

Package: polymer (via r-universe)

October 14, 2026

Title Overlay Any Number of Polygon Layers with a Triangle Mesh

Version 0.3.0

Description Overlay polygon layers by building one constrained Delaunay triangulation of every input edge from every layer, with crossings inserted as new vertices, and recording which input features cover each triangle. The mesh and its index answer overlay questions for any number of layers at once: where features overlap, how many times, and which features are involved. Overlay pieces are dissolved from the mesh, so they form an exact partition with shared vertices, unlike pairwise polygon clipping. Input is anything the 'wk' package reads, the vertex pool comes from 'wkpool' and the triangulation from 'cdtr'.

License GPL-3

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

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

Depends R (>= 4.1.0)

Imports cdtr, graphics, polyclip, wk (>= 0.9.4), wkpool (>= 0.4.0)

Suggests sf, testthat (>= 3.0.0)

Remotes hypertidy/cdtr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

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

Date/Publication 2026-10-14 05:28:37 UTC

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

RemoteRef HEAD
RemoteSha ac2e837eedbde09dfb07688b479cf4c09a144b02

Contents

ABC	2
field	2
fuse	3
plot.polymer	4
polymer	5
print.polymer	6
Index	7

ABC	<i>Simple overlapping layers</i>
-----	----------------------------------

Description

A, B and C are three partially overlapping polygon layers with different attributes: A is four squares in a 2 by 2 grid over the unit square, B is a single triangle and C a single thin rectangle.

Format

- 'sf' data frames with no coordinate reference system:
- A: 4 rows, layer, a and geometry.
 - B: 1 row, b and geometry.
 - C: 1 row, c and geometry.

Examples

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

field	<i>Field and soil polygons</i>
-------	--------------------------------

Description

Two polygon layers used to show overlay: soil has 12 polygons with a made-up soil type, field has two rectangular fields that cross them. The example comes from a question on the GIS Stack Exchange (<https://gis.stackexchange.com/a/230161>), using the Luxembourg boundaries shipped with the 'raster' package.

Format

'sf' data frames in longitude and latitude (WGS84):

- soil: 12 rows, soil (character) and geometry.
- field: 2 rows, field (character) and geoms.

Examples

```
polymer(soil, field)
```

 fuse

Fuse a polymer into overlay pieces

Description

Dissolve the triangles of a polymer into polygons, one for each distinct set of covering input features. This is the n-ary intersection of all layers (the same partition that `sf::st_intersection()` computes for a single layer), derived from the mesh rather than from pairwise overlay.

Usage

```
fuse(x, ...)
```

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

Arguments

x	A polymer (see polymer()).
...	Ignored.
n	Minimum number of covering features, for <code>layer_n()</code> .

Details

`layer_n()` keeps the pieces covered by at least n features.

Pieces are dissolved along mesh edges, so they share vertices exactly where they touch. Outer rings are counter-clockwise and holes clockwise. Every piece is a MULTIPOLYGON, with one polygon for each connected part.

Value

A data frame with one row per piece and columns

- n: the number of input features covering the piece.
- origins: a list column, each element a data frame of layer and feature (row within its layer) for the features covering the piece.
- geometry: a `wk::wkb()` vector with the polymer's coordinate reference system. Convert with `sf::st_as_sf()` if you work with 'sf'.

Examples

```

pool <- polymer(A, B, C)
pieces <- fuse(pool)
pieces[c("n", "geometry")]

## which features made the deepest piece
pieces$origins[[which.max(pieces$n)]]

## pieces covered two or more times
plot(pool)
wk::wk_plot(layer_n(pool, 2)$geometry, add = TRUE, col = "grey")
wk::wk_plot(layer_n(pool, 3)$geometry, add = TRUE, col = "dodgerblue")

layer_n(pool, 3)

```

plot.polymer

Plot a polymer

Description

Draw the triangles of the mesh. With `show_intersection = TRUE` the triangles covered by two or more input features are filled.

Usage

```

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

```

Arguments

<code>x</code>	A polymer.
<code>...</code>	Arguments passed to <code>graphics::polygon()</code> , such as <code>col</code> and <code>border</code> .
<code>show_intersection</code>	Logical, fill the triangles covered more than once (default <code>FALSE</code>).

Value

`x`, invisibly.

Examples

```

plot(polymer(A, B, C))
plot(polymer(A, B, C), show_intersection = TRUE, col = "firebrick")

```

polymer

Build a polymer mesh from polygon layers

Description

Combine one or more polygon layers into a single triangle mesh, where every input edge from every layer is kept as a chain of mesh edges, and record which input features cover each triangle.

Usage

```
polymer(..., engine = "cdtr")
```

Arguments

...	Polygon layers. Each can be anything the 'wk' package can read (an 'sf' data frame or <code>sfc</code> , a <code>wk::wkb()</code> or <code>wk::wkt()</code> vector, and so on). Data frames keep their non-geometry columns as attributes.
engine	The triangulation engine. Only "cdtr" is available.

Details

All the inputs' edges are triangulated together in one constrained Delaunay triangulation. Where edges from different features cross, the crossing is inserted as a new vertex, so the mesh is a partition of the plane that respects every input boundary. Vertices are shared on exact coordinate equality before triangulation.

Each triangle is then labelled with the features that contain it: a triangle is inside a feature when its centroid is inside an odd number of the feature's rings, so holes are subtracted. Triangles covered by no feature (gaps enclosed by the inputs) are dropped.

The result is the mesh and its index, not a set of polygons. Use `fuse()` or `layer_n()` to get polygons back.

Value

A polymer object, a list with

- `mesh`: a `wkpool_tri` (see `wkpool::new_wkpool_tri()`) holding the triangles on a shared vertex pool. The vertex table has an `origin` column: "input" for input vertices and "crossing" for vertices inserted where input edges cross.
- `index`: a data frame with one row per triangle and covering feature: `triangle` (row in mesh), `layer` (position of the layer in ...) and `feature` (row within that layer).
- `layers`: a list with one data frame of attributes per layer (no geometry).
- `crs`: the coordinate reference system shared by the inputs.

Examples

```
pool <- polymer(A, B, C)
pool
head(pool$index)

## the mesh is a 'wkpool' triangle pool
pool$mesh
```

<code>print.polymer</code>	<i>Print a polymer</i>
----------------------------	------------------------

Description

Print a short summary of a polymer mesh: the number of layers, input features, triangles, crossing vertices and the deepest overlap.

Usage

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

Arguments

<code>x</code>	A polymer.
<code>...</code>	Ignored.

Value

`x`, invisibly.

Examples

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

Index

A (ABC), [2](#)

ABC, [2](#)

B (ABC), [2](#)

C (ABC), [2](#)

field, [2](#)

fuse, [3](#)

fuse(), [5](#)

graphics::polygon(), [4](#)

layer_n(fuse), [3](#)

layer_n(), [5](#)

plot.polymer, [4](#)

polymer, [5](#)

polymer(), [3](#)

print.polymer, [6](#)

soil(field), [2](#)

wkpool::new_wkpool_tri(), [5](#)