Package ‘bigDM’

June 27, 2022

Type Package

Title Scalable Bayesian Disease Mapping Models for High-Dimensional Data

Version 0.4.2

Date 2022-06-27

Maintainer Aritz Adin <aritz.adin@unavarra.es>

Description Implements several spatial and spatio-temporal scalable disease mapping models for high-dimensional count data using the INLA technique for approximate Bayesian inference in latent Gaussian models (Orozco-Acosta et al., 2021 <doi:10.1016/j.spasta.2021.100496> and Orozco-Acosta et al., 2022 <arXiv:2201.08323>). The creation and develpment of this package has been supported by Project MTM2017-82553-R (AEI/FEDER, UE) and Project PID2020-113125RB-I00/MCIN/AEI/10.13039/501100011033. It has also been partially funded by the Public University of Navarra (project PJUPNA2001).

URL https://github.com/spatialstatisticsupna/bigDM

BugReports https://github.com/spatialstatisticsupna/bigDM/issues

Depends R (>= 4.0.0)

Additional_repositories https://inla.r-inla-download.org/R/stable

Imports crayon, future, future.apply, MASS, Matrix, methods, parallel, RColorBrewer, Rdpack, sf, spatialreg, spdep, stats, utils, rlist

Suggests bookdown, INLA (>= 21.11.22), knitr, rmarkdown, testthat (>= 3.0.0), tmap

RdMacros Rdpack

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.2.0

Config/testthat/edition 3

NeedsCompilation no

Author Aritz Adin [aut, cre] (<https://orcid.org/0000-0003-3232-6147>),
Erick Orozco-Acosta [aut] (<https://orcid.org/0000-0002-1170-667X>),
Maria Dolores Ugarte [aut] (<https://orcid.org/0000-0002-3505-8400>)

Repository CRAN

Date/Publication 2022-06-27 09:10:02 UTC

R topics documented:

`bigDM-package` *Scalable Bayesian Disease Mapping Models for High-Dimensional Data*

Description

This package implements several (scalable) spatial and spatio-temporal Poisson mixed models for high-dimensional areal count data in a fully Bayesian setting using the integrated nested Laplace approximation (INLA) technique.

Details

Below, there is a list with a brief overview of all package functions:

`add_neighbour`	Adds isolated areas (polygons) to its nearest neighbour
`CAR_INLA`	Fits several spatial CAR models for high-dimensional count data
`clustering_partition`	Obtain a spatial partition using the DBSC algorithm
`connect_subgraphs`	Merges disjoint connected subgraphs
`divide_carto`	Divides the spatial domain into subregions
`mergeINLA`	Merges `inla` objects for partition models
`random_partition`	Defines a random partition of the spatial domain based on a regular grid
`STCAR_INLA`	Fits several spatio-temporal CAR models for high-dimensional count data
———————-	————————————————————————————-

Author(s)

Maintainer: Aritz Adin <aritz.adin@unavarra.es>

This work has been supported by Project MTM2017-82553-R (AEI/FEDER, UE) and Project PID2020-113125RB-I00/MCIN/AEI/10.13039/501100011033. It has also been partially funded by the Public University of Navarra (project PJUPNA2001) and by la Caixa Foundation (ID 1000010434), Caja Navarra Foundation and UNED Pamplona, under agreement LCF/PR/PR15/51100007 (project REF P/13/20).

References

Orozco-Acosta E, Adin A, Ugarte MD (2021). “Scalable Bayesian modeling for smoothing disease mapping risks in large spatial data sets using INLA.” *Spatial Statistics*, **41**, 100496. doi: 10.1016/j.spasta.2021.100496.

Orozco-Acosta E, Adin A, Ugarte MD (2022). “Parallel and distributed Bayesian modelling for analysing high-dimensional spatio-temporal count data.” https://arxiv.org/abs/2201.08323.

See Also

See the following vignettes for further details and examples using this package:

1. bigDM: fitting spatial models
2. bigDM: parallel and distributed modelling
3. bigDM: fitting spatio-temporal models

Examples

```
## See the examples for CAR_INLA and STCAR_INLA functions
```

add_neighbour *Add isolated areas (polygons) to its nearest neighbour*

Description

The function returns a neighbour list of class nb and its associated spatial adjacency matrix computed by adding isolated areas to its nearest neighbour (in terms of Euclidean distance between centroids) using the knearneigh function of ’spdep’ package.

Usage

```
add_neighbour(carto, nb = NULL, plot = FALSE)
```

Arguments

carto — object of class SpatialPolygonsDataFrame or sf.

nb — optional argument with the neighbour list of class nb. If NULL (default), this object is computed from the carto argument.

plot — logical value (default FALSE), if TRUE then the computed neighbourhood graph is plotted.

Value

This function returns a list with the following two elements:

- nb: the modified neighbours's list
- W: associated spatial adjacency matrix of class CsparseMatrix

Examples

```
library(spdep)

## Load the Spanish colorectal cancer mortality data ##
data(Carto_SpainMUN)

## Compute the neighbour list from spatial polygons ##
nb_SpainMUN <- poly2nb(Carto_SpainMUN)
summary(nb_SpainMUN) # 1 region with no links

## Add isolated area to its nearest neighbour ####
carto.mod <- add_neighbour(carto=Carto_SpainMUN, nb=nb_SpainMUN)
summary(carto.mod$nb) # 0 region with no links
```

Carto_SpainMUN *Spanish colorectal cancer mortality data*

Description

sf object containing the polygons of the municipalities of continental Spain and simulated colorectal cancer mortality data.

Usage

```
Carto_SpainMUN
```

Format

Formal class `sf`; the data contains a data.frame with 7907 rows and 11 variables.

- ID: character vector of geographic identifiers
- name: character vector of municipality names
- lat: numeric vector of longitude values
- long: numeric vector of latitude values
- area: municipality polygon areas in square meters
- perimeter: municipality polygon perimeters in degree units
- obs: observed number of cases
- exp: expected number of cases
- SMR: standardized mortality ratios
- region: character vector of autonomous regions
- geometry: sfc_MULTIPOLYGON

`CAR_INLA` *Fit a (scalable) spatial Poisson mixed model to areal count data, where several CAR prior distributions can be specified for the spatial random effect.*

Description

Fit a spatial Poisson mixed model to areal count data. The linear predictor is modelled as

$$\log r_i = \alpha + \mathbf{x_i}^{'}\beta + \xi_i, \quad \text{for} \quad i = 1, \ldots, n;$$

where α is a global intercept, $\mathbf{x_i}^{'} = (x_{i1}, \ldots, x_{ip})$ is a p-vector of standardized covariates in the i-th area, $\beta = (\beta_1, \ldots, \beta_p)$ is the p-vector of fixed effects coefficients, and ξ_i is a spatially structured random effect. Several conditional autoregressive (CAR) prior distributions can be specified for the spatial random effect, such as the intrinsic CAR prior (Besag et al. 1991), the convolution or BYM prior (Besag et al. 1991), the CAR prior proposed by Leroux et al. (1999), and the reparameterization of the BYM model given by Dean et al. (2001) named BYM2 (Riebler et al. 2016).

If covariates are included in the model, two different approaches can be used to address the potential confounding issues between the fixed effects and the spatial random effects of the model: restricted regression and the use of orthogonality constraints. At the moment, only continuous covariates can be included in the model as potential risk factors, which are automatically standardized before fitting the model. See Adin et al. (2021) for further details.

Three main modeling approaches can be considered:

- the usual model with a global spatial random effect whose dependence structure is based on the whole neighbourhood graph of the areal units (`model="global"` argument)

- a Disjoint model based on a partition of the whole spatial domain where independent spatial CAR models are simultaneously fitted in each partition (`model="partition"` and `k=0` arguments)
- a modelling approach where *k*-order neighbours are added to each partition to avoid border effects in the Disjoint model (`model="partition"` and `k>0` arguments).

For both the Disjoint and k-order neighbour models, parallel or distributed computation strategies can be performed to speed up computations by using the 'future' package (Bengtsson 2021).

Inference is conducted in a fully Bayesian setting using the integrated nested Laplace approximation (INLA; Rue et al. (2009)) technique through the R-INLA package (https://www.r-inla.org/). For the scalable model proposals (Orozco-Acosta et al. 2021), approximate values of the Deviance Information Criterion (DIC) and Watanabe-Akaike Information Criterion (WAIC) can also be computed.

Usage

```
CAR_INLA(
  carto = NULL,
  ID.area = NULL,
  ID.group = NULL,
  O = NULL,
  E = NULL,
  X = NULL,
  confounding = NULL,
  W = NULL,
  prior = "Leroux",
  model = "partition",
  k = 0,
  strategy = "simplified.laplace",
  PCpriors = FALSE,
  seed = NULL,
  n.sample = 1000,
  compute.fixed = FALSE,
  compute.DIC = TRUE,
  save.models = FALSE,
  plan = "sequential",
  workers = NULL,
  merge.strategy = "original"
)
```

Arguments

`carto`	object of class `SpatialPolygonsDataFrame` or `sf`. This object must contain at least the target variables of interest specified in the arguments `ID.area`, `O` and `E`.
`ID.area`	character; name of the variable which contains the IDs of spatial areal units.
`ID.group`	character; name of the variable which contains the IDs of the spatial partition (grouping variable). Only required if `model="partition"`.

`O` character; name of the variable which contains the observed number of disease cases for each areal units.

`E` character; name of the variable which contains either the expected number of disease cases or the population at risk for each areal unit.

`X` a character vector containing the names of the covariates within the `carto` object to be included in the model as fixed effects, or a matrix object playing the role of the fixed effects design matrix. For the latter case, the row names must match with the IDs of the spatial units defined by the `ID.area` variable. If `X=NULL` (default), only a global intercept is included in the model as fixed effect.

`confounding` one of either `NULL`, `"restricted"` (restricted regression) or `"constraints"` (orthogonal constraints), which specifies the estimation method used to alleviate spatial confounding between fixed and random effects. If only an intercept is considered in the model (`X=NULL`), the default value `confounding=NULL` will be set. At the moment, only works for the *Global model* (specified through the `model="global"` argument).

`W` optional argument with the binary adjacency matrix of the spatial areal units. If `NULL` (default), this object is computed from the `carto` argument (two areas are considered as neighbours if they share a common border).

`prior` one of either `"Leroux"` (default), `"intrinsic"`, `"BYM"` or `"BYM2"`, which specifies the prior distribution considered for the spatial random effect.

`model` one of either `"global"` or `"partition"` (default), which specifies the *Global model* or one of the scalable model proposal's (*Disjoint model* and *k-order neighbourhood model*, respectively).

`k` numeric value with the neighbourhood order used for the partition model. Usually k=2 or 3 is enough to get good results. If k=0 (default) the *Disjoint model* is considered. Only required if `model="partition"`.

`strategy` one of either `"gaussian"`, `"simplified.laplace"` (default), `"laplace"` or `"adaptive"`, which specifies the approximation strategy considered in the `inla` function.

`PCpriors` logical value (default `FALSE`); if `TRUE` then penalised complexity (PC) priors are used for the precision parameter of the spatial random effect. Only works if arguments `prior="intrinsic"` or `prior="BYM2"` are specified.

`seed` numeric (default `NULL`); control the RNG of the `inla.qsample` function. See `help(inla.qsample)` for further information.

`n.sample` numeric; number of samples to generate from the posterior marginal distribution of the risks. Default to 1000.

`compute.fixed` logical value (default `FALSE`); if `TRUE` then the overall log-risk α is computed. Only works if `k=0` argument (*Disjoint model*) is specified.

`compute.DIC` logical value; if `TRUE` (default) then approximate values of the Deviance Information Criterion (DIC) and Watanabe-Akaike Information Criterion (WAIC) are computed.

`save.models` logical value (default `FALSE`); if `TRUE` then a list with all the `inla` submodels is saved in '/temp/' folder, which can be used as input argument for the `mergeINLA` function.

`plan` one of either `"sequential"` or `"cluster"`, which specifies the computation strategy used for model fitting using the 'future' package. If `plan="sequential"` (default) the models are fitted sequentially and in the current R session (local machine). If `plan="cluster"` the models are fitted in parallel on external R sessions (local machine) or distributed in remote compute nodes.

`workers` character or vector (default `NULL`) containing the identifications of the local or remote workers where the models are going to be processed. Only required if `plan="cluster"`.

`merge.strategy` one of either `"mixture"` or `"original"` (default), which specifies the merging strategy to compute posterior marginal estimates of relative risks. See `mergeINLA` for further details.

Details

For a full model specification and further details see the vignettes accompanying this package.

Value

This function returns an object of class `inla`. See the `mergeINLA` function for details.

References

Adin A, Goicoa T, Hodges JS, Schnell P, Ugarte MD (2021). "Alleviating confounding in spatio-temporal areal models with an application on crimes against women in India." *Statistical Modelling*, 1471082X211015452. doi: 10.1177/1471082X211015452.

Bengtsson H (2021). "A unifying framework for parallel and distributed processing in R using futures." *The R Journal (accepted article)*. `https://journal.r-project.org/archive/2021/RJ-2021-048/index.html`.

Besag J, York J, Mollié A (1991). "Bayesian image restoration, with two applications in spatial statistics." *Annals of the Institute of Statistical Mathematics*, **43**(1), 1–20. doi: 10.1007/bf00116466.

Dean CB, Ugarte MD, Militino AF (2001). "Detecting interaction between random region and fixed age effects in disease mapping." *Biometrics*, **57**(1), 197–202. doi: 10.1111/j.0006341x.2001.00197.x.

Leroux BG, Lei X, Breslow N (1999). "Estimation of disease rates in small areas: A new mixed model for spatial dependence." In Halloran ME, Berry D (eds.), *Statistical Models in Epidemiology, the Environment, and Clinical Trials*, 179–191. Springer-Verlag: New York.

Riebler A, Sørbye SH, Simpson D, Rue H (2016). "An intuitive Bayesian spatial model for disease mapping that accounts for scaling." *Statistical methods in medical research*, **25**(4), 1145–1165. doi: 10.1177/0962280216660421.

Rue H, Martino S, Chopin N (2009). "Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **71**(2), 319–392. doi: 10.1111/j.14679868.2008.00700.x.

Orozco-Acosta E, Adin A, Ugarte MD (2021). "Scalable Bayesian modeling for smoothing disease mapping risks in large spatial data sets using INLA." *Spatial Statistics*, **41**, 100496. doi: 10.1016/j.spasta.2021.100496.

Examples

```
## Not run:
if(require("INLA", quietly=TRUE)){

  ## Load the Spain colorectal cancer mortality data ##
  data(Carto_SpainMUN)

  ## Fit the Global model with a Leroux CAR prior distribution ##
  Global <- CAR_INLA(carto=Carto_SpainMUN, ID.area="ID", O="obs", E="exp",
                     prior="Leroux", model="global", strategy="gaussian")

  summary(Global)

  ## Fit the Disjoint model with a Leroux CAR prior distribution ##
 Disjoint <- CAR_INLA(carto=Carto_SpainMUN, ID.area="ID", ID.group="region", O="obs", E="exp",
                       prior="Leroux", model="partition", k=0, strategy="gaussian")
  summary(Disjoint)

  ## Fit the 1st order neighbourhood model with a Leroux CAR prior distribution ##
 order1 <- CAR_INLA(carto=Carto_SpainMUN, ID.area="ID", ID.group="region", O="obs", E="exp",
                     prior="Leroux", model="partition", k=1, strategy="gaussian")
  summary(order1)

  ## Fit the 2nd order neighbourhood model with a Leroux CAR prior distribution ##
 order2 <- CAR_INLA(carto=Carto_SpainMUN, ID.area="ID", ID.group="region", O="obs", E="exp",
                     prior="Leroux", model="partition", k=2, strategy="gaussian")
  summary(order2)
}

## End(Not run)
```

clustering_partition *Obtain a partition of the spatial domain using the density-based spatial clustering (DBSC) algorithm described in Santafé et al. (2021)*

Description

The function takes an object of class `SpatialPolygonsDataFrame` or `sf` and defines a spatial partition using the DBSC algorithm described in Santafé et al. (2021).

Usage

```
clustering_partition(
  carto,
  ID.area = NULL,
  var = NULL,
  n.cluster = 10,
  min.size = NULL,
  W = NULL,
  l = 1,
  Wk = NULL,
  distance = "euclidean",
  verbose = TRUE
)
```

Arguments

`carto` object of class `SpatialPolygonsDataFrame` or `sf`.

`ID.area` character; name of the variable which contains the IDs of spatial areal units.

`var` character; name of the variable which contains the data of interest to compute spatial clusters, usually the vector of log-SMR.

`n.cluster` numeric; value to fix the number of cluster centers in the DBSC algorithm. Default to 10.

`min.size` numeric (default `NULL`); value to fix the minimum size of areas in each spatial partition.

`W` optional argument with the binary adjacency matrix of the spatial areal units. If `NULL` (default), this object is computed from the `carto` argument (two areas are considered as neighbours if they share a common border).

`l` numeric value with the neighbourhood order used to assign areas to each cluster. If `k=1` (default), only areas that share a common border are considered.

`Wk` previously computed binary adjacency matrix of l-order neighbours. If this argument is included (default `NULL`), the parameter `l` is ignored.

`distance` the distance measure to be used (default `"euclidean"`). See the `method` argument of `dist` function for other options.

`verbose` logical value (default `TRUE`); indicates if the function runs in verbose mode.

Details

The DBSC algorithm implemented in this function is a new spatial clustering algorithm based on the density clustering algorithm introduced by Rodriguez and Laio (2014) and the posterior modification presented by Wang and Song (2016). This algorithm is able to obtain a single clustering partition of the data by automatically detecting clustering centers and assigning each area to its nearest cluster centroid. The algorithm has its basis in the assumption that cluster centers are points with high local density and relatively large distance to other points with higher local densities. See Santafé et al. (2021) for more details.

Value

`sf` object with the original data and a grouping variable named 'ID.group'.

References

Rodriguez A, Laio A (2014). "Clustering by fast search and find of density peaks." *Science*, **344**(6191), 1492–1496. doi: 10.1126/science.1242072.

Santafé G, Adin A, Lee D, Ugarte MD (2021). "Dealing with risk discontinuities to estimate cancer mortality risks when the number of small areas is large." *Statistical Methods in Medical Research*, **30**(1), 6–21. doi: 10.1177/0962280220946502.

Wang G, Song Q (2016). "Automatic clustering via outward statistical testing on density metrics." *IEEE Transactions on Knowledge and Data Engineering*, **28**(8), 1971–1985. doi: 10.1109/TKDE.2016.2535209.

Examples

```
## Not run:
library(foreign)
library(maptools)
library(rgdal)
library(tmap)

## Load the Spain colorectal cancer mortality data ##
data(Carto_SpainMUN)

## Define a spatial partition using the DBSC algorithm ##
Carto_SpainMUN$logSMR <- log(Carto_SpainMUN$obs/Carto_SpainMUN$exp+0.0001)

carto.new <- clustering_partition(carto=Carto_SpainMUN, ID.area="ID", var="logSMR",
                                  n.cluster=20, l=2, min.size=100, verbose=TRUE)
table(carto.new$ID.group)

## Plot of the grouping variable 'ID.group' ##
carto.partition <- unionSpatialPolygons(as(carto.new,"Spatial"),carto.new$ID.group)

tm_shape(carto.new) +
        tm_polygons(col="ID.group") +
        tm_shape(carto.partition) +
        tm_borders(col="black", lwd=2) +
        tm_layout(legend.outside=TRUE)

## End(Not run)
```

connect_subgraphs *Merge disjoint connected subgraphs*

Description

The function returns a neighbour list of class nb and its associated spatial adjacency matrix computed by merging disjoint connected subgraphs through its nearest polygon centroids.

Usage

```
connect_subgraphs(carto, ID.area = NULL, nb = NULL, plot = FALSE)
```

Arguments

`carto` object of class `SpatialPolygonsDataFrame` or `sf`.

`ID.area` character vector of geographic identifiers.

`nb` optional argument with the neighbours list of class `nb`. If `NULL` (default), this object is computed from the `carto` argument.

`plot` logical value (default `FALSE`), if `TRUE` then the computed neighbourhood graph is ploted.

Details

This function first calls the `add_neighbour` function to search for isolated areas.

Value

This function returns a list with the following two elements:

- `nb`: the modified neighbours list
- `W`: associated spatial adjacency matrix of class `CsparseMatrix`

Examples

```
library(spdep)

## Load the Spain colorectal cancer mortality data ##
data(Carto_SpainMUN)

## Select the polygons (municipalities) of the 'Comunidad Valenciana' region ##
carto <- Carto_SpainMUN[Carto_SpainMUN$region=="Comunidad Valenciana",]

carto.nb <- poly2nb(carto)
n.comp.nb(carto.nb)$nc # 2 disjoint connected subgraphs

## Plot the spatial polygons and its neighbourhood graph
op <- par(mfrow=c(1,2), pty="s")

plot(carto$geometry, main="Original neighbourhood graph")
plot(carto.nb, st_centroid(st_geometry(carto), of_largest_polygon=TRUE),
     pch=19, cex=0.5, col="red", add=TRUE)

## Use the 'connect_subgraphs' function ##
carto.mod <- connect_subgraphs(carto=carto, ID.area="ID", nb=carto.nb, plot=TRUE)
title(main="Modified neighbourhood graph")

n.comp.nb(carto.mod$nb)$nc==1

par(op)
```

`Data_LungCancer` *Spanish lung cancer mortality data*

Description

`data.frame` object containing simulated lung cancer mortality data in the 7907 municipalities of continental Spain during the period 1991-2015.

Usage

```
Data_LungCancer
```

Format

Formal class `data.frame` with 197.675 rows and 5 colunmns.

- ID: character vector of geographic identifiers
- year: numeric vector of year's identifiers
- obs: observed number of cases
- exp: expected number of cases
- SMR: standardized mortality ratios

`divide_carto` *Divide the spatial domain into subregions*

Description

The function takes an object of class `SpatialPolygonsDataFrame` or `sf` and divides it into subregions according to some grouping variable.

Usage

```
divide_carto(carto, ID.group = NULL, k = 0, plot = FALSE)
```

Arguments

`carto`	object of class `SpatialPolygonsDataFrame` or `sf`.
`ID.group`	character vector of grouping identifiers.
`k`	numeric value with the neighbourhood order to add polygons at the border of the spatial subdomains. If k=0 (default) a disjoint partition is defined.
`plot`	logical value (default `FALSE`), if `TRUE` then the spatial polygons within each subdomain are ploted.

Value

List of `sf` objects with the spatial polygons of each subdomain.

Examples

```
## Not run:
library(tmap)

## Load the Spain colorectal cancer mortality data ##
data(Carto_SpainMUN)

## Plot of the grouping variable 'region' ##
tm_shape(Carto_SpainMUN) +
  tm_polygons(col="region") +
  tm_layout(legend.outside=TRUE)

## Disjoint partition ##
carto.k0 <- divide_carto(carto=Carto_SpainMUN, ID.group="region", k=0)

## Partition + 1st order neighbours ##
carto.k1 <- divide_carto(carto=Carto_SpainMUN, ID.group="region", k=1)

## Partition + 2nd order neighbours ##
carto.k2 <- divide_carto(carto=Carto_SpainMUN, ID.group="region", k=2)

## Plot the spatial polygons for the autonomous region of Castilla y Leon ##
plot(carto.k2$`Castilla y Leon`$geometry, col="dodgerblue4", main="Castilla y Leon")
plot(carto.k1$`Castilla y Leon`$geometry, col="dodgerblue", add=TRUE)
plot(carto.k0$`Castilla y Leon`$geometry, col="lightgrey", add=TRUE)

## End(Not run)
```

mergeINLA *Merge* `inla` *objects for partition models*

Description

The function takes local models fitted for each subregion of the whole spatial domain and unifies them into a single `inla` object. This function is valid for both Disjoint and *k*-order neighbourhood models.

Usage

```
mergeINLA(
  inla.models = list(),
  k = NULL,
  ID.area = "Area",
  ID.year = NULL,
  O = "O",
  E = "E",
  seed = NULL,
  n.sample = 1000,
  compute.fixed = FALSE,
  compute.DIC = TRUE,
  merge.strategy = "original"
)
```

Arguments

`inla.models` list of multiple objects of class `inla`.

`k` numeric value with the neighbourhood order used for the partition model. If k=0 the *Disjoint model* is considered.

`ID.area` character; name of the variable which contains the IDs of spatial areal units. Default to `"Area"`.

`ID.year` character; name of the variable which contains the IDs of time points. Default to `"NULL"` (for spatial models).

`O` character; name of the variable which contains the observed number of disease cases for each areal units. Default to `"O"`.

`E` character; name of the variable which contains either the expected number of disease cases or the population at risk for each areal unit. Default to `"E"`.

`seed` numeric; control the RNG of `inla.qsample` (see `help(inla.qsample)` for further information). Defaults to `NULL`.

`n.sample` numeric; number of samples to generate from the posterior marginal distribution of the risks. Default to 1000.

`compute.fixed` logical value (default `FALSE`); if `TRUE` then the overall log-risk α is computed. Only works if `k=0` argument (*Disjoint model*) is specified. CAUTION: This method might be very time consuming.

`compute.DIC` logical value; if `TRUE` (default) then approximate values of the Deviance Information Criterion (DIC) and Watanabe-Akaike Information Criterion (WAIC) are computed.

`merge.strategy` one of either `"mixture"` or `"original"` (default), which specifies the merging strategy to compute posterior marginal estimates of relative risks.

Details

If the Disjoint model is fitted (`k=0` argument), the log-risk surface is just the union of the posterior estimates of each submodel. However, a single estimate of the overall log-risk α can be computed by extracting samples from the joint posterior distribution of the linear predictors using the `inla.posterior.sample` function of R-INLA. After joining the S samples from each submodel, we define

$$\alpha^s = \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=1}^{T}\log r_{it}, \quad \text{for} \quad s=1,\ldots,S$$

and then compute the kernel density estimate of α.

If the *k*-order neighbourhood model is fitted (`k>0` argument), note that the final risk surface $\mathbf{r} = (r_1, \ldots, r_{nT})'$ is no longer the union of the posterior estimates obtained from each submodel. Since multiple log-risk estimates can be obtained for some areal-time units from the different local submodel, their posterior estimates must be properly combined to obtain a single posterior distribution for each r_{it}. Two different merging strategies could be considered. If the `merge.strategy="mixture"` argument is specified, mixture distributions of the estimated posterior probability density functions with weights proportional to the conditional predictive ordinates (CPO) are computed. If the `merge.strategy="original"` argument is specified (default option), the posterior marginal estimate ot the areal-unit corresponding to the original submodel is selected.

See Orozco-Acosta et al. (2021) and Orozco-Acosta et al. (2022) for more details.

Value

This function returns an object of class `inla` containing the following elements:

`summary.fixed` If `compute.fixed=TRUE` a data.frame containing the mean, standard deviation, quantiles and mode of the model's intercept.

`marginals.fixed`
If `compute.fixed=TRUE` a list containing the posterior marginal density of the model's intercept.

`summary.fixed.partition`
A data.frame containing the mean, standard deviation, quantiles and mode of the model's intercept for each partition.

`marginals.fixed.partition`
A list containing the posterior marginal density of the model's intercept for each partition.

`summary.random` If `k=0` a list with a data.frame containing the mean, standard deviation, quantiles and mode of the model's random effects.

`marginals.random`
If `k=0` a list containing the posterior marginal densities of the model's random effects.

`summary.linear.predictor`
If `k=0` a data.frame containing the mean, standard deviation, quantiles and mode of the log-risks (or log-rates) in the model.

`marginals.linear.predictor`
If `k=0` a list containing the posterior marginal densities of the log-risks (or log-rates) in the model.

`summary.fitted.values`
A data.frame containing the mean, standard deviation, quantiles, mode and cdf of the risks (or rates) in the model.

`marginals.fitted.values`
A list containing the posterior marginal densities of the risks (or rates) in the model.

`logfile` A list of the log files of each submodel.

`version` A list containing information about the R-INLA version.

`cpu.used` The sum of cpu times used by the `inla` function for each submodel (`Pre`, `Running` and `Post`), and the cpu time of the merging process `Merging`.

Examples

```
## See the vignettes accompanying this package for an example of its use.
```

`random_partition` *Define a random partition of the spatial domain based on a regular grid*

Description

The function takes an object of class `SpatialPolygonsDataFrame` or `sf` and defines a random partition of the spatial polygons based on a regular grid over the whole domain using the `st_make_grid` function of the `sf` package.

Usage

```
random_partition(
  carto,
  rows = 3,
  columns = 3,
  min.size = 50,
  max.size = 1000,
  prop.zero = NULL,
  O = NULL
)
```

Arguments

`carto` object of class `SpatialPolygonsDataFrame` or `sf`.

`rows` integer; number of rows to define the regular grid. Default to 3.

`columns` integer; number of columns to define the regular grid. Default to 3.

`min.size` numeric; value to fix the minimum number of areas in each spatial partition (if `NULL`, this step is skipped). Default to 50.

`max.size` numeric; value to fix the maximum number of areas in each spatial partition (if `NULL`, this step is skipped). Default to 600.

`prop.zero` numeric; value between 0 and 1 that indicates the maximum proportion of areas with no cases for each spatial partition.

`O` character; name of the variable which contains the observed number of disease cases for each areal units. Only required if `prop.zero` argument is set.

Details

After defining a random partition of the spatial polygons based on a regular grid, the subregions with number of areas smaller than the value given by the `min.size` are merged to its nearest neighbour. Then, the subregions with number of areas greater than the value given by the `max.size` argument are divided. Finally, if `prop.zero` argument is set, the subregions with proportion of areas with zero cases below that threshold are merged to its smallest neighbour.

Value

sf object with the original data and a grouping variable named 'ID.group'

Examples

```
## Not run:
library(tmap)

## Load the Spain colorectal cancer mortality data ##
data(Carto_SpainMUN)

## Random partition based on a 3x3 regular grid (with no size restrictions) ##
carto.r1 <- random_partition(carto=Carto_SpainMUN, rows=3, columns=3,
                             min.size=NULL, max.size=NULL)
table(carto.r1$ID.group)

part1 <- aggregate(carto.r1[,"geometry"], by=list(ID.group=carto.r1$ID.group), head)

tm_shape(carto.r1) +
  tm_polygons(col="ID.group") +
  tm_shape(part1) + tm_borders(col="black", lwd=2) +
  tm_layout(main.title="3x3 regular grid (with no size restrictions)",
            main.title.position="center", main.title.size=1,
            legend.outside=TRUE)


## Random partition based on a 6x4 regular grid (with size restrictions) ##
carto.r2 <- random_partition(carto=Carto_SpainMUN, rows=6, columns=4,
                             min.size=50, max.size=600)
table(carto.r2$ID.group)

part2 <- aggregate(carto.r2[,"geometry"], by=list(ID.group=carto.r2$ID.group), head)

tm_shape(carto.r2) +
  tm_polygons(col="ID.group") +
  tm_shape(part2) + tm_borders(col="black", lwd=2) +
  tm_layout(main.title="6x4 regular grid (min.size=50, max.size=600)",
            main.title.position="center", main.title.size=1,
            legend.outside=TRUE)


## Random partition based on a 6x4 regular grid (with size and proportion of zero restrictions) ##
carto.r3 <- random_partition(carto=Carto_SpainMUN, rows=6, columns=4,
                             min.size=50, max.size=600, prop.zero=0.5, O="obs")
table(carto.r3$ID.group)

part3 <- aggregate(carto.r3[,"geometry"], by=list(ID.group=carto.r3$ID.group), head)

tm_shape(carto.r3) +
  tm_polygons(col="ID.group") +
  tm_shape(part3) + tm_borders(col="black", lwd=2) +
  tm_layout(main.title="6x4 regular grid (min.size=50, max.size=600, prop.zero=0.5)",
            main.title.position="center", main.title.size=1,
            legend.outside=TRUE)

## End(Not run)
```

STCAR_INLA *Fit a (scalable) spatio-temporal Poisson mixed model to areal count data.*

Description

Fit a spatio-temporal Poisson mixed model to areal count data, where several CAR prior distributions for the spatial random effects, first and second order random walk priors for the temporal random effects, and different types of spatio-temporal interactions described in Knorr-Held (2000) can be specified. The linear predictor is modelled as

$$\log r_{it} = \alpha + \xi_i + \gamma_t + \delta_{it}, \quad \text{for} \quad i = 1, \ldots, n; \quad t = 1, \ldots, T$$

where α is a global intercept, ξ_i is a spatially structured random effect, γ_t is a temporally structured random effect, and δ_{it} is the space-time interaction effect. If the interaction term is dropped, an additive model is obtained. To ensure model identifiability, sum-to-zero constraints are imposed over the random effects of the model. Details on the derivation of these constraints can be found in Goicoa et al. (2018).

As in the CAR_INLA function, three main modelling approaches can be considered:

- the usual model with a global spatial random effect whose dependence structure is based on the whole neighbourhood graph of the areal units (model="global" argument)
- a Disjoint model based on a partition of the whole spatial domain where independent spatial CAR models are simultaneously fitted in each partition (model="partition" and k=0 arguments)
- a modelling approach where *k*-order neighbours are added to each partition to avoid border effects in the Disjoint model (model="partition" and k>0 arguments).

For both the Disjoint and k-order neighbour models, parallel or distributed computation strategies can be performed to speed up computations by using the 'future' package (Bengtsson 2021).

Inference is conducted in a fully Bayesian setting using the integrated nested Laplace approximation (INLA; Rue et al. (2009)) technique through the R-INLA package (https://www.r-inla.org/). For the scalable model proposals (Orozco-Acosta et al. 2022), approximate values of the Deviance Information Criterion (DIC) and Watanabe-Akaike Information Criterion (WAIC) can also be computed.

Usage

```
STCAR_INLA(
  carto = NULL,
  data = NULL,
  ID.area = NULL,
  ID.year = NULL,
  ID.group = NULL,
  O = NULL,
  E = NULL,
  W = NULL,
  spatial = "Leroux",
  temporal = "rw1",
  interaction = "TypeIV",
  model = "partition",
  k = 0,
  strategy = "simplified.laplace",
  PCpriors = FALSE,
  seed = NULL,
  n.sample = 1000,
  compute.fixed = FALSE,
  compute.DIC = TRUE,
  save.models = FALSE,
  plan = "sequential",
  workers = NULL,
  merge.strategy = "original"
)
```

Arguments

`carto` object of class `SpatialPolygonsDataFrame` or `sf`. This object must contain at least the variable with the identifiers of the spatial areal units specified in the argument `ID.area`.

`data` object of class `data.frame` which must contain the target variables of interest specified in the arguments `ID.area`, `ID.year`, `O` and `E`.

`ID.area` character; name of the variable which contains the IDs of spatial areal units. The values of this variable must match those given in the `carto` and `data` variable.

`ID.year` character; name of the variable which contains the IDs of time points.

`ID.group` character; name of the variable which contains the IDs of the spatial partition (grouping variable). Only required if `model="partition"`.

`O` character; name of the variable which contains the observed number of disease cases for each areal and time point.

`E` character; name of the variable which contains either the expected number of disease cases or the population at risk for each areal unit and time point.

`W` optional argument with the binary adjacency matrix of the spatial areal units. If `NULL` (default), this object is computed from the `carto` argument (two areas are considered as neighbours if they share a common border).

`spatial` one of either `"Leroux"` (default), `"intrinsic"`, `"BYM"` or `"BYM2"`, which specifies the prior distribution considered for the spatial random effect.

`temporal` one of either `"rw1"` (default) or `"rw2"`, which specifies the prior distribution considered for the temporal random effect.

`interaction` one of either `"none"`, `"TypeI"`, `"TypeII"`, `"TypeIII"` or `"TypeIV"` (default), which specifies the prior distribution for the space-time interaction random effect.

`model` one of either `"global"` or `"partition"` (default), which specifies the *Global model* or one of the scalable model proposal's (*Disjoint model* and *k-order neighbourhood model*, respectively).

`k` numeric value with the neighbourhood order used for the partition model. Usually k=2 or 3 is enough to get good results. If k=0 (default) the *Disjoint model* is considered. Only required if `model="partition"`.

`strategy` one of either `"gaussian"`, `"simplified.laplace"` (default), `"laplace"` or `"adaptive"`, which specifies the approximation strategy considered in the `inla` function.

`PCpriors` logical value (default `FALSE`); if `TRUE` then penalised complexity (PC) priors are used for the precision parameter of the spatial random effect. Only works if arguments `spatial="intrinsic"` or `spatial="BYM2"` are specified.

`seed` numeric (default `NULL`); control the RNG of the `inla.qsample` function. See `help(inla.qsample)` for further information.

`n.sample` numeric; number of samples to generate from the posterior marginal distribution of the risks. Default to 1000.

`compute.fixed` logical value (default `FALSE`); if `TRUE` then the overall log-risk α is computed. Only works if `k=0` argument (*Disjoint model*) is specified.

`compute.DIC` logical value; if `TRUE` (default) then approximate values of the Deviance Information Criterion (DIC) and Watanabe-Akaike Information Criterion (WAIC) are computed.

`save.models` logical value (default `FALSE`); if `TRUE` then a list with all the `inla` submodels is saved in '/temp/' folder, which can be used as input argument for the `mergeINLA` function.

`plan` one of either `"sequential"` or `"cluster"`, which specifies the computation strategy used for model fitting using the 'future' package. If `plan="sequential"` (default) the models are fitted sequentially and in the current R session (local machine). If `plan="cluster"` the models are fitted in parallel on external R sessions (local machine) or distributed in remote compute nodes.

`workers` character or vector (default `NULL`) containing the identifications of the local or remote workers where the models are going to be processed. Only required if `plan="cluster"`.

`merge.strategy` one of either `"mixture"` or `"original"` (default), which specifies the merging strategy to compute posterior marginal estimates of relative risks. See `mergeINLA` for further details.

Details

For a full model specification and further details see the vignettes accompanying this package.

Value

This function returns an object of class inla. See the mergeINLA function for details.

References

Goicoa T, Adin A, Ugarte MD, Hodges JS (2018). “In spatio-temporal disease mapping models, identifiability constraints affect PQL and INLA results.” *Stochastic Environmental Research and Risk Assessment*, **32**(3), 749–770. doi: 10.1007/s0047701714050.

Knorr-Held L (2000). “Bayesian modelling of inseparable space-time variation in disease risk.” *Statistics in Medicine*, **19**(17-18), 2555–2567.

Orozco-Acosta E, Adin A, Ugarte MD (2021). “Scalable Bayesian modeling for smoothing disease mapping risks in large spatial data sets using INLA.” *Spatial Statistics*, **41**, 100496. doi: 10.1016/j.spasta.2021.100496.

Orozco-Acosta E, Adin A, Ugarte MD (2022). “Parallel and distributed Bayesian modelling for analysing high-dimensional spatio-temporal count data.” https://arxiv.org/abs/2201.08323.

Examples

```
## Not run:
if(require("INLA", quietly=TRUE)){

  ## Load the sf object that contains the spatial polygons of the municipalities of Spain ##
  data(Carto_SpainMUN)
  str(Carto_SpainMUN)

  ## Create province IDs ##
  Carto_SpainMUN$ID.prov <- substr(Carto_SpainMUN$ID,1,2)

  ## Load simulated data of lung cancer mortality data during the period 1991-2015 ##
  data("Data_LungCancer")
  str(Data_LungCancer)

  ## Fit the Disjoint model with a BYM2 spatial random effect,
  ## RW1 temporal random effect and Type I interaction random effect ##
  Disjoint <- STCAR_INLA(carto=Carto_SpainMUN, data=Data_LungCancer,
                      ID.area="ID", ID.year="year", O="obs", E="exp", ID.group="ID.prov",
                         spatial="BYM2", temporal="rw1", interaction="TypeI",
                         model="partition", k=0, strategy="gaussian",
                         plan="cluster", workers=rep("localhost",2))
  summary(Disjoint)

  ## Fit the 1st order neighbourhood model with a BYM2 spatial random effect,
  ## RW1 temporal random effect and Type I interaction random effect ##
  order1 <- STCAR_INLA(carto=Carto_SpainMUN, data=Data_LungCancer,
                     ID.area="ID", ID.year="year", O="obs", E="exp", ID.group="ID.prov",
                       spatial="BYM2", temporal="rw1", interaction="TypeI",
                       model="partition", k=1, strategy="gaussian",
                       plan="cluster", workers=rep("localhost",2))
  summary(order1)
}
```

```
## End(Not run)
```

Index