Package 'gsClusterDetect'

July 27, 2026

Type Package

Title Utilities for Geo-Spatial Cluster Detection and Significance Classification

Version 1.0.1

Description Provides utilities for manipulating time series of location-based counts of events to detect geo-spatial clusters. Significance of these clusters is determined using a set of models that classify based on a learned relationship between observed and the log(observed/expected) ratio of counts. The approach implemented here is similar to prospective space-time estimation of clusters using the scan statistic.

URL https://github.com/lmullany/gsClusterDetect, https://lmullany.github.io/gsClusterDetect/

BugReports https://github.com/lmullany/gsClusterDetect/issues

License Apache License (>= 2)

Encoding UTF-8

LazyData true

Imports cli, data.table (>= 1.16.0), sf

Depends R (>= 4.3)

Suggests ggplot2, plotly, tigris, testthat (>= 3.0.0), withr, knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

RoxygenNote 8.0.0

NeedsCompilation no

Author Luke C. Mullany [aut, cre], Howard S. Burkom [aut]

Maintainer Luke C. Mullany <luke.mullany@jhuapl.edu>

Repository CRAN

Date/Publication 2026-07-27 15:10:02 UTC

Contents

`add_location_counts` *Add location counts to cluster location list*

Description

Add counts of individual cluster locations. Operates on the output list of the `compress_clusters()` component. Calculates individual location counts for each cluster, and appends to the cluster location list.

Usage

```
add_location_counts(cluster_list, cases)
```

Arguments

`cluster_list`	output list from 'compress_clusters' (i.e. an object of class 'clusters'), which contains two elements: a data frame of cluster summary rows and a data frame of the locations in each cluster
`cases`	original data in 3-column format of location, count, date

Value

the cluster list from compress_clusters with individual location counts appended

Examples

```
case_grid <- generate_case_grids(
  example_count_data, example_count_data[, max(date)]
)
nci <- gen_nearby_case_info(
  cg = case_grid,
  distance_matrix = county_distance_matrix("OH")[["distance_matrix"]],
  distance_limit = 25
)
obs_exp_grid <- generate_observed_expected(
  nearby_counts = nci,
  case_grid = case_grid
)
cla <- add_spline_threshold(oe_grid = obs_exp_grid)
# use compress clusters to reduce
cla <- compress_clusters_fast(
  cluster_alert_table = cla,
  distance_matrix = county_distance_matrix("OH")[["distance_matrix"]]
)
# Now add the location counts
add_location_counts(
  cluster_list = cla,
  cases = example_count_data
)
```

add_spline_threshold *Use spline lookup to restrict 'ObservedExpectedGrid' to potential clusters*

Description

Function takes a spline lookup table (or uses package default), and an object of class 'Observed-ExpectedGrid' and identifies which rows in each potential centroid have observed over expected values that exceed a threshold for that observed value

Usage

```
add_spline_threshold(oe_grid, spline_lookup = NULL)
```

Arguments

oe_grid	An object of class 'ObservedExpectedGrid' generated by `generate_observed_expected()`
spline_lookup	default NULL; either a spline lookup table, which is a data frame that has at least two columns: including "observed" and "spl_thresh", OR a string indicating to use one of the built in lookup tables: i.e. one of `"001"`, `"005"`, `"01"`, `"05"`. If NULL, the default table will be 01 (i.e. `spline_01` dataset)

Value

an object of class 'ClusterAlertTable' which is simply a data frame containing rows of the input 'oe_grid" that represent the reduced set of candidate alert clusters

Examples

```
case_grid <- generate_case_grids(
  example_count_data, example_count_data[, max(date)]
)
nci <- gen_nearby_case_info(
  cg = case_grid,
  distance_matrix = county_distance_matrix("OH")[["distance_matrix"]],
  distance_limit = 25
)
obs_exp_grid <- generate_observed_expected(
  nearby_counts = nci,
  case_grid = case_grid
)
add_spline_threshold(oe_grid = obs_exp_grid)
add_spline_threshold(oe_grid = obs_exp_grid, spline_lookup = "01")
```

check_vars *check for variables in frame*

Description

Function checks for variables in frame

Usage

```
check_vars(d, required)
```

Arguments

d	input data frame to check for variables
required	vector of column names that must be in ‘d‘

Value

No return value, called for side effects

compress_clusters *Compress a cluster_alert_table*

Description

Function reduces an object of class ‘ClusterAlertTable“ to the final set of clusters and locations. The idea of this function is to retain only the most significant, non-overlapping clusters from the cluster alert table. The surrogate for significance is ’alertGap’, or log(observed/expected) minus the threshold that the spline assigns to the observed value. The logic in this function keeps two running tables, the table ‘dt_keep‘ of clusters to be kept, in descending order of ‘alertGap‘, and ‘dt_clust‘, the remaining rows of the cluster alert table, which are reduced each time a cluster is accepted into ‘dt_keep‘. Each row of the cluster alert table represents a candidate cluster, with a column ‘target‘, which is the cluster center, and a column ‘location‘, the most distant location from the center. Each time a cluster is accepted into ‘dt_keep‘, the remaining rows of ‘dt_clust‘ are dropped if either ‘target‘ or ‘location‘ is the center of the newly accepted cluster. in ‘dt_keep‘

Usage

```
compress_clusters(cluster_alert_table, distance_matrix)
```

Arguments

cluster_alert_table	an object of class ‘ClusterAlertTable‘
distance_matrix	a square distance matrix, named on both dimensions or a list of distance vectors, one for each location

Value

an object of class ‘clusters‘, which is simply a a list including a a data.frame of clusters and another frame of individual location counts

Examples

```
case_grid <- generate_case_grids(
  example_count_data, example_count_data[, max(date)]
)
nci <- gen_nearby_case_info(
  cg = case_grid,
  distance_matrix = county_distance_matrix("OH")[["distance_matrix"]],
  distance_limit = 25
)
obs_exp_grid <- generate_observed_expected(
  nearby_counts = nci,
  case_grid = case_grid
)
cla <- add_spline_threshold(oe_grid = obs_exp_grid)
compress_clusters(
  cluster_alert_table = cla,
  distance_matrix = county_distance_matrix("OH")[["distance_matrix"]]
)
```

---

compress_clusters_fast

Fast version of compress clusters

Description

Function reduces an object of class ClusterAlertTable to the final set of clusters and locations. The idea of this function is to retain only the most significant, non-overlapping clusters from the cluster alert table. The surrogate for significance is 'alertGap', or log(observed/expected) minus the threshold that the spline assigns to the observed value‘.

Usage

```
compress_clusters_fast(cluster_alert_table, distance_matrix)
```

Arguments

cluster_alert_table	an object of class ‘ClusterAlertTable‘
distance_matrix	a square distance matrix, named on both dimensions or a list of distance vectors, one for each location

Value

an object of class ‘clusters‘, which is simply a a list including a a data.frame of clusters and another frame of individual location counts

Examples

```
case_grid <- generate_case_grids(
  example_count_data, example_count_data[, max(date)]
)
nci <- gen_nearby_case_info(
  cg = case_grid,
  distance_matrix = county_distance_matrix("OH")[["distance_matrix"]],
  distance_limit = 25
)
obs_exp_grid <- generate_observed_expected(
  nearby_counts = nci,
  case_grid = case_grid
)
cla <- add_spline_threshold(oe_grid = obs_exp_grid)
compress_clusters_fast(
  cluster_alert_table = cla,
  distance_matrix = county_distance_matrix("OH")[["distance_matrix"]]
)
```

---

counties *County Location Dataset*

Description

A data set that provides latitude and longitude for each county in the United Sates

Usage

```
counties
```

Format

A data frame with 3,144 rows and 6 columns:

state_name, state full and abbreviated names for states

state_fips, fips state and county fips codes

longitude, latitude numeric coordinates for fips

Source

‘tigris‘ package

`county_distance_matrix` *Get distance matrix for counties within a state*

Description

Function returns a list of counties and a matrix with the distance between those counties. leverages a built in dataset (‘counties‘).

Usage

```
county_distance_matrix(
  st,
  unit = c("miles", "kilometers", "meters"),
  source = c("tigris", "rnssp")
)
```

Arguments

`st`	two-character string denoting a state, or "US". If "US", then this is equivalent to calling `us_distance_matrix()`.
`unit`	string, one of "miles" (default), "kilometers", or "meters". Indicating the desired unit for the distances
`source`	string indicating either "tigris" (default) or "rnssp". Both are built-in datasets (i.e. are part of this package). The default ("tigris") uses county names and locations as found in tigris 2024. The "rnssp" option uses a package-stored version of the publicly available shape file for counties from Rnssp package at https://cdcgov.github.io/Rnssp/

Value

a named list of length two; first element (‘loc_vec‘) is a vector of locations and the second element (‘distance_matrix‘) is a square matrix containing the pairwise distance (in the given ‘unit‘) between all locations.

Examples

```
county_distance_matrix("MD", source = "tigris")
county_distance_matrix("WI", source = "rnssp", unit = "kilometers")
```

`create_custom_dist_list` *Create a sparse distance list from custom location data*

Description

This function is a custom-data version of `create_dist_list()`. It returns a list of named numeric vectors where each list element contains only locations within `threshold` distance units of a target location.

Usage

```
create_custom_dist_list(
  df,
  label_var,
  lat_var,
  long_var,
  threshold,
  unit = c("miles", "kilometers", "meters")
)
```

Arguments

`df`	data.frame containing label and coordinate columns
`label_var`	character scalar; column name used as location label (must be unique and non-missing)
`lat_var`	character scalar; latitude column name.
`long_var`	character scalar; longitude column name.
`threshold`	numeric scalar distance cutoff in units of `unit`
`unit`	string, one of "miles" (default), "kilometers", or "meters"

Value

a named list, where each element, named by a target location, is a named vector of distances that are within ‘threshold‘ ‘units‘ of the target.

Examples

```
md <- tract_generator("MD")
dlist <- create_custom_dist_list(
  df = md,
  label_var = "geoid",
  lat_var = "latitude",
  long_var = "longitude",
  threshold = 15,
  unit = "miles"
)
```

`create_dist_list` *Generalized distance list as sparse list*

Description

This function is an alternative to the package functions that create a square distance matrix of dimension N, with all pairwise distances. In this approach a list of named vectors is returned, where there is one element in the list for each location, and each named vector holds the distance within 'threshold' of the location.

Usage

```
create_dist_list(
  level,
  threshold,
  st = NULL,
  county = NULL,
  unit = c("miles", "kilometers", "meters")
)
```

Arguments

`level`	string either "state", "county", "zip", or "tract"
`threshold`	numeric value; include in each location-specific named vector only those locations that a within 'threshold' distance units of the target. Reasonable thresholds might be 250 (miles), 50 (miles), 15 (miles) and 3 (miles) for county, zip, and tract, respectively, but these can be adjusted. Note if a different unit other than miles is used, then the user should also adjust this parameter appropriately
`st`	string; optional to specify a state; if NULL distances are returned for all zip codes, counties, or states in the US
`county`	string vector of 3-fips to restrict within `st`; ignored unless `level` is "tract"
`unit`	string one of miles (default), kilometers, or meters; this is the unit relevant to the threshold

Value

a named list, where each element, named by a target location, is a named vector of distances that are within 'threshold' 'units' of the target.

Examples

```
create_dist_list(
  level = "tract",
  threshold = 3,
  st = "MD"
)
create_dist_list(
```

```
  level = "county",
  threshold = 50,
  st = "CA",
  unit = "kilometers"
)
```

`custom_distance_matrix` *Build a Distance Matrix from a Custom Data Frame*

Description

Generates an all-pairs distance matrix from latitude/longitude coordinates in a user-supplied data frame. Row and column names of the matrix are set from a unique label variable.

Usage

```
custom_distance_matrix(
  df,
  unit = c("miles", "kilometers", "meters"),
  label_var,
  lat_var,
  long_var
)
```

Arguments

`df`	A `data.frame` containing label and coordinate columns.
`unit`	Character string; one of `"miles"` (default), `"kilometers"`, or `"meters"`.
`label_var`	Character scalar; column name to use for matrix row/column names. Values in this column must be unique and non-missing.
`lat_var`	Character scalar; column name containing latitude values.
`long_var`	Character scalar; column name containing longitude values.

Value

A list with:

loc_vec Character vector of location labels (same order as matrix dimensions)

distance_matrix Square numeric matrix of pairwise distances in requested units

Examples

```
md <- tract_generator("24")
dm <- custom_distance_matrix(
  md,
  label_var = "geoid", lat_var = "latitude", long_var = "longitude"
)
dim(dm[["distance_matrix"]])

names(md) <- c("tract_id", "lat", "lon")
dm_km <- custom_distance_matrix(
  md,
  unit = "kilometers",
  label_var = "tract_id",
  lat_var = "lat",
  long_var = "lon"
)
```

---

example_count_data *Example Count Dataset*

Description

Synthetic county-level example count data for package examples and tests. Generation included a synthetic injection of cases near the end of the time series to ensure that clusters are detected in this example dataset.

Usage

```
example_count_data
```

Format

A data frame with 11,264 rows and 4 columns:

location county FIPS code as character

date date of observation

count non-negative integer daily count

Source

package authors

find_clusters *Find clusters*

Description

Function will return clusters, given a frame of case counts by location and date, a distance matrix, a spline lookup table, and other parameters

Usage

```
find_clusters(
  cases,
  distance_matrix,
  detect_date,
  spline_lookup = NULL,
  baseline_length = 90,
  max_test_window_days = 7,
  guard_band = 0,
  distance_limit = 15,
  baseline_adjustment = c("add_one", "add_one_global", "add_test", "none"),
  adj_constant = 1,
  min_clust_cases = 0,
  max_clust_cases = Inf,
  use_fast = TRUE,
  return_interim = FALSE
)
```

Arguments

cases	a frame of case counts by location and date
distance_matrix	a square distance matrix, named on both dimensions or a list of distance vectors, one for each location
detect_date	a date that indicates the end of the test window in which we are looking for clusters
spline_lookup	default NULL; either a spline lookup table, which is a data frame that has at least two columns: including "observed" and "spl_thresh", OR a string indicating to use one of the built in lookup tables: i.e. one of "001", "005", "01", "05". If NULL, the default table will be 01 (i.e. spline_01 dataset)
baseline_length	integer (default = 90) number of days in the baseline interval
max_test_window_days	integer (default = 7) number of days for the test window
guard_band	integer (default = 0) buffer days between baseline and test interval

`distance_limit` numeric (default=15) maximum distance to consider cluster size. Note that the units of the value default (miles) should be the same unit as the values in the distance matrix

`baseline_adjustment`
one of four string options: "add_one" (default), "add_one_global", "add_test", or "none". All methods except for "none" will ensure that the log(obs/expected) is always defined (i.e. avoids expected =0). For the default, this will add 1 to the expected for any individual calculation if expected would otherwise be zero. "add_one_global", will add one to all baseline location case counts. For "add_test_interval", each location in the baseline is increased by the number of cases in that location during the test interval. If "none", no adjustment is made.

`adj_constant` numeric (default=1.0); this is the constant to be added if `baseline_adjustment == 'add_one'` or `baseline_adjustment == 'add_one'`

`min_clust_cases`
(default = 0); minimum number of case within a returned cluster.

`max_clust_cases`
(default = Inf); maximum number of cases within a returned cluster.

`use_fast` boolean (default = TRUE) - set to TRUE to use the fast version of the compress clusters function

`return_interim` boolean (default = FALSE) - set to TRUE to return all interim objects of the `find_clusters()` function

Value

returns a list of two of two dataframes.

Examples

```
find_clusters(
  cases = example_count_data,
  distance_matrix = county_distance_matrix("OH")[["distance_matrix"]],
  detect_date = example_count_data[, max(date)],
  distance_limit = 50
)
```

`generate_case_grids` *Get test period and baseline period case counts*

Description

Given raw case counts by location, and some dates and other params return a list of frames and values that summarize the number or cases in the baseline and test period

Usage

```
generate_case_grids(
  cases,
  detect_date,
  baseline_length = 90,
  max_test_window_days = 7,
  guard_band = 0,
  baseline_adjustment = c("add_one", "add_one_global", "add_test", "none"),
  adj_constant = 1
)
```

Arguments

`cases`	frame of cases with counts, location(s) and dates
`detect_date`	date to end examination of detection of clusters
`baseline_length`	number of days (integer) used for baseline detection (default = 90)
`max_test_window_days`	integer, max number of days in a detected cluster, defaults to 7
`guard_band`	integer (default=0) number of days buffer between test interval and baseline
`baseline_adjustment`	one of three string options: "add_one" (default), "add_test", or "none". All methods except for "none" will ensure that the log(obs/expected) is always defined (i.e. avoids expected =0). For the default, this will add 1 to the expected for any individual calculation if expected would otherwise be zero. For "add_test_interval", each location in the baseline is increased by the number of cases in that location during the test interval. If "none", no adjustment is made.
`adj_constant`	numeric (default=1.0); this is the constant to be added if `baseline_adjustment == 'add_one'` or `baseline_adjustment == 'add_one'`

Value

an object of class ‘CaseGrids‘ contain a list of items

- ‘baseline_counts_by_location‘: a frame of counts over the baseline interval by location
- ‘case_grid‘: a frame of cases during the test dates, with reverse cumulative counts within location, by date
- ‘case_grid_totals_by_date‘: reverse cumulative sum of counts over all locations, by date
- ‘test_cases‘: case location counts only during the test dates
- ‘detect_date‘: the detect date passed to this function
- ‘baseline_total‘: an integer holding the total counts over all locations and dates

Examples

```
dd <- example_count_data[, max(date)]
generate_case_grids(
  cases = example_count_data,
  detect_date = dd
)
```

`generate_heatmap` *Generate heatmap of data*

Description

Generate a ggplot heatmap of count information by date and location given a frame of count-by-location-and-date data.

Usage

```
generate_heatmap(heatmap_data, plot_type = c("ggplot", "plotly"), ...)
```

Arguments

`heatmap_data`	data frame generated by ‘generate_heatmap_data‘
`plot_type`	string indicating either a "ggplot" or "plotly" result. If the requested backend is unavailable, the function warns and falls back to the other backend when available.
`...`	passed onto plotly

Value

a ggplot or plotly object

Examples

```
hd <- generate_heatmap_data(example_count_data)
generate_heatmap(hd)
generate_heatmap(hd, plot_type = "plotly")
```

`generate_heatmap_data` *Get heat map data from a set of location, date, count data*

Description

Generate heat map data frame count information by date and location given an input frame of count-by-location-and-date data.

Usage

```
generate_heatmap_data(
  data,
  end_date = NULL,
  locations = NULL,
  baseline_length = 90,
  test_length = 7,
  guard = 0,
  break_points = c(-1, 2, 4, 9, 19, Inf),
  break_labels = c("0-1", "2-4", "5-9", "10-19", "20+")
)
```

Arguments

`data`	data frame with (at least) three columns: location, date, count
`end_date`	date indicating end of test interval; if not provided the last date in 'dt' will be used
`locations`	a vector of locations to subset the table; if none provided then all locations will be used
`baseline_length`	numeric (default=90) number of days in baseline interval
`test_length`	numeric (default=7) number of days in test interval
`guard`	numeric (default=0) number of days between baseline and test interval
`break_points`	break points for the discrete groups (default = `c(-1,2,4,9,19,Inf)`)
`break_labels`	string vector of labels for the groups (default = `c("0-1", "2-4", "5-9", "10-19", "20+")`)

Value

a data frame of heat map data

Examples

```
generate_heatmap_data(
  data = example_count_data
)
```

`generate_observed_expected` *Generate the observed and expected information*

Description

Function takes an object of class 'NearbyClusterGrids', as returned from `gen_nearby_case_info()`, and an object of class 'CaseGrids', as returned from `generate_case_grids()`, and adds observed and expected information.

Usage

```
generate_observed_expected(
  nearby_counts,
  case_grid,
  adjust = FALSE,
  adj_constant = 1
)
```

Arguments

`nearby_counts`	an object of class 'NearbyClusterGrids'
`case_grid`	an object of class 'CaseGrids'
`adjust`	boolean default TRUE, set to `FALSE` to avoid adding one to the expected when it is zero. Could result in errors.
`adj_constant`	numeric (default=1.0); this is the constant to be added if `baseline_adjustment == 'add_one'` or `baseline_adjustment == 'add_one'`

Value

a dataframe of class 'ObservedExpectedGrid', which is simply a data frame with the observed and expected calculation incorporated for all target locations

Examples

```
case_grid <- generate_case_grids(
  example_count_data,
  example_count_data[, max(date)]
)
nci <- gen_nearby_case_info(
  cg = case_grid,
  distance_matrix = county_distance_matrix("OH")[["distance_matrix"]],
  distance_limit = 25
)
generate_observed_expected(
  nearby_counts = nci,
  case_grid = case_grid
)
```

`generate_summary_table` *Summary count-by-location-and-date data, given baseline and test interval lengths, and an end-date for the test interval*

Description

Function will return a summary data frame of information related to a given count-by-location-and-date dataset, provided the user gives the count data, a set of locations, and the length of the baseline and test intervals, and and end date for the test interval. Note that a guard, a buffer between the end of the baseline interval and the test interval can be provided.

Usage

```
generate_summary_table(
  data,
  end_date = NULL,
  locations = NULL,
  baseline_length = 90,
  test_length = 7,
  guard = 0,
  cut_vec = c(0, 1.5, 2.5, 5.5, 10.5, Inf),
  cut_labels = c("Nr. Locs, daily mean 1 or less", "Nr. Locs, daily mean 2",
   "Nr. Locs, daily mean 3-5", "Nr. Locs, daily mean 6-10", "Nr. Locs, daily mean >10")
)
```

Arguments

`data`	data frame with (at least) three columns: location, date, count
`end_date`	date indicating end of test interval; if not provided the last date in ‘dt‘ will be used
`locations`	a vector of locations to subset the table; if none provided then all locations will be used
`baseline_length`	numeric (default=90) number of days in baseline interval
`test_length`	numeric (default=7) number of days in test interval
`guard`	numeric (default=0) number of days between baseline and test interval
`cut_vec`	numeric vector of n cut points to examine categories of daily mean counts
`cut_labels`	character vector of labels for the n-1 categories created by ‘cut_vec‘

Value

data frame of summary statistics

Examples

```
generate_summary_table(
  data = example_count_data
)
```

`generate_time_series_data` *Generate time series data*

Description

Function returns a time series of counts-by-location-and-date data, given length of baseline and test intervals, and an end date for the test-interval

Usage

```
generate_time_series_data(
  data,
  end_date = NULL,
  locations = NULL,
  baseline_length = 90,
  test_length = 7,
  guard = 0
)
```

Arguments

`data`	data frame with (at least) three columns: location, date, count
`end_date`	date indicating end of test interval; if not provided the last date in ‘dt‘ will be used
`locations`	a vector of locations to subset the table; if none provided then all locations will be used
`baseline_length`	numeric (default=90) number of days in baseline interval
`test_length`	numeric (default=7) number of days in test interval
`guard`	numeric (default=0) number of days between baseline and test interval

Value

a dataframe of time series data

Examples

```
generate_time_series_data(
  data = example_count_data
)
```

`generate_time_series_plot` *Generate timeseries plot data*

Description

Generate a timeseries plot of count information by date and location given a frame of count-by-location-and-date data and an optional end_date

Usage

```
generate_time_series_plot(
  time_series_data,
  end_date = NULL,
  plot_type = c("ggplot", "plotly"),
  locations = "All Locations",
  ...
)
```

Arguments

`time_series_data`	data frame generated by ‘generate_time_series_data‘
`end_date`	optional end date to truncate date
`plot_type`	string indicating either a "ggplot" or "plotly" result. If the requested backend is unavailable, the function warns and falls back to the other backend when available.
`locations`	string indicating location name (defaults to "All Locations")
`...`	passed onto plotly

Value

a ggplot or plotly object

Examples

```
ts <- generate_time_series_data(example_count_data)
generate_time_series_plot(ts)
generate_time_series_plot(ts, plot_type = "plotly")
```

`gen_nearby_case_info` *Return baseline and test period case grids restricting by distance*

Description

Function takes a distance matrix between locations, a set of baseline period case sums by location, and grid of test period cases by date and location, and given a distance limit, returns two frames: 1. A frame that has for each location, a list of nearby locations and the cumulative sum of cases from those locations (over increasing distance) in the baseline period 2. A frame that has for each location, a list of nearby locations and the observed cumulative sum of cases by date (over increasing distance) in the test period

Usage

```
gen_nearby_case_info(cg, distance_matrix, distance_limit)
```

Arguments

`cg`	object of class 'CaseGrids', such as returned from the `generate_case_grids()`
`distance_matrix`	a square distance matrix, named on both dimensions or a list of distance vectors, one for each location
`distance_limit`	numeric value indicating the distance threshold to define "near" locations; must be input in the same units as the distances in the 'distance_matrix'. Note that if passing the list version of distance_matrix, this limit has already been used in that construction and thus is ignored

Value

an object of class 'NearbyClusterGrids' which is list of two dataframes, including "baseline" (has the nearby information for baseline counts) and "test" (which holds the nearby information for test interval counts)

Examples

```
case_grid <- generate_case_grids(
  example_count_data, example_count_data[, max(date)]
)
nci <- gen_nearby_case_info(
  cg = case_grid,
  distance_matrix = county_distance_matrix("OH")[["distance_matrix"]],
  distance_limit = 25
)
```

get_baseline_dates *Generate baseline dates vector*

Description

Function to generate baseline dates given an end date and test length, plus optional guard, and length of baseline

Usage

```
get_baseline_dates(end_date, test_length, baseline_length, guard = 0)
```

Arguments

end_date	End date of the test interval
test_length	(integer) length of the test interval in days
baseline_length	(integer) length of baseline period in days
guard	(integer) default = 0; buffer between end of baseline and start of test interval

Value

vector of dates

Examples

```
get_baseline_dates(
  end_date = "2025-01-01",
  test_length = 10,
  baseline_length = 90
)
```

get_nearby_locations *Get nearby locations*

Description

Given a location, a square distance matrix, and numeric value (radius_miles), this helper function returns a 2-column data frame listing the locations within that radius

Usage

```
get_nearby_locations(center_location, distance_matrix, radius_miles)
```

Arguments

center_location	location
distance_matrix	a distance matrix
radius_miles	a numeric value >0

Value

a data.table

Examples

```
dm <- zip_distance_matrix("MD")$distance_matrix
nearby_locations <- get_nearby_locations("21228", dm, 10)
```

---

get_test_dates *Generate test dates vector*

Description

Function to generate test dates given an end date and test length

Usage

```
get_test_dates(end_date, test_length)
```

Arguments

end_date	End date of the test interval
test_length	(integer) length of the test interval in days

Value

vector of dates

Examples

```
get_test_dates(
  end_date = "2025-01-01",
  test_length = 10
)
```

map_clusters *Map Clusters*

Description

Utility for mapping clusters. The function ingest an object ‘cl‘ as returned by `find_clusters()` and a shape file provisioned by the caller, along with a string name of a column in the shape file that uniquely defines the locations. Note that this column will be used to merge with the clusters and therefore must aligned with the labels in ‘cl‘ cluster locations. The function returns a basic plotly map object that can be further modified by the user

Usage

```
map_clusters(
  cl,
  s,
  s_id,
  label_id = NULL,
  label = c("none", "cluster_centers", "cluster_locations", "all"),
  engine = c("plotly", "ggplot"),
  point_crs = NULL
)
```

Arguments

`cl`	an object of class "clusters" as returned by `find_clusters()`
`s`	shape file; must be of class sf
`s_id`	string unique identifier of ‘s‘
`label_id`	string column of ‘s‘ that indicates display label for the row in ‘s‘ (default is NULL)
`label`	for `engine = "ggplot"`, indicates which locations should receive visible text labels. Valid choices are `"none"`, `"cluster_centers"`, `"cluster_locations"`, and `"all"`. The default is `"none"`. This argument is ignored when `engine = "plotly"` because plotly maps always include hover labels for all locations.
`engine`	string label to indicate plotting engine; either "plotly" (default) or "ggplot"
`point_crs`	optional coordinate reference system used to compute representative points when ‘s‘ is in longitude/latitude coordinates. If ‘NULL‘, EPSG:3857 is used as a general-purpose fallback. The resulting points are transformed back to the CRS of ‘s‘ before plotting

Examples

```
if (
  requireNamespace("tigris", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE)
) {
```

```
  # get some data
  dd <- example_count_data[, max(date)]
  # get a distance matrix
  dm <- create_dist_list("county", 50, st = "OH")
  # find the clusters
  cl <- find_clusters(
    cases = example_count_data,
    detect_date = dd,
    distance_matrix = dm
  )
  # get shape file
  ohio_shape <- tigris::counties("OH", cb = TRUE, class = "sf")

  # prepare map data
  md <- map_clusters(cl, ohio_shape, "GEOID")
}
```

`prepare_map_data` *Prepare cluster result and shape file for mapping*

Description

Function ingest an clusters object as returned by `find_clusters()`, a shape file, and a unique identifier for the locations in that shape file, and prepares these objects for mapping.

Usage

```
prepare_map_data(cl, s, s_id, label_id = NULL, point_crs = NULL)
```

Arguments

`cl`	an object of class "clusters" as returned by `find_clusters()`
`s`	shape file; must be of class sf
`s_id`	string unique identifier of ‘s‘
`label_id`	string column of ‘s‘ that indicates display label for the row in ‘s‘ (default is NULL)
`point_crs`	optional coordinate reference system used to compute representative points when ‘s‘ is in longitude/latitude coordinates. If ‘NULL‘, EPSG:3857 is used as a general-purpose fallback. The resulting points are transformed back to the CRS of ‘s‘ before plotting

Examples

```
# example code
# get some data
dd <- example_count_data[, max(date)]
# get a distance matrix
dm <- create_dist_list("county", 50, st = "OH")
```

```
# find the clusters
cl <- find_clusters(
  cases = example_count_data,
  detect_date = dd,
  distance_matrix = dm
)
# get shape file
ohio_shape <- tigris::counties("OH", cb = TRUE, class = "sf")

# prepare map data
md <- prepare_map_data(cl, ohio_shape, "GEOID")
md <- prepare_map_data(cl, ohio_shape, "GEOID", label_id = "NAME")
```

`reduce_clusters_to_min` *Filter clusters on minimum overall count*

Description

Function takes a set of clusters identified via `compress_clusters()` and a minimum threshold for counts, and reduces the identified clusters to only those clusters where the total number of observed across the cluster meets that minimum threshold.

Usage

```
reduce_clusters_to_min(cl, minimum = 0)
```

Arguments

`cl`	a object of class `clusters`, as returned from `compress_clusters`
`minimum`	numeric (default = 0); minimum number across all locations in a cluster in order to retain

Value

an object of class `clusters`

Examples

```
cl <- find_clusters(
  cases = example_count_data,
  distance_matrix = county_distance_matrix("OH")[["distance_matrix"]],
  detect_date = example_count_data[, max(date)],
  distance_limit = 50
)
reduce_clusters_to_min(cl, 50)
```

spline_001 *Spline Lookup Table - 0.001*

Description

Spline threshold lookup table, p-value = 0.001

Usage

```
spline_001
```

Format

A data frame with 399 rows and 2 columns:

observed number of observed in cluster

spl_thresh log observed-over-expected above which cluster is significant at the 0.001 level

Source

package authors

spline_005 *Spline Lookup Table - 0.005*

Description

Spline threshold lookup table, p-value = 0.005

Usage

```
spline_005
```

Format

A data frame with 399 rows and 2 columns:

observed number of observed in cluster

spl_thresh log observed-over-expected above which cluster is significant at the 0.005 level

Source

package authors

spline_01 *Spline Lookup Table - 0.01*

Description

Spline threshold lookup table, p-value = 0.01

Usage

```
spline_01
```

Format

A data frame with 399 rows and 2 columns:

observed number of observed in cluster

spl_thresh log observed-over-expected above which cluster is significant at the 0.01 level

Source

package authors

spline_05 *Spline Lookup Table - 0.05*

Description

Spline threshold lookup table, p-value = 0.05

Usage

```
spline_05
```

Format

A data frame with 399 rows and 2 columns:

observed number of observed in cluster

spl_thresh log observed-over-expected above which cluster is significant at the 0.05 level

Source

package authors

states *State Location Dataset*

Description

A data set that provides latitude and longitude for each state in the United Sates

Usage

```
states
```

Format

A data frame with 56 rows and 5 columns:

state_name, state full and abbreviated names for states

state_fips state fips codes

longitude, latitude numeric coordinates for state fips

Source

'tigris' package

state_distance_matrix *Get distance matrix for states in the US*

Description

Function returns a list of states and a matrix with the distance between those states. leverages a built in dataset ('states')

Usage

```
state_distance_matrix(unit = c("miles", "kilometers", "meters"))
```

Arguments

unit	string, one of "miles" (default), "kilometers", or "meters". Indicating the desired unit for the distances

Value

a named list of length two; first element ('loc_vec') is a vector of locations and the second element ('distance_matrix') is a square matrix containing the pairwise distance (in the given 'unit') between all locations.

Examples

```
state_distance_matrix()
state_distance_matrix(unit = "kilometers")
```

`st_injects` *Add data counts for parameterized injected clusters*

Description

Function st_injects returns a list of two objects 1. a full dataset as a data.table with inject counts added according to design parameters. 2. a table of only the inject counts, locations, and dates.

Usage

```
st_injects(
  cases,
  distance_matrix,
  target_loc,
  center_decile,
  radius_miles,
  nr_cases,
  nr_days,
  end_date
)
```

Arguments

`cases`	data frame of cases
`distance_matrix`	a distance matrix
`target_loc`	a location into which the injection should occur
`center_decile`	an integer value between 1 and 10, inclusive
`radius_miles`	a numeric value >0
`nr_cases`	number of cases to inject
`nr_days`	number of days over which we want to inject cases
`end_date`	last date of injection

Value

a two-element list; each element is a dataframe. The first is the full dataset with injected cases and the second is the injected cases only

Examples

```
cases <- example_count_data
dm <- county_distance_matrix("OH")
target_loc <- "39175"
scen1 <- st_injects(
  cases = cases,
  distance_matrix = dm[["distance_matrix"]],
  target_loc = target_loc,
  center_decile = 7,
  radius_miles = 70,
  nr_cases = 100,
  nr_days = 4,
  end_date = "2025-02-05"
)
```

`tract_distance_matrix` *Build a Tract Distance Matrix for a State*

Description

Creates an all-pairs distance matrix between census tract centroids for a state, using state abbreviation input similar to `zip_distance_matrix()`.

Usage

```
tract_distance_matrix(
  st,
  county = NULL,
  unit = c("miles", "kilometers", "meters"),
  use_cache = TRUE,
  ...
)
```

Arguments

`st`	Character scalar; 2-character USPS state abbreviation (for example, `"MD"`).
`county`	A three-digit FIPS code (string) of the county or counties to subset on. This can also be a county name or vector of names.
`unit`	Character string; one of `"miles"` (default), `"kilometers"`, or `"meters"`.
`use_cache`	Logical; if `TRUE`, enables `options(tigris_use_cache = TRUE)`.
`...`	arguments passed on to tigris::tracts

Value

A list with:

loc_vec Character vector of tract GEOIDs (same order as matrix dimensions)

distance_matrix Square numeric matrix of pairwise distances in requested units

Examples

```
md_dm <- tract_distance_matrix("MD")
dim(md_dm$distance_matrix)
md_dm_km <- tract_distance_matrix("MD", unit = "kilometers")
```

`tract_generator` *Generate Census Tract Centroids for a State*

Description

Pulls census tracts using **tigris**, computes tract centroids, and returns a three-column **data.table** with GEOID, latitude, and longitude.

Usage

```
tract_generator(st, county = NULL, use_cache = TRUE, ...)
```

Arguments

`st`	Character scalar; either a 2-digit state FIPS code (for example, `"24"`) or a 2-letter USPS abbreviation (for example, `"MD"`).
`county`	A three-digit FIPS code (string) of the county or counties to subset on. This can also be a county name or vector of names.
`use_cache`	a boolean, defaults to TRUE, to set tigris option to use cache
`...`	arguments to be passed on to tigris::tracts()

Value

A `data.table` with columns:

geoid 11-digit tract GEOID (`state(2)` + `county(3)` + `tract(6)`)

latitude Centroid latitude in WGS84

longitude Centroid longitude in WGS84

Examples

```
md_tracts <- tract_generator("24")
md_tracts2 <- tract_generator("MD")
howard_county_tracts <- tract_generator("MD", county = "027")
head(md_tracts)
```

us_distance_matrix *Get distance matrix for all counties in the US*

Description

Function returns a list of counties and a matrix with the distance between those counties. leverages a built in dataset (‘counties‘). Note that the generation of this matrix can take a few seconds. Note: it is better and faster to use `create_dist_list()`.

Usage

```
us_distance_matrix(unit = c("miles", "kilometers", "meters"))
```

Arguments

`unit`	string, one of "miles" (default), "kilometers", or "meters". Indicating the desired unit for the distances

Value

a named list of length two; first element (‘loc_vec‘) is a vector of locations and the second element (‘distance_matrix‘) is a square matrix containing the pairwise distance (in the given ‘unit‘) between all locations.

Examples

```
# Takes ~ 10 seconds, depending on machine
us_distance_matrix(unit = "kilometers")
```

zipcodes *Zipcode Location Dataset*

Description

A data set that provides latitude and longitude for each zipcode in the United Sates

Usage

```
zipcodes
```

Format

A data frame with 42,482 rows and 11 columns:

id serial integer id (1, 2, 3, .. etc)

zip_code 5 digit string for zipcode

state state abbreviation

county county name

region region name

region_id id for region

region_name region name

pop, modified undocumented

latitude, longitude numeric coordinates for zipcode

Source

unknown

---

zip_distance_matrix *Get distance matrix for zip codes within a state*

Description

Function returns a list of zipcodes and a matrix with the distance between those zip codes. leverages a built in dataset (‘zipcodes‘) that maps zipcodes to counties.

Usage

```
zip_distance_matrix(st, unit = c("miles", "kilometers", "meters"))
```

Arguments

st	two-character string denoting a state
unit	string, one of "miles" (default), "kilometers", or "meters". Indicating the desired unit for the distances

Value

a named list of length two; first element (‘loc_vec‘) is a vector of locations and the second element (‘distance_matrix‘) is a square matrix containing the pairwise distance (in the given ‘unit‘) between all locations.

Examples

```
zip_distance_matrix("MD")
```

Index