Package ‘ggvoronoi’

February 19, 2019

Version 0.8.3

Date 2019-02-19

Title Voronoi Diagrams and Heatmaps with 'ggplot2'

Author Robert C. Garrett [aut, cre],
Austin Nar [aut],
Thomas J. Fisher [aut],
Karsten Maurer [ctb]

Maintainer Robert C. Garrett <garretrc@miamioh.edu>

Description
Easy creation and manipulation of Voronoi diagrams using 'deldir' with visualization in 'ggplot2'.

Depends R (>= 2.10), ggplot2 (>= 2.2.0)

Imports deldir, sp, rgeos, raster, methods

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests knitr, rmarkdown, ggmap, dplyr, scales, testthat, vdiffr, mapproj

VignetteBuilder knitr

URL https://github.com/garretrc/ggvoronoi/

NeedsCompilation no

Repository CRAN

Date/Publication 2019-02-19 22:00:03 UTC

R topics documented:

`fortify_voronoi` *Create Data Frame from voronoi_polygons output*

Description

Allows output of `voronoi_polygon` to be plotted using `geom_polygon`.

Usage

```
fortify_voronoi(vor_spdf)
```

Arguments

`vor_spdf`	Output from `voronoi_polygon` (`SpatialPolygonsDataFrame`)

Examples

```
set.seed(45056)
x <- sample(1:200,100)
y <- sample(1:200,100)
points <- data.frame(x, y,
                     distance = sqrt((x-100)^2 + (y-100)^2))
circle <- data.frame(x = 100*(1+cos(seq(0, 2*pi, length.out = 2500))),
                     y = 100*(1+sin(seq(0, 2*pi, length.out = 2500))),
                     group = rep(1,2500))

vor_spdf <- voronoi_polygon(data=points,x="x",y="y",outline=circle)
vor_df <- fortify_voronoi(vor_spdf)

ggplot(vor_df) +
    geom_polygon(aes(x=x,y=y,fill=distance,group=group))
```

`geom_voronoi` *Voronoi Diagrams with ggplot2*

Description

Convenience function for use with `stat_voronoi`. Uses `geom_polygon` as the default geom and `stat_voronoi` as the default stat. To plot region borders instead of a choropleth map, use `stat_voronoi` with `geom="path"`, or use `fill=NA` with `color="black"`.

Usage

```
geom_voronoi(mapping = NULL, data = NULL, stat = StatVoronoi,
  position = "identity", ..., na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, outline = NULL)
```

Arguments

`mapping` Set of aesthetic mappings created by `aes()` or `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.

`stat` The statistical transformation to use on the data for this layer, as a string.

`position` Position adjustment, either as a string, or the result of a call to a position adjustment function.

`...` Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

`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()`.

`outline` `data.frame` with first column x/longitude, second column y/latitude, and a group column denoting islands or pieces.

Examples

```
set.seed(45056)
x <- sample(1:200,100)
y <- sample(1:200,100)
points <- data.frame(x, y,
                     distance = sqrt((x-100)^2 + (y-100)^2))
circle <- data.frame(x = 100*(1+cos(seq(0, 2*pi, length.out = 2500))),
                     y = 100*(1+sin(seq(0, 2*pi, length.out = 2500))),
                     group = rep(1,2500))
```

```
ggplot(points) +
    geom_voronoi(aes(x=x,y=y,fill=distance))

ggplot(points) +
    geom_voronoi(aes(x=x,y=y,fill=distance),outline=circle)

ggplot(points) +
    geom_voronoi(aes(x=x,y=y,fill=distance),fill=NA,color="black")
```

ncdc_locations *Locations and Elevation of NCDC Weather Stations*

Description

A dataset containing identifying information as well as coordinates and elevation for each unique North American land-based weather station in the National Climatic Data Center. Note that sites with a missing elevation in the raw data (recorded as -999.9) have been removed. Some sites have a potentially erroneous recorded elevation of zero meters.

Usage

```
ncdc_locations
```

Format

A data frame with 65804 rows and 6 variables:

ID ID number to connect the data back to other NCDC data

country country that the site is in. USA, Canada or Mexico

state two character state/province abbreviation that the site is in

lat latitude coordinate

long longitude coordinate

elev site elevation in meters

Source

National Climatic Data Center

`oxford_bikes` *Bike Rack Locations in Oxford, Ohio.*

Description

A dataset with the location of bike racks in Oxford, Ohio. Includes Miami University's campus and Uptown Oxford. GPS coordinates were measured for each bike rack via handheld GPS receivers. Off-campus data were gathered for a Miami University GIS class project and On-campus data were gathered for a Miami University Kinesiology research project.

Usage

```
oxford_bikes
```

Format

A data frame with 205 rows and 3 variables:

x longitude coordinate

y latitude coordinate

name location name, if recorded

Source

Madeline Maurer (Miami University Department of Geography); Justin Hopkins, Dr. Helaine Alessio and Amanda Meiering (Miami University Department of Kinesiology)

`oxford_map` *Google Map of Oxford, Ohio.*

Description

A ggmap object containing a static google map image of Oxford, Ohio. Zoomed in to show the Uptown Oxford and Miami University campus areas.

Usage

```
oxford_map
```

Format

Large ggmap object

Source

Google Maps static map api (accessed from 'ggmap' package)

stat_voronoi *Voronoi Diagrams with ggplot2*

Description

See `geom_voronoi` for general use. Use `geom="polygon"` for choropleth heatmap or `geom="path"` for region borders.

Usage

```
stat_voronoi(mapping = NULL, data = NULL, geom = "polygon",
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, outline = NULL, ...)
```

Arguments

`mapping` Set of aesthetic mappings created by `aes()` or `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.

`geom` The geometric object to use display the data

`position` Position adjustment, either as a string, or the result of a call to a position adjustment function.

`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()`.

`outline` `data.frame` with first column x/longitude, second column y/latitude, and a group column denoting islands or pieces.

`...` Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

Examples

```
set.seed(45056)
x <- sample(1:200,100)
y <- sample(1:200,100)
points <- data.frame(x, y,
                     distance = sqrt((x-100)^2 + (y-100)^2))
circle <- data.frame(x = 100*(1+cos(seq(0, 2*pi, length.out = 2500))),
                     y = 100*(1+sin(seq(0, 2*pi, length.out = 2500))),
                     group = rep(1,2500))

ggplot(points) +
    stat_voronoi(aes(x=x,y=y,fill=distance))

ggplot(points) +
    stat_voronoi(aes(x=x,y=y),geom="path")

ggplot(points) +
    stat_voronoi(aes(x=x,y=y,fill=distance),outline=circle)
```

---

voronoi_polygon *Voronoi Diagram from Data Frame*

Description

Create a Voronoi diagram for analysis or plotting with geom_polygon.

Usage

```
voronoi_polygon(data, x = "x", y = "y", outline = NULL,
  data.frame = FALSE)
```

Arguments

data	data.frame containing a set of points to make voronoi regions and any additional desired columns.
x	numeric vector (for example longitude).
y	numeric vector (for example latitude).
outline	data.frame with first column x/longitude, second column y/latitude, and a group column denoting islands or pieces.
data.frame	output as data.frame? You will lose information if you do this. For use in StatVoronoi.

Examples

```
set.seed(45056)
x <- sample(1:200,100)
y <- sample(1:200,100)
points <- data.frame(x, y,
                     distance = sqrt((x-100)^2 + (y-100)^2))
circle <- data.frame(x = 100*(1+cos(seq(0, 2*pi, length.out = 2500))),
                     y = 100*(1+sin(seq(0, 2*pi, length.out = 2500))),
                     group = rep(1,2500))

vor_spdf <- voronoi_polygon(data=points,x=”x”,y=”y”,outline=circle)
vor_df <- fortify_voronoi(vor_spdf)

ggplot(vor_df) +
    geom_polygon(aes(x=x,y=y,fill=distance,group=group))
```

Index