Package: SSNbayes (via r-universe)

October 28, 2024

Type Package

Title Bayesian Spatio-Temporal Analysis in Stream Networks

Version 0.1.0

Depends R (>= 4.0.0)

Imports plyr, dplyr, rstan, mtsdi, sf, methods, SSN2, sfnetworks, igraph, geosphere, osmdata, purrr, leaflet

Description Fits Bayesian spatio-temporal models and makes predictions on stream networks using the approach by Santos-Fernandez, Edgar, et al. (2022).``Bayesian spatio-temporal models for stream networks'' and Santos-Fernandez, Edgar, et al. (2023). ``SSNbayes: An R Package for Bayesian Spatio-Temporal Modelling on Stream Networks''. In these models, spatial dependence is captured using stream distance and flow connectivity, while temporal autocorrelation is modelled using vector autoregression methods.

License GPL-2

Encoding UTF-8

RoxygenNote 7.3.2

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

VignetteBuilder knitr

URL https://github.com/EdgarSantos-Fernandez/SSNbayes

BugReports https://github.com/EdgarSantos-Fernandez/SSNbayes/issues

Config/testthat/edition 3

Repository https://edgarsantos-fernandez.r-universe.dev

RemoteUrl https://github.com/edgarsantos-fernandez/ssnbayes

RemoteRef HEAD

RemoteSha 44f320669fcecfa924cb3c6209e0258005f87c4d

Contents

collapse *Collapses a SpatialStreamNetwork object into a data frame*

Description

Collapses a SpatialStreamNetwork object into a data frame

Usage

```
collapse(ssn, par = "afvArea")
```

Arguments

`ssn`	An S4 SpatialStreamNetwork object created with SSN2 package.
`par`	A spatial parameter such as the computed_afv (additive function value).

Details

The parameters (par) has to be present in the observed data frame via ssn_get_data(n, name = "obs"). More details of the argument par can be found in the additive.function() from SSN .

Value

A data frame with the lat and long of the line segments in the network. The column line_id refers to the ID of the line.

Examples

```
#require("SSN2")
#path <- system.file("extdata/clearwater.ssn", package = "SSNbayes")
#ssn <- SSN2::ssn_import(path, predpts = "preds", overwrite  = TRUE)
#t.df <- collapse(ssn, par = 'afvArea')
```

convert_network_to_graph

Combines two ‘sf‘ objects, the sensor locations and river network, in to an ‘sfnetwork‘.

Description

Combines two ‘sf‘ objects, the sensor locations and river network, in to an ‘sfnetwork‘.

Usage

```
convert_network_to_graph(river_network, sensor_locations)
```

Arguments

`river_network` An ‘sf‘ object of ‘MULTILINESTRING‘(s) containing the streams and rivers, cropped to the sensor locations.

`sensor_locations` An ‘sf‘ dataframe containing sensor locations and ‘locID‘.

Details

This function "snaps" sensor locations to the nearest point on the network. Additionally, the sensor locations are stored in the graph, ‘g‘, and can be accessed via ‘V(g)$type‘, as "sensor".

Value

An ‘sfnetwork‘ graph containing the river network and sensors.

Author(s)

Sean Francis

dist_weight_mat *Creates a list containing the stream distances and weights*

Description

Creates a list containing the stream distances and weights

Usage

```
dist_weight_mat(path = path, net = 1, addfunccol = "addfunccol")
```

Arguments

path	Path to the files
net	(optional) A network from the SSN2 object
addfunccol	(optional) A parameter to compute the spatial weights

Value

A list of matrices

Examples

```
path <- system.file("extdata/clearwater.ssn", package = "SSNbayes")
mat_all <- dist_weight_mat(path, net = 2, addfunccol='afvArea')
```

dist_weight_mat_preds *Creates a list of distances and weights between observed and prediction sites*

Description

The output matrices are symmetric except the hydrologic distance matrix D.

Usage

```
dist_weight_mat_preds(path = path, net = 1, addfunccol = "addfunccol")
```

Arguments

path	Path with the name of the SpatialStreamNetwork object
net	(optional) A network from the SpatialStreamNetwork object
addfunccol	(optional) A parameter to compute the spatial weights

Value

A list of matrices

Examples

```
## Not run:
path <- system.file("extdata/clearwater.ssn", package = "SSNbayes")
mat_all_pred <- dist_weight_mat_preds(path, net = 2, addfunccol='afvArea')
## End(Not run)
```

format_sensor_data *Converts a dataframe containing sensor data in to an ‘sf‘ object*

Description

Converts a dataframe containing sensor data in to an ‘sf‘ object

Usage

```
format_sensor_data(sensor_data, lon_name, lat_name)
```

Arguments

sensor_data	A dataframe containing sensor locations and a column called ‘locID‘ denoting the location IDs.
lon_name	The column name containing the longitude of the sensor.
lat_name	The column name containing the latitude of the sensor.

Value

An ‘sf‘ dataframe containing sensor locations and ‘locID‘.

Author(s)

Sean Francis

generate_osm_ssn *Generates an Open Street Maps Spatial Stream Network.*

Description

It will take a dataframe containing sensor locations, with a coulmn ‘locID‘ denoting location IDs (can be string or numeric). Requires a root sensor location, in which all sensor locations are connected to, and flow towards.

Usage

```
generate_osm_ssn(
  sensor_data,
  lon_name,
  lat_name,
  root_loc,
  plot_network = TRUE,
  gen_pred_sites = FALSE,
  num_pred_sites = NA
)
```

Arguments

`sensor_data`	A dataframe containing sensor locations and a column called ‘locID‘ denoting the location IDs.
`lon_name`	The column name containing the longitude of the sensor.
`lat_name`	The column name containing the latitude of the sensor.
`root_loc`	A sensor location which all water flows towards.
`plot_network`	A bool indicating whether to plot the network. Useful for determining potential errors. Will be saved in object.
`gen_pred_sites`	A bool indicating whether to generate new prediction sites.
`num_pred_sites`	Required if ‘gen_pred_sites=TRUE‘. An integer indicating the number of prediction sites to generate.

Details

Assumes longitude and latitude data are in the 4326 coordinate reference system. If they are not, data will need to be converted beforehand using ‘st_transform()‘. Predictions on sites using gen_pred_sites is under development specially for tail-down models.

Value

An ‘osm_ssn‘ object, containing: the graph network as an ‘sfnetwork‘; the distance adjacency matrices; if ‘plot_network=TRUE‘, an interactive leaflet plot, if ‘gen_pred_sites=TRUE‘, an ‘sf‘ object containing the locations of the predictions sites.

Author(s)

Sean Francis

Examples

```
# require('SSNdata')
# clear <- readRDS(system.file("extdata/clear_obs.RDS", package = "SSNdata"))
#
# # Generate osm ssn
# clear_osm_ssn <- generate_osm_ssn(clear,
#                                   "long", "lat",
#                                   root_loc = 12,
#                                   plot_network = TRUE,
#                                   gen_pred_sites = TRUE,
#                                   num_pred_sites = 20)
# # Show plot
# clear_osm_ssn$plot
```

generate_prediction_locations

Creates an ‘sf‘ dataframe containing equally spaced points on a river network

Description

Creates an ‘sf‘ dataframe containing equally spaced points on a river network

Usage

```
generate_prediction_locations(river_network, num_pred_sites)
```

Arguments

river_network	An ‘sf‘ object of ‘MULTILINESTRING‘(s) containing the streams and rivers, cropped to the sensor locations.
num_pred_sites	An integer indicating the number of prediction sites to generate.

Value

An ‘sf‘ dataframe of ‘POINT‘(s) containing prediction sites.

Author(s)

Sean Francis

generate_river_network *Takes an ‘sf‘ object as input, and queries Open Street Maps for rivers and streams within the bounds of the sensor locations.*

Description

Takes an ‘sf‘ object as input, and queries Open Street Maps for rivers and streams within the bounds of the sensor locations.

Usage

```
generate_river_network(sensor_locations, root_loc)
```

Arguments

sensor_locations	An ‘sf‘ dataframe containing sensor locations and ‘locID‘.
root_loc	A sensor location which all water flows towards.

Details

This function queries Open Street Maps using ‘osmdata‘ package in R. Crops the data to the bounding box of the sensor locations. Removes all lines not connected to the outlet/root_loc.

Value

An ‘sf‘ object of ‘MULTILINESTRING‘(s) containing the streams and rivers, cropped to the sensor location, with all lines not containing the outlet/root_loc removed.

Author(s)

Sean Francis

gen_distance_matrices *Creates a list containing the stream distances and weights*

Description

Creates a list containing the stream distances and weights

Usage

```
gen_distance_matrices(network, sensor_locations, root_loc)
```

Arguments

`network` An ‘sfnetwork‘ containing the river network and sensor locations.

`sensor_locations` An ‘sf‘ dataframe containing sensor locations and ‘locID‘.

`root_loc` A sensor location which all water flows towards.

Value

A list of matrices.

Author(s)

Sean Francis

`krig` *Internal function used to perform spatio-temporal prediction in R using a stanfit object from ssnbayes()*

Description

Use predict.ssnbayes() instead. It will take an observed and a prediction data frame. It requires the same number of observation/locations per day. It requires location id (locID) and points id (pid). The locID are unique for each site. The pid is unique for each observation. Missing values are allowed in the response but not in the covariates.

Usage

```
krig(
  object = object,
  mat_all_preds = mat_all_preds,
  nsamples = 10,
  start = 1,
  chunk_size = 50,
  obs_data = obs_data,
  pred_data = pred_data,
  net = net,
  seed = seed
)
```

Arguments

`object` A stanfit object returned from ssnbayes

`mat_all_preds` A list with the distance/weights matrices

`nsamples` The number of samples to draw from the posterior distributions. (nsamples <= iter)

`start` (optional) The starting location id `chunk_size` (optional) the number of locID to make prediction from

`obs_data` The observed data frame

`pred_data` The predicted data frame

`net` (optional) Network from the SSN object

`seed` (optional) A seed for reproducibility

Value

A data frame

Author(s)

Edgar Santos-Fernandez

krig2 *Internal function used to perform spatio-temporal prediction in R using a stanfit object from ssnbayes()*

Description

Use predict.ssnbayes() instead. It will take an observed and a prediction data frame. It requires the same number of observation/locations per day. It requires location id (locID) and points id (pid). The locID are unique for each site. The pid is unique for each observation. Missing values are allowed in the response but not in the covariates.

Usage

```
krig2(
  object = object,
  mat_all_preds = mat_all_preds,
  nsamples = 10,
  start = 1,
  chunk_size = 50,
  obs_data = obs_data,
  pred_data = pred_data,
  net = net,
  seed = seed
)
```

Arguments

`object` A stanfit object returned from ssnbayes

`mat_all_preds` A list with the distance/weights matrices

`nsamples` The number of samples to draw from the posterior distributions. (nsamples <= iter)

`start`	(optional) The starting location id
`chunk_size`	(optional) the number of locID to make prediction from
`obs_data`	The observed data frame
`pred_data`	The predicted data frame
`net`	(optional) Network from the SSN object
`seed`	(optional) A seed for reproducibility

Value

A data frame

Author(s)

Edgar Santos-Fernandez

`mylm` *A simple modeling function using a formula and data*

Description

A simple modeling function using a formula and data

Usage

```
mylm(formula, data)
```

Arguments

`formula`	A formula as in lm()
`data`	A data.frame containing the elements specified in the formula

Value

A list of matrices

Author(s)

Jay ver Hoef

Examples

```
options(na.action='na.pass')
data("iris")
out_list = mylm(formula = Petal.Length ~ Sepal.Length + Sepal.Width, data = iris)
```

predict.ssnbayes *Performs spatio-temporal prediction in R using an ssnbayes object from a fitted model.*

Description

It will take an observed and a prediction data frame. It requires the same number of observation/locations per day. It requires location id (locID) and points id (pid). The locID are unique for each site. The pid is unique for each observation. Missing values are allowed in the response but not in the covariates.

Usage

```
## S3 method for class 'ssnbayes'
predict(
  object = object,
  ...,
  path = path,
  obs_data = obs_data,
  pred_data = pred_data,
  net = net,
  nsamples = nsamples,
  addfunccol = addfunccol,
  locID_pred = locID_pred,
  chunk_size = chunk_size,
  seed = seed
)
```

Arguments

`object`	A stanfit object returned from ssnbayes
`...`	Other parameters
`path`	Path with the name of the SpatialStreamNetwork object
`obs_data`	The observed data frame
`pred_data`	The predicted data frame
`net`	(optional) Network from the SSN object
`nsamples`	The number of samples to draw from the posterior distributions. (nsamples <= iter)
`addfunccol`	The variable used for spatial weights
`locID_pred`	(optional) the location id for the predictions. Used when the number of pred locations is large.
`chunk_size`	(optional) the number of locID to make prediction from
`seed`	(optional) A seed for reproducibility

Details

The returned data frame is melted to produce a long dataset. See examples. Currently, the predict() function produces predictions for normal random variables. However, this can be easily transformed in to counts (Poisson distributed) and presence/absence (binomial distributed).

Value

A data frame with the location (locID), time point (date), plus the MCMC draws from the posterior from 1 to the number of iterations. The locID0 column is an internal consecutive location ID (locID) produced in the predictions, starting at max(locID(observed data)) + 1. It is used internally in the way predictions are made in chunks.

Author(s)

Edgar Santos-Fernandez

Examples

```
#require('SSNdata')
#clear_preds <- readRDS(system.file("extdata/clear_preds.RDS", package = "SSNdata"))
#clear_preds$y <- NA
#pred <- predict(object = fit_ar,
#                path = path,
#                obs_data = clear,
#                pred_data = clear_preds,
#                net = 2,
#                nsamples = 100, # numb of samples from the posterior
#                addfunccol = 'afvArea', # var for spatial weights
#                locID_pred = locID_pred,
#                chunk_size = 60)
```

predict.ssnbayes2 *Performs spatio-temporal prediction in R using an ssnbayes object from a fitted model.*

Description

It will take an observed and a prediction data frame. It requires the same number of observation/locations per day. It requires location id (locID) and points id (pid). The locID are unique for each site. The pid is unique for each observation. Missing values are allowed in the response but not in the covariates.

Usage

```
## S3 method for class 'ssnbayes2'
predict(
  object = object,
  ...,
  use_osm_ssn = TRUE,
  osm_ssn = osm_ssn,
  path = path,
  obs_data = obs_data,
  pred_data = pred_data,
  net = net,
  nsamples = nsamples,
  addfunccol = addfunccol,
  locID_pred = locID_pred,
  chunk_size = chunk_size,
  seed = seed
)
```

Arguments

object	A stanfit object returned from ssnbayes
...	Other parameters
use_osm_ssn	Use a supplied osm_ssn instead a SpatialStreamNetwork object.
osm_ssn	The osm_ssn to be used.
path	If not using an osm_ssn, path with the name of the SpatialStreamNetwork object.
obs_data	The observed data frame
pred_data	The predicted data frame
net	(optional) Network from the SSN object
nsamples	The number of samples to draw from the posterior distributions. (nsamples <= iter)
addfunccol	If not using an osm_ssn, the variable used for spatial weights
locID_pred	(optional) the location id for the predictions. Used when the number of pred locations is large.
chunk_size	(optional) the number of locID to make prediction from
seed	(optional) A seed for reproducibility

Details

The returned data frame is melted to produce a long dataset. See examples.

Value

A data frame with the location (locID), time point (date), plus the MCMC draws from the posterior from 1 to the number of iterations. The locID0 column is an internal consecutive location ID (locID) produced in the predictions, starting at max(locID(observed data)) + 1. It is used internally in the way predictions are made in chunks.

Author(s)

Edgar Santos-Fernandez

Examples

```
# require('SSNdata')
# require('sf')
#
# clear <- readRDS(system.file("extdata/clear_obs.RDS", package = "SSNdata"))
#
# clear_osm_ssn <- generate_osm_ssn(clear, "long", "lat", root_loc = 12, plot_network = TRUE)
#
# formula = y ~ SLOPE + elev + air_temp + sin + cos
#
# family = "gaussian"
#
# # Note - missing data must be imputed before using the predict() function.
# # This can be done using:
# data_impute <- mtsdi::mnimput(
#   formula = temp ~ SLOPE + elev + h2o_area + air_temp + sin + cos,
#   dataset = clear,
#   eps = 1e-3,
#   ts = FALSE,
#   method = "glm"
# )$filled.dataset
#
# clear <- left_join(clear, data_impute, by = c("elev", "air_temp", "sin", "cos", "SLOPE"))
#
# clear$y <- clear$temp.y
#
# fit_ar <- ssnbayes2(formula = formula,
#                     data = clear,
#                     osm_ssn = clear_osm_ssn,
#                     family = family,
#                     time_method = list("ar", "date"),
#                     space_method = list('use_osm_ssn', c("Exponential.taildown")),
#                     iter = 2000,
#                     warmup = 1000,
#                     chains = 3,
#                     cores = 3)
#
# # Get the coordinate reference system of the near_X and near_Y variables
# data_crs <- system.file("extdata/clearwater.ssn", package = "SSNbayes") %>%
#   SSN2::ssn_import(predpts = "preds", overwrite  = TRUE) %>%
#   SSN2::ssn_get_data(name = "preds") %>%
#   st_crs
#
# # Load in the predictions, and convert to 4326 CRS
# clear_preds <- readRDS(system.file("extdata/clear_preds.RDS", package = "SSNbayes")) %>%
#   st_as_sf(coords = c("NEAR_X","NEAR_Y"),
#            crs = data_crs) %>%
#   st_transform(crs = 4326)
```

```
#
# # Extract the long and lat columns
# xy <- st_coordinates(clear_preds)
#
# colnames(xy) <- c("long", "lat")
#
# Merge back in with the prediction dataset
# clear_preds <- cbind(clear_preds, xy) %>%
#   data.frame() %>%
#   select(-geometry)
#
# same_names <- intersect(names(clear), names(clear_preds))
#
# # Combine all observed and prediction sites into 1 dataframe
# all_sites <- rbind(clear %>% select(same_names),
#                    data.frame(clear_preds) %>% select(same_names)
#                    )
#
#
# clear_preds_osm_ssn <- generate_osm_ssn(sensor_data = all_sites,
#                                         lon_name = "long", lat_name = "lat",
#                                         root_loc = 12,
#                                         plot_network = TRUE,
#                                         gen_pred_sites = FALSE)
#
#
#
# locs <- clear_preds_osm_ssn$dist_mat_all$e %>% colnames
#
#
# clear_krig <- clear %>%
#   filter(locID %in% locs)
#
# clear_krig_preds <- clear_preds %>%
#   filter(locID %in% locs)
#
#
# preds <- predict(object = fit_ar,
#                  use_osm_ssn = TRUE,
#                  osm_ssn = clear_preds_osm_ssn,
#                  obs_data = clear_krig,
#                  pred_data = clear_krig_preds,
#                  seed = seed,
#                  nsamples = 25,
#                  chunk_size = length(unique(clear_krig_preds$locID))
#                  )
#
# # Condense data to posterior point estimates
# ys <- reshape2::melt(preds, id.vars = c('locID0', 'locID', 'date'), value.name ='y')
# ys$iter <- gsub("[^0-9.-]", "", ys$variable)
# ys$variable <- NULL
#
# ys <- data.frame(ys) %>% dplyr::group_by(date, locID, locID0) %>%
#   dplyr::summarise("sd" = sd(y, na.rm=T),
#                   "y_pred" = mean(y, na.rm=T))
#
# ys <- dplyr::arrange(ys, locID)
#'
```

pred_ssnbayes *Internal function used to perform spatio-temporal prediction in R using a stanfit object from ssnbayes()*

Description

Use predict.ssnbayes() instead. It will take an observed and a prediction data frame. It requires the same number of observation/locations per day. It requires location id (locID) and points id (pid). The locID are unique for each site. The pid is unique for each observation. Missing values are allowed in the response but not in the covariates.

Usage

```
pred_ssnbayes(
  object = object,
  use_osm_ssn = TRUE,
  osm_ssn = osm_ssn,
  path = path,
  obs_data = obs_data,
  pred_data = pred_data,
  net = 1,
  nsamples = 100,
  addfunccol = "afvArea",
  locID_pred = locID_pred,
  chunk_size = chunk_size,
  seed = seed
)
```

Arguments

`object`	A stanfit object returned from ssnbayes
`use_osm_ssn`	Use a supplied osm_ssn instead a SpatialStreamNetwork object.
`osm_ssn`	The osm_ssn to be used.
`path`	If not using an osm_ssn, path with the name of the SpatialStreamNetwork object.
`obs_data`	The observed data frame
`pred_data`	The predicted data frame
`net`	(optional) Network from the SSN object
`nsamples`	The number of samples to draw from the posterior distributions. (nsamples <= iter)

`addfunccol` If not using an osm_ssn, the variable used for spatial weights.

`locID_pred` (optional) the location id for the predictions. Used when the number of pred locations is large.

`chunk_size` (optional) the number of locID to make prediction from

`seed` (optional) A seed for reproducibility

Value

A data frame

Author(s)

Edgar Santos-Fernandez

Examples

```
#pred <- pred_ssnbayes(path = path,
#obs_data = clear,
#stanfit = fit_ar,
#pred_data = preds,
#net = 2,
#nsamples = 100, # number of samples to use from the posterior in the stanfit object
#addfunccol = 'afvArea') # variable used for spatial weights
```

`ssnbayes` *Fits a mixed linear regression model using Stan*

Description

It requires the same number of observation/locations per day. It requires location id (locID) and points id (pid). The locID are unique for each site. The pid is unique for each observation. Missing values are allowed in the response but not in the covariates.

Usage

```
ssnbayes(
  formula = formula,
  data = data,
  path = path,
  time_method = time_method,
  space_method = space_method,
  iter = 3000,
  warmup = 1500,
  chains = 3,
  refresh = max(iter/100, 1),
  net = 1,
  addfunccol = addfunccol,
  loglik = FALSE,
  ppd = FALSE,
  seed = seed
)
```

Arguments

`formula` A formula as in lm()

`data` A long data frame containing the locations, dates, covariates and the response variable. It has to have the locID and date. No missing values are allowed in the covariates. The order in this data.fame MUST be: spatial locations (1 to S) at time t=1, then locations (1 to S) at t=2 and so on.

`path` Path with the name of the SpatialStreamNetwork object

`time_method` A list specifying the temporal structure (ar = Autorregressive; var = Vector autorregression) and coumn in the data with the time variable.

`space_method` A list defining if use or not of an SSN object and the spatial correlation structure. The second element is the spatial covariance structure. A 3rd element is a list with the lon and lat for Euclidean distance models.

`iter` Number of iterations

`warmup` Warm up samples

`chains` Number of chains

`refresh` Sampler refreshing rate

`net` The network id (optional). Used when the SSN object contains multiple networks.

`addfunccol` Variable to compute the additive function. Used to compute the spatial weights.

`loglik` Logic parameter denoting if the loglik will be computed by the model.

`ppd` Produce the posterior predictive distribution

`seed` (optional) A seed for reproducibility

Details

Missing values are not allowed in the covariates and they must be imputed before using ssnbayes(). Many options can be found in https://cran.r-project.org/web/views/MissingData.html The pid in the data has to be consecutive from 1 to the number of observations. Users can use the SpatialStreamNetwork created with the SSN package. This will provide the spatial stream information used to compute covariance matrices. If that is the case, the data has to have point ids (pid) matching the ones in SSN distance matrices, so that a mapping can occur.

Value

A list with the model fit

It returns a ssnbayes object (similar to stan returns). It includes the formula used to fit the model. The output can be transformed into the stanfit class using class(fits) <- c("stanfit").

Author(s)

Edgar Santos-Fernandez

Examples

```
## Not run:
#options(mc.cores = parallel::detectCores())
# Import SpatialStreamNetwork object
#path <- system.file("extdata/clearwater.ssn", package = "SSNbayes")
#n <- SSN2::ssn_import(path, predpts = "preds", overwrite  = TRUE)
## Imports a data.frame containing observations and covariates
#clear <- readRDS(system.file("extdata/clear_obs.RDS", package = "SSNbayes"))
#fit_ar <- ssnbayes(formula = y ~ SLOPE + elev + h2o_area + air_temp + sin + cos,
#                   data = clear,
#                   path = path,
#                   time_method = list("ar", "date"),
#                   space_method = list('use_ssn', c("Exponential.taildown")),
#                   iter = 2000,
#                   warmup = 1000,
#                   chains = 3,
#                   net = 2, # second network on the ssn object
#                   addfunccol='afvArea')
#space_method options examples
#use list('no_ssn', 'Exponential.Euclid', c('lon', 'lat')) if no ssn object is available

## End(Not run)
```

ssnbayes2 — *Fits a mixed linear regression model using Stan. This is an updated version of ssnbayes()*

Description

It requires the same number of observation/locations per day. It requires location id (locID) and points id (pid). The locID are unique for each site. The pid is unique for each observation. Missing values are allowed in the response and in the covariates. Missing values are imputed using the ‘mtsdi‘ pacakge, with a ‘glm‘ family.

Usage

```
ssnbayes2(
  formula = formula,
  family = family,
  data = data,
  osm_ssn = osm_ssn,
  path = NA,
  time_method = time_method,
  space_method = space_method,
  iter = 3000,
  warmup = 1500,
  chains = 3,
  cores = 3,
  refresh = max(iter/100, 1),
  net = NA,
  addfunccol = NA,
  loglik = FALSE,
  seed = seed
)
```

Arguments

`formula` A formula as in lm()

`family` A description of the response distribution and link function to be used in the model. Must be one of: ’gaussian’, ’binomial’, ’poisson".

`data` A long data frame containing the locations, dates, covariates and the response variable. It has to have the locID and date. No missing values are allowed in the covariates. The order in this data.fame MUST be: spatial locations (1 to S) at time t=1, then locations (1 to S) at t=2 and so on.

`osm_ssn` An ‘osm_ssn‘ generated using ‘generate_osm_ssn()‘.

`path` File path with the name of the SpatialStreamNetwork object

`time_method` A list specifying the temporal structure (ar = Autorregressive; var = Vector autorregression) and column in the data with the time variable.

`space_method` A list, first element must be one of ’use_ssn’, ’use_osm_ssn’, ’no_ssn. Whether to use SSN or osm ssn, and the spatial correlation structure. The second element is the spatial covariance structure. A 3rd element is a list with the lon and lat for Euclidean distance models.

`iter` Number of iterations

`warmup` Warm up samples

`chains` Number of chains

`cores` Number of cores

`refresh` Sampler refreshing rate

`net` The network id (optional). Used when the SSN object cotains multiple networks.

`addfunccol` Variable to compute the additive function. Used to compute the spatial weights.

`loglik` Logic parameter denoting if the loglik will be computed by the model.

`seed` (optional) A seed for reproducibility

Details

Missing values on the covariates and response can be imputed by default using mtsdi::mnimput(). We strongly recommend the documentation for this function be read before use. The pid in the data has to be consecutive from 1 to the number of observations. Users can use the SpatialStreamNetwork created with the SSN package. This will provide the spatial stream information used to compute covariance matrices. If that is the case, the data has to have point ids (pid) matching the ones in SSN distance matrices, so that a mapping can occur.

Value

A list with the model fit

It returns a ssnbayes object (similar to stan returns). It includes the formula used to fit the model. The output can be transformed into the stanfit class using class(fits) <- c("stanfit").

Author(s)

Edgar Santos-Fernandez

Examples

```
## Not run:
#options(mc.cores = parallel::detectCores())
#require(SSNdata)
#formula = temp ~ SLOPE + elev + h2o_area + air_temp + sin + cos
## Imports a data.frame containing observations and covariates
#clear <- readRDS(system.file("extdata/clear_obs.RDS", package = "SSNdata"))
#family <-  "gaussian"

# # If using osm_ssn:
# # Generate osm_ssn
# clear_osm_ssn <- generate_osm_ssn(clear, "long", "lat", root_loc = 12, plot_network = TRUE)
# fit_ar <- ssnbayes2(formula,
#                     data = clear,
#                     osm_ssn = clear_osm_ssn,
#                     family = family,
#                     time_method = list("ar", "date"),
#                     space_method = list('use_osm_ssn', c("Exponential.taildown")),
#                     iter = 2000,
#                     warmup = 1000,
#                     chains = 3,
#                     cores = 3)
#
#
#
# If not using osm_ssn, and instead using SSN:
# Import SpatialStreamNetwork object
# path <- system.file("extdata/clearwater.ssn", package = "SSNbayes")
# fit_ar <- ssnbayes2(formula = formula,
#                     data = clear,
#                     path = path,
#                     family = family,
#                     time_method = list("ar", "date"),
#                     space_method = list('use_ssn', c("Exponential.taildown")),
#                     iter = 2000,
#                     warmup = 1000,
#                     chains = 3,
#                     cores = 3,
#                     net = 2, # second network on the ssn object
#                     addfunccol='afvArea')
#
```

```
#
#
# #space_method options examples
# #use list('no_ssn', 'Exponential.Euclid', c('lon', 'lat')) if no ssn object is available

## End(Not run)
```

Index