Package 'spatcovar'

September 8, 2026

Title Construct Spatial Covariates from Polygon Data

Version 0.1.0

Description Provides a consistent interface for constructing commonly used spatial covariates from polygon data. Computes polygon areas, distances to reference features, point and line intersection counts, line lengths within polygons, polygon overlap areas and shares, and raster zonal summaries. Handles coordinate reference system validation, geometry repair, unit conversion, row preservation, and standardised missing value semantics while relying on established spatial libraries for the underlying geometry operations.

License MIT + file LICENSE

URL https://github.com/emre-cebeci/spatcovar

BugReports https://github.com/emre-cebeci/spatcovar/issues

Encoding UTF-8

Imports exactextractr (>= 0.9.0), sf (>= 1.0.0), terra, units

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Emre Cebeci [aut, cre]

Maintainer Emre Cebeci <cebeciemre1@gmail.com>

Repository CRAN

Date/Publication 2026-09-08 13:30:02 UTC

Contents

`example_grid` *Example source polygon grid*

Description

Returns a small `sf` object with three overlapping rectangular zones. Each zone has a numeric `value` attribute for use in overlap examples. All geometries use EPSG:32632 (UTM zone 32N).

Usage

```
example_grid()
```

Value

An `sf` object with 3 polygons, a `zone_id` column, and a `value` column.

Examples

```
example_grid()
```

`example_lines` *Example line features*

Description

Returns a small `sf` object with four synthetic line features. Some lines cross polygon boundaries; others are contained within or outside the example regions. All geometries use EPSG:32632 (UTM zone 32N).

Usage

```
example_lines()
```

Value

An `sf` object with 4 lines and a `route_id` column.

Examples

```
example_lines()
```

`example_points` *Example point features*

Description

Returns a small `sf` object with 15 synthetic point locations distributed across the example polygon regions. All geometries use EPSG:32632 (UTM zone 32N).

Usage

```
example_points()
```

Value

An `sf` object with 15 points and a `site_id` column.

Examples

```
example_points()
```

`example_polygons` *Example polygon regions*

Description

Returns a small `sf` object with six synthetic polygon regions for use in examples and tests. All geometries use EPSG:32632 (UTM zone 32N).

Usage

```
example_polygons()
```

Value

An `sf` object with 6 polygons and a `name` column.

Examples

```
example_polygons()
```

example_raster *Example raster*

Description

Returns a small single-layer terra::SpatRaster with 16 rows and 20 columns of known values covering the extent of `example_polygons()`. Includes some `NA` cells in the upper-right corner and valid zero values.

Usage

```
example_raster()
```

Details

The raster uses EPSG:32632 (UTM zone 32N) with bounding box `xmin = 398000, xmax = 432000, ymin = 5398000, ymax` (1700 m in X by 1500 m in Y cell resolution).

Value

A single-layer terra::SpatRaster object.

Examples

```
example_raster()
```

spat_area *Compute polygon area*

Description

Calculates the area of each polygon in an `sf` object and appends the result as a new column.

Usage

```
spat_area(
  x,
  unit = "km2",
  name = NULL,
  crs = NULL,
  repair = TRUE,
  diagnostics = FALSE,
  overwrite = FALSE
)
```

Arguments

x	An sf object with POLYGON or MULTIPOLYGON geometry. Must have a known coordinate reference system.
unit	Character. Area unit for the output: "m2", "km2" (default), "ha", or "mi2".
name	Character or NULL. Name of the output column. If NULL (default), the column is named dynamically based on unit (e.g., "area_km2", "area_m2", "area_ha", "area_mi2").
crs	Optional. A CRS specification (EPSG code, WKT, or proj4string) to project to before computing area. If NULL (default), the input CRS is used.
repair	Logical. If TRUE (default), invalid geometries are repaired on an internal copy before computation. The returned geometry is always the original target geometry.
diagnostics	Logical. If TRUE, attach a diagnostics summary as an attribute. Default is FALSE.
overwrite	Logical. If TRUE, overwrite an existing column named name. Default is FALSE.

Value

The input sf object with a new numeric column containing area values in the requested unit. Row count, row order, and the original geometry are preserved.

CRS behaviour

For geographic (longitude/latitude) coordinate reference systems, spat_area() uses sf::st_area() which computes geodesic area via the s2 geometry library. For projected CRS the computation is planar. If crs is supplied, an internal copy is projected to that CRS before computing area.

Examples

```
regions <- example_polygons()
spat_area(regions)
spat_area(regions, unit = "ha")
spat_area(regions, unit = "m2", name = "custom_area")
```

---

spat_count *Count source features intersecting each polygon*

Description

Counts the number of source features in y that spatially intersect each target polygon in x.

Usage

```
spat_count(
  x,
  y,
  name = "count",
  repair = TRUE,
  diagnostics = FALSE,
  overwrite = FALSE
)
```

Arguments

x	An sf object with POLYGON or MULTIPOLYGON geometry. Must have a known coordinate reference system.
y	An sf object containing source features (any geometry type). Must have a known coordinate reference system.
name	Character. Name of the output column. Default is "count".
repair	Logical. If TRUE (default), invalid geometries are repaired on internal copies.
diagnostics	Logical. If TRUE, attach a diagnostics summary as an attribute.
overwrite	Logical. If TRUE, overwrite an existing column named name.

Value

The input sf object with a new integer column containing the count of intersecting source features. Polygons with no intersecting features receive 0L.

Intersection semantics

A source feature is counted if it spatially intersects the target polygon, which includes features that touch the boundary. The function uses sf::st_intersects() internally.

Each feature in y counts as one source feature regardless of its sub-geometry cardinality. A MULTIPOINT feature with 10 sub-points is one source feature. Cast to POINT with sf::st_cast() before calling spat_count() if per-point counts are desired. The same principle applies to MULTILINESTRING and MULTIPOLYGON.

Both x and y must have known coordinate reference systems.

Examples

```
regions <- example_polygons()
sites <- example_points()
spat_count(regions, sites)
```

spat_diagnostics *Retrieve diagnostics from a spatcovar result*

Description

Extracts the diagnostics summary attached by a `spat_*()` function when called with `diagnostics = TRUE`.

Usage

```
spat_diagnostics(x)
```

Arguments

`x`	An `sf` object returned by a `spat_*()` function with `diagnostics = TRUE`.

Value

A named list with operation details, or `NULL` if no diagnostics are attached. When printed, produces a human-readable summary.

Diagnostic attributes

Diagnostics represent the most recent `spatcovar` operation performed on the object. Intermediate transformations (subsetting, column manipulation, joins) may drop the attribute; this is expected behaviour and not an error.

Examples

```
regions <- example_polygons()
result <- spat_area(regions, diagnostics = TRUE)
spat_diagnostics(result)
```

spat_distance *Compute distance from polygons to reference features*

Description

Calculates the distance from each polygon in `x` to the reference features in `y` and appends the result as a new column.

Usage

```
spat_distance(
  x,
  y,
  method = "minimum",
  unit = "km",
  name = NULL,
  crs = NULL,
  repair = TRUE,
  diagnostics = FALSE,
  overwrite = FALSE
)
```

Arguments

x	An sf object with POLYGON or MULTIPOLYGON geometry. Must have a known coordinate reference system.
y	An sf object containing the reference features (any geometry type). Must have a known coordinate reference system.
method	Character. Distance method: "minimum" (default), "centroid", or "point_on_surface".
unit	Character. Distance unit: "m", "km" (default), or "mi".
name	Character or NULL. Name of the output column. If NULL (default), the column is named dynamically based on unit (e.g., "dist_km", "dist_m", "dist_mi").
crs	Optional. A CRS specification to project to before computing distances.
repair	Logical. If TRUE (default), invalid geometries are repaired on internal copies.
diagnostics	Logical. If TRUE, attach a diagnostics summary as an attribute.
overwrite	Logical. If TRUE, overwrite an existing column named name.

Value

The input sf object with a new numeric column containing distances in the requested unit.

Distance methods

"minimum" Minimum geometry-to-geometry distance. For each polygon, this is the shortest distance to the nearest feature in y. Implemented using sf::st_nearest_feature() to identify the nearest candidate, then sf::st_distance() with by_element = TRUE. This avoids constructing a full pairwise distance matrix.

"centroid" Distance from the centroid of each polygon in x to the nearest feature in y.

"point_on_surface" Distance from a guaranteed-interior point of each polygon in x to the nearest feature in y.

CRS behaviour

For geographic CRS, `sf::st_distance()` computes geodesic great-circle distances via the s2 geometry library. For projected CRS the computation is Euclidean. If `crs` is supplied, internal copies of both `x` and `y` are projected before computing distances. Both `x` and `y` must have known coordinate reference systems.

Examples

```
regions <- example_polygons()
sites <- example_points()
spat_distance(regions, sites)
spat_distance(regions, sites, unit = "m")
spat_distance(regions, sites, method = "centroid", name = "centroid_dist_km")
```

`spat_length` *Compute total line length within each polygon*

Description

Clips line features in `y` to each target polygon in `x` and computes the total length of clipped line segments.

Usage

```
spat_length(
  x,
  y,
  unit = "km",
  name = NULL,
  crs = NULL,
  repair = TRUE,
  diagnostics = FALSE,
  overwrite = FALSE
)
```

Arguments

`x`	An `sf` object with POLYGON or MULTIPOLYGON geometry. Must have a known coordinate reference system.
`y`	An `sf` object with LINESTRING or MULTILINESTRING geometry. Must have a known coordinate reference system.
`unit`	Character. Length unit: `"m"`, `"km"` (default), or `"mi"`.
`name`	Character or `NULL`. Name of the output column. If `NULL` (default), the column is named dynamically based on `unit` (e.g., `"length_km"`, `"length_m"`, `"length_mi"`).

crs	Optional. A projected CRS specification to project to before clipping and measuring length. Required when x has a geographic CRS.
repair	Logical. If TRUE (default), invalid geometries are repaired on internal copies.
diagnostics	Logical. If TRUE, attach a diagnostics summary as an attribute.
overwrite	Logical. If TRUE, overwrite an existing column named name.

Value

The input sf object with a new numeric column containing total line length in the requested unit. Polygons with no intersecting lines receive 0.

CRS requirement

spat_length() requires a projected (planar) coordinate reference system for accurate length measurement. If x has a geographic CRS and crs is not supplied, the function raises an error asking the user to supply a projected CRS. If crs is supplied, it must also be a projected (planar) CRS. Both x and y must have known coordinate reference systems.

Examples

```
regions <- example_polygons()
routes <- example_lines()
spat_length(regions, routes)
spat_length(regions, routes, unit = "m")
```

---

spat_overlap *Compute polygon-polygon overlap measures*

Description

Calculates overlap between target polygons in x and source polygons in y. Returns one of three measures: total overlap area, share of target area covered, or count of overlapping source features.

Usage

```
spat_overlap(
  x,
  y,
  measure = "area",
  unit = "km2",
  name = NULL,
  crs = NULL,
  repair = TRUE,
  diagnostics = FALSE,
  overwrite = FALSE
)
```

Arguments

x	An sf object with POLYGON or MULTIPOLYGON geometry. Must have a known coordinate reference system.
y	An sf object with POLYGON or MULTIPOLYGON geometry (source polygons). Must have a known coordinate reference system.
measure	Character. Overlap measure: "area" (default), "share", or "count".
unit	Character. Area unit for the "area" measure: "m2", "km2" (default), "ha", or "mi2". Ignored for other measures.
name	Character or NULL. Name of the output column. If NULL (default), the column is named dynamically: "overlap_{unit}" for area (e.g., "overlap_km2", "overlap_m2"), "overlap_share" for share, and "overlap_count" for count.
crs	Optional. A CRS specification to project to before computing overlap area.
repair	Logical. If TRUE (default), invalid geometries are repaired on internal copies.
diagnostics	Logical. If TRUE, attach a diagnostics summary as an attribute.
overwrite	Logical. If TRUE, overwrite an existing column named name.

Value

The input sf object with a new column containing the requested overlap measure.

Overlap measures

"area" Total area of overlap between each target polygon and all source polygons. When source polygons overlap each other, overlap is summed without deduplication.

"share" Fraction of each target polygon's area that is covered by source polygons. A warning is issued if source polygons have positive-area duplicate coverage among themselves, because the share can exceed 1.

"count" Number of source features that spatially intersect each target polygon.

CRS behaviour

For the "area" and "share" measures, overlap area is computed using sf::st_area() which handles geodesic area on geographic CRS. If crs is supplied, internal copies are projected first. Both x and y must have known coordinate reference systems.

Examples

```
regions <- example_polygons()
zones <- example_grid()
spat_overlap(regions, zones)
spat_overlap(regions, zones, unit = "ha")
spat_overlap(regions, zones, measure = "share", name = "zone_coverage")
```

spat_raster *Compute raster zonal statistics for each polygon*

Description

Extracts zonal statistics from a single-layer raster for each polygon in `x` using the exactextractr engine for coverage-fraction-weighted extraction.

Usage

```
spat_raster(
  x,
  raster,
  stats = "mean",
  name = NULL,
  repair = TRUE,
  diagnostics = FALSE,
  overwrite = FALSE
)
```

Arguments

`x`	An `sf` object with POLYGON or MULTIPOLYGON geometry. Must have a known coordinate reference system.
`raster`	A file path (character) or single-layer terra::SpatRaster object. Must have a known coordinate reference system.
`stats`	Character vector. Statistics to compute. Allowed values: `"mean"`, `"median"`, `"min"`, `"max"`, `"sum"`, `"count"`, `"stdev"`. Default is `"mean"`.
`name`	Character or `NULL`. Prefix for output column names. Output columns are named `"{name}_{stat}"` when `name` is supplied, or `"{stat}"` when `name` is `NULL` (default).
`repair`	Logical. If `TRUE` (default), invalid geometries are repaired on internal copies.
`diagnostics`	Logical. If `TRUE`, attach a diagnostics summary as an attribute.
`overwrite`	Logical. If `TRUE`, overwrite existing columns.

Value

The input `sf` object with new numeric columns for each requested statistic.

Extraction semantics

Statistics are computed using `exactextractr::exact_extract()` which weights pixel contributions by the fraction of each pixel covered by the polygon (the coverage fraction):

`"mean"` Mean cell value, weighted by the fraction of each cell covered by the polygon.

`"sum"` Sum of non-NA cell values, multiplied by the fraction of each cell covered by the polygon.

`"median"` Median cell value, weighted by the fraction of each cell covered by the polygon.

`"min"` Minimum non-NA value across all cells wholly or partially covered by the polygon (unweighted).

`"max"` Maximum non-NA value across all cells wholly or partially covered by the polygon (unweighted).

`"count"` Sum of fractions of raster cells with non-NA values covered by the polygon (the effective number of covered cells). This value can be fractional for polygons that partially overlap raster cells.

`"stdev"` Population standard deviation of cell values, weighted by the fraction of each cell covered by the polygon.

Valid raster values of zero are preserved as valid observations. If a target polygon has no valid non-NA raster cells contributing to the extraction (either because it lies outside the raster extent or because all covered cells are NA/NoData), all requested statistics for that polygon evaluate to `NA`.

Multilayer rasters

`spat_raster()` supports single-layer rasters only in this release. To extract statistics from multiple layers, supply one raster layer at a time.

CRS handling

Both `x` and `raster` must have known coordinate reference systems. If the polygon CRS differs from the raster CRS, polygon copies are transformed to match the raster CRS before extraction.

Examples

```
regions <- example_polygons()
rst <- example_raster()
spat_raster(regions, rst, stats = c("mean", "min", "max"))
spat_raster(regions, rst, stats = "count", name = "cell")
```

Index