Package 'geocausal'

June 26, 2026

Type Package

Title Causal Inference with Spatio-Temporal Data

Version 0.4.2

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> and Mukaigawara, et al. (2024) <doi:10.31219/osf.io/5kc6f>.

License MIT + file LICENSE

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

Encoding UTF-8

LazyData true

Suggests elevatr, gridExtra, knitr, readr, gridGraphics

Imports crsuggest, ggthemes, data.table, dplyr, furrr, ggplot2, ggpubr, mclust, progressr, purrr, Rglpk, sf, spatstat.explore, spatstat.geom, spatstat.model, spatstat.univar, spatstat.random, terra, tibble, tidyr, tidyselect, tidyterra

Depends R (>= 3.5.0)

Config/roxygen2/version 8.0.0

NeedsCompilation no

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

Repository CRAN

Date/Publication 2026-06-26 09:50:02 UTC

Contents

`airstrikes` *airstrikes*

Description

A dataset of airstrikes in Iraq (February 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_2006 *airstrikes_2006*

Description

A dataset of airstrikes in Iraq in 2006 (2006/1/1-2006/9/24)

Usage

```
airstrikes_2006
```

Format

A tibble with 2101 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_2006
```

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

Details

‘conv_owin_into_sf()‘ converts the window to a polygonal object, extracts its boundary coordinates (closing the ring if necessary), and returns the same boundary in several representations: an ‘sf‘ ‘POLYGON‘, a coordinate data frame, an ‘sfc_POLYGON‘, an ‘sf‘ object, and a ‘SpatialPolygonsDataFrame‘. It is mostly used internally by other functions that need different spatial object classes.

Value

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

See Also

Other spatial utility functions: `get_dist_focus()`, `get_dist_line()`, `get_distexp()`, `get_elev()`, `get_window()`

`dx_outpred` *Perform out-of-sample prediction*

Description

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

Usage

```
dx_outpred(
  hfr,
  ratio,
  dep_var,
  indep_var,
  ndim = 128,
  resolution = NULL,
  window
)
```

Arguments

`hfr`	hyperframe
`ratio`	numeric. ratio between training and test sets
`dep_var`	dependent variables
`indep_var`	independent variables
`ndim`	the number of grids. By default, '128' (128 x 128).
`resolution`	the resolution in km per pixel. If specified, overrides 'ndim'. For example, 'resolution = 5' creates ~5km x 5km grid cells.
`window`	owin object

Details

'dx_outpred()' splits the hyperframe chronologically into a training set (the first 'ratio' fraction of rows) and an implied test set, fits a Poisson point process model to the training data with 'spatstat.model::mppm()', and predicts the conditional intensity over the full hyperframe. It returns the predicted intensity images together with the integrated (estimated) event counts and the per-period sums of log intensities, which can be compared against observed counts to assess out-of-sample fit. Training and test sets are assumed to share the same window.

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

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

[get_obs_dens()]

Other diagnostic functions: `dx_supthin()`

Examples

```
# Prepare data: airstrikes (treatment) in Iraq, 2006 (first 60 days)
dat <- airstrikes_2006[airstrikes_2006$type == "Airstrike", ]
dat$type <- "airstrike"
dat$time <- as.numeric(dat$date - min(dat$date) + 1)
dat <- dat[dat$time <= 60, ]
hfr <- get_hfr(data = dat, col = "type", window = iraq_window,
               time_col = "time", time_range = c(1, 60),
               coordinates = c("longitude", "latitude"), combine = FALSE)
hfr$dist_bag <- rep(list(get_dist_focus(window = iraq_window, lon = 44.366,
                                        lat = 33.315, ndim = 64)), nrow(hfr))

# Out-of-sample prediction with an 80/20 split
pred <- dx_outpred(hfr, ratio = 0.8, dep_var = "airstrike",
                   indep_var = "dist_bag", ndim = 64, window = iraq_window)
```

---

dx_supthin *Perform superthinning tests*

Description

‘dx_supthin()‘ performs superthinning tests to examine model validity.

Usage

```
dx_supthin(
  hfr,
  dep_var,
  indep_var,
  window,
  rescale = 1,
  max_r = 50,
  n_sample = 1000,
  nsim = 1000,
  unit = "km"
)
```

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)
window	owin object
rescale	conversion as needed (namely when the unit of distance of the owin object is in meters). By default = 1 (no conversion)
max_r	max distance in which the envelope tests are performed
n_sample	the number of points to sample. by default = 1000, if the number of points are smaller than this, no sampling is performed
nsim	the number of simulations to perform for the envelope tests
unit	distance units after conversion. By default "km"

Details

'dx_supthin()' assesses model validity through superthinning. It fits a Poisson point process model with 'spatstat.model::mppm()', then for each time period thins the observed points and superposes simulated points so that, under a correctly specified model, the combined pattern is approximately homogeneous Poisson at the constant rate 'cval'. Global envelope tests on the L- and K-functions ('spatstat.explore::envelope()') are computed on the pooled, optionally subsampled ('n_sample'), pattern using 'nsim' simulations out to 'max_r'. The returned 'supthin' data frame stacks the L- and K-function envelopes for plotting.

Value

A list of resulting dataframe ('result_data'), windows ('window_list'), data for distance quantiles, and a window object for the entire window

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

[get_obs_dens()]

Other diagnostic functions: dx_outpred()

Examples

```
# Prepare data: airstrikes (treatment) in Iraq, 2006 (first 60 days)
dat <- airstrikes_2006[airstrikes_2006$type == "Airstrike", ]
dat$type <- "airstrike"
dat$time <- as.numeric(dat$date - min(dat$date) + 1)
dat <- dat[dat$time <= 60, ]
hfr <- get_hfr(data = dat, col = "type", window = iraq_window,
               time_col = "time", time_range = c(1, 60),
               coordinates = c("longitude", "latitude"), combine = FALSE)
hfr$dist_bag <- rep(list(get_dist_focus(window = iraq_window, lon = 44.366,
                                        lat = 33.315, ndim = 64)), nrow(hfr))

# Superthinning test (use a larger nsim in real applications)
supthin <- dx_supthin(hfr, dep_var = "airstrike", indep_var = "dist_bag",
                      window = iraq_window, nsim = 19)
```

---

get_adaptive_baseline_dens

Generate adaptive intervention densities based on historical data

---

Description

'get_adaptive_baseline_dens()' takes a hyperframe for historical data and returns fitted densities for the current data. The output is used as counterfactural densities.

Usage

```
get_adaptive_baseline_dens(
  hfr_hist,
  hfr_current,
  dep_var,
  indep_var,
  ngrid = 100,
  window
)
```

Arguments

hfr_hist	hyperframe for historical data
hfr_current	hyperframe for current data
dep_var	The name of the dependent variable Since we need to obtain the counterfactural 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_adaptive_baseline_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 * 'actual_counts': the number of events (actual counts)

See Also

Other density estimation functions: `get_base_dens()`, `get_cf_dens()`, `get_cf_dens_adaptive()`, `get_cf_sum_log_intens()`, `get_obs_dens()`, `get_power_dens()`, `sim_cf_dens()`, `sim_power_dens()`

---

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).

Usage

```
get_base_dens(
  window,
  ndim = 128,
  resolution = NULL,
  out_data = NULL,
  out_coordinates = c("longitude", "latitude"),
  input_crs = 4326,
  unit_scale = 1000
)
```

Arguments

`window`	owin object
`ndim`	the number of dimensions of grid cells (ndim^2). By default, ndim = 128.
`resolution`	the resolution in km per pixel. If specified, overrides 'ndim'. For example, 'resolution = 5' creates ~5km x 5km grid cells.
`out_data`	dataframe
`out_coordinates`	vector of column names of longitudes and latitudes (in this order)
`input_crs`	the CRS of the input `coordinates`. Defaults to `4326`
`unit_scale`	parameter to convert meters to kilometers (WGS84 decimal degrees). The function will transform these to match the `window` projection

Details

'get_base_dens()' estimates a baseline (out-of-sample) density from point data supplied in 'out_data'. The points are projected to match the CRS attached to 'window', converted to a 'ppp' object, and smoothed by a kernel density estimate whose bandwidth is chosen by Scott's rule of thumb ('spatstat.explore::bw.scott()'). The result is normalized to integrate to one. The output grid is controlled either by 'ndim' or, if supplied, by 'resolution' (in km per pixel), which overrides 'ndim'. The window must carry a CRS attribute (e.g., created via 'get_window()' with a 'target_crs').

Value

an im object of baseline density

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

[get_cf_dens()]

Other density estimation functions: get_adaptive_baseline_dens(), get_cf_dens(), get_cf_dens_adaptive(), get_cf_sum_log_intens(), get_obs_dens(), get_power_dens(), sim_cf_dens(), sim_power_dens()

Examples

```
# Baseline density from out-of-sample (2007--2008) insurgency data
base <- get_base_dens(window = iraq_window, ndim = 64,
                      out_data = insurgencies,
                      out_coordinates = c("longitude", "latitude"))
```

---

get_cate *Generate a Hajek estimator for heterogeneity analysis*

Description

A function that returns a Hajek estimator of CATE for a spatial or spatio-temporal effect modifier

Usage

```
get_cate(
  obs,
  cf1,
  cf2,
  treat,
  pixel_count_out,
  lag,
  trunc_level = 0.95,
  time_after = TRUE,
  entire_window = NULL,
  em = NULL,
  E_mat = NULL,
  nbase = 6,
  spline_type = "ns",
  intercept = TRUE,
  eval_values = NULL,
  eval_mat = NULL,
  test_beta = NULL,
  save_weights = TRUE,
  ...
)
```

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$column'.
pixel_count_out	column of a hyperframe that summarizes the number of outcome events in each pixel
lag	integer that specifies lags to calculate causal estimates.
trunc_level	the level of truncation for the weights (0-1).
time_after	whether to include one unit time difference between treatment and outcome. By default = TRUE
entire_window	owin object (the entire region of interest)
em	treat column of a hyperframe that summarizes the effect modifier data. In the form of 'hyperframe$column'. It can be NULL if E_mat is provided.
E_mat	optional covariance matrix (excluding the intercept) for the effect modifier. If provided, then the regression model will be based on this matrix. If 'intercept = TRUE', then a column of 1 will be add to 'E_mat'.
nbase	number of bases for splines
spline_type	type of splines. Either '"ns"' or '"bs"'.
intercept	whether to include intercept in the regression model. Default is TRUE.
eval_values	a vector of values of the effect modifier for which CATE will be evaluated. Default is a 'seq(a,b,length.out=20)' where 'a' and 'b' are minimum and maximum values of the effect modifier.
eval_mat	evaluated spline basis (excluding the intercept) matrix at 'eval_values'. If 'intercept = TRUE', then a column of 1 will be add to 'eval_mat'.
test_beta	a vector of integers contain the indices of the coefficients that are included in the hypothesis test. By default, the null hypothesis is that all coefficient (except the intercept is 0). See details below
save_weights	whether to save weights. Default is 'TRUE'
...	arguments passed onto the function

Details

'E_mat' should be a matrix or array of dimensions n by m where n is the product of image dimensions and number of time period, and m is 'nbase'-'intercept'. If you want to construct your own covariate matrix 'E_mat', you should use 'get_em_vec()' to convert the effect modifer(usually a column of a hyperframe) to a vector, and then construct the splines basis based on the vector. The covariate matrix'E_mat' should not the column for intercept. The function 'get_cate()' will conduct a hypothesis testing on whether all the selected coefficients are 0. 'test_beta' is a vector of positive integers specifying the indices of the chosen beta. The coefficients (except the intercept) are indexed by '1,2,...,nbase-intercept'. By default, it test whether all the coefficients(except the intercept) are 0, and this is testing the the heterogeneity effect of the effect modifier.

Value

list of the following: ‘est_beta‘: estimated regression coefficient ‘V_beta‘: estimated asymptotic covariance matrix of regression coefficient (normalized by total time periods) ‘chisq_stat‘: observed chi-square statistics for the hypothesis test ‘p.value‘: observed chi-square statistics for the hypothesis test ‘specification‘: information about the specification of the spline basis and the values on which the CATE is estimated ‘est_eval‘: estimated CATE evaluted at chosen values ‘V_eval‘: estimated asymptotic covariance matrix of the estimated CATE values (normalized by total time periods) ‘mean_effect‘: Mean of the pseudo pixel effect ‘total_effect‘: Mean of the pseudo effect for the window ‘entire_window‘. It is equal to mean effect times the total number of pixels inside the chosen window

References

Zhou, L., Imai, K., Lyall, J. and Papadogeorgou, G. (2024). Estimating heterogeneous treatment effects for spatio-temporal causal inference: how economic assistance moderates the effects of airstrikes on insurgent violence. arXiv preprint. doi:10.48550/arXiv.2412.15128

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

[get_cate_sens()], [get_em_vec()], [pixel_count_ppp()]

Other causal effect estimation functions: `get_est()`, `get_estimates()`, `get_var_bound()`, `get_weighted_surf()`

Examples

```
# Prepare data: airstrikes (treatment) and insurgencies (outcome), Iraq 2006
dat <- rbind(airstrikes_2006[airstrikes_2006$type == "Airstrike", ],
             insurgencies_2006)
dat$type <- ifelse(dat$type == "Airstrike", "airstrike", "insurgency")
dat$time <- as.numeric(dat$date - min(dat$date) + 1)
dat <- dat[dat$time <= 60, ]
hfr <- get_hfr(data = dat, col = "type", window = iraq_window,
               time_col = "time", time_range = c(1, 60),
               coordinates = c("longitude", "latitude"), combine = FALSE)
hfr$dist_bag <- rep(list(get_dist_focus(window = iraq_window, lon = 44.366,
                                        lat = 33.315, ndim = 64)), nrow(hfr))

# Observed density (propensity score) and counterfactual densities
obs <- get_obs_dens(hfr, dep_var = "airstrike", indep_var = "dist_bag",
                    ndim = 64, window = iraq_window)
base <- get_base_dens(window = iraq_window, ndim = 64,
                      out_data = insurgencies,
                      out_coordinates = c("longitude", "latitude"))
cf1 <- get_cf_dens(expected_number = 2, base_dens = base, window = iraq_window)
cf2 <- get_cf_dens(expected_number = 4, base_dens = base, window = iraq_window)

# Outcome counts by pixel, and CATE over the distance-from-Baghdad modifier
hfr$pc_insurgency <- pixel_count_ppp(hfr$insurgency, ndim = 64)
cate <- get_cate(obs = obs, cf1 = cf1, cf2 = cf2,
```

```
treat = hfr$airstrike, pixel_count_out = hfr$pc_insurgency,
lag = 1, entire_window = iraq_window,
em = hfr$dist_bag, nbase = 4)
```

`get_cate_sens` *Sensitivity analysis for conditional average treatment effects*

Description

'get_cate_sens()' evaluates how robust conditional average treatment effect (CATE) estimates are to possible unmeasured confounding. The function projects each period-specific weighted surface onto the same effect-modifier basis used by 'get_cate()', and then checks whether each non-intercept basis coefficient can be shifted to zero under each sensitivity parameter 'gamma'.

Usage

```
get_cate_sens(
  obs,
  cf1,
  cf2,
  treat,
  pixel_count_out,
  lag,
  trunc_level = 0.95,
  time_after = TRUE,
  entire_window = NULL,
  em = NULL,
  E_mat = NULL,
  nbase = 6,
  spline_type = "ns",
  intercept = TRUE,
  eval_values = NULL,
  eval_mat = NULL,
  gamma_vals = seq(1.05, 1.2, by = 0.05),
  save_weights = TRUE,
  tol_slack = 1e-06,
  grid_init = 15,
  max_refine = 5,
  ...
)
```

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$column'.
pixel_count_out	column of a hyperframe that summarizes the number of outcome events in each pixel
lag	integer that specifies lags to calculate causal estimates.
trunc_level	the level of truncation for the weights (0-1).
time_after	whether to include one unit time difference between treatment and outcome. By default = TRUE
entire_window	owin object (the entire region of interest)
em	column of a hyperframe that summarizes the effect modifier data. In the form of 'hyperframe$column'. It can be NULL if 'E_mat' is provided.
E_mat	optional covariate matrix (excluding the intercept) for the effect modifier.
nbase	number of bases for splines
spline_type	type of splines. Either '"ns"' or '"bs"'.
intercept	whether to include intercept in the regression model. Default is TRUE.
eval_values	currently unused; reserved for future CATE-value sensitivity output.
eval_mat	currently unused; reserved for future CATE-value sensitivity output.
gamma_vals	numeric vector of sensitivity parameters (>= 1) at which to check zero-attainability. By default, 'seq(1.05, 1.2, by = 0.05)'.
save_weights	whether to save weights. Default is 'TRUE'
tol_slack	numeric tolerance used to determine whether zero is attainable. Default is '1e-6'.
grid_init	number of grid points used in each adaptive lambda search step. Default is 15.
max_refine	maximum number of adaptive lambda refinement steps. Default is 5.
...	arguments passed onto the spline basis function

Details

Unlike 'get_sens()', which returns lower and upper bounds for an average treatment effect, 'get_cate_sens()' checks zero-attainability for the projected coefficients. For each 'gamma', 'zero_attainable' indicates whether the corresponding non-intercept basis coefficient is no longer robust under the current allowance for unmeasured confounding. The 'robust_gamma' value summarizes the largest sensitivity level at which all included coefficients remain robust. The underlying linear programs are solved with 'Rglpk::Rglpk_solve_LP()'.

'gamma = 1' corresponds to no unmeasured confounding. Larger values allow more deviation from the observed treatment process.

Value

list of the following: 'beta': a data frame with 'basis', 'gamma', and 'zero_attainable' for the non-intercept basis coefficients. 'robust_gamma': the smallest coefficient-specific robust gamma across all non-intercept basis coefficients. 'specification': information about the spline basis, evaluated values, and sensitivity parameters.

References

Zhou, L., Imai, K., Lyall, J. and Papadogeorgou, G. (2024). Estimating heterogeneous treatment effects for spatio-temporal causal inference: how economic assistance moderates the effects of airstrikes on insurgent violence. arXiv preprint. doi:10.48550/arXiv.2412.15128

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

Other sensitivity analysis functions: `get_sens()`

`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

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

[get_obs_dens()], [get_est()]

Other density estimation functions: `get_adaptive_baseline_dens()`, `get_base_dens()`, `get_cf_dens_adaptive()`, `get_cf_sum_log_intens()`, `get_obs_dens()`, `get_power_dens()`, `sim_cf_dens()`, `sim_power_dens()`

Examples

```
# Baseline density from out-of-sample (2007--2008) insurgency data
base <- get_base_dens(window = iraq_window, ndim = 64,
                      out_data = insurgencies,
                      out_coordinates = c("longitude", "latitude"))

# Counterfactual scenario: two airstrikes per day on average,
# distributed proportionally to the baseline density
cf <- get_cf_dens(expected_number = 2, base_dens = base,
                  window = iraq_window)
```

`get_cf_dens_adaptive` *Get counterfactual densities*

Description

'get_cf_dens_adaptive' takes the scale_factor, baseline density to construct counterfactural intensities .

Usage

```
get_cf_dens_adaptive(baseline_den, scale_factor)
```

Arguments

`baseline_den`	baseline density (obs object obtained from 'get_adaptive_baseline_dens')
`scale_factor`	a positive number that scales the baseline intensity

Value

a list of the following: * 'intens_grid_cells': im object of counterfactural intensities 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

See Also

Other density estimation functions: `get_adaptive_baseline_dens()`, `get_base_dens()`, `get_cf_dens()`, `get_cf_sum_log_intens()`, `get_obs_dens()`, `get_power_dens()`, `sim_cf_dens()`, `sim_power_dens()`

`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 or a list of im objects)
`treatment_data`	In the form of hyperframe$column

Details

'get_cf_sum_log_intens()' evaluates a counterfactual density at the observed treatment locations for each time period using bilinear interpolation ('spatstat.geom::interp.im()'), and returns the sum of the log intensities per period. 'cf_dens' may be a single 'im' object reused across all periods or a list of 'im' objects with one per period. Points falling on the boundary, where the intensity cannot be interpolated, are dropped; a period with no usable points contributes a sum of zero.

Value

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

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

Other density estimation functions: `get_adaptive_baseline_dens()`, `get_base_dens()`, `get_cf_dens()`, `get_cf_dens_adaptive()`, `get_obs_dens()`, `get_power_dens()`, `sim_cf_dens()`, `sim_power_dens()`

`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",
  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"'
`use_raw`	logical. 'use_raw' specifies whether to use the raw value of expectations or percentiles. By default, 'FALSE'.

Details

'get_distexp()' builds a sequence of nested sub-windows defined by percentiles of the distance map 'dist_map', then integrates each simulated counterfactual density over those sub-windows to obtain the expected number of treatment events within each distance band. By default the expectations are standardized by the total integral of the first counterfactual density; set 'use_raw = TRUE' to return raw expected counts instead. The output is a 'distlist' object suitable for plotting.

Value

A list of resulting dataframe ('result_data'), windows ('window_list'), data for distance quantiles, and a window object for the entire window

See Also

Other spatial utility functions: `conv_owin_into_sf()`, `get_dist_focus()`, `get_dist_line()`, `get_elev()`, `get_window()`

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 = NULL,
  ndim = NULL,
  mile = FALSE,
  preprocess = FALSE,
  input_crs = 4326,
  unit_scale = 1000
)
```

Arguments

window	owin object
lon	vector of longitudes
lat	vector of latitudes
resolution	resolution of raster (distance map) (in km; by default, 1). Ignored if npixel is set.
ndim	number of pixels for both dimensions (e.g., 256 for 256x256). If set, resolution is ignored.
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.
input_crs	the CRS of the focus points (defaults to 4326). These points are internally projected to match the window CRS to ensure isotropic distance calculations.
unit_scale	set to the same value as the parameter in 'get_window()' function. This parameter converts the coordinate values so that they alingn with the unit (km) of the owin object

Details

This function performs its computation using the 'future' framework (via 'furrr'); to run it in parallel, set 'future::plan(future::multisession)' before calling this function. The results are identical regardless of the plan.

Value

an im object

See Also

Other spatial utility functions: conv_owin_into_sf(), get_dist_line(), get_distexp(), get_elev(), get_window()

`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 = NULL,
  line_data = NULL,
  mile = FALSE,
  resolution = NULL,
  ndim = NULL,
  preprocess = TRUE,
  unit_scale = 1000
)
```

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 (distance map) (in km; by default, 1). Ignored if ndim is set.
ndim	number of pixels for both dimensions (e.g., 256 for 256x256). If set, resolution is ignored.
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.
unit_scale	set to the same value as the parameter in 'get_window()' function. This parameter converts the coordinate values so that they alingn with the unit (km) of the owin object

Details

The function ensures spatial integrity by automatically projecting the `line_data` or shapefile to match the Coordinate Reference System (CRS) of the `window`.

This function performs its computation using the ‘future‘ framework (via ‘furrr‘); to run it in parallel, set ‘future::plan(future::multisession)‘ before calling this function. The results are identical regardless of the plan.

Value

an im object

See Also

Other spatial utility functions: `conv_owin_into_sf()`, `get_dist_focus()`, `get_distexp()`, `get_elev()`, `get_window()`

`get_elev` *Get elevation data*

Description

‘get_elev()‘ 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.

Details

‘get_elev()‘ reads a shapefile and downloads elevation data clipped to that region via ‘elevatr::get_elev_raster()‘ (the resolution argument ‘z‘ must be passed through ‘...‘). The raster is then converted into a ‘spatstat.geom::im‘ object, flipping the matrix vertically so that the raster (top-left origin) aligns with the ‘spatstat‘ (bottom-left origin) convention.

Value

an im object (unit: meters)

See Also

Other spatial utility functions: `conv_owin_into_sf()`, `get_dist_focus()`, `get_dist_line()`, `get_distexp()`, `get_window()`

`get_em_vec` *convert a list of im objects to a vector*

Description

'get_em_vec()' get the vector form of a column of a hyperframe that summarizes the effect modifier data in heterogeneity analysis

Usage

```
get_em_vec(
  em,
  outcome_pixel_count = NULL,
  time_after = TRUE,
  entire_window = NULL,
  lag
)
```

Arguments

`em`	column of a hyperframe that summarizes effect modifier data. In the form of 'hyperframe$column'.
`outcome_pixel_count`	A list of integer-valued pixel images giving outcome event counts per spatial pixel, typically obtained from 'pixel_count_ppp()'.
`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). If given, then the values outside the region will be set to 'NA'.
`lag`	integer that specifies lags to calculate causal estimates

Details

The function 'get_em_vec()' get the vector form of the effect modifier in the heterogeneity analysis. It is useful if you want to construct the variance matrix 'E_mat' that is passed to the function 'get_cate()'

Value

a numeric vector giving the effect modifier values, flattened across pixels and time periods, with values outside 'entire_window' (if supplied) set to 'NA'.

References

Zhou, L., Imai, K., Lyall, J. and Papadogeorgou, G. (2024). Estimating heterogeneous treatment effects for spatio-temporal causal inference: how economic assistance moderates the effects of airstrikes on insurgent violence. arXiv preprint. doi:10.48550/arXiv.2412.15128

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

Other data preparation functions: `get_hfr()`, `get_hist()`, `imls_to_arr()`, `pixel_count_ppp()`, `smooth_ppp()`

`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,
  save_weights = TRUE
)
```

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)
save_weights	whether to save weights

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

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

[get_obs_dens()], [get_cf_dens()], [get_sens()]

Other causal effect estimation functions: get_cate(), get_estimates(), get_var_bound(), get_weighted_surf()

Examples

```
# Prepare data: airstrikes (treatment) and insurgencies (outcome), Iraq 2006
dat <- rbind(airstrikes_2006[airstrikes_2006$type == "Airstrike", ],
             insurgencies_2006)
dat$type <- ifelse(dat$type == "Airstrike", "airstrike", "insurgency")
dat$time <- as.numeric(dat$date - min(dat$date) + 1)
dat <- dat[dat$time <= 60, ]
hfr <- get_hfr(data = dat, col = "type", window = iraq_window,
               time_col = "time", time_range = c(1, 60),
               coordinates = c("longitude", "latitude"), combine = FALSE)
dist_baghdad <- get_dist_focus(window = iraq_window, lon = 44.366,
                               lat = 33.315, ndim = 64)
hfr$dist_bag <- rep(list(dist_baghdad), nrow(hfr))

# Observed density (propensity score) and counterfactual densities
obs <- get_obs_dens(hfr, dep_var = "airstrike", indep_var = "dist_bag",
                    ndim = 64, window = iraq_window)
base <- get_base_dens(window = iraq_window, ndim = 64,
                      out_data = insurgencies,
                      out_coordinates = c("longitude", "latitude"))
cf1 <- get_cf_dens(expected_number = 2, base_dens = base, window = iraq_window)
cf2 <- get_cf_dens(expected_number = 4, base_dens = base, window = iraq_window)

# Smoothed outcomes
hfr$sm_insurgency <- smooth_ppp(hfr$insurgency, method = "abramson", ndim = 64)

# Causal estimates comparing the two scenarios
est <- get_est(obs = obs, cf1 = cf1, cf2 = cf2,
               treat = hfr$airstrike, sm_out = hfr$sm_insurgency,
               lag = 1, entire_window = iraq_window,
               use_dist = TRUE, dist_map = dist_baghdad,
               dist = c(100, 200, 300), trunc_level = 0.95)
```

---

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

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

Other causal effect estimation functions: `get_cate()`, `get_est()`, `get_var_bound()`, `get_weighted_surf()`

`get_hfr` *Create a hyperframe*

Description

'get_hfr()' takes a dataframe with time and location variables and generates a hyperframe with point patterns.

Usage

```
get_hfr(
  data,
  col,
  window,
  time_col,
  time_range,
  coordinates = c("longitude", "latitude"),
  combine = TRUE,
  input_crs = 4326,
  unit_scale = 1000,
  check_duplicates = FALSE
)
```

Arguments

`data`	dataframe. The dataframe must have time and location variables.
`col`	the name of the column for types of events of interest
`window`	owin object (for more information, refer to 'spatstat.geom::owin()'). Must be a projected window (e.g., created via `get_window` with a `target_crs` specified). This function automatically detects the projection from the window and projects the event data accordingly.
`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.
`input_crs`	the CRS of the input `coordinates`. Defaults to `4326` (WGS84 decimal degrees). The function will transform these to match the `window` projection
`unit_scale`	parameter to convert meters to kilometers

check_duplicates
: logical. ‘check_duplicates‘ specifies whether to run ‘spatstat‘’s duplicate-point check when constructing the point pattern of each time period. By default, ‘FALSE‘, so the check (and the warning "data contain duplicated points") is skipped, because coincident events are legitimate in event data. Set to ‘TRUE‘ to re-enable the check as a diagnostic for possible data-recording issues; the constructed point patterns are identical either way.

Details

Event data often contain multiple events at identical coordinates (e.g., several events recorded at the same location within one time period). Such duplicated points are legitimate observations and are retained as they are; each point contributes separately to subsequent intensity estimation. For this reason, ‘get_hfr()‘ skips ‘spatstat‘’s duplicate-point check by default (‘check_duplicates = FALSE‘), so the warning "data contain duplicated points" is not issued. Set ‘check_duplicates = TRUE‘ to re-enable the check as a diagnostic for potential data-recording issues. Downstream functions that pool points across time periods (e.g., ‘get_hist()‘, ‘smooth_ppp()‘, ‘get_base_dens()‘, ‘dx_supthin()‘) always skip the check, since duplicates are unavoidable there.

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‘.

See Also

[get_window()], [smooth_ppp()]

Other data preparation functions: `get_em_vec()`, `get_hist()`, `imls_to_arr()`, `pixel_count_ppp()`, `smooth_ppp()`

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

See Also

Other data preparation functions: `get_em_vec()`, `get_hfr()`, `imls_to_arr()`, `pixel_count_ppp()`, `smooth_ppp()`

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_linear_prog` *Solve linear program for sensitivity bounds at a given gamma*

Description

Solves the pair of linear programs that yield the upper and lower bounds of the Hajek-weighted estimand under a given sensitivity parameter ‘gamma‘. Used internally by ‘get_sens()‘.

Usage

```
get_linear_prog(wto, this_gamma)
```

Arguments

`wto`	list whose first element is the numerator (daily Hajek-weighted outcomes) and whose second element is the denominator (daily Hajek weights for the window of interest), as returned by ‘sens_weighted_surf()‘.
`this_gamma`	sensitivity parameter (>= 1). ‘gamma = 1‘ corresponds to no unmeasured confounding.

Details

‘get_linear_prog()‘ is an internal function to ‘get_sens()‘. It relies on ‘Rglpk::Rglpk_solve_LP()‘ to solve the transformed linear program.

Value

a tibble with columns ‘high‘ and ‘low‘ giving the upper and lower bounds of the estimand at ‘this_gamma‘.

`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, ndim = 128, resolution = NULL, 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)
ndim	the number of grid cells that is used to generate observed densities. By default = 128 (128 x 128). Notice that as you increase 'ndim', the process gets computationally demanding.
resolution	the resolution in km per pixel. If specified, overrides 'ndim'. For example, 'resolution = 5' creates ~5km x 5km grid cells.
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 * 'deviance': deviance * 'null_deviance': null deviance * 'dispersion': dispersion parameter * 'res_df': average residuals as a dataframe * '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 * 'actual_counts': the number of events (actual counts) * 'window': window object used as an input

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

[get_cf_dens()], [get_est()]

Other density estimation functions: `get_adaptive_baseline_dens()`, `get_base_dens()`, `get_cf_dens()`, `get_cf_dens_adaptive()`, `get_cf_sum_log_intens()`, `get_power_dens()`, `sim_cf_dens()`, `sim_power_dens()`

Examples

```
# Prepare data: airstrikes (treatment) and insurgencies (outcome), Iraq 2006
dat <- rbind(airstrikes_2006[airstrikes_2006$type == "Airstrike", ],
             insurgencies_2006)
```

```
dat$type <- ifelse(dat$type == "Airstrike", "airstrike", "insurgency")
dat$time <- as.numeric(dat$date - min(dat$date) + 1)
dat <- dat[dat$time <= 60, ]
hfr <- get_hfr(data = dat, col = "type", window = iraq_window,
               time_col = "time", time_range = c(1, 60),
               coordinates = c("longitude", "latitude"), combine = FALSE)

# Covariate surface: distance from Baghdad
hfr$dist_bag <- rep(list(get_dist_focus(window = iraq_window, lon = 44.366,
                                        lat = 33.315, ndim = 64)), nrow(hfr))

# Observed density of the treatment (propensity score)
obs <- get_obs_dens(hfr, dep_var = "airstrike", indep_var = "dist_bag",
                    ndim = 64, 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

Details

'get_power_dens()' raises each target density to the power given by the corresponding entry of 'priorities', multiplies the results together, and normalizes the product to integrate to one over 'window'. The resulting power density is used to shift the locations of counterfactual events relative to the baseline density.

Value

an im object of power densities

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

[get_cf_dens()], [sim_power_dens()]

Other density estimation functions: `get_adaptive_baseline_dens()`, `get_base_dens()`, `get_cf_dens()`, `get_cf_dens_adaptive()`, `get_cf_sum_log_intens()`, `get_obs_dens()`, `sim_cf_dens()`, `sim_power_dens()`

Examples

```
# A target density: proportional to the distance from Baghdad
dist_baghdad <- get_dist_focus(window = iraq_window,
                               lon = 44.366, lat = 33.315, ndim = 64)
power <- get_power_dens(target_dens = list(dist_baghdad / sum(dist_baghdad)),
                        priorities = 1, window = iraq_window)
```

---

get_sens *Sensitivity analysis for counterfactual contrasts*

Description

'get_sens()' computes bounds on the causal contrast between two counterfactual densities across a grid of sensitivity parameters 'gamma'. At each value of 'gamma', linear programming is used to obtain worst-case upper and lower bounds of the Hajek-weighted estimand for each scenario; the returned bounds describe how robust the causal contrast is to possible violations of the no-unmeasured-confounding assumption.

Usage

```
get_sens(
  obs,
  cf1,
  cf2,
  treat,
  sm_out,
  lag,
  entire_window,
  window,
  gamma_vals = seq(1, 1.2, by = 0.01),
  time_after = TRUE,
  trunc_level = NA,
```

```
  tol_slack = 1e-06,
  grid_init = 15,
  max_refine = 5
)
```

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
lag	integer that specifies lags to calculate causal estimates
entire_window	owin object (the entire region of interest)
window	owin object (the sub-window over which the contrast is evaluated)
gamma_vals	numeric vector of sensitivity parameters (>= 1) at which to compute bounds. By default, 'seq(1, 1.2, by = 0.01)'.
time_after	whether to include one unit time difference between treatment and outcome. By default = TRUE
trunc_level	the level of truncation for the weights (0-1)
tol_slack	numeric tolerance used to determine whether zero is attainable. Default is '1e-6'.
grid_init	number of grid points used in each adaptive lambda search step. Default is 15.
max_refine	maximum number of adaptive lambda refinement steps. Default is 5.

Details

'gamma = 1' corresponds to no unmeasured confounding and recovers point estimates analogous to those of 'get_est()'. As 'gamma' increases, the bounds widen, reflecting greater allowance for unobserved confounding. The 'lb' and 'ub' columns retain the conservative bounds from the original implementation. The 'zero_attainable' column indicates whether the result is no longer robust under the current value of 'gamma', or equivalently under the current allowance for unmeasured confounding. The underlying linear programs are solved with 'Rglpk::Rglpk_solve_LP()'.

This function performs its computation using the 'future' framework (via 'furrr'); to run it in parallel, set 'future::plan(future::multisession)' before calling this function. The results are identical regardless of the plan.

Value

a tibble with columns: * 'gamma': the sensitivity parameter * 'lb': lower bound on the causal contrast at 'gamma' * 'ub': upper bound on the causal contrast at 'gamma' * 'zero_attainable': whether zero is attainable at 'gamma' * 'robust_gamma': the largest gamma at which zero is not attainable

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

Other sensitivity analysis functions: `get_cate_sens()`

`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

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

Other causal effect estimation functions: `get_cate()`, `get_est()`, `get_estimates()`, `get_weighted_surf()`

`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,
  cate = 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.
`cate`	whether to perform the heterogeneity 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).

This function performs its computation using the 'future' framework (via 'furrr'); to run it in parallel, set 'future::plan(future::multisession)' before calling this function. The results are identical regardless of the plan.

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

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

Other causal effect estimation functions: `get_cate()`, `get_est()`, `get_estimates()`, `get_var_bound()`

`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, target_crs = NULL, unit_scale = 1000)
```

Arguments

`load_path`	path to the shp file
`target_crs`	target CRS (if users want to specify it; as an EPSG code)
`unit_scale`	parameter to convert meters to kilometers (by default = 1000 so that the unit is set to km)

Details

'get_window()' reads a shapefile and converts its boundary into a 'spatstat.geom::owin' object. When 'target_crs' is not supplied and the data are in longitude/latitude, a Cartesian CRS is suggested automatically via 'crsuggest::suggest_crs()' and the geometry is projected accordingly. Polygon vertices are reordered to anti-clockwise as required by 'owin', the window is rescaled by 'unit_scale' (so the default unit becomes kilometers), and the CRS is stored as an attribute for downstream functions.

Value

owin object

See Also

[get_hfr()]

Other spatial utility functions: conv_owin_into_sf(), get_dist_focus(), get_dist_line(), get_distexp(), get_elev()

Examples

```
# An example window from the shapefile shipped with the sf package
win <- get_window(load_path = system.file("shape/nc.shp", package = "sf"),
                  target_crs = 32119)
```

---

imls_to_arr *convert a list of im objects to a three-dimensional array*

Description

'imls_to_arr()' convert a list of im object to a 3D array

Usage

```
imls_to_arr(imls, start = 1, end = NULL, entire_window = NULL, ...)
```

Arguments

imls	a list of im objects (imlist)
start	the index of the first im to be converted. Default is 1.
end	the index of the last im to be converted. If not provided, then it will be set to the length of the list.
entire_window	a owin object. If given, then the values outside the region will be set to 'NA'.
...	Additional arguments passed to spatstat conversion functions.

Details

'imls_to_arr()' is a internal function for 'imls_to_vec()'. By default, it returns a three-dimensional array of dimension n by m by$l$ where n and m are the dimensions of the im objects, and l is the length of the list. All the im objects in the list need to have the same dimensions.

Value

a three-dimensional array of dimension n by m by l, where n and m are the pixel dimensions of the 'im' objects and l is the number of selected list elements. When 'entire_window' is supplied, cells outside that window are set to 'NA'.

See Also

Other data preparation functions: `get_em_vec()`, `get_hfr()`, `get_hist()`, `pixel_count_ppp()`, `smooth_ppp()`

`insurgencies` *insurgencies*

Description

A dataset of insurgencies data in Iraq (February to July 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
```

insurgencies_2006 *insurgencies_2006*

Description

A dataset of insurgencies data in Iraq in 2006 (2006/1/1-2006/9/24)

Usage

```
insurgencies_2006
```

Format

A tibble with 37701 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_2006
```

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
```

`pixel_count_ppp` *Get number of events in a pixel*

Description

'pixel_count_ppp()' takes a column of hyperframes (ppp objects) and gets the number of events in each pixel.

Usage

```
pixel_count_ppp(
  data,
  resolution = NULL,
  ndim = NULL,
  W = NULL,
  weights = NULL,
  DivideByPixelArea = FALSE,
  ...
)
```

Arguments

`data`	the name of a hyperframe and column of interest.
`resolution`	resolution of raster (distance map) (in km)
`ndim`	the number of dimensions of grid cells (ndim^2). Users need to set either resolution or ndim.
`W`	Optional window mask (object of class '"owin"') determining the pixel raster. 'data' should be in the form of '"hyperframe$column"'.
`weights`	Optional vector of weights associated with the points.
`DivideByPixelArea`	Logical value determining whether the resulting pixel values should be devided by the pixel area. Default value is 'False'.
`...`	parameters passed on to the function.

Details

This function performs its computation using the 'future' framework (via 'furrr'); to run it in parallel, set 'future::plan(future::multisession)' before calling this function. The results are identical regardless of the plan.

Value

im objects

See Also

Other data preparation functions: `get_em_vec()`, `get_hfr()`, `get_hist()`, `imls_to_arr()`, `smooth_ppp()`

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)

# Get the number of events for each pixel
pixel_count_ppp(data = dat_hfr$all_combined)
```

---

plot.cate *Plot estimated CATE*

Description

Plot method for objects of class ‘cate‘, returned by [get_cate()]. Visualizes the estimated conditional average treatment effects (CATE) across the chosen evaluation values, or the estimated regression coefficients, with confidence intervals.

Usage

```
## S3 method for class 'cate'
plot(
  x,
  ...,
  result = "cate",
  type = "l",
  scale = 1,
  xrange = NULL,
  main = "",
  xlab = "",
  ylim = NULL
)
```

Arguments

x	an object of class ‘cate‘, typically the output of [get_cate()].
...	additional arguments. Currently ignored.
result	specify which values will be used for plot. Default is "cate" - If ‘result‘ is "cate", then estimated cate values will be used - If ‘result‘ is "beta", then the estimated regression coefficients will be used

type	The type of plot to draw. This argument will be ignored if ‘result‘ = "beta". Default is "l". - If ‘type‘ is "p", points with error bars will be drawn. - If ‘type‘ is "l", lines with shaded region will be drawn. - If ‘type‘ is a vector of strings, each element specifies the type for the corresponding ‘eval_values‘ value.
scale	a positive number specifying the scale by which the estimates will be scaled. If provided, the estimates will be scaled by this value. Default is NULL, which means no scaling is applied.
xrange	an optional vector of two values the range of x shown.
main	title
xlab	label of x-axis
ylim	an optional vector of two values specifying the limits of y

Value

A ‘ggplot‘ object showing the estimated CATE (or regression coefficients) with confidence intervals.

See Also

[get_cate()]

`plot.cflist` *Plot simulated counterfactual densities*

Description

Plot method for objects of class ‘cflist‘, returned by [sim_cf_dens()]. Takes the simulated counterfactual densities and their powers and returns a faceted counterfactual density image over the range of parameters.

Usage

```
## S3 method for class 'cflist'
plot(
  x,
  ...,
  color = c("white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", "#771F59FF"),
  grayscale = FALSE
)
```

Arguments

x	an object of class ‘cflist‘, typically the output of [sim_cf_dens()].
...	additional arguments. Currently ignored.
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

A ‘ggplot‘ object (a faceted arrangement of counterfactual density images, one per power).

See Also

[sim_cf_dens()]

`plot.distlist` *Plot distance-based expectations*

Description

Plot method for objects of class ‘distlist‘, returned by [get_distexp()]. Visualizes the expected number (or proportion) of treatment events covered as a function of distance from the focus, and optionally the windows defined by distance quantiles.

Usage

```
## S3 method for class 'distlist'
plot(
  x,
  ...,
  dist_map_unit = "km",
  grayscale = FALSE,
  win_plot = FALSE,
  use_raw = FALSE
)
```

Arguments

`x`	an object of class ‘distlist‘, typically the output of [get_distexp()].
`...`	additional arguments. Currently ignored.
`dist_map_unit`	either ‘"km"‘ or ‘"mile"‘
`grayscale`	grayscale or not. By default, FALSE.
`win_plot`	whether to plot windows as well. By default, FALSE
`use_raw`	logical. ‘use_raw‘ specifies whether to use the raw value of expectations or percentiles. By default, ‘FALSE‘.

Value

A ‘ggplot‘ object. If ‘win_plot = TRUE‘, the arranged window plots; otherwise the expectation-versus-distance plot.

See Also

[get_distexp()]

plot.est *Plot estimates*

Description

Plot method for objects of class ‘est‘, returned by [get_est()]. Visualizes the estimated causal effects per time period (point estimates with 90 weighted counterfactual density surfaces.

Usage

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

Arguments

x	an object of class ‘est‘, typically the output of [get_est()].
...	additional arguments. Currently ignored.
surface	whether to produce the surface plot. By default, FALSE
lim	limits of the scale. By default, NA. To set limits manually, provide a vector of max and min

Value

If ‘surface = FALSE‘ (the default), a single ‘ggplot‘ object showing the causal effects per time period. If ‘surface = TRUE‘, a named list with two ‘ggplot‘ objects, ‘surface‘ (the difference in average weighted surfaces) and ‘expectation‘ (the causal effects per time period).

See Also

[get_est()]

plot.hyperframe *Plot a hyperframe*

Description

Plot method for objects of class ‘hyperframe‘ (as created by [get_hfr()]). Visualizes the selected point pattern (‘ppp‘) or pixel image (‘im‘) column(s) over the specified time period(s), arranging multiple time periods and/or columns by faceting or combining them.

Usage

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

Arguments

x	an object of class ‘hyperframe‘, typically the output of [get_hfr()].
...	additional arguments. Currently ignored.
col	the name/s of a column of interest.
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
main	title To specify multiple columns, users should list column names as a character vector.
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

Value

A ‘ggplot‘ object visualizing the selected column(s) and time period(s).

See Also

[get_hfr()]

plot.im *Plot a pixel image*

Description

Plot method for objects of class ‘im‘ (spatstat pixel images, such as the density images produced by the package’s density functions). Renders the image as a filled raster over its observation window, with optional value transformation and grayscale rendering.

Usage

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

Arguments

x	an object of class ‘im‘ (a spatstat pixel image), for example a density image produced by a ‘get_*_dens()‘ function.
...	additional arguments. Currently ignored.
main	title
scalename	the name of the scale (for images only)
grayscale	whether to use grayscale. By default, FALSE.
transf	a function to transform the pixel values (by default, NULL)
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

Value

A ‘ggplot‘ object rendering the pixel image.

`plot.imlist` *Plot a list of pixel images*

Description

Plot method for objects of class ‘imlist‘ (a list of spatstat pixel images, such as a sequence of density images over time periods). Renders one or more images as filled rasters over their observation window, arranging multiple frames into a common-legend grid.

Usage

```
## S3 method for class 'imlist'
plot(
  x,
  ...,
  main = "image",
  lim = NA,
  transf = NULL,
  frame = 1,
  scalename = "Density",
  color = c("white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", "#771F59FF"),
  grayscale = FALSE,
  ncol = NA,
  nrow = NA
)
```

Arguments

`x`	an object of class ‘imlist‘ (a list of spatstat pixel images), for example a sequence of density images produced by a ‘get_*_dens()‘ function.
`...`	additional arguments. Currently ignored.
`main`	title
`lim`	limits of the scale. By default, NA. To set limits manually, provide a vector or max and min
`transf`	a function to transform the pixel values (by default, NULL)
`frame`	the element number of the list object (by default, 1)
`scalename`	the name of the scale
`color`	the color scale. By default, "white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", and "#771F59FF".
`grayscale`	grayscale or not. By default, FALSE.
`ncol`	the number of columns (if plotitng multiple images at once)
`nrow`	the number of rows (if plotting multiple images at once)

Value

A ‘ggplot‘ object. For a single frame, the rendered image; for multiple frames, an arranged grid of images with a common legend.

`plot.list` *Plot a list of pixel images or point patterns*

Description

Plot method for objects of class ‘list‘. If the list contains spatstat pixel images (‘im‘), they are rendered as filled rasters; if it contains point patterns (‘ppp‘), the points are plotted over the observation window. Multiple frames are arranged into a grid (images) or combined/faceted (point patterns).

Usage

```
## S3 method for class 'list'
plot(
  x,
  ...,
  main = "list",
  lim = NA,
  transf = NULL,
  frame = 1,
  combined = TRUE,
  scalename = "Density",
  color = c("white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", "#771F59FF"),
  grayscale = FALSE,
  ncol = NA,
  nrow = NA
)
```

Arguments

`x`	a ‘list‘ of spatstat pixel images (‘im‘) or point patterns (‘ppp‘).
`...`	additional arguments. Currently ignored.
`main`	title
`lim`	limits of the scale. By default, NA. To set limits manually, provide a vector or max and min
`transf`	a function to transform the pixel values (by default, NULL)
`frame`	the element number of the list object (by default, 1)
`combined`	logical. ‘combined‘ specifies whether to combine all the point processes to one plot. This argument applies only to point-pattern lists with multiple time periods. By default = TRUE
`scalename`	the name of the scale

color	the color scale. By default, "white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", and "#771F59FF".
grayscale	grayscale or not. By default, FALSE.
ncol	the number of columns (if plotitng multiple images at once)
nrow	the number of rows (if plotting multiple images at once)

Value

A ‘ggplot‘ object visualizing the images or point patterns in the list.

`plot.obs` *Plot observed densities*

Description

Plot method for objects of class ‘obs‘, returned by [get_obs_dens()]. Compares actual versus predicted event counts over time, shows the corresponding residuals, and maps the average residual field. Up to three fitted densities can be overlaid for comparison.

Usage

```
## S3 method for class 'obs'
plot(
  x,
  ...,
  dens_2 = NA,
  dens_3 = NA,
  lim = c(-1, 1),
  time_unit = NA,
  combined = TRUE,
  color_actual = "darkgrey",
  color_dens_1 = "#f68f46ff",
  color_dens_2 = "#593d9cff",
  color_dens_3 = "#efe350ff"
)
```

Arguments

x	an object of class ‘obs‘, typically the output of [get_obs_dens()].
...	additional arguments. Currently ignored.
dens_2	density 2 (if any). By default, ‘NA‘.
dens_3	density 3 (if any). By default, ‘NA‘.
lim	limits of the scale for the average residual fields. By default, c(0, -1). To set limits manually, provide a vector of max and min
time_unit	x-axis label of the output

combined	whether to combine the two plots. By default, TRUE. If TRUE, then the plot function produces one ggplot object. If FALSE, three objects (two ggplot and one dataframe) will be produced.
color_actual	name of the color for the actual density
color_dens_1	name of the color for the predicted density
color_dens_2	name of the color for another predicted density (usually for the out-of-sample prediction)
color_dens_3	another color for an alternate density for out-of-sample prediction

Value

If ‘combined = TRUE‘ (the default), a single arranged ‘ggplot‘ object combining the comparison, residual, and (when applicable) average-residual-field panels. If ‘combined = FALSE‘, a named list with the underlying data frame (‘plot_data‘) and the individual ‘ggplot‘ objects (‘plot_compare‘, ‘plot_residual‘, ‘plot_arf‘).

See Also

[get_obs_dens()]

---

plot.powerlist *Plot simulated power densities*

Description

Plot method for objects of class ‘powerlist‘, returned by [sim_power_dens()]. Takes the simulated power densities and their priorities and returns a faceted power density image over the range of parameters.

Usage

```
## S3 method for class 'powerlist'
plot(
  x,
  ...,
  color = c("white", "#F8DAC5FF", "#F4825AFF", "#D2204CFF", "#771F59FF"),
  grayscale = FALSE
)
```

Arguments

x	an object of class ‘powerlist‘, typically the output of [sim_power_dens()].
...	additional arguments. Currently ignored.
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

A ‘ggplot‘ object (a faceted arrangement of power density images).

See Also

[sim_power_dens()]

`plot.ppplist` *Plot a list of point patterns*

Description

Plot method for objects of class ‘ppplist‘ (a list of spatstat point patterns, such as observed events across time periods). Plots the points over the observation window, either combining multiple time periods into one plot or faceting them.

Usage

```
## S3 method for class 'ppplist'
plot(x, ..., frame = 1, main = "ppp", combined = TRUE)
```

Arguments

`x`	an object of class ‘ppplist‘ (a list of spatstat point patterns).
`...`	additional arguments. Currently ignored.
`frame`	the element number of the list object (by default, 1)
`main`	title
`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

A ‘ggplot‘ object plotting the selected point pattern(s).

plot.supthin *Plot the results of superthinning tests*

Description

Plot method for objects of class ‘supthin‘, returned by [dx_supthin()]. Visualizes the observed and simulated-envelope summary functions (the K-function and the centered L-function) used to diagnose departures from complete spatial randomness after superthinning.

Usage

```
## S3 method for class 'supthin'
plot(x, ...)
```

Arguments

x	an object of class ‘supthin‘, typically the output of [dx_supthin()].
...	additional arguments. Currently ignored.

Value

A named list with two ‘ggplot‘ objects, ‘plot_k‘ (the K-function with simulation envelope) and ‘plot_l‘ (the centered L-function with simulation envelope).

See Also

[dx_supthin()]

plot.weights *Plot weights*

Description

Plot method for objects of class ‘weights‘, derived from the estimation produced by [get_est()]. Displays histograms of the (standardized or unstandardized) inverse-probability weights, faceted by intervention.

Usage

```
## S3 method for class 'weights'
plot(x, ..., type_weights = "standardized", binwidth = NULL)
```

Arguments

x	an object of class ‘weights‘, derived from the output of [get_est()].
...	additional arguments. Currently ignored.
type_weights	the type of weights to plot. - If ’plot_weights’ is ‘standardized‘, histogram of standardized weights will be generated. - If ’plot_weights’ is ‘unstandardized‘, histogram of unstandardized weights will be generated. Default is ‘standardized‘.
binwidth	bin width of the histogram. Default is NULL

Value

A ‘ggplot‘ object showing histograms of the weights, faceted by intervention.

See Also

[get_est()]

---

print.cate *Print results*

Description

Print method for objects of class ‘cate‘, returned by [get_cate()]. Builds and returns a data frame of the estimated CATE values at the chosen evaluation points, together with their 90

Usage

```
## S3 method for class 'cate'
print(x, ...)
```

Arguments

x	an object of class ‘cate‘, typically the output of [get_cate()].
...	additional arguments. Currently ignored.

Details

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

Value

A data frame with one row per evaluation value, containing the evaluation value, the point estimate, and the lower/upper bounds of the 90 and 95

See Also

[get_cate()]

print.est *Print results*

Description

Print method for objects of class ‘est‘, returned by [get_est()]. Builds and returns a data frame of the estimated causal effects per time period, together with their 90

Usage

```
## S3 method for class 'est'
print(x, ...)
```

Arguments

x	an object of class ‘est‘, typically the output of [get_est()].
...	additional arguments. Currently ignored.

Details

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

Value

A data frame with one row per window, containing the point estimate and the lower/upper bounds of the 90

See Also

[get_est()]

sens_weighted_surf *Generate weighted surfaces for sensitivity analysis*

Description

A variant of ‘get_weighted_surf()‘ that retains per-period Hajek weights for both the entire region and a window of interest, so that downstream linear-programming bounds can be computed.

Usage

```
sens_weighted_surf(
  obs_dens,
  cf_dens,
  treatment_data,
  smoothed_outcome,
  lag,
  entire_window,
  window,
  time_after,
  truncation_level = 0.95
)
```

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)
window	owin object (the sub-window of interest for which bounds are computed)
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

'sens_weighted_surf()' is an internal function used by 'get_sens()'.

Value

list of the following: * 'haj_wt_daily_out': Hajek-weighted daily outcomes integrated over the window of interest * 'haj_weights_window_of_interest': per-period Hajek weights for the window of interest * 'haj_weights_entire_window': per-period Hajek weights for the entire window

`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)
```

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

Details

'sim_cf_dens()' combines a baseline density with each of the power densities produced by 'sim_power_dens()'. For every simulated set of priorities, the baseline density is multiplied by the corresponding power density, normalized over 'window', and rescaled to the supplied 'expected_number' of events. The result is a 'cflist' object holding the list of counterfactual densities together with the associated powers and expected number.

Value

list of counterfactual densities, power as numerics, and expected number as a numeric

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

[sim_power_dens()], [get_distexp()]

Other density estimation functions: `get_adaptive_baseline_dens()`, `get_base_dens()`, `get_cf_dens()`, `get_cf_dens_adaptive()`, `get_cf_sum_log_intens()`, `get_obs_dens()`, `get_power_dens()`, `sim_power_dens()`

`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)
```

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

Details

‘sim_power_dens()‘ generates a sequence of power densities by holding the priorities of the fixed target densities at ‘priorities‘ while varying the priority of ‘dens_manip‘ across the values in ‘priorities_manip‘. For each such value, the product of the powered target densities is normalized over ‘window‘. The result is a ‘powerlist‘ object containing the list of power densities and the manipulated priorities, and is typically passed to ‘sim_cf_dens()‘.

Value

list of densities and priorities

References

Papadogeorgou, G., Imai, K., Lyall, J. and Li, F. (2022). Causal inference with spatio-temporal data: estimating the effects of airstrikes on insurgent violence in Iraq. *Journal of the Royal Statistical Society Series B*, 84(5), 1969–1999. doi:10.1111/rssb.12548

Mukaigawara, M., Imai, K., Lyall, J. and Papadogeorgou, G. (2025). Spatiotemporal causal inference with arbitrary spillover and carryover effects. arXiv preprint. doi:10.48550/arXiv.2504.03464

See Also

Other density estimation functions: `get_adaptive_baseline_dens()`, `get_base_dens()`, `get_cf_dens()`, `get_cf_dens_adaptive()`, `get_cf_sum_log_intens()`, `get_obs_dens()`, `get_power_dens()`, `sim_cf_dens()`

`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,
  resolution = NULL,
  ndim = NULL,
  ngroups = NULL
)
```

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).
`resolution`	resolution of raster (distance map) (in km)
`ndim`	the number of dimensions of grid cells (ndim^2). Users need to set either resolution or ndim.
`ngroups`	('method = "abramson"' only) the number of groups into which the point-specific bandwidths are partitioned, passed on to 'spatstat.univar::densityAdaptiveKernel()'. By default, 'NULL', which uses 'spatstat''s default (the square root of the number of points). Smaller values speed up the computation at the cost of coarser bandwidth approximation.

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 the ‘sampling‘ option is included in this function: with ‘sampling‘, the model is initialized with a random subset of the points, which considerably reduces the computation time for large datasets.

Another, more precise, method for smoothing outcomes is adaptive smoothing (‘method = "abramson"‘). 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.

Abramson’s rule requires a pilot estimate of the spatially varying density, which ‘smooth_ppp()‘ obtains automatically as follows. First, the points of all time periods are pooled into a single point pattern. Second, a global bandwidth is selected for this pooled pattern by Scott’s rule of thumb (‘spatstat.explore::bw.scott()‘, with separate bandwidths for the two coordinates), and the pilot density is estimated by fixed-bandwidth Gaussian kernel smoothing of the pooled pattern with these bandwidths. Third, point-specific bandwidths are computed with ‘spatstat.explore::bw.abram.ppp()‘ following the inverse-square-root rule, $h(u) = h_0 \min\{\tilde{f}(u)^{-1/2}/\gamma, \mathrm{trim}\}$, where h_0 is the global bandwidth (the mean of the two Scott bandwidths), $\tilde{f}(u)$ is the pilot density, γ is the geometric mean of the $\tilde{f}(u)^{-1/2}$ terms over the observed points (which puts h_0 on the scale of a fixed bandwidth), and the trimming value (5, the ‘spatstat‘ default) prevents excessively large bandwidths at isolated points (Hall and Marron, 1988). Finally, the point-specific bandwidths are assigned back to the points of each time period, and each time period is smoothed by ‘spatstat.univar::densityAdaptiveKernel()‘ with these bandwidths.

Parallel computation: ‘smooth_ppp()‘ smooths the point patterns of all time periods using the ‘future‘ framework (via ‘furrr‘), which runs sequentially unless a parallel backend is registered. To smooth the time periods in parallel, set a parallel plan before calling this function, e.g., ‘future::plan(future::multisession)‘. The results are identical regardless of the plan. Parallelization is most beneficial for ‘method = "abramson"‘, whose computation time is dominated by the adaptive smoothing of each time period; for ‘method = "mclust"‘, the model fitting itself is not parallelized, and the one-time cost of launching the parallel workers may outweigh the gains for the (fast) smoothing stage.

Value

im objects as a list

References

Abramson, I. S. (1982). On bandwidth variation in kernel estimates — a square root law. *The Annals of Statistics*, 10(4), 1217–1223.

Hall, P. and Marron, J. S. (1988). Variable window width kernel density estimates of probability densities. *Probability Theory and Related Fields*, 80, 37–49.

Scott, D. W. (1992). *Multivariate Density Estimation: Theory, Practice and Visualization*. New York: Wiley.

See Also

Other data preparation functions: `get_em_vec()`, `get_hfr()`, `get_hist()`, `imls_to_arr()`, `pixel_count_ppp()`

Examples

```
# Insurgency outcomes in Iraq, 2006 (first 60 days)
dat <- insurgencies_2006
dat$time <- as.numeric(dat$date - min(dat$date) + 1)
dat <- dat[dat$time <= 60, ]
hfr <- get_hfr(data = dat, col = "type", window = iraq_window,
               time_col = "time", time_range = c(1, 60),
               coordinates = c("longitude", "latitude"), combine = TRUE)

# Adaptive smoothing of the outcomes
# (to run in parallel, call future::plan(future::multisession) first)
smoothed <- smooth_ppp(hfr$all_combined, method = "abramson", ndim = 64)
```

---

summary.cate *Summarize results*

Description

Summary method for objects of class ‘cate‘, returned by [get_cate()]. Summarizes the estimated regression coefficients, the CATE evaluated at the chosen values (both with 90 test of whether at least one non-intercept coefficient differs from zero.

Usage

```
## S3 method for class 'cate'
summary(object, ..., significance_level = 0.05)
```

Arguments

`object`	an object of class ‘cate‘, typically the output of [get_cate()].
`...`	additional arguments. Currently ignored.
`significance_level`	Numeric scalar between 0 and 1, inclusive, representing the significance level for the chi-square test. The test is used to determine whether at least one of the coefficients (except the intercept) is not equal to 0. Default is 0.05

Details

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

Value

A named list with three elements: ‘result_betas‘, a data frame of the estimated regression coefficients with 90 ‘result_values‘, a data frame of the CATE evaluated at the chosen values with 90 chi-square statistic, its p-value, the significance level, and the test decision.

See Also

[get_cate()]

---

summary.est *Summarize results*

Description

Summary method for objects of class ‘est‘, returned by [get_est()]. Summarizes the estimated causal effects per time period (with 90 confidence intervals) together with the associated windows.

Usage

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

Arguments

`object`	an object of class ‘est‘, typically the output of [get_est()].
`...`	additional arguments. Currently ignored.

Details

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

Value

A named list with two elements: ‘result‘, a data frame of the point estimates and 90 of windows used in estimation.

See Also

[get_est()]

summary.obs *Summarize results*

Description

Summary method for objects of class ‘obs‘, returned by [get_obs_dens()]. Prints a short summary of the fitted observed-density model (dispersion parameter, deviance, and null deviance) and returns the estimated model coefficients.

Usage

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

Arguments

object	an object of class ‘obs‘, typically the output of [get_obs_dens()].
...	additional arguments. Currently ignored.

Details

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

Value

The estimated model coefficients (‘object$coef‘), returned alongside the printed model summary.

See Also

[get_obs_dens()]

Index