Package 'ggpolypath'

May 8, 2026

Type Package

Title Polygons with Holes for the Grammar of Graphics

Version 0.4.0

Description Tools for working with polygons with holes in 'ggplot2', with a new 'geom' for drawing a 'polypath' applying the 'evenodd' or 'winding' rules.

URL https://mdsumner.github.io/ggpolypath/, http://rpubs.com/kohske/3522/

BugReports https://github.com/mdsumner/ggpolypath/issues/

Depends R (>= 3.1), ggplot2 (>= 2.1.0)

Suggests rmarkdown, knitr

LazyData yes

License GPL-3

VignetteBuilder knitr

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

Author Michael D. Sumner [aut, cre], Kohske Takahashi [ctb] (original author of 'geom_holygon')

Maintainer Michael D. Sumner <mdsumner@gmail.com>

Repository CRAN

Date/Publication 2025-07-24 05:50:02 UTC

Contents

dathome *Simple polygon data*

Description

A "home" profile of three objects with multiple parts as two related data frames.

Format

dathome is the metadata, a data frame with columns:

name A descriptive name

colour A colour to distinguish each object

FAD An arbitrary numeric value

object_ Key attribute, linking this object to its geometry in maphome

maphome is the geometry, a data frame with columns:

object_ Key attribute, linking this row to its metadata in dathome

branch_ Group attribute, unique values identify a closed ring

island_ Logical, TRUE for "island" vs. "hole"

order_ Numeric value to identify sort order within branch

x_,y_ x and y coordinate

Details

maphome is the geometry

Examples

```
ggplot(maphome) + aes(x = x_, y = y_, group = branch_, fill = object_) +
geom_polypath() + geom_path() + facet_wrap(~object_, nrow = nrow(dathome))
```

gardenstate *Province polygons with inland waters as holes.*

Description

A data frame of coordinates and geometry classifiers of the garden state, South Australia.

Format

gardenstate is the geometry, a data frame with columns:

x,y x and y coordinate

id Key attribute for the objects

piece,part Group attribute, unique values identify a closed ring, part is the part 'id' within an object

hole Logical, FALSE for "island" vs. "hole"

order Numeric value to identify sort order within branch

Details

The PROJ.4 string for this map is:

+proj=lcc +lat_1=-47 +lat_2=-17 +lat_0=-32 +lon_0=136 +x_0=0 +y_0=0 +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs

Examples

```
gs <- ggplot(gardenstate)
gs <- gs + aes(x = x, y = y, group = group, fill = id)
gs +  geom_polypath() + geom_path()
```

geom_polypath *Geom polypath, a polygon filled path that can include holes.*

Description

Polygons are drawn by tracing a 'path' of linked vertices and applying rule to differentiate the inside and the outside of the area traversed. The 'evenodd' rule provides the normal expected behaviour seen in simple GIS geometry and is immune to self-intersections and the orientation of the path (clockwise or anti-clockwise). The 'winding' rule behaves differently for self-intersections depending on relative orientation of the interacting paths.

Usage

```
geom_polypath(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  rule = "winding",
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used the override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount. • A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count". • For more information and other ways to specify the stat, see the layer stat documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".

- For more information and other ways to specify the position, see the layer position documentation.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.

`rule` character value specifying the path fill mode: either "winding" or "evenodd", see `polypath`

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

Details

See https://en.wikipedia.org/wiki/Even-odd_rule and https://en.wikipedia.org/wiki/Nonzero-rule for more details.

Value

a ggplot2 layer

See Also

polypath and pathGrob geom_polygon for the implementation on polygonGrob, geom_map for a convenient way to tie the values and coordinates together, geom_path for an unfilled polygon, geom_ribbon for a polygon anchored on the x-axis

Examples

```
# When using geom_polypath, you will typically need two data frames:
# one contains the coordinates of each polygon (positions),  and the
# other the values associated with each polygon (values).  An id
# variable links the two together.
# Normally this would not be created manually, but by using \code{\link{fortify}}
# to generate it from the Spatial classes in the `sp` package.

## the built-in data \code{\link{home}} uses nested data frames
library(ggplot2)
ggplot(maphome) + aes(x = x_, y = y_, group = branch_, fill = factor(object_)) +
geom_polypath()

## this is the same example built from scratch
positions = data.frame(x = c(0, 0, 46, 46, 0, 7, 13, 13, 7, 7, 18, 24,
24, 18, 18, 31, 37, 37, 31, 31, 18.4, 18.4, 18.6, 18.8, 18.8,
18.6, 18.4, 31, 31, 37, 37, 31, 0, 21, 31, 37, 46, 0, 18, 18,
24, 24, 18, 18.4, 18.6, 18.8, 18.8, 18.6, 18.4, 18.4),
y = c(0, 19, 19, 0, 0, 6, 6, 13, 13, 6, 1, 1, 12, 12, 1, 4, 4, 11, 11,
4, 6.89999999999999, 7.49999999999999, 7.69999999999999, 7.49999999999999,
6.89999999999999, 6.69999999999999, 6.89999999999999, 27, 34,
34, 24, 27, 19, 32, 27, 24, 19, 19, 1, 12, 12, 1, 1, 6.89999999999999,
6.69999999999999, 6.89999999999999, 7.49999999999999, 7.69999999999999,
 7.49999999999999, 6.89999999999999),
 id = c(1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L,
 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 2L, 2L,
 2L, 2L, 2L, 2L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L),
 group = c(1L, 1L, 1L, 1L, 1L, 2L, 2L, 2L, 2L, 2L, 3L, 3L, 3L, 3L, 3L, 4L,
 4L, 4L, 4L, 4L, 5L, 5L, 5L, 5L, 5L, 5L, 5L, 6L, 6L, 6L, 6L, 6L, 7L,
 7L, 7L, 7L, 7L, 7L, 8L, 8L, 8L, 8L, 8L, 9L, 9L, 9L, 9L, 9L, 9L, 9L))

values <- data.frame(
  id = unique(positions$id),
  value = c(2, 5.4, 3)
)

#  manually merge the two together
datapoly <- merge(values, positions, by = c("id"))

# the entire house
(house <- ggplot(datapoly, aes(x = x, y = y)) + geom_polypath(aes(fill = value, group = group)))

# just the front wall (and chimney), with its three parts, the first of which has three holes
wall <- ggplot(datapoly[datapoly$id == 1, ], aes(x = x, y = y))
wall + geom_polypath(aes(fill = id, group = group))
```

Index