Package 'tmap.mapgl'

May 9, 2026

Title Extensions to 'tmap' with Two New Modes: 'mapbox' and 'maplibre'

Type Package

Description The 'tmap' package provides two plotting modes for static and interactive thematic maps. This package extends 'tmap' with two additional modes based on 'Mapbox GL JS' and 'MapLibre GL JS'. These modes feature interactive vector tiles, globe views, and other modern web-mapping capabilities, while maintaining a consistent 'tmap' interface across all plotting modes.

License GPL-3

Version 0.2-1

Encoding UTF-8

Depends R (>= 4.1),

Imports tmap (>= 4.3), mapgl (>= 0.4.1), terra, rlang, cli, stars, data.table, grDevices, colorspace, htmltools, htmlwidgets, sf, tmaptools, units

Suggests knitr

Config/Needs/website bookdown, rmarkdown, r-tmap/tmap, walkerke/mapgl, dplyr, osmdata, mapview

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

BugReports https://github.com/r-tmap/tmap.mapgl/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 14:40:29 UTC

Contents

`tmap.mapgl-package` *Extensions to 'tmap' with Two New Modes: 'mapbox' and 'maplibre'*

Description

The 'tmap' package provides two plotting modes for static and interactive thematic maps. This package extends 'tmap' with two additional modes based on 'mapbox' <https://mapbox.com> and 'maplibre' <https://maplibre.org>. These modes feature interactive vector tiles, globe views, and other modern web-mapping capabilities, while using the same 'tmap' syntax for creating maps (Tennekes 2018, <doi:10.32614/RJ-2018-027>).

Author(s)

Martijn Tennekes <mtennekes@gmail.com>

See Also

Useful links:

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

Examples

```
## Not run:
library(tmap)
library(tmap.mapgl)
# getting API: https://walker-data.com/mapgl/articles/getting-started.html
# check API envir var: Sys.getenv("MAPBOX_PUBLIC_TOKEN")

tmap_mode("mapbox")
tm_shape(World) +
  tm_polygons("HPI", fill.scale = tm_scale_intervals(values = "brewer.rd_yl_gn"))
```

```
tm_shape(NLD_dist) +
  tm_polygons("employment_rate",
    fill.scale = tm_scale_intervals(values = "scico.roma"),
    lwd = 0.1) +
tm_shape(NLD_muni) +
  tm_polygons(fill = NULL, lwd = 1) +
tm_mapbox(pitch = 60) +
tm_basemap("mapbox.dark")

## End(Not run)
```

---

tmap_mapbox *Export tmap to mapbox and maplibre*

Description

* ‘tmap_mapbox()‘ returns a [‘mapgl‘][mapgl::mapboxgl()] object (‘"mapbox" mode‘) * ‘tmap_maplibre()‘ a [‘mapgl‘][mapgl::maplibregl()] object (‘"maplibre"‘ mode).

Usage

```
tmap_mapbox(x, show = FALSE, ...)

tmap_maplibre(x, show = FALSE, ...)
```

Arguments

x	a tmap object.
show	show the map?
...	passed on to [‘tmap‘][tmap::print.tmap()]

Value

a [‘mapgl‘][mapgl::mapboxgl()] object (‘"mapbox" mode‘) or a [‘mapgl‘][mapgl::maplibregl()] object (‘"maplibre"‘ mode). In case small multiples are shown, a list is returned.

Examples

```
library(tmap)
library(tmap.mapgl)
map = tm_shape(World) + tm_polygons()
tmap_maplibre(map, show = TRUE)
```

`tm_draw` *Map component: draw toolbox*

Description

Map component that adds a draw toolbox

Usage

```
tm_draw(download_button, show_measurements, stack, position, group_id, z)
```

Arguments

`download_button`	show download button? By default ‘TRUE‘.
`show_measurements`	show measurements? By default ‘TRUE‘. The measurement unit is set in [‘tmap‘][tmap::tm_shape()]
`stack`	stack with other map components, either ‘"vertical"‘ or ‘"horizontal"‘.
`position`	The position specification of the component: an object created with ‘tm_pos_in()‘ or ‘tm_pos_out()‘. Or, as a shortcut, a vector of two values, specifying the x and y coordinates. The first is ‘"left"‘, ‘"center"‘ or ‘"right"‘ (or upper case, meaning tighter to the map frame), the second ‘"top"‘, ‘"center"‘ or ‘"bottom"‘. Numeric values are also supported, where 0, 0 means left bottom and 1, 1 right top. See also vignette about positioning. In case multiple components should be combined (stacked), use ‘group_id‘ and specify ‘component‘ in [tm_comp_group()].
`group_id`	Component group id name. All components (e.g. legends, titles, etc) with the same ‘group_id‘ will be grouped. The specifications of how they are placed (e.g. stacking, margins etc.) are determined in [tm_comp_group()] where its argument ‘id‘ should correspond to ‘group_id‘.
`z`	z index, e.g. the place of the component relative to the other componets

Value

a [tmap::tmap-element]

See Also

Vignette about components

tm_fullscreen *Map component: fullscreen button*

Description

Map component that adds a fullscreen button

Usage

```
tm_fullscreen(stack, position, group_id, z)
```

Arguments

stack	stack with other map components, either ‘"vertical"‘ or ‘"horizontal"‘.
position	The position specification of the component: an object created with ‘tm_pos_in()‘ or ‘tm_pos_out()‘. Or, as a shortcut, a vector of two values, specifying the x and y coordinates. The first is ‘"left"‘, ‘"center"‘ or ‘"right"‘ (or upper case, meaning tighter to the map frame), the second ‘"top"‘, ‘"center"‘ or ‘"bottom"‘. Numeric values are also supported, where 0, 0 means left bottom and 1, 1 right top. See also vignette about positioning. In case multiple components should be combined (stacked), use ‘group_id‘ and specify ‘component‘ in [tm_comp_group()].
group_id	Component group id name. All components (e.g. legends, titles, etc) with the same ‘group_id‘ will be grouped. The specifications of how they are placed (e.g. stacking, margins etc.) are determined in [tm_comp_group()] where its argument ‘id‘ should correspond to ‘group_id‘.
z	z index, e.g. the place of the component relative to the other componets

Value

a [tmap::tmap-element]

See Also

Vignette about components

tm_geocoder *Map component: geocoder search bar*

Description

Map component that adds a geocoder search bar

Usage

```
tm_geocoder(placeholder, stack, position, group_id, z)
```

Arguments

placeholder	placeholder text. Default is ‘"Search"‘.
stack	stack with other map components, either ‘"vertical"‘ or ‘"horizontal"‘.
position	The position specification of the component: an object created with ‘tm_pos_in()‘ or ‘tm_pos_out()‘. Or, as a shortcut, a vector of two values, specifying the x and y coordinates. The first is ‘"left"‘, ‘"center"‘ or ‘"right"‘ (or upper case, meaning tighter to the map frame), the second ‘"top"‘, ‘"center"‘ or ‘"bottom"‘. Numeric values are also supported, where 0, 0 means left bottom and 1, 1 right top. See also vignette about positioning. In case multiple components should be combined (stacked), use ‘group_id‘ and specify ‘component‘ in [tm_comp_group()].
group_id	Component group id name. All components (e.g. legends, titles, etc) with the same ‘group_id‘ will be grouped. The specifications of how they are placed (e.g. stacking, margins etc.) are determined in [tm_comp_group()] where its argument ‘id‘ should correspond to ‘group_id‘.
z	z index, e.g. the place of the component relative to the other componets

Value

a [tmap::tmap-element]

See Also

Vignette about components

---

tm_mapbox *Mapbox mode options*

Description

Mapbox mode options. These options are specific to the mapbox mode.

Usage

```
tm_mapbox(pitch, control.position, control.collapse, zoom)
```

Arguments

pitch	The pitch angle
control.position	The position of the layer control box
control.collapse	Should the layer control box be collapsed?
zoom	The zoom level of the map

Value

a [tmap::tmap-element]

Examples

```
## Not run:
library(tmap)
library(tmap.mapgl)
# getting API: https://walker-data.com/mapgl/articles/getting-started.html
# check API envir var: Sys.getenv("MAPBOX_PUBLIC_TOKEN")

tmap_mode("mapbox")
tm_shape(World) +
  tm_polygons("HPI", fill.scale = tm_scale_intervals(values = "brewer.rd_yl_gn"))

tm_shape(NLD_dist) +
  tm_polygons("employment_rate",
    fill.scale = tm_scale_intervals(values = "scico.roma"),
    lwd = 0.1) +
tm_shape(NLD_muni) +
  tm_polygons(fill = NULL, lwd = 1) +
tm_mapbox(pitch = 60) +
tm_basemap("mapbox.dark")

## End(Not run)
```

---

tm_maplibre *Maplibre mode options*

Description

Maplibre mode options. These options are specific to the maplibre mode.

Usage

```
tm_maplibre(pitch, control.position, control.collapse, zoom)
```

Arguments

pitch	The pitch angle
control.position	The position of the layer control box
control.collapse	Should the layer control box be collapsed?
zoom	The zoom level of the map

Value

a [tmap::tmap-element]

Examples

```
library(tmap)
library(tmap.mapgl)
tmap_mode("maplibre")
tm_shape(World) +
  tm_polygons("HPI", fill.scale = tm_scale_intervals(values = "brewer.rd_yl_gn"))

tm_shape(NLD_dist) +
  tm_polygons("employment_rate",
    fill.scale = tm_scale_intervals(values = "scico.roma"),
    lwd = 0.1) +
  tm_shape(NLD_muni) +
tm_polygons(fill = NULL, lwd = 1) +
tm_mapbox(pitch = 60) +
tm_basemap("carto.dark_matter")
```

---

tm_polygons_3d *Map layer: polygons in 3d (experimental)*

Description

Map layer that draws 3d (extruded) polygons. Supported visual variables are: 'height' (the), 'fill' (the fill color), 'col' (the border color), 'lwd' (line width), 'lty' (line type), 'fill_alpha' (fill color alpha transparency) and 'col_alpha' (border color alpha transparency).

Usage

```
tm_polygons_3d(
  height = tmap::tm_const(),
  height.scale = tmap::tm_scale(),
  height.legend = tmap::tm_legend_hide(),
  height.chart = tmap::tm_chart_none(),
  height.free = NA,
  fill = tmap::tm_const(),
  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,
  lty = tmap::tm_const(),
  lty.scale = tmap::tm_scale(),
  lty.legend = tmap::tm_legend(),
  lty.chart = tmap::tm_chart_none(),
  lty.free = NA,
  fill_alpha = tmap::tm_const(),
  fill_alpha.scale = tmap::tm_scale(),
  fill_alpha.legend = tmap::tm_legend(),
  fill_alpha.chart = tmap::tm_chart_none(),
  fill_alpha.free = NA,
  col_alpha = tmap::tm_const(),
  col_alpha.scale = tmap::tm_scale(),
  col_alpha.legend = tmap::tm_legend(),
  col_alpha.chart = tmap::tm_chart_none(),
  col_alpha.free = NA,
  linejoin = "round",
  lineend = "round",
  plot.order = tmap::tm_plot_order("lwd", reverse = TRUE, na.order = "bottom"),
  zindex = NA,
  group = NA,
  group.control = "check",
  popup.vars = NA,
  popup.format = list(),
  hover = NA,
  id = "",
  options = opt_tm_polygons_3d()
)

opt_tm_polygons_3d(
  polygons.only = "ifany",
  height.max = "10%",
  height.min = "0.1%"
)
```

Arguments

`height`, `height.scale`, `height.legend`, `height.chart`, `height.free`
: ‘r .doc_vv("height")‘

`fill`, `fill.scale`, `fill.legend`, `fill.chart`, `fill.free`
: ‘r .doc_vv("fill")‘

`col`, `col.scale`, `col.legend`, `col.chart`, `col.free`
: ‘r .doc_vv("col")‘

`lwd`, `lwd.scale`, `lwd.legend`, `lwd.chart`, `lwd.free`
: ‘r .doc_vv("lwd")‘

`lty`, `lty.scale`, `lty.legend`, `lty.chart`, `lty.free`
: ‘r .doc_vv("lty")‘

`fill_alpha`, `fill_alpha.scale`, `fill_alpha.chart`, `fill_alpha.legend`, `fill_alpha.free`
: ‘r .doc_vv("fill_alpha")‘

`col_alpha`, `col_alpha.scale`, `col_alpha.legend`, `col_alpha.chart`, `col_alpha.free`
: ‘r .doc_vv("col_alpha")‘

`linejoin`, `lineend`
: Line join and line end. See [gpar()][grid::gpar()] for 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.‘fill‘) 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

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

`height.max`, `height.min`
: Maximum and minimum height. The data variable values assigned to ‘height‘ are are scaled between these two values. Each can be an absolute number (in meters) or relative to the bounding box area, more specifically, the percentage of the square root of the bounding box area. The default values are ‘"10%"‘ for ‘height.max‘ and ‘0.1%‘ for ‘height.min‘. The latter is not set to 0 because of artefacts with maplibre in globe view.

Details

The visual variable arguments (e.g. ‘col‘) can be specified with either a data variable name (e.g., a spatial vector attribute or a raster layer of the object specified in [tm_shape()]), or with a visual value (for ‘col‘, a color is expected). See vignette about visual variables.

Multiple values can be specified: in that case facets are created. These facets can be combined with other faceting data variables, specified with [tm_facets()]. See vignette about facets.

- The ‘*.scale‘ arguments determine the used scale to map the data values to visual variable values. These can be specified with one of the available ‘tm_scale_*()‘ functions. The default is specified by the tmap option ([tm_options()]) ‘scales.var‘. See vignette about scales.

- The ‘*.legend‘ arguments determine the used legend, specified with [tm_legend()]. The default legend and its settings are determined by the tmap options ([tm_options()]) ‘legend.‘ . See vignette about legends.

- The ‘*.chart‘ arguments specify additional charts, specified with ‘tm_chart_‘, e.g. [tm_chart_histogram()]. See vignette about charts.

- The ‘*.free‘ arguments determine whether scales are applied freely across facets, or shared. A logical value is required. They can also be specified with a vector of three logical values; these determine whether scales are applied freely per facet dimension. This is only useful when facets are applied (see [tm_facets()]). There are maximally three facet dimensions: rows, columns, and pages. This only applies for a facet grid ([tm_facets_grid()]). For instance, ‘col.free = c(TRUE, FALSE, FALSE)‘ means that for the visual variable ‘col‘, each row of facets will have its own scale, and therefore its own legend. For facet wraps and stacks ([tm_facets_wrap()] and [tm_facets_stack()]) there is only one facet dimension, so the ‘*.free‘ argument requires only one logical value.

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.

See Also

Choropleth example (1) and choropleth example (2)

Examples

```
library(tmap)
library(tmap.mapgl)
tmap_mode("maplibre")
NLD_dist$pop_dens = NLD_dist$population / NLD_dist$area
tm_shape(NLD_dist) +
  tm_polygons_3d(height = "pop_dens",
    fill = "edu_appl_sci",
    fill.scale = tm_scale_intervals(style = "kmeans", values = "-pu_gn"),
    fill.legend = tm_legend("Univeristy degree")) +
tm_maplibre(pitch = 45) +
tm_crs(3857)
```

tm_rain *Map layer: let it rain!*

Description

Map layer that generates rain. Only available in "mapbox" mode.

Usage

```
tm_rain(
  density = 0.5,
  intensity = 1,
  color = "#a8adbc",
  opacity = 0.7,
  center_thinning = 0.57,
  direction = c(0, 80),
  droplet_size = c(2.6, 18.2),
  distortion_strength = 0.7,
  vignette = 1,
  vignette_color = "#464646"
)
```

Arguments

density	A number between 0 and 1 controlling the rain particles density. Default is 0.5.
intensity	A number between 0 and 1 controlling the rain particles movement speed. Default is 1.
color	A string specifying the color of the rain droplets. Default is "#a8adbc".
opacity	A number between 0 and 1 controlling the rain particles opacity. Default is 0.7.
center_thinning	A number between 0 and 1 controlling the thinning factor of rain particles from center. Default is 0.57.
direction	A numeric vector of length 2 defining the azimuth and polar angles of the rain direction. Default is c(0, 80).
droplet_size	A numeric vector of length 2 controlling the rain droplet size (x - normal to direction, y - along direction). Default is c(2.6, 18.2).
distortion_strength	A number between 0 and 1 controlling the rain particles screen-space distortion strength. Default is 0.7.
vignette	A number between 0 and 1 controlling the screen-space vignette rain tinting effect intensity. Default is 1.0.
vignette_color	A string specifying the rain vignette screen-space corners tint color. Default is "#464646".

Details

Arguments and their default values have been taken from [mapbox::set_rain()] (package version 0.2.2)

Value

tmap layer

See Also

[tm_show()]

---

tm_snow *Map layer: let it snow!*

Description

Map layer that generates snow.

Usage

```
tm_snow(
  density = 0.85,
  intensity = 1,
  color = "#ffffff",
  opacity = 1,
  center_thinning = 0.4,
  direction = c(0, 50),
  flake_size = 0.71,
  vignette = 0.3,
  vignette_color = "#ffffff"
)
```

Arguments

density	A number between 0 and 1 controlling the snow particles density. Default is 0.85.
intensity	A number between 0 and 1 controlling the snow particles movement speed. Default is 1.0.
color	A string specifying the color of the snow particles. Default is "#ffffff".
opacity	A number between 0 and 1 controlling the snow particles opacity. Default is 1.0.
center_thinning	A number between 0 and 1 controlling the thinning factor of snow particles from center. Default is 0.4.
direction	A numeric vector of length 2 defining the azimuth and polar angles of the snow direction. Default is c(0, 50).

`flake_size`	A number between 0 and 5 controlling the snow flake particle size. Default is 0.71.
`vignette`	A number between 0 and 1 controlling the snow vignette screen-space effect. Default is 0.3.
`vignette_color`	A string specifying the snow vignette screen-space corners tint color. Default is "#ffffff".

Details

Arguments and their default values have been taken from [mapbox::set_snow()] (package version 0.2.2)

Value

tmap layer

See Also

[tm_rain()]

Index