

# Package ‘INLAspacetime’

April 21, 2024

**Type** Package

**Title** Spatial and Spatio-Temporal Models using 'INLA'

**Version** 0.1.8

**Description** Prepare objects to implement models over spatial and spacetime domains with the 'INLA' package (<<https://www.r-inla.org>>). These objects contain data to for the 'cgeneric' interface in 'INLA', enabling fast parallel computations. We implemented the spatial barrier model, see Bakka et. al. (2019) <[doi:10.1016/j.spasta.2019.01.002](https://doi.org/10.1016/j.spasta.2019.01.002)>, and some of the spatio-temporal models in Lindgren et. al. (2023) <[doi:10.48550/arXiv.2006.04917](https://doi.org/10.48550/arXiv.2006.04917)>. Details are provided in the available vignettes and from the URL bellow.

**URL** <https://github.com/eliaskrainski/INLAspacetime>

**Additional\_repositories** <https://inla.r-inla-download.org/R/testing>

**BugReports** <https://github.com/eliaskrainski/INLAspacetime/issues>

**License** GPL (>= 2)

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**NeedsCompilation** yes

**Depends** R (>= 4.3), Matrix, fmesher

**Imports** graphics, grDevices, methods, stats, utils, sp, sf, terra

**Suggests** INLA (>= 24.02.09), inlabru (>= 2.10.1), knitr, ggplot2, rmarkdown, parallel, data.table, rnaturalearth

**VignetteBuilder** knitr

**BuildVignettes** true

**Author** Elias Teixeira Krainski [cre, aut, cph] (<<https://orcid.org/0000-0002-7063-2615>>), Finn Lindgren [aut] (<<https://orcid.org/0000-0002-5833-2011>>), Haavard Rue [aut] (<<https://orcid.org/0000-0002-0222-1881>>)

**Maintainer** Elias Teixeira Krainski <[eliaskrainski@gmail.com](mailto:eliaskrainski@gmail.com)>

**Repository** CRAN

**Date/Publication** 2024-04-21 21:02:44 UTC

**R topics documented:**

ar2cor	2
ar2precision	3
barrierModel.define	4
bru_get_mapper.stModel_cgeneric	5
cWhittleMatern	6
downloadUtilFiles	7
Earth_poly	7
ghcndSelect	8
Heron	9
INLAspacetime	10
Jmatrices	10
mesh.dual	11
mesh2d	11
mesh2fem	12
mesh2projector	13
outDetect	13
paramsUtils	14
spde2precision	15
stats.inla	16
stdSubs	18
stlines	18
stModel.define	20
stModel.matrices	21
stModel.precision	22
upperPadding	22
worldMap	23
world_grid	24
<b>Index</b>	<b>25</b>

---

ar2cor	<i>Illustrative code to compute the auto-correlation for an AR2 model.</i>
--------	----------------------------------------------------------------------------

---

**Description**

Computes the auto-correlation.

**Usage**

```
ar2cor(a1, a2, k = 30)
```

**Arguments**

a1	the first auto-regression coefficient.
a2	the second auto-regression coefficient.
k	maximum lag for evaluating the auto-correlation.

**Value**

the autocorrelation as a vector or matrix, whenever a1 or a2 are scalar or vector.

**Details**

Let the second order auto-regression model defined as  $x_t + a_1 x_{t-1} + a_2 x_{t-2} = w_t$  where  $w_t \sim N(0, 1)$ .

**See Also**

[ar2precision](#)

**Examples**

```
plot(ar2cor(-1.7, 0.963), type = "o")
```

---

ar2precision

*Precision matrix for an AR2 model.*

---

**Description**

Creates a precision matrix as a sparse matrix object considering the specification stated in Details.

**Usage**

```
ar2precision(n, a)
```

**Arguments**

n                    the size of the model.  
a                    a length three vector with the coefficients. See details.

**Value**

the precision matrix as a sparse matrix object with edge correction.

**Details**

Let the second order auto-regression model be defined as

$$a_0 x_t + a_1 x_{t-1} + a_2 x_{t-2} = w_t, w_t \sim N(0, 1).$$

The n times n symmetric precision matrix Q for  $x_1, x_2, \dots, x_n$  has the following non-zero elements:

$$Q_{1,1} = Q_{n,n} = a_0^2$$

$$Q_{2,2} = Q_{n-1,n-1} = a_0^2 + a_1^2$$

$$\begin{aligned}
 Q_{1,2} &= Q_{2,1} = Q_{n-1,n} = Q_{n,n-1} = a_0 a_1 \\
 Q_{t,t} &= q_0 = a_0^2 + a_1^2 + a_2^2, t = 3, 4, \dots, n-2 \\
 Q_{t,t-1} &= Q_{t-1,t} = q_1 = a_1(a_0 + a_2), t = 3, 4, \dots, n-1 \\
 Q_{t,t-2} &= Q_{t-2,t} = q_2 = a_2 a_0, t = 3, 4, \dots, n
 \end{aligned}$$

**See Also**[ar2cor](#)**Examples**

```
ar2precision(7, c(1, -1.5, 0.9))
```

---

barrierModel.define    *Define a spacetime model object for the f() call.*

---

**Description**

Define a spacetime model object for the f() call.

**Usage**

```
barrierModel.define(
  mesh,
  barrier.triangles,
  prior.range,
  prior.sigma,
  range.fraction = 0.1,
  constr = FALSE,
  debug = FALSE,
  verbose = FALSE,
  useINLAprecomp = TRUE,
  libpath = NULL
)
```

**Arguments**

mesh	a spatial mesh
barrier.triangles	a integer vector to specify which triangles centers are in the barrier domain
prior.range	numeric vector containing U and a to define the probability statements $P(\text{range} < U) = a$ used to setup the PC-prior for range. If a = 0 then U is taken to be the fixed value for the range.
prior.sigma	numeric vector containing U and a to define the probability statements $P(\text{range} > U) = a$ used to setup the PC-prior for sigma. If a = 0 then U is taken to be the fixed value for sigma.

range.fraction	numeric to specify the fraction of the range for the barrier domain. Default value is 0.1. This has to be specified with care in order to have it small enough to make it act as barrier but not too small in order to prevent numerical issues.
constr	logical to indicate if the integral of the field over the domain is to be constrained to zero. Default value is FALSE.
debug	logical indicating if to run in debug mode.
verbose	logical indicating if to print parameter values.
useINLAPrecomp	logical indicating if is to be used shared object pre-compiled by INLA. This will not be considered if the argument libpath is provided.
libpath	string to the shared object. Default is NULL.

### Details

See the paper.

### Value

objects to be used in the f() formula term in INLA.

---

bru\_get\_mapper.stModel\_cgeneric

*Mapper object for automatic inlabru interface*

---

### Description

Return an inlabru bru\_mapper object that can be used for computing model matrices for the space-time model components. The bru\_get\_mapper() function is called by the inlabru methods to automatically obtain the needed mapper object (from inlabru 2.7.0.9001; before that, use mapper = bru\_get\_mapper(model) explicitly).

### Usage

```
bru_get_mapper.stModel_cgeneric(model, ...)
```

### Arguments

model	The model object (of class stModel_cgeneric, from stModel.define or barrierModel_cgeneric, from barrierModel.define)
...	Unused.

### Value

A bru\_mapper object of class bru\_mapper\_multi with sub-mappers space and time based on the model smesh and tmesh or mesh objects.

### See Also

[inlabru::bru\\_get\\_mapper\(\)](#)

---

cWhittleMatern                      *Computes the Whittle-Matern correlation function.*

---

### Description

This computes the correlation function as derived in Matern model, see Matern (1960) eq. (2.4.7). For  $\nu=1$ , see Whittle (1954) eq. (68). For the limiting case of  $\nu=0$ , see Besag (1981) eq. (14-15).

### Usage

```
cWhittleMatern(x, range, nu, kappa = sqrt(8 * nu)/range)
```

### Arguments

x	distance.
range	practical range (our preferred parametrization) given as $\text{range} = \sqrt{8 * \nu} / \text{kappa}$ , where $\text{kappa}$ is the scale parameter in the specialized references.
nu	process smoothness parameter.
kappa	scale parameter, commonly considered in the specialized literature.

### Value

the correlation.

### Details

Whittle, P. (1954) On Stationary Processes in the Plane. *Biometrika*, Vol. 41, No. 3/4, pp. 434-449. <http://www.jstor.org/stable/2332724>

Matern, B. (1960) Spatial Variation: Stochastic models and their application to some problems in forest surveys and other sampling investigations. PhD Thesis.

Besag, J. (1981) On a System of Two-Dimensional Recurrence Equations. *JRSS-B*, Vol. 43 No. 3, pp. 302-309. <https://www.jstor.org/stable/2984940>

### Examples

```
plot(function(x) cWhittleMatern(x, 1, 5),
      bty = "n", las = 1,
      xlab = "Distance", ylab = "Correlation"
)
plot(function(x) cWhittleMatern(x, 1, 1), add = TRUE, lty = 2)
plot(function(x) cWhittleMatern(x, 1, 0.5), add = TRUE, lty = 3)
abline(h = 0.139, lty = 3, col = gray(0.5, 0.5))
```

---

downloadUtilFiles      *Download files from the NOAA's GHCN daily data*

---

**Description**

Download files from the NOAA's GHCN daily data

**Usage**

```
downloadUtilFiles(data.dir, year = 2022, force = FALSE)
```

**Arguments**

data.dir	the folder to store the files.
year	the year of the daily weather data.
force	logical indicating if it is to force the download. If FALSE each file will be downloaded if it does not exists locally yet.

**Value**

a named character vector with the local file names: daily.data, stations.all, elevation.

---

Earth\_poly      *Function to define the boundary Earch polygon in longlat projection for a given resolution.*

---

**Description**

Function to define the boundary Earch polygon in longlat projection for a given resolution.

**Usage**

```
Earth_poly(resol = 300, crs = "+proj=moll +units=km")
```

**Arguments**

resol	is the number of subdivisions along the latitude coordinates and half the number of subdivisions along the longitude coordinates.
crs	a string with the projection. Default is the Mollweide projection with units in kilometers.

**Value**

a 'st\_sfc' object with the Earth polygon.

---

ghcndSelect	<i>Select data from the daily dataset</i>
-------------	-------------------------------------------

---

## Description

Select data from the daily dataset

## Usage

```
ghcndSelect(  
  gzfile,  
  variable = c("TMIN", "TAVG", "TMAX"),  
  station = NULL,  
  qflag = "",  
  verbose = TRUE,  
  astype = as.integer  
)
```

## Arguments

gzfile	the local filename for the daily data file file. E.g. 2023.csv.gz from <a href="https://www.ncei.noaa.gov/pub/data/ghcn/daily/by_year/">https://www.ncei.noaa.gov/pub/data/ghcn/daily/by_year/</a> see references below.
variable	string with the variable name(s) to be selected
station	string (vector) with the station(s) to be selected
qflag	a string with quality control flag(s)
verbose	logical indicating if progress is to be printed
astype	function to convert data to a class, default is set to convert the data to integer.

## Value

if more than one variable, it returns an array whose dimensions are days, stations, variables. If one variable, then it returns a matrix whose dimensions are days, stations.

## Details

The default selects TMIN, TAVG and TMAX and return it as integer because the original data is also integer with units in 10 Celcius degrees.

## Warning

It can take time to execute if, for example, the data.table package is not available.



**References**

Menne, M., Durre, I., Vose, R., Gleason, B. and Houston, T. (2012) An overview of the global historical climatology network-daily database. *Journal of Atmospheric and Oceanic Technology*, 897–910.

---

Heron

*Internal util functions for polygon properties.*

---

**Description**

This computes the area of a triangle given its three coordinates.

**Usage**

Heron(x, y)

Area(x, y)

s2trArea(tr, R = 1)

flatArea(tr)

Stiffness(tr)

**Arguments**

x, y	coordinate vectors.
tr	the triangle coordinates
R	the radius of the spherical domain

**Details**

Function used internally to compute the area of a triangle.

**Value**

the area of a 2d triangle  
the area of a 2d polygon  
the area of a triangle in S2  
the area of a triangle  
the stiffness matrix for a triangle

**Warning**

Internal functions, not exported.

---

`INLAspacetime`*Spatial and Spatio-Temporal Models using INLA*

---

**Description**

This package main purpose is to provide user friendly functions to fit temporal, spatial and space-time models using the INLA software available at [www.r-inla.org](http://www.r-inla.org) as well the inlabru package available

**Usage**

```
INLAspacetime()
```

**Value**

opens the Vignettes directory on a browser

---

`Jmatrices`*The 2nd order temporal matrices with boundary correction*

---

**Description**

The 2nd order temporal matrices with boundary correction

**Usage**

```
Jmatrices(tmesh)
```

**Arguments**

`tmesh`            Temporal mesh

**Details**

Temporal GMRF representation with stationary boundary conditions as in Appendix E of the paper.

**Value**

return a list of temporal finite element method matrices for the supplied mesh.

---

mesh.dual	<i>Extracts the dual of a mesh object.</i>
-----------	--------------------------------------------

---

**Description**

Extracts the dual of a mesh object.

**Usage**

```
mesh.dual(
  mesh,
  returnclass = c("list", "sf", "sv", "SP"),
  mc.cores = getOption("mc.cores", 2L)
)
```

**Arguments**

mesh	a 2d mesh object.
returnclass	if 'list' return a list of polygon coordinates, if "sf" return a 'sf' sfc_multipolygon object, if "sv" return a 'terra', SpatVector object, if "SP" return a 'sp' SpatialPolygons object.
mc.cores	number of threads to be used.

**Value**

one of the three in 'returnclass'

---

mesh2d	<i>Illustrative code for building a mesh in 2d domain.</i>
--------	------------------------------------------------------------

---

**Description**

Creates a mesh object. This is just a test code. For efficient, reliable and general code use the **fmesher** package.

**Usage**

```
mesh2d(loc, domain, max.edge, offset, SP = TRUE)
```

**Arguments**

loc	a two column matrix with location coordinates.
domain	a two column matrix defining the domain.
max.edge	the maximum edge length.
offset	the length of the outer extension.
SP	logical indicating if the output will include the SpatialPolygons.

**Value**

a mesh object.

**Warning**

This is just for illustration purpose and one should consider the efficient function available at the INLA package.

---

mesh2fem

*Illustrative code for Finite Element matrices of a mesh in 2d domain.*

---

**Description**

Illustrative code for Finite Element matrices of a mesh in 2d domain.

Illustrative code for Finite Element matrices when some triangles are in a barrier domain.

**Usage**

```
mesh2fem(mesh, order = 2, barrier.triangles = NULL)
```

```
mesh2fem.barrier(mesh, barrier.triangles = NULL)
```

**Arguments**

mesh            a 2d mesh object.

order           the desired order.

barrier.triangles  
                 integer index to specify the triangles in the barrier domain

**Value**

a list object containing the FE matrices.

a list object containing the FE matrices for the barrier problem.

---

mesh2projector	<i>Illustrative code to build the projector matrix for SPDE models.</i>
----------------	-------------------------------------------------------------------------

---

**Description**

Creates a projector matrix object.

**Usage**

```
mesh2projector(
  mesh,
  loc = NULL,
  lattice = NULL,
  xlim = NULL,
  ylim = NULL,
  dims = c(100, 100)
)
```

**Arguments**

mesh	a 2d mesh object.
loc	a two columns matrix with the locations to project for.
lattice	Unused; feature not supported by this illustration.
xlim, ylim	vector with the boundary limits.
dims	the number of subdivisions over each boundary limits.

**Value**

the projector matrix as a list with sparse matrix object at x\$proj\$A..

**Warning**

This is just for illustration purpose and one should consider the efficient functions available in the INLA and inlabru packages, e.g. `inlabru::fm_evaluator`.

---

outDetect	<i>Detect outliers in a time series considering the raw data and a smoothed version of it.</i>
-----------	------------------------------------------------------------------------------------------------

---

**Description**

Detect outliers in a time series considering the raw data and a smoothed version of it.

**Usage**

```
outDetect(x, weights = NULL, ff = c(7, 7))
```

**Arguments**

**x** numeric vector

**weights** non-increasing numeric vector used as weights for computing a smoothed vector as a rooling window average. Default is null and then  $w_j$  is proportional to  $j$  in the equation in the Details below.

**ff** numeric length two vector with the factors used to consider how many times the standard deviation one data point is out to be considered as an outlier.

**Value**

logical vector indicating if the data is an outlier with attributes as detailed bellow.

- attr(, 'm') is the mean of  $x$ .
- attr(, 's') is the standard deviation of  $x$ .
- attr(, 'ss') is the standard deviation for the smoothed data  $y_t$  that is defined as

$$y_t = \sum_{k=j}^h w_j * (x_{t-j} + x_{t+j})/2$$

Both  $s$  and  $ss$  are used to define outliers if

$$|x_t - m|/s > ff_1 \text{ or } |x_t - y_t|/ss > ff_2$$

- attr(, 'xs') the smoothed time series  $y_t$

---

paramsUtils	<i>Functions to help converting from/to user/internal parametrization. The internal parameters are 'gamma_s', 'gamma_t', 'gamma_E' The user parameters are 'r_s', 'r_t', 'sigma'</i>
-------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

**Description**

Functions to help converting from/to user/internal parametrization. The internal parameters are 'gamma\_s', 'gamma\_t', 'gamma\_E' The user parameters are 'r\_s', 'r\_t', 'sigma'

Convert from user parameters to SPDE parameters

Convert from SPDE parameters to user parameters

**Usage**

```
lgsConstant(lg.s, alpha, smanifold)
```

```
params2gammas(lparams, alpha.t, alpha.s, alpha.e, smanifold = "R2")
```

```
gammas2params(lgammas, alpha.t, alpha.s, alpha.e, smanifold = "R2")
```

**Arguments**

lg.s	the logarithm of the SPDE parameter $\gamma_s$
alpha	the resulting spatial order.
smanifold	spatial domain manifold, which could be "S1", "S2", "R1", "R2" and "R3".
lparams	$\log(\text{spatial range, temporal range, sigma})$
alpha.t	temporal order of the SPDE
alpha.s	spatial order of the spatial differential operator in the non-separable part.
alpha.e	spatial order of the spatial differential operator in the separable part.
lgammas	numeric of length 3 with $\log(\gamma_k)$ model parameters. The parameter order is $\log(\text{gamma.s, gamma.t, gamma.e})$

**Details**

See equation (23) in the paper.

See equations (19), (20) and (21) in the paper.

See equations (19), (20) and (21) in the paper.

**Value**

the part of  $\sigma$  from the spatial constant and  $\gamma_s$ .

$\log(\text{gamma.s, gamma.t, gamma.e})$

$\log(\text{spatial range, temporal range, sigma})$

**Examples**

```
params2gammas(log(c(1, 1, 1)), 1, 2, 1, "R2")
gammas2params(log(c(0, 0, 0)), 1, 2, 1, "R2")
```

---

```
spde2precision
```

*Illustrative code to build the precision matrix for SPDE kind models.*

---

**Description**

Creates a precision matrix as a sparse matrix object. For general code look at the functions in the INLA package.

**Usage**

```
spde2precision(kappa, fem, alpha)
```

**Arguments**

kappa	the scale parameter.
fem	a list containing the Finite Element matrices.
alpha	the smoothness parameter.

**Value**

the precision matrix as a sparse matrix object.

**Warning**

This is just for illustration purpose and one should consider the efficient function available in the INLA package.

---

stats.inla

*To retrieve goodness of fit statistics for a specific model class.*

---

**Description**

Extracts dic, waic and log-cpo from an output returned by the inla function from the INLA package or by the bru function from the inlabru package, and computes log-po, mse, mae, crps and scrps for a given input. A summary is applied considering the user imputed function, which by default is the mean.

**Usage**

```
stats.inla(m, i = NULL, y, fsummarize = mean)
```

**Arguments**

m                    an inla output object.  
 i                    an index to subset the estimated values.  
 y                    observed to compare against.  
 fsummarize        the summary function, the default is `base::mean()`.

**Value**

A named numeric vector with the extracted statistics.

**Details**

It assumes Gaussian posterior predictive distributions! Considering the defaults, for  $n$  observations,  $y_i, i = 1, 2, \dots, n$ , we have

. dic

$$\sum_i d_i/n$$

where  $d_i$  is the dic computed for observation  $i$ .

. waic

$$\sum_i w_i/n$$

where  $w_i$  is the waic computed for observation  $i$ .



. lcpo

$$-\sum_i \log(p_i)/n$$

where  $p_i$  is the cpo computed for observation  $i$ .

For the log-po, crps, and scrps scores it assumes a Gaussian predictive distribution for each observation  $y_i$  which the following definitions:  $z_i = (y_i - \mu_i)/\sigma_i$ ,  $\mu_i$  is the posterior mean for the linear predictor,  $\sigma_i = \sqrt{v_i + 1/\tau_y}$ ,  $\tau_y$  is the observation posterior mean,  $v_i$  is the posterior variance of the linear predictor for  $y_i$ .

Then we consider  $\phi()$  the density of a standard Gaussian variable and  $\psi()$  the corresponding Cumulative Probability Distribution.

. lpo

$$-\sum_i \log(\phi(z_i))/n$$

. crps

$$\sum_i r_i/n$$

where

$$r_i = \sigma_i/\sqrt{\pi} - 2\sigma_i\phi(z_i) + (y_i - \mu_i)(1 - 2\psi(z_i))$$

. scrps

$$\sum_i s_i/n$$

where

$$s_i = -\log(2\sigma_i/\sqrt{\pi})/2 - \sqrt{\pi}(\phi(z_i) - \sigma_i z_i/2 + z_i\psi(z_i))$$

### Warning

All the scores are negatively oriented which means that smaller scores are better.

### References

Held, L. and Schrödle, B. and Rue, H. (2009). Posterior and Cross-validatory Predictive Checks: A Comparison of MCMC and INLA. *Statistical Modelling and Regression Structures* pp 91–110. [https://link.springer.com/chapter/10.1007/978-3-7908-2413-1\\_6](https://link.springer.com/chapter/10.1007/978-3-7908-2413-1_6).

Bolin, D. and Wallin, J. (2022) Local scale invariance and robustness of proper scoring rules. *Statistical Science*. doi:10.1214/22STS864.

---

stdSubs *To check unusual low/high variance segments*

---

### Description

To check unusual low/high variance segments

### Usage

```
stdSubs(x, nsub = 12, fs = 15)
```

### Arguments

x	numeric vector
nsub	number for the segments length
fs	numeric to use for detecting too high or too low local standard deviations.

### Value

logical indicating if any of the st are fs times lower/higher the average of st, where is returned as an attribute:

- attr(, 'st') numeric vector with the standard deviation at each segment of the data.

---

stlines *To visualize time series over space.*

---

### Description

To visualize time series over space.

### Usage

```
stlines(
  stdata,
  spatial,
  group = NULL,
  nmax.group = NULL,
  xscale = 1,
  yscale = 1,
  colour = NULL,
  ...
)

stpoints(
```

```

    stdata,
    spatial,
    group = NULL,
    nmax.group = NULL,
    xscale = 1,
    yscale = 1,
    colour = NULL,
    ...
)

```

### Arguments

stdata	matrix with the data, each column is a location.
spatial	an object with one of class defined in the sp package.
group	an integer vector indicating to which spatial unit each time series belongs to. Default is NULL and then it is assumed that each time series belongs to each spatial unit.
nmax.group	an integer indicating the maximum number of time series to be plotted over each spatial unit. Default is NULL, so all will be drawn.
xscale	numeric to define a scaling factor in the horizontal direction.
yscale	numeric to define a scaling factor in the vertical direction.
colour	color (may be a vector, one for each time series). Default is NULL and it will generate colors considering the average of each time series. These automatic colors are defined using the <code>rgb()</code> function with <code>alpha=0.5</code> . It considers the relative rank of each time series mean, $r$ . $r$ is then used for red, $1-r$ is used for blue and a triangular function, $1-2* 1-r/2 $ , is considered for green. That is, time series with mean among the lowest time series averages are shown in blue and those among the highest temperatures are shown in red. The transition from blue to red goes so that the intermediate ones are shown in light green.
...	further arguments to be passed for the lines function.

### Details

Scaling the times series is needed before drawing it over the map. The area of the bounding box for the spatial object divided by the number of locations is the standard scaling factor. This is further multiplied by the user given `xscale` and `yscale`.

### Value

add lines to an existing plot

### Functions

- `stlines()`: each time series over the map centered at the location.
- `stpoints()`: each time series over the map centered at the location.

**Warning**

if there are too many geographical locations, it will not look good

---

stModel.define                      *Define a spacetime model object for the f() call.*

---

**Description**

Define a spacetime model object for the f() call.

**Usage**

```
stModel.define(
  smesh,
  tmesh,
  model,
  control.priors,
  constr = FALSE,
  debug = FALSE,
  verbose = FALSE,
  useINLAp recomp = TRUE,
  libpath = NULL
)
```

**Arguments**

smesh	a spatial mesh
tmesh	a temporal mesh
model	a three characters string to specify the smoothness alpha (each one as integer) parameters. Currently it considers the 102, 121, 202 and 220 models.
control.priors	a named list with parameter priors. E.g. prior.rs, prior.rt and prior.sigma as vectors with length two (U, a) to define the corresponding PC-prior such that $P(r_s < U) = a$ , $P(r_t < U) = a$ or $P(\text{sigma} > U) = a$ . If $a=0$ then U is taken to be the fixed value of the parameter.
constr	logical to indicate if the integral of the field over the domain is to be constrained to zero. Default value is FALSE.
debug	logical indicating if to run in debug mode.
verbose	logical indicating if to print parameter values.
useINLAp recomp	logical indicating if is to be used shared object pre-compiled by INLA. Not considered if libpath is provided.
libpath	string to the shared object. Default is NULL.

**Details**

See the paper.

**Value**

objects to be used in the  $f()$  formula term in INLA.

---

<code>stModel.matrices</code>	<i>Define the spacetime model matrices.</i>
-------------------------------	---------------------------------------------

---

**Description**

This function computes all the matrices needed to build the precision matrix for spatio-temporal model, as in Lindgren et. al. (2023)

**Usage**

```
stModel.matrices(smesh, tmesh, model, constr = FALSE)
```

**Arguments**

<code>smesh</code>	a mesh object over the spatial domain.
<code>tmesh</code>	a mesh object over the time domain.
<code>model</code>	a string identifying the model. So far we have the following models: '102', '121', '202' and '220' models.
<code>constr</code>	logical to indicate if the integral of the field over the domain is to be constrained to zero. Default value is FALSE.

**Details**

See the paper for details.

**Value**

a list containing needed objects for model definition.

1. 'manifold' to specify the a string with the model identification
2. a length three vector with the constants  $c_1$ ,  $c_2$  and  $c_3$
3. the vector  $d$
4. the matrix  $T$
5. the model matrices  $M_1, \dots, M_m$

---

`stModel.precision`      *Spacetime precision matrix.*

---

### Description

To build the the precision matrix for a spacetime model given the temporal and the spatial meshes.

### Usage

```
stModel.precision(smash, tmesh, model, theta, verbose = FALSE)
```

### Arguments

<code>smash</code>	a mesh object over the spatial domain.
<code>tmesh</code>	a mesh object over the time domain.
<code>model</code>	a string identifying the model. So far we have the following models: '102', '121', '202' and '220' models.
<code>theta</code>	numeric vector of length three with $\log(\gamma_s, \gamma_t, \gamma_e)$ .
<code>verbose</code>	logical to print intermediate objects.

### Value

a (sparse) precision matrix, as in Lindgren et. al. (2023)

---

`upperPadding`      *Prepare a matrix or a list of matrices for use in some 'cgeneric' code.*

---

### Description

Define a graph of the union of the supplied matrices and return the row ordered diagonal plus upper triangle after padding with zeroes each one so that all the returned matrices have the same pattern.

### Usage

```
upperPadding(M, relative = FALSE)
```

### Arguments

<code>M</code>	a matrix or a list of matrices
<code>relative</code>	logical. If a list of matrices is supplied, it indicates if it is to be returned a relative index and the value for each matrix. See details.

**Details**

If `relative=FALSE`, each columns of 'xx' is the elements of the corresponding matrix after being padded to fill the pattern of the union graph. If `relative=TRUE`, each element of 'xx' would be a list with a relative index, 'r', for each non-zero elements of each matrix is returned relative to the union graph, the non-lower elements, 'x', of the corresponding matrix, and a vector, 'o', with the number of non-zero elements for each line of each resulting matrix.

**Value**

If a unique matrix is given, return the upper triangle considering the 'T' representation in the Matrix package. If a list of matrices is given, return a list of two elements: 'graph' and 'xx'. The 'graph' is the union of the graph from each matrix. If `relative=FALSE`, 'xx' is a matrix with number of column equals the number of matrices imputed. If `relative=TRUE`, it is a list of length equal the number of matrices imputed. See details.

**Examples**

```
A <- sparseMatrix(
  i = c(1, 1, 2, 3, 3, 5),
  j = c(2, 5, 3, 4, 5, 5),
  x = -(0:5), symmetric = TRUE
)
A
upperPadding(A)
B <- Diagonal(nrow(A), -colSums(A))
list(a = A, a = B)
upperPadding(list(a = A, b = B))
upperPadding(list(a = A, b = B), relative = TRUE)
```

---

worldMap

*Helper functions to retrieve the world map, a world polygon, and create grid centers.*


---

**Description**

Retrieve the map of the countries

**Usage**

```
worldMap(
  crs = "+proj=moll +units=km",
  scale = "medium",
  returnclass = c("sf", "sv")
)
```

**Arguments**

crs	a string with the projection. Default is the Mollweide projection with units in kilometers.
scale	see the help from the 'ne_countries' of the 'rnaturalearth' package
returnclass	defined by the <a href="#">sf</a> package if returnclass = "sf" or <a href="#">terra</a> if returnclass = "sv".

**Value**

the world map of the countries.

**References**

The land and ocean maps are obtained with the 'rnaturalearth' package.

---

world_grid	<i>Define a regular grid in 'Mollweide' projection, with units in kilometers.</i>
------------	-----------------------------------------------------------------------------------

---

**Description**

Define a regular grid in 'Mollweide' projection, with units in kilometers.

**Usage**

```
world_grid(size = 50, domain)
```

**Arguments**

size	the (in kilometers) of the grid cells.
domain	if provided it should be an sf or sfc object. In this case, the grid cells with centers falling inside will be retrieved.

**Value**

a 'sf' points object with the centers of a grid set within Earth (and the supplied domain)



# Index

ar2cor, [2](#), [4](#)  
ar2precision, [3](#), [3](#)  
Area (Heron), [9](#)  
  
barrierModel.define, [4](#)  
base::mean(), [16](#)  
bru\_get\_mapper  
    (bru\_get\_mapper.stModel\_cgeneric),  
    [5](#)  
bru\_get\_mapper.stModel\_cgeneric, [5](#)  
  
cWhittleMatern, [6](#)  
  
downloadUtilFiles (downloadUtilFiles), [7](#)  
downloadUtilFiles, [7](#)  
  
Earth\_poly, [7](#)  
  
flatArea (Heron), [9](#)  
  
gammas2params (paramsUtils), [14](#)  
ghcndSelect, [8](#)  
  
Heron, [9](#)  
  
inlabru::bru\_get\_mapper(), [5](#)  
INLAspacetime, [10](#)  
  
Jmatrices, [10](#)  
  
lgsConstant (paramsUtils), [14](#)  
  
mesh.dual, [11](#)  
mesh2d, [11](#)  
mesh2fem, [12](#)  
mesh2projector, [13](#)  
  
outDetect, [13](#)  
  
params2gammas (paramsUtils), [14](#)  
paramsUtils, [14](#)  
  
s2trArea (Heron), [9](#)  
  
sf, [24](#)  
spde2precision, [15](#)  
stats.inla, [16](#)  
stdSubs, [18](#)  
Stiffness (Heron), [9](#)  
stlines, [18](#)  
stModel.define, [20](#)  
stModel.matrices, [21](#)  
stModel.precision, [22](#)  
stpoints (stlines), [18](#)  
  
terra, [24](#)  
  
upperPadding, [22](#)  
  
world\_grid, [24](#)  
worldMap, [23](#)