Package 'rmapshaper'

December 4, 2025

Type Package

Title Client for 'mapshaper' for 'Geospatial' Operations

Version 0.6.0

Description Edit and simplify 'geojson', 'Spatial', and 'sf' objects. This is wrapper around the 'mapshaper' 'JavaScript' library by Matthew Bloch <https://github.com/mbloch/mapshaper/> to perform topologically-aware polygon simplification, as well as other operations such as clipping, erasing, dissolving, and converting 'multi-part' to 'single-part' geometries.

License MIT + file LICENSE

URL https://github.com/ateucher/rmapshaper, http://andyteucher.ca/rmapshaper/

BugReports https://github.com/ateucher/rmapshaper/issues

Imports methods, geojsonsf (>= 2.0.5), jsonify (>= 1.2.3), readr (>= 2.1.0), sf (>= 1.0.0), sp (>= 1.4-0), V8 (>= 6.0.0)

Suggests knitr, magrittr, rmarkdown, testthat (>= 3.0.0), jsonlite, covr, units, withr

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.3

Config/testthat/edition 3

Config/testthat/parallel TRUE

NeedsCompilation no

Author Andy Teucher [aut, cre] (ORCID: <https://orcid.org/0000-0002-7840-692X>), Kenton Russell [aut] (JavaScript support), Matthew Bloch [cph] (mapshaper Javascript library)

Maintainer Andy Teucher <andy.teucher@gmail.com>

Repository CRAN

Date/Publication 2025-12-04 12:50:08 UTC

Contents

`apply_mapshaper_commands`
Apply a mapshaper command string to a geojson object

Description

Apply a mapshaper command string to a geojson object

Usage

```
apply_mapshaper_commands(
  data,
  command,
  force_FC = TRUE,
  sys = FALSE,
  sys_mem = getOption("mapshaper.sys_mem", default = 8),
  quiet = getOption("mapshaper.sys_quiet", default = FALSE),
  gj2008 = FALSE
)
```

Arguments

`data`	character containing geojson or path to geojson file. If a file path, `sys` must be true.
`command`	valid mapshaper command string
`force_FC`	should the output be forced to be a FeatureCollection (or sf object or Spatial*DataFrame) even if there are no attributes? Default `TRUE`. If FALSE and there are no attributes associated with the geometries, a GeometryCollection (or Spatial object with no dataframe, or sfc) will be output.

sys	Should the system mapshaper be used instead of the bundled mapshaper? Gives better performance on large files. Requires the mapshaper node package to be installed and on the PATH.
sys_mem	How much memory (in GB) should be allocated if using the system mapshaper (sys = TRUE)? Default 8. Ignored if sys = FALSE. This can also be set globally with the option "mapshaper.sys_mem"
quiet	If sys = TRUE, should the mapshaper messages be silenced? Default FALSE. This can also be set globally with the option "mapshaper.sys_quiet"
gj2008	Generate output that is consistent with the pre-RFC 7946 GeoJSON spec (dating to 2008). Polygon rings are CW and holes are CCW, which is the opposite of the default RFC 7946-compatible output. This should be rarely needed, though may be useful when preparing data for D3-based data visualizations (such as plotly::plot_ly()). Default FALSE

Value

geojson

Examples

```
nc <- sf::read_sf(system.file("gpkg/nc.gpkg", package = "sf"))
rmapshaper::apply_mapshaper_commands(geojsonsf::sf_geojson(nc), "-clean")
```

---

check_sys_mapshaper *Check the system mapshaper*

Description

Check the system mapshaper

Usage

```
check_sys_mapshaper(command = "mapshaper-xl", verbose = TRUE)
```

Arguments

command	either "mapshaper-xl" (default) or "mapshaper"
verbose	Print a message stating mapshaper's current version? Default TRUE

Value

character path to mapshaper executable if appropriate version is installed, otherwise throws an error

drop_null_geometries *Drop features from a* geo_json *FeatureCollection with null geometries*

Description

Drop features from a geo_json FeatureCollection with null geometries

Usage

```
drop_null_geometries(x)
```

Arguments

x	a geo_json FeatureCollection

Value

a geo_json FeatureCollection with Features with null geometries removed

ms_clip *Remove features or portions of features that fall outside a clipping area.*

Description

Removes portions of the target layer that fall outside the clipping layer or bounding box.

Usage

```
ms_clip(target, clip = NULL, bbox = NULL, remove_slivers = FALSE, ...)
```

Arguments

target	the target layer from which to remove portions. One of: • geo_json or character points, lines, or polygons; • SpatialPolygons, SpatialLines, SpatialPoints; • sf or sfc points, lines, or polygons object
clip	the clipping layer (polygon). One of: • geo_json or character polygons; • SpatialPolygons*; • sf or sfc polygons object
bbox	supply a bounding box instead of a clipping layer to extract from the target layer. Supply as a numeric vector: c(minX, minY, maxX, maxY).

`remove_slivers` Remove tiny sliver polygons created by clipping. (Default `FALSE`)

`...` Arguments passed on to `apply_mapshaper_commands`

`force_FC` should the output be forced to be a FeatureCollection (or sf object or Spatial*DataFrame) even if there are no attributes? Default `TRUE`. If FALSE and there are no attributes associated with the geometries, a GeometryCollection (or Spatial object with no dataframe, or sfc) will be output.

`sys` Should the system mapshaper be used instead of the bundled mapshaper? Gives better performance on large files. Requires the mapshaper node package to be installed and on the PATH.

`sys_mem` How much memory (in GB) should be allocated if using the system mapshaper (`sys = TRUE`)? Default 8. Ignored if `sys = FALSE`. This can also be set globally with the option `"mapshaper.sys_mem"`

`quiet` If `sys = TRUE`, should the mapshaper messages be silenced? Default `FALSE`. This can also be set globally with the option `"mapshaper.sys_quiet"`

`gj2008` Generate output that is consistent with the pre-RFC 7946 GeoJSON spec (dating to 2008). Polygon rings are CW and holes are CCW, which is the opposite of the default RFC 7946-compatible output. This should be rarely needed, though may be useful when preparing data for D3-based data visualizations (such as `plotly::plot_ly()`). Default `FALSE`

Value

clipped target in the same class as the input target

Examples

```
if (rmapshaper:::v8_version() >= "6") {
  library(geojsonsf, quietly = TRUE)
  library(sf)

  poly <- structure("{\"type\":\"FeatureCollection\",
    \"features\":[{\"type\":\"Feature\",\"properties\":{},
    \"geometry\":{\"type\":\"Polygon\",\"coordinates\":
    [[[52.8658,-44.7219],[53.7702,-40.4873],[55.3204,-37.5579],
    [56.2757,-37.917],[56.184,-40.6443],[61.0835,-40.7529],
    [58.0202,-43.634],[61.6699,-45.0678],[62.737,-46.2841],
    [55.7763,-46.2637],[54.9742,-49.1184],[52.799,-45.9386],
    [52.0329,-49.5677],[50.1747,-52.1814],[49.0098,-52.3641],
    [52.7068,-45.7639],[43.2278,-47.1908],[48.4755,-45.1388],
    [50.327,-43.5207],[48.0804,-41.2784],[49.6307,-40.6159],
    [52.8658,-44.7219]]]}}]}", class = c("geojson", "json"))
  poly <- geojson_sf(poly)
  plot(poly)

  clip_poly <- structure('{
  "type": "Feature",
  "properties": {},
  "geometry": {
  "type": "Polygon",
  "coordinates": [
```

```
  [
  [51, -40],
  [55, -40],
  [55, -45],
  [51, -45],
  [51, -40]
  ]
  ]
  }
  }', class = c("geojson", "json"))
  clip_poly <- geojson_sf(clip_poly)
  plot(clip_poly)

  out <- ms_clip(poly, clip_poly)
  plot(out, add = TRUE)
}
```

`ms_dissolve` *Aggregate shapes in a polygon or point layer.*

Description

Aggregates using specified field, or all shapes if no field is given. For point layers, replaces a group of points with their centroid.

Usage

```
ms_dissolve(
  input,
  field = NULL,
  sum_fields = NULL,
  copy_fields = NULL,
  weight = NULL,
  snap = TRUE,
  ...
)
```

Arguments

`input`	spatial object to dissolve. One of: • `geo_json` or `character` points or polygons; • `SpatialPolygons`, or `SpatialPoints`
`field`	the field to dissolve on
`sum_fields`	fields to sum
`copy_fields`	fields to copy. The first instance of each field will be copied to the aggregated feature.

weight Name of an attribute field for generating weighted centroids (points only).

snap Snap together vertices within a small distance threshold to fix small coordinate misalignment in adjacent polygons. Default TRUE.

... Arguments passed on to apply_mapshaper_commands

force_FC should the output be forced to be a FeatureCollection (or sf object or Spatial*DataFrame) even if there are no attributes? Default TRUE. If FALSE and there are no attributes associated with the geometries, a GeometryCollection (or Spatial object with no dataframe, or sfc) will be output.

sys Should the system mapshaper be used instead of the bundled mapshaper? Gives better performance on large files. Requires the mapshaper node package to be installed and on the PATH.

sys_mem How much memory (in GB) should be allocated if using the system mapshaper (sys = TRUE)? Default 8. Ignored if sys = FALSE. This can also be set globally with the option "mapshaper.sys_mem"

quiet If sys = TRUE, should the mapshaper messages be silenced? Default FALSE. This can also be set globally with the option "mapshaper.sys_quiet"

gj2008 Generate output that is consistent with the pre-RFC 7946 GeoJSON spec (dating to 2008). Polygon rings are CW and holes are CCW, which is the opposite of the default RFC 7946-compatible output. This should be rarely needed, though may be useful when preparing data for D3-based data visualizations (such as plotly::plot_ly()). Default FALSE

Value

the same class as the input

Examples

```
library(geojsonsf)
library(sf)

poly <- structure('{"type":"FeatureCollection",
  "features":[
  {"type":"Feature",
  "properties":{"a": 1, "b": 2},
  "geometry":{"type":"Polygon","coordinates":[[
  [102,2],[102,3],[103,3],[103,2],[102,2]
  ]]}}
  ,{"type":"Feature",
  "properties":{"a": 5, "b": 3},
  "geometry":{"type":"Polygon","coordinates":[[
  [100,0],[100,1],[101,1],[101,0],[100,0]
  ]]}}]}', class = c("geojson", "json"))
poly <- geojson_sf(poly)
plot(poly)
length(poly)
poly

# Dissolve the polygon
```

```
out <- ms_dissolve(poly)
plot(out)
length(out)
out

# Dissolve and summing columns
out <- ms_dissolve(poly, sum_fields = c("a", "b"))
plot(out)
out
```

`ms_erase` *Remove features or portions of features that fall inside a specified area*

Description

Removes portions of the target layer that fall inside the erasing layer or bounding box.

Usage

```
ms_erase(target, erase = NULL, bbox = NULL, remove_slivers = FALSE, ...)
```

Arguments

`target`	the target layer from which to remove portions. One of: • `geo_json` or `character` points, lines, or polygons; • `SpatialPolygons`, `SpatialLines`, `SpatialPoints`
`erase`	the erase layer (polygon). One of: • `geo_json` or `character` polygons; • `SpatialPolygons*`
`bbox`	supply a bounding box instead of an erasing layer to remove from the target layer. Supply as a numeric vector: `c(minX, minY, maxX, maxY)`.
`remove_slivers`	Remove tiny sliver polygons created by erasing. (Default `FALSE`)
`...`	Arguments passed on to `apply_mapshaper_commands` `force_FC` should the output be forced to be a FeatureCollection (or sf object or Spatial*DataFrame) even if there are no attributes? Default `TRUE`. If FALSE and there are no attributes associated with the geometries, a GeometryCollection (or Spatial object with no dataframe, or sfc) will be output. `sys` Should the system mapshaper be used instead of the bundled mapshaper? Gives better performance on large files. Requires the mapshaper node package to be installed and on the PATH. `sys_mem` How much memory (in GB) should be allocated if using the system mapshaper (`sys = TRUE`)? Default 8. Ignored if `sys = FALSE`. This can also be set globally with the option `"mapshaper.sys_mem"`

quiet If sys = TRUE, should the mapshaper messages be silenced? Default FALSE. This can also be set globally with the option "mapshaper.sys_quiet"

gj2008 Generate output that is consistent with the pre-RFC 7946 GeoJSON spec (dating to 2008). Polygon rings are CW and holes are CCW, which is the opposite of the default RFC 7946-compatible output. This should be rarely needed, though may be useful when preparing data for D3-based data visualizations (such as plotly::plot_ly()). Default FALSE

Value

erased target in the same format as the input target

Examples

```
if (rmapshaper:::v8_version() >= "6") {
  library(geojsonsf, quietly = TRUE)
  library(sf)

  points <- structure("{\"type\":\"FeatureCollection\",
    \"features\":[{\"type\":\"Feature\",\"properties\":{},
    \"geometry\":{\"type\":\"Point\",\"coordinates\":
    [52.8658,-44.7219]}},{\"type\":\"Feature\",\"properties\":{},
    \"geometry\":{\"type\":\"Point\",\"coordinates\":
    [53.7702,-40.4873]}},{\"type\":\"Feature\",\"properties\":{},
    \"geometry\":{\"type\":\"Point\",\"coordinates\":[55.3204,-37.5579]}},
    {\"type\":\"Feature\",\"properties\":{},\"geometry\":
    {\"type\":\"Point\",\"coordinates\":[56.2757,-37.917]}},
    {\"type\":\"Feature\",\"properties\":{},\"geometry\":
    {\"type\":\"Point\",\"coordinates\":[56.184,-40.6443]}},
    {\"type\":\"Feature\",\"properties\":{},\"geometry\":
    {\"type\":\"Point\",\"coordinates\":[61.0835,-40.7529]}},
    {\"type\":\"Feature\",\"properties\":{},\"geometry\":
    {\"type\":\"Point\",\"coordinates\":[58.0202,-43.634]}}]}",
    class = c("geojson", "json"))
  points <- geojson_sf(points)
  plot(points)

  erase_poly <- structure('{
  "type": "Feature",
  "properties": {},
  "geometry": {
  "type": "Polygon",
  "coordinates": [
  [
  [51, -40],
  [55, -40],
  [55, -45],
  [51, -45],
  [51, -40]
  ]
  ]
  }
```

```
  }', class = c("geojson", "json"))
  erase_poly <- geojson_sf(erase_poly)

  out <- ms_erase(points, erase_poly)
  plot(out, add = TRUE)
}
```

`ms_explode` *Convert multipart lines or polygons to singlepart*

Description

For objects of class `Spatial` (e.g., `SpatialPolygonsDataFrame`), you may find it faster to use `sp::disaggregate`.

Usage

```
ms_explode(input, ...)
```

Arguments

`input`	One of: • `geo_json` or `character` multipart lines, or polygons; • multipart `SpatialPolygons`, `SpatialLines`; • `sf` or `sfc` multipart lines, or polygons object

`...`	Arguments passed on to `apply_mapshaper_commands`
	`force_FC` should the output be forced to be a FeatureCollection (or sf object or Spatial*DataFrame) even if there are no attributes? Default `TRUE`. If FALSE and there are no attributes associated with the geometries, a GeometryCollection (or Spatial object with no dataframe, or sfc) will be output.
	`sys` Should the system mapshaper be used instead of the bundled mapshaper? Gives better performance on large files. Requires the mapshaper node package to be installed and on the PATH.
	`sys_mem` How much memory (in GB) should be allocated if using the system mapshaper (`sys = TRUE`)? Default 8. Ignored if `sys = FALSE`. This can also be set globally with the option `"mapshaper.sys_mem"`
	`quiet` If `sys = TRUE`, should the mapshaper messages be silenced? Default `FALSE`. This can also be set globally with the option `"mapshaper.sys_quiet"`
	`gj2008` Generate output that is consistent with the pre-RFC 7946 GeoJSON spec (dating to 2008). Polygon rings are CW and holes are CCW, which is the opposite of the default RFC 7946-compatible output. This should be rarely needed, though may be useful when preparing data for D3-based data visualizations (such as `plotly::plot_ly()`). Default `FALSE`

Details

There is currently no method for SpatialMultiPoints

Value

same class as input

Examples

```
library(geojsonsf)
library(sf)

poly <- "{\"type\":\"FeatureCollection\",\"features\":
          [\n{\"type\":\"Feature\",\"geometry\":{\"type\":
          \"MultiPolygon\",\"coordinates\":[[[[102,2],[102,3],
          [103,3],[103,2],[102,2]]],[[[100,0],[100,1],[101,1],
          [101,0],[100,0]]]]},\"properties\":{\"a\":0}}\n]}"

poly <- geojson_sf(poly)
plot(poly)
length(poly)
poly

# Explode the polygon
out <- ms_explode(poly)
plot(out)
length(out)
out
```

---

ms_filter_fields *Delete fields in the attribute table*

Description

Removes all fields except those listed in the `fields` parameter

Usage

```
ms_filter_fields(input, fields, ...)
```

Arguments

`input`	spatial object to filter fields on. One of: • `geo_json` or `character` points, lines, or polygons; • `SpatialPolygonsDataFrame`, `SpatialLinesDataFrame`, `SpatialPointsDataFrame`; • `sf` object
`fields`	character vector of fields to retain.

... Arguments passed on to apply_mapshaper_commands

sys Should the system mapshaper be used instead of the bundled mapshaper? Gives better performance on large files. Requires the mapshaper node package to be installed and on the PATH.

sys_mem How much memory (in GB) should be allocated if using the system mapshaper (sys = TRUE)? Default 8. Ignored if sys = FALSE. This can also be set globally with the option "mapshaper.sys_mem"

quiet If sys = TRUE, should the mapshaper messages be silenced? Default FALSE. This can also be set globally with the option "mapshaper.sys_quiet"

Value

object with only specified attributes retained, in the same class as the input

Examples

```
library(geojsonsf)
library(sf)

poly <- structure("{\"type\":\"FeatureCollection\",
                  \"features\":[{\"type\":\"Feature\",
                  \"properties\":{\"a\": 1, \"b\":2, \"c\": 3},
                  \"geometry\":{\"type\":\"Polygon\",
                  \"coordinates\":[[[102,2],[102,4],[104,4],[104,2],[102,2]]]}}]}",
                  class = c("geojson", "json"))
poly <- geojson_sf(poly)
poly

# Filter (keep) fields a and b, drop c
out <- ms_filter_fields(poly, c("a", "b"))
out
```

---

ms_filter_islands *Remove small detached polygons (islands)*

Description

Remove small detached polygons, keeping those with a minimum area and/or a minimum number of vertices. Optionally remove null geometries.

Usage

```
ms_filter_islands(
  input,
  min_area = NULL,
  min_vertices = NULL,
  drop_null_geometries = TRUE,
  ...
)
```

Arguments

input	spatial object to filter. One of: • geo_json or character polygons; • SpatialPolygons*; • sf or sfc polygons object
min_area	minimum area of polygons to retain. Area is calculated using planar geometry, except for the area of unprojected polygons, which is calculated using spherical geometry in units of square meters.
min_vertices	minimum number of vertices in polygons to retain.
drop_null_geometries	should features with empty geometries be dropped? Default TRUE. Ignored for SpatialPolyons*, as it is always TRUE.
...	Arguments passed on to apply_mapshaper_commands force_FC should the output be forced to be a FeatureCollection (or sf object or Spatial*DataFrame) even if there are no attributes? Default TRUE. If FALSE and there are no attributes associated with the geometries, a GeometryCollection (or Spatial object with no dataframe, or sfc) will be output. sys Should the system mapshaper be used instead of the bundled mapshaper? Gives better performance on large files. Requires the mapshaper node package to be installed and on the PATH. sys_mem How much memory (in GB) should be allocated if using the system mapshaper (sys = TRUE)? Default 8. Ignored if sys = FALSE. This can also be set globally with the option "mapshaper.sys_mem" quiet If sys = TRUE, should the mapshaper messages be silenced? Default FALSE. This can also be set globally with the option "mapshaper.sys_quiet" gj2008 Generate output that is consistent with the pre-RFC 7946 GeoJSON spec (dating to 2008). Polygon rings are CW and holes are CCW, which is the opposite of the default RFC 7946-compatible output. This should be rarely needed, though may be useful when preparing data for D3-based data visualizations (such as plotly::plot_ly()). Default FALSE

Value

object with only specified features retained, in the same class as the input

Examples

```
library(geojsonsf)
library(sf)

poly <- structure("{\"type\":\"FeatureCollection\",
          \"features\":[{\"type\":\"Feature\",\"properties\":{},
          \"geometry\":{\"type\":\"Polygon\",
          \"coordinates\":[[[102,2],[102,4],[104,4],[104,2],[102,2]]]}},
          {\"type\":\"Feature\",\"properties\":{},
          \"geometry\":{\"type\":\"Polygon\",
          \"coordinates\":[[[100,2],[98,4],[101.5,4],[100,2]]]}},
```

```
            {\"type\":\"Feature\",\"properties\":{},
            \"geometry\":{\"type\":\"Polygon\",
            \"coordinates\":[[[100,0],[100,1],[101,1],[101,0],[100,0]]]}}]}",
            class = c("geojson", "json"))

poly <- geojson_sf(poly)
plot(poly)

out <- ms_filter_islands(poly, min_area = 12391399903)
plot(out)
```

`ms_innerlines` *Create a line layer consisting of shared boundaries with no attribute data*

Description

Create a line layer consisting of shared boundaries with no attribute data

Usage

```
ms_innerlines(input, ...)
```

Arguments

`input`	input polygons object to convert to inner lines. One of: • `geo_json` or `character` polygons; • `SpatialPolygons*`; • `sf` or `sfc` polygons object

`...`	Arguments passed on to `apply_mapshaper_commands`
	`force_FC` should the output be forced to be a FeatureCollection (or sf object or Spatial*DataFrame) even if there are no attributes? Default `TRUE`. If FALSE and there are no attributes associated with the geometries, a GeometryCollection (or Spatial object with no dataframe, or sfc) will be output.
	`sys` Should the system mapshaper be used instead of the bundled mapshaper? Gives better performance on large files. Requires the mapshaper node package to be installed and on the PATH.
	`sys_mem` How much memory (in GB) should be allocated if using the system mapshaper (`sys = TRUE`)? Default 8. Ignored if `sys = FALSE`. This can also be set globally with the option `"mapshaper.sys_mem"`
	`quiet` If `sys = TRUE`, should the mapshaper messages be silenced? Default `FALSE`. This can also be set globally with the option `"mapshaper.sys_quiet"`
	`gj2008` Generate output that is consistent with the pre-RFC 7946 GeoJSON spec (dating to 2008). Polygon rings are CW and holes are CCW, which is the opposite of the default RFC 7946-compatible output. This should be rarely needed, though may be useful when preparing data for D3-based data visualizations (such as `plotly::plot_ly()`). Default `FALSE`

Value

lines in the same class as the input layer, but without attributes

Examples

```
library(geojsonsf)
library(sf)

poly <- structure('{"type":"FeatureCollection",
            "features":[
              {"type":"Feature",
                "properties":{"foo": "a"},
                "geometry":{"type":"Polygon","coordinates":[[
                  [102,2],[102,3],[103,3],[103,2],[102,2]
                  ]]}}
              ,{"type":"Feature",
                "properties":{"foo": "a"},
                "geometry":{"type":"Polygon","coordinates":[[
                  [103,3],[104,3],[104,2],[103,2],[103,3]
                  ]]}},
              {"type":"Feature",
                "properties":{"foo": "b"},
                "geometry":{"type":"Polygon","coordinates":[[
                  [102,1],[102,2],[103,2],[103,1],[102,1]
                  ]]}},
              {"type":"Feature",
                "properties":{"foo": "b"},
                "geometry":{"type":"Polygon","coordinates":[[
                  [103,1],[103,2],[104,2],[104,1],[103,1]
                  ]]}}]}', class = c("geojson", "json"))

poly <- geojson_sf(poly)
plot(poly)

out <- ms_innerlines(poly)
plot(out)
```

`ms_lines` *Convert polygons to topological boundaries (lines)*

Description

Convert polygons to topological boundaries (lines)

Usage

```
ms_lines(input, fields = NULL, ...)
```

Arguments

input	input polygons object to convert to inner lines. One of: • geo_json or character polygons; • SpatialPolygons*; • sf or sfc polygons object
fields	character vector of field names. If left as NULL (default), external (unshared) boundaries are attributed as TYPE 0 and internal (shared) boundaries are TYPE 1. Giving a field name adds an intermediate level of hierarchy at TYPE 1, with the lowest-level internal boundaries set to TYPE 2. Supplying a character vector of field names adds additional levels of hierarchy.
...	Arguments passed on to apply_mapshaper_commands force_FC should the output be forced to be a FeatureCollection (or sf object or Spatial*DataFrame) even if there are no attributes? Default TRUE. If FALSE and there are no attributes associated with the geometries, a GeometryCollection (or Spatial object with no dataframe, or sfc) will be output. sys Should the system mapshaper be used instead of the bundled mapshaper? Gives better performance on large files. Requires the mapshaper node package to be installed and on the PATH. sys_mem How much memory (in GB) should be allocated if using the system mapshaper (sys = TRUE)? Default 8. Ignored if sys = FALSE. This can also be set globally with the option "mapshaper.sys_mem" quiet If sys = TRUE, should the mapshaper messages be silenced? Default FALSE. This can also be set globally with the option "mapshaper.sys_quiet" gj2008 Generate output that is consistent with the pre-RFC 7946 GeoJSON spec (dating to 2008). Polygon rings are CW and holes are CCW, which is the opposite of the default RFC 7946-compatible output. This should be rarely needed, though may be useful when preparing data for D3-based data visualizations (such as plotly::plot_ly()). Default FALSE

Value

topological boundaries as lines, in the same class as the input

Examples

```
library(geojsonsf)
library(sf)

poly <- structure('{"type":"FeatureCollection",
             "features":[
             {"type":"Feature",
             "properties":{"foo": "a"},
             "geometry":{"type":"Polygon","coordinates":[[
             [102,2],[102,3],[103,3],[103,2],[102,2]
             ]]}}
             ,{"type":"Feature",
             "properties":{"foo": "a"},
```

```
              "geometry":{"type":"Polygon","coordinates":[[
              [103,3],[104,3],[104,2],[103,2],[103,3]
              ]]}},
              {"type":"Feature",
              "properties":{"foo": "b"},
              "geometry":{"type":"Polygon","coordinates":[[
              [102.5,1],[102.5,2],[103.5,2],[103.5,1],[102.5,1]
              ]]}}]}', class = c("geojson", "json"))

poly <- geojson_sf(poly)
summary(poly)
plot(poly)

out <- ms_lines(poly)
summary(out)
plot(out)
```

`ms_points` *Create points from a polygon layer*

Description

Can be generated from the polygons by specifying `location` to be `"centroid"` or `"inner"`, OR by specifying fields in the attributes of the layer containing `x` and `y` coordinates.

Usage

```
ms_points(input, location = NULL, x = NULL, y = NULL, ...)
```

Arguments

`input`	input polygons object to convert to points. One of: • `geo_json` or `character` polygons; • `SpatialPolygons*`; • `sf` or `sfc` polygons object
`location`	either `"centroid"` or `"inner"`. If `"centroid"`, creates points at the centroid of the largest ring of each polygon feature. if `"inner"`, creates points in the interior of the largest ring of each polygon feature. Inner points are located away from polygon boundaries. Must be `NULL` if `x` and `y` are specified. If left as `NULL` (default), will use centroids.
`x`	name of field containing `x` coordinate values. Must be `NULL` if `location` is specified.
`y`	name of field containing `y` coordinate values. Must be `NULL` if `location` is specified.
`...`	Arguments passed on to `apply_mapshaper_commands`

force_FC should the output be forced to be a FeatureCollection (or sf object or Spatial*DataFrame) even if there are no attributes? Default TRUE. If FALSE and there are no attributes associated with the geometries, a GeometryCollection (or Spatial object with no dataframe, or sfc) will be output.

sys Should the system mapshaper be used instead of the bundled mapshaper? Gives better performance on large files. Requires the mapshaper node package to be installed and on the PATH.

sys_mem How much memory (in GB) should be allocated if using the system mapshaper (sys = TRUE)? Default 8. Ignored if sys = FALSE. This can also be set globally with the option "mapshaper.sys_mem"

quiet If sys = TRUE, should the mapshaper messages be silenced? Default FALSE. This can also be set globally with the option "mapshaper.sys_quiet"

gj2008 Generate output that is consistent with the pre-RFC 7946 GeoJSON spec (dating to 2008). Polygon rings are CW and holes are CCW, which is the opposite of the default RFC 7946-compatible output. This should be rarely needed, though may be useful when preparing data for D3-based data visualizations (such as plotly::plot_ly()). Default FALSE

Value

points in the same class as the input.

Examples

```
library(geojsonsf)
library(sf)

poly <- structure("{\"type\":\"FeatureCollection\",
           \"features\":[{\"type\":\"Feature\",\"properties\":
           {\"x_pos\": 1, \"y_pos\": 2},
           \"geometry\":{\"type\":\"Polygon\",
           \"coordinates\":[[[102,2],[102,4],[104,4],[104,2],[102,2]]]}},
           {\"type\":\"Feature\",\"properties\":{\"x_pos\": 3, \"y_pos\": 4},
           \"geometry\":{\"type\":\"Polygon\",
           \"coordinates\":[[[100,2],[98,4],[101.5,4],[100,2]]]}},
           {\"type\":\"Feature\",\"properties\":{\"x_pos\": 5, \"y_pos\": 6},
           \"geometry\":{\"type\":\"Polygon\",
           \"coordinates\":[[[100,0],[100,1],[101,1],[101,0],[100,0]]]}}]}",
           class = c("geojson", "json"))

poly <- geojson_sf(poly)
summary(poly)
plot(poly)

# Convert to points using centroids
out <- ms_points(poly, location = "centroid")
summary(out)
plot(out)

# Can also specify locations using attributes in the data
out <- ms_points(poly, x = "x_pos", y = "y_pos")
```

```
summary(out)
plot(out)
```

`ms_simplify` *Topologically-aware geometry simplification.*

Description

Uses mapshaper to simplify polygons.

Usage

```
ms_simplify(
  input,
  keep = 0.05,
  method = NULL,
  weighting = 0.7,
  keep_shapes = FALSE,
  no_repair = FALSE,
  snap = TRUE,
  explode = FALSE,
  drop_null_geometries = TRUE,
  snap_interval = NULL,
  ...
)
```

Arguments

`input`	spatial object to simplify. One of: • `geo_json` or `character` polygons or lines; • `SpatialPolygons*` or `SpatialLines*`; • `sf` or `sfc` polygons or lines object
`keep`	proportion of points to retain (0-1; default 0.05)
`method`	simplification method to use: `"vis"` for Visvalingam algorithm, or `"dp"` for Douglas-Peuker algorithm. If left as `NULL` (default), uses Visvalingam simplification but modifies the area metric by underweighting the effective area of points at the vertex of more acute angles, resulting in a smoother appearance. See this link for more information.
`weighting`	Coefficient for weighting Visvalingam simplification (default is 0.7). Higher values produce smoother output. weighting=0 is equivalent to unweighted Visvalingam simplification.
`keep_shapes`	Prevent small polygon features from disappearing at high simplification (default `FALSE`)
`no_repair`	disable intersection repair after simplification (default `FALSE`).

snap Snap together vertices within a small distance threshold to fix small coordinate misalignment in adjacent polygons. Default TRUE.

explode Should multipart polygons be converted to singlepart polygons? This prevents small shapes from disappearing during simplification if keep_shapes = TRUE. Default FALSE

drop_null_geometries should Features with null geometries be dropped? Ignored for Spatial* objects, as it is always TRUE.

snap_interval Specify snapping distance in source units, must be a numeric. Default NULL

... Arguments passed on to apply_mapshaper_commands

force_FC should the output be forced to be a FeatureCollection (or sf object or Spatial*DataFrame) even if there are no attributes? Default TRUE. If FALSE and there are no attributes associated with the geometries, a GeometryCollection (or Spatial object with no dataframe, or sfc) will be output.

sys Should the system mapshaper be used instead of the bundled mapshaper? Gives better performance on large files. Requires the mapshaper node package to be installed and on the PATH.

sys_mem How much memory (in GB) should be allocated if using the system mapshaper (sys = TRUE)? Default 8. Ignored if sys = FALSE. This can also be set globally with the option "mapshaper.sys_mem"

quiet If sys = TRUE, should the mapshaper messages be silenced? Default FALSE. This can also be set globally with the option "mapshaper.sys_quiet"

gj2008 Generate output that is consistent with the pre-RFC 7946 GeoJSON spec (dating to 2008). Polygon rings are CW and holes are CCW, which is the opposite of the default RFC 7946-compatible output. This should be rarely needed, though may be useful when preparing data for D3-based data visualizations (such as plotly::plot_ly()). Default FALSE

Value

a simplified representation of the geometry in the same class as the input

Examples

```
# With a simple geojson object
poly <- structure('{
 "type": "Feature",
 "properties": {},
 "geometry": {
   "type": "Polygon",
   "coordinates": [[
     [-70.603637, -33.399918],
     [-70.614624, -33.395332],
     [-70.639343, -33.392466],
     [-70.659942, -33.394759],
     [-70.683975, -33.404504],
     [-70.697021, -33.419406],
     [-70.701141, -33.434306],
```

```
        [-70.700454, -33.446339],
        [-70.694274, -33.458369],
        [-70.682601, -33.465816],
        [-70.668869, -33.472117],
        [-70.646209, -33.473835],
        [-70.624923, -33.472117],
        [-70.609817, -33.468107],
        [-70.595397, -33.458369],
        [-70.587158, -33.442901],
        [-70.587158, -33.426283],
        [-70.590591, -33.414248],
        [-70.594711, -33.406224],
        [-70.603637, -33.399918]
      ]]
   }
  }', class = c("geojson", "json"))

  ms_simplify(poly, keep = 0.1)

  # With an sf object

  poly_sf <- geojsonsf::geojson_sf(poly)
  ms_simplify(poly_sf, keep = 0.5)
```

Index