

# Package ‘treeSS’

July 2, 2026

**Type** Package

**Title** Tree-Spatial Scan Statistic for Cluster Detection

**Version** 0.2.4

**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

**SystemRequirements** GNU make

**Encoding** UTF-8

**LazyData** true

**LazyDataCompression** xz

**VignetteBuilder** knitr

**RoxygenNote** 7.3.2

**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>>),  
 Geiziane S. Oliveira [aut],  
 Luiz H. Duczmal [aut]

**Maintainer** Allan Quadros <[allanvcq@gmail.com](mailto:allanvcq@gmail.com)>

**Repository** CRAN

**Date/Publication** 2026-07-02 07:00:02 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 . . . . .                | 12 |
| generate_example_data . . . . .    | 14 |
| get_cluster_regions . . . . .      | 15 |
| london_boroughs_map . . . . .      | 17 |
| london_collisions . . . . .        | 18 |
| london_tree . . . . .              | 19 |
| print.circular_scan . . . . .      | 20 |
| print.sequential_scan . . . . .    | 20 |
| print.treespatial_scan . . . . .   | 21 |
| print.tree_scan . . . . .          | 22 |
| rj_map . . . . .                   | 22 |
| rj_mortality . . . . .             | 23 |
| rj_tree . . . . .                  | 24 |
| sequential_scan . . . . .          | 25 |
| summary.circular_scan . . . . .    | 28 |
| summary.sequential_scan . . . . .  | 28 |
| summary.treespatial_scan . . . . . | 29 |
| summary.tree_scan . . . . .        | 29 |
| treespatial_scan . . . . .         | 30 |
| tree_scan . . . . .                | 32 |

**Index** **35**

---

treeSS-package

*treeSS: Tree-Spatial Scan Statistic for Cluster Detection*

---

## 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:

- Andre L. F. Cançado ([ORCID](#))
- Geiziane S. Oliveira
- Luiz H. Duczmal

## 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

|       |  |
|-------|--|
| data  | A data.frame in long format with one row per (region, leaf) observation. The cases, region_id and node_id arguments name columns of this data.frame. |
| cases | Column of data giving case counts. Given as an unquoted column name (a string or expression also works).   |

region\_id      Column of data giving the region identifier.  
 node\_id        Column of data giving the leaf identifier.  
 tree            A data.frame with columns node\_id and parent\_id. As an alternative, pass  
                  tree\_node\_id and tree\_parent\_id.  
 tree\_node\_id, tree\_parent\_id  
                  Optional. Parallel vectors as an alternative to tree.

### 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](https://doi.org/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

*Build Candidate Spatial Zones Using Circular Windows*


---

### 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

*Filter Overlapping Secondary Clusters*


---

### Description

Removes overlapping secondary clusters from the output of [treespatial\\_scan](#), [circular\\_scan](#), or [tree\\_scan](#).

### Usage

```
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. |

### Details

For [treespatial\\_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**

[treespatial\\_scan](#), [circular\\_scan](#), [tree\\_scan](#), [sequential\\_scan](#)

**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
)
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 `treespacial_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")
treespacial_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-cd10.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 [treespatial\\_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     *Extract Cluster Membership for Each Region*

---

### 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 "treespatial_scan" or "circular_scan" object (single-pass scan), or a "sequential_scan" object (Zhang, Assuncao and Kulldorff, 2010).   |
| n_clusters | 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). |
| overlap    | Logical. If TRUE (default), returns a facet-ready data.frame: all regions are replicated for each cluster panel, with a panel column for faceting and cluster marked only for member regions. If FALSE, returns one row per region assigned to its highest-ranked cluster (for single-panel maps).              |
| ...        | 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     *London Borough Boundaries*

---

## 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      *London Road Collisions (2022)*

---

### 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 | <i>London Road Collisions - Hierarchy</i> |
|-------------|---|

---

## 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)
```

---

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

|          |  |
|----------|--|
| x        | An object of class "circular_scan".  |
| max_show | 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 max_show = -1L (or any value larger than the cluster size) to print every region. Mirrors the convention used by <b>tibble</b> for printing wide tables. |
| ...      | 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 maximises the likelihood ratio for an aggregated denominator) or many regions; set max_show = -1L to print every value. Mirrors the convention used by <b>tibble</b> . |
| ...      | 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            *Print Method for tree\_scan Objects*

---

### 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.

---

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")
```

---

|              |   |
|--------------|---|
| rj_mortality | <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)
```

---

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-neighbour 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

|   |   |
|---|---|
| <code>data</code>                         | A data.frame whose columns are named by the cases, population and (depending on scan type) <code>region_id</code> , <code>x</code> , <code>y</code> , <code>node_id</code> arguments.   |
| <code>cases</code>                        | Column of data giving case counts. Given as an unquoted column name (a string or expression also works).  |
| <code>population</code>                   | Column of data giving the denominator.  |
| <code>region_id, x, y</code>              | Columns of data (omit for tree-only scan).  |
| <code>node_id</code>                      | Column of data (omit for circular scan).  |
| <code>tree</code>                         | Tree as a 2-column data.frame ( <code>node_id</code> , <code>parent_id</code> ), or use <code>tree_node_id/tree_parent_id</code> . Omit for circular scan.  |
| <code>tree_node_id, tree_parent_id</code> | Optional. Parallel vectors as an alternative to <code>tree</code> .   |
| <code>max_iter</code>                     | Integer. Maximum number of iterations (including the primary MLC). Default 5.   |
| <code>alpha</code>                        | Numeric. Significance level. Iteration stops when the MLC p-value of the current reduced data exceeds <code>alpha</code> . Default 0.05.  |
| <code>nsim</code>                         | Integer. Monte Carlo replications per iteration. Default 999.   |
| <code>max_pop_pct</code>                  | Numeric. Maximum zone-size constraint, passed to the inner scans. Default 0.5.  |
| <code>buffer_size</code>                  | Integer. Number of nearest-neighbour 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 <code>buffer_size = 0</code> is generally adequate. Ignored for tree-only scans. |
| <code>model</code>                        | Character. "poisson" (default) or "binomial".   |
| <code>seed</code>                         | Integer or NULL. Seed for the first iteration; subsequent iterations advance the RNG state to avoid using the same seed across iterations.  |
| <code>verbose</code>                      | Logical. Print progress at each iteration.  |
| <code>n_cores</code>                      | 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/s1065102500670w

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

|        |   |
|--------|---|
| object | An object of class "circular_scan".   |
| ...    | 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 <a href="#">print.circular_scan</a> 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 <code>print.sequential_scan()</code> . |

**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**

|                     |  |
|---------------------|--|
| <code>object</code> | An object of class "treespatial_scan".   |
| <code>...</code>    | Further arguments passed to <code>print.treespatial_scan()</code> . In particular, <code>max_show</code> controls how many leaf IDs and region IDs of the most-likely cluster are printed in full before truncating with "... and N more"; see <a href="#">print.treespatial_scan</a> 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
```

*Summary Method for tree\_scan Objects*

---

**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 `print.tree_scan()`. In particular, `max_show` 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/s1065102500670w

**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 Repp) 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, 12
- london\_boroughs\_map, 17
- london\_collisions, 18
- london\_tree, 19
- rj\_map, 22
- rj\_mortality, 23
- rj\_tree, 24

- sequential\_scan, 3, 10–12, 16, 25, 32

- summary.circular\_scan, 28

- summary.sequential\_scan, 28

- summary.tree\_scan, 29

- summary.treespatial\_scan, 29

- tree\_scan, 3, 10–12, 27, 32, 32

- treespatial\_scan, 3, 10–14, 16, 27, 30, 34

- treeSS (treeSS-package), 3

- treeSS-package, 3

- aggregate\_tree, 3, 4, 32, 34

- 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, 27, 32, 34

- filter\_clusters, 3, 10, 11, 15, 16, 26, 27, 31, 32

- fl\_deaths, 12

- generate\_example\_data, 14

- get\_cluster\_regions, 3, 10, 15, 17, 27, 31, 32

- london\_boroughs\_map, 17, 18

- london\_collisions, 17, 18, 19

- london\_tree, 18, 19

- print.circular\_scan, 20, 28

- print.sequential\_scan, 20

- print.tree\_scan, 22, 30

- print.treespatial\_scan, 21, 29

- rj\_map, 22

- rj\_mortality, 23, 23, 24, 25

- rj\_tree, 23, 24, 24