Package: polymer (via r-universe)

September 6, 2024

Title Flexible and Intuitive Overlay Methods

Version 0.2.0.9001

Description General intersections via a triangle pool from disparate polygon inputs. Overlap is determined via finite-element decomposition of all component edges in all inputs into triangles. Then triangles instances are classified (by point-in-polygon lookup) by objects within layers.

License GPL-3

URL https://github.com/hypertidy/polymer, https://hypertidy.github.io/polymer/

BugReports https://github.com/hypertidy/polymer/issues

Depends R (>= 3.4.0)

Imports dplyr, gibble, magrittr, rlang, RTriangle, sf, silicate (>= 0.2.0), tidyr, purrr, polyclip, tibble

Suggests covr, testthat

ByteCompile true

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/hypertidy/polymer

RemoteRef HEAD

RemoteSha b0a66bf329dbd15e14b14dc88a4a6028da7f3607

Contents

`polymer-package` *Polymer*

Description

Find overlapping polygons regions by converting a collection of sf data frame polygon layers to a single pool of triangles. The pool is used for subsequent queries, what layers intersect and how, maintaining links to input data of all layers.

`ABC` *Simple data examples*

Description

A, B, C are three partially overlaying polygon layers with different sets of attributes.

`field` *data*

Description

Data from online discussion.

layer_n *N intersections*

Description

Find all fragments that are intersected by any other feature in any layer.

Usage

```
layer_n(x, n = 2, ..., keep_index = FALSE)
```

Arguments

x	polymer
n	minimum number of intersections to keep
...	ignored for now
keep_index	for expert use only, maintains the list of triangle indexes on the sf output (and sf cannot plot if that is present)

Details

Returns a simple features data frame with all triangles that occur n times with n = 2 as a minimum. Each triangle feature contains a nested data frame in idx that keeps the links to the input layers by layer, object and path.

Value

sf data frame

Examples

```
library(sf)
plot(A["layer"], reset = TRUE)
plot(B, add = TRUE, col = "hotpink")
plot(C, add = TRUE, col = "firebrick")

sb <- polymer(A, B, C)
plot(layer_n(sb), add = TRUE, col = "grey")
plot(layer_n(sb, n = 3), add = TRUE, col = "dodgerblue")
```

plot.polymer *Plot polymer*

Description

The default plot shows only the mesh. If `show_intersection = TRUE`, the part of the mesh that has 2 intersecting regions or more is contrasted to the rest.

Usage

```
## S3 method for class 'polymer'
plot(x, ..., show_intersection = FALSE)
```

Arguments

`x`	polymer
`...`	arguments to polypath
`show_intersection`	logical, plot the intersection region contrasted to the pool (default `FALSE`)

Value

the input, invisibly

Examples

```
plot(polymer(A, B, C))
library(sf)
example(st_read)
nc <- nc[1:5, ]
x <- polymer(nc, st_jitter(nc, amount = 0.1))
plot(x)
```

polymer *Polymer*

Description

Convert a collection of sf data frame polygon layers to a single pool of triangles.

Usage

```
polymer(...)
```

Arguments

`...`	sf polygon data frame inputs

Details

Each triangle is identified by which path in the inputs it belongs to. None of this is very useable yet. Holes can be identified but aren't at the moment, any path that is a hole is identified per triangle.

`input` is a list with all input objects `primitives` is the triangulation object `geometry_map` is the paths with their row count `index` is the mapping between triangle and path/s

Value

a polymer, see details

Examples

```
polymer(A, B, C)
```

`print.polymer` *Print polymer*

Description

Print a short description of the polymer contents.

Usage

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

Arguments

`x`	polymer
`...`	ignored

Value

x invisibly

Examples

```
polymer(A, B, C)
```

Index