Package: areaOfEffect (via r-universe)

May 29, 2026

Title Classify Points by Distance to Polygon Boundaries

Version 0.2.5

Description Classifies spatial points relative to polygon boundaries, labeling each point as ``core" (inside), ``halo" (in a buffer zone), or pruning it (outside both). Handles projection, buffering, and point-in-polygon operations automatically. The default buffer produces equal core and halo areas, providing a scale-independent definition of ``near the boundary." An optional mask clips the buffer to relevant areas such as coastlines.

License MIT + file LICENSE

URL https://gillescolling.com/areaOfEffect/, https://github.com/gcol33/areaOfEffect

BugReports https://github.com/gcol33/areaOfEffect/issues

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports methods, sf

Suggests ggplot2, lwgeom, testthat (>= 3.0.0), knitr, rmarkdown, svglite

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 3.5)

LazyData true

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://gcol33.r-universe.dev

Date/Publication 2026-05-29 11:16:31 UTC

RemoteUrl https://github.com/gcol33/areaofeffect

RemoteRef HEAD

RemoteSha 1b0a40968c42d93917036483b25d1817dcc67e25

Contents

`[.aoe_result` *Subset method for aoe_result*

Description

Preserves aoe_result attributes when subsetting.

Usage

```
## S3 method for class 'aoe_result'
x[i, ...]
```

Arguments

`x`	An aoe_result object
`i`	Row indices
`...`	Additional arguments passed to sf subsetting

Value

An aoe_result object (or sf if support_id removed)

---

aoe *Classify and Prune Points by Area of Effect*

Description

Given a set of points and one or more support polygons, `aoe()` classifies points as "core" (inside original support) or "halo" (inside the area of effect but outside original support), pruning all points outside.

Usage

```
aoe(
  points,
  support = NULL,
  scale = NULL,
  area = NULL,
  method = c("buffer", "stamp"),
  reference = NULL,
  mask = NULL,
  largest_polygon = TRUE,
  coords = NULL
)
```

Arguments

`points`	An `sf` object with POINT geometries.
`support`	One of: • `sf` object with POLYGON/MULTIPOLYGON geometries • Country name or ISO code: `"France"`, `"FR"`, `"FRA"` • Vector of countries: `c("France", "Germany")` • Missing: auto-detects countries containing the points
`scale`	Numeric scale factor (default `sqrt(2) - 1`, approximately 0.414). Controls the size of the halo relative to the core: • `sqrt(2) - 1` (default): equal core/halo areas, ratio 1:1 • `1`: area ratio 1:3 (halo is 3x core area) For `method = "buffer"`, determines the target halo area as `original_area * ((1 + scale)^2 - 1)`. For `method = "stamp"`, the multiplier `1 + scale` is applied to distances from the reference point. Cannot be used together with `area`.

`area` Numeric area proportion (alternative to `scale`). Specifies the target halo area as a proportion of the original support area. For example, `area = 1` means halo area equals the original support area.

Unlike `scale`, this parameter accounts for masking: the function finds the scale that produces the target halo area *after* mask intersection. This is useful when you need a specific effective area regardless of how much gets clipped by coastlines or borders.

Cannot be used together with `scale`.

`method` Method for computing the area of effect:

- `"buffer"` (default): Uniform buffer around the support boundary. Robust for any polygon shape. Buffer distance is calculated to achieve the target halo area.
- `"stamp"`: Scale vertices outward from the centroid (or reference point). Preserves shape proportions but only guarantees containment for star-shaped polygons. May leave small gaps for highly concave shapes.

`reference` Optional `sf` object with a single POINT geometry.

If `NULL` (default), the centroid of each support is used. Only valid when `support` has a single row and `method = "stamp"`.

`mask` Optional mask for clipping the area of effect. Can be:

- `sf` object with POLYGON or MULTIPOLYGON geometry
- `"land"`: use the bundled global land mask to exclude sea areas If provided, each area of effect is intersected with this mask.

`largest_polygon`

Logical (default `TRUE`). When the support contains multiple polygons (e.g., mainland plus islands), use only the largest polygon by area. This is typically the mainland. Points near dropped polygons will be pruned entirely (not classified). Set to `FALSE` to include all polygons, in which case `area = "equal"` uses total area with redistribution across all polygons.

`coords` Column names for coordinates when `points` is a data.frame, e.g. `c("lon", "lat")`. If `NULL`, auto-detects common names.

Details

By default, the area of effect is computed using a buffer that produces equal core and halo areas. This means the AoE has twice the area of the original support, split evenly between core (inside) and halo (outside).

Buffer method (default):

Computes a uniform buffer distance d such that the buffered area equals the target. The buffer distance is found by solving:

$$\pi d^2 + P \cdot d = A_{target}$$

where P is the perimeter and A_{target} is the desired halo area.

Stamp method:

Applies an affine transformation to each vertex:

$$p' = r + (1 + s)(p - r)$$

where r is the reference point (centroid), p is each vertex, and s is the scale factor. This method preserves shape proportions but only guarantees the AoE contains the original for star-shaped polygons (where the centroid can "see" all boundary points).

Points exactly on the original support boundary are classified as "core".

The support geometry is validated internally using `sf::st_make_valid()`.

Value

An `aoe_result` object (extends `sf`) containing only the supported points, with columns:

point_id Original point identifier (row name or index)

support_id Identifier for which support the classification refers to

aoe_class Classification: `"core"` or `"halo"`

When multiple supports are provided, points may appear multiple times (once per support whose AoE contains them).

The result has S3 methods for `print()`, `summary()`, and `plot()`. Use `aoe_geometry()` to extract the AoE polygons.

Examples

```
library(sf)

# Single support
support <- st_as_sf(
  data.frame(id = 1),
  geometry = st_sfc(st_polygon(list(
    cbind(c(0, 10, 10, 0, 0), c(0, 0, 10, 10, 0))
  ))),
  crs = 32631
)

pts <- st_as_sf(
  data.frame(id = 1:4),
  geometry = st_sfc(
    st_point(c(5, 5)),
    st_point(c(2, 2)),
    st_point(c(15, 5)),
    st_point(c(30, 30))
  ),
  crs = 32631
)

result <- aoe(pts, support)

# Multiple supports (e.g., admin regions)
supports <- st_as_sf(
  data.frame(region = c("A", "B")),
  geometry = st_sfc(
    st_polygon(list(cbind(c(0, 10, 10, 0, 0), c(0, 0, 10, 10, 0)))),
    st_polygon(list(cbind(c(8, 18, 18, 8, 8), c(0, 0, 10, 10, 0))))
```

```
  ),
  crs = 32631
)

result <- aoe(pts, supports)
# Points near the boundary may appear in both regions' AoE
```

---

aoe_area *Compute Area Statistics for AoE*

Description

Calculate area statistics for the original supports and their areas of effect, including expansion ratios, masking effects, and core/halo balance.

Usage

```
aoe_area(x)
```

Arguments

x	An `aoe_result` object returned by `aoe()`.

Details

With scale s, the AoE expands by multiplier $(1 + s)$ from centroid, resulting in $(1 + s)^2$ times the area. The theoretical halo:core ratio is $(1 + s)^2 - 1$:

- Scale 1 (default): ratio 3.0 (core 1 part, halo 3 parts)
- Scale 0.414: ratio 1.0 (equal areas)

Masking reduces the halo (and thus the ratio) when the AoE extends beyond hard boundaries.

Value

An `aoe_area_result` data frame with one row per support:

support_id Support identifier

area_core Area of core region (same as original support)

area_halo Area of halo region (AoE minus core, after masking)

area_aoe Total AoE area after masking

halo_core_ratio Ratio of halo to core area (theoretically 3.0 without mask)

pct_masked Percentage of theoretical AoE area removed by masking

Examples

```
library(sf)

support <- st_as_sf(
  data.frame(id = 1),
  geometry = st_sfc(st_polygon(list(
    cbind(c(0, 10, 10, 0, 0), c(0, 0, 10, 10, 0))
  ))),
  crs = 32631
)

pts <- st_as_sf(
  data.frame(id = 1:3),
  geometry = st_sfc(
    st_point(c(5, 5)),
    st_point(c(15, 5)),
    st_point(c(2, 2))
  ),
  crs = 32631
)

result <- aoe(pts, support)
aoe_area(result)
```

---

aoe_border *Classify Points by Distance from a Border*

Description

Given a set of points and a border (line), `aoe_border()` classifies points by their side relative to the border and their distance from it. Creates equal-area buffer zones on both sides of the border.

Usage

```
aoe_border(
  points,
  border,
  width = NULL,
  area = NULL,
  halo_width = NULL,
  halo_area = NULL,
  mask = NULL,
  bbox = NULL,
  side_names = c("side_1", "side_2"),
  coords = NULL
)
```

Arguments

points	An sf object with POINT geometries, or a data.frame with coordinate columns.
border	An sf object with LINESTRING or MULTILINESTRING geometry representing the border.
width	Buffer width in meters (for projected CRS) or degrees (for geographic CRS). Creates core zone within this distance of the border. Cannot be used together with area.
area	Target area for each side's core zone. The function finds the buffer width that produces this area per side. If mask is provided, the width is adjusted to achieve the target area *after* masking. Cannot be used together with width.
halo_width	Width of the halo zone beyond the core. If NULL (default), equals the core width for symmetric zones.
halo_area	Target area for each side's halo zone. Alternative to halo_width. If NULL and halo_width is NULL, defaults to equal area as core.
mask	Optional mask for clipping the buffer zones. Can be: • sf object with POLYGON or MULTIPOLYGON geometry • "land": use the bundled global land mask to exclude sea areas
bbox	Optional bounding box to limit the study area. Can be: • sf or sfc object (uses its bounding box) • Named vector: c(xmin = ..., ymin = ..., xmax = ..., ymax = ...) • NULL: no bbox restriction (uses buffer extent)
side_names	Character vector of length 2 naming the sides. Default is c("side_1", "side_2"). The first name is assigned to the left side of the border (when traversing from start to end).
coords	Column names for coordinates when points is a data.frame.

Details

The function creates symmetric buffer zones around a border line:

1. **Core zone**: Points within width (or area) distance of the border
2. **Halo zone**: Points beyond core but within width + halo_width
3. **Pruned**: Points outside the halo zone (not returned)

Each zone is split by the border line to determine which side the point falls on.

Equal area across sides:

When using the area parameter, the buffer width is calculated to produce equal area on both sides of the border. With masking, the width is adjusted so that the *masked* area on each side equals the target.

Value

An `aoe_border_result` object (extends `sf`) containing classified points with columns:

point_id Original point identifier

side Which side of the border: value from `side_names`

aoe_class Distance class: `"core"` or `"halo"`

Points outside the study area are pruned (not returned).

Examples

```
library(sf)

# Create a border line
border <- st_as_sf(
  data.frame(id = 1),
  geometry = st_sfc(st_linestring(matrix(
    c(0, 0, 100, 100), ncol = 2, byrow = TRUE
  ))),
  crs = 32631
)

# Create points
pts <- st_as_sf(
  data.frame(id = 1:6),
  geometry = st_sfc(
    st_point(c(10, 20)),   # near border, side 1
    st_point(c(30, 10)),   # near border, side 2
    st_point(c(50, 80)),   # far from border, side 1
    st_point(c(80, 40)),   # far from border, side 2
    st_point(c(5, 5)),     # very close to border
    st_point(c(200, 200))  # outside study area
  ),
  crs = 32631
)

# Classify by distance from border
result <- aoe_border(pts, border, width = 20)
```

---

aoe_expand *Adaptive AoE Expansion to Capture Minimum Points*

Description

Expands the area of effect just enough to capture at least `min_points`, subject to hard caps on expansion. This is useful when a fixed scale leaves some supports with insufficient data for stable modelling.

Usage

```
aoe_expand(
  points,
  support = NULL,
  min_points,
  max_area = 2,
  max_dist = NULL,
  method = c("buffer", "stamp"),
  reference = NULL,
  mask = NULL,
  coords = NULL
)
```

Arguments

points	An sf object with POINT geometries.
support	One of: • sf object with POLYGON/MULTIPOLYGON geometries • Country name or ISO code: "France", "FR", "FRA" • Vector of countries: c("France", "Germany") • Missing: auto-detects countries containing the points
min_points	Minimum number of points to capture in the AoE. The function finds the smallest scale that includes at least this many points.
max_area	Maximum halo area as a proportion of the original support area. Default is 2, meaning halo area cannot exceed twice the support area (total AoE <= 3x original). Set to Inf to disable.
max_dist	Maximum expansion distance in CRS units. For the buffer method, this is the maximum buffer distance. For the stamp method, this is converted to a maximum scale based on the support's characteristic radius. Default is NULL (no distance cap).
method	Method for computing the area of effect: • "buffer" (default): Uniform buffer around the support boundary. Robust for any polygon shape. Buffer distance is calculated to achieve the target halo area. • "stamp": Scale vertices outward from the centroid (or reference point). Preserves shape proportions but only guarantees containment for star-shaped polygons. May leave small gaps for highly concave shapes.
reference	Optional sf object with a single POINT geometry. If NULL (default), the centroid of each support is used. Only valid when support has a single row and method = "stamp".
mask	Optional mask for clipping the area of effect. Can be: • sf object with POLYGON or MULTIPOLYGON geometry • "land": use the bundled global land mask to exclude sea areas If provided, each area of effect is intersected with this mask.
coords	Column names for coordinates when points is a data.frame, e.g. c("lon", "lat"). If NULL, auto-detects common names.

Details

Unlike `aoe()`, which applies consistent geometry across all supports, `aoe_expand()` adapts the scale per-support based on local point density. Use with caution: this can make AoEs incomparable across regions with different point densities.

Algorithm:

For each support, binary search finds the minimum scale where point count >= min_points. The search is bounded by:

- Lower: scale = 0 (core only)
- Upper: minimum of max_area cap and max_dist cap

If the caps prevent reaching min_points, a warning is issued and the result uses the maximum allowed scale.

Caps:

Two caps ensure AoE doesn't expand unreasonably:

max_area (relative): Limits halo area to `max_area` times the original. The corresponding scale is `sqrt(1 + max_area) - 1`. Default max_area = 2 means scale <= 0.732 (total area <= 3x).

max_dist (absolute): Limits expansion distance in CRS units. For buffer method, this is the buffer distance directly. For stamp method, converted to scale via `max_dist / characteristic_radius` where characteristic_radius = sqrt(area / pi).

Value

An `aoe_result` object (same as `aoe()`) with additional attributes:

target_reached Logical: was `min_points` achieved for all supports? Use `attr(result, "expansion_info")` for per-support details.

expansion_info Data frame with per-support expansion details: support_id, scale_used, points_captured, target_reached, cap_hit.

See Also

`aoe()` for fixed-scale AoE computation

Examples

```
library(sf)

# Create a support with sparse points
support <- st_as_sf(
  data.frame(id = 1),
  geometry = st_sfc(st_polygon(list(
    cbind(c(0, 100, 100, 0, 0), c(0, 0, 100, 100, 0))
  ))),
  crs = 32631
)

# Points scattered around
set.seed(42)
```

```
pts <- st_as_sf(
  data.frame(id = 1:50),
  geometry = st_sfc(lapply(1:50, function(i) {
    st_point(c(runif(1, -50, 150), runif(1, -50, 150)))
  })),
  crs = 32631
)

# Expand until we have at least 20 points
result <- aoe_expand(pts, support, min_points = 20)

# Check expansion info
attr(result, "expansion_info")
```

`aoe_geometry` *Extract AoE Geometries*

Description

Extract the original support polygons and/or the area of effect polygons from an `aoe_result` object.

Usage

```
aoe_geometry(x, which = c("aoe", "original", "both"), support_id = NULL)
```

Arguments

`x`	An `aoe_result` object returned by `aoe()`.
`which`	Which geometry to extract: `"aoe"` (default), `"original"`, or `"both"`.
`support_id`	Optional character or numeric vector specifying which support(s) to extract. If `NULL` (default), extracts all.

Value

An `sf` object with polygon geometries and columns:

support_id Support identifier

type `"original"` or `"aoe"`

Examples

```
library(sf)

support <- st_as_sf(
  data.frame(region = c("A", "B")),
  geometry = st_sfc(
    st_polygon(list(cbind(c(0, 10, 10, 0, 0), c(0, 0, 10, 10, 0)))),
    st_polygon(list(cbind(c(15, 25, 25, 15, 15), c(0, 0, 10, 10, 0))))
```

```
  ),
  crs = 32631
)

pts <- st_as_sf(
  data.frame(id = 1:4),
  geometry = st_sfc(
    st_point(c(5, 5)),
    st_point(c(12, 5)),
    st_point(c(20, 5)),
    st_point(c(27, 5))
  ),
  crs = 32631
)

result <- aoe(pts, support)

# Get AoE polygons
aoe_polys <- aoe_geometry(result, "aoe")

# Get both original and AoE for comparison
both <- aoe_geometry(result, "both")

# Filter to one support (uses row names as support_id)
region_1 <- aoe_geometry(result, "aoe", support_id = "1")
```

`aoe_sample` *Stratified Sampling from AoE Results*

Description

Sample points from an `aoe_result` with control over core/halo balance. This is useful when core regions dominate due to point density, and you want balanced representation for modelling.

Usage

```
aoe_sample(x, ...)

## Default S3 method:
aoe_sample(x, ...)

## S3 method for class 'aoe_result'
aoe_sample(
  x,
  n = NULL,
  ratio = c(core = 0.5, halo = 0.5),
  replace = FALSE,
  by = c("overall", "support"),
```

```
  ...
)
```

Arguments

x	An aoe_result object returned by aoe() or aoe_expand().
...	Additional arguments passed to methods.
n	Total number of points to sample. If NULL, uses all available points subject to the ratio constraint (i.e., downsamples the larger group).
ratio	Named numeric vector specifying the target proportion of core and halo points. Must sum to 1. Default is c(core = 0.5, halo = 0.5) for equal representation.
replace	Logical. Sample with replacement? Default is FALSE. If FALSE and n exceeds available points in a stratum, that stratum contributes all its points.
by	Character. Stratification grouping: • "overall" (default): sample from all points regardless of support • "support": apply ratio within each support separately

Details

Sampling modes:

Fixed n: When n is specified, the function samples exactly n points (or fewer if not enough available), distributed according to ratio.

Balanced downsampling: When n is NULL, the function downsamples the larger stratum to match the smaller one according to ratio. For example, with ratio c(core = 0.5, halo = 0.5) and 100 core + 20 halo points, it returns 20 core + 20 halo = 40 points.

Multiple supports:

With by = "support", sampling is done independently within each support, then results are combined. This ensures each support contributes balanced samples. With by = "overall", all points are pooled first.

Value

An aoe_result object containing the sampled points, preserving all original columns and attributes. Has additional attribute sample_info with details about the sampling.

See Also

aoe() for computing AoE classifications

Examples

```
library(sf)

support <- st_as_sf(
  data.frame(id = 1),
  geometry = st_sfc(st_polygon(list(
    cbind(c(0, 100, 100, 0, 0), c(0, 0, 100, 100, 0))
```

```
  ))),
  crs = 32631
)

# Many points in core, few in halo
set.seed(42)
pts <- st_as_sf(
  data.frame(id = 1:60),
  geometry = st_sfc(c(
    lapply(1:50, function(i) st_point(c(runif(1, 10, 90), runif(1, 10, 90)))),
    lapply(1:10, function(i) st_point(c(runif(1, 110, 140), runif(1, 10, 90))))
  )),
  crs = 32631
)

result <- aoe(pts, support, scale = 1)

# Balance core/halo (downsamples core to match halo)
balanced <- aoe_sample(result)

# Fixed sample size with 70/30 split
sampled <- aoe_sample(result, n = 20, ratio = c(core = 0.7, halo = 0.3))
```

`aoe_sample.aoe_border_result` *Stratified Sampling from Border AoE Results*

Description

Sample points from an `aoe_border_result` with control over side and/or core/halo balance.

Usage

```
## S3 method for class 'aoe_border_result'
aoe_sample(
  x,
  n = NULL,
  ratio = NULL,
  by = c("side", "class"),
  replace = FALSE,
  ...
)
```

Arguments

`x`	An `aoe_border_result` object returned by `aoe_border()`.
`n`	Total number of points to sample. If `NULL`, uses all available points subject to the ratio constraint.

ratio	Named numeric vector specifying target proportions. Names should match the side names used in aoe_border() (e.g., c(side_1 = 0.5, side_2 = 0.5)) or use c(core = 0.5, halo = 0.5) for distance-based sampling. Must sum to 1.
by	Character. What to stratify by: • "side" (default): sample by side of the border • "class": sample by core/halo classification
replace	Logical. Sample with replacement? Default is FALSE.
...	Additional arguments (ignored).

Value

An aoe_border_result object containing the sampled points.

Examples

```
## Not run:
result <- aoe_border(pts, border, width = 1000,
                     side_names = c("west", "east"))

# Equal sampling from each side
balanced <- aoe_sample(result, ratio = c(west = 0.5, east = 0.5))

# Sample by core/halo instead
by_class <- aoe_sample(result, ratio = c(core = 0.5, halo = 0.5),
                       by = "class")

## End(Not run)
```

---

aoe_summary *Summarize Area of Effect Results*

Description

Compute summary statistics for an AoE classification result, including counts and proportions of core vs halo points per support.

Usage

```
aoe_summary(x)
```

Arguments

x	An sf object returned by aoe().

Value

A data frame with one row per support, containing:

support_id Support identifier

n_total Total number of supported points

n_core Number of core points

n_halo Number of halo points

prop_core Proportion of points that are core

prop_halo Proportion of points that are halo

Examples

```
library(sf)

support <- st_as_sf(
  data.frame(id = 1),
  geometry = st_sfc(st_polygon(list(
    cbind(c(0, 10, 10, 0, 0), c(0, 0, 10, 10, 0))
  ))),
  crs = 32631
)

pts <- st_as_sf(
  data.frame(id = 1:4),
  geometry = st_sfc(
    st_point(c(5, 5)),
    st_point(c(2, 2)),
    st_point(c(15, 5)),
    st_point(c(12, 5))
  ),
  crs = 32631
)

result <- aoe(pts, support)
aoe_summary(result)
```

---

countries *World Country Polygons with Pre-calculated AoE Bounds*

Description

An `sf` object containing country polygons from Natural Earth (1:50m scale) with pre-calculated bounding boxes for area of effect analysis.

Usage

```
countries
```

Format

An sf data frame with 237 rows and 9 variables:

iso2 ISO 3166-1 alpha-2 country code (e.g., "FR", "BE")

iso3 ISO 3166-1 alpha-3 country code (e.g., "FRA", "BEL")

name Country name

continent Continent name

bbox Original bounding box (xmin, ymin, xmax, ymax) in Mollweide

bbox_equal_area AoE bounding box at scale sqrt(2)-1 (equal areas)

bbox_equal_ray AoE bounding box at scale 1 (equal linear distance)

halo_equal_area_scale Scale factor that produces halo area = country area (with land mask)

geometry Country polygon in WGS84 (EPSG:4326)

Source

Natural Earth https://www.naturalearthdata.com/

Examples

```
# Get France
france <- countries[countries$iso3 == "FRA", ]

# Use directly with aoe()
```

---

country_halos *Pre-computed Equal-Area Country Halos*

Description

A named list of pre-computed halo geometries for each country where the halo area equals the country area (area proportion = 1). These halos account for land masking (sea areas excluded).

Usage

```
country_halos
```

Format

A named list with ISO3 country codes as names. Each element is either an sfc geometry (POLYGON or MULTIPOLYGON) in WGS84, or NULL if computation failed for that country.

Details

Each halo is a "donut" shape: the area between the original country boundary and the expanded boundary, clipped to land.

Source

Computed from Natural Earth country polygons and land mask.

Examples

```
# Get France's equal-area halo
france_halo <- country_halos[["FRA"]]
```

`get_country` *Get Country Polygon by Name or ISO Code*

Description

Quick accessor for country polygons from the bundled dataset.

Usage

```
get_country(x)
```

Arguments

x	Country name, ISO2 code, or ISO3 code (case-insensitive)

Value

An `sf` object with the country polygon, or error if not found.

Examples

```
get_country("Belgium")
get_country("BE")
get_country("BEL")
```

`land` *Global Land Mask*

Description

An `sf` object containing the global land polygon from Natural Earth (1:50m scale). Used for masking area of effect computations to exclude ocean areas.

Usage

```
land
```

Format

An sf data frame with 1 row:

name Description ("Global Land")

geometry Land multipolygon in WGS84 (EPSG:4326)

Source

Natural Earth https://www.naturalearthdata.com/

Examples

```
# Use as mask to exclude sea

dummy <- sf::st_as_sf(
  data.frame(id = 1),
  geometry = sf::st_sfc(sf::st_point(c(14.5, 47.5))),
  crs = 4326
)
result <- aoe(dummy, "AT", mask = land)
```

---

plot.aoe_border_result

Plot method for aoe_border_result

Description

Plot method for aoe_border_result

Usage

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

Arguments

x	An aoe_border_result object
...	Additional arguments passed to plot

Value

NULL (called for side effect)

`plot.aoe_result` *Plot method for aoe_result*

Description

Visualize an AoE classification result, showing points colored by class and optionally the support and AoE boundaries.

Usage

```
## S3 method for class 'aoe_result'
plot(
  x,
  support_id = NULL,
  show_aoe = TRUE,
  show_original = TRUE,
  col_core = "#2E7D32",
  col_halo = "#F57C00",
  col_original = "#000000",
  col_aoe = "#9E9E9E",
  pch = 16,
  cex = 0.8,
  main = NULL,
  ...
)
```

Arguments

`x`	An aoe_result object
`support_id`	Optional: filter to specific support(s)
`show_aoe`	Logical; show AoE boundary (default TRUE)
`show_original`	Logical; show original support boundary (default TRUE)
`col_core`	Color for core points (default "#2E7D32", green)
`col_halo`	Color for halo points (default "#F57C00", orange)
`col_original`	Color for original support boundary (default "#000000")
`col_aoe`	Color for AoE boundary (default "#9E9E9E")
`pch`	Point character (default 16)
`cex`	Point size (default 0.8)
`main`	Plot title (default auto-generated)
`...`	Additional arguments passed to plot

Value

Invisibly returns x

Examples

```
library(sf)

support <- st_as_sf(
  data.frame(id = 1),
  geometry = st_sfc(st_polygon(list(
    cbind(c(0, 10, 10, 0, 0), c(0, 0, 10, 10, 0))
  ))),
  crs = 32631
)

set.seed(42)
pts <- st_as_sf(
  data.frame(id = 1:50),
  geometry = st_sfc(lapply(1:50, function(i) {
    st_point(c(runif(1, -5, 15), runif(1, -5, 15)))
  })),
  crs = 32631
)

result <- aoe(pts, support)
plot(result)
```

---

print.aoe_area_result *Print method for aoe_area_result*

Description

Print method for aoe_area_result

Usage

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

Arguments

x	An aoe_area_result object
...	Additional arguments (ignored)

Value

Invisibly returns x

`print.aoe_border_result` *Print method for aoe_border_result*

Description

Print method for aoe_border_result

Usage

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

Arguments

`x`	An aoe_border_result object
`...`	Additional arguments (ignored)

Value

Invisibly returns x

`print.aoe_expand_result` *Print method for aoe_expand_result*

Description

Print method for aoe_expand_result

Usage

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

Arguments

`x`	An aoe_expand_result object
`...`	Additional arguments (ignored)

Value

Invisibly returns x

print.aoe_result *Print method for aoe_result*

Description

Print method for aoe_result

Usage

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

Arguments

x	An aoe_result object
...	Additional arguments passed to print.sf

Value

Invisibly returns x

print.aoe_summary_result *Print method for aoe_summary_result*

Description

Print method for aoe_summary_result

Usage

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

Arguments

x	An aoe_summary_result object
...	Additional arguments (ignored)

Value

Invisibly returns x

`summary.aoe_result` *Summary method for aoe_result*

Description

Summary method for aoe_result

Usage

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

Arguments

`object`	An aoe_result object
`...`	Additional arguments (ignored)

Value

An aoe_summary_result object

Index