Package 'tmap.glyphs'

May 9, 2026

License GPL-3

Title Extension to 'tmap' for Creating Glyphs

Type Package

Description Provides new layer functions to 'tmap' for drawing glyphs. A glyph is a small chart (e.g., donut chart) shown at specific map locations to visualize multivariate or time-series data. The functions work with the syntax of 'tmap' and allow flexible control over size, layout, and appearance.

Version 0.1-1

Encoding UTF-8

Depends R (>= 3.5.0),

Imports tmap (>= 4.3), data.table

Config/Needs/check Nowosad/spDataLarge, lwgeom, r-tmap/tmap

Config/Needs/coverage Nowosad/spDataLarge, lwgeom

Config/Needs/website bookdown, rmarkdown, r-tmap/tmap

URL https://github.com/r-tmap/tmap.glyphs, https://r-tmap.github.io/tmap.glyphs/

BugReports https://github.com/r-tmap/tmap.glyphs/issues

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Martijn Tennekes [aut, cre]

Maintainer Martijn Tennekes <mtennekes@gmail.com>

Repository CRAN

Date/Publication 2026-05-09 14:40:33 UTC

Contents

`tmap.glyphs-package` *Extension for 'tmap': glyphs can be created*

Description

New layer functions available for 'tmap' that are used to create glyph maps.

Author(s)

Martijn Tennekes <mtennekes@gmail.com>

See Also

Useful links:

- https://github.com/r-tmap/tmap.glyphs
- https://r-tmap.github.io/tmap.glyphs/
- Report bugs at https://github.com/r-tmap/tmap.glyphs/issues

---

`opt_tm_flowers` *Map layer: flowers*

Description

Map layer that draw flowers as glyphs

Usage

```
opt_tm_flowers(
  start = 0,
  direction = 1,
  inner = 0.4,
  fill_hole = NA,
  points_only = "ifany",
  point_per = "feature",
  on_surface = FALSE,
  clustering = FALSE,
  icon.scale = 6,
  just = NA,
  grob.dim = c(width = 48, height = 48, render.width = 256, render.height = 256)
)

tm_flowers(
  parts = tmap::tm_vars(multivariate = TRUE),
```

```
  parts.scale = tm_scale_multi(),
  parts.legend = tmap::tm_legend_hide(),
  parts.chart = tmap::tm_chart_none(),
  parts.free = NA,
  size = tmap::tm_const(),
  size.scale = tmap::tm_scale(),
  size.legend = tmap::tm_legend(),
  size.chart = tmap::tm_chart_none(),
  size.free = NA,
  fill.scale = tmap::tm_scale(),
  fill.legend = tmap::tm_legend(),
  fill.chart = tmap::tm_chart_none(),
  fill.free = NA,
  col = tmap::tm_const(),
  col.scale = tmap::tm_scale(),
  col.legend = tmap::tm_legend(),
  col.chart = tmap::tm_chart_none(),
  col.free = NA,
  lwd = tmap::tm_const(),
  lwd.scale = tmap::tm_scale(),
  lwd.legend = tmap::tm_legend(),
  lwd.chart = tmap::tm_chart_none(),
  lwd.free = NA,
  plot.order = tmap::tm_plot_order("DATA", reverse = FALSE),
  zindex = NA,
  group = NA,
  group.control = "check",
  popup.vars = NA,
  popup.format = list(),
  hover = "",
  id = "",
  options = opt_tm_flowers()
)
```

Arguments

start	starting angle of the pies. 0 means top
direction	direction in which the pies are stacked. 1 means clockwise, 0 counterclockwise
inner	proportion of the inner circle
fill_hole	should the hole be filled? Either 'FALSE' or a fill color.
points_only	should only point geometries of the shape object (defined in [tm_shape()]) be plotted? By default '"ifany"', which means 'TRUE' in case a geometry collection is specified.
point_per	specification of how spatial points are mapped when the geometry is a multi line or a multi polygon. One of "feature", "segment" or "largest". The first generates a spatial point for every feature, the second for every segment (i.e. subfeature), the third only for the largest segment (subfeature). Note that the last two options can be significant slower.

`on_surface` In case of polygons, centroids are computed. Should the points be on the surface? If 'TRUE', which is slower than the default 'FALSE', centroids outside the surface are replaced with points computed with [sf::st_point_on_surface()].

`clustering` in interactive modes (e.g. `"view"` mode), should clustering be applied at lower zoom levels? Either 'FALSE' (default), 'TRUE', or a mode specific specification, e.g. for `"view"` mode `markerClusterOptions`.

`icon.scale` scaling number that determines how large the icons (or grobs) are in plot mode in comparison to proportional symbols (such as bubbles). For view mode, use the argument 'grob.dim'

`just` justification of the text relative to the point coordinates. Either one of the following values: `"left"` , `"right"`, `"center"`, `"bottom"`, and `"top"`, or a vector of two values where first value specifies horizontal and the second value vertical justification. Besides the mentioned values, also numeric values between 0 and 1 can be used. 0 means left justification for the first value and bottom justification for the second value. Note that in view mode, only one value is used.

`grob.dim` vector of four values that determine how grob objects (see details) are shown in view mode. The first and second value are the width and height of the displayed icon. The third and fourth value are the width and height of the rendered png image that is used for the icon. Generally, the third and fourth value should be large enough to render a graphic successfully. Only needed for the view mode.

`parts`, `parts.scale`, `parts.legend`, `parts.chart`, `parts.free`
Variables that determine the size of the parts

`size`, `size.scale`, `size.legend`, `size.chart`, `size.free`
Variables that determine the size of the donut

`col`, `col.scale`, `col.legend`, `col.chart`, `col.free`
Visual variable that determines the col color. See details.

`lwd`, `lwd.scale`, `lwd.legend`, `lwd.chart`, `lwd.free`
Visual variable that determines the line width. See details.

`plot.order` Specification in which order the spatial features are drawn. See [tm_plot_order()] for details.

`zindex` Map layers are drawn on top of each other. The 'zindex' numbers (one for each map layer) determines the stacking order. By default the map layers are drawn in the order they are called.

`group` Name of the group to which this layer belongs. This is only relevant in view mode, where layer groups can be switched (see 'group.control')

`group.control` In view mode, the group control determines how layer groups can be switched on and off. Options: '"radio"' for radio buttons (meaning only one group can be shown), '"check"' for check boxes (so multiple groups can be shown), and '"none"' for no control (the group cannot be (de)selected).

`popup.vars` names of data variables that are shown in the popups in '"view"' mode. Set popup.vars to 'TRUE' to show all variables in the shape object. Set popup.vars to 'FALSE' to disable popups. Set popup.vars to a character vector of variable names to those those variables in the popups. The default ('NA') depends on whether visual variables (e.g.'col') are used. If so, only those are shown. If not all variables in the shape object are shown.

popup.format — list of formatting options for the popup values. See the argument 'legend.format' for options. Only applicable for numeric data variables. If one list of formatting options is provided, it is applied to all numeric variables of 'popup.vars'. Also, a (named) list of lists can be provided. In that case, each list of formatting options is applied to the named variable.

hover — name of the data variable that specifies the hover labels (view mode only). Set to 'FALSE' to disable hover labels. By default 'FALSE', unless 'id' is specified. In that case, it is set to 'id',

id — name of the data variable that specifies the indices of the spatial features. Only used for '"view"' mode.

options — options passed on to the corresponding 'opt_<layer_function>' function

fill, fill.scale, fill.legend, fill.chart, fill.free — Visual variable that determines the fill color. See details.

Value

a [tmap::tmap-element], supposed to be stacked after [tmap::tm_shape()] using the '+' operator. The 'opt_<layer_function>' function returns a list that should be passed on to the 'options' argument.

Examples

```
library(tmap)

tm_shape(World) +
  tm_polygons(fill = "white", popup.vars = FALSE) +
tm_shape(World) +
 tm_flowers(
    parts = tm_vars(c("gender", "press", "footprint",
       "well_being", "inequality"), multivariate = TRUE),
fill.scale = tm_scale(values = "friendly5"),
size = 1.5,
    popup.vars = c("gender", "press", "footprint", "well_being","inequality"),
    id = "name") +
tm_basemap(NULL) +
tm_layout(bg.color = "grey90")


# make leaf sizes consistent: the larger, the better
# use ranking instead of values

q = function(x) {
r = rank(x)
r[is.na(x)] = NA
r = r / max(r, na.rm = TRUE)
r
}

World$rank_well_being = q((World$well_being / 8))
World$rank_footprint = q(((50 - World$footprint) / 50))
World$rank_inequality = q(((65 - World$inequality) / 65))
```

```
World$rank_press = q(1 - ((100 - World$press) / 100))
World$rank_gender = q(1 - World$gender)

tm_shape(World) +
  tm_polygons(fill = "white", popup.vars = FALSE) +
tm_shape(World) +
  tm_flowers(
    parts =
      tm_vars(c("rank_gender", "rank_press", "rank_footprint",
         "rank_well_being", "rank_inequality"), multivariate = TRUE),
    fill.scale = tm_scale(values = "friendly5"),
    size = 1.5,
    popup.vars = c("rank_gender", "rank_press", "rank_footprint",
        "rank_well_being","rank_inequality"), id = "name") +
tm_basemap(NULL) +
tm_layout(bg.color = "grey90")

ttmp()
```

`opt_tm_pies` *Map layer: donuts and pies*

Description

Map layer that draw donuts or pies as glyphs

Usage

```
opt_tm_pies(
  start = 0,
  direction = 1,
  inner = 0,
  fill_hole = FALSE,
  points_only = "ifany",
  point_per = "feature",
  on_surface = FALSE,
  icon.scale = 6,
  just = NA,
  grob.dim = c(width = 48, height = 48, render.width = 256, render.height = 256)
)

tm_pies(..., options = opt_tm_pies())

opt_tm_donuts(
  start = 0,
  direction = 1,
  inner = 0.4,
  fill_hole = NA,
```

```
  points_only = "ifany",
  point_per = "feature",
  on_surface = FALSE,
  clustering = FALSE,
  icon.scale = 6,
  just = NA,
  grob.dim = c(width = 48, height = 48, render.width = 256, render.height = 256)
)

tm_donuts(
  parts = tmap::tm_vars(multivariate = TRUE),
  parts.scale = tm_scale_composition(),
  parts.legend = tmap::tm_legend_hide(),
  parts.chart = tmap::tm_chart_none(),
  parts.free = NA,
  size = tmap::tm_const(),
  size.scale = tmap::tm_scale(),
  size.legend = tmap::tm_legend(),
  size.chart = tmap::tm_chart_none(),
  size.free = NA,
  fill.scale = tmap::tm_scale(),
  fill.legend = tmap::tm_legend(),
  fill.chart = tmap::tm_chart_none(),
  fill.free = NA,
  col = tmap::tm_const(),
  col.scale = tmap::tm_scale(),
  col.legend = tmap::tm_legend(),
  col.chart = tmap::tm_chart_none(),
  col.free = NA,
  lwd = tmap::tm_const(),
  lwd.scale = tmap::tm_scale(),
  lwd.legend = tmap::tm_legend(),
  lwd.chart = tmap::tm_chart_none(),
  lwd.free = NA,
  plot.order = tmap::tm_plot_order("DATA", reverse = FALSE),
  zindex = NA,
  group = NA,
  group.control = "check",
  popup.vars = NA,
  popup.format = list(),
  hover = "",
  id = "",
  options = opt_tm_donuts()
)
```

Arguments

start	starting angle of the pies. 0 means top

`direction` direction in which the pies are stacked. 1 means clockwise, 0 counterclockwise

`inner` proportion of the inner circle

`fill_hole` should the hole be filled? Either 'FALSE' or a fill color.

`points_only` should only point geometries of the shape object (defined in [tm_shape()]) be plotted? By default '"ifany"', which means 'TRUE' in case a geometry collection is specified.

`point_per` specification of how spatial points are mapped when the geometry is a multi line or a multi polygon. One of `"feature"`, `"segment"` or `"largest"`. The first generates a spatial point for every feature, the second for every segment (i.e. subfeature), the third only for the largest segment (subfeature). Note that the last two options can be significant slower.

`on_surface` In case of polygons, centroids are computed. Should the points be on the surface? If 'TRUE', which is slower than the default 'FALSE', centroids outside the surface are replaced with points computed with [sf::st_point_on_surface()].

`icon.scale` scaling number that determines how large the icons (or grobs) are in plot mode in comparison to proportional symbols (such as bubbles). For view mode, use the argument 'grob.dim'

`just` justification of the text relative to the point coordinates. Either one of the following values: `"left"` , `"right"`, `"center"`, `"bottom"`, and `"top"`, or a vector of two values where first value specifies horizontal and the second value vertical justification. Besides the mentioned values, also numeric values between 0 and 1 can be used. 0 means left justification for the first value and bottom justification for the second value. Note that in view mode, only one value is used.

`grob.dim` vector of four values that determine how grob objects (see details) are shown in view mode. The first and second value are the width and height of the displayed icon. The third and fourth value are the width and height of the rendered png image that is used for the icon. Generally, the third and fourth value should be large enough to render a graphic successfully. Only needed for the view mode.

`...` passed on to 'tm_donuts'

`options` options passed on to the corresponding 'opt_<layer_function>' function

`clustering` in interactive modes (e.g. `"view"` mode), should clustering be applied at lower zoom levels? Either 'FALSE' (default), 'TRUE', or a mode specific specification, e.g. for `"view"` mode `markerClusterOptions`.

`parts`, `parts.scale`, `parts.legend`, `parts.chart`, `parts.free`
Variables that determine the size of the parts

`size`, `size.scale`, `size.legend`, `size.chart`, `size.free`
Variables that determine the size of the donut

`col`, `col.scale`, `col.legend`, `col.chart`, `col.free`
Visual variable that determines the col color. See details.

`lwd`, `lwd.scale`, `lwd.legend`, `lwd.chart`, `lwd.free`
Visual variable that determines the line width. See details.

`plot.order` Specification in which order the spatial features are drawn. See [tm_plot_order()] for details.

`zindex` Map layers are drawn on top of each other. The ‘zindex‘ numbers (one for each map layer) determines the stacking order. By default the map layers are drawn in the order they are called.

`group` Name of the group to which this layer belongs. This is only relevant in view mode, where layer groups can be switched (see ‘group.control‘)

`group.control` In view mode, the group control determines how layer groups can be switched on and off. Options: ‘"radio"‘ for radio buttons (meaning only one group can be shown), ‘"check"‘ for check boxes (so multiple groups can be shown), and ‘"none"‘ for no control (the group cannot be (de)selected).

`popup.vars` names of data variables that are shown in the popups in ‘"view"‘ mode. Set popup.vars to ‘TRUE‘ to show all variables in the shape object. Set popup.vars to ‘FALSE‘ to disable popups. Set popup.vars to a character vector of variable names to those those variables in the popups. The default (‘NA‘) depends on whether visual variables (e.g.‘col‘) are used. If so, only those are shown. If not all variables in the shape object are shown.

`popup.format` list of formatting options for the popup values. See the argument ‘legend.format‘ for options. Only applicable for numeric data variables. If one list of formatting options is provided, it is applied to all numeric variables of ‘popup.vars‘. Also, a (named) list of lists can be provided. In that case, each list of formatting options is applied to the named variable.

`hover` name of the data variable that specifies the hover labels (view mode only). Set to ‘FALSE‘ to disable hover labels. By default ‘FALSE‘, unless ‘id‘ is specified. In that case, it is set to ‘id‘,

`id` name of the data variable that specifies the indices of the spatial features. Only used for ‘"view"‘ mode.

`fill`, `fill.scale`, `fill.legend`, `fill.chart`, `fill.free`
Visual variable that determines the fill color. See details.

Value

a [tmap::tmap-element], supposed to be stacked after [tmap::tm_shape()] using the ‘+‘ operator. The ‘opt_<layer_function>‘ function returns a list that should be passed on to the ‘options‘ argument.

Examples

```
library(tmap)

ZH_muni = NLD_muni[NLD_muni$province == "Zuid-Holland", ]

ZH_muni$income_middle = 100 - ZH_muni$income_high - ZH_muni$income_low

which.max(ZH_muni$population)

ZH_muni$population[c(10,26)] = 500000

ZH_muni$income_high[1:15] = NA

tm_shape(ZH_muni) +
```

```
  tm_polygons() +
  tm_donuts(
   parts = tm_vars(c("income_low", "income_middle", "income_high"), multivariate = TRUE),
    fill.scale = tm_scale_categorical(values = "-pu_gn_div"),
    size = "population",
    lwd = 1,
    size.scale = tm_scale_continuous(ticks = c(50000, 100000, 250000, 500000)),
    options = opt_tm_donuts(fill_hole = FALSE))


tm_shape(ZH_muni) +
tm_polygons() +
tm_pies(
    parts = tm_vars(c("income_low", "income_middle", "income_high"), multivariate = TRUE),
  fill.scale = tm_scale_categorical(values = "-pu_gn_div"),
  size = "population",
  lwd = 1,
  size.scale = tm_scale_continuous(ticks = c(50000, 100000, 250000, 500000)))
```

`tm_scale_composition` *Scales: composition*

Description

Scales in tmap are configured by the family of functions with prefix 'tm_scale'. Such function should be used for the input of the '.scale' arguments in the layer functions (e.g. 'fill.scale' in [tm_polygons()]). The function 'tm_scale_composition()' is used for the creation of composition glyphs, such as pie charts and donut charts.

Usage

```
tm_scale_composition(
  values = NA,
  values.repeat = FALSE,
  values.range = NA,
  values.scale = 1,
  value.na = NA,
  value.null = NA,
  value.neutral = NA,
  labels = NULL,
  label.na = NA,
  label.null = NA
)
```

Arguments

values	(generic scale argument) The visual values. For colors (e.g. 'fill' or 'col' for 'tm_polygons()') this is a palette name from the 'cols4all' package (see [cols4all::c4a()]) or vector of colors, for size (e.g. 'size' for 'tm_symbols()')

these are a set of sizes (if two values are specified they are interpret as range), for symbol shapes (e.g. ‘shape‘ for [tm_symbols()]) these are a set of symbols, etc. The tmap option ‘values.var‘ contains the default values per visual variable and in some cases also per data type.

`values.repeat` (generic scale argument) Should the values be repeated in case there are more categories?

`values.range` (generic scale argument) Range of the values. Vector of two numbers (both between 0 and 1) where the first determines the minimum and the second the maximum. Full range, which means that all values are used, is encoded as ‘c(0, 1)‘. For instance, when a grey scale is used for color (from black to white), ‘c(0,1)‘ means that all colors are used, ‘0.25, 0.75‘ means that only colors from dark grey to light grey are used (more precisely ‘"grey25"‘ to ‘"grey75"‘), and ‘0, 0.5‘ means that only colors are used from black to middle grey (‘"grey50"‘). When only one number is specified, this is interpreted as the second number (where the first is set to 0). Default values can be set via the tmap option ‘values.range‘.

`values.scale` (generic scale argument) Scaling of the values. Only useful for size-related visual variables, such as ‘size‘ of [tm_symbols()] and ‘lwd‘ of [tm_lines()].

`value.na` (generic scale argument) Value used for missing values. See tmap option ‘"value.na"‘ for defaults per visual variable.

`value.null` (generic scale argument) Value used for NULL values. See tmap option ‘"value.null"‘ for defaults per visual variable. Null data values occur when out-of-scope features are shown (e.g. for a map of Europe showing a data variable per country, the null values are applied to countries outside Europe).

`value.neutral` (generic scale argument) Value that can be considered neutral. This is used for legends of other visual variables of the same map layer. E.g. when both ‘fill‘ and ‘size‘ are used for [tm_symbols()] (using filled circles), the size legend items are filled with the ‘value.neutral‘ color from the ‘fill.scale‘ scale, and fill legend items are bubbles of size ‘value.neutral‘ from the ‘size.scale‘ scale.

`labels` (generic scale argument) Labels

`label.na` (generic scale argument) Label for missing values

`label.null` (generic scale argument) Label for null (out-of-scope) values

Value

tmap scale object to be used for the ‘.scale‘ arguments in the tmap layer functions

---

`tm_scale_multi` *Scales: multivariate*

Description

Scales in tmap are configured by the family of functions with prefix ‘tm_scale‘. Such function should be used for the input of the ‘.scale‘ arguments in the layer functions (e.g. ‘fill.scale‘ in [tm_polygons()]). The function ‘tm_scale_multi()‘ is used for the creation of glyphs, which take one or multiple normalized (between 0 and 1) values. E.g. the flower glyph.

Usage

```
tm_scale_multi(
  values = NA,
  values.repeat = FALSE,
  values.range = NA,
  values.scale = 1,
  value.na = NA,
  value.null = NA,
  value.neutral = NA,
  labels = NULL,
  label.na = NA,
  label.null = NA
)
```

Arguments

values	(generic scale argument) The visual values. For colors (e.g. 'fill' or 'col' for 'tm_polygons()') this is a palette name from the 'cols4all' package (see [cols4all::c4a()]) or vector of colors, for size (e.g. 'size' for 'tm_symbols()') these are a set of sizes (if two values are specified they are interpret as range), for symbol shapes (e.g. 'shape' for [tm_symbols()]) these are a set of symbols, etc. The tmap option 'values.var' contains the default values per visual variable and in some cases also per data type.
values.repeat	(generic scale argument) Should the values be repeated in case there are more categories?
values.range	(generic scale argument) Range of the values. Vector of two numbers (both between 0 and 1) where the first determines the minimum and the second the maximum. Full range, which means that all values are used, is encoded as 'c(0, 1)'. For instance, when a grey scale is used for color (from black to white), 'c(0,1)' means that all colors are used, '0.25, 0.75' means that only colors from dark grey to light grey are used (more precisely '"grey25"' to '"grey75"'), and '0, 0.5' means that only colors are used from black to middle grey ('"grey50"'). When only one number is specified, this is interpreted as the second number (where the first is set to 0). Default values can be set via the tmap option 'values.range'.
values.scale	(generic scale argument) Scaling of the values. Only useful for size-related visual variables, such as 'size' of [tm_symbols()] and 'lwd' of [tm_lines()].
value.na	(generic scale argument) Value used for missing values. See tmap option '"value.na"' for defaults per visual variable.
value.null	(generic scale argument) Value used for NULL values. See tmap option '"value.null"' for defaults per visual variable. Null data values occur when out-of-scope features are shown (e.g. for a map of Europe showing a data variable per country, the null values are applied to countries outside Europe).
value.neutral	(generic scale argument) Value that can be considered neutral. This is used for legends of other visual variables of the same map layer. E.g. when both 'fill' and 'size' are used for [tm_symbols()] (using filled circles), the size legend items

	are filled with the ‘value.neutral‘ color from the ‘fill.scale‘ scale, and fill legend items are bubbles of size ‘value.neutral‘ from the ‘size.scale‘ scale.
`labels`	(generic scale argument) Labels
`label.na`	(generic scale argument) Label for missing values
`label.null`	(generic scale argument) Label for null (out-of-scope) values

Value

tmap scale object to be used for the ‘.scale‘ arguments in the tmap layer functions

Index