

Package ‘treeSS’

October 2, 2026

Type Package

Title Tree-Spatial Scan Statistic for Cluster Detection

Version 0.2.9

Description Implements the tree-spatial scan statistic for detecting clusters that combine both spatial and hierarchical structures, as proposed by Cancado et al. (2025) <[doi:10.1007/s10651-025-00670-w](https://doi.org/10.1007/s10651-025-00670-w)>. The method extends Kulldorff (1997) <[doi:10.1080/03610929708831995](https://doi.org/10.1080/03610929708831995)> circular spatial scan statistic and the tree-based scan statistic of Kulldorff et al. (2003) <[doi:10.1111/1541-0420.00039](https://doi.org/10.1111/1541-0420.00039)> by searching for anomalies in both geographic regions and branches of hierarchical trees simultaneously. The package also provides standalone implementations of Kulldorff's circular spatial scan statistic and the tree-based scan statistic. Statistical significance is assessed via Monte Carlo simulation under a Poisson or binomial model, with optional 'OpenMP' parallelization.

License GPL (>= 3)

URL <https://github.com/allanvc/treeSS>

BugReports <https://github.com/allanvc/treeSS/issues>

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.0), stats

LinkingTo Rcpp

Suggests testthat (>= 3.0.0), knitr, rmarkdown, sf, ggplot2

SystemRequirements GNU make

Encoding UTF-8

LazyData true

LazyDataCompression xz

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Allan Quadros [aut, cre] (ORCID:
<https://orcid.org/0000-0003-3250-5380>),
 Andre L. F. Cançado [aut] (ORCID:
<https://orcid.org/0000-0002-7019-8521>)

Maintainer Allan Quadros <allanvcq@gmail.com>

Config/roxygen2/version 8.1.0

Repository CRAN

Date/Publication 2026-10-02 16:20:02 UTC

Contents

treeSS-package	3
aggregate_tree	4
build_zones	5
chicago_crimes	6
chicago_map	8
chicago_tree	8
circular_scan	9
filter_clusters	11
fl_deaths	13
generate_example_data	14
get_cluster_regions	15
london_boroughs_map	17
london_collisions	18
london_tree	19
most_likely_cluster	20
plot.treess	21
plot.tree_scan	23
print.circular_scan	24
print.sequential_scan	24
print.treespatial_scan	25
print.tree_scan	26
pvalue	26
rj_map	27
rj_mortality	28
rj_tree	29
secondary_clusters	30
sequential_scan	31
summary.circular_scan	34
summary.sequential_scan	34
summary.treespatial_scan	35
summary.tree_scan	36
treespatial_scan	36
tree_scan	39

Index

41

Description

Implements the tree-spatial scan statistic for detecting clusters that combine both spatial and hierarchical structures, as proposed by Cançado et al. (2025). The method extends Kulldorff's (1997) circular spatial scan statistic and the tree-based scan statistic (Kulldorff et al. 2003) by searching for anomalies in both geographic regions and branches of hierarchical trees simultaneously.

Main functions

`treespacial_scan` Performs the tree-spatial scan statistic, detecting clusters defined by pairs of spatial zones and tree branches.

`circular_scan` Performs Kulldorff's circular spatial scan statistic.

`tree_scan` Performs the tree-based scan statistic.

Helper functions

`build_zones` Constructs candidate spatial zones using circular windows centered on region centroids.

`aggregate_tree` Aggregates case counts from leaves to internal nodes of the hierarchical tree.

`filter_clusters` Removes overlapping secondary clusters from a single-pass scan output (Cançado et al. 2025, Sec. 5.1.1).

`sequential_scan` Sequential adjustment for secondary clusters: removes the most likely cluster's regions and re-runs the scan with a fresh Monte Carlo simulation (Zhang, Assunção & Kulldorff 2010).

`get_cluster_regions` Returns a tidy region-by-cluster data.frame suitable for plotting with any mapping package.

Author(s)

Maintainer: Allan Quadros <allanvcq@gmail.com> ([ORCID](#))

Authors:

- Allan Quadros <allanvcq@gmail.com> ([ORCID](#))
- Andre L. F. Cançado ([ORCID](#))

References

Cançado, A. L. F., Oliveira, G. S., Quadros, A. V. C., & Duczmal, L. (2025). A tree-spatial scan statistic. *Environmental and Ecological Statistics*, 32, 953–978. doi:10.1007/s1065102500670w

Kulldorff, M. (1997). A spatial scan statistic. *Communications in Statistics - Theory and Methods*, 26(6), 1481–1496.

Kulldorff, M., Fang, Z., & Walsh, S. J. (2003). A tree-based scan statistic for database disease surveillance. *Biometrics*, 59(2), 323–331.

Zhang, Z., Assunção, R., & Kulldorff, M. (2010). Spatial scan statistics adjusted for multiple clusters. *Journal of Probability and Statistics*, 2010, 642379.

See Also

Useful links:

- <https://github.com/allanvc/treeSS>
- Report bugs at <https://github.com/allanvc/treeSS/issues>

aggregate_tree	<i>Aggregate Case Counts from Leaves to All Nodes in a Hierarchical Tree</i>
----------------	--

Description

Given case counts at the leaf level (supplied as columns of a long-format data `data.frame`) and a tree, aggregates case counts upward from child nodes to parent nodes, producing a complete case matrix indexed by all nodes (rows) and regions (columns).

Usage

```
aggregate_tree(
  data,
  cases,
  region_id,
  node_id,
  tree = NULL,
  tree_node_id = NULL,
  tree_parent_id = NULL
)
```

Arguments

<code>data</code>	A <code>data.frame</code> in long format with one row per (region, leaf) observation. The <code>cases</code> , <code>region_id</code> and <code>node_id</code> arguments name columns of this <code>data.frame</code> .
<code>cases</code>	Column of data giving case counts. Given as an unquoted column name (a string or expression also works).
<code>region_id</code>	Column of data giving the region identifier.
<code>node_id</code>	Column of data giving the leaf identifier.
<code>tree</code>	A <code>data.frame</code> with columns <code>node_id</code> and <code>parent_id</code> . As an alternative, pass <code>tree_node_id</code> and <code>tree_parent_id</code> .
<code>tree_node_id</code> , <code>tree_parent_id</code>	Optional. Parallel vectors as an alternative to <code>tree</code> .

Details

This function is exposed for inspection and pedagogical use; the scan functions call it internally on the matrix they build from your input data.

Value

A matrix of dimensions $m \times k$ (nodes x regions), with rows ordered by `tree$node_id` and columns by region.

References

Cancado, A. L. F., Oliveira, G. S., Quadros, A. V. C., & Duczmal, L. (2025). A tree-spatial scan statistic. *Environmental and Ecological Statistics*, 32, 953-978. doi:10.1007/s1065102500670w

Examples

```
tree <- data.frame(
  node_id = c(1, 2, 3, 4, 5, 6, 7),
  parent_id = c(NA, 1, 1, 2, 2, 3, 3)
)
# Leaves are 4, 5, 6, 7; long-format data.frame
dat <- data.frame(
  cases = c(10, 5, 3, 8, 2, 7, 4, 1, 6, 3, 9, 2),
  region_id = rep(1:3, each = 4),
  node_id = rep(c(4, 5, 6, 7), times = 3)
)
aggregate_tree(dat, cases = cases, region_id = region_id,
  node_id = node_id, tree = tree)
```

build_zones	<i>Build Candidate Spatial Zones Using Circular Windows</i>
-------------	---

Description

Constructs the set of candidate spatial zones for the circular scan statistic. Each zone is defined by a circular window centered on a region’s centroid, containing all regions whose centroids fall within the circle.

Usage

```
build_zones(regions, max_pop = NULL)
```

Arguments

- regions A data.frame with columns region_id, population, x, and y. Coordinates x and y represent centroids of each region.
- max_pop A numeric value specifying the maximum population allowed inside a zone. If NULL (default), it is set to 50% of the total population.

Details

For each region j , regions are sorted by distance from j . Regions are added incrementally to the zone as long as the total population does not exceed `max_pop`. This generates a nested set of circular windows centered on each region.

Value

A list of zones. Each zone is a list with elements:

- center** Integer index of the center region.
- region_idx** Integer vector of region indices inside the zone.
- population** Total population inside the zone.

References

Kulldorff, M. (1997). A spatial scan statistic. *Communications in Statistics - Theory and Methods*, 26(6), 1481–1496.

Examples

```
regions <- data.frame(
  region_id = 1:5,
  population = c(1000, 2000, 1500, 800, 1200),
  x = c(0, 1, 2, 0.5, 1.5),
  y = c(0, 0, 0, 1, 1)
)
zones <- build_zones(regions, max_pop = 3000)
length(zones) # number of candidate zones
```

chicago_crimes	Chicago Crime (2023)
----------------	----------------------

Description

Combined long-format dataset of crime incidents in the 77 community areas of Chicago, 2023. Use together with [chicago_tree](#).

Usage

```
chicago_crimes
```

Format

A data.frame in long format with columns:

region_id Integer identifier of the community area.

area_number Official community area number (1–77).

name Community area name.

population Total incidents in the area. This is a compositional denominator (sums to the total incident count of the city) and is useful for analyses that ask "which crime types over-occur in which areas relative to the citywide profile". Not a population at risk.

pop_residential Residential population of the community area. Use this as the denominator for incidence-rate analyses (incidents per resident). Source: U.S. Census Bureau, ACS 2020 5-year estimates, aggregated to community areas by CMAP Community Data Snapshots. The 77 values sum to approximately 2.71M, matching the published Chicago population.

x Longitude of the centroid.

y Latitude of the centroid.

node_id Character leaf identifier from [chicago_tree](#).

cases Integer count of crime incidents.

Source

Crime incidents: City of Chicago Data Portal, Crimes – 2001 to Present.

Residential population: U.S. Census Bureau ACS 2020 5-year estimates, aggregated to community areas by the Chicago Metropolitan Agency for Planning (CMAP), Community Data Snapshots (2023 release). <https://cmap.illinois.gov/data/community-data-snapshots/>

See Also

[chicago_tree](#), [chicago_map](#)

Examples

```
data(chicago_crimes)
head(chicago_crimes)
cat("Total incidents:", sum(chicago_crimes$cases), "\n")
cat("Total residents:",
    sum(unique(chicago_crimes[, c("area_number", "pop_residential")])$pop_residential),
    "\n")
```

`chicago_map`*Chicago Community Area Boundaries*

Description

Simplified polygon boundaries for the 77 Chicago community areas.

Usage`chicago_map`**Format**

An sf data.frame with 77 rows and 3 columns:

NAME Community area name.

AREA_NUM Community area number (integer, 1–77).

geometry MULTIPOLYGON geometry in WGS84 (EPSG:4326).

Source

City of Chicago Data Portal, Community Area Boundaries.

See Also

[chicago_crimes](#)

Examples

```
data(chicago_map)
cat("Community areas:", nrow(chicago_map), "\n")
```

`chicago_tree`*Chicago Crime - Hierarchy*

Description

Hierarchical classification of crime incidents by type, description and location group.

Usage`chicago_tree`

Format

A data.frame with 2,841 rows and 2 columns:

node_id Character identifier of each node.

parent_id Character identifier of the parent node. NA for the root.

Details

4-level tree: Root -> Crime Type -> Description -> Location Group. 2,486 leaf nodes. Leaf node IDs follow the pattern "<TYPE> | <DESCRIPTION> | <LOCATION>".

Source

Constructed from the leaf-level codes in [chicago_crimes](#). See data-raw/ in the GitHub repository for the build script.

See Also

[chicago_crimes](#)

Examples

```
data(chicago_tree)
head(chicago_tree)
```

circular_scan

Kulldorff's Circular Spatial Scan Statistic

Description

Performs Kulldorff's circular spatial scan statistic for detecting spatial clusters. Inputs are supplied through a single data data.frame with one row per region (cases must already be aggregated to the region level); the relevant columns are referenced by name.

Usage

```
circular_scan(
  data,
  cases,
  population,
  region_id,
  x,
  y,
  max_pop_pct = 0.5,
  nsim = 999L,
  alpha = 0.05,
  n_secondary = 1000L,
```

```

model = c("poisson", "binomial"),
seed = NULL,
n_cores = 1L
)

```

Arguments

data	A data.frame with one row per region. The cases, population, region_id, x and y arguments name columns of this data.frame.
cases	Column of data giving the total cases in each region. Given as an unquoted column name (a string or expression also works).
population	Column of data giving the population (or denominator) of each region.
region_id	Column of data giving the region identifier.
x, y	Columns of data giving the region centroid coordinates.
max_pop_pct	Numeric. Default 0.5.
nsim	Integer. Number of MC simulations. Default 999.
alpha	Numeric. Significance level. Default 0.05.
n_secondary	Integer. Default 1000.
model	Character. "poisson" or "binomial".
seed	Integer or NULL. Random seed for the Monte Carlo loop. When non-NULL, the user's pre-existing RNG state is saved on entry and restored on exit, so the seed argument affects only this call and does not leak into subsequent draws in the user's session.
n_cores	Integer. OpenMP threads.

Value

An object of class "circular_scan".

References

Kulldorff, M. (1997). A spatial scan statistic. *Communications in Statistics - Theory and Methods*, 26(6), 1481-1496.

See Also

[filter_clusters](#), [tree_scan](#), [treespatial_scan](#), [get_cluster_regions](#), [sequential_scan](#)

Examples

```

set.seed(42)
n <- 20
regions <- data.frame(
  region_id = 1:n,
  cases     = rpois(n, lambda = 10),
  population = rep(1000, n),
  x         = runif(n, 0, 10),

```

```

      y          = runif(n, 0, 10)
    )
    regions$cases[1:5] <- rpois(5, lambda = 30)

    result <- circular_scan(
      regions,
      cases      = cases,
      population = population,
      region_id  = region_id,
      x          = x,
      y          = y,
      nsim       = 99
    )
    print(result)

```

filter_clusters	<i>Filter Overlapping Secondary Clusters</i>
-----------------	--

Description

Removes overlapping secondary clusters from the output of [treespatial_scan](#), [circular_scan](#), or [tree_scan](#).

Usage

```

filter_clusters(result, alpha = NULL, ...)

## Default S3 method:
filter_clusters(result, alpha = NULL, ...)

## S3 method for class 'tree_scan'
filter_clusters(result, alpha = NULL, ...)

## S3 method for class 'circular_scan'
filter_clusters(result, alpha = NULL, ...)

## S3 method for class 'treespatial_scan'
filter_clusters(result, alpha = NULL, ...)

```

Arguments

result	An object of class "treespatial_scan", "circular_scan", or "tree_scan".
alpha	Numeric. Significance level for filtering. Default is NULL, which uses the alpha from the original analysis.
...	Further arguments passed to methods (currently unused).

Details

For `treespacial_scan` objects, two clusters overlap if they have BOTH tree overlap (one node is ancestor/descendant of the other) AND spatial overlap (same center or identical set of regions). This follows Cancado et al. (2025).

For `circular_scan` objects, the criterion follows Kulldorff (1997): a secondary cluster is distinct if its center is not contained in any previously retained cluster's zone, and no previously retained cluster's center is contained in the candidate zone.

For `tree_scan` objects, a secondary cluster is distinct if its node is NOT an ancestor or descendant of any previously retained node. This follows Kulldorff et al. (2003).

When a dominant signal saturates the candidate pool (making distinct secondary clusters hard to find), or when clusters shadow each other, consider using [sequential_scan](#) (Zhang, Assuncao and Kulldorff, 2010), which re-runs the scan after removing each detected cluster.

Value

A data.frame of distinct significant clusters ordered by descending LLR.

References

- Kulldorff, M. (1997). A spatial scan statistic. *Communications in Statistics - Theory and Methods*, 26(6), 1481-1496.
- Kulldorff, M., Fang, Z., & Walsh, S. J. (2003). A tree-based scan statistic for database disease surveillance. *Biometrics*, 59(2), 323-331.
- Cancado, A. L. F., Oliveira, G. S., Quadros, A. V. C., & Duczmal, L. (2025). A tree-spatial scan statistic. *Environmental and Ecological Statistics*, 32, 953-978. doi:10.1007/s1065102500670w

See Also

[treespacial_scan](#), [circular_scan](#), [tree_scan](#), [sequential_scan](#)

Examples

```
data(london_collisions); data(london_tree)
result <- treespacial_scan(
  london_collisions,
  cases          = cases,
  population      = population,
  region_id       = region_id,
  x               = x,
  y               = y,
  node_id         = node_id,
  tree            = london_tree,
  nsim            = 99, seed = 42
)
filter_clusters(result)
```

fl_deaths

*Florida General Mortality Data (2016)***Description**

Death counts by county and ICD-10 cause of death for the state of Florida, USA, 2016. This is **raw data** – the user builds the ICD-10 tree and the cases matrix in the analysis script, making it a pedagogical example of the full workflow.

Usage

fl_deaths

Format

A data.frame with 3,066 rows and 6 columns:

county_fips 5-digit FIPS code (character), e.g. "12086".

county_name County name, e.g. "Miami-Dade County".

icd10_code ICD-10 cause of death code, e.g. "I25.1".

icd10_desc Description of the ICD-10 code, e.g. "Atherosclerotic heart disease".

deaths Number of deaths (integer).

population County population estimate (integer).

Details

Covers 65 of Florida's 67 counties (2 small counties excluded by CDC suppression rules), 253 ICD-10 codes, and 157,000 total deaths. All ages, all causes.

To use with [treespatial_scan](#), rename and merge with centroids:

```
# Centroids from tigris
data <- transform(fl_deaths,
                  region_id = county_fips,
                  node_id   = icd10_code,
                  cases      = deaths)
# Add x, y from a centroids table
data <- merge(data, centroids, by = "county_fips")
treespatial_scan(data, cases, population, region_id, x, y, node_id,
                  tree = fl_tree, ...)
```

County polygons and centroids can be obtained via `tigris::counties(state = "FL", cb = TRUE)`.

The ICD-10 descriptions in `icd10_desc` can be used to build a lookup table: `unique(fl_deaths[, c("icd10_code", "icd10_desc")])`.

Source

CDC WONDER Compressed Mortality File 1999–2016 (<https://wonder.cdc.gov/cmfi-icd10.html>).

Examples

```
data(fl_deaths)
head(fl_deaths)
cat("Counties:", length(unique(fl_deaths$county_fips)), "\n")
cat("ICD-10 codes:", length(unique(fl_deaths$icd10_code)), "\n")
cat("Total deaths:", sum(fl_deaths$deaths), "\n")
```

generate_example_data *Generate Example Data for Tree-Spatial Scan*

Description

Creates a synthetic dataset for demonstrating and testing the tree-spatial scan statistic. Returns parallel vectors (cases, population, region_id, x, y, node_id) and a tree, matching the input format expected by [treespatial_scan](#).

Usage

```
generate_example_data(
  n_regions = 50L,
  pop_per_region = 1000,
  cluster_regions = 1:7,
  cluster_leaves = c(3, 4),
  rr = 2,
  Cg = 200L,
  seed = NULL
)
```

Arguments

n_regions	Integer. Default 50.
pop_per_region	Numeric. Default 1000.
cluster_regions	Integer vector. Default 1:7.
cluster_leaves	Integer vector. Default c(3, 4).
rr	Numeric. Relative risk. Default 2.0.
Cg	Integer. Cases per branch. Default 200.
seed	Integer or NULL. Random seed. When non-NULL, the user's pre-existing RNG state is saved on entry and restored on exit, so the seed argument affects only the result of the call. Default NULL (the user's session-level RNG state is used as-is and is not modified by the function).

Value

A list with vector components ready to feed into `treespacial_scan`: `cases`, `population`, `region_id`, `x`, `y`, `node_id`, plus the tree (`data.frame`) and a `true_cluster` list describing the injected cluster.

Examples

```
ex <- generate_example_data(seed = 42)
dat <- data.frame(
  cases = ex$cases, population = ex$population, region_id = ex$region_id,
  x = ex$x, y = ex$y, node_id = ex$node_id
)
result <- treespatial_scan(
  dat,
  cases      = cases,
  population = population,
  region_id  = region_id,
  x          = x,
  y          = y,
  node_id    = node_id,
  tree       = ex$tree,
  nsim       = 99
)
print(result)
```

get_cluster_regions	<i>Extract Cluster Membership for Each Region</i>
---------------------	---

Description

Returns a `data.frame` indicating which cluster (if any) each region belongs to. This is the primary output for visualization — use it with any mapping package (`ggplot2`, `leaflet`, `tmap`, etc.).

Usage

```
get_cluster_regions(result, n_clusters = 1L, overlap = TRUE, ...)

## Default S3 method:
get_cluster_regions(result, n_clusters = 1L, overlap = TRUE, ...)

## S3 method for class 'sequential_scan'
get_cluster_regions(result, n_clusters = 1L, overlap = TRUE, ...)
```

Arguments

result	A "treespacial_scan" or "circular_scan" object (single-pass scan), or a "sequential_scan" object (Zhang, Assuncao and Kulldorff, 2010).
--------	---

<code>n_clusters</code>	Integer. (Single-pass methods only.) Number of clusters to extract. 1 returns only the most likely cluster. Values greater than 1 use <code>filter_clusters</code> internally to identify distinct secondary clusters. Default is 1. Ignored for sequential scan inputs (all detected iterations are returned).
<code>overlap</code>	Logical. If TRUE (default), returns a facet-ready data.frame: all regions are replicated for each cluster panel, with a <code>panel</code> column for faceting and <code>cluster</code> marked only for member regions. If FALSE, returns one row per region assigned to its highest-ranked cluster (for single-panel maps).
<code>...</code>	Further arguments passed to methods.

Details

This is a generic with methods for objects returned by `treespatial_scan`, `circular_scan`, and `sequential_scan`.

Value

A data.frame with columns from `result$regions` plus:

cluster Integer cluster number (1 = most likely / first iteration, 2 = first secondary / second iteration, etc.), or NA if the region is not in the cluster.

node_id The tree node of the cluster, or NA.

llr The log-likelihood ratio of the cluster, or NA.

pvalue The p-value of the cluster, or NA.

panel (Only when `overlap = TRUE`) A two-line label for `facet_wrap`, with the cluster identifier on the first line and the test statistic on the second. For single-pass scans the label looks like `"#1 P209\n(LR=39.6)"`; for sequential scans it looks like `"Iter 1: P209\n(LR=39.6, p=0.005)"`. The newline keeps long node identifiers from overflowing the strip in multi-panel layouts.

See Also

`treespatial_scan`, `circular_scan`, `sequential_scan`, `filter_clusters`

Examples

```
data(london_collisions); data(london_tree)
result <- treespatial_scan(
  london_collisions,
  cases      = cases,
  population = population,
  region_id  = region_id,
  x          = x,
  y          = y,
  node_id    = node_id,
  tree       = london_tree,
  nsim       = 99, seed = 42
)
```

```
# Long format suitable for merging with a polygon layer.  
cr <- get_cluster_regions(result, n_clusters = 2L, overlap = TRUE)  
head(cr)
```

london_boroughs_map	<i>London Borough Boundaries</i>
---------------------	----------------------------------

Description

Simplified polygon boundaries for the 33 London boroughs. Use with [london_collisions](#) for map-based visualization of cluster results.

Usage

```
london_boroughs_map
```

Format

An sf data.frame with 33 rows and 3 columns:

NAME Borough name, e.g. "Westminster".

GSS_CODE ONS geography code, e.g. "E09000033".

geometry MULTIPOLYGON geometry in WGS84 (EPSG:4326).

Details

Geometries are simplified (`st_simplify(dTolerance = 0.001)`) to reduce file size. For full-resolution boundaries, download from the source URL.

To merge with cluster results from [get_cluster_regions](#), use: `merge(london_boroughs_map, cr, by.x = "NAME", by.y = "borough")`.

Source

London Datastore, Statistical GIS Boundary Files for London (see <https://data.london.gov.uk/dataset/statistical-gis-boundary-files-london/>). Contains National Statistics data (c) Crown copyright and database right 2014.

See Also

[london_collisions](#)

Examples

```
data(london_boroughs_map)  
cat("Boroughs:", nrow(london_boroughs_map), "\n")
```

london_collisions	<i>London Road Collisions (2022)</i>
-------------------	--------------------------------------

Description

Combined long-format dataset of road traffic collisions in the 33 London boroughs, 2022. Each row represents a (borough, collision-category) combination with at least one collision. Use together with [london_tree](#).

Usage

```
london_collisions
```

Format

A data.frame in long format with columns:

region_id Integer identifier of the borough.

borough Borough name.

population Total collisions in the borough (used as population denominator).

x Longitude of the borough centroid.

y Latitude of the borough centroid.

node_id Character leaf identifier from [london_tree](#).

cases Integer count of collisions.

Source

UK Department for Transport, STATS19 data via the stats19 R package.

See Also

[london_tree](#), [london_boroughs_map](#)

Examples

```
data(london_collisions)
head(london_collisions)
cat("Total collisions:", sum(london_collisions$cases), "\n")
```

london_tree

London Road Collisions - Hierarchy

Description

Hierarchical classification of road collisions by light conditions, road type, and junction detail.

Usage

london_tree

Format

A data.frame with 81 rows and 2 columns:

node_id Character identifier of each node.

parent_id Character identifier of the parent node. NA for the root.

Details

3-level tree: Root -> Light conditions (Daylight/Darkness) -> Road type (5 types) -> Junction detail (7 types). 68 leaf categories.

Source

Constructed from the leaf-level codes in [london_collisions](#). See data-raw/ in the GitHub repository for the build script.

See Also

[london_collisions](#)

Examples

```
data(london_tree)
head(london_tree)
```

most_likely_cluster	<i>Extract the Most Likely Cluster from a Scan Result</i>
---------------------	---

Description

Returns the most likely cluster detected by a scan, without requiring the user to access the internal list structure of the result object.

Usage

```
most_likely_cluster(x, ...)

## S3 method for class 'treess'
most_likely_cluster(x, ...)

## S3 method for class 'sequential_scan'
most_likely_cluster(x, ...)
```

Arguments

x	A scan result object: "circular_scan", "tree_scan", "treespatial_scan", or "sequential_scan" (all of which inherit from "treess").
...	Further arguments passed to methods (currently unused).

Value

A named list describing the most likely cluster. The components depend on the scan type: for `circular_scan`, region IDs, cases, expected count, population, relative risk, and log-likelihood ratio; for `tree_scan`, additionally the node ID and leaf IDs; for `treespatial_scan`, both the tree node (with leaf IDs) and the spatial zone (with region IDs). For `sequential_scan`, the cluster detected in the first iteration is returned (subsequent iterations are secondary clusters by construction; see [secondary_clusters](#)).

See Also

[secondary_clusters](#), [pvalue](#), [get_cluster_regions](#)

Examples

```
ex <- generate_example_data(n_regions = 16, seed = 1)
dat <- data.frame(cases = ex$cases, population = ex$population,
                 region_id = ex$region_id, x = ex$x, y = ex$y,
                 node_id = ex$node_id)
res <- treespatial_scan(dat, cases = cases,
                      population = population,
                      region_id = region_id, x = x, y = y,
                      node_id = node_id, tree = ex$tree,
                      nsim = 19, seed = 42)
```

```
most_likely_cluster(res)
```

plot.treess

Plot the Clusters Detected by a Scan

Description

Draws the most likely cluster and, optionally, the distinct secondary clusters (single-pass scans) or every iteration (sequential scans) of a spatial scan result. Without a polygon layer the method uses base graphics and the region centroids stored in the result; with an **sf** polygon layer it builds a faceted **ggplot2** choropleth.

Usage

```
## S3 method for class 'treess'
plot(
  x,
  map = NULL,
  map_id = NULL,
  key = NULL,
  n_clusters = 1L,
  palette = NULL,
  na_fill = "gray95",
  wrap = 30L,
  ...
)
```

Arguments

x	A "circular_scan", "treespatial_scan", or "sequential_scan" object (all inherit from "treess").
map	Optional sf data frame with one polygon per region. If NULL (default), the region centroids stored in the result are plotted with base graphics.
map_id	Name of the column of map that identifies the region. Required when map is supplied.
key	Optional two-column data.frame translating the scan's region identifiers into the values of map[[map_id]]: the first column must hold region_id values as used in the scan, the second the matching value in map[[map_id]]. Omit when map[[map_id]] already contains the scan's region_id values.
n_clusters	Integer. Number of clusters to draw for single-pass scans: 1 (default) draws the most likely cluster only; larger values add the distinct secondary clusters retained by filter_clusters , one panel each. Ignored for "sequential_scan" objects, where every iteration is drawn.
palette	Character vector of fill colors, recycled over clusters. The default is a four-color qualitative palette.

<code>na_fill</code>	Fill color for regions outside the cluster of a panel.
<code>wrap</code>	Integer. Width at which panel labels are wrapped.
<code>...</code>	For the base-graphics mode, further arguments passed to <code>plot</code> (e.g. <code>cex</code> , <code>main</code>). Ignored in the ggplot2 mode.

Details

The method is a convenience wrapper around `get_cluster_regions`: it performs the region-to-polygon join, replicates the map once per cluster panel, and applies a fixed palette. Users who prefer another mapping toolkit (e.g. **leaflet** or **tmap**) can call `get_cluster_regions()` directly and join its output to their own layer.

Value

In the base-graphics mode, the cluster-region table from `get_cluster_regions()` is returned invisibly. In the **ggplot2** mode a `ggplot` object is returned, which can be modified further with `+`.

See Also

[get_cluster_regions](#), [filter_clusters](#), [sequential_scan](#)

Examples

```
data(london_collisions); data(london_tree)
res <- treespatial_scan(
  london_collisions,
  cases = cases, population = population, region_id = region_id,
  x = x, y = y, node_id = node_id, tree = london_tree,
  nsim = 99, seed = 42
)

# Base graphics on centroids: most likely cluster
plot(res)

# Two distinct clusters, one panel each
plot(res, n_clusters = 2)

# Choropleth with the shipped borough polygons (requires sf, ggplot2).
# The scan uses integer region_id; the map is keyed by borough NAME,
# so a two-column key translates one into the other.
if (requireNamespace("sf", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE)) {
  data(london_boroughs_map)
  key <- unique(london_collisions[, c("region_id", "borough")])
  plot(res, map = london_boroughs_map, map_id = "NAME", key = key)
}
```

plot.tree_scan	<i>Plot the Top Tree Cuts of a Tree-Only Scan</i>
----------------	---

Description

A "tree_scan" result has no geography. Its plot method draws a dot chart of the highest-scoring tree cuts by log-likelihood ratio, marking those significant at the scan's alpha.

Usage

```
## S3 method for class 'tree_scan'
plot(x, n = 10L, ...)
```

Arguments

x	A "tree_scan" object.
n	Integer. Number of top cuts to display (default 10).
...	Further arguments passed to dotchart .

Value

The table of plotted cuts, invisibly.

See Also

[tree_scan](#), [plot.treess](#)

Examples

```
tree <- data.frame(node_id = c(1, 2, 3, 4, 5, 6, 7, 8),
  parent_id = c(NA, 1, 1, 2, 2, 3, 3, 3))
leaf_data <- data.frame(node_id = c(4, 5, 6, 7, 8),
  cases = c(50, 5, 3, 2, 4))
res <- tree_scan(leaf_data, cases = cases, node_id = node_id,
  tree = tree, nsim = 99, seed = 1)
plot(res, n = 8)
```

```
print.circular_scan
```

Print Method for circular_scan Objects

Description

Print Method for circular_scan Objects

Usage

```
## S3 method for class 'circular_scan'
print(x, max_show = 10L, ...)
```

Arguments

<code>x</code>	An object of class "circular_scan".
<code>max_show</code>	Integer. Maximum number of region IDs to display in full before truncating with "... and N more". The default of 10L keeps console output compact for large clusters; set <code>max_show = -1L</code> (or any value larger than the cluster size) to print every region. Mirrors the convention used by tibble for printing wide tables.
<code>...</code>	Further arguments passed to or from other methods.

Value

Invisibly returns `x` (the input object of class "circular_scan"), unchanged. Called for the side effect of printing a human-readable summary of the scan result to the console: the total cases and population, the number of regions scanned, the number of Monte Carlo simulations, and the contents of the most likely cluster (region IDs, cases, expected count, population, relative risk, log-likelihood ratio, and p-value).

```
print.sequential_scan
```

Print Method for sequential_scan Objects

Description

Print Method for sequential_scan Objects

Usage

```
## S3 method for class 'sequential_scan'
print(x, max_show = 10L, ...)
```

Arguments

x	An object of class "sequential_scan".
max_show	Integer. Maximum number of region IDs (or leaf IDs, for tree-only scans) to display in full per iteration before truncating with "... and N more". Default 10L; set max_show = -1L to print every value.
...	Further arguments passed to or from other methods.

Value

Invisibly returns x (the input object of class "sequential_scan"), unchanged. Called for the side effect of printing a human-readable summary of the sequential scan result to the console: the scan type, the buffer size, the number of iterations performed, the significance threshold, the number of significant clusters, and a per-iteration table of detected clusters.

```
print.treespatial_scan
```

Print Method for treespatial_scan Objects

Description

Print Method for treespatial_scan Objects

Usage

```
## S3 method for class 'treespatial_scan'
print(x, max_show = 10L, ...)
```

Arguments

x	An object of class "treespatial_scan".
max_show	Integer. Maximum number of leaf IDs and region IDs to display in full before truncating with "... and N more". The default of 10L keeps console output compact when the most likely cluster spans many leaves (for example, when the root node maximizes the likelihood ratio for an aggregated denominator) or many regions; set max_show = -1L to print every value. Mirrors the convention used by tibble .
...	Further arguments passed to or from other methods.

Value

Invisibly returns x (the input object of class "treespatial_scan"), unchanged. Called for the side effect of printing a human-readable summary of the scan result to the console: the total population, the number of regions and tree nodes, the number of Monte Carlo simulations, and the contents of the most likely cluster (tree node, leaf IDs, region IDs, cases, expected count, population, relative risk, log-likelihood ratio, and p-value).

print.tree_scan	<i>Print Method for tree_scan Objects</i>
-----------------	---

Description

Print Method for tree_scan Objects

Usage

```
## S3 method for class 'tree_scan'
print(x, max_show = 10L, ...)
```

Arguments

x	An object of class "tree_scan".
max_show	Integer. Maximum number of leaf IDs to display in full before truncating with "... and N more". The default of 10L keeps console output compact when a node spans hundreds or thousands of leaves (such as the root of a deep tree); set max_show = -1L to print every leaf.
...	Further arguments passed to or from other methods.

Value

Invisibly returns x (the input object of class "tree_scan"), unchanged. Called for the side effect of printing a human-readable summary of the scan result to the console: the total cases and population, the number of tree nodes, the number of Monte Carlo simulations, the contents of the most likely cluster (node ID, leaf IDs, cases, expected count, log-likelihood ratio, and p-value), and the top significant cuts at the chosen significance threshold.

pvalue	<i>Extract the Monte Carlo P-Value from a Scan Result</i>
--------	---

Description

Returns the Monte Carlo p-value(s) of a scan result without requiring the user to access the internal list structure.

Usage

```
pvalue(x, ...)

## S3 method for class 'treess'
pvalue(x, ...)

## S3 method for class 'sequential_scan'
pvalue(x, ...)
```

Arguments

`x` A scan result object (see [most_likely_cluster](#)).
`...` Further arguments passed to methods (currently unused).

Value

For `circular_scan`, `tree_scan`, and `treespatial_scan`, a single numeric value: the Monte Carlo p-value of the most likely cluster. For `sequential_scan`, a named numeric vector with one p-value per iteration.

See Also

[most_likely_cluster](#)

Examples

```
ex <- generate_example_data(n_regions = 16, seed = 1)
dat <- data.frame(cases = ex$cases, population = ex$population,
                 region_id = ex$region_id, x = ex$x, y = ex$y,
                 node_id = ex$node_id)
res <- treespatial_scan(dat, cases = cases,
                      population = population,
                      region_id = region_id, x = x, y = y,
                      node_id = node_id, tree = ex$tree,
                      nsim = 19, seed = 42)

pvalue(res)
```

rj_map

Rio de Janeiro Municipal Boundaries

Description

Polygon boundaries for the 92 municipalities of Rio de Janeiro state, used to map clusters from the `rj_mortality` analysis.

Usage

```
rj_map
```

Format

An sf data.frame with 92 rows and 3 columns:

ibge_code 6-digit IBGE municipality code; join key to `rj_mortality$ibge_code`.

NAME Municipality name.

geometry MULTIPOLYGON geometry in WGS84 (EPSG:4326).

Source

Municipal boundaries (Malhas Municipais) from the Brazilian Institute of Geography and Statistics (IBGE).

See Also

[rj_mortality](#)

Examples

```
data(rj_map)
cat("Municipalities:", nrow(rj_map), "\n")
```

<code>rj_mortality</code>	<i>Rio de Janeiro Infant Mortality (2016)</i>
---------------------------	---

Description

Combined long-format dataset of infant mortality in the 92 municipalities of Rio de Janeiro state, Brazil, 2016. Each row represents a (municipality, ICD-10 leaf) combination with at least one death; missing combinations are implicitly zero. Use together with [rj_tree](#).

Usage

```
rj_mortality
```

Format

- A data.frame in long format with columns:
- region_id** Integer identifier of the municipality.
 - ibge_code** 6-digit IBGE municipality code.
 - name** Municipality name.
 - population** IBGE municipal population estimate for 2016.
 - live_births** Number of live births in 2016 (SINASC). Used as the population denominator in Section 5.2 of Cancado et al. (2025).
 - x** Longitude of the municipality centroid.
 - y** Latitude of the municipality centroid.
 - node_id** Character ICD-10 leaf code (4 characters), matching a leaf of [rj_tree](#).
 - cases** Integer count of infant deaths.

Details

To reproduce Section 5.2 of Cancado et al. (2025), use `live_births` as the population denominator:

```
data <- rj_mortality
data$population <- data$live_births
treespatial_scan(data, cases, population, region_id, x, y, node_id,
                  tree = rj_tree, ...)
```

Municipality polygons are provided as the `rj_map` dataset.

Source

Population: IBGE municipal estimates. Live births: DATASUS/SINASC via TabNet. Deaths: DATASUS/SIM microdata via OpenDATASUS (<https://opendatasus.saude.gov.br>). Centroids: geobr.

References

Cancado, A.L.F., Oliveira, G.S., Quadros, A.V.C., Duczmal, L. (2025). A tree-spatial scan statistic. *Environmental and Ecological Statistics*, 32, 953–978. doi:10.1007/s1065102500670w

See Also

[rj_tree](#)

Examples

```
data(rj_mortality)
head(rj_mortality)
cat("Total deaths:", sum(rj_mortality$cases), "\n")
```

rj_tree

Rio de Janeiro Infant Mortality - ICD-10 Tree

Description

ICD-10 hierarchy used by the `rj_mortality` dataset. Three levels: Chapter -> 3-character category -> 4-character subcategory.

Usage

```
rj_tree
```

Format

A data.frame with 622 rows and 2 columns:

node_id Character identifier of each ICD-10 node.

parent_id Character identifier of the parent node. NA for the root.

Details

The tree has 410 leaves (4-character ICD-10 codes) and 622 total nodes.

Source

Constructed from the leaf-level codes in [rj_mortality](#), which in turn come from the Brazilian Mortality Information System (SIM) maintained by DATASUS. See `data-raw/` in the GitHub repository for the build script.

See Also

[rj_mortality](#)

Examples

```
data(rj_tree)
head(rj_tree)
```

secondary_clusters	<i>Extract Secondary Clusters from a Scan Result</i>
--------------------	--

Description

Returns the table of secondary cluster candidates stored in a scan result. Note that these are unfiltered candidates: overlapping candidates are not removed and, for `circular_scan` and `treespacial_scan`, per-candidate p-values are not attached. Use [filter_clusters](#) to obtain distinct significant clusters.

Usage

```
secondary_clusters(x, ...)

## S3 method for class 'treess'
secondary_clusters(x, ...)

## S3 method for class 'tree_scan'
secondary_clusters(x, ...)

## S3 method for class 'sequential_scan'
secondary_clusters(x, ...)
```

Arguments

<code>x</code>	A scan result object (see most_likely_cluster).
<code>...</code>	Further arguments passed to methods (currently unused).

Value

A data.frame. For circular_scan and treespatial_scan, the top secondary candidates ordered by decreasing log-likelihood ratio. For tree_scan, the table of significant cuts at the alpha used in the original call. For sequential_scan, the per-iteration cluster table beyond the first iteration (the first iteration is the most likely cluster; see [most_likely_cluster](#)).

See Also

[filter_clusters](#), [most_likely_cluster](#)

Examples

```
ex <- generate_example_data(n_regions = 16, seed = 1)
dat <- data.frame(cases = ex$cases, population = ex$population,
                 region_id = ex$region_id, x = ex$x, y = ex$y,
                 node_id = ex$node_id)
res <- treespatial_scan(dat, cases = cases,
                      population = population,
                      region_id = region_id, x = x, y = y,
                      node_id = node_id, tree = ex$tree,
                      nsim = 19, seed = 42)
head(secondary_clusters(res))
```

sequential_scan

Sequential Scan for Secondary Clusters

Description

Implements the sequential adjustment of Zhang, Assuncao and Kulldorff (2010) for detecting secondary clusters. After the most likely cluster (MLC) is detected and found significant, the regions composing it (optionally together with a buffer of nearest-neighbor regions) are *removed* from the dataset – treated as a “lake” with no population and no cases – and the scan is re-run on the reduced data with a fresh Monte Carlo simulation. The procedure is iterated until the MLC of the current reduced data is no longer significant, or max_iter is reached.

Usage

```
sequential_scan(
  data,
  cases,
  population,
  region_id = NULL,
  x = NULL,
  y = NULL,
  node_id = NULL,
  tree = NULL,
  tree_node_id = NULL,
```

```

    tree_parent_id = NULL,
    max_iter = 5L,
    alpha = 0.05,
    nsim = 999L,
    max_pop_pct = 0.5,
    buffer_size = 0L,
    model = c("poisson", "binomial"),
    seed = NULL,
    verbose = TRUE,
    n_cores = 1L
)

```

Arguments

data	A data.frame whose columns are named by the cases, population and (depending on scan type) region_id, x, y, node_id arguments.
cases	Column of data giving case counts. Given as an unquoted column name (a string or expression also works).
population	Column of data giving the denominator.
region_id, x, y	Columns of data (omit for tree-only scan).
node_id	Column of data (omit for circular scan).
tree	Tree as a 2-column data.frame (node_id, parent_id), or use tree_node_id/tree_parent_id. Omit for circular scan.
tree_node_id, tree_parent_id	Optional. Parallel vectors as an alternative to tree.
max_iter	Integer. Maximum number of iterations (including the primary MLC). Default 5.
alpha	Numeric. Significance level. Iteration stops when the MLC p-value of the current reduced data exceeds alpha. Default 0.05.
nsim	Integer. Monte Carlo replications per iteration. Default 999.
max_pop_pct	Numeric. Maximum zone-size constraint, passed to the inner scans. Default 0.5.
buffer_size	Integer. Number of nearest-neighbor regions to remove together with each detected cluster, computed by Euclidean distance from each cluster region to the remaining ones. Default 0. Zhang et al. (2010) report that the type I error and power are essentially insensitive to the buffer size and that buffer_size = 0 is generally adequate. Ignored for tree-only scans.
model	Character. "poisson" (default) or "binomial".
seed	Integer or NULL. Seed for the first iteration; subsequent iterations advance the RNG state to avoid using the same seed across iterations.
verbose	Logical. Print progress at each iteration.
n_cores	Integer. OpenMP threads passed to inner scans.

Details

This differs from [filter_clusters](#), which scans the single-pass candidate pool for non-overlapping clusters (the original Cancado et al. 2025 criterion). The sequential approach computes a fresh, un-conservative p-value for each secondary cluster, conditional on the prior detections.

Inputs are supplied through a single data `data.frame`; the relevant columns are referenced by name. Which columns are supplied selects one of three scan types:

- **tree-spatial**: all of cases, population, region_id, x, y, node_id, tree are required (data in long format, one row per (region, leaf));
- **circular**: cases, population, region_id, x, y (no node_id / tree; one row per region);
- **tree-only**: cases, population, node_id, tree (no spatial inputs). `buffer_size` is ignored for the tree-only case.

Value

An object of class "sequential_scan" with components:

clusters A `data.frame` with one row per iteration performed, with columns `iteration`, `node_id` (only for tree-spatial / tree-only scans), `n_regions` (only for scans with a spatial component), `cases`, `expected`, `population`, `rr` (only for scans with a spatial component), `llr`, `pvalue`, and a logical significant flag (`pvalue <= alpha`). The list- columns `region_ids` and/or `leaf_ids` carry the cluster composition.

iterations A list with the full scan result of each iteration.

regions, tree, alpha, nsim, buffer_size, n_iter, scan_type Bookkeeping fields. `n_iter` is the number of iterations actually performed.

References

- Zhang, Z., Assuncao, R., & Kulldorff, M. (2010). Spatial scan statistics adjusted for multiple clusters. *Journal of Probability and Statistics*, 2010, 642379.
- Cancado, A. L. F., Oliveira, G. S., Quadros, A. V. C., & Duczmal, L. H. (2025). A tree-spatial scan statistic. *Environmental and Ecological Statistics*, 32, 953-978. doi:[10.1007/s1065102500670](https://doi.org/10.1007/s1065102500670)
- Kulldorff, M. (1997). A spatial scan statistic. *Communications in Statistics - Theory and Methods*, 26(6), 1481-1496.

See Also

[treespatial_scan](#), [circular_scan](#), [tree_scan](#), [filter_clusters](#), [get_cluster_regions](#)

Examples

```
data(london_collisions); data(london_tree)
result <- sequential_scan(
  london_collisions,
  cases      = cases,
  population = population,
  region_id  = region_id,
  x          = x,
```

```

    y          = y,
    node_id    = node_id,
    tree       = london_tree,
    max_iter = 3, nsim = 99, seed = 42
  )
  print(result)

```

summary.circular_scan *Summary Method for circular_scan Objects*

Description

Prints the same fields as `print.circular_scan()`, followed by a numeric summary of the simulated log-likelihood-ratio distribution under the null. Useful for checking that the Monte Carlo replicates produced a reasonable spread relative to the observed test statistic.

Usage

```

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

```

Arguments

<code>object</code>	An object of class "circular_scan".
<code>...</code>	Further arguments passed to <code>print.circular_scan()</code> . In particular, <code>max_show</code> controls how many region IDs of the most-likely cluster are printed in full before truncating with "... and N more"; see print.circular_scan for details.

Value

Invisibly returns `object` (the input object of class "circular_scan"), unchanged. Called for the side effect of printing the same fields as `print.circular_scan()`, followed by a numeric summary (min, quartiles, median, mean, max) of the simulated log-likelihood-ratio distribution under the null.

summary.sequential_scan
Summary Method for sequential_scan Objects

Description

Summary Method for sequential_scan Objects

Usage

```

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

```

Arguments

object	An object of class "sequential_scan".
...	Passed to print.sequential_scan().

Value

Invisibly returns object. Called for the side effect of printing the same fields as print.sequential_scan() followed by, for each iteration, the cluster composition (region IDs or leaf IDs).

summary.treespatial_scan

Summary Method for treespatial_scan Objects

Description

Prints the same fields as print.treespatial_scan(), followed by the top-10 branches by total case count and a numeric summary of the simulated log-likelihood-ratio distribution under the null. Useful for diagnosing how heavily one branch of the tree dominates the dataset and whether the observed maximum LR is a clear outlier against the simulated null.

Usage

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

Arguments

object	An object of class "treespatial_scan".
...	Further arguments passed to print.treespatial_scan(). In particular, max_show controls how many leaf IDs and region IDs of the most-likely cluster are printed in full before truncating with "... and N more"; see print.treespatial_scan for details.

Value

Invisibly returns object (the input object of class "treespatial_scan"), unchanged. Called for the side effect of printing the same fields as print.treespatial_scan(), followed by the top-10 branches by total case count and a numeric summary (min, quartiles, median, mean, max) of the simulated log-likelihood-ratio distribution under the null.

summary.tree_scan	<i>Summary Method for tree_scan Objects</i>
-------------------	---

Description

Prints the same fields as `print.tree_scan()`, followed by the full table of all candidate cuts (not just significant ones) ordered by decreasing log-likelihood ratio.

Usage

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

Arguments

object	An object of class "tree_scan".
...	Further arguments passed to <code>print.tree_scan()</code> . In particular, <code>max_show</code> controls how many leaf IDs of the most-likely cluster are printed in full before truncating with "... and N more"; see print.tree_scan for details.

Value

Invisibly returns object (the input object of class "tree_scan"), unchanged. Called for the side effect of printing the same fields as `print.tree_scan()`, followed by the full table of all candidate cuts (not just significant ones) ordered by decreasing log-likelihood ratio.

treespatial_scan	<i>Tree-Spatial Scan Statistic</i>
------------------	------------------------------------

Description

Performs the tree-spatial scan statistic, combining Kulldorff's circular scan (spatial clusters) with the tree-based scan (hierarchical data mining). Searches all combinations of spatial zones and tree branches to identify pairs (z, g) with significantly more cases than expected.

Usage

```
treespatial_scan(
  data,
  cases,
  population,
  region_id,
  x,
  y,
  node_id,
```

```

    tree = NULL,
    tree_node_id = NULL,
    tree_parent_id = NULL,
    max_pop_pct = 0.5,
    nsim = 999L,
    alpha = 0.05,
    model = c("poisson", "binomial"),
    seed = NULL,
    n_cores = 1L
  )

```

Arguments

data	A data.frame in long format with one row per (region, tree-leaf) observation. The cases, population, region_id, x, y and node_id arguments name columns of this data.frame.
cases	Column of data giving the number of cases observed for each (region, leaf) row. Given as an unquoted column name (a string or an expression on columns also works).
population	Column of data giving the population (or denominator) of the region for each row. The same value should be repeated across all rows of a given region; if it varies, the first occurrence per region is used and a warning is issued.
region_id	Column of data giving the region identifier.
x, y	Columns of data giving the region centroid coordinates. Like population, these should be constant within region.
node_id	Column of data giving the tree leaf identifier for each row. Each value must match a leaf of the tree.
tree	A data.frame with columns node_id and parent_id. The root node(s) must have parent_id = NA. As an alternative, pass tree_node_id and tree_parent_id as parallel vectors instead of this argument.
tree_node_id, tree_parent_id	Optional. Parallel vectors describing the tree edges, used as an alternative to tree. If both tree and these vectors are supplied, an error is raised.
max_pop_pct	Numeric. Maximum proportion of total population allowed inside a zone. Default 0.5.
nsim	Integer. Number of Monte Carlo simulations. Default 999.
alpha	Numeric. Significance level. Default 0.05.
model	Character. "poisson" (default) or "binomial".
seed	Integer or NULL. Random seed for the Monte Carlo loop. When non-NULL, the user's pre-existing RNG state is saved on entry and restored on exit, so the seed argument affects only this call and does not leak into subsequent draws in the user's session.
n_cores	Integer. OpenMP threads for the Monte Carlo loop. Default 1L (serial).

Details

Inputs are supplied through a single data `data.frame` in long format (one row per (region, tree-leaf) observation); the case, population, region, coordinate and leaf columns are referenced by name. The user is responsible for choosing which column to use as population (e.g., total population, live births, person-years), making the choice of denominator explicit.

Secondary clusters. The returned object contains the most likely cluster as well as the full set of evaluated (zone, branch) pairs in `secondary_clusters`. To obtain the distinct secondary clusters as described in Section 5.1.1 of Cancado et al. (2025) (filtering out pairs that overlap in regions or branches with already-retained clusters), use `filter_clusters` or `get_cluster_regions` with `n_clusters > 1`.

Value

An object of class "treespatial_scan".

References

Cancado, A. L. F., Oliveira, G. S., Quadros, A. V. C., & Duczmal, L. (2025). A tree-spatial scan statistic. *Environmental and Ecological Statistics*, 32, 953–978. doi:[10.1007/s10651-025-00670-w](https://doi.org/10.1007/s10651-025-00670-w)

See Also

[circular_scan](#), [tree_scan](#), [aggregate_tree](#), [filter_clusters](#), [get_cluster_regions](#), [sequential_scan](#)

Examples

```
set.seed(123)
n_regions <- 10
tree <- data.frame(
  node_id = c(1, 2, 3, 4, 5, 6, 7),
  parent_id = c(NA, 1, 1, 2, 2, 3, 3)
)
# Long-format data.frame: one row per (region, leaf) combination
crimes <- expand.grid(region_id = 1:n_regions, node_id = c(4, 5, 6, 7))
xy <- data.frame(region_id = 1:n_regions,
  x = runif(n_regions, 0, 10),
  y = runif(n_regions, 0, 10))
crimes <- merge(crimes, xy, by = "region_id")
crimes$population <- 1000
crimes$cases <- rpois(nrow(crimes), lambda = 5)
crimes$cases[crimes$node_id == 4 & crimes$region_id %in% 1:3] <- rpois(3, 30)

result <- treespatial_scan(
  crimes,
  cases = cases,
  population = population,
  region_id = region_id,
  x = x,
  y = y,
  node_id = node_id,
  tree = tree,
```

```

      nsim      = 99
    )
    print(result)

```

tree_scan

Tree-Based Scan Statistic

Description

Performs the tree-based scan statistic for detecting clusters in hierarchical data. Uses a Poisson or binomial model with Monte Carlo simulation (implemented in C++ via Rcpp) for significance testing.

Usage

```

tree_scan(
  data,
  cases,
  node_id,
  tree = NULL,
  population = NULL,
  tree_node_id = NULL,
  tree_parent_id = NULL,
  nsim = 999L,
  alpha = 0.05,
  model = c("poisson", "binomial"),
  seed = NULL,
  n_cores = 1L
)

```

Arguments

data	A data.frame with one row per leaf node. The cases, node_id and (optional) population arguments name columns of this data.frame.
cases	Column of data giving case counts at the leaf level. Given as an unquoted column name (a string or expression also works).
node_id	Column of data giving the leaf identifier for each row. Each value must match a leaf of the tree.
tree	A data.frame with columns node_id and parent_id. Root node(s) must have parent_id = NA. Alternatively, pass the tree as parallel vectors via tree_node_id and tree_parent_id.
population	Optional column of data giving the population at the leaf level. For the binomial model, population is the number of trials (cases + controls) per leaf and is required. For the Poisson model, defaults to 1 per leaf when omitted.

tree_node_id, tree_parent_id	Optional parallel vectors describing the tree as an alternative to the tree data.frame. Both must have the same length, and the root node(s) must have tree_parent_id = NA. Ignored when tree is supplied.
nsim	Integer. Number of Monte Carlo simulations. Default is 999.
alpha	Numeric. Significance level. Default is 0.05.
model	Character. Likelihood model: either "poisson" (default) or "binomial".
seed	Integer or NULL. Random seed for the Monte Carlo loop. When non-NULL, the user's pre-existing RNG state is saved on entry and restored on exit, so the seed argument affects only this call and does not leak into subsequent draws in the user's session.
n_cores	Integer. Number of OpenMP threads for the Monte Carlo loop. Default is 1L (serial). Set higher to parallelize.

Details

Inputs are supplied through a single data data.frame with one row per leaf (rows are matched to the tree by node_id, so they need not be pre-ordered); counts are summed within leaf.

Value

An object of class "tree_scan" (see package help for details).

References

Kulldorff, M., Fang, Z., & Walsh, S. J. (2003). A tree-based scan statistic for database disease surveillance. *Biometrics*, 59(2), 323–331.

See Also

[circular_scan](#), [treespatial_scan](#), [aggregate_tree](#)

Examples

```
tree <- data.frame(
  node_id = c(1, 2, 3, 4, 5, 6, 7, 8),
  parent_id = c(NA, 1, 1, 2, 2, 3, 3, 3)
)
# One row per leaf (leaves are 4, 5, 6, 7, 8)
leaf_data <- data.frame(
  node_id = c(4, 5, 6, 7, 8),
  cases = c(50, 5, 3, 2, 4),
  pop = c(100, 100, 100, 100, 100)
)

result <- tree_scan(leaf_data, cases = cases, node_id = node_id,
  tree = tree, population = pop, nsim = 99)
print(result)
```

Index

* datasets

- chicago_crimes, [6](#)
- chicago_map, [8](#)
- chicago_tree, [8](#)
- fl_deaths, [13](#)
- london_boroughs_map, [17](#)
- london_collisions, [18](#)
- london_tree, [19](#)
- rj_map, [27](#)
- rj_mortality, [28](#)
- rj_tree, [29](#)

aggregate_tree, [3](#), [4](#), [38](#), [40](#)

build_zones, [3](#), [5](#)

chicago_crimes, [6](#), [8](#), [9](#)

chicago_map, [7](#), [8](#)

chicago_tree, [6](#), [7](#), [8](#)

circular_scan, [3](#), [9](#), [11](#), [12](#), [16](#), [33](#), [38](#), [40](#)

dotchart, [23](#)

filter_clusters, [3](#), [10](#), [11](#), [16](#), [21](#), [22](#), [30](#),
[31](#), [33](#), [38](#)

fl_deaths, [13](#)

generate_example_data, [14](#)

get_cluster_regions, [3](#), [10](#), [15](#), [17](#), [20](#), [22](#),
[33](#), [38](#)

london_boroughs_map, [17](#), [18](#)

london_collisions, [17](#), [18](#), [19](#)

london_tree, [18](#), [19](#)

most_likely_cluster, [20](#), [27](#), [30](#), [31](#)

plot, [22](#)

plot.tree_scan, [23](#)

plot.treess, [21](#), [23](#)

print.circular_scan, [24](#), [34](#)

print.sequential_scan, [24](#)

print.tree_scan, [26](#), [36](#)

print.treespatial_scan, [25](#), [35](#)

pvalue, [20](#), [26](#)

rj_map, [27](#)

rj_mortality, [28](#), [28](#), [29](#), [30](#)

rj_tree, [28](#), [29](#), [29](#)

secondary_clusters, [20](#), [30](#)

sequential_scan, [3](#), [10](#), [12](#), [16](#), [22](#), [31](#), [38](#)

summary.circular_scan, [34](#)

summary.sequential_scan, [34](#)

summary.tree_scan, [36](#)

summary.treespatial_scan, [35](#)

tree_scan, [3](#), [10–12](#), [23](#), [33](#), [38](#), [39](#)

treespatial_scan, [3](#), [10–16](#), [33](#), [36](#), [40](#)

treeSS (treeSS-package), [3](#)

treeSS-package, [3](#)