Package 'geocausal'

September 21, 2023

Type Package

Title Causal Inference with Spatio-Temporal Data

Version 0.2.0

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, rmarkdown

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

RoxygenNote 7.2.3

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 2023-09-21 02:40:02 UTC

R topics documented:

`airstrikes` *airstrikes*

Description

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

Usage

```
airstrikes
```

Format

A tibble with 936 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 January 2006) that can be used to construct baseline densities

Usage

```
airstrikes_base
```

Format

A tibble with 148 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 baseline densities*

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,
  grayscale = FALSE,
  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)
grayscale	logical. ‘grayscale‘ specifies whether to convert plot to grayscale (by default, FALSE).
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

list of the following: * ‘density‘: im object of baseline density * ‘density_plot‘: ggplot object of baseline density

Examples

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

---

get_base_sum *Summarize baseline densities*

Description

‘get_base_sum()‘ takes dataframe and summarizes it.

Usage

```
get_base_sum(data, time_column)
```

Arguments

data	dataframe
time_column	the name of time variable

Value

list of the following: * ‘plot‘: ggplot object that shows temporal distribution of data * ‘quantile‘: quantiles of counts * ‘mean‘: the mean * ‘variance‘: the variance

Examples

```
get_base_sum(data = airstrikes_base,
             time_column = "date")
```

`get_causal_cont` *Generate a figure with causal contrasts*

Description

A function that returns a figure with causal contrasts

Usage

```
get_causal_cont(scenario_1, scenario_2, dist_map_unit = "km", grayscale)
```

Arguments

`scenario_1`	a counterfactual scenario (the output of ‘get_estimates()‘ function)
`scenario_2`	another counterfactual scenario
`dist_map_unit`	either ‘"km"‘ or ‘"mile"‘
`grayscale`	whether to grayscale the figures

Details

‘get_causal_cont()‘ returns a figure with causal contrasts. A causal contrast is the difference between expected numbers of outcome events of two counterfactual scenarios. The baseline scenario is ‘scenario_1‘.

Value

a list of ggplot objects and the data

`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,
  grayscale = FALSE
)
```

Arguments

`expected_number`	the expected number of observations.
`base_dens`	baseline density (im object)
`power_dens`	power density (im object)
`window`	owin object
`grayscale`	logical. 'grayscale' specifies whether to convert plot to grayscale (by default, FALSE).

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

A list of counterfactual density (im object) and a ggplot object

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_dist_based_exp *Get the expectation of treatment events with arbitrary distances*

Description

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

Usage

```
get_dist_based_exp(
  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

ggplot object that summarizes how expectations change over distances from a focus

---

get_dist_focus *Get distance maps*

Description

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

Usage

```
get_dist_focus(
  window,
  longitude,
  latitude,
  resolution,
  grayscale,
  mile = FALSE,
  preprocess = FALSE
)
```

Arguments

window	owin object
longitude	vector of longitudes
latitude	vector of latitudes
resolution	resolution of raster objects
grayscale	logical. ‘grayscale‘ specifies whether to convert plot to grayscale (by default, FALSE).
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

A list of im and ggplot object

Examples

```
get_dist_focus(window = iraq_window,
               longitude = c(44.366), #Baghdad
               latitude = c(33.315),
               resolution = 0.5,
               grayscale = FALSE,
               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,
  grayscale,
  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.)
grayscale	logical. 'grayscale' specifies whether to convert plot to grayscale (by default, FALSE).
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

A list of im and ggplot object

---

get_elevation *Get elevation data*

Description

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

Usage

```
get_elevation(load_path, ...)
```

Arguments

load_path	path to the shp file (note: a folder)
...	other parameters passed to 'elevatr::get_elev_raster()'

Value

list of a raster layer, an im object, and a ggplot object of altitudes (in meters).

`get_estimates` *Get causal estimates comparing two scenarios*

Description

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

Usage

```
get_estimates(
  obs_dens,
  cf_dens,
  treatment_data,
  smoothed_outcome,
  lag,
  entire_window,
  dist_map,
  dist,
  dist_map_unit = "km",
  grayscale
)
```

Arguments

`obs_dens`	observed density
`cf_dens`	counterfactual density
`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)
`dist_map`	im object (distance map); notice that the unit (either mile or km) is inherited from the distance map
`dist`	a vector of distances for which the function calculates the expectations (e.g., 'c(50, 100, 150, 200)')
`dist_map_unit`	either '"km"' or '"mile"'
`grayscale`	logical. 'grayscale' specifies whether to convert plot to grayscale (by default, FALSE).

Value

list of the following: ‘average_expected_events‘: Hajek estimators (counts, the entire window) ‘average_expected_events_quantiles‘: percentiles of Hajek estimators ‘weights‘: weights ‘average_weights‘: mean of ‘weights‘ ‘plot‘: plot showing distance-based expectations ‘distances‘: distances ‘distances_window‘: as window objects ‘expectation_plot‘: plot of expectations ‘window_plot‘: plot of windows

---

`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,
  subtype_column,
  window,
  time_column,
  time_range,
  coordinates = c("longitude", "latitude"),
  combined = TRUE
)
```

Arguments

`data`	dataframe. The dataframe must have time and location variables. Location variables should be standard coordinates (i.e., longitudes and latitudes).
`subtype_column`	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_column`	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.
`combined`	logical. ‘combined‘ 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 'combined = 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,
        subtype_column = "type",
        window = iraq_window,
        time_column = "time",
        time_range = c(1, 2),
        coordinates = c("longitude", "latitude"),
        combined = 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,
                   subtype_column = "type",
                   window = iraq_window,
                   time_column = "time",
                   time_range = c(1, max(dat_out$time)),
                   coordinates = c("longitude", "latitude"),
                   combined = 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,
                   subtype_column = "type",
                   window = iraq_window,
                   time_column = "time",
                   time_range = c(1, max(dat_out$time)),
                   coordinates = c("longitude", "latitude"),
                   combined = TRUE)

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

dat_hfr$dist_bagh <- dist_baghdad$distance_im

# 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, grayscale = FALSE)
```

Arguments

target_dens	list of target densities
priorities	vector of priorities for each of target densities
window	owin object
grayscale	logical. 'grayscale' specifies whether to convert plot to grayscale (by default, FALSE).

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,
                                  longitude = c(43.158),
                                  latitude = c(36.349),
                                  resolution = 0.5,
                                  grayscale = FALSE,
                                  mile = FALSE,
                                  preprocess = FALSE)

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

# Power density
get_power_dens(target_dens = list(dist_from_mosul[[1]], dist_from_baghd[[1]]),
               priorities = c(3, 2),
               window = iraq_window,
               grayscale = FALSE)
```

`get_smoothed_outcome` *Smooth outcome events*

Description

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

Usage

```
get_smoothed_outcome(
  data_interest,
  method,
  initialization = TRUE,
  sampling = 0.05
)
```

Arguments

`data_interest`	the name of a hyperframe and column of interest. ‘data_interest‘ should be in the form of ‘"hyperframe$column"‘.
`method`	methods for smoothing ppp objects. Either ‘"mclust"‘ or ‘"abramson"‘. See details.
`initialization`	logical. ‘initialization‘ specifies whether to use a smaller number of samples to initialize fitting the Gaussian mixture model. By default = TRUE
`sampling`	numeric between 0 and 1. ‘sampling‘ determines the proportion of data to use for initialization (see ‘initialization‘). By default, ‘0.05‘, which means that ‘get_smoothed_outcome()‘ uses 5% of samples for initialization.

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 ‘initialization‘ 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,
                   subtype_column = "type",
                   window = iraq_window,
                   time_column = "time",
                   time_range = c(1, max(dat_out$time)),
                   coordinates = c("longitude", "latitude"),
                   combined = TRUE)

# Smoothing outcome
get_smoothed_outcome(data_interest = dat_hfr$all_combined,
                     method = "mclust",
                     initialization = TRUE,
                     sampling = 0.05)
```

---

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 11872 rows and 4 variables:

date Date (YYYY-MM-DD)
longitude Longitudes (decimal)
latitude Latitudees (decimal)
type Types of insurgencies (improvised explosive devices (IED) or small arms fire (SAF))

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
```

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
)
```

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).

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,
                          grayscale = FALSE,
                          ndim = 64)

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

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

# Simulation of power density
sim_power_mosul <- sim_power_dens(target_dens = list(dist_from_baghd$distance_im),
                                  dens_manip = dist_from_mosul$distance_im,
                                  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$density,
            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,
  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
`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,
                                  longitude = c(43.158),
                                  latitude = c(36.349),
                                  resolution = 0.5,
                                  grayscale = FALSE,
                                  mile = FALSE,
                                  preprocess = FALSE)

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

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

---

vis_hfr *Visualize hyperframes*

Description

‘vis_hfr()‘ takes a hyperframe and visualizes columns that users specify. ‘vis_hfr()‘ is used mainly for the visualization of the output of [get_hfr()] function.

Usage

```
vis_hfr(hfr, subtype_column, time_column = "time", range, combined = TRUE)
```

Arguments

hfr	hyperframe
subtype_column	The name/s of a column of interest. To specify multiple columns, users should list column names as a character vector.
time_column	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")‘)

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

Value

ggplot object that displays ppp objects of interest

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
dat_hfr <- get_hfr(data = dat,
                   subtype_column = "type",
                   window = iraq_window,
                   time_column = "time",
                   time_range = c(1, 2),
                   coordinates = c("longitude", "latitude"),
                   combined = FALSE)

# Visualization
vis_hfr(hfr = dat_hfr,
        subtype_column = c("treat"),
        time_column = "time",
        range = c(1:2),
        combined = TRUE)
```

---

vis_obs_dens *Visualize observed densities*

Description

‘vis_obs_dens()‘ visualizes actual and predicted counts generated by [get_obs_dens()].

Usage

```
vis_obs_dens(
  actual_data,
  dens_1,
  dens_2 = NA,
  dens_3 = NA,
  color_actual = "darkgrey",
  color_dens_1 = "#D55E00",
  color_dens_2 = "#0072B2",
```

```
  color_dens_3 = "#009E73",
  time_unit
)
```

Arguments

actual_data	actual data in the form of 'hyperframe$column'.
dens_1	im object that represents densities
dens_2	density 2 (if any). By default, 'NA'.
dens_3	density 3 (if any). By default, 'NA'.
color_actual	color for actual data. By default, '"darkgrey"'.
color_dens_1	color for actual data. By default, '"#D55E00"'.
color_dens_2	color for actual data. By default, '"#0072B2"'.
color_dens_3	color for actual data. By default, '"#009E73"'.
time_unit	x-axis label of the output

Value

list of dataframe and two ggplot objects for comparison of densities and residuals

Index