Package 'geocausal'

March 25, 2024

Type Package

Title Causal Inference with Spatio-Temporal Data

Version 0.3.0

Maintainer Mitsuru Mukaigawara <mitsuru_mukaigawara@g.harvard.edu>

Description Spatio-temporal causal inference based on point process data. You provide the raw data of locations and timings of treatment and outcome events, specify counterfactual scenarios, and the package estimates causal effects over specified spatial and temporal windows. See Papadogeorgou, et al. (2022) <doi:10.1111/rssb.12548>.

License MIT + file LICENSE

URL https://github.com/mmukaigawara/geocausal

Encoding UTF-8

LazyData true

Suggests elevatr, geosphere, gridExtra, ggthemes, knitr, readr

Imports data.table, dplyr, furrr, ggplot2, ggpubr, latex2exp, mclust, progressr, purrr, sf, spatstat.explore, spatstat.geom, spatstat.model, terra, tidyr, tidyselect, tidyterra

RoxygenNote 7.3.1

Depends R (>= 3.5.0)

NeedsCompilation no

Author Mitsuru Mukaigawara [cre, aut] (<https://orcid.org/0000-0001-6530-2083>), Georgia Papadogeorgou [aut] (<https://orcid.org/0000-0002-1982-2245>), Jason Lyall [aut] (<https://orcid.org/0000-0001-9117-7503>), Kosuke Imai [aut] (<https://orcid.org/0000-0002-2748-1022>)

Repository CRAN

Date/Publication 2024-03-25 21:30:02 UTC

R topics documented:

airstrikes . . . 2
airstrikes_base . . . 3
conv_owin_into_sf . . . 4
get_base_dens . . . 4
get_cf_dens . . . 5
get_cf_sum_log_intens . . . 6
get_distexp . . . 6
get_dist_focus . . . 7
get_dist_line . . . 8
get_elev . . . 9
get_est . . . 10
get_estimates . . . 11
get_hfr . . . 12
get_hist . . . 14
get_obs_dens . . . 15
get_power_dens . . . 16
get_var_bound . . . 17
get_weighted_surf . . . 18
get_window . . . 19
insurgencies . . . 19
iraq_window . . . 20
plot.est . . . 20
plot.hyperframe . . . 21
plot.im . . . 22
plot.obs . . . 22
predict_obs_dens . . . 23
sim_cf_dens . . . 24
sim_power_dens . . . 25
smooth_ppp . . . 27
summary.est . . . 28
summary.obs . . . 29

Index **30**

`airstrikes` *airstrikes*

Description

A subset of airstrikes data in Iraq (March to June 2007)

Usage

```
airstrikes
```

Format

A tibble with 3938 rows and 4 variables:

date Date (YYYY-MM-DD)

longitude Longitudes (decimal)

latitude Latitudees (decimal)

type Types of airstrikes (airstrikes or shows of force (SOF))

Examples

```
airstrikes
```

`airstrikes_base` *airstrikes_base*

Description

A subset of airstrikes data in Iraq (a subset of airstrikes in 2006) that can be used to construct baseline densities

Usage

```
airstrikes_base
```

Format

A tibble with 808 rows and 3 variables:

date Date

longitude Longitudes (decimal)

latitude Latitudees (decimal)

Examples

```
airstrikes_base
```

conv_owin_into_sf *Convert windows into sf objects*

Description

'conv_owin_into_sf' takes an owin object and converts it to sf-related objects. This function is mostly an internal function of other functions.

Usage

```
conv_owin_into_sf(window)
```

Arguments

window	owin object

Value

list of polygon, dataframe, sfc_POLYGON, sf, and SpatialPolygonsDataFrame objects

get_base_dens *Get the baseline density*

Description

'get_base_dens()' takes a dataframe and returns the baseline densities using Scott's rule of thumb (out-of-sample data) or fitting an inhomogeneous Poisson model (in-sample data) by regressing the in-sample data on time-invariant covariates.

Usage

```
get_base_dens(
  window,
  option,
  ndim = 256,
  out_data,
  out_coordinates = c("longitude", "latitude"),
  hfr,
  dep_var,
  indep_var,
  ratio
)
```

Arguments

window	owin object
option	"in" (using in-sample data) or "out" (using out-of-sample data)
ndim	the number of dimensions of grid cells (ndim^2). By default, ndim = 256.
out_data	dataframe (if using out-of-sample data)
out_coordinates	vector of column names of longitudes and latitudes (in this order) (if using in-sample data)
hfr	hyperframe (if using in-sample data)
dep_var	the name of the dependent variable (if using in-sample data)
indep_var	the names of time-invariant independent varaibles (if using in-sample data)
ratio	for random sampling of data (if using in-sample data)

Value

an im object of baseline density

Examples

```
get_base_dens(option = "out",
              out_data = airstrikes_base,
              out_coordinates = c("longitude", "latitude"),
              window = iraq_window,
              ndim = 256)
```

---

get_cf_dens *Get counterfactual densities*

Description

'get_cf_dens' takes the target (expected) number, baseline density, and power density, and generates a hyperframe with counterfactual densities.

Usage

```
get_cf_dens(expected_number, base_dens, power_dens = NA, window)
```

Arguments

expected_number	the expected number of observations.
base_dens	baseline density (im object)
power_dens	power density (im object)
window	owin object

Details

There are two ways of generating counterfactual densities. First, users can keep the locations of observations as they are and change the expected number of observations. In this case, users do not have to set ‘power_dens‘ and simply modify ‘expected_number‘. Alternatively, users can shift the locations as well. In this case, ‘power_dens‘ must be specified. To obtain power densities, refer to [get_power_dens()].

Value

an im object of a counterfactual density

`get_cf_sum_log_intens` *Calculate the log counterfactual densities*

Description

A function that takes a hyperframe and returns the log counterfactual densities ie, the numerator of the equation

Usage

```
get_cf_sum_log_intens(cf_dens, treatment_data)
```

Arguments

`cf_dens`	A counterfactual density (an im object)
`treatment_data`	In the form of hyperframe$column

Value

A numeric vector of sums of log densities for each time period

---

`get_distexp` *Get the expectation of treatment events with arbitrary distances*

Description

‘get_distexp()‘ takes counterfactual densities and and returns the expected number of treatment events based on distances from a user-specified focus.

Usage

```
get_distexp(
  cf_sim_results,
  entire_window,
  dist_map,
  dist_map_unit = "km",
  grayscale = FALSE,
  use_raw = FALSE
)
```

Arguments

cf_sim_results	output of 'sim_cf_dens()'
entire_window	owin object of the entire region
dist_map	im object whose cell values are the distance from a focus (e.g., city)
dist_map_unit	either '"km"' or '"mile"'
grayscale	logical. 'grayscale' specifies whether to convert plot to grayscale (by default, FALSE).
use_raw	logical. 'use_raw' specifies whether to use the raw value of expectations or percentiles. By default, 'FALSE'.

Value

A list of ggplot objects that summarizes how expectations change over distances from a focus ('expectation_plot') and summarizes distances and areas ('window_plot'). Note that the second object can not necessarily be well drawn depending on how windows are defined.

---

get_dist_focus *Get distance maps*

Description

'get_dist_focus()' generates a distance map from focus locations.

Usage

```
get_dist_focus(window, lon, lat, resolution, mile = FALSE, preprocess = FALSE)
```

Arguments

window	owin object
lon	vector of longitudes
lat	vector of latitudes
resolution	resolution of raster objects

mile	logical. ‘mile‘ specifies whether to return the output in miles instead of kilometers (by default, FALSE).
preprocess	logical. ‘preprocess‘ specifies whether to first pick the potentially closest point. It is recommended to set ‘preprocess = TRUE‘ if users need to obtain distances from many points.

Details

‘get_dist_focus()‘ depends on ‘geosphere::distVincentyEllipsoid()‘. Since it calculates accurate distances considering the ellipsoid, the process sometimes becomes computationally demanding, namely when we need to obtain distances from many points. In that case, users can set ‘preprocess = TRUE‘. With this option, ‘get_dist_focus()‘ calculates distances from points by first identifying the closest point using ‘sf::st_nearest_feature()‘ with approximations. This process is more efficient than computing distances from all the points with ‘geosphere::distVincentyEllipsoid()‘ and then obtaining the minimum of all the distances. By default, ‘get_dist_focus()‘ returns distances in kilometers unless users set ‘mile = TRUE‘.

Value

an im object

Examples

```
get_dist_focus(window = iraq_window,
               lon = c(44.366), #Baghdad
               lat = c(33.315),
               resolution = 0.5,
               mile = FALSE,
               preprocess = FALSE)
```

---

get_dist_line *Get distance maps from lines and polygons*

Description

‘get_dist_line()‘ generates a distance map from lines and polygons.

Usage

```
get_dist_line(
  window,
  path_to_shapefile,
  line_data = NULL,
  mile = FALSE,
  resolution,
  preprocess = TRUE
)
```

Arguments

window	owin object
path_to_shapefile	path to shapefile
line_data	sfc_MULTILINESTRING file (If available. If not, 'get_dist_line()' creates it from a shapefile.)
mile	logical. 'mile' specifies whether to return the output in miles instead of kilometers (by default, FALSE).
resolution	resolution of raster objects
preprocess	logical. 'preprocess' specifies whether to first pick the potentially closest point. It is recommended to set 'preprocess = TRUE' if users need to obtain distances from many points.

Value

an im object

---

get_elev *Get elevation data*

Description

'get_elevation()' takes a directory that hosts shapefile and returns an owin object of altitudes.

Usage

```
get_elev(load_path, ...)
```

Arguments

load_path	path to the shp file (note: a folder)
...	other parameters passed to 'elevatr::get_elev_raster()'. The resolution argument z must be specified.

Value

an im object (unit: meters)

get_est *Get causal estimates comparing two scenarios*

Description

'get_est()' generates causal estimates comparing two counterfactual scenarios.

Usage

```
get_est(
  obs,
  cf1,
  cf2,
  treat,
  sm_out,
  mediation = FALSE,
  obs_med_log_sum_dens = NA,
  cf1_med_log_sum_dens = NA,
  cf2_med_log_sum_dens = NA,
  lag,
  time_after = TRUE,
  entire_window,
  use_dist,
  windows,
  dist_map,
  dist,
  trunc_level = NA
)
```

Arguments

obs	observed density
cf1	counterfactual density 1
cf2	counterfactual density 2
treat	column of a hyperframe that summarizes treatment data. In the form of 'hyperframe$column'.
sm_out	column of a hyperframe that summarizes the smoothed outcome data
mediation	whether to perform causal mediation analysis (don't use; still in development). By default, FALSE.
obs_med_log_sum_dens	sum of log densities of mediators for the observed (don't use; still in development)
cf1_med_log_sum_dens	sum of log densities of mediators for counterfactual 1 (don't use; still in development)

`cf2_med_log_sum_dens`
: sum of log densities of mediators for counterfactual 2 (don't use; still in development)

`lag`
: integer that specifies lags to calculate causal estimates

`time_after`
: whether to include one unit time difference between treatment and outcome. By default = TRUE

`entire_window`
: owin object (the entire region of interest)

`use_dist`
: whether to use distance-based maps. By default, TRUE

`windows`
: a list of owin objects (if 'use_dist = FALSE')

`dist_map`
: distance map (an im object, if 'use_dist = TRUE')

`dist`
: distances (a numeric vector within the max distance of 'dist_map')

`trunc_level`
: the level of truncation for the weights (0-1)

Details

The level of truncation indicates the quantile of weights at which weights are truncated. That is, if 'trunc_level = 0.95', then all weights are truncated at the 95 percentile of the weights.

Value

list of the following: 'cf1_ave_surf': average weighted surface for scenario 1 'cf2_ave_surf': average weighted surface for scenario 2 'est_cf': estimated effects of each scenario 'est_causal': estimated causal contrasts 'var_cf': variance upper bounds for each scenario 'var_causal': variance upper bounds for causal contrasts 'windows': list of owin objects

---

`get_estimates` *Generate a Hajek estimator*

Description

A function that returns a Hajek estimator of causal contrasts

Usage

```
get_estimates(
  weighted_surf_1,
  weighted_surf_2,
  use_dist = TRUE,
  windows,
  dist_map,
  dist,
  entire_window
)
```

Arguments

`weighted_surf_1`	a weighted surface for scenario 1
`weighted_surf_2`	another weighted surface for scenario 2
`use_dist`	whether to use distance-based maps. By default, TRUE
`windows`	a list of owin objects (if ‘use_dist = FALSE‘)
`dist_map`	distance map (an im object, if ‘use_dist = TRUE‘)
`dist`	distances (a numeric vector within the max distance of ‘dist_map‘)
`entire_window`	an owin object of the entire map

Details

‘get_estimates()‘ is an internal function to ‘get_est()‘ function, performing the estimation analysis after ‘get_weighted_surf()‘ function

Value

list of Hajek estimators for each scenario (‘est_haj‘), causal contrasts (Hajek estimator) as a matrix (‘est_tau_haj_matrix‘), and causal contrast (scenario 2 - scenario 1) as a numeric vector (‘est_tau_haj_cf2_vs_cf1‘), along with weights, windows, and smoothed outcomes

---

`get_hfr` *Create a hyperframe*

Description

‘get_hfr()‘ takes a dataframe with time and location variables and generates a hyperframe with point patterns. ‘get_hfr()‘ is usually the first function that users employ in order to perform spatiotemporal causal inference analytic methods.

Usage

```
get_hfr(
  data,
  col,
  window,
  time_col,
  time_range,
  coordinates = c("longitude", "latitude"),
  combine = TRUE
)
```

Arguments

data	dataframe. The dataframe must have time and location variables. Location variables should be standard coordinates (i.e., longitudes and latitudes).
col	the name of the column for subtypes of events of interest
window	owin object (for more information, refer to 'spatstat.geom::owin()'). Basically, an owin object specifies the geographical boundaries of areas of interest.
time_col	the name of the column for time variable. Note that the time variable must be integers.
time_range	numeric vector. 'time_range' specifies the range of the time variable (i.e., min and max of the time variable). The current version assumes that the unit of this time variable is dates.
coordinates	character vector. 'coordinates' specifies the names of columns for locations. By default, 'c("longitude", "latitude")' in this order. Note that the coordinates must be in decimal degree formats.
combine	logical. 'combine' tells whether to generate output for all subtypes of events combined. By default, 'TRUE', which means that a column of ppp objects with all subtypes combined is generated in the output.

Value

A hyperframe is generated with rows representing time and columns representing the following: * The first column: time variable * The middle columns: ppp objects (see 'spatstat.geom::ppp()') generated for each subtype of events of interest * The last column (if 'combine = TRUE'): ppp objects with all subtypes combined. This column is named as 'all_combined'.

Examples

```
# Data
dat <- data.frame(time = c(1, 1, 2, 2),
                  longitude = c(43.9, 44.5, 44.1, 44.0),
                  latitude = c(33.6, 32.7, 33.6, 33.5),
                  type = rep(c("treat", "out"), 2))

# Hyperframe
get_hfr(data = dat,
        col = "type",
        window = iraq_window,
        time_col = "time",
        time_range = c(1, 2),
        coordinates = c("longitude", "latitude"),
        combine = FALSE)
```

get_hist *Obtain histories of treatment or outcome events*

Description

'get_hist()' takes a hyperframe and time and columns of interest, and generates histories of events of interest.

Usage

```
get_hist(tt, Xt, Yt = NA, lag, window, x_only = TRUE)
```

Arguments

tt	values of the time variable of interest for which 'get_hist()' generates histories
Xt	the name of a treatment column
Yt	the name of an outcome column
lag	numeric. 'lag' specifies the number of time periods over which 'get_hist()' aggregates treatment and outcome columns.
window	owin object.
x_only	logical. 'x_only' specifies whether to generate only treatment history (no outcome history). By default, 'FALSE'.

Value

list of treatment and outcome histories

Examples

```
dat_out <- insurgencies[1:100, ]
dat_out$time <- as.numeric(dat_out$date - min(dat_out$date) + 1)

# Hyperframe
dat_hfr <- get_hfr(data = dat_out,
                   col = "type",
                   window = iraq_window,
                   time_col = "time",
                   time_range = c(1, max(dat_out$time)),
                   coordinates = c("longitude", "latitude"),
                   combine = TRUE)

# Histories
lapply(1:nrow(dat_hfr), get_hist,
       Xt = dat_hfr$all_outcome,
       lag = 1, window = iraq_window)
```

get_obs_dens *Generate observed densities*

Description

'get_obs_dens()' takes a hyperframe and returns observed densities. The output is used as propensity scores.

Usage

```
get_obs_dens(hfr, dep_var, indep_var, ngrid = 100, window)
```

Arguments

hfr	hyperframe
dep_var	The name of the dependent variable. Since we need to obtain the observed density of treatment events, 'dep_var' should be the name of the treatment variable.
indep_var	vector of names of independent variables (covariates)
ngrid	the number of grid cells that is used to generate observed densities. By default = 100. Notice that as you increase 'ngrid', the process gets computationally demanding.
window	owin object

Details

'get_obs_dens()' assumes the poisson point process model and calculates observed densities for each time period. It depends on 'spatstat.model::mppm()'. Users should note that the coefficients in the output are not directly interpretable, since they are the coefficients inside the exponential of the poisson model.

Value

list of the following: * 'indep_var': independent variables * 'coef': coefficients * 'intens_grid_cells': im object of observed densities for each time period * 'estimated_counts': the number of events that is estimated by the poisson point process model for each time period * 'sum_log_intens': the sum of log intensities for each time period

Examples

```
# Data
dat_out <- insurgencies[1:100, ]
dat_out$time <- as.numeric(dat_out$date - min(dat_out$date) + 1)

# Hyperframe
dat_hfr <- get_hfr(data = dat_out,
                   col = "type",
                   window = iraq_window,
```

```
                    time_col = "time",
                    time_range = c(1, max(dat_out$time)),
                    coordinates = c("longitude", "latitude"),
                    combine = TRUE)

# Covariates
dist_baghdad <- get_dist_focus(window = iraq_window,
                               lon = c(44.366), #Baghdad
                               lat = c(33.315),
                               resolution = 0.1,
                               mile = FALSE,
                               preprocess = FALSE)

dat_hfr$dist_bagh <- dist_baghdad

# Observed density
get_obs_dens(dat_hfr,
             dep_var = "all_combined",
             indep_var = c("dist_bagh"),
             ngrid = 100,
             window = iraq_window)
```

`get_power_dens` *Get power densities*

Description

'get_power_dens()' takes the target densities and their priorities and returns a power density.

Usage

```
get_power_dens(target_dens, priorities, window)
```

Arguments

`target_dens`	list of target densities
`priorities`	vector of priorities for each of target densities
`window`	owin object

Value

list of an im object and a ggplot object of power densities

Examples

```
# Density 1: Distance from Mosul
dist_from_mosul <- get_dist_focus(window = iraq_window,
                                  lon = c(43.158),
                                  lat = c(36.349),
```

```
                                    resolution = 0.5,
                                    mile = FALSE,
                                    preprocess = FALSE)

# Density 2: Distance from Baghdad
dist_from_baghd <- get_dist_focus(window = iraq_window,
                                    lon = c(44.366),
                                    lat = c(33.315),
                                    resolution = 0.5,
                                    mile = FALSE,
                                    preprocess = FALSE)

# Power density
get_power_dens(target_dens = list(dist_from_mosul, dist_from_baghd),
               priorities = c(3, 2),
               window = iraq_window)
```

`get_var_bound` *Calculate variance upper bounds*

Description

A function that calculates variance upper bounds

Usage

```
get_var_bound(estimates)
```

Arguments

`estimates`	an object returned from ‘get_est()‘ function

Details

‘get_var_bound()‘ is an internal function to ‘get_estimates()‘ function, performing the estimation analysis after ‘get_est()‘ function

Value

list of variance upper bounds for each scenario (‘bound_haj‘) and causal contrasts (‘bound_tau_haj‘). Note that this function returns variance upper bounds for Hajek estimators

`get_weighted_surf` *Generate average weighted surfaces*

Description

A function that returns averaged weighted surfaces (both IPW and Hajek) along with weights

Usage

```
get_weighted_surf(
  obs_dens,
  cf_dens,
  mediation = FALSE,
  obs_med_log_sum_dens,
  cf_med_log_sum_dens,
  treatment_data,
  smoothed_outcome,
  lag,
  entire_window,
  time_after,
  truncation_level = truncation_level
)
```

Arguments

`obs_dens`	observed density
`cf_dens`	counterfactual density
`mediation`	whether to perform causal mediation analysis. By default, FALSE.
`obs_med_log_sum_dens`	sum of log densities of mediators for the observed (if 'mediation = TRUE')
`cf_med_log_sum_dens`	sum of log densities of mediators for counterfactual (if 'mediation = TRUE')
`treatment_data`	column of a hyperframe that summarizes treatment data. In the form of 'hyperframe$column'.
`smoothed_outcome`	column of a hyperframe that summarizes the smoothed outcome data
`lag`	integer that specifies lags to calculate causal estimates
`entire_window`	owin object (the entire region of interest)
`time_after`	whether to include one unit time difference between treatment and outcome
`truncation_level`	the level at which the weights are truncated (see 'get_estimates()')

Details

'get_weighted_surf()' is an internal function to 'get_estimates()' function. If 'time_after' is TRUE, then this function uses treatment data and weights from lag to nrow(data)-1, and outcome data from lag+1 to nrow(data).

Value

list of an average weighted surface ('avarage_surf', an 'im' object), a Hajek average weighted surface ('average_weighted_surf_haj', an 'im' object), weights, and smoothed outcomes

---

get_window *Generate a window*

Description

'get_window()' takes a directory that hosts a shapefile and returns an owin object.

Usage

```
get_window(load_path)
```

Arguments

load_path	path to the shp file

Value

owin object

---

insurgencies *insurgencies*

Description

A subset of insurgencies data in Iraq (March to June 2007)

Usage

```
insurgencies
```

Format

A tibble with 68573 rows and 4 variables:

date Date (YYYY-MM-DD)

longitude Longitudes (decimal)

latitude Latitudees (decimal)

type Types of insurgencies (improvised explosive devices (IED), small arms fire (SAF), or other)

Examples

```
insurgencies
```

`iraq_window` *iraq_window*

Description

An owin object of Iraq

Usage

```
iraq_window
```

Format

A polygonal object:

type Polygonal
xrange Range (longitude)
yrange Range (latitude)
bdry Boundaries
units Units

Examples

```
iraq_window
```

`plot.est` *Plot estimates*

Description

Plot estimates

Usage

```
## S3 method for class 'est'
plot(x, ..., lim = NA)
```

Arguments

`x`	input
`...`	arguments passed on to the function
`lim`	limits of the scale. By default, NA. To set limits manually, provide a vector or max and min

`plot.hyperframe` *Plot estimates*

Description

Plot estimates

Usage

```
## S3 method for class 'hyperframe'
plot(
  x,
  ...,
  col,
  time_col = "time",
  range,
  lim = NA,
  scalename = NA,
  color = c("white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", "#771F59FF"),
  combined = TRUE
)
```

Arguments

`x`	input
`...`	arguments passed on to the function
`col`	the name/s of a column of interest. To specify multiple columns, users should list column names as a character vector.
`time_col`	The name of the column of time variable. By default, ‘"time"‘. Note that the time variable must be integers.
`range`	vector that specifies the range of tiem variable (e.g., ‘c("2007-01-01", "2007-01-31")‘)
`lim`	limits of the scale. By default, NA. To set limits manually, provide a vector or max and min
`scalename`	the name of the scale (for images only)
`color`	the color scale. By default, "white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", and "#771F59FF".
`combined`	logical. ‘combined‘ specifies whether to combine all the point processes to one plot. This argument applies only to the case when users specify one column with multiple time periods. By default = TRUE

plot.im *Plot im*

Description

Plot im

Usage

```
## S3 method for class 'im'
plot(
  x,
  ...,
  main = "Image object",
  scalename = "Density",
  grayscale = "FALSE",
  color = c("white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", "#771F59FF"),
  lim = NA
)
```

Arguments

x	input
...	arguments passed on to the function
main	title
scalename	the name of the scale (for images only)
grayscale	whether to use grayscale. By default, FALSE.
color	the color scale. By default, "white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", and "#771F59FF".
lim	limits of the scale. By default, NA. To set limits manually, provide a vector or max and min

plot.obs *Plot observed densities*

Description

Plot observed densities

Usage

```
## S3 method for class 'obs'
plot(x, ..., dens_2 = NA, dens_3 = NA, actual_data = NA, time_unit = NA)
```

Arguments

x	input
...	arguments passed on to the function
dens_2	density 2 (if any). By default, 'NA'.
dens_3	density 3 (if any). By default, 'NA'.
actual_data	actual data in the form of 'hyperframe$column'.
time_unit	x-axis label of the output

---

predict_obs_dens *Perform out-of-sample prediction*

Description

'predict_obs_dens()' performs out-of-sample prediction (separating data into training and test sets). It assumes that training and test sets have the same window.

Usage

```
predict_obs_dens(hfr, ratio, dep_var, indep_var, ngrid = 100, window)
```

Arguments

hfr	hyperframe
ratio	numeric. ratio between training and test sets
dep_var	dependent variables
indep_var	independent variables
ngrid	the number of grids. By default, '100'.
window	owin object

Value

list of the following: * 'indep_var': independent variables * 'coef': coefficients * 'intens_grid_cells': im object of observed densities for each time period * 'estimated_counts': the number of events that is estimated by the poisson point process model for each time period * 'sum_log_intens': the sum of log intensities for each time period * 'training_row_max': the max row ID of the training set

`sim_cf_dens` *Simulate counterfactual densities*

Description

'sim_cf_dens()' takes a list of power densities and returns simulated counterfactual densities.

Usage

```
sim_cf_dens(
  expected_number,
  base_dens,
  power_sim_results,
  window,
  grayscale = FALSE,
  color = c("white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", "#771F59FF")
)
```

Arguments

`expected_number`	the expected number of observations
`base_dens`	the baseline density (im object)
`power_sim_results`	the results obtained by 'simulate_power_density()'
`window`	owin object
`grayscale`	logical. 'grayscale' specifies whether to convert plot to grayscale (by default, FALSE).
`color`	the color scale. By default, "white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", and "#771F59FF".

Value

list of counterfactual densities, a ggplot, and priorities

Examples

```
# Baseline density
baseline <- get_base_dens(option = "out",
                          out_data = airstrikes_base,
                          out_coordinates = c("longitude", "latitude"),
                          window = iraq_window,
                          ndim = 64)

# Density 1: Distance from Mosul
dist_from_mosul <- get_dist_focus(window = iraq_window,
                                  lon = c(43.158),
```

```
                                    lat = c(36.349),
                                    resolution = 0.5,
                                    mile = FALSE,
                                    preprocess = FALSE)

# Density 2: Distance from Baghdad
dist_from_baghd <- get_dist_focus(window = iraq_window,
                                    lon = c(44.366),
                                    lat = c(33.315),
                                    resolution = 0.5,
                                    mile = FALSE,
                                    preprocess = FALSE)

# Simulation of power density
sim_power_mosul <- sim_power_dens(target_dens = list(dist_from_baghd),
                                    dens_manip = dist_from_mosul,
                                    priorities = 1,
                                    priorities_manip = c(1, 2, 5, 10, 15, 50),
                                    window = iraq_window,
                                    grayscale = FALSE)

# Simulation of counterfactual density
sim_cf_dens(expected_number = 3,
            base_dens = baseline,
            power_sim_results = sim_power_mosul,
            window = iraq_window,
            grayscale = FALSE)
```

`sim_power_dens` *Simulate power densities*

Description

A function that takes the target densities and their priorities and returns a power density image over a range of parameters

Usage

```
sim_power_dens(
  target_dens,
  dens_manip,
  priorities,
  priorities_manip,
  window,
  color = c("white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", "#771F59FF"),
  grayscale = FALSE
)
```

Arguments

target_dens	list of target densities. This should always be a list, even if there is only one target density.
dens_manip	a target density for which we manipulate the value of priorities
priorities	numeric. 'priorities' specifies the priority for the target density that we do not manipulate.
priorities_manip	vector of priorities for the density that we manipulate.
window	owin object
color	the color scale. By default, "white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", and "#771F59FF".
grayscale	logical. 'grayscale' specifies whether to convert plot to grayscale (by default, FALSE).

Value

list of densities, plot, and priorities

Examples

```
# Density 1: Distance from Mosul
dist_from_mosul <- get_dist_focus(window = iraq_window,
                                  lon = c(43.158),
                                  lat = c(36.349),
                                  resolution = 0.5,
                                  mile = FALSE,
                                  preprocess = FALSE)

# Density 2: Distance from Baghdad
dist_from_baghd <- get_dist_focus(window = iraq_window,
                                  lon = c(44.366),
                                  lat = c(33.315),
                                  resolution = 0.5,
                                  mile = FALSE,
                                  preprocess = FALSE)

# Simulation
sim_power_dens(target_dens = list(dist_from_baghd),
               dens_manip = dist_from_mosul,
               priorities = 1,
               priorities_manip = c(1, 2, 5, 10, 15, 50),
               window = iraq_window,
               grayscale = FALSE)
```

smooth_ppp *Smooth outcome events*

Description

‘smooth_ppp()‘ takes a column of hyperframes (ppp objects) and smoothes them.

Usage

```
smooth_ppp(data, method, sampling = NA)
```

Arguments

data	the name of a hyperframe and column of interest. ‘data‘ should be in the form of ‘"hyperframe$column"‘.
method	methods for smoothing ppp objects. Either ‘"mclust"‘ or ‘"abramson"‘. See details.
sampling	numeric between 0 and 1. ‘sampling‘ determines the proportion of data to use for initialization. By default, NA (meaning that it uses all data without sampling).

Details

To smooth ppp objects, users can choose either the Gaussian mixture model (‘method = "mclust"‘) or Abramson’s adaptive smoothing (‘method = "abramson"‘). The Gaussian mixture model is essentially the method that performs model-based clustering of all the observed points. In this package, we employ the EII model (equal volume, round shape (spherical covariance)). This means that we model observed points by several Gaussian densities with the same, round shape. This is why this model is called fixed-bandwidth smoothing. This is a simple model to smooth observed points, yet given that analyzing spatiotemporal data is often computationally demanding, it is often the best place to start (and end). Sometimes this process can also take time, which is why an option for ‘init‘ is included in this function.

Another, more precise, method for smoothing outcomes is adaptive smoothing (‘method = "abram"‘). This method allows users to vary bandwidths based on ‘Abramson (1982)‘. Essentially, this model assumes that the bandwidth is inversely proportional to the square root of the target densities. Since the bandwidth is adaptive, the estimation is usually more precise than the Gaussian mixture model. However, the caveat is that this method is often extremely computationally demanding.

Value

im objects

Examples

```
# Time variable
dat_out <- insurgencies[1:100, ]
dat_out$time <- as.numeric(dat_out$date - min(dat_out$date) + 1)

# Hyperframe
dat_hfr <- get_hfr(data = dat_out,
                   col = "type",
                   window = iraq_window,
                   time_col = "time",
                   time_range = c(1, max(dat_out$time)),
                   coordinates = c("longitude", "latitude"),
                   combine = TRUE)

# Smoothing outcome
smooth_ppp(data = dat_hfr$all_combined,
           method = "mclust",
           sampling = 0.05)
```

`summary.est` *Summarize results*

Description

'summary' functions take the output and summarize it.

Usage

```
## S3 method for class 'est'
summary(object, ...)
```

Arguments

`object`	an output object
`...`	arguments passed on to the function

Details

Currently, observed densities (class: obs) and estimates (class: est) are supported by this function.

summary.obs *Summarize results*

Description

'summary' functions take the output and summarize it.

Usage

```
## S3 method for class 'obs'
summary(object, ...)
```

Arguments

object	an output object
...	arguments passed on to the function

Details

Currently, observed densities (class: obs) and estimates (class: est) are supported by this function.

Index

∗ **datasets**
airstrikes, 2
airstrikes_base, 3
insurgencies, 19
iraq_window, 20

airstrikes, 2
airstrikes_base, 3

conv_owin_into_sf, 4

get_base_dens, 4
get_cf_dens, 5
get_cf_sum_log_intens, 6
get_dist_focus, 7
get_dist_line, 8
get_distexp, 6
get_elev, 9
get_est, 10
get_estimates, 11
get_hfr, 12
get_hist, 14
get_obs_dens, 15
get_power_dens, 16
get_var_bound, 17
get_weighted_surf, 18
get_window, 19

insurgencies, 19
iraq_window, 20

plot.est, 20
plot.hyperframe, 21
plot.im, 22
plot.obs, 22
predict_obs_dens, 23

sim_cf_dens, 24
sim_power_dens, 25
smooth_ppp, 27
summary.est, 28
summary.obs, 29