threads="1:1")
## continue from the save state, sample the remaining 2
xxx = inla.qsample(n=n-1, Q=Q, seed = -1, num.threads="1:1")
## 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 = inla.reorderings(),
  method = c("solve", "forward", "backward")
)
```

Arguments

Q	A SPD matrix, either as a (dense) matrix or sparse-matrix
B	The right hand side matrix, either as a (dense) matrix or sparse-matrix.
reordering	The type of reordering algorithm to be used for TAUCS; 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 (using the TAUCS library).
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
nb <- n-1
QQ = matrix(rnorm(n^2), n, n)
QQ <- QQ %*% t(QQ)

Q = inla.as.sparse(QQ)
B = matrix(rnorm(n*nb), n, nb)

X = inla.qsolve(Q, B, method = "solve")
XX = inla.qsolve(Q, B, method = "solve", reordering = inla.qreordering(Q))
print(paste("err solve1", sum(abs( Q %*% X - B))))
print(paste("err solve2", sum(abs( Q %*% XX - B))))

## the forward and backward solve is tricky, as after permutation and with Q=LL', then L is
## lower triangular, but L in the orginal ordering is not lower triangular. if the rhs is iid
## noise, this is not important. to control the reordering, then the 'taucs' library must be
## used.
inla.setOption(smtp = 'taucs')

## case 1. use the matrix as is, no reordering
r <- "identity"
L = t(chol(Q))
X = inla.qsolve(Q, B, method = "forward", reordering = r)
XX = inla.qsolve(Q, B, method = "backward", reordering = r)
print(paste("err forward ", sum(abs(L %*% X - B))))
print(paste("err backward", sum(abs(t(L) %*% XX - B))))

## case 2. use a reordering from the library
r <- inla.qreordering(Q)
im <- r$ireordering
m <- r$reordering
print(cbind(idx = 1:n, m, im) )
Qr <- Q[im, im]
L = t(chol(Qr))[m, m]

X = inla.qsolve(Q, B, method = "forward", reordering = r)
XX = inla.qsolve(Q, B, method = "backward", reordering = r)
print(paste("err forward ", sum(abs( L %*% X - B))))
print(paste("err backward", sum(abs( t(L) %*% XX - B))))
```

---

read.graph *Read and write a graph-object*

Description

Construct a graph-object from a file or a matrix; 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'
plot(x, y, ...)

## S3 method for class 'inla.graph'
summary(object, ...)

## S3 method for class 'inla.graph.summary'
print(x, ...)
```

Arguments

...	Additional arguments. In inla.read.graph, then it is the graph definition (object, matrix, character, filename), plus extra arguments. In inla.write.graph it is extra arguments to inla.read.graph.
size.only	Only read the size 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
filename	The filename of the graph.
mode	The mode of the file; ascii-file or a (gzip-compressed) binary.
x	An inla.graph -object
y	Not used
object	An inla.graph -object

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 • idis 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
- meanis a factor with one level for each connected component of size larger than one, otherwise NA

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 from a file
g.file1 <- tempfile() # E.g. "g.dat"
cat("3 1 1 2 2 1 1 3 0\n", file = g.file1)
g = inla.read.graph(g.file1)
## writing an inla.graph-object to file
g.file2 = inla.write.graph(g, mode="binary", filename = tempfile())
## re-reading it from that file
gg = inla.read.graph(g.file2)
summary(g)
summary(gg)

## Not run:
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"))

## building a graph from adjacency matrix
adjacent = matrix(0, nrow = 4, ncol = 4)
adjacent[1,4] = adjacent[4,1] = 1
adjacent[2,4] = adjacent[4,2] = 1
adjacent[2,3] = adjacent[3,2] = 1
adjacent[3,4] = adjacent[4,3] = 1
g = inla.read.graph(adjacent)
plot(g)
summary(g)

## End(Not run)
```

rgeneric.define *rgeneric models*

Description

A framework for defining latent models in R

Usage

```
inla.rgeneric.ar1.model(
  cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
  theta = NULL
)

inla.rgeneric.ar1.model.opt(
  cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
  theta = NULL
)

inla.rgeneric.iid.model(
  cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
  theta = NULL
)

inla.rgeneric.define(
  model = NULL,
  debug = FALSE,
  compile = TRUE,
  optimize = FALSE,
  ...
)

inla.rgeneric.wrapper(
  cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
  model,
  theta = NULL
)

inla.rgeneric.q(
  rmodel,
  cmd = c("graph", "Q", "mu", "initial", "log.norm.const", "log.prior", "quit"),
  theta = NULL
)
```

Arguments

cmd	An allowed request
theta	Values of theta
model	The definition of the model; see `inla.rgeneric.ar1.model`
debug	Logical. Enable debug output

compile	Logical. Compile the definition of the model or not.
optimize	Logical. With this option TRUE, then model pass only the values of Q and not the whole matrix. Please see the vignette for details and inla.rgeneric.ar1.model.opt for an example.
...	Named list of variables that defines the environment of model
rmodel	The rgeneric model-object, the output of inla.rgeneric.define

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

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.internal(Q, constr = NULL, eps = sqrt(.Machine$double.eps))

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(MASS::ginv(as.matrix(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(MASS::ginv(as.matrix(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(MASS::ginv(as.matrix(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

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

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.

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 = 3L, include.lincomb = TRUE, ...)

## S3 method for class 'summary.inla'
print(x, digits = 3L, ...)
```

Arguments

object	a fitted inla object as produced by inla.
digits	Integer Number of digits
include.lincomb	Logcial Include the summary for the the linear combinations or not
...	other arguments.
x	a summary.inla object produced by summary.inla

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

If the option short.summary is set to TRUE using inla.setOption, then a less verbose summary variant will be used, which might be more suitable for Markdown documents.

Value

summary.inla returns an object of class summary.inla, a list of components to print.

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().
verbose	If TRUE, produce a more detailed output.
...	further arguments passed to or from other methods.
x	an object of class "summary.inla.mesh", usually a result of a call to summary.inla.mesh().

Author(s)

Finn Lindgren <finn.lindgren@gmail.com>

summary.scopy *Computes the mean and stdev for the spline from* scopy

Description

This function computes the mean and stdev for the spline function that is implicite from an scopy model component

Usage

```
inla.summary.scopy(result, name, by = 0.05, range = c(0, 1))
```

Arguments

result	An inla-object, ie the output from an inla() call
name	The name of the scopy model component see ?inla::f and argument extraconstr
by	The resolution of the results, in the scale where distance between two nearby locations is 1
range	The range of the locations, in (from, to)

Value

A `data.frame` with locations, mean and stdev. if `name` is not found, NULL is returned.

Author(s)

Havard Rue <hrue@r-inla.org>

Examples

```
## see example in inla.doc("scopy")
```

Surg *Surgical: Institutional ranking*

Description

This example considers mortality rates in 12 hospitals performing cardiac surgery in babies

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

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

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.

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

∗ **Survival**
inla.surv, 382
∗ **control**
control.bgev, 11
control.compute, 12
control.expert, 13
control.family, 14
control.fixed, 16
control.gcpo, 17
control.group, 19
control.hazard, 20
control.inla, 21
control.lincomb, 25
control.link, 26
control.lp.scale, 27
control.mix, 27
control.mode, 28
control.pardiso, 29
control.pom, 30
control.predictor, 31
control.scopy, 32
control.update, 33
control.vb, 34
∗ **datasets**
BivMetaAnalysis, 9
Cancer, 10
Drivers, 39
Epil, 40
Germany, 46
Kidney, 388
Leuk, 390
Munich, 399
nwEngland, 400
Oral, 400
PRborder, 421
PRprec, 422
Salm, 440
Scotland, 442
Seeds, 442
SPDEtoy, 443
Surg, 446
SurvSim, 446
Tokyo, 447
Zambia, 447
∗ **fmesher**
inla.stack.remove.unused, 377
∗ **graph**
graph.convert, 47
∗ **models**
inla.surv, 382
∗ **plot**
plot.inla, 415
1djmarginal *(*joint.marginal*)*, 385

agaussian *(*inla.agaussian*)*, 55
aggregate.gaussian *(*inla.agaussian*)*, 55
ar.dpacf *(*pc.ar*)*, 403
ar.pacf2acf *(*inla.ar.pacf2phi*)*, 56
ar.pacf2phi *(*inla.ar.pacf2phi*)*, 56
ar.phi2acf *(*inla.ar.pacf2phi*)*, 56
ar.phi2pacf *(*inla.ar.pacf2phi*)*, 56
ar.rpacf *(*pc.ar*)*, 403
as.inla.mdata *(*inla.mdata*)*, 87
as.inla.mesh.segment, 8
as.inla.surv *(*inla.surv*)*, 382

barrier *(*inla.barrier.pcmatern*)*, 59
barrier *(*inla.barrier*)*, 58
binary.install *(*inla.binary.install*)*, 60
BivMetaAnalysis, 9

Cancer, 10
cbind.data.frames *(*inla.coxph*)*, 63
cgeneric, 10
changelog *(*inla.changelog*)*, 61
collect.results *(*inla.collect.results*)*, 62
contour(), *106*
control.bgev, 11, *13, 14*, *16–18*, *20, 21*, *25–33*, *35*
control.compute, *11*, 12, *14*, *16–18*, *20, 21*, *25–33*, *35*
control.compute(), *75*
control.expert, *11*, *13*, 13, *16–18*, *20, 21*, *25–33*, *35*
control.family, *11*, *13, 14*, 14, *17, 18*, *20, 21*, *25–33*, *35*

control.fixed, *11*, *13, 14*, *16*, 16, *18*, *20, 21*, *25–33*, *35*
control.gcpo, *11*, *13, 14*, *16, 17*, 17, *20, 21*, *25–33*, *35*
control.group, *11*, *13, 14*, *16–18*, 19, *21*, *25–33*, *35*
control.hazard, *11*, *13, 14*, *16–18*, *20*, 20, *25–33*, *35*
control.inla, *11*, *13, 14*, *16–18*, *20, 21*, 21, *26–35*
control.lincomb, *11*, *13, 14*, *16–18*, *20, 21*, *25*, 25, *27–33*, *35*
control.link, *11*, *13, 14*, *16–18*, *20, 21*, *25, 26*, 26, *27–33*, *35*
control.lp.scale, *11*, *13, 14*, *16–18*, *20, 21*, *25–27*, 27, *28–33*, *35*
control.mix, *11*, *13, 14*, *16–18*, *20, 21*, *25–27*, 27, *29–33*, *35*
control.mode, *11*, *13, 14*, *16–18*, *20, 21*, *25–28*, 28, *30–33*, *35*
control.pardiso, *11*, *13, 14*, *16–18*, *20, 21*, *25–29*, 29, *31–33*, *35*
control.pom, *11*, *13, 14*, *16–18*, *20, 21*, *25–30*, 30, *32, 33*, *35*
control.predictor, *11*, *13, 14*, *16–18*, *20, 21*, *25–31*, 31, *33*, *35*
control.scopy, *11*, *13, 14*, *16–18*, *20, 21*, *25–32*, 32, *33*, *35*
control.update, *11*, *13, 14*, *16–18*, *20, 21*, *25–33*, 33, *35*
control.vb, *11*, *13, 14*, *16–18*, *20, 21*, *25–33*, 34
control.vb(), *25*
cormat.dim2p *(*pc.cormat*)*, 406
cormat.dtheta *(*pc.cormat*)*, 406
cormat.p2dim *(*pc.cormat*)*, 406
cormat.permute *(*pc.cormat*)*, 406
cormat.R2r *(*pc.cormat*)*, 406
cormat.r2R *(*pc.cormat*)*, 406
cormat.R2theta *(*pc.cormat*)*, 406
cormat.r2theta *(*pc.cormat*)*, 406
cormat.rtheta *(*pc.cormat*)*, 406
cormat.theta2R *(*pc.cormat*)*, 406
cormat.theta2r *(*pc.cormat*)*, 406
coxph *(*inla.coxph*)*, 63
cpo.inla *(*inla.cpo*)*, 64
crs_wkt, 35
crs_wkt(), *66*
cut, 37

debug.graph, 38
dev.new(), *68*
dmarginal *(*marginal*)*, 394
Drivers, 39
dryrun, 39

emarginal *(*marginal*)*, 394
Epil, 40
extract.groups, 40

f, 41
f(), *51*, *54*, *74*, *80*
fgn, 45
fmesher::fm_as_segm(), *8, 9*
fmesher::fm_bary(), *72*
fmesher::fm_CRS(), *65*
fmesher::fm_crs_is_geocent(), *36*
fmesher::fm_crs_is_identical(), *79*
fmesher::fm_ellipsoid_radius(), *36*
fmesher::fm_fem(), *71*
fmesher::fm_mesh_1d(), *88*
fmesher::fm_mesh_2d_inla(), *89*
fmesher::fm_nonconvex_hull(), *341*
fmesher::fm_nonconvex_hull_inla(), *341*
fmesher::fm_proj4string(), *66*
fmesher::fm_raw_basis(), *71, 72*, *92*
fmesher::fm_rcdt_2d_inla(), *96*
fmesher::fm_segm(), *105*
fmesher::fm_split_lines(), *72*
fmesher::fm_transform(), *375*
fmesher::fm_wkt(), *35*, *66*
fmesher::fm_wkt_as_wkt_tree(), *57*
fmesher::fm_wkt_is_geocent(), *36*
fmesher::fm_wkt_predef(), *66*
fmesher::fm_wkt_set_ellipsoid_radius(), *36*
fmesher::fmesher_split_lines(), *72*

geobugs2inla *(*graph.convert*)*, 47
Germany, 46
get.inlaEnv *(*inla.get.inlaEnv*)*, 73
graph.convert, 47
graph.matrix, 47
graph2matrix *(*graph.matrix*)*, 47

hpdmarginal *(*marginal*)*, 394
hyperpar.inla *(*inla.hyperpar*)*, 76
hyperpar.inla(), *45*, *47*
hyperpar.sample *(*inla.hyperpar.sample*)*, 78
hyperpar.sampler *(*inla.hyperpar.sample*)*, 78

idx, 49
iidkd.sample *(*inla.iidkd.sample*)*, 79
INLA *(*INLA-package*)*, 8

inla, 50, *55*, *389*
inla(), *11–14*, *16, 17*, *19–21*, *25–34*, *45*, *47*, *62*, *65*, *77*, *81*, *87*, *347, 348*, *361*, *382, 383*, *386*, *394*, *396*, *416*, *421*, *444*
inla-class, 55
INLA-package, 8
inla.1djmarginal *(*joint.marginal*)*, 385
inla.agaussian, 55
inla.ar.pacf2acf *(*inla.ar.pacf2phi*)*, 56
inla.ar.pacf2phi, 56
inla.ar.phi2acf *(*inla.ar.pacf2phi*)*, 56
inla.ar.phi2pacf *(*inla.ar.pacf2phi*)*, 56
inla.as.CRS.list *(*inla.CRSargs*)*, 66
inla.as.CRSargs.list *(*inla.CRSargs*)*, 66
inla.as.dgTMatrix *(*inla.as.sparse*)*, 57
inla.as.list.CRS *(*inla.CRSargs*)*, 66
inla.as.list.CRSargs *(*inla.CRSargs*)*, 66
inla.as.sparse, 57
inla.as.wkt.wkt_tree *(*inla.as.wkt_tree.wkt*)*, 57
inla.as.wkt_tree.wkt, 57
inla.barrier, 58
inla.barrier *(*inla.barrier.pcmatern*)*, 59
inla.barrier.pcmatern, 59
inla.binary.install, 60
inla.cgeneric.define *(*cgeneric*)*, 10
inla.cgeneric.q *(*cgeneric*)*, 10
inla.changelog, 61
inla.changes *(*inla.changelog*)*, 61
inla.collect.results, 62
inla.contour.segment *(*inla.mesh.segment*)*, 105
inla.coxph, 63
inla.cpo, 64
inla.CRS, 65
inla.CRS(), *67*, *375*, *417, 418*
inla.crs_get_ellipsoid_radius *(*crs_wkt*)*, 35
inla.crs_get_lengthunit *(*crs_wkt*)*, 35
inla.crs_get_wkt *(*crs_wkt*)*, 35
inla.crs_is_geocent *(*crs_wkt*)*, 35
inla.crs_set_ellipsoid_radius *(*crs_wkt*)*, 35
inla.crs_set_lengthunit *(*crs_wkt*)*, 35
inla.CRSargs, 66
inla.cut *(*cut*)*, 37
inla.debug.graph *(*debug.graph*)*, 38
inla.delaunay *(*inla.mesh.create*)*, 96
inla.delaunay(), *90*
inla.dev.new, 68
inla.diameter, 68
inla.dmarginal *(*marginal*)*, 394
inla.doc, 69
inla.doc(), *42*, *51*, *85*
inla.dryrun *(*dryrun*)*, 39
inla.emarginal *(*marginal*)*, 394
inla.external.lib, 70
inla.extract.el, 70
inla.fallback_PROJ6 *(*inla.has_PROJ6*)*, 75
inla.fgn *(*fgn*)*, 45
inla.fmesher.smorg, 71
inla.generate.colors, 72
inla.geobugs2inla *(*graph.convert*)*, 47
inla.get.inlaEnv, 73
inla.getOption *(*inla.option*)*, 342
inla.graph *(*read.graph*)*, 436
inla.graph2matrix *(*graph.matrix*)*, 47
inla.group, 73
inla.group.cv, 74
inla.has_PROJ6, 75
inla.hpdmarginal *(*marginal*)*, 394
inla.hyperpar, 76
inla.hyperpar(), *396*
inla.hyperpar.sample, 78
inla.hyperpar.sampler *(*inla.hyperpar.sample*)*, 78
inla.identical.CRS, 79
inla.identical.CRS(), *66*
inla.idx *(*idx*)*, 49
inla.iidkd.sample, 79
inla.inla.doc *(*inla.doc*)*, 69
inla.is.marginal *(*marginal*)*, 394
inla.joint.marginal *(*joint.marginal*)*, 385
inla.jp.define *(*jp*)*, 387
inla.knmodels, 80
inla.knmodels(), *83*
inla.knmodels.sample, 82
inla.knmodels.sample(), *81*
inla.ks.plot, 83
inla.lattice2node *(*lattice2node*)*, 388
inla.likelihood, 84
inla.link *(*link*)*, 392
inla.list.models, 85
inla.list.models(), *41*
inla.make.lincomb *(*make.lincomb*)*, 393
inla.make.lincombs *(*make.lincomb*)*, 393
inla.marginal *(*marginal*)*, 394
inla.matern.cov, 86
inla.matrix2graph *(*graph.matrix*)*, 47
inla.matrix2vector *(*lattice2node*)*, 388
inla.mdata, 87
inla.merge *(*merge.inla*)*, 397

inla.mesh (inla.mesh.create), 96
inla.mesh(), 92, 93, 98–100, 102, 103, 105, 344, 355, 364, 368, 370, 371, 401, 418, 419
inla.mesh.1d, 88
inla.mesh.1d(), 89, 92, 98, 99, 103, 355, 356, 368, 369, 371, 373
inla.mesh.1d.A (inla.mesh.1d.bary), 89
inla.mesh.1d.bary, 89
inla.mesh.1d.fem(), 99
inla.mesh.2d, 89
inla.mesh.2d(), 92, 94, 96, 98, 106, 365, 369, 373, 398, 445
inla.mesh.assessment, 91
inla.mesh.basis, 92
inla.mesh.basis(), 365, 369, 373
inla.mesh.boundary, 93
inla.mesh.components, 94, 95
inla.mesh.create, 96
inla.mesh.create(), 90, 93, 94, 104–106, 369, 373, 398, 445
inla.mesh.create.helper(), 93
inla.mesh.deriv, 98
inla.mesh.fem, 99
inla.mesh.interior (inla.mesh.boundary), 93
inla.mesh.lattice, 99
inla.mesh.lattice(), 97, 98, 103, 104
inla.mesh.map (inla.mesh.map.lim), 101
inla.mesh.map.lim, 101
inla.mesh.project, 102
inla.mesh.project(), 102
inla.mesh.projector (inla.mesh.project), 102
inla.mesh.query, 104
inla.mesh.query(), 98
inla.mesh.segment, 105
inla.mesh.segment(), 9, 41, 90, 93, 97, 98, 104, 342, 391
inla.mmarginal (marginal), 394
inla.models, 107
inla.nmix.fitted (inla.nmix.lambda.fitted), 338
inla.nmix.lambda.fitted, 338
inla.node2lattice (lattice2node), 388
inla.nonconvex.hull, 340
inla.nonconvex.hull(), 90
inla.not_for_PROJ4 (inla.has_PROJ6), 75
inla.not_for_PROJ6 (inla.has_PROJ6), 75
inla.option, 342
inla.options (inla.option), 342
inla.over_sp_mesh, 344
inla.pardiso (pardiso), 402
inla.pc.alphaw (pc.alphaw), 402
inla.pc.ar (pc.ar), 403
inla.pc.cor0 (pc.cor0), 404
inla.pc.cor1 (pc.cor1), 405
inla.pc.cormat (pc.cormat), 406
inla.pc.dalphaw (pc.alphaw), 402
inla.pc.dcor0 (pc.cor0), 404
inla.pc.dcor1 (pc.cor1), 405
inla.pc.ddof (pc.ddof), 407
inla.pc.dgamma (pc.gamma), 408
inla.pc.dgammacount (pc.gammacount), 409
inla.pc.dgevtail (pc.gevtail), 410
inla.pc.dof (pc.ddof), 407
inla.pc.dprec (pc.prec), 413
inla.pc.dsn (pc.sn), 414
inla.pc.gamma (pc.gamma), 408
inla.pc.gammacount (pc.gammacount), 409
inla.pc.gevtail (pc.gevtail), 410
inla.pc.multvar (pc.multvar), 411
inla.pc.palphaw (pc.alphaw), 402
inla.pc.pcor0 (pc.cor0), 404
inla.pc.pcor1 (pc.cor1), 405
inla.pc.pgamma (pc.gamma), 408
inla.pc.pgammacount (pc.gammacount), 409
inla.pc.pgevtail (pc.gevtail), 410
inla.pc.pprec (pc.prec), 413
inla.pc.prec (pc.prec), 413
inla.pc.psn (pc.sn), 414
inla.pc.qalphaw (pc.alphaw), 402
inla.pc.qcor0 (pc.cor0), 404
inla.pc.qcor1 (pc.cor1), 405
inla.pc.qgamma (pc.gamma), 408
inla.pc.qgammacount (pc.gammacount), 409
inla.pc.qgevtail (pc.gevtail), 410
inla.pc.qprec (pc.prec), 413
inla.pc.qsn (pc.sn), 414
inla.pc.ralphaw (pc.alphaw), 402
inla.pc.rcor0 (pc.cor0), 404
inla.pc.rcor1 (pc.cor1), 405
inla.pc.rgamma (pc.gamma), 408
inla.pc.rgammacount (pc.gammacount), 409
inla.pc.rgevtail (pc.gevtail), 410
inla.pc.rprec (pc.prec), 413
inla.pc.rsn (pc.sn), 414
inla.pc.sn (pc.sn), 414
inla.pc.t (pc.ddof), 407
inla.plot (plot.inla), 415
inla.pmarginal (marginal), 394
inla.posterior.sample (inla.sample), 350
inla.priors.used, 345
inla.prune, 346

inla.q *(inla.qstat)*, 346
inla.qdel *(inla.qstat)*, 346
inla.qget *(inla.qstat)*, 346
inla.qinv *(qinv)*, 431
inla.qlog *(inla.qstat)*, 346
inla.qmarginal *(marginal)*, 394
inla.qnuke *(inla.qstat)*, 346
inla.qreordering *(qreordering)*, 432
inla.qreordering(), *48*
inla.qsample *(qsample)*, 433
inla.qsample(), *363*
inla.qsolve *(qsolve)*, 435
inla.qstat, 346
inla.rbind.data.frames *(inla.coxph)*, 63
inla.read.graph *(read.graph)*, 436
inla.read.graph(), *48*
inla.remote *(inla.ssh.copy.id)*, 377
inla.reorderings, 348
inla.requires_PROJ6 *(inla.has_PROJ6)*, 75
inla.rerun, 348
inla.rgeneric.ar1.model *(rgeneric.define)*, 439
inla.rgeneric.define *(rgeneric.define)*, 439
inla.rgeneric.iid.model *(rgeneric.define)*, 439
inla.rgeneric.q *(rgeneric.define)*, 439
inla.rgeneric.wrapper *(rgeneric.define)*, 439
inla.rjmarginal *(joint.marginal)*, 385
inla.rmarginal *(marginal)*, 394
inla.row.kron, 349
inla.sample, 350
inla.scale.model *(scale.model)*, 441
inla.set.control.bgev.default *(control.bgev)*, 11
inla.set.control.compute.default *(control.compute)*, 12
inla.set.control.expert.default *(control.expert)*, 13
inla.set.control.family.default *(control.family)*, 14
inla.set.control.fixed.default *(control.fixed)*, 16
inla.set.control.gcpo.default *(control.gcpo)*, 17
inla.set.control.group.default *(control.group)*, 19
inla.set.control.hazard.default *(control.hazard)*, 20
inla.set.control.inla.default *(control.inla)*, 21
inla.set.control.lincomb.default *(control.lincomb)*, 25
inla.set.control.link.default *(control.link)*, 26
inla.set.control.lp.scale.default *(control.lp.scale)*, 27
inla.set.control.mix.default *(control.mix)*, 27
inla.set.control.mode.default *(control.mode)*, 28
inla.set.control.pardiso.default *(control.pardiso)*, 29
inla.set.control.pom.default *(control.pom)*, 30
inla.set.control.predictor.default *(control.predictor)*, 31
inla.set.control.scopy.default *(control.scopy)*, 32
inla.set.control.update.default *(control.update)*, 33
inla.set.control.vb.default *(control.vb)*, 34
inla.setOption *(inla.option)*, 342
inla.simplify.curve, 354
inla.simplify.curve(), *106*, *341*
inla.smarginal *(marginal)*, 394
inla.sp2segment *(as.inla.mesh.segment)*, 8
inla.sp_get_crs, 376
inla.sp_get_crs(), *36*, *66*
inla.spde.make.A, 355
inla.spde.make.A(), *349*, *356–358*, *380*
inla.spde.make.block.A, 356
inla.spde.make.block.A(), *356*
inla.spde.make.index, 357
inla.spde.make.index(), *356*, *380*
inla.spde.models, 358
inla.spde.models(), *361*, *362*
inla.spde.precision, 360
inla.spde.precision(), *363*
inla.spde.result, 361
inla.spde.sample, 363
inla.spde1 *(inla.spde1.create)*, 364
inla.spde1(), *359*
inla.spde1.create, 364
inla.spde1.models *(inla.spde.models)*, 358
inla.spde1.precision *(inla.spde.precision)*, 360
inla.spde1.result *(inla.spde.result)*, 361
inla.spde2 *(inla.spde2.generic)*, 366

inla.spde2(), *359*, *364*, *367*
inla.spde2.generic, 366
inla.spde2.generic(), *361*, *369*, *373*
inla.spde2.matern, 367
inla.spde2.matern(), *355*, *361*, *362*, *365*, *367*, *370*, *371*, *373*, *401*
inla.spde2.matern.sd.basis, 370
inla.spde2.models *(*inla.spde.models*)*, 358
inla.spde2.models(), *367*
inla.spde2.pcmatern, 371
inla.spde2.pcmatern(), *369*
inla.spde2.precision *(*inla.spde.precision*)*, 360
inla.spde2.result *(*inla.spde.result*)*, 361
inla.spde2.result(), *358*
inla.spde2.theta2phi0 *(*inla.spde2.matern*)*, 367
inla.spde2.theta2phi0(), *361*
inla.spde2.theta2phi1 *(*inla.spde2.matern*)*, 367
inla.spde2.theta2phi1(), *361*
inla.spde2.theta2phi2 *(*inla.spde2.matern*)*, 367
inla.spde2.theta2phi2(), *361*
inla.spTransform, 375
inla.spy *(*graph.matrix*)*, 47
inla.spy(), *438*
inla.ssh.copy.id, 377
inla.stack *(*inla.stack.remove.unused*)*, 377
inla.stack.remove.unused, 377
inla.summary.scopy *(*summary.scopy*)*, 445
inla.surv, 382
inla.surv(), *47*
inla.tjmarginal *(*joint.marginal*)*, 385
inla.tmarginal *(*marginal*)*, 394
inla.update, 384
inla.upgrade *(*inla.update*)*, 384
inla.vector2matrix *(*lattice2node*)*, 388
inla.version, 384
inla.wkt_get_ellipsoid_radius *(*crs_wkt*)*, 35
inla.wkt_get_lengthunit *(*crs_wkt*)*, 35
inla.wkt_is_geocent *(*crs_wkt*)*, 35
inla.wkt_predef *(*inla.CRS*)*, 65
inla.wkt_set_ellipsoid_radius *(*crs_wkt*)*, 35
inla.wkt_set_lengthunit *(*crs_wkt*)*, 35
inla.wkt_tree_get_item *(*inla.as.wkt_tree.wkt*)*, 57
inla.wkt_tree_set_item *(*inla.as.wkt_tree.wkt*)*, 57
inla.wkt_unit_params *(*crs_wkt*)*, 35
inla.write.graph *(*read.graph*)*, 436
inla.zmarginal *(*marginal*)*, 394
is.inla.mdata *(*inla.mdata*)*, 87
is.inla.surv *(*inla.surv*)*, 382

joint.marginal, 385
jp, 387

Kidney, 388
knmodels *(*inla.knmodels*)*, 80
ks.plot *(*inla.ks.plot*)*, 83
ks.test(), *83*, *84*

lattice2node, 388
Leuk, 390
Leukemia *(*Leuk*)*, 390
likelihood *(*inla.likelihood*)*, 84
lines.inla.mesh.segment, 391
link, 392
list.models *(*inla.list.models*)*, 85

make.lincomb, 393
make.lincombs *(*make.lincomb*)*, 393
marginal, 394
matrix2vector *(*lattice2node*)*, 388
mdata *(*inla.mdata*)*, 87
merge.inla, 397
meshbuilder, 398
meshbuilder(), *91*
mmarginal *(*marginal*)*, 394
Munich, 399

NewEngland *(*nwEngland*)*, 400
nmix.lambda.fitted *(*inla.nmix.lambda.fitted*)*, 338
node2lattice *(*lattice2node*)*, 388
nwEngland, 400

Oral, 400

pacf2acf *(*inla.ar.pacf2phi*)*, 56
pacf2phi *(*inla.ar.pacf2phi*)*, 56
param2.matern.orig, 401
pardiso, 402
pc.alphaw, 402
pc.ar, 403
pc.cor0, 404
pc.cor1, 405
pc.cormat, 406
pc.dalphaw *(*pc.alphaw*)*, 402
pc.dcor0 *(*pc.cor0*)*, 404

pc.dcor1 (pc.cor1), 405
pc.ddof, 407
pc.dgamma (pc.gamma), 408
pc.dgammacount (pc.gammacount), 409
pc.dgevtail (pc.gevtail), 410
pc.dof (pc.ddof), 407
pc.dprec (pc.prec), 413
pc.dsn (pc.sn), 414
pc.gamma, 408
pc.gammacount, 409
pc.gevtail, 410
pc.multvar, 411
pc.palphaw (pc.alphaw), 402
pc.pcor0 (pc.cor0), 404
pc.pcor1 (pc.cor1), 405
pc.pgamma (pc.gamma), 408
pc.pgammacount (pc.gammacount), 409
pc.pgevtail (pc.gevtail), 410
pc.pprec (pc.prec), 413
pc.prec, 413
pc.psn (pc.sn), 414
pc.qalphaw (pc.alphaw), 402
pc.qcor0 (pc.cor0), 404
pc.qcor1 (pc.cor1), 405
pc.qgamma (pc.gamma), 408
pc.qgammacount (pc.gammacount), 409
pc.qgevtail (pc.gevtail), 410
pc.qprec (pc.prec), 413
pc.qsn (pc.sn), 414
pc.ralphaw (pc.alphaw), 402
pc.rcor0 (pc.cor0), 404
pc.rcor1 (pc.cor1), 405
pc.rgamma (pc.gamma), 408
pc.rgammacount (pc.gammacount), 409
pc.rgevtail (pc.gevtail), 410
pc.rprec (pc.prec), 413
pc.rsn (pc.sn), 414
pc.sn, 414
pc.t (pc.ddof), 407
phi2acf (inla.ar.pacf2phi), 56
phi2pacf (inla.ar.pacf2phi), 56
plot.CRS (plot.inla.CRS), 417
plot.CRS(), *66*
plot.inla, 415
plot.inla.CRS, 417
plot.inla.graph (read.graph), 436
plot.inla.mesh, 418
plot.inla.mesh(), *421*
plot.inla.surv (inla.surv), 382
plot.inla.trimesh, 420
plot.inla.trimesh(), *419*
pmarginal (marginal), 394
posterior.sample (inla.sample), 350
PRborder, 421
print.inla, 421
print.inla.graph.summary (read.graph), 436
print.inla.jmarginal (joint.marginal), 385
print.inla.mdata (inla.mdata), 87
print.inla.q (inla.qstat), 346
print.inla.surv (inla.surv), 382
print.summary.inla (summary.inla), 444
print.summary.inla.jmarginal (joint.marginal), 385
print.summary.inla.mesh (summary.inla.mesh), 445
priors.used (inla.priors.used), 345
PRprec, 422
prune (inla.prune), 346

qinv, 431
qmarginal (marginal), 394
qreordering, 432
qsample, 433
qsolve, 435

read.graph, 436
reorderings (inla.reorderings), 348
rerun (inla.rerun), 348
rgeneric (rgeneric.define), 439
rgeneric.define, 439
rjmarginal (joint.marginal), 385
rmarginal (marginal), 394

Salm, 440
scale.model, 441
Scotland, 442
Seeds, 442
smarginal (marginal), 394
sp::CRS(), *66*
sp::over(), *344*
sp::SpatialPolygons(), *344*
SPDEtoy, 443
spy (graph.matrix), 47
ssh.copy.id (inla.ssh.copy.id), 377
summary.inla, 444
summary.inla.graph (read.graph), 436
summary.inla.jmarginal (joint.marginal), 385
summary.inla.mesh, 445
summary.inla.mesh(), *445*
summary.inla.q (inla.qstat), 346
summary.scopy, 445
summary.surv.inla (summary.inla), 444

Surg, 446
SurvSim, 446

tjmarginal *(*joint.marginal*)*, 385
Tokyo, 447

vector2matrix *(*lattice2node*)*, 388
version *(*inla.version*)*, 384

write.graph *(*read.graph*)*, 436

Zambia, 447
zmarginal *(*marginal*)*, 394