Package: glow (via r-universe)

October 25, 2024

Title Make Plots that Glow

Version 0.13.0

Date 2024-09-24

Maintainer Travers Ching <traversc@gmail.com>

Description Provides a framework for creating plots with glowing points.

License GPL-3

Biarch true

Encoding UTF-8

LinkingTo Rcpp, RcppEigen, RcppParallel

Imports Rcpp, RcppParallel (>= 5.1.2), R6

Depends ggplot2

Suggests knitr, rmarkdown, viridisLite, magick, EBImage, qs2

VignetteBuilder knitr

RoxygenNote 7.3.2

URL https://github.com/traversc/glow

BugReports https://github.com/traversc/glow/issues

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

RemoteUrl https://github.com/traversc/glow

RemoteRef HEAD

RemoteSha ec0babec9f237b36b150098cc2b39fb019b6d40b

Contents

`additive_alpha` *additive_alpha*

Description

Simulates additive blending on a dark to light color scale.

Usage

```
additive_alpha(colors)
```

Arguments

`colors`	colors

Details

In R plotting (both ggplot and base R) blending is performed by alpha blending, which is an averaging effect. When combining light and glow effects, additive blending is more appropriate.

This function simulates additive blending by increasing color on a color scale to compensate for the averaging effect of alpha blending.

Note: this function is only appropriate for dark to light color scales.

Value

A simulated additive scale of the input colors.

Examples

```
m_solid <- viridisLite::magma(12)
m_additive <- additive_alpha(m_solid)
```

circular_palette *circular_palette*

Description

A helper function for circularizing color palettes.

Usage

```
circular_palette(n, pal_function=rainbow, invert=FALSE, ...)
```

Arguments

n	Number of colors to output (must be divisible by 2)
pal_function	The base palette to circularize
invert	Whether to invert the palette See details.
...	Arguments passed to 'pal_function'

Details

This function is useful when the color represents radial data. E.g. position on a sphere or circle. The 'invert' parameter reverses the order of circularization. E.g. if your circular palette would be red to blue to red, 'invert' changes this blue to red to blue.

Value

A circular color palette

Examples

```
colors <- circular_palette(n=1000, pal_function=rainbow)
t <- seq(0,2*pi, length.out=1000)
plot(sin(t), cos(t), col = colors, pch = 19)
```

clifford_attractor *clifford_attractor*

Description

A 2D chaotic attractor created by Clifford Pickover.

Usage

```
clifford_attractor(n_iter, A=1.886, B=-2.357, C=-0.328, D=0.918, x0=0.1, y0=0)
```

Arguments

n_iter	number of points to generate
A	see details
B	see details
C	see details
D	see details
x0	The initial x-coordinate
y0	The initial y-coordinate

Details

A clifford attractor is a 2D chaotic attractor given by the two equations: X[i+1] = sin(a*Y[i]) - c*cos(a*X[i]) Y[i+1] = sin(b*X[i]) - d*cos(b*Y[i]) https://en.wikipedia.org/wiki/List_of_chaotic_maps https://stackoverflow.com/q/51122970/2723734

Value

A dataframe containing X and Y coordinates of clifford attractor points, the angle and the distance between successive points.

Examples

```
cliff_points <- clifford_attractor(1e4, 1.886,-2.357,-0.328, 0.918, 0.1, 0)
color_pal <- circular_palette(n=144, pal_function=rainbow)
cliff_points$color <- map_colors(color_pal, cliff_points$angle, min_limit=-pi, max_limit=pi)

gm <- GlowMapper4$new(xdim=240, ydim=240, blend_mode = "additive", nthreads=1)
gm$map(x=cliff_points$x, y=cliff_points$y, radius=0.1, color=cliff_points$color)
pd <- gm$output_dataframe(saturation = 1)

ggplot() +
  geom_raster(data = pd, aes(x = x, y = y, fill = rgb(r,g,b,a)), show.legend=FALSE) +
  coord_fixed(gm$aspect(), xlim = gm$xlim(), ylim = gm$ylim()) +
  scale_fill_identity() +
  theme_night()
```

GlowMapper *GlowMapper*

Description

This class provides a framework for creating scatter plots based on a glow simulation. Points are mapped with a gaussian gradient to a raster with specified dimensions and properties.

Usage

```
m <- GlowMapper$new(xdim=1000, ydim=800, blend_mode = "screen", contrast_limit = 1e5, nthreads = 1)

m$map(x, y, radius, intensity = 1, distance_exponent = 2, xlimits = c(NA_real_, NA_real_), ylimits = c(N

m$output_raw(saturation = NA_real_)

m$output_dataframe(saturation = NA_real_)

m$aspect()

m$xlim()

m$ylim()
```

Arguments

xdim - The first dimension of the output matrix raster.

ydim - The second dimension of the output matrix raster.

blend_mode - Either screen or additive blending mode. See details.

contrast_limit - Determines the distance to search from a point. You shouldn't need to change this unless you have a lot of points in the plot stacked on top of each other.

nthreads - Number of threads to use.

x - X coordinate of points.

y - Y coordinate of points.

radius - Relative spread of glow intensity. The radius should be proportional to the x and y-ranges of the plot. Values between 1/10 to 1/100 of the range of the plot generally produce good results.

intensity - Maximum intensity at the center of a point.

distance_exponent - Exponent of the distance calculation when calculating intensities. A value of 2 corresponds to euclidean distance; a value of 1 corresponds to manhattan distance.

xlimits - The x-limits of the output plot. If NA, the limits are +/- 5% of the maximum and minimum points.

ylimits - The y-limits of the output plot. If NA, the limits are +/- 5% of the maximum and minimum points.

append - Whether to add to the existing output or overwrite.

saturation - When retrieving the output with `$output_raw` or `$output_dataframe`, maximum intensity values are capped at the given value. This is often useful when using additive blend mode to increase contrast.

Details

`$new()` creates a new GlowMapper object, which holds parameters, plotting data, and the output (a matrix of glow intensities). Creates a canvas to plot point data. With additive blending, the

intensities of each point are added arithmetically, which is how light intensities are added in the physical world. This is equivalent to an fast/approximate un-normalized 2D kernel density estimate.

With "screen" blending, two intensities are added according to the formula: `I_out = 1 - (1-I_a)*(1-I_b)`. Both additive blending and screen blending are commutative operations, meaning the order of points in a plot does not affect the output.

Screen blending can often improve contrast in a plot and is the default.

`$map()` maps points to the canvas.

`$output_raw()` output raw matrix rasters. Useful for plotting in base R.

`$output_dataframe()` output the raster as a dataframe with XY coordinates. This is meant to pipe directly into ggplot.

`$aspect()`, `$xlim()`, `$ylim()` return the aspect ratio, x-limits and y-limits of the raster. These functions are intended to be used with plotting functions (e.g. `ggplot2::coord_fixed()`) so that the output raster is not distorted. See example below.

Examples

```
# Plot Data: x,y,r
x <- numeric(length=50)
y <- numeric(length=50)
r <- numeric(length=50)
for(t in 1:50) {
  xy <- exp(1i * t/2 - t/12)
  x[t] <- Re(xy)
  y[t] <- Im(xy)
  r[t] <- sqrt(x[t]^2 + y[t]^2)
}

# New class object
m <- GlowMapper$new(xdim=500, ydim = 400, blend_mode = "screen")

# Map data on to raster
m$map(x=x, y=y, intensity = 1, radius = r/4 + 0.1, distance_exponent = 2)

# Output raster data as a dataframe
pd <- m$output_dataframe(saturation = 1)

# Plot with ggplot
ggplot(pd, aes(x = x, y = y, fill = value)) +
  geom_raster(show.legend = FALSE) +
  scale_fill_gradientn(colors=additive_alpha(c("black", "purple", "white"))) +
  coord_fixed(ratio = m$aspect(), xlim = m$xlim(), ylim = m$ylim(), expand = FALSE) +
  theme_night(bgcolor = "black")
```

Description

This class provides a framework for creating scatter plots based on a glow simulation with explicit color intensities. Points are mapped with a gaussian gradient to a raster with specified dimensions and properties.

Usage

```
m <- GlowMapper$new(xdim=1000, ydim=800, blend_mode = "additive", background_color = "#00000000", contr

m$map(x, y, radius, color = NULL, r=NULL, g=NULL, b=NULL, distance_exponent = 2, xlimits = c(NA_real_, N

m$output_raw(saturation = 1, saturation_mode = "overflow")

m$output_dataframe(saturation = 1, saturation_mode = "overflow")

m$aspect()

m$xlim()

m$ylim()
```

Arguments

xdim - The first dimension of the output matrix raster.

ydim - The second dimension of the output matrix raster.

blend_mode - Either screen or additive blending mode. See details.

background_color - A color that can be coerced to RGBA with 'col2rgb', or a vector of four values between 0 and 1.

contrast_limit - Determines the distance to search from a point. You shouldn't need to change this unless you have a lot of points in the plot stacked on top of each other.

nthreads - Number of threads to use.

x - X coordinate of points.

y - Y coordinate of points.

radius - Relative spread of glow intensity. The radius should be proportional to the x and y-ranges of the plot. Values between 1/10 to 1/100 of the range of the plot generally produce good results.

color - Color of points. If NULL, r, g, and b parameters must be defined (and vice versa).

r - Red intensity of points. Must be between 0 and 1 if using screen blending.

g - Green intensity of points. Must be between 0 and 1 if using screen blending.

b - Blue intensity of points. Must be between 0 and 1 if using screen blending.

distance_exponent - Exponent of the distance calculation when calculating intensities. A value of 2 corresponds to euclidean distance; a value of 1 corresponds to manhattan distance.

xlimits - The x-limits of the output plot. If NA, the limits are +/- 5% of the maximum and minimum points.

ylimits - The y-limits of the output plot. If NA, the limits are +/- 5% of the maximum and minimum points.

append - Whether to add to the existing output or overwrite.

saturation - When retrieving the output with `$output_raw` or `$output_dataframe`, maximum intensity values are capped at the given value. This is often useful when using additive blend mode to increase contrast.

saturation_mode - When intensity values are above the saturation threshold, values can be overflowed into other color channels ("overflow") or simply clipped at the threshold ("clip"). "Overflow" always produces a gradient to white for intensities above the threshold, which may produce artistically better results.

Details

This 'GlowMapper4' class is similar to the 'GlowMapper' class, but instead of a single intensity matrix output, color is specified explicitly.

`$new()` creates a new GlowMapper object, which holds parameters, plotting data, and the output (a matrix of glow intensities). Creates a canvas to plot point data. With additive blending, the intensities of each point are added arithmetically, which is how light intensities are added in the physical world. This is equivalent to an fast/approximate un-normalized 2D kernel density estimate.

With "screen" blending, two intensities are added according to the formula: `I_out = 1 - (1-I_a)*(1-I_b)`. Both additive blending and screen blending are commutative operations, meaning the order of points in a plot does not affect the output.

Screen blending can often improve contrast in a plot and is the default.

`$map()` maps points to the canvas.

`$output_raw()` output raw matrix rasters (a list of four matrices, one for each RGBA channel). Useful for plotting in base R.

`$output_dataframe()` output the raster as a dataframe with XY coordinates. This is meant to pipe directly into ggplot.

`$aspect()`, `$xlim()`, `$ylim()` return the aspect ratio, x-limits and y-limits of the raster. These functions are intended to be used with plotting functions (e.g. `ggplot2::coord_fixed()`) so that the output raster is not distorted. See example below.

Examples

```
# Plot Data: x,y,r
x <- numeric(length=50)
y <- numeric(length=50)
r <- numeric(length=50)
color <- character(length=50)
for(t in 1:50) {
  xy <- exp(1i * t/2 - t/12)
  x[t] <- Re(xy)
  y[t] <- Im(xy)
  r[t] <- sqrt(x[t]^2 + y[t]^2)
  color[t] <- rgb(t/50,0,1-t/50)
}
```

```
# New class object
m <- GlowMapper4$new(xdim=500, ydim = 400, blend_mode = "additive")

# Map data on to raster
m$map(x=x, y=y, color = color, radius = r/4 + 0.1, distance_exponent = 2)

# Output raster data as a dataframe
pd <- m$output_dataframe(saturation = 1, saturation_mode = "overflow")

# Plot with ggplot
ggplot(pd, aes(x = x, y = y, fill = rgb(r,g,b,a))) +
  geom_raster(show.legend = FALSE) +
  scale_fill_identity() +
  coord_fixed(ratio = m$aspect(), xlim = m$xlim(), ylim = m$ylim(), expand = FALSE) +
  theme_night(bgcolor = "black")
```

`LightMapper` *LightMapper*

Description

This class provides a framework for creating scatter plots based on a point light simulation. Points are mapped with a inverse power gradient to a raster with specified dimensions and properties.

Usage

```
m <- LightMapper$new(xdim=1000, ydim=800, blend_mode = "screen", contrast_limit = 1e5, nthreads = 1)

m$map(x, y, radius, intensity = 1, radius, falloff_exponent = 1, distance_exponent = 2, xlimits = c(NA_r

m$output_raw(saturation = NA_real_)

m$output_dataframe(saturation = NA_real_)

m$aspect()

m$xlim()

m$ylim()
```

Arguments

xdim - The first dimension of the output matrix raster.

ydim - The second dimension of the output matrix raster.

blend_mode - Either screen or additive blending mode. See details.

nthreads - Number of threads to use.

x - X coordinate of points.

y - Y coordinate of points.

radius - Relative spread of glow intensity. The radius should be proportional to the x and y-ranges of the plot. Values between 1/10 to 1/100 of the range of the plot generally produce good results.

intensity - Maximum intensity at the center of a point.

falloff_exponent - Exponent to determine how fast light intensity decreases from the point origin. A value of 0.5 corresponds to a linear falloff; a value of 2 corresponds to an inverse square. Generally you want this value to be high, otherwise you'll flood your plot with light.

distance_exponent - Exponent of the distance calculation when calculating intensities. A value of 2 corresponds to euclidean distance; a value of 1 corresponds to manhattan distance.

xlimits - The x-limits of the output plot. If NA, the limits are +/- 5% of the maximum and minimum points.

ylimits - The y-limits of the output plot. If NA, the limits are +/- 5% of the maximum and minimum points.

append - Whether to add to the existing output or overwrite.

saturation - When retrieving the output with `$output_raw` or `$output_dataframe`, maximum intensity values are capped at the given value. This is often useful when using additive blend mode to increase contrast.

Details

`$new()` creates a new LightMapper object, which holds parameters, plotting data, and the output (a matrix of glow intensities). Creates a canvas to plot point data. With additive blending, the intensities of each point are added arithmetically, which is how light intensities are added in the physical world. This is equivalent to an fast/approximate un-normalized 2D kernel density estimate.

#' With "screen" blending, two intensities are added according to the formula: `I_out = 1 - (1-I_a)*(1-I_b)`. Both additive blending and screen blending are commutative operations, meaning the order of points in a plot does not affect the output.

Note: Mapping "lights" (inverse power intensity) is much slower than "glow" effects (gaussian intensities) for various reasons. Plotting more than a few hundred points with `LightMapper` or `LightMapper4` may be computationally prohibitive.

Screen blending can often improve contrast in a plot and is the default.

`$map()` maps points to the canvas.

`$output_raw()` output raw matrix rasters. Useful for plotting in base R.

`$output_dataframe()` output the raster as a dataframe with XY coordinates. This is meant to pipe directly into ggplot.

`$aspect()`, `$xlim()`, `$ylim()` return the aspect ratio, x-limits and y-limits of the raster. These functions are intended to be used with plotting functions (e.g. `ggplot2::coord_fixed()`) so that the output raster is not distorted. See example below.

Examples

```
# Plot Data: x,y,r
x <- numeric(length=50)
```

```
y <- numeric(length=50)
r <- numeric(length=50)
for(t in 1:50) {
  xy <- exp(1i * t/2 - t/12)
  x[t] <- Re(xy)
  y[t] <- Im(xy)
  r[t] <- sqrt(x[t]^2 + y[t]^2)
}

# New class object
m <- LightMapper$new(xdim=500, ydim = 400, blend_mode = "screen")

# Map data on to raster
m$map(x=x, y=y, intensity = 1, radius = r/100, falloff_exponent = 0.5, distance_exponent = 2)

# Output raster data as a dataframe
pd <- m$output_dataframe(saturation = 1)

# Plot with ggplot
ggplot(pd, aes(x = x, y = y, fill = value)) +
  geom_raster(show.legend = FALSE) +
  scale_fill_gradientn(colors=additive_alpha(c("black", "purple", "white"))) +
  coord_fixed(ratio = m$aspect(), xlim = m$xlim(), ylim = m$ylim(), expand = FALSE) +
  theme_night(bgcolor = "black")
```

LightMapper4 *LightMapper4*

Description

This class provides a framework for creating scatter plots based on a point light simulation with explicit color intensities. Points are mapped with a inverse power gradient to a raster with specified dimensions and properties.

Usage

```
m <- GlowMapper$new(xdim=1000, ydim=800, blend_mode = "additive", background_color = "#00000000", nthre

m$map(x, y, radius, color = NULL, r=NULL, g=NULL, b=NULL, falloff_exponent = 1, distance_exponent = 2, x

m$output_raw(saturation = 1, saturation_mode = "overflow")

m$output_dataframe(saturation = 1, saturation_mode = "overflow")

m$aspect()

m$xlim()

m$ylim()
```

Arguments

xdim - The first dimension of the output matrix raster.

ydim - The second dimension of the output matrix raster.

blend_mode - Either screen or additive blending mode. See details.

background_color - A color that can be coerced to RGBA with 'col2rgb', or a vector of four values between 0 and 1.

nthreads - Number of threads to use.

x - X coordinate of points.

y - Y coordinate of points.

radius - Relative spread of glow intensity. The radius should be proportional to the x and y-ranges of the plot. Values between 1/10 to 1/100 of the range of the plot generally produce good results.

color - Color of points. If NULL, r, g, and b parameters must be defined (and vice versa).

r - Red intensity of points. Must be between 0 and 1 if using screen blending.

g - Green intensity of points. Must be between 0 and 1 if using screen blending.

b - Blue intensity of points. Must be between 0 and 1 if using screen blending.

falloff_exponent - Exponent to determine how fast light intensity decreases from the point origin. A value of 0.5 corresponds to a linear falloff; a value of 2 corresponds to an inverse square. Generally you want this value to be high, otherwise you'll flood your plot with light.

distance_exponent - Exponent of the distance calculation when calculating intensities. A value of 2 corresponds to euclidean distance; a value of 1 corresponds to manhattan distance.

xlimits - The x-limits of the output plot. If NA, the limits are +/- 5% of the maximum and minimum points.

ylimits - The y-limits of the output plot. If NA, the limits are +/- 5% of the maximum and minimum points.

append - Whether to add to the existing output or overwrite.

saturation - When retrieving the output with `$output_raw` or `$output_dataframe`, maximum intensity values are capped at the given value. This is often useful when using additive blend mode to increase contrast.

saturation_mode - When intensity values are above the saturation threshold, values can be overflowed into other color channels ("overflow") or simply clipped at the threshold ("clip"). "Overflow" always produces a gradient to white for intensities above the threshold, which may produce artistically better results.

Details

This 'LightMapper4' class is similar to the 'LightMapper' class, but instead of a single intensity matrix output, color is specified explicitly.

`$new()` creates a new LightMapper4 object, which holds parameters, plotting data, and the output (a matrix of glow intensities). Creates a canvas to plot point data. With additive blending, the intensities of each point are added arithmetically, which is how light intensities are added in the physical world. This is equivalent to an fast/approximate un-normalized 2D kernel density estimate.

With "screen" blending, two intensities are added according to the formula: `I_out = 1 - (1-I_a)*(1-I_b)`. Both additive blending and screen blending are commutative operations, meaning the order of points in a plot does not affect the output.

Screen blending can often improve contrast in a plot and is the default.

`$map()` maps points to the canvas.

`$output_raw()` output raw matrix rasters (a list of four matrices, one for each RGBA channel). Useful for plotting in base R.

`$output_dataframe()` output the raster as a dataframe with XY coordinates. This is meant to pipe directly into ggplot.

`$aspect()`, `$xlim()`, `$ylim()` return the aspect ratio, x-limits and y-limits of the raster. These functions are intended to be used with plotting functions (e.g. `ggplot2::coord_fixed()`) so that the output raster is not distorted. See example below.

Examples

```
# Plot Data: x,y,r
x <- numeric(length=50)
y <- numeric(length=50)
r <- numeric(length=50)
color <- character(length=50)
for(t in 1:50) {
  xy <- exp(1i * t/2 - t/12)
  x[t] <- Re(xy)
  y[t] <- Im(xy)
  r[t] <- sqrt(x[t]^2 + y[t]^2)
  color[t] <- rgb(t/50,0,1-t/50)
}

# New class object
m <- LightMapper4$new(xdim=500, ydim = 400, blend_mode = "additive")

# Map data on to raster
m$map(x=x, y=y, color = color, radius = r/30+0.01, falloff_exponent = 1, distance_exponent = 2)

# Output raster data as a dataframe
pd <- m$output_dataframe(saturation = 1)

# Plot with ggplot
ggplot(pd, aes(x = x, y = y, fill = rgb(r,g,b,a))) +
  geom_raster(show.legend = FALSE) +
  scale_fill_identity() +
  coord_fixed(ratio = m$aspect(), xlim = m$xlim(), ylim = m$ylim(), expand = FALSE) +
  theme_night(bgcolor = "black")
```

`light_cool_colors` *light_cool_colors*

Description

A light color palette.

Usage

```
light_cool_colors(...)
```

Arguments

...	Arguments passed to the function returned by ‘colorRampPalette‘

Details

A simple light color palette gradient from dark blue to light blue intended for a heatmap with a white or light color background.

Equivalent to ‘colorRampPalette(c("#1133AA", "#CCFFFF"))(...)‘.

Value

A light color palette function

Examples

```
light_colors <- light_cool_colors(144)
plot(1:144, 1:144, col = light_colors, pch = 19)
```

---

light_heat_colors *light_heat_colors*

Description

A light color palette.

Usage

```
light_heat_colors(...)
```

Arguments

...	Arguments passed to the function returned by ‘colorRampPalette‘

Details

A simple light color palette gradient from red to orange to gold to yellow intended for a heatmap with a white or light color background.

Equivalent to ‘colorRampPalette(c("red", "darkorange2", "darkgoldenrod1", "gold1", "yellow2"))(...)‘.

Value

A light color palette function

Examples

```
light_colors <- light_heat_colors(144)
plot(1:144, 1:144, col = light_colors, pch = 19)
```

`map_colors` *map_colors*

Description

A helper function for manually mapping colors to a vector of colors.

Usage

```
map_colors(colors, x, min_limit=NULL, max_limit=NULL)
```

Arguments

`colors`	A vector of colors
`x`	A numeric vector of data
`min_limit`	The minimum value to scale the color to. Must be outside the range of data. If NULL, the minimum is determined by the data.
`max_limit`	The maximum value to scale the color to. Must be outside the range of data. If NULL, the maximum is determined by the data.

Details

A helper function for manually mapping colors to a vector of colors. It is useful when you want to manually specify color data, rather than relying on ggplot functions.

Value

A vector of colors.

Examples

```
cliff_points <- clifford_attractor(1e4, 1.886,-2.357,-0.328, 0.918, 0.1, 0)
color_pal <- circular_palette(n=144, pal_function=rainbow)
cliff_points$color <- map_colors(color_pal, cliff_points$angle, min_limit=-pi, max_limit=pi)

gm <- GlowMapper4$new(xdim=240, ydim=240, blend_mode = "additive", nthreads=1)
gm$map(x=cliff_points$x, y=cliff_points$y, radius=0.1, color=cliff_points$color)
pd <- gm$output_dataframe(saturation = 1)
```

```
ggplot() +
  geom_raster(data = pd, aes(x = x, y = y, fill = rgb(r,g,b,a)), show.legend=FALSE) +
  coord_fixed(gm$aspect(), xlim = gm$xlim(), ylim = gm$ylim()) +
  scale_fill_identity() +
  theme_night()
```

`mollweide_projection` *mollweide_projection*

Description

Performs a cartographic mollweide projection from polar coordinates (latitude/longitude) to X-Y map coordinates

Usage

```
mollweide_projection(latitude, longitude, meridian)
```

Arguments

`latitude`	Latitude (aka declination) of points
`longitude`	Longitude (aka right ascension) of points
`meridian`	The x=0 center of the plot

Details

This function uses the "Newton-Raphson with fast convergence everywhere" algorithm.

Latitude and longitude should be in units of radians not degrees. Latitude ranges from +/- pi/2 and longitude ranges from +/- pi.

Value

X/Y coordinates

See Also

https://en.wikipedia.org/wiki/Talk:Mollweide_projection

Examples

```
longitude <- pi / 4
latitude <- pi / 4
mollweide_projection(longitude, latitude, meridian = 0)
```

relxy *relxy*

Description

Helper functions for specifying the ‘radius‘ parameter in ‘GlowMapper$map‘ and similar.

Usage

```
relx(r, mode = "data")

rely(r, mode = "data")
```

Arguments

r	Radius of point data relative to X or Y range of the plot, a value between 0 and 1.
mode	One of "data" (default) or "plot". Whether to use a radius relative to the extent of the data range or the plot range (which could have been adjusted manually).

Details

Helper functions for specifying the ‘radius‘ parameter relative to the range of the plot as a proportion. I.e., a value of 0.02 corresponds to 2

Value

A class structure for input to ‘GlowMapper$map‘

Examples

```
gm <- GlowMapper$new(xdim=480, ydim=240, blend_mode = "additive", nthreads=4)

gm$map(x=1:10, y=runif(10)*100, radius=relx(0.01), intensity = 1)
pd <- gm$output_dataframe(saturation = 1)

ggplot() +
  geom_raster(data = pd, aes(x = x, y = y, fill = value), show.legend=FALSE) +
  coord_fixed(gm$aspect(), xlim = gm$xlim(), ylim = gm$ylim()) +
  scale_fill_gradientn(colors = additive_alpha(viridisLite::viridis(12))) +
  theme_night()

# Relative radius to y-range
gm$map(x=1:10, y=runif(10)*100, radius=rely(0.01))
pd <- gm$output_dataframe(saturation = 1)
ggplot() +
  geom_raster(data = pd, aes(x = x, y = y, fill = value), show.legend=FALSE) +
  coord_fixed(gm$aspect(), xlim = gm$xlim(), ylim = gm$ylim()) +
  scale_fill_gradientn(colors = additive_alpha(viridisLite::viridis(12))) +
  theme_night()
```

theme_night *theme_night*

Description

A dark ggplot2 theme with a default black background intended to be used with the glow package.

Usage

```
theme_night(bgcolor = "black", base_size = 14, base_family = "")
```

Arguments

bgcolor	Background color, default black. Generally you want to match the background with the lowest color value on a color scale.
base_size	Base default font size.
base_family	Base font family.

Details

The theme is heavily modified from the minimal ggplot theme. It is intended to be use with dark background colors and should not be used with white or light backgrounds.

Value

A ggplot2 theme.

Examples

```
ggplot(mtcars, aes(x = mpg, y = wt)) +
  geom_point(color = "white") +
  theme_night(bgcolor = "black")
```

Index