Package: explodemap (via r-universe)

October 21, 2026

Title Hierarchical Exploded-View Cartography

Version 0.5.0

Description Tools for generating hierarchical exploded-view maps from dense administrative boundary data. The package applies rigid-body translations to polygon geometries using a centroid-driven vector field, preserving the internal geometry of each feature while separating units within and across regions. Parameters can be derived analytically from dataset geometry using closed-form models for regional separation and local expansion. The package also includes grouped layouts, optional bounded collision refinement, and an interactive focus-map widget for selected-area inspection in 'htmlwidgets' and 'Shiny'. It implements the methodology described in George Arthur (2026) <https://github.com/PrigasG/explodemap> ``A Hierarchical Vector-Based Framework for Multi-Scale Exploded-View Cartography''.

License MIT + file LICENSE

URL https://prigasg.github.io/explodemap/,
https://github.com/PrigasG/explodemap,
https://huggingface.co/spaces/Prigas89/explodemap-gallery,
https://huggingface.co/spaces/Prigas89/spatial-pipeline-studio

BugReports https://github.com/PrigasG/explodemap/issues

Depends R (>= 4.1.0)

Imports digest, dplyr, ggplot2, htmlwidgets, purrr, rlang, sf, utils

Suggests bslib, dragmapr, ggiraph, jsonlite, knitr, later, rmarkdown, shiny, testthat (>= 3.0.0), tigris

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libuv1-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-09-21 02:23:05 UTC

RemoteUrl https://github.com/prigasg/explodemap

RemoteRef HEAD

RemoteSha b2fbf96717dfa6c1cdaf986af690ff9a21f309d0

Contents

anchor_table . . . 3
apply_region_offsets . . . 4
as_dragmapr . . . 5
as_dragmapr_state . . . 6
as_hhs_states . . . 6
as_sf . . . 7
assign_spatial_groups . . . 8
cache_clear . . . 9
cache_list . . . 9
calibration_row . . . 10
child_layout_offsets . . . 10
compare_layouts . . . 11
compute_stats . . . 12
connector_geometry . . . 12
count_geometry_vertices . . . 13
derive_params . . . 14
diagnose_layout . . . 14
displacement_magnitudes . . . 15
e_fingerprint . . . 16
estimate_block_radii . . . 16
explode_grouped . . . 17
explode_section . . . 19
explode_sf . . . 21
explode_sf_with_lookup . . . 22
explode_state . . . 23
export_topojson . . . 25
focus_map . . . 27
focus_map_preset . . . 31
focusMapProxy . . . 32
group_palette . . . 33
hhs_display_offsets . . . 33
hhs_focus_map . . . 34
hhs_layout . . . 34
hhs_palette . . . 35
hhs_region_names . . . 35

hhs_regions 36
layout_children 36
layout_objective 37
layout_offsets 38
layout_regions 39
optimize_grouped_layout 40
original_geometry 41
plot.exploded_map 42
plot.grouped_exploded_map 42
plot.layout_quality_report 43
prepare_explodemap_input 43
print.exploded_map 44
print.grouped_exploded_map 45
read_drag_offsets 45
simplify_to_vertex_budget 46
spatial_join_index 47
summary.exploded_map 48
summary.grouped_exploded_map 48
transition_data 49
update_exploded_layout 50
validate_explodemap_input 50
validate_spatial_join 51

Index **53**

`anchor_table` *Access grouped-layout anchor data*

Description

Access grouped-layout anchor data

Usage

```
anchor_table(layout)
```

Arguments

`layout`	A `grouped_exploded_map`.

Value

A data frame with one row per group anchor.

apply_region_offsets *Apply documented display offsets by region*

Description

Applies rigid display offsets to all features in each named region. This is intended as a reproducible finishing step for publication or dashboard layouts after an exploded map has already been computed.

Usage

```
apply_region_offsets(
  x,
  offsets,
  region_col = NULL,
  offset_region_col = "region",
  dx_col = "dx_m",
  dy_col = "dy_m",
  update_plots = TRUE
)
```

Arguments

`x`	An `sf`, `exploded_map`, or `grouped_exploded_map` object.
`offsets`	A data frame, or a path to a CSV file, containing one row per region and numeric offset columns.
`region_col`	Name of the region column in `x`. For `exploded_map` and `grouped_exploded_map` objects, the value recorded in `x$diagnostics` is used when this is `NULL`.
`offset_region_col`	Name of the region column in `offsets`.
`dx_col`	Name of the horizontal offset column in `offsets`.
`dy_col`	Name of the vertical offset column in `offsets`.
`update_plots`	Logical; when `TRUE`, rebuild stored ggplot objects for exploded map objects.

Details

Offsets are interpreted in the coordinate units of the projected geometry being adjusted. The default column names `dx_m` and `dy_m` assume a metric projected CRS, matching the package's exploded-map workflow.

Value

For an `sf` input, an `sf` object with the same attributes and CRS as `x`; geometries belonging to each listed region are translated by the supplied offsets, and the normalized offset table is stored in the `display_offsets` attribute. For an `exploded_map`, returns the same S3 object structure with

sf_exp, sf_exp_wgs, stored plots, and display_offsets updated. For a grouped_exploded_map, returns the same S3 object structure with sf_grouped, sf_grouped_wgs, stored plots, and display_offsets updated. The offsets are rigid translations, so polygon shape, area, and within-region relative geometry are preserved.

Examples

```
library(sf)
x <- st_sf(
  region = c("A", "B"),
  geometry = st_sfc(
    st_point(c(0, 0)),
    st_point(c(10, 0)),
    crs = 3857
  )
)
offsets <- data.frame(region = "B", dx_m = 5, dy_m = 2)
apply_region_offsets(x, offsets, region_col = "region")
```

---

as_dragmapr *Convert a grouped layout to a drag-edit handoff object (legacy)*

Description

Legacy / low-level handoff (superseded, but not deprecated).

This is the original handoff: it bundles the grouped geometry and the absolute anchors into a dragmapr_layout list. The preferred path is now as_dragmapr_state(), which emits a geometry-free dragmapr::d_state() – the shared editorial composition contract that state = arguments accept across focus_map(), render_dragged_map(), and update_exploded_layout().

as_dragmapr() remains supported for existing offset-table / CSV workflows and is not scheduled for removal, but new code should prefer the state-first API.

Usage

```
as_dragmapr(result)
```

Arguments

result	A grouped_exploded_map.

Value

A dragmapr_layout list containing editable sf geometry and anchors.

See Also

as_dragmapr_state() for the preferred state-first handoff.

as_dragmapr_state — *Convert a grouped layout to a dragmapr editing state*

Description

Produces a `dragmapr::d_state()` – the shared editorial composition contract – from a `grouped_exploded_map`. Unlike `as_dragmapr()`, which bundles geometry and absolute anchors into a handoff list, this emits the geometry-free *state* that the interactive editor and static renderers consume: the exploded anchors (`anchor_x`/`anchor_y`) are converted to the deltas (`dx_m`/`dy_m`, in metres) relative to the current region centroids that `dragmapr` expects, and the projected CRS plus a provenance `geometry_id` are recorded so the state can be reapplied safely in a later session.

Usage

```
as_dragmapr_state(result, geometry_id = NULL, level = NULL)
```

Arguments

`result`	A `grouped_exploded_map`.
`geometry_id`	Optional single string identifying the source geometry. Defaults to the layout label in `result$diagnostics$label`.
`level`	Optional geography-level label. Defaults to the region column.

Details

Keep `layout` (the computed geometry + diagnostics, i.e. `result`) and `state` (this editorial overlay) separate. The layout can be recomputed freely while the state preserves manual edits, which is what makes the `compute -> compose -> render` workflow reproducible.

Value

A `dragmapr_state` (see `dragmapr::d_state()`).

as_hhs_states — *Standardize state polygons for HHS reference maps*

Description

Adds `hhs_region`, `state_label`, and `state_geoid` columns to state or territory polygons and transforms long-lat data to a projected CRS.

Usage

```
as_hhs_states(states, crs = 5070)
```

Arguments

states	An sf object with either hhs_region, region, STUSPS, or NAME columns.
crs	Target projected CRS used when states is long-lat. Default is EPSG:5070.

Value

An sf object filtered to HHS-assigned states and territories.

---

as_sf — *Extract plain sf geometry from an exploded layout*

Description

Converts an exploded_map or grouped_exploded_map back to a plain sf data frame so the layout can flow into downstream GIS work (writing files, spatial joins, further sf analysis) without carrying the layout metadata.

Usage

```
as_sf(layout, which = c("exploded", "original"))
```

Arguments

layout	An exploded_map or grouped_exploded_map object.
which	Which stored geometry layer to return. For exploded_map: "exploded" (default) or "original". For grouped_exploded_map: "grouped" (default), "local", or "original".

Value

A plain sf object with the selected geometry layer active. All attribute columns and the CRS are kept as stored. Layout metadata (stats, params, plots, refinement, diagnostics, anchors) is dropped. The "exploded"/"grouped" layers carry the displaced geometries; "original" carries the input geometries.

Examples

```
sq <- function(xmin, ymin, size = 1) {
  sf::st_polygon(list(matrix(
    c(xmin, ymin, xmin + size, ymin, xmin + size, ymin + size,
      xmin, ymin + size, xmin, ymin),
    ncol = 2, byrow = TRUE
  )))
}
x <- sf::st_sf(
  id = c("a1", "a2", "b1", "b2"),
  region = c("A", "A", "B", "B"),
  geometry = sf::st_sfc(sq(0, 0), sq(2, 0), sq(10, 0), sq(12, 0), crs = 3857)
```

```
)
layout <- explode_sf(x, region_col = "region", plot = FALSE, quiet = TRUE)

# Displaced geometries, ready for st_write() or further analysis
exp_sf <- as_sf(layout)
sf::st_crs(exp_sf)

# The untouched input geometries
orig_sf <- as_sf(layout, which = "original")
```

`assign_spatial_groups` *Assign generated spatial groups*

Description

Assign generated spatial groups

Usage

```
assign_spatial_groups(
  x,
  method = c("quadrants", "clusters", "individual"),
  groups = 6L,
  seed = 1L,
  group_col = "region"
)
```

Arguments

`x`	An `sf` object.
`method`	One of `"quadrants"`, `"clusters"`, or `"individual"`.
`groups`	Number of clusters for `method = "clusters"` (minimum 2).
`seed`	Random seed for deterministic clustering.
`group_col`	Name of the output grouping column.

Value

`x` with `group_col` assigned.

cache_clear *Clear explodemap download cache*

Description

Clear explodemap download cache

Usage

```
cache_clear(key = NULL)
```

Arguments

key	Specific cache key to clear, or NULL to clear all

Value

Invisibly returns `NULL`. The function is called for its side effect of deleting cached `.rds` files and reporting what was removed.

cache_list *List cached datasets*

Description

List cached datasets

Usage

```
cache_list()
```

Value

Invisibly returns a character vector of cached `.rds` file names. The visible output is a message listing the cache directory and file sizes, or a message that the cache is empty.

calibration_row *Extract calibration row from an exploded_map*

Description

Returns a one-row data.frame suitable for binding across datasets to build a calibration table.

Usage

```
calibration_row(x)
```

Arguments

x	An exploded_map object

Value

A one-row data.frame

child_layout_offsets *Access a child layout*

Description

Access a child layout

Usage

```
child_layout_offsets(layout)

child_layout_geometry(layout)
```

Arguments

layout	An explodemap_child_layout from layout_children().

Value

child_layout_offsets() returns the renderer-neutral offset table; child_layout_geometry() returns the composed projected sf geometry.

compare_layouts *Compare displacement between two exploded layouts*

Description

Summarises how per-feature displacement changes between two layouts of the same input (e.g. different `alpha_r`/`alpha_l`/`gamma` choices or before/after collision refinement). Both layouts must be of the same class and cover the same input features in the same row order; this is verified by comparing the original geometries pairwise, so comparing two unrelated layouts of equal size is an error.

Usage

```
compare_layouts(before, after)
```

Arguments

`before`, `after`	`exploded_map` objects (or both `grouped_exploded_map` objects) to compare.

Value

A list with two elements:

- `per_feature`: `data.frame` with one row per feature and columns `distance_before`, `distance_after`, and `delta` (after minus before), in metres.
- `summary`: `data.frame` with one row per layout (`"before"`, `"after"`) and columns `total`, `mean`, and `max` displacement.

Examples

```
sq <- function(xmin, ymin, size = 1) {
  sf::st_polygon(list(matrix(
    c(xmin, ymin, xmin + size, ymin, xmin + size, ymin + size,
      xmin, ymin + size, xmin, ymin),
    ncol = 2, byrow = TRUE
  )))
}
x <- sf::st_sf(
  id = c("a1", "a2", "b1", "b2"),
  region = c("A", "A", "B", "B"),
  geometry = sf::st_sfc(sq(0, 0), sq(2, 0), sq(10, 0), sq(12, 0), crs = 3857)
)
small <- explode_sf(x, region_col = "region", alpha_r = 10, alpha_l = 10,
                    plot = FALSE, quiet = TRUE)
big <- explode_sf(x, region_col = "region", alpha_r = 100, alpha_l = 100,
                  plot = FALSE, quiet = TRUE)
cmp <- compare_layouts(small, big)
cmp$summary
```

compute_stats — *Compute geometry statistics for parameter derivation*

Description

Extracts w_bar, R_local, n_bar, n_regions, and the tightness ratio from a projected sf object with a grouping column.

Usage

```
compute_stats(
  sf_obj,
  region_col,
  centroid_fun = c("centroid", "point_on_surface")
)
```

Arguments

sf_obj	Projected sf object
region_col	Name of the grouping column
centroid_fun	"centroid" (default) or "point_on_surface"

Value

A named list of geometry statistics

connector_geometry — *Connector geometry for computed layout movement*

Description

Builds one line per moved group or feature, from the original anchor to the final anchor.

Usage

```
connector_geometry(
  layout,
  level = c("group", "feature"),
  id_col = NULL,
  threshold_m = 1,
  include_unmoved = FALSE,
  require_stable_id = identical(level, "feature")
)
```

Arguments

layout	A grouped_exploded_map.
level	Either "group" or "feature".
id_col	Optional feature ID column for level = "feature".
threshold_m	Minimum movement distance to mark a connector visible.
include_unmoved	Include zero-distance rows.
require_stable_id	Require a real feature ID column for feature-level persistent handoffs. When TRUE, row-number fallback IDs are rejected. Defaults to TRUE when level = "feature", FALSE otherwise.

Value

An sf line layer with movement attributes.

---

count_geometry_vertices

Count geometry vertices

Description

Counts coordinate rows in an sf or sfc object. This is a lightweight utility for upload limits, performance warnings, simplification decisions, and diagnostics.

Usage

```
count_geometry_vertices(x)
```

Arguments

x	An sf, sfc, or geometry object accepted by sf::st_geometry().

Value

A single numeric vertex count.

derive_params *Derive displacement parameters from geometry statistics*

Description

Implements Analytical Results 1 and 2 from the paper: alpha_r = gamma_r * w_bar / (2 * sin(pi / n_regions)) alpha_l = gamma_l * 2 * R_local / sqrt(n_bar)

Usage

```
derive_params(stats, gamma_r = 3, gamma_l = 1.136, p = 1.25)
```

Arguments

stats	Output of compute_stats()
gamma_r	Regional clearance coefficient (default 3.0)
gamma_l	Local clearance coefficient (default 1.136)
p	Distance scaling exponent (default 1.25)

Value

Named list with alpha_r, alpha_l, p, gamma_r, gamma_l

diagnose_layout *Diagnose grouped layout quality*

Description

Diagnose grouped layout quality

Usage

```
diagnose_layout(result, label_col = NULL, label_size = NULL, state = NULL)
```

Arguments

result	A grouped_exploded_map object.
label_col	Optional label column for approximate label overlap checks.
label_size	Approximate label size in map units.
state	Optional dragmapr_state. When supplied, its manual offsets are applied to the grouped layout before every geometry, label, gap, and displacement diagnostic is calculated.

Value

A layout_quality_report list.

displacement_magnitudes

Per-feature displacement magnitudes of an exploded layout

Description

Measures how far each feature moved from its original position by comparing feature centroids before and after displacement. Useful for summarising a layout and for comparing parameter choices (see `compare_layouts()`).

Usage

```
displacement_magnitudes(layout)
```

Arguments

`layout`	An `exploded_map` or `grouped_exploded_map` object.

Value

A `data.frame` with one row per feature (in feature order) and columns `dx`, `dy` (signed displacement components) and `distance` (Euclidean displacement magnitude), all in metres.

Examples

```
sq <- function(xmin, ymin, size = 1) {
  sf::st_polygon(list(matrix(
    c(xmin, ymin, xmin + size, ymin, xmin + size, ymin + size,
      xmin, ymin + size, xmin, ymin),
    ncol = 2, byrow = TRUE
  )))
}
x <- sf::st_sf(
  id = c("a1", "a2", "b1", "b2"),
  region = c("A", "A", "B", "B"),
  geometry = sf::st_sfc(sq(0, 0), sq(2, 0), sq(10, 0), sq(12, 0), crs = 3857)
)
layout <- explode_sf(x, region_col = "region", plot = FALSE, quiet = TRUE)
head(displacement_magnitudes(layout))
```

e_fingerprint *Fingerprint explodemap geometry and grouping*

Description

Fingerprint explodemap geometry and grouping

Usage

```
e_fingerprint(
  x,
  id_col = NULL,
  group_col = NULL,
  include_geometry = TRUE,
  include_parameters = TRUE,
  require_stable_id = FALSE
)
```

Arguments

`x`	An `sf` object or grouped explodemap layout.
`id_col`	Optional feature ID column.
`group_col`	Optional group column.
`include_geometry`	Include EWKB geometry bytes.
`include_parameters`	Include layout parameters (when `x` is a grouped explodemap layout) and the package version in the fingerprint.
`require_stable_id`	Require a real feature ID column instead of row-number fallback IDs.

Value

A stable MD5 fingerprint string.

estimate_block_radii *Estimate block radius for each region*

Description

Computes the 85th percentile of distances from child-unit centroids to their region centroid. The 85th percentile is preferred over the maximum because the maximum is sensitive to outlier units and produces over-conservative block radii.

Usage

```
estimate_block_radii(
  sf_obj,
  region_col,
  quantile_p = 0.85,
  centroid_fun = c("centroid", "point_on_surface")
)
```

Arguments

`sf_obj`	Projected sf object with region column
`region_col`	Grouping column name
`quantile_p`	Quantile for radius estimation (default 0.85)
`centroid_fun`	"centroid" or "point_on_surface"

Value

data.frame with columns: region, block_radius, cx, cy, n_units

---

`explode_grouped` *Create a three-level grouped exploded map*

Description

Combines Level 1 (local explosion within regions) with Level 2/3 (anchor-based region block placement). This is the full three-level extension from Section 12 of the paper.

Usage

```
explode_grouped(
  sf_obj,
  region_col,
  mode = c("auto", "auto_collision", "manual"),
  anchors = NULL,
  initial_layout = NULL,
  preserve_manual = FALSE,
  alpha_l = NULL,
  p = 1.25,
  gamma_l = 1.136,
  kappa = 1.8,
  padding = 50000,
  delta = 15000,
  lambda = 0.18,
  eta = 0.18,
  padding_sep = 20000,
  anchor_expand = NULL,
```

```
  anchor_buffer = NULL,
  density_scale = NULL,
  block_sep = NULL,
  max_iter = 60,
  fix_invalid = TRUE,
  allow_other = FALSE,
  centroid_fun = c("centroid", "point_on_surface"),
  plot = TRUE,
  export = NULL,
  label = "Grouped Layout",
  quiet = FALSE
)
```

Arguments

`sf_obj`	Projected sf object with region column
`region_col`	Grouping column name
`mode`	"auto", "auto_collision", or "manual"
`anchors`	For mode = "manual": data.frame with anchor positions
`initial_layout`	Optional existing/manual anchor layout used when `preserve_manual = TRUE`.
`preserve_manual`	When `TRUE`, use `initial_layout` as starting anchors so parameter changes refine an existing composition instead of replacing it.
`alpha_l`	Local expansion parameter for Level 1 (in metres)
`p`	Distance scaling exponent (default 1.25)
`gamma_l`	Local clearance coefficient (default 1.136); used if alpha_l is NULL
`kappa`	Radial expansion factor (default 1.8)
`padding`	Base padding (default 50000)
`delta`	Log-density scaling (default 15000)
`lambda`	Spring coefficient (default 0.18)
`eta`	Repulsion step (default 0.18)
`padding_sep`	Minimum block separation (default 20000)
`anchor_expand`, `anchor_buffer`, `density_scale`, `block_sep`	Optional aliases for `kappa`, `padding`, `delta`, and `padding_sep`, respectively. These names are convenient in Shiny dashboards where the controls describe the visual effect rather than the solver term.
`max_iter`	Max collision iterations (default 60)
`fix_invalid`	Auto-repair invalid geometries (default TRUE)
`allow_other`	Keep features labeled `"Other"` in the output without moving them: they are excluded from the local explosion and anchor placement, then recombined unchanged. Defaults to `FALSE` (error on `"Other"` unless handled).
`centroid_fun`	"centroid" or "point_on_surface"

plot	Print plots (default TRUE). Automatically suppressed inside a live Shiny session; use plot.grouped_exploded_map() inside renderPlot().
export	NULL, TRUE, or file path
label	Title for plots
quiet	If TRUE, suppress message() output. Default FALSE.

Details

The guarantees of Propositions 1-3 apply strictly at Level 1. Higher levels preserve structural grouping and directional correspondence rather than topological coverage.

Value

A grouped_exploded_map S3 object (inherits from exploded_map)

---

explode_section *Explode one selected section and keep the rest as context*

Description

explode_section() is a dashboard-oriented helper for exploratory maps. It applies an exploded layout only to the requested section, then recombines the untouched remainder of the layer as geographic context. The result can be passed directly to focus_map() with context_col to fade or hide the non-selected features.

Usage

```
explode_section(
  sf_obj,
  section_col,
  section,
  region_col = section_col,
  layout = c("explode", "grouped"),
  context = c("fade", "hide", "none"),
  role_col = ".explodemap_role",
  all_values = "all",
  ...
)
```

Arguments

sf_obj	Projected sf object.
section_col	Column containing the high-level sections users choose from, such as "North", "Central", and "South".
section	Selected section value. Values in all_values explode the full layer.

region_col	Column used for the explosion inside the selected section. Defaults to section_col. For municipality drill-downs, this is often a county column. Must have at least two distinct values within the selected section; the section_col default is only usable with section = "all".
layout	"explode" for explode_sf() or "grouped" for explode_grouped().
context	"fade" or "hide" keeps non-selected features in the returned object as context; "none" drops them.
role_col	Name of the role column added to the output. Focus features are marked "focus" and context features are marked "context".
all_values	Values that mean "all sections". Default "all".
...	Passed to explode_sf() or explode_grouped().

Value

An exploded_map or grouped_exploded_map object with recombined focus/context geometry and extra diagnostics.

Examples

```
poly <- function(xmin, ymin, xmax, ymax) {
  sf::st_polygon(list(rbind(
    c(xmin, ymin), c(xmax, ymin), c(xmax, ymax),
    c(xmin, ymax), c(xmin, ymin)
  )))
}

municipalities <- sf::st_sf(
  NAME = c("A", "B", "C", "D"),
  nj_region = c("South", "South", "North", "North"),
  county_name = c("Atlantic", "Cape May", "Bergen", "Hudson"),
  geometry = sf::st_sfc(
    poly(0, 0, 1000, 1000),
    poly(2000, 0, 3000, 1000),
    poly(0, 2000, 1000, 3000),
    poly(2000, 2000, 3000, 3000),
    crs = 3857
  )
)

focused <- explode_section(
  municipalities,
  section_col = "nj_region",
  section = "South",
  region_col = "county_name",
  alpha_r = 1800,
  alpha_l = 1200,
  plot = FALSE,
  quiet = TRUE
)
```

```
focus_map(
  focused,
  label_col = "NAME",
  context_col = ".explodemap_role",
  context_mode = "fade"
)
```

---

explode_sf *Explode any sf object with an existing grouping column*

Description

Explode any sf object with an existing grouping column

Usage

```
explode_sf(
  sf_obj,
  region_col = "region",
  gamma_r = 3,
  gamma_l = 1.136,
  p = 1.25,
  alpha_r = NULL,
  alpha_l = NULL,
  refine = FALSE,
  refine_min_gap = NULL,
  refine_max_shift = NULL,
  refine_max_iter = 20,
  refine_step = 0.5,
  refine_within = c("region", "all"),
  allow_other = FALSE,
  fix_invalid = TRUE,
  centroid_fun = c("centroid", "point_on_surface"),
  plot = TRUE,
  export = NULL,
  label = "Custom Dataset",
  quiet = FALSE
)
```

Arguments

sf_obj	Projected sf object (metric CRS)
region_col	Name of the column defining groups
gamma_r	Regional clearance coefficient (default 3.0)
gamma_l	Local clearance coefficient (default 1.136)
p	Distance scaling exponent (default 1.25)

alpha_r	Optional manual override for regional separation (in metres). May be supplied independently of alpha_l.
alpha_l	Optional manual override for local expansion (in metres). May be supplied independently of alpha_r.
refine	If TRUE, apply a bounded collision-refinement pass after the analytical displacement. Default is FALSE.
refine_min_gap	Optional minimum boundary-to-boundary gap in map units. If NULL and refine = TRUE, uses 2% of the characteristic diameter.
refine_max_shift	Optional maximum correction per feature in map units. If NULL and refine = TRUE, uses 10% of alpha_r + alpha_l.
refine_max_iter	Maximum refinement iterations.
refine_step	Fraction of each gap deficit corrected per iteration.
refine_within	Refine pairs within each "region" (default) or across "all" features.
allow_other	If TRUE, permits "Other" units
fix_invalid	If TRUE, auto-repairs invalid geometries
centroid_fun	"centroid" (default) or "point_on_surface"
plot	Print plots on return. Automatically suppressed when called inside a live Shiny session; use plot.exploded_map() inside renderPlot() instead.
export	NULL, TRUE, or file path
label	Title for plots
quiet	If TRUE, suppress all message() output. Default FALSE.

Value

An exploded_map S3 object

---

explode_sf_with_lookup

Explode any sf object using an external lookup table

Description

Joins the lookup to sf_obj before exploding. Unmatched units are labelled "Other".

Usage

```
explode_sf_with_lookup(
  sf_obj,
  join_col,
  lookup,
  lookup_key = join_col,
```

```
  region_col = "region",
  quiet = FALSE,
  ...
)
```

Arguments

sf_obj	Projected sf object
join_col	Column in sf_obj to join on
lookup	data.frame with join key and region column
lookup_key	Column name in lookup matching join_col
region_col	Column name in lookup containing region labels
quiet	If TRUE, suppress message() output. Default FALSE. Passed through to explode_sf().
...	Passed to explode_sf()

Value

An exploded_map S3 object

---

explode_state *Explode a US state from TIGER/Line data*

Description

[Experimental] Downloads administrative boundaries automatically, groups them into regions, derives displacement parameters via Analytical Results 1-2, and returns an exploded_map S3 object.

Usage

```
explode_state(
  state_fips = NULL,
  crs = NULL,
  region_map = NULL,
  level = c("cousub", "county"),
  n_regions = NULL,
  sf_data = NULL,
  gamma_r = 3,
  gamma_l = 1.136,
  p = 1.25,
  alpha_r = NULL,
  alpha_l = NULL,
  refine = FALSE,
  refine_min_gap = NULL,
  refine_max_shift = NULL,
  refine_max_iter = 20,
```

```
  refine_step = 0.5,
  refine_within = c("region", "all"),
  allow_other = FALSE,
  fix_invalid = TRUE,
  centroid_fun = c("centroid", "point_on_surface"),
  plot = TRUE,
  export = NULL,
 label = if (!is.null(state_fips)) paste0("FIPS ", state_fips) else "Custom Dataset",
  quiet = FALSE
)
```

Arguments

`state_fips` 2-digit FIPS code (e.g. `"34"` for NJ). Required for automatic downloads; optional when `sf_data` is supplied.

`crs` Projected CRS EPSG code (metric units, e.g. `32111`). Required for automatic downloads; when `sf_data` is supplied the data must already be in a projected metric CRS and `crs` is used only for re-projection if the current CRS differs.

`region_map` Named list mapping region labels to county/COUSUB name vectors (e.g. `list(North = c("Bergen", "Essex"), South = c(...))`. For `level = "county"` this is optional; omit it to use automatic k-means region assignment. For `level = "cousub"` it is required.

`level` Boundary level: `"cousub"` (county subdivisions, default) or `"county"` (county polygons).

`n_regions` For `level = "county"` with automatic k-means assignment: integer number of regions. If `NULL` (default), the count is derived automatically as `max(2, min(6, round(sqrt(n_counties / 8))))`. Ignored when `region_map` is supplied.

`sf_data` Optional pre-projected `sf` polygon object to use instead of downloading from TIGER/Line. Must already have a metric projected CRS. For `level = "county"` with named `region_map`, the object must contain a `NAME` column with county names.

`gamma_r` Regional clearance coefficient (default 3.0)

`gamma_l` Local clearance coefficient (default 1.136)

`p` Distance scaling exponent (default 1.25)

`alpha_r` Optional manual override for regional separation (in metres). May be supplied independently of `alpha_l`.

`alpha_l` Optional manual override for local expansion (in metres). May be supplied independently of `alpha_r`.

`refine` If TRUE, apply a bounded collision-refinement pass after the analytical displacement. Default is FALSE.

`refine_min_gap` Optional minimum boundary-to-boundary gap in map units. If NULL and `refine = TRUE`, uses 2% of the characteristic diameter.

`refine_max_shift` Optional maximum correction per feature in map units. If NULL and `refine = TRUE`, uses 10% of `alpha_r + alpha_l`.

refine_max_iter	Maximum refinement iterations.
refine_step	Fraction of each gap deficit corrected per iteration.
refine_within	Refine pairs within each "region" (default) or across "all" features.
allow_other	If TRUE, permits units mapped to "Other"
fix_invalid	If TRUE, auto-repairs invalid geometries
centroid_fun	"centroid" (default) or "point_on_surface"
plot	Print plots on return (default TRUE). Automatically suppressed when called inside a live Shiny session; use plot.exploded_map() inside renderPlot() instead.
export	NULL (no export), TRUE (auto-named GeoJSON), or a file path
label	Title for plots and print output
quiet	If TRUE, suppress all message() output (useful inside Shiny reactive() and observe() where messages are invisible to users). Default FALSE.

Details

Two boundary levels are supported via the level argument:

- "cousub" (default) — Census county subdivision (COUSUB) units. Best for the northeastern and midwestern states where municipalities tile the state completely. Requires a named region_map.
- "county" — county-level boundaries. Works well for all 50 states. region_map is optional: when omitted, regions are assigned automatically by k-means clustering on county centroids. Pass n_regions to override the automatic cluster count.

You can bypass the TIGER/Line download entirely by passing a pre-projected sf object via sf_data. The state_fips argument is then only used for the default label; it may be omitted if you supply your own label.

Value

An exploded_map S3 object

---

export_topojson *Export an exploded map as TopoJSON*

Description

Converts an sf, exploded_map, or grouped_exploded_map object to TopoJSON using the external mapshaper command-line tool. The input is first written as a temporary GeoJSON file, then converted by mapshaper.

Usage

```
export_topojson(x, file, simplify = NULL, overwrite = FALSE)
```

Arguments

x	An sf, exploded_map, or grouped_exploded_map object. For exploded_map objects, the WGS84 exploded geometry (sf_exp_wgs) is exported. For grouped_exploded_map objects, the WGS84 grouped geometry (sf_grouped_wgs) is exported.
file	Output file path. Should end in .topojson or .json.
simplify	Optional simplification proportion passed to mapshaper -simplify. Must be a single number between 0 and 1 (exclusive). For example, simplify = 0.5 retains 50% of vertices. Default is NULL (no simplification). Note that simplification modifies polygon geometry and therefore breaks the exact geometry preservation guarantee of Proposition 1.
overwrite	Logical; if TRUE, overwrite file if it already exists. Default is FALSE.

Details

This is intended as a convenience helper for downstream tools such as Power BI, D3.js, and Observable that prefer or require TopoJSON input.

This function requires the external mapshaper command-line tool to be installed and available on the system path. It can be installed with:

```
npm install -g mapshaper
```

The mapshaper tool is not an R package dependency; it is invoked via system2(). If mapshaper is not found, the function errors with an informative message.

Value

Invisibly returns the output file path.

Examples

```
## Not run:
# Two-level export
result <- explode_sf(my_sf, region_col = "region", plot = FALSE)
export_topojson(result, "exploded.topojson")

# Three-level grouped export
grp <- explode_grouped(my_sf, region_col = "region", plot = FALSE)
export_topojson(grp, "grouped.topojson")

# With simplification (breaks Proposition 1 guarantee)
export_topojson(result, "simplified.topojson", simplify = 0.5)

# Raw sf object
export_topojson(my_sf, "raw.topojson", overwrite = TRUE)

## End(Not run)
```

focus_map *Interactive focus-map viewer*

Description

[Experimental] Renders spatial features as a smooth, interactive SVG map. Click any polygon to zoom in and lift it from the map with a "toast" effect; right-click or press Escape to reset. Camera transitions use D3's optimal zoom interpolation for fluid 60 fps motion with zero server round-trips.

Usage

```
focus_map(
  x,
  label_col = NULL,
  id_col = NULL,
  group_col = NULL,
  group_palette = NULL,
  context_col = NULL,
  state = NULL,
  restore_selection = FALSE,
  context_values = "context",
  context_mode = c("fade", "hide", "show"),
  coordinate_system = c("auto", "longlat", "planar"),
  context_fill = "#cfd9df",
  context_opacity = 0.18,
  context_clickable = FALSE,
  focus_preset = c("none", "municipal", "drilldown", "municipal_drilldown"),
  simplify = TRUE,
  fill = "#2d6ea3",
  fill_opacity = 0.58,
  stroke = "#ffffff",
  lift_scale = 1.16,
  focus_padding = 40,
  focus_size = 0.76,
  min_focus_width = 0,
  min_focus_height = 0,
  tiny_feature_threshold = 48,
  tiny_feature_boost = 1,
  max_zoom = NULL,
  origin_context = c("none", "socket", "inset", "both"),
 origin_context_position = c("bottom-left", "bottom-right", "top-left", "top-right"),
  focus_context_opacity = 0.3,
  show_drag_zoom = FALSE,
  show_group_labels = FALSE,
  group_labels = NULL,
  font_size = 14,
  show_labels = TRUE,
```

```
  show_sidebar = TRUE,
  performance_mode = NULL,
  info_cols = NULL,
  info_labels = NULL,
  info_title = NULL,
  info_position = c("top-right", "top-left", "bottom-right", "bottom-left"),
  info_card_scale = 1,
  area_min = 5000,
  width_min = 95,
  height_min = 28,
  width = "100%",
  height = "600px",
  elementId = NULL
)

focusmapOutput(outputId, width = "100%", height = "600px")

renderFocusmap(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`x`	An `sf`, `exploded_map`, or `grouped_exploded_map` object.
`label_col`	Character. Column name for polygon labels. Auto-detected if `NULL`.
`id_col`	Optional stable feature ID column for Shiny selection events. Defaults to row order if `NULL`.
`group_col`	Character. Optional column for region/group colouring. Polygons sharing a group value share a hue.
`group_palette`	Optional named character vector of colours for `group_col` values. Names should match group values; unmatched groups fall back to the widget palette.
`context_col`	Optional column identifying features that should remain as geographic context rather than active focus features.
`state`	Optional `dragmapr_state` editorial composition. Region offsets are applied before the widget data is transformed/simplified.
`restore_selection`	When `TRUE` and `state` carries a `selected_feature`, the map opens focused on that feature (matched by `id_col` value or feature name), reproducing a saved composition's focus. Defaults to `FALSE` so the map opens in its normal idle view. No-op when the selection is empty or the feature is not found.
`context_values`	Character vector of values in `context_col` that mark context features. Default `"context"`.
`context_mode`	How context features are drawn: `"fade"` keeps them visible but muted, `"hide"` makes them invisible, and `"show"` draws them normally.
`coordinate_system`	Coordinate system used by the widget. `"longlat"` transforms data to WGS84 and renders with a geographic projection. `"planar"` preserves projected coordinates and renders with a fitted planar projection. `"auto"` currently uses `"longlat"` for backwards compatibility.

`context_fill` Fill colour for context features when `context_mode = "fade"`.

`context_opacity`
Fill opacity for faded context features.

`context_clickable`
Should context features remain clickable? Default `FALSE`.

`focus_preset` Optional named preset for common interactive workflows. `"municipal"` tunes small-area focus, source cues, drag zoom, and dense layer performance. `"drilldown"` tunes context fading and source cues for selected-section maps. `"municipal_drilldown"` combines both. Explicit arguments supplied by the user override preset defaults.

`simplify` Controls geometry simplification for rendering performance. `TRUE` (default) applies a conservative tolerance of approximately one map unit. Longitude/latitude data is simplified in a local metric projection to avoid latitude-dependent distortion. A positive number sets a custom tolerance in the data's coordinate units; longitude/latitude tolerances above 0.1 degrees are rejected as unsafe. `FALSE` disables simplification. Only affects the widget copy — the original data is never modified.

`fill` Fill colour (used when `group_col` is `NULL`). Default `"#2d6ea3"`.

`fill_opacity` Fill opacity. Default `0.58`.

`stroke` Stroke colour. Default `"#ffffff"`.

`lift_scale` Initial toast lift scale. Default `1.16`. Increase this to make the lifted feature larger.

`focus_padding` Extra screen-space padding in pixels around the lifted feature during focus. Increase this if large lifted features feel too close to the map edge.

`focus_size` Target fraction of the map viewport the lifted feature may occupy. Increase this to make selected areas appear larger while preserving `focus_padding`.

`min_focus_width`, `min_focus_height`
Minimum focused feature width and height in screen pixels. When a selected feature is very small, the widget may zoom past the usual density-aware default until the lifted feature reaches these dimensions. Set to `0` to disable either constraint.

`tiny_feature_threshold`
Screen-pixel size below which a selected feature receives an adaptive lift-scale boost. Set to `0` to disable.

`tiny_feature_boost`
Maximum multiplier applied to `lift_scale` for the smallest features. Values below `1` are not allowed.

`max_zoom` Optional maximum camera zoom. If `NULL`, a density-aware default is used.

`origin_context` How the selected feature's source location should be shown while focused. The default `"none"` keeps focus maps visually unchanged unless this feature is explicitly enabled. `"socket"` keeps the source outline in the main map, `"inset"` shows a small overview map, `"both"` uses both, and `"none"` disables the cue.

`origin_context_position`
Position for the overview inset: `"bottom-left"`, `"bottom-right"`, `"top-left"`, or `"top-right"`.

focus_context_opacity
: Fill opacity for non-selected features while a feature is focused. Lower values make tiny selected areas easier to read.

show_drag_zoom
: Show a widget-level drag-zoom toggle. When enabled, users can draw a marquee rectangle to zoom into dense clusters while ordinary feature clicks continue to focus the map. Shift-drag works as a shortcut even when the button is hidden.

show_group_labels
: Show one passive label per `group_col` value on the base map. Labels do not intercept pointer events, so feature click, hover, focus, and drag-zoom behavior are unchanged.

group_labels
: Optional named character vector or list for `show_group_labels`. Names should match `group_col` values; unmatched groups use their raw group value.

font_size
: Label font size in px. Default `14`.

show_labels
: Show labels on lifted shapes? Default `TRUE`.

show_sidebar
: Deprecated and has no effect. Will be removed in a future version.

performance_mode
: Logical or `NULL`. If `NULL`, dense layers automatically use shorter camera transitions and lighter in-flight rendering. Set `TRUE` to force it or `FALSE` to disable it.

info_cols
: Optional character vector of columns to show in a non-blocking focus card when a feature is selected.

info_labels
: Optional named character vector or list for display labels in the focus card. Names should match `info_cols`.

info_title
: Optional column to use as the focus card title. Defaults to `label_col`.

info_position
: Position for the focus card: `"top-right"`, `"top-left"`, `"bottom-right"`, or `"bottom-left"`.

info_card_scale
: Relative size for the focus card. Values above `1` make the card larger; values below `1` make it more compact.

area_min
: Min screen area (px^2) for label visibility.

width_min
: Min screen width (px) for label visibility.

height_min
: Min screen height (px) for label visibility.

width
: Widget width. Default `"100%"`.

height
: Widget height. Default `"600px"`.

elementId
: Optional element ID.

outputId
: Shiny output ID.

expr
: Expression that returns a `focus_map()` widget.

env
: Environment in which to evaluate `expr`.

quoted
: Logical. Is `expr` already quoted?

Details

Accepts raw `sf` objects, `exploded_map` results from `explode_sf`, or `grouped_exploded_map` results from `explode_grouped`. For exploded objects, the displaced (WGS 84) geometry is used automatically.

Value

An `htmlwidgets` object.

Examples

```
poly <- function(xmin, ymin, xmax, ymax) {
  sf::st_polygon(list(rbind(
    c(xmin, ymin), c(xmax, ymin), c(xmax, ymax),
    c(xmin, ymax), c(xmin, ymin)
  )))
}

counties <- sf::st_sf(
  NAME = c("A", "B"),
  region = c("North", "South"),
  geometry = sf::st_sfc(
    poly(-74.2, 40.0, -74.0, 40.2),
    poly(-73.9, 40.0, -73.7, 40.2),
    crs = 4326
  )
)

focus_map(counties, label_col = "NAME", group_col = "region")
```

`focus_map_preset` *Focus-map option presets*

Description

[Experimental] Returns a named list of `focus_map()` arguments for common interactive mapping workflows. Presets are intentionally plain lists so they can be inspected, modified, or passed through `do.call()`.

Usage

```
focus_map_preset(
  name = c("municipal", "drilldown", "municipal_drilldown", "none")
)
```

Arguments

`name`	Preset name. `"municipal"` tunes small-area focus maps with adaptive sizing, source cues, drag zoom, and dense-layer performance. `"drilldown"` tunes selected-section maps with faded context and source cues. `"municipal_drilldown"` combines both.

Value

A named list of focus_map() arguments.

Examples

```
focus_map_preset("municipal")
```

---

focusMapProxy *[Experimental] Focus-map Shiny proxy*

Description

Focus-map Shiny proxy

Usage

```
focusMapProxy(outputId, session = NULL)

update_focus_labels(proxy, show = TRUE)

update_focus_palette(proxy, palette)

update_focus_data(proxy, data, ...)
```

Arguments

outputId	Shiny output ID for an existing focus_map().
session	Shiny session. Defaults to the current reactive domain.
proxy	A focusmap_proxy.
show	Logical; show focus labels.
palette	Named character vector of group colours.
data	New sf, exploded_map, or grouped_exploded_map data.
...	Passed to focus_map() when building the replacement payload.

Details

update_focus_data() swaps the widget's geometry and rebuilds it, which returns the map to its idle (unfocused) view – a data swap is treated as a fresh dataset. To keep the user focused on a feature across the swap, pass a state and restore_selection = TRUE through ...; these flow to focus_map() and the new feature is re-focused on load:

```
update_focus_data(proxy, new_sf, id_col = "GEOID",
                  state = state, restore_selection = TRUE)
```

Value

A focusmap_proxy object.

group_palette — *Create a stable group palette*

Description

Create a stable group palette

Usage

```
group_palette(groups, palette = "Dark 3", existing = NULL)
```

Arguments

groups	Character vector of group names.
palette	Palette name passed to `grDevices::hcl.colors()`.
existing	Optional named colour vector. Existing assignments are retained and colours are generated only for new groups.

Value

A named character vector of colours.

hhs_display_offsets — *HHS reference-layout display offsets*

Description

These offsets are a small documented finishing step for the national HHS reference view. They are interpreted in metres.

Usage

```
hhs_display_offsets()
```

Value

A data frame with `region`, `dx_m`, and `dy_m`.

hhs_focus_map *[Experimental] Create a focus-map widget for HHS regions*

Description

Create a focus-map widget for HHS regions

Usage

```
hhs_focus_map(
  states,
  apply_offsets = TRUE,
  show_region_labels = TRUE,
  show_drag_zoom = TRUE,
  show_labels = TRUE,
  ...
)
```

Arguments

`states`	An `sf` object of state or territory polygons.
`apply_offsets`	Logical. Apply `hhs_display_offsets()` before drawing?
`show_region_labels`	Logical. Show one passive HHS label per region.
`show_drag_zoom`	Logical. Show the drag-to-zoom button.
`show_labels`	Logical. Show labels on lifted states/territories.
`...`	Additional arguments passed to `focus_map()`.

Value

An `htmlwidgets` focus-map widget.

hhs_layout *Build an HHS reference layout*

Description

Build an HHS reference layout

Usage

```
hhs_layout(
  states,
  apply_offsets = TRUE,
  offsets = hhs_display_offsets(),
  source = "user data"
)
```

Arguments

`states`	An `sf` object of state or territory polygons.
`apply_offsets`	Logical. Apply the documented gallery display offsets?
`offsets`	Offset data frame, usually from `hhs_display_offsets()`.
`source`	Optional source label stored in the returned list.

Value

A list with `states`, `labels`, and `source`.

`hhs_palette` *HHS region colour palette*

Description

HHS region colour palette

Usage

```
hhs_palette()
```

Value

A named character vector keyed by region id.

`hhs_region_names` *HHS region display names*

Description

HHS region display names

Usage

```
hhs_region_names(short = FALSE)
```

Arguments

`short`	Logical. When `TRUE`, returns labels such as `"HHS 1"`; otherwise returns labels such as `"1 - Boston"`.

Value

A named character vector keyed by region id.

hhs_regions — *HHS regional assignments for states and territories*

Description

Returns the lookup table used by the explodemap gallery's national HHS view. It includes the 50 states, District of Columbia, Puerto Rico, U.S. Virgin Islands, Guam, American Samoa, and the Northern Mariana Islands.

Usage

```
hhs_regions()
```

Value

A data frame with state name, postal/territory code, region id, and display labels.

layout_children — *Lay out child geography around a parent*

Description

Computes a renderer-neutral child expansion in metres. Child features move radially away from the parent anchor, then a bounded axis-aligned bounding-box or circle solver separates collisions while an attraction term and bearing guard preserve the source geography's mental map. The returned base offsets are algorithmic movement; manual/editorial movement belongs in a `dragmapr_state` and can be composed downstream.

Usage

```
layout_children(
  x,
  id_col,
  parent = NULL,
  parent_geometry = NULL,
  scale = 1.6,
  radial_kick = NULL,
  gap = NULL,
  collision = c("bbox", "circle"),
  attraction = 0.08,
  max_bearing_drift = 0.22,
  bounds = NULL,
  max_iter = 180L
)
```

Arguments

`x`	Projected polygon `sf` object containing child features.
`id_col`	Stable unique feature identifier column.
`parent`	Optional single parent identifier recorded in the output.
`parent_geometry`	Optional parent `sf`, `sfc`, or `sfg` geometry whose representative point anchors the expansion. Defaults to the union of `x`.
`scale`	Radial expansion multiplier applied to source vectors.
`radial_kick`	Additional outward movement in metres. Defaults to two percent of the child layer's bounding-box diagonal.
`gap`	Minimum separation in metres. Defaults to one percent of the child layer's bounding-box diagonal.
`collision`	Collision approximation: feature bounding boxes or centroid circles derived from those boxes.
`attraction`	Fraction of the remaining distance to the geographic target applied during each refinement iteration.
`max_bearing_drift`	Maximum angular drift from the source bearing, in radians.
`bounds`	Optional bounding box supplied as an `sf`/`sfc` object, `st_bbox`, or numeric `c(xmin, ymin, xmax, ymax)`.
`max_iter`	Maximum collision-refinement iterations. Each iteration scans all feature pairs, so cost grows quadratically with the number of features: for large `x`, lower this (the iteration cap is also reduced automatically above 2,000 features) or lay out a subset of children.

Value

An `explodemap_child_layout` with `offsets`, composed `geometry`, source geometry, diagnostics, and parameters. `offsets` contains stable IDs, source/final anchors, base offsets, distance, bearing, bearing drift, and residual collision counts.

`layout_objective` *Build layout objective weights*

Description

Build layout objective weights

Usage

```
layout_objective(
  overlap = 10,
  displacement = 2,
  unused_space = 1,
  label_overlap = 4
)
```

Arguments

`overlap`	Weight for polygon overlap as a fraction of total polygon area.
`displacement`	Weight for mean displacement as a fraction of the original layout diagonal.
`unused_space`	Weight for unused canvas space.
`label_overlap`	Weight for approximate label overlaps as a fraction of all possible label pairs.

Value

Named numeric vector of objective weights.

`layout_offsets` *Computed offsets from a grouped exploded layout*

Description

Returns the base movement created by `explodemap`. These offsets are distinct from manual/editorial offsets stored by `dragmapr`.

Usage

```
layout_offsets(
  layout,
  level = c("group", "feature"),
  id_col = NULL,
  require_stable_id = identical(level, "feature")
)
```

Arguments

`layout`	A `grouped_exploded_map`.
`level`	Either `"group"` or `"feature"`.
`id_col`	Optional feature ID column for `level = "feature"`.
`require_stable_id`	Require a real feature ID column for feature-level persistent handoffs. When `TRUE`, row-number fallback IDs are rejected. Defaults to `TRUE` when `level = "feature"`, `FALSE` otherwise.

Value

A data frame of computed offsets.

layout_regions *Compute region anchor positions for grouped layouts*

Description

Implements the anchor layout procedure from Section 12. Supports three modes: automatic radial placement, automatic with collision resolution, and manual user-specified anchors.

Usage

```
layout_regions(
  sf_obj,
  region_col,
  mode = c("auto", "auto_collision", "manual"),
  anchors = NULL,
  initial_layout = NULL,
  preserve_manual = FALSE,
  kappa = 1.8,
  padding = 50000,
  delta = 15000,
  lambda = 0.18,
  eta = 0.18,
  padding_sep = 20000,
  max_iter = 60,
  quantile_p = 0.85,
  centroid_fun = c("centroid", "point_on_surface"),
  quiet = FALSE
)
```

Arguments

`sf_obj`	Projected sf object with region column
`region_col`	Grouping column name
`mode`	"auto" (radial only), "auto_collision" (radial + solver), or "manual"
`anchors`	For mode = "manual": data.frame with columns (region_col, anchor_x, anchor_y)
`initial_layout`	Optional data.frame with columns (region_col, anchor_x, anchor_y), or a grouped layout object. Used when `preserve_manual = TRUE`.
`preserve_manual`	If `TRUE`, use `initial_layout` as starting anchors before optional collision refinement.
`kappa`	Radial expansion factor (default 1.8)
`padding`	Base padding in map units (default 50000)
`delta`	Log-density scaling factor (default 15000)

lambda	Spring coefficient for collision solver (default 0.18)
eta	Repulsion step size for collision solver (default 0.18)
padding_sep	Minimum separation between blocks (default 20000)
max_iter	Max iterations for collision solver (default 60)
quantile_p	Quantile for block radius estimation (default 0.85)
centroid_fun	"centroid" or "point_on_surface"
quiet	If TRUE, suppress message() output. Default FALSE.

Value

A data.frame with one row per region: the region_col column, source centroid (cx, cy), estimated block_radius, n_units, and the computed anchor_x/anchor_y (plus solver target_x/target_y; the auto_collision mode adds .converged/.iterations).

---

optimize_grouped_layout

Search grouped-layout parameters

Description

Search grouped-layout parameters

Usage

```
optimize_grouped_layout(
  x,
  region_col,
  objective = c("no_overlap", "compact", "balanced", "stable"),
  weights = layout_objective(),
  label_col = NULL,
  label_size = NULL,
  grid = NULL,
  kappa = c(1.4, 1.8, 2.2),
  padding = c(25000, 50000, 75000),
  delta = c(5000, 15000, 30000),
  padding_sep = c(10000, 20000, 40000),
  ...
)
```

Arguments

x	Projected sf object with region column.
region_col	Grouping column name.
objective	Objective presets: "no_overlap", "compact", "balanced", and/or "stable".
weights	Named weights from layout_objective().

`label_col`, `label_size`
: Optional label column and approximate label size. When `label_col` is supplied the search becomes label-aware: each candidate layout is scored for approximate label overlaps (weighted by the `label_overlap` term of `weights`). Left `NULL` (the default), label overlaps contribute zero, so the search result is identical to before.

`grid`
: Optional data.frame of parameter combinations.

`kappa`, `padding`, `delta`, `padding_sep`
: Candidate values used when `grid` is `NULL`.

`...`
: Passed to `explode_grouped()`.

Details

The default grid evaluates 81 parameter combinations, each running a full `explode_grouped()` layout plus diagnostics. Pass a smaller `grid` to cut compute time.

Value

A `grouped_exploded_map` with `optimization` metadata.

`original_geometry` *Access geometry from a grouped exploded layout*

Description

These accessors expose the stable geometry components of a `grouped_exploded_map` without requiring applications to inspect the layout's internal list structure.

Usage

```
original_geometry(layout)

local_geometry(layout)

final_geometry(layout)

group_geometry(layout)
```

Arguments

`layout`
: A `grouped_exploded_map`.

Value

An `sf` object.

plot.exploded_map *Plot an exploded_map object*

Description

Plot an exploded_map object

Usage

```
## S3 method for class 'exploded_map'
plot(x, which = c("exploded", "original", "both"), ...)
```

Arguments

x	An exploded_map object
which	"exploded" (default), "original", or "both"
...	Ignored

Value

Invisibly returns x, the original exploded_map object. The method is called for its side effect of drawing the stored ggplot2 original and/or exploded map objects.

plot.grouped_exploded_map *Plot a grouped_exploded_map object*

Description

Plot a grouped_exploded_map object

Usage

```
## S3 method for class 'grouped_exploded_map'
plot(x, which = c("grouped", "original", "local", "all"), ...)
```

Arguments

x	A grouped_exploded_map object
which	"grouped" (default), "original", "local", or "all"
...	Ignored

Value

Invisibly returns x, the original grouped_exploded_map object. The method is called for its side effect of drawing the stored ggplot2 grouped, original, and/or local map objects.

plot.layout_quality_report

Plot a grouped-layout diagnostic

Description

Renders the region anchors as block circles inside the layout's canvas frame, so unused space is visible as the gap between the blocks and the frame, and draws a red segment between every overlapping region pair. The subtitle summarises overlaps, label overlaps, and canvas utilization.

Usage

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

Arguments

x	A `layout_quality_report` from `diagnose_layout()`.
...	Unused.

Value

A `ggplot` object.

prepare_explodemap_input

Prepare input for grouped explodemap layouts

Description

Standardizes common columns (`unit_id`, `unit_name`, and `region`) while reporting each preparation choice.

Usage

```
prepare_explodemap_input(
  x,
  group_col = NULL,
  id_col = NULL,
  label_col = NULL,
  group_method = c("column", "quadrants", "clusters", "individual"),
  cluster_count = 6L,
  target_crs = NULL,
  make_valid = TRUE,
  target_vertices = NULL,
  seed = 1L
)
```

Arguments

`x`	An `sf` polygon object.
`group_col`, `id_col`, `label_col`	Optional source columns.
`group_method`	Grouping method when `group_col` is not supplied.
`cluster_count`	Number of clusters for generated cluster groups.
`target_crs`	CRS for layout work. Use `NULL` to keep the source CRS.
`make_valid`	Repair geometries with `sf::st_make_valid()`.
`target_vertices`	Optional simplification budget.
`seed`	Seed for generated clusters.

Value

An `explodemap_prepared_input` object with `data`, `report`, `mapping`, and `warnings`.

`print.exploded_map` *Print an exploded_map object*

Description

Print an exploded_map object

Usage

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

Arguments

`x`	An `exploded_map` object
`...`	Ignored

Value

Invisibly returns `x`, the original `exploded_map` object. The method is called for its side effect of printing a compact diagnostic overview of units, regions, derived parameters, and optional refinement information.

print.grouped_exploded_map *Print a grouped_exploded_map object*

Description

Print a grouped_exploded_map object

Usage

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

Arguments

x	A grouped_exploded_map object
...	Ignored

Value

Invisibly returns x, the original grouped_exploded_map object. The method is called for its side effect of printing a compact diagnostic overview of units, regions, grouping mode, and grouped-layout parameters.

read_drag_offsets *Read drag-helper offset CSVs into a region offset data frame*

Description

Reads one or more drag-helper offset files and returns a tidy data frame ready for apply_region_offsets. The drag helper tool exports a CSV named {state}_drag_offsets.csv (e.g. nj_drag_offsets.csv) with three columns: region, dx_m, and dy_m. This function provides a convenient way to load that file – by explicit path, by state abbreviation, or a mix of both – and prints a compact summary so you can verify the offsets before applying them.

Usage

```
read_drag_offsets(x, dir = ".", quiet = FALSE)
```

Arguments

x	A character vector of one or more items, each of which is either: • a path to a CSV file (e.g. "outputs/nj_drag_offsets.csv"), or • a US state abbreviation (e.g. "NJ" or "nj"), which is resolved to {tolower(x)}_drag_offsets.csv inside dir.
dir	Directory to search when an element of x is a state abbreviation rather than a full path. Defaults to the current working directory.
quiet	Logical; when FALSE (default), prints a compact summary of the offsets that were read.

Details

Typical workflow after running the drag helper tool:

```
# 1. Compute the exploded layout
nj <- explode_state("34", crs = 32111, region_map = list(...))

# 2. Read the offsets exported from the drag helper
offsets <- read_drag_offsets("NJ", dir = "outputs/drag_helpers")

# 3. Apply them as a reproducible finishing step
nj_display <- apply_region_offsets(nj, offsets)
```

You can also pass the CSV path directly to apply_region_offsets without calling read_drag_offsets() first – both accept a file path. read_drag_offsets() is most useful when you want to inspect the offsets interactively, load multiple state files at once, or resolve files by abbreviation rather than full path.

Value

A data frame with columns region (character), dx_m (numeric), and dy_m (numeric). When multiple files are supplied an additional state column (the lower-case abbreviation or file stem) is prepended and rows from all files are combined.

See Also

apply_region_offsets

---

simplify_to_vertex_budget

Simplify geometry to a vertex budget

Description

Simplify geometry to a vertex budget

Usage

```
simplify_to_vertex_budget(
  x,
  target_vertices = 350000,
  max_tolerance = NULL,
  preserve_features = TRUE,
  preserve_topology = TRUE
)
```

Arguments

`x`	An `sf` object.
`target_vertices`	Desired maximum vertex count.
`max_tolerance`	Optional maximum simplification tolerance.
`preserve_features`	Require row count, non-empty geometries, and polygon geometry types to be preserved.
`preserve_topology`	Passed to `sf::st_simplify()`.

Value

An `explodemap_simplification` object.

`spatial_join_index` *Build a stable spatial join index*

Description

Creates a small renderer- and source-neutral lookup table for validating external data joins against spatial features. Jurisdiction-specific parsing and alias rules remain application responsibilities.

Usage

```
spatial_join_index(x, id_col, name_col = NULL, parent_col = NULL)
```

Arguments

`x`	Spatial or ordinary data frame.
`id_col`	Stable unique feature identifier column.
`name_col`	Optional display-name column.
`parent_col`	Optional parent identifier column.

Value

A `spatial_join_index` data frame with `feature_id`, optional `feature_name` and `parent_id`, and the original feature row.

---

summary.exploded_map *Summary of an exploded_map object*

Description

Summary of an exploded_map object

Usage

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

Arguments

`object`	An `exploded_map` object
`...`	Ignored

Value

Invisibly returns `object`, the original `exploded_map` object. The method is called for its side effect of printing a human-readable summary of dataset size, geometry statistics, displacement parameters, refinement diagnostics, and implied calibration coefficients.

---

summary.grouped_exploded_map *Summary of a grouped_exploded_map object*

Description

Summary of a grouped_exploded_map object

Usage

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

Arguments

`object`	A `grouped_exploded_map` object
`...`	Ignored

Value

Invisibly returns `object`, the original `grouped_exploded_map` object. The method is called for its side effect of printing dataset size, grouping diagnostics, anchor parameters, and anchor radii.

`transition_data` *Renderer-neutral transition data*

Description

Describes original-to-final movement for groups or features. Rendering systems can use this table for SVG, MapLibre, deck.gl, or static animation plans without inspecting layout internals.

Usage

```
transition_data(
  layout,
  level = c("group", "feature"),
  id_col = NULL,
  ordering = NULL,
  require_stable_id = identical(level, "feature")
)
```

Arguments

`layout`	A `grouped_exploded_map`.
`level`	Either `"group"` or `"feature"`.
`id_col`	Optional feature ID column for `level = "feature"`.
`ordering`	Optional ordering column for `level = "feature"`.
`require_stable_id`	Require a real feature ID column for feature-level persistent handoffs. When `TRUE`, row-number fallback IDs are rejected. Defaults to `TRUE` when `level = "feature"`, `FALSE` otherwise.

Value

A data frame with movement, distance, angle, and animation order.

update_exploded_layout

Update an exploded grouped layout after manual composition

Description

Update an exploded grouped layout after manual composition

Usage

```
update_exploded_layout(
  result,
  region_offsets,
  label_offsets = NULL,
  update_plots = TRUE
)
```

Arguments

result	A grouped_exploded_map.
region_offsets	Data frame of manual anchors (anchor_x, anchor_y) or offsets (dx_m, dy_m) by region.
label_offsets	Optional label-offset table stored on the result.
update_plots	Rebuild stored ggplots.

Value

Updated grouped_exploded_map.

validate_explodemap_input

Validate explodemap input

Description

Validate explodemap input

Usage

```
validate_explodemap_input(
  x,
  group_col,
  id_col = NULL,
  label_col = NULL,
  max_features = Inf,
  max_groups = Inf,
  max_vertices = Inf,
  min_groups = 2L
)
```

Arguments

`x`	An `sf` object.
`group_col`	Grouping column.
`id_col`	Optional unique feature ID column.
`label_col`	Optional feature label column.
`max_features`, `max_groups`, `max_vertices`	Optional limits.
`min_groups`	Minimum number of groups required.

Value

An `explodemap_validation` object.

`validate_spatial_join` *Validate an external data-to-spatial join*

Description

Reports exact stable-ID matches and separates unmatched, duplicated, ambiguous-name, and wrong-parent rows. It does not mutate or join the input; applications remain in control of policy for aliases and fallbacks.

Usage

```
validate_spatial_join(data, index, key_col, name_col = NULL, parent_col = NULL)
```

Arguments

`data`	External data frame to validate.
`index`	A `spatial_join_index()` object.
`key_col`	Stable identifier column in `data`.

`name_col`	Optional display-name column in `data`, used only to report ambiguous fallback names.
`parent_col`	Optional parent identifier column in `data`, compared with `index$parent_id` for exact key matches.

Value

A `spatial_join_validation` list containing `matched`, `unmatched`, `ambiguous`, `duplicate`, `wrong_parent`, and summary counts.

Index

anchor_table, 3
apply_region_offsets, 4, *45, 46*
as_dragmapr, 5
as_dragmapr(), *6*
as_dragmapr_state, 6
as_dragmapr_state(), *5*
as_hhs_states, 6
as_sf, 7
assign_spatial_groups, 8

cache_clear, 9
cache_list, 9
calibration_row, 10
child_layout_geometry
(child_layout_offsets), 10
child_layout_offsets, 10
compare_layouts, 11
compare_layouts(), *15*
compute_stats, 12
compute_stats(), *14*
connector_geometry, 12
count_geometry_vertices, 13

derive_params, 14
diagnose_layout, 14
diagnose_layout(), *43*
displacement_magnitudes, 15

e_fingerprint, 16
estimate_block_radii, 16
explode_grouped, 17, *30*
explode_grouped(), *20*, *41*
explode_section, 19
explode_sf, 21, *30*
explode_sf(), *20*, *23*
explode_sf_with_lookup, 22
explode_state, 23
export_topojson, 25

final_geometry (original_geometry), 41

focus_map, 27
focus_map(), *19*, *31*, *32*, *34*
focus_map_preset, 31
focusmapOutput (focus_map), 27
focusMapProxy, 32

grDevices::hcl.colors(), *33*
group_geometry (original_geometry), 41
group_palette, 33

hhs_display_offsets, 33
hhs_focus_map, 34
hhs_layout, 34
hhs_palette, 35
hhs_region_names, 35
hhs_regions, 36

layout_children, 36
layout_children(), *10*
layout_objective, 37
layout_objective(), *40*
layout_offsets, 38
layout_regions, 39
local_geometry (original_geometry), 41

optimize_grouped_layout, 40
original_geometry, 41

plot.exploded_map, 42
plot.exploded_map(), *22*, *25*
plot.grouped_exploded_map, 42
plot.grouped_exploded_map(), *19*
plot.layout_quality_report, 43
prepare_explodemap_input, 43
print.exploded_map, 44
print.grouped_exploded_map, 45

read_drag_offsets, 45
renderFocusmap (focus_map), 27

sf::st_geometry(), *13*

sf::st_make_valid(), *44*
sf::st_simplify(), *47*
simplify_to_vertex_budget, 46
spatial_join_index, 47
spatial_join_index(), *51*
summary.exploded_map, 48
summary.grouped_exploded_map, 48

transition_data, 49

update_exploded_layout, 50
update_focus_data *(*focusMapProxy*)*, 32
update_focus_labels *(*focusMapProxy*)*, 32
update_focus_palette *(*focusMapProxy*)*, 32

validate_explodemap_input, 50
validate_spatial_join, 51