Package 'tmap'

January 5, 2019

License GPL-3

Title Thematic Maps

Type Package

LazyLoad yes

Description Thematic maps are geographical maps in which spatial data distributions are visualized. This package offers a flexible, layer-based, and easy to use approach to create thematic maps, such as choropleths and bubble maps.

Version 2.2

Date 2019-01-04

Depends R (>= 3.0), methods

Imports tmaptools (>= 2.0-1), sf (>= 0.7-1), units (>= 0.6-1), raster (>= 2.7-15), grid, RColorBrewer, viridisLite, classInt, htmltools, htmlwidgets, leaflet (>= 2.0.2), mapview, stats

Suggests lwgeom (>= 0.1-4), rmapshaper, rmarkdown, knitr, png, cartogram, osmdata, ggplot2, dplyr, tidyr, testthat, covr

URL https://github.com/mtennekes/tmap

BugReports https://github.com/mtennekes/tmap/issues

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation no

Author Martijn Tennekes [aut, cre],
Joel Gombin [ctb],
Sebastian Jeworutzki [ctb],
Kent Russell [ctb],
Richard Zijdeman [ctb],
John Clouse [ctb],
Robin Lovelace [ctb],
Jakub Nowosad [ctb],
Jannes Muenchow [ctb]

Maintainer Martijn Tennekes <mtennekes@gmail.com>

Repository CRAN

Date/Publication 2019-01-05 08:10:03 UTC

R topics documented:

tmap-package . . . 3
+.tmap . . . 6
land . . . 6
metro . . . 7
print.tmap . . . 7
qtm . . . 8
rivers . . . 11
theme_ps . . . 12
tmap-element . . . 12
tmap_animation . . . 14
tmap_arrange . . . 15
tmap_format . . . 16
tmap_icons . . . 17
tmap_last . . . 18
tmap_leaflet . . . 19
tmap_mode . . . 20
tmap_options . . . 22
tmap_save . . . 26
tmap_style . . . 28
tmap_style_catalogue . . . 29
tmap_tip . . . 29
tm_add_legend . . . 30
tm_basemap . . . 31
tm_compass . . . 32
tm_credits . . . 33
tm_facets . . . 35
tm_fill . . . 39
tm_grid . . . 44
tm_iso . . . 46
tm_layout . . . 47
tm_lines . . . 55
tm_logo . . . 60
tm_minimap . . . 61
tm_raster . . . 62
tm_scale_bar . . . 66
tm_sf . . . 67
tm_shape . . . 69
tm_symbols . . . 71
tm_text . . . 81
tm_view . . . 87
tm_xlab . . . 89
World . . . 90

Index **91**

tmap-package *Thematic Map Visualization*

Description

Thematic maps are geographical maps in which spatial data distributions are visualized. This package offers a flexible, layer-based, and easy to use approach to create thematic maps, such as choropleths and bubble maps. It is based on the grammar of graphics, and resembles the syntax of ggplot2.

Details

This page provides a brief overview of all package functions. See `vignette("tmap-getstarted")` for a short introduction with examples.

Quick plotting method

qtm	Plot a thematic map

Main plotting method

Shape specification:

tm_shape	Specify a shape object

Aesthetics base layers:

tm_polygons	Create a polygon layer (with borders)
tm_symbols	Create a layer of symbols
tm_lines	Create a layer of lines
tm_raster	Create a raster layer
tm_text	Create a layer of text labels
tm_basemap	Create a layer of basemap tiles
tm_tiles	Create a layer of overlay tiles

Aesthetics derived layers:

tm_fill	Create a polygon layer (without borders)
tm_borders	Create polygon borders

tm_bubbles	Create a layer of bubbles
tm_squares	Create a layer of squares
tm_dots	Create a layer of dots
tm_markers	Create a layer of markers
tm_iso	Create a layer of iso/contour lines
tm_rgb	Create a raster layer of an image

Faceting (small multiples)

tm_facets	Define facets

Attributes:

tm_grid	Create grid lines
tm_scale_bar	Create a scale bar
tm_compass	Create a map compass
tm_credits	Create a text for credits
tm_logo	Create a logo
tm_xlab and tm_ylab	Create axis labels
tm_minimap	Create a minimap (view mode only)

Layout element:

tm_layout	Adjust the layout (main function)
tm_legend	Adjust the legend
tm_view	Configure the interactive view mode
tm_style	Apply a predefined style
tm_format	Apply a predefined format

Change options:

tmap_mode	Set the tmap mode: "plot" or "view"
ttm	Toggle between the modes
tmap_options	Set global tmap options (from tm_layout, tm_view, and a couple of others)
tmap_style	Set the default style

Create icons:

tmap_icons	Specify icons for markers or proportional symbols

Output functions

print	Plot in graphics device or view interactively in web browser or RStudio's viewer pane
tmap_last	Redraw the last map
tmap_leaflet	Obtain a leaflet widget object
tmap_animation	Create an animation
tmap_arrange	Create small multiples of separate maps
tmap_save	Save thematic maps (either as image or HTML file)

Spatial datasets

World	World country data (sf object of polygons)
NLD_prov	Netherlands province data (sf object of polygons)
NLD_muni	Netherlands municipal data (sf object of polygons)
metro	Metropolitan areas (sf object of points)
rivers	Rivers (sf object of lines)
land	Global land cover (RasterBrick object)

Author(s)

Martijn Tennekes <mtennekes@gmail.com>

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

```
vignette("tmap-getstarted")
```

+.tmap *Stacking of tmap elements*

Description

The plus operator allows you to stack tmap-elements, and groups of tmap-elements.

Usage

```
## S3 method for class 'tmap'
e1 + e2
```

Arguments

e1	first tmap-element
e2	second tmap-element

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

tmap-element and vignette("tmap-getstarted")

land *Spatial data of global land cover*

Description

Spatial data of global land cover, of class RasterBrick. The data includes a population times series from 1950 to (forecasted) 2030. All metro areas with over 1 million inhabitants in 2010 are included.

Usage

```
data(land)
```

Details

Important: publication of these maps is only allowed when cited to Tateishi et al. (2014), and when "Geospatial Information Authority of Japan, Chiba University and collaborating organizations." is shown. See http://www.iscgm.org/gm/glcnmo.html#use.

Source

http://www.iscgm.org/gm/glcnmo.html

References

Production of Global Land Cover Data - GLCNMO2008, Tateishi, R., Thanh Hoan, N., Kobayashi, T., Alsaaideh, B., Tana, G., Xuan Phong, D. (2014), Journal of Geography and Geology, 6 (3).

---

metro *Spatial data of metropolitan areas*

Description

Spatial data of metropolitan areas, of class sf. The data includes a population times series from 1950 to (forecasted) 2030. All metro areas with over 1 million inhabitants in 2010 are included.

Usage

```
data(metro)
```

Source

https://esa.un.org/unpd/wup/

References

United Nations, Department of Economic and Social Affairs, Population Division (2014). World Urbanization Prospects: The 2014 Revision, CD-ROM Edition.

---

print.tmap *Draw thematic map*

Description

Draw thematic map. If the tmap mode is set to "plot" (see tmap_mode), the map is plot in the current graphics device. If the mode is set to "view", the map is shown interactively as an htmlwidget.

Usage

```
## S3 method for class 'tmap'
print(x, vp = NULL, return.asp = FALSE,
  mode = getOption("tmap.mode"), show = TRUE, knit = FALSE,
  options = NULL, ...)

knit_print.tmap(x, ..., options = NULL)
```

Arguments

x	tmap object. A tmap object is created with qtm or by stacking tmap-elements.
vp	viewport to draw the plot in. This is particularly useful for insets.
return.asp	Logical that determines whether the aspect ratio of the map is returned. In that case, grid.newpage() will be called, but without plotting of the map. This is used by tmap_save to determine the aspect ratio of the map.
mode	the mode of tmap: "plot" (static) or "view" (interactive). See tmap_mode for details.
show	logical that determines whether to show to map. Obviously TRUE by default, but show=FALSE can be useful for just obtaining the returned objects.
knit	should knit_print be enabled, or the normal print function?
options	options passed on to knitprint
...	not used

Value

If mode=="plot", then a list is returned with the processed shapes and the metadata. If mode=="view", a leaflet object is returned (see also tmap_leaflet)

---

qtm *Quick thematic map plot*

Description

Draw a thematic map quickly. This function is a convenient wrapper of the main plotting method of stacking tmap-elements. Without arguments or with a search term, this functions draws an interactive map.

Usage

```
qtm(shp, fill = NA, symbols.size = NULL, symbols.col = NULL,
  symbols.shape = NULL, dots.col = NULL, text = NULL,
  text.size = 1, text.col = NA, lines.lwd = NULL, lines.col = NULL,
  raster = NA, borders = NA, by = NULL, scale = NA, title = NA,
  projection = NULL, bbox = NULL, basemaps = NA, overlays = NA,
  style = NULL, format = NULL, ...)
```

Arguments

shp	One of • shape object, which is an object from a class defined by the sf, sp, or raster package. For instance, an sf object, an SpatialPolygons(DataFrame), or a RasterBrick.

- Not specified, i.e. `qtm()` is executed. In this case a plain interactive map is shown.
- A OSM search string, e.g. `qtm("Amsterdam")`. In this case a plain interactive map is shown positioned according to the results of the search query (from OpenStreetMap nominatim)

`fill` either a color to fill the polygons, or name of the data variable in `shp` to draw a choropleth. Only applicable when `shp` contains polygons. Set `fill = NULL` to draw only polygon borders. See also argument `borders`.

`symbols.size` either the size of the symbols or a name of the data variable in `shp` that specifies the sizes of the symbols. See also the `size` argument of `tm_symbols`. Only applicable when `shp` contains spatial points, lines, or polygons.

`symbols.col` either the color of the symbols or a name of the data variable in `shp` that specifies the colors of the symbols. See also the `col` arugment of `tm_symbols`. Only applicable when `shp` contains spatial points, lines, or polygons.

`symbols.shape` either the shape of the symbols or a name of the data variable in `shp` that specifies the shapes of the symbols. See also the `shape` arugment of `tm_symbols`. Only applicable when `shp` contains spatial points, lines, or polygons.

`dots.col` name of the data variable in `shp` for the dot map that specifies the colors of the dots. If `dots.col` is specified instead `symbols.col`, dots instead of bubbles are drawn (unless `symbols.shape` is specified).

`text` Name of the data variable that contains the text labels. Only applicable when `shp` contains spatial points, lines, or polygons.

`text.size` Font size of the text labels. Either a constant value, or the name of a numeric data variable. Only applicable when `shp` contains spatial points, lines, or polygons.

`text.col` name of the data variable in `shp` for the that specifies the colors of the text labels. Only applicable when `shp` contains spatial points, lines, or polygons.

`lines.lwd` either a line width or a name of the data variable that specifies the line width. Only applicable when `shp` contains spatial lines.

`lines.col` either a line color or a name of the data variable that specifies the line colors. Only applicable when `shp` contains spatial lines.

`raster` either a color or a name of the data variable that specifices the raster colors. Only applicable when `shp` is a spatial raster.

`borders` color of the polygon borders. Use `NULL` to omit the borders.

`by` data variable name by which the data is split, or a vector of two variable names to split the data by two variables (where the first is used for the rows and the second for the columns). See also `tm_facets`

`scale` numeric value that serves as the global scale parameter. All font sizes, symbol sizes, border widths, and line widths are controlled by this value. The parameters `symbols.size`, `text.size`, and `lines.lwd` can be scaled seperately with respectively `symbols.scale`, `text.scale`, and `lines.scale`. See also `...`.

`title` main title. For legend titles, use `X.style`, where X is the layer name (see `...`).

`projection` Either a `crs` object or a character value. If it is a character, it can either be a `PROJ.4` character string or a shortcut. See `get_proj4` for a list of shortcut values. By default, the projection is used that is defined in the `shp` object itself, which can be obtained with `get_projection`.

bbox	bounding box. Arugment passed on to tm_shape
basemaps	name(s) of the provider or an URL of a tiled basemap. It is a shortcut to tm_basemap. Set to NULL to disable basemaps. By default, it is set to the tmap option basemaps.
overlays	name(s) of the provider or an URL of a tiled overlay map. It is a shortcut to tm_tiles.
style	Layout options (see tm_layout) that define the style. See tmap_style for details.
format	Layout options (see tm_layout) that define the format. See tmap_format for details.
...	arguments passed on to the tm_* functions. The prefix of these arguments should be with the layer function name without "tm_" and a period. For instance, the palette for polygon fill color is called fill.palette. The following prefixes are supported: shape., fill., borders., polygons., symbols., dots., lines., raster., text., layout., grid., facets., and view.. Arguments that have a unique name, i.e. that does not exist in any other layer function, e.g. convert2density, can also be called without prefix.

Details

The first argument is a shape object (normally specified by tm_shape). The next arguments, from fill to raster, are the aesthetics from the main layers. The remaining arguments are related to the map layout. Any argument from any main layer function, such as tm_polygons, can be specified (see ...). It is also possible to stack tmap-elements on a qtm plot. See examples.

By default, a scale bar is shown. This option can be set with tmap_options (argument qtm.scalebar). A minimap is shown by default when qtm is called without arguments of with a search term. This option can be set with tmap_options (argument qtm.minimap).

Value

tmap-element

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

vignette("tmap-getstarted")

Examples

```
data(World, rivers, metro)

# just the map
qtm(World)

# choropleth
```

```
qtm(World, fill = "economy", format = "World", style = "col_blind")

# choropleth with more specifications
qtm(World, fill="HPI", fill.n = 9, fill.palette = "div",
    fill.title = "Happy Planet Index", fill.id = "name",
    style = "gray", format = "World")
# this map can also be created with the main plotting method,
# which is recommended in this case.
## Not run:
tm_shape(World) +
    tm_polygons("HPI", n = 9, palette = "div",
        title = "Happy Planet Index", id = "name") +
tm_style("gray") +
tm_format("World")

## End(Not run)

# bubble map
## Not run:
qtm(World, borders = NULL) +
qtm(metro, symbols.size = "pop2010",
    symbols.title.size= "Metropolitan Areas",
    symbols.id= "name",
    format = "World")

## End(Not run)

# dot map
## Not run:
current.mode <- tmap_mode("view")
qtm(metro, bbox = "China")
tmap_mode(current.mode) # restore mode

## End(Not run)

## Not run:
# without arguments, a plain interactive map is shown (the mode is set to view)
qtm()

# search query for OpenStreetMap nominatim
qtm("Amsterdam")

## End(Not run)
```

`rivers` *Spatial data of rivers*

Description

Spatial data of rivers, of class `sf`

Usage

```
data(rivers)
```

Source

http://www.naturalearthdata.com

---

theme_ps *ggplot2 theme for proportional symbols*

Description

ggplot2 theme for proportional symbols. By default, this theme only shows the plotting area, so without titles, axes, and legend

Usage

```
theme_ps(base_size = 12, base_family = "", plot.axes = FALSE,
  plot.legend = FALSE)
```

Arguments

base_size	base size
base_family	base family
plot.axes	should the axes be shown?
plot.legend	should the legend(s) be shown?

---

tmap-element *tmap element*

Description

Building block for drawing thematic maps. All element functions have the prefix tm_.

Details

The fundamental, and hence required element is tm_shape, which specifies the shape object, and also specifies the projection and bounding box.

The elements that serve as aesthetics layers are

Base layers:

tm_polygons	Create a polygon layer (with borders)
tm_symbols	Create a layer of symbols
tm_lines	Create a layer of lines

tm_raster	Create a raster layer
tm_text	Create a layer of text labels
tm_basemap	Create a layer of basemap tiles
tm_tiles	Create a layer of overlay tiles

Derived layers:

tm_fill	Create a polygon layer (without borders)
tm_borders	Create polygon borders
tm_bubbles	Create a layer of bubbles
tm_squares	Create a layer of squares
tm_dots	Create a layer of dots
tm_markers	Create a layer of markers
tm_iso	Create a layer of iso/contour lines
tm_rgb	Create a raster layer of an image

The layers can be stacked by simply adding them with the + symbol. The combination of the elements described above form one group. Multiple groups can be stacked. Each group should start with tm_shape.

Attributes layers:

tm_grid	Create grid lines
tm_scale_bar	Create a scale bar
tm_compass	Create a map compass
tm_credits	Create a text for credits
tm_logo	Create a logo
tm_xlab and tm_ylab	Create axis labels
tm_minimap	Create a minimap (view mode only)

Layout element:

tm_layout	Adjust the layout (main function)
tm_legend	Adjust the legend
tm_view	Configure the interactive view mode
tm_style	Apply a predefined style
tm_format	Apply a predefined format

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

vignette("tmap-getstarted")

The examples in each of the element functions

`tmap_animation` *Create animation*

Description

Create a gif or mpeg animation from a tmap plot. The free tool ImageMagick is required.

Usage

```
tmap_animation(tm, filename = "animation.gif", width = NA,
  height = NA, delay = 40, loop = TRUE, restart.delay = 0)
```

Arguments

`tm`	tmap object. In order to create a series of tmap plots, which will be the frames of the animation, it is important to set nrow and ncol in `tm_facets`, for otherwise a small multiples plot is generated. Commonly, where one map is shown at a time, both nrow and ncol are set to 1.
`filename`	filename of the video (should be a .gif or .mpg file)
`width`	width of the animation file (in pixels)
`height`	height of the animation file (in pixels)
`delay`	delay time between images (in 1/100th of a second)
`loop`	logical that determined whether the animation is looped, or an integer value that determines how many times the animation is looped.
`restart.delay`	delay time between the loops (in 1/100th of a second)

Note

Not only tmap plots are supported, but any series of R plots.

Examples

```
## Not run:
data(NLD_prov)

m1 <- tm_shape(NLD_prov) +
      tm_polygons("yellow") +
  tm_facets(along = "name")

tmap_animation(m1, filename="Dutch_provinces.gif", width=800, delay=40)

data(World, metro)
```

```
m2 <- tm_shape(World, simplify = 0.5) +
          tm_fill() +
      tm_shape(metro) +
          tm_bubbles(size = paste0("pop", seq(1970, 2030, by=10)),
              col = "purple",
              border.col = "black", border.alpha = .5,
              scale = 2) +
      tm_facets(free.scales.symbol.size = FALSE, nrow=1,ncol=1) +
      tm_format("World", scale=.5)

tmap_animation(m2, filename="World population.gif", width=1200, delay=100)

## End(Not run)
```

---

tmap_arrange *Arrange small multiples in grid layout*

Description

Arrange small multiples in a grid layout. Normally, small multiples are created by specifying multiple variables for one aesthetic or by specifying the by argument (see tm_facets). This function can be used to arrange custom small multiples in a grid layout.

Usage

```
tmap_arrange(..., ncol = NA, nrow = NA, sync = FALSE, asp = 0,
  outer.margins = 0.02)

knit_print.tmap_arrange(x, ..., options = NULL)

## S3 method for class 'tmap_arrange'
print(x, knit = FALSE, ..., options = NULL)
```

Arguments

...	tmap objects or one list of tmap objects. The number of multiples that can be plot is limited (see details).
ncol	number of columns
nrow	number of rows
sync	logical. Should the navigation in view mode (zooming and panning) be synchronized? By default FALSE.
asp	aspect ratio. The aspect ratio of each map. Normally, this is controlled by the asp argument from tm_layout (also a tmap option). This argument will overwrite it, unless set to NULL. The default value for asp is 0, which means that the aspect ratio is adjusted to the size of the device divided by the number of columns and rows. When asp is set to NA, which is also the default value for tm_layout, the aspect ratio will be adjusted to the used shapes.

outer.margins	outer.margins, numeric vector four or a single value. If defines the outer margins for each multiple. If will overwrite the outer.margins argument from tm_layout, unless set to NULL.
x	a tmap_arrange object (returned from tmap_arrange)
options	options passed on to knitprint
knit	should knit_print be enabled, or the normal print function?

Details

The global option tmap.limits controls the limit of the number of facets that are plotted. By default, tmap_options(tmap.limits=c(facets.view=4, facets.plot=64)). The maximum number of interactive facets is set to four since otherwise it may become very slow.

Examples

```
data(World)
w1 <- qtm(World, projection = "eck4", title="Eckert IV")
w2 <- qtm(World, projection = "merc", title="Mercator")
w3 <- qtm(World, projection = "gall", title="Gall stereographic")
w4 <- qtm(World, projection = "robin", title="Robinsin")

current.mode <- tmap_mode("plot")
tmap_arrange(w1, w2, w3, w4)
tmap_mode(current.mode)
```

---

tmap_format *Get or add format options*

Description

Format options are tmap options that are shape dependent. With tmap_format() the predefined formats can be retrieved. The values for a specific format can be retrieved with tmap_format(format), where format is the name of the format. The function tmap_format_add is used to add a format.

Usage

```
tmap_format(format)

tmap_format_add(..., name)
```

Arguments

format	name of the format. Run tmap_format() to see the choices.
...	options from tm_layout or tm_view. Can also be a list of those options.
name	name of the new format.

Value

the function `tmap_format()` returns the names of the available formats. When `format` is defined, it returns the option list corresponding the that format.

See Also

`tm_layout` for predefined styles, `tmap_style_catalogue` to create a style catalogue of all available styles, and `tmap_options` for tmap options.

`tmap_options` for tmap options

Examples

```
# available formats
tmap_format()

# create option list to be used as a new format
World_small <- tmap_format("World")
World_small$scale <- 2

# add format
tmap_format_add(World_small, name = "World_small")

# observe that World_small is successfully added:
tmap_format()

data(World)

qtm(World, fill="HPI", format="World_small")
```

---

tmap_icons *Specify icons*

Description

Specifies icons from a png images, which can be used as markers in thematic maps. The function `marker_icon` is the specification of the default marker.

Usage

```
tmap_icons(file, width = 48, height = 48, keep.asp = TRUE,
  just = c("center", "center"), as.local = TRUE, ...)

marker_icon()
```

Arguments

file	character value/vector containing the file path(s) or url(s).
width	width of the icon. If keep.asp, this is interpreted as the maximum width.
height	height of the icon. If keep.asp, this is interpreted as the maximum height.
keep.asp	keep the aspect ratio of the png image. If TRUE and the aspect ratio differs from width/height either width or height is adjusted accordingly.
just	justification of the icons relative to the point coordinates. The first value specifies horizontal and the second value vertical justification. Possible values are: "left" , "right", "center", "bottom", and "top". Numeric values of 0 specify left alignment and 1 right alignment. The default value of just is c("center", "center").
as.local	if the file is a url, should it be saved to local temporary file?
...	arguments passed on to icons. When iconWidth, iconHeight, iconAnchorX and iconAnchorY are specified, they override width and height, and just.

Value

icon data (see icons)

See Also

tm_symbols

---

tmap_last *Retrieve the last map to be modified or created*

Description

Retrieve the last map to be modified or created. Works in the same way as ggplot2's last_plot, although there is a difference: last_map returns the last call instead of the stacked tmap-elements.

Usage

```
tmap_last()
```

Value

call

See Also

tmap_save

tmap_leaflet *Create a leaflet widget from a tmap object*

Description

Create a leaflet widget from a tmap object. An interactive map (see tmap_mode) is an automatically generated leaflet widget. With this function, this leaflet widget is obtained, which can then be changed or extended by using leaflet's own methods.

Usage

```
tmap_leaflet(x, mode = "view", show = FALSE, add.titles = TRUE, ...)
```

Arguments

x	tmap object. A tmap object is created with qtm or by stacking tmap-elements.
mode	the mode of tmap, which is set to "view" in order to obtain the leaflet object. See tmap_mode for details.
show	should the leaflet map be shown? FALSE by default
add.titles	add titles to leaflet object
...	arguments passed on to print.tmap

Value

leaflet object

See Also

tmap_mode, tm_view, print.tmap

Examples

```
# world choropleth/bubble map of the world
data(World, metro)
metro$growth <- (metro$pop2020 - metro$pop2010) / (metro$pop2010 * 10) * 100

map1 <- tm_shape(metro) +
tm_bubbles("pop2010", col = "growth",
   border.col = "black", border.alpha = .5,
   style="fixed", breaks=c(-Inf, seq(0, 6, by=2), Inf),
   palette="-RdYlBu", contrast=1,
   title.size="Metro population",
   title.col="Growth rate (%)", id="name") +
tm_layout(legend.bg.color = "grey90", legend.bg.alpha=.5, legend.frame=TRUE)


lf <- tmap_leaflet(map1)
```

```
# show leaflet widget
lf

# add marker
require(leaflet)
lf %>% leaflet::addMarkers(2.2945, 48.8582, popup = "Eiffel tower")

## Not run:
# alternative
eiffelTower <- geocode_OSM("Eiffel Tower, Paris", as.SPDF = TRUE)

map1 +
tm_shape(eiffelTower) +
tm_markers()

## End(Not run)
```

---

tmap_mode *Set tmap mode to static plotting or interactive viewing*

Description

Set tmap mode to static plotting or interactive viewing. The global option `tmap.mode` determines the whether thematic maps are plot in the graphics device, or shown as an interactive leaflet map (see also `tmap_options`. The function `tmap_mode` is a wrapper to set this global option. The convenient function `ttm` is a toggle switch between the two modes. Tip 1: use `tmap_mode` in scripts and `ttm` in the console. Tip 2: use `ttm` in combination with `tmap_last` to redraw the last map in the other mode.

Usage

```
tmap_mode(mode = c("plot", "view"))

ttm()
```

Arguments

mode	one of
	`"plot"` Thematic maps are shown in the graphics device. This is the default mode, and supports all tmap's features, such as small multiples (see `tm_facets`) and extensive layout settings (see `tm_layout`). It is recommended for saving static maps (see `tmap_save`).
	`"view"` Thematic maps are viewed interactively in the web browser or RStudio's Viewer pane. Maps are fully interactive with tiles from OpenStreetMap or other map providers (see `tm_tiles`). See also `tm_view` for options related to the `"view"` mode. This mode generates a `leaflet` widget, which can also be directly obtained with `tmap_leaflet`. With RMarkdown, it is possible to publish it to an HTML page. There are a couple of constraints in comparison to `"plot"`:

- The map is always projected according to the Web Mercator projection. Although this projection is the de facto standard for interactive web-based mapping, it lacks the equal-area property, which is important for many thematic maps, especially choropleths (see examples from tm_shape).
- Small multiples are not supported
- The legend cannot be made for aesthetics regarding size, which are symbol size and line width.
- Text labels are not supported (yet)
- The layout options set with tm_layout) regarding map format are not used. However, the styling options still apply.

Value

the mode before changing

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

vignette("tmap-getstarted"), tmap_last to show the last map, tm_view for viewing options, and tmap_leaflet for obtaining a leaflet widget, and tmap_options for tmap options.

Examples

```
# world choropleth/bubble map of the world
data(World, metro)
metro$growth <- (metro$pop2020 - metro$pop2010) / (metro$pop2010 * 10) * 100

map1 <- tm_shape(World) +
tm_polygons("income_grp", palette="-Blues", contrast=.7, id="name", title="Income group") +
tm_shape(metro) +
tm_bubbles("pop2010", col = "growth",
border.col = "black", border.alpha = .5,
style="fixed", breaks=c(-Inf, seq(0, 6, by=2), Inf),
palette="-RdYlBu", contrast=1,
title.size="Metro population",
title.col="Growth rate (%)", id="name",
popup.vars = c("pop2010", "pop2020", "growth")) +
tm_layout(legend.bg.color = "grey90", legend.bg.alpha=.5, legend.frame=TRUE)

# initial mode: "plot"
current.mode <- tmap_mode("plot")

# plot map
map1

# switch to other mode: "view"
ttm()
```

```
# view map
map1

## Not run:
# choropleth of the Dutch population in interactive mode:
require(tmaptools)
data(NLD_muni, NLD_prov)
NLD_muni$pop_dens <- calc_densities(NLD_muni, var = "population")

tm_shape(NLD_muni) +
tm_fill(col="pop_dens",
style="kmeans",
title = "Population (per km^2)", id = "name") +
tm_borders("grey25", alpha=.5) +
tm_shape(NLD_prov) +
tm_borders("grey40", lwd=2)

## End(Not run)

# restore current mode
tmap_mode(current.mode)
```

`tmap_options` *Options for tmap*

Description

Get or set global options for tmap. The behaviour of `tmap_options` is similar to `options`: all tmap options are retrieved when this function is called without arguments. When arguments are specified, the corresponding options are set, and the old values are silently returned as a list. The function `tmap_options_reset` is used to reset all options back to the default values (also the `style` is reset to `"white"`). Differences with the default values can be shown with `tmap_options_diff`. The function `tmap_options_save` can be used to save the current options as a new style. See details below on how to create a new style.

Usage

```
tmap_options(..., unit, limits, max.categories, max.raster, basemaps,
  basemaps.alpha, overlays, overlays.alpha, qtm.scalebar, qtm.minimap,
  show.messages)

tmap_options_diff()

tmap_options_reset()

tmap_options_save(style)
```

Arguments

...	options from tm_layout or tm_view. Note that the difference with using tm_layout or tm_view directly, is that options set with tmap_options remain for the entire session (unless changed with tmap_options or tmap_style). It can also be a single unnamed argument which is a named list of options (similar behaviour as options).
unit	this is the default value for the unit argument of tm_shape. It specifies the unit of measurement, which is used in the scale bar and the calculation of density values. By default (when loading the package), it is "metric". Other valid values are "imperial", "km", "m", "mi", and "ft".
limits	this option determines how many facets (small multiples) are allowed for per mode. It should be a vector of two numeric values named facets.view and facets.plot. By default (i.e. when loading the package), it is set to c(facets.view = 4, facets.plot
max.categories	in case col is the name of a categorical variable in the layer functions (e.g. tm_polygons), this value determines how many categories (levels) it can have maximally. If the number of levels is higher than max.categories, then levels are combined.
max.raster	the maximum size of rasters, in terms of number of raster cells. It should be a vector of two numeric values named plot and view, which determines the size in plotting and viewing mode. The default values are c(plot = 1e7, view = 1e6). Rasters that are larger will be shown at a decreased resolution.
basemaps	default basemaps. Basemaps are normally configured with tm_basemap. When this is not done, the basemaps specified by this option are shown (in view mode). Vector of one or more names of baselayer maps, or NULL if basemaps should be omitted. For options see the list leaflet::providers, which can be previewed at http://leaflet-extras.github.io/leaflet-providers/preview. Also supports URL's for tile servers, such as "http://{s}.tile.openstreetmap.org/{z}/{ If a named vector is provided, the names are used in the layer control legend (similar to the group argument of tm_basemap. See also overlays, which is the default option for overlay tiles.
basemaps.alpha	default transparency (opacity) value for the basemaps. Can be a vector of values, one for each basemap.
overlays	default overlay tilemaps. Overlays tilemaps are shown as front layer (in contrast to basemaps, which are background layers), so they are only useful when they are semi-transparent. Like basemaps, a vector of tilemaps is expected, or NULL is overlays should be omitted.
overlays.alpha	default transparency (opacity) value for the overlay maps. Can be a vector of values, one for each overlay map.
qtm.scalebar	should a scale bar be added to interactive maps created with qtm. In other words, should tm_scale_bar() be added automatically? The value NA means that the scale bar is only added when qtm is called without arguments or with a search term. The default value is TRUE.
qtm.minimap	should a minimap be added to interactive maps created with qtm. In other words, should tm_minimap() be added automatically? The value NA means that the minimap is only added in navigation mode (i.e. when qtm is called without arguments or with a search term. The default value is FALSE.

show.messages	should messages be shown?
style	style name

Details

The options can be divided into three parts: one part contains the arguments from tm_layout, one part contains the arguments from tm_view, and one part contains options that can only be set with tmap_options. Observe that the options from tm_layout and tm_view can also be set with those functions. It is recommended to use tmap_options when setting specific options during global session. However, options that are only relevant for a specific map can better be set with tm_layout or tm_view.

A new style can be created in two ways. The first approach is to use the function tmap_options_save, which takes a snapshot of the current tmap options. E.g., tmap_options_save("my_style") will save the current tmap options as a style called "my_style". See the examples in which a style called "red" is created. The second way to create a style is to create a list with tmap options and with a attribute called style. This approach is illustrated in the last example, in which a style called "black" is created.

The newly created style, say "my_style", will be accessible globally via tmap_style("my_style") and + tm_style("my_style") until the R session is restarted or tmap is reloaded. In order to save the style for future use or sharing, obtain the option list as follows: my_style <- tmap_options() and save the object my_style in the usual way. Next time, the style can be loaded simply by running tmap_options(my_style), which corresponds to the second way to create a style (see the paragraph above).

See Also

tm_layout, tm_view, and tmap_style

Examples

```
# load data
data(World)

# get current options
str(tmap_options())

# get current style
tmap_style()

# plot map (with default options)
tm_shape(World) + tm_polygons("HPI")

# change style to cobalt
tmap_style("cobalt")

# observe the changed options
tmap_options_diff()

# plot the map again
tm_shape(World) + tm_polygons("HPI")
```

```
##############################
# define red style
##############################

# change the background color
tmap_options(bg.color = "red")

# note that the current style is modified
tmap_style()

# observe the changed options
tmap_options_diff()

# save the current options as style "red"
tmap_options_save("red")

# plot the map again
tm_shape(World) + tm_polygons("HPI")

# the specified arguments of tm_layout and tm_view will override the options temporarily:
tm_shape(World) + tm_polygons("HPI") + tm_layout(bg.color="purple")

# when tm_style_ is called, it will override all options temporarily:
tm_shape(World) + tm_polygons("HPI") + tm_layout(bg.color="purple") + tm_style("classic")

# reset all options
tmap_options_reset()

# check style and options
tmap_style()
tmap_options_diff()

##############################
# define black style
##############################

# create style list with style attribute
black_style  <- structure(
    list(
bg.color = "black",
aes.color = c(fill = "grey40", borders = "grey40",
  symbols = "grey80", dots = "grey80",
              lines = "white", text = "white",
  na = "grey30", null = "grey15"),
aes.palette = list(seq = "plasma", div = "PiYG", cat = "Dark2"),
attr.color = "white",
panel.label.color = "white",
panel.label.bg.color = "grey40",
main.title.color = "white"
),
style = "black"
)
```

```
# assign the style
tmap_options(black_style)

# observe that "black" is a new style
tmap_style()

# plot the world map again, this time with the newly created black style
tm_shape(World) +
tm_polygons("HPI")

# reset all options
tmap_options_reset()
```

`tmap_save` *Save tmap*

Description

Save tmap to a file, such as png, jpg, or pdf.

Usage

```
tmap_save(tm = NULL, filename = NULL, width = NA, height = NA,
  units = NA, dpi = 300, outer.margins = NA, asp = NULL,
  scale = NA, insets_tm = NULL, insets_vp = NULL,
  add.titles = TRUE, verbose = NULL, ...)
```

Arguments

`tm`	tmap object
`filename`	filename including extension, and optionally the path. The extensions pdf, eps, svg, wmf (Windows only), png, jpg, bmp, or tiff are supported. If the extension is missing, the file will be saved as png image
`width`	width. Units are set with the argument `units`. If set to `NA` and `height` is specified, it will be `height` * aspect ratio. If both `width` and `height` are not specified, then the width of the current plotting window will be taken.
`height`	height. Units are set with the argument `units`. If set to `NA` and `width` is specified, it will be `width` / aspect ratio. If both `width` and `height` are not specified, then the height of the current plotting window will be taken.
`units`	units for width and height (`"in"`, `"cm"`, or `"mm"`). By default, pixels (`"px"`) are used if either width or height is set to a value greater than 50. Else, the units are inches (`"in"`)
`dpi`	dots per inch. Only applicable for raster graphics.
`outer.margins`	overrides the outer.margins argument of `tm_layout` (unless set to `NA`)

asp	if specified, it overrides the asp argument of tm_layout. Tip: set to 0 if map frame should be placed on the edges of the image.
scale	overrides the scale argument of tm_layout (unless set to NA)
insets_tm	tmap object of an inset map, or a list of tmap objects of multiple inset maps. The number of tmap objects should be equal to the number of viewports specified with insets_vp.
insets_vp	viewport of an inset map, or a list of viewports of multiple inset maps. The number of viewports should be equal to the number of tmap objects specified with insets_tm.
add.titles	add titles to leaflet object
verbose	Deprecated. It is now controlled by the tmap option show.messages (see tmap_options)
...	arguments passed on to device functions or to saveWidget

Examples

```
## Not run:
data(NLD_muni, NLD_prov)
m <- tm_shape(NLD_muni) +
     tm_fill(col="population", convert2density=TRUE,
                 style="kmeans",
                 title=expression("Population (per " * km^2 * ")")) +
     tm_borders("black", alpha=.5) +
 tm_shape(NLD_prov) +
     tm_borders("grey25", lwd=2) +
tm_style("classic") +
tm_format("NLD", inner.margins = c(.02, .15, .06, .15)) +
    tm_scale_bar(position = c("left", "bottom")) +
    tm_compass(position=c("right", "bottom"))

tmap_save(m, "choropleth.png", height=7)

data(World)
m2 <- tm_shape(World) +
tm_fill("well_being", id="name", title="Well-being") +
tm_format("World")

# save image
tmap_save(m2, "World_map.png", width=1920, height=1080, asp=0)

# cut left inner margin to make sure Antarctica is snapped to frame
tmap_save(m2 + tm_layout(inner.margins = c(0, -.1, 0.05, 0.01)),
        "World_map2.png", width=1920, height=1080, asp=0)

# save interactive plot
tmap_save(m2, "World_map.html")

## End(Not run)
```

`tmap_style` *Set or get the default tmap style*

Description

Set or get the default tmap style. Without arguments, the current style is returned. Also the available styles are displayed. When a style is set, the corresponding tmap options (see `tmap_options`) will be set accordingly. The default style (i.e. when loading the package) is `"white"`.

Usage

```
tmap_style(style)
```

Arguments

`style`	name of the style. When omitted, `tmap_style` returns the current style and also shows all available styles. When the style is specified, `tmap_style` sets the style accordingly. Note that in that case, all tmap options (see `tmap_options`) will be reset according to the style definition. See `tm_layout` for predefined styles, and `tmap_style_catalogue` for creating a catalogue.

Details

Note that `tm_style` is used within a plot call (so it only affects that plot), whereas `tmap_style` sets the style globally.

After loading a style, the options that defined this style (i.e. the difference with the default `"white"` style) can be obtained by `tmap_options_diff`.

The documentation of `tmap_options` (details and the examples) shows how a new style is created.

Value

the style before changing

See Also

`tmap_options` for tmap options, and `tmap_style_catalogue` to create a style catalogue of all available styles.

Examples

```
data(World)

current.style <- tmap_style("classic")
qtm(World, fill="life_exp", fill.title="Life expectancy")

tmap_style("cobalt")
qtm(World, fill="life_exp", fill.title="Life expectancy")
```

```
# restore current style
tmap_style(current.style)
```

`tmap_style_catalogue` *Create a style catalogue*

Description

Create a style catalogue for each predefined tmap style. The result is a set of png images, one for each style.

Usage

```
tmap_style_catalogue(path = "./tmap_style_previews", styles = NA)

tmap_style_catalog(path = "./tmap_style_previews", styles = NA)
```

Arguments

`path`	path where the png images are stored
`styles`	vector of styles function names (see `tmap_style`) for which a preview is generated. By default, a preview is generated for all loaded styles.

`tmap_tip` *Get a tip about tmap*

Description

Generates a tip with an example. The tip and example code are printed, and the example itself is executed.

Usage

```
tmap_tip(latest.version = FALSE)
```

Arguments

`latest.version`	should only tips be generated from the latest version of tmap? False by default.

Examples

```
tmap_tip()
```

tm_add_legend *Add manual legend*

Description

Creates a tmap-element that adds a manual legend.

Usage

```
tm_add_legend(type = c("fill", "symbol", "text", "line"),
  labels = NULL, col = NULL, size = NULL, shape = NULL,
  lwd = NULL, lty = NULL, text = NULL, alpha = NA,
  border.col = "black", border.lwd = 1, border.alpha = NA,
  title = "", is.portrait = TRUE, legend.format = list(),
  reverse = FALSE, z = NA)
```

Arguments

type	type of legend. One of "fill", "symbol", "text", or "line"
labels	legend labels
col	legend colors
size	legend symbol sizes (if type=="symbol")
shape	legend symbol shapes (if type=="symbol")
lwd	legend line widths (if type=="line")
lty	legend line types (if type=="line")
text	legend texts (if type=="text")
alpha	legend fill transparency
border.col	legend border col (if type is "fill" or "symbol")
border.lwd	legend border width (if type is "fill" or "symbol")
border.alpha	legend border alpha (if type is "fill" or "symbol")
title	legend title
is.portrait	is legend portrait (TRUE) or landscape (FALSE)?
legend.format	options to format the legend, see tm_symbols (the description of the argument legend.format) for details. Note that many of these arguments are not applicable for tm_add_legend since labels should be a character vector. However, some options could still be handy, e.g. list(text.align = "right").
reverse	are the legend items reversed (by default FALSE)?
z	legend stack position

See Also

tm_symbols for an example

tm_basemap *Draw a tile layer*

Description

Creates a tmap-element that draws a tile layer. This feature is only available in view mode. For plot mode, a tile image can be retrieved by read_osm. The function tm_basemap draws the tile layer as basemap (i.e. as bottom layer), whereas tm_tiles draws the tile layer as overlay layer (where the stacking order corresponds to the order in which this layer is called). Note that basemaps are shown by default (see details).

Usage

```
tm_basemap(server = NA, group = NA, alpha = NA)

tm_tiles(server, group = NA, alpha = 1)
```

Arguments

server	name of the provider or an URL. The list of available providers can be obtained with leaflet::providers. See http://leaflet-extras.github.io/leaflet-providers/preview for a preview of those. When a URL is provided, it should be in template format, e.g. "http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png". Use NULL in tm_basemap to disable the basemaps.
group	name of the group to which this layer belongs in view mode. Each group can be selected or deselected in the layer control item. Groups can either be specified as base or overlay groups in tm_view (arguments base.groups and overlay.groups). Tile layers generated with tm_basemap will be base groups whereas tile layers generated with tm_tiles will be overlay groups.
alpha	alpha

Details

When tm_basemap is not specified, the default basemaps are shown, which can be configured by the basemaps arugument in tmap_options. By default (for style "white") three basemaps are drawn: c("Esri.WorldGrayCanvas", "OpenStreetMap", "Esri.WorldTopoMap"). To disable basemaps, add tm_basemap(NULL) to the plot, or set tmap_options(basemaps = NULL). Similarly, when tm_tiles is not specified, the overlay maps specified by the overlays argument in in tmap_options are shown as front layer. By default, this argument is set to NULL, so no overlay maps are shown by default. See examples.

Examples

```
## Not run:
current.mode <- tmap_mode("view")

data(World, metro)
```

```
tm_basemap(leaflet::providers$Stamen.Watercolor) +
tm_shape(metro, bbox = "India") + tm_dots(col = "red", group = "Metropolitan areas") +
tm_tiles(paste0("http://services.arcgisonline.com/arcgis/rest/services/Canvas/",
    "World_Light_Gray_Reference/MapServer/tile/{z}/{y}/{x}"), group = "Labels")


# Use tmap optinos to set the basemap and overlay map permanently during the R session:
opts <- tmap_options(basemaps = c(Canvas = "Esri.WorldGrayCanvas", Imagery = "Esri.WorldImagery"),
   overlays = c(Labels = paste0("http://services.arcgisonline.com/arcgis/rest/services/Canvas/",
                               "World_Light_Gray_Reference/MapServer/tile/{z}/{y}/{x}")))

qtm(World, fill = "HPI", fill.palette = "RdYlGn")

# restore options
tmap_options(opts)

# restore current mode
tmap_mode(current.mode)

## End(Not run)
```

`tm_compass` *Map compass*

Description

Creates a map compass.

Usage

```
tm_compass(north = 0, type = NA, fontsize = 0.8, size = NA,
  show.labels = 1, cardinal.directions = c("N", "E", "S", "W"),
  text.color = NA, color.dark = NA, color.light = NA, lwd = 1,
  position = NA, just = NA)
```

Arguments

`north`	north direction in degrees: 0 means up, 90 right, etc.
`type`	compass type, one of: `"arrow"`, `"4star"`, `"8star"`, `"radar"`, `"rose"`. The default is controlled by `tm_layout` (which uses `"arrow"` for the default style)
`fontsize`	relative font size
`size`	size of the compass in number of text lines. The default values depend on the `type`: for `"arrow"` it is 2, for `"4star"` and `"8star"` it is 4, and for `"radar"` and `"rose"` it is 6.
`show.labels`	number that specifies which labels are shown: 0 means no labels, 1 (default) means only north, 2 means all four cardinal directions, and 3 means the four cardinal directions and the four intercardinal directions (e.g. north-east).

`cardinal.directions`
: labels that are used for the cardinal directions north, east, south, and west.

`text.color`
: color of the text. By default equal to the argument `attr.color` of `tm_layout`.

`color.dark`
: color of the dark parts of the compass, typically (and by default) black.

`color.light`
: color of the light parts of the compass, typically (and by default) white.

`lwd`
: line width of the compass

`position`
: position of the compass. Vector of two values, specifying the x and y coordinates. Either this vector contains "left", "LEFT", "center", "right", or "RIGHT" for the first value and "top", "TOP", "center", "bottom", or "BOTTOM" for the second value, or this vector contains two numeric values between 0 and 1 that specifies the x and y value of the left bottom corner of the compass. The uppercase values correspond to the position without margins (so tighter to the frame). The default value is controlled by the argument `"attr.position"` of `tm_layout`.

`just`
: Justification of the attribute relative to the point coordinates. The first value specifies horizontal and the second value vertical justification. Possible values are: `"left"` , `"right"`, `"center"`, `"bottom"`, and `"top"`. Numeric values of 0 specify left/bottom alignment and 1 right/top alignment. This option is only used, if `position` is specified by numeric coordinates. The default value is controlled by the argument `"attr.just"` of `tm_layout`.

Examples

```
current.mode <- tmap_mode("plot")

data(NLD_muni)

qtm(NLD_muni, theme = "NLD") + tm_compass()
qtm(NLD_muni, theme = "NLD") + tm_compass(type="radar", position=c("left", "top"), show.labels = 3)

# restore current mode
tmap_mode(current.mode)
```

---

tm_credits *Credits text*

Description

Creates a text annotation that could be used for credits or acknowledgements.

Usage

```
tm_credits(text, size = 0.7, col = NA, alpha = NA, align = "left",
  bg.color = NA, bg.alpha = NA, fontface = NA, fontfamily = NA,
  position = NA, just = NA)
```

Arguments

text	text. Multiple lines can be created with the line break symbol "\n". Facets can have different texts: in that case a vector of characters is required. Use "" to omit the credits for specific facets.
size	relative text size
col	color of the text. By default equal to the argument attr.color of tm_layout.
alpha	transparency number between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of col is used (normally 1).
align	horizontal alignment: "left" (default), "center", or "right". Only applicable if text contains multiple lines
bg.color	background color for the text
bg.alpha	Transparency number between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the bg.color is used (normally 1).
fontface	font face of the text. By default, determined by the fontface argument of tm_layout.
fontfamily	font family of the text. By default, determined by the fontfamily argument of tm_layout.
position	position of the text. Vector of two values, specifying the x and y coordinates. Either this vector contains "left", "LEFT", "center", "right", or "RIGHT" for the first value and "top", "TOP", "center", "bottom", or "BOTTOM" for the second value, or this vector contains two numeric values between 0 and 1 that specifies the x and y value of the center of the text. The uppercase values correspond to the position without margins (so tighter to the frame). The default value is controlled by the argument "attr.position" of tm_layout.
just	Justification of the attribute relative to the point coordinates. The first value specifies horizontal and the second value vertical justification. Possible values are: "left" , "right", "center", "bottom", and "top". Numeric values of 0 specify left/bottom alignment and 1 right/top alignment. This option is only used, if position is specified by numeric coordinates. The default value is controlled by the argument "attr.just" of tm_layout.

See Also

tm_xlab

Examples

```
current.mode <- tmap_mode("plot")

data(NLD_muni, NLD_prov)

tm_shape(NLD_muni) +
    tm_fill(col="population", convert2density=TRUE,
        style="kmeans", title = expression("Population (per " * km^2 * ")")) +
    tm_borders("grey25", alpha=.5) +
    tm_shape(NLD_prov) +
    tm_borders("grey40", lwd=2) +
```

```
tm_format("NLD", bg.color="white", frame = TRUE) +
tm_credits("(c) Statistics Netherlands (CBS) and\nKadaster Nederland", position=c("left", "bottom"))

# restore current mode
tmap_mode(current.mode)
```

`tm_facets` *Small multiples*

Description

Creates a `tmap-element` that specifies facets (small multiples). Small multiples can be created in two ways: 1) by specifying the `by` argument with one or two variable names, by which the data is grouped, 2) by specifying multiple variable names in any of the aesthetic argument of the layer functions (for instance, the argument `col` in `tm_fill`). This function further specifies the facets, for instance number of rows and columns, and whether the coordinate and scales are fixed or free (i.e. independent of each other). An overview of the different approaches to create facets is provided in the examples.

Usage

```
tm_facets(by = NULL, along = NULL, as.layers = FALSE, ncol = NA,
  nrow = NA, free.coords = !as.layers, drop.units = TRUE,
  drop.empty.facets = TRUE, drop.NA.facets = FALSE, sync = NA,
  showNA = NA, textNA = "Missing", free.scales = is.null(by) &&
  is.null(along), free.scales.fill = free.scales,
  free.scales.symbol.size = free.scales,
  free.scales.symbol.col = free.scales,
  free.scales.symbol.shape = free.scales,
  free.scales.text.size = free.scales,
  free.scales.text.col = free.scales,
  free.scales.line.col = free.scales,
  free.scales.line.lwd = free.scales, free.scales.raster = free.scales,
  inside.original.bbox = FALSE, scale.factor = 2,
  drop.shapes = drop.units)
```

Arguments

`by`	data variable name by which the data is split, or a vector of two variable names to split the data by two variables (where the first is used for the rows and the second for the columns).
`along`	data variable name by which the data is split and plotted on separate pages. This is especially useful for animations made with `tmap_animation`. The `along` argument can be used in combination with the `by` argument. It is only supported in `"plot"` mode (so not in `"view"` mode).
`as.layers`	logical that determines whether facets are shown as different layers in `"view"` mode. By default `FALSE`, i.e. facets are drawn as small multiples.

ncol number of columns of the small multiples grid. Not applicable if by contains two variable names.

nrow number of rows of the small multiples grid. Not applicable if by contains two variable names.

free.coords logical. If the by argument is specified, should each map has its own coordinate ranges? By default TRUE, unless facets are shown in as different layers (as.layers = TRUE)

drop.units logical. If the by argument is specified, should non-selected spatial units be dropped? If FALSE, they are plotted where mapped aesthetics are regarded as missing values. Not applicable for raster shapes. By default TRUE.

drop.empty.facets
logical. If the by argument is specified, should empty facets be dropped? Empty facets occur when the by-variable contains unused levels. When TRUE and two by-variables are specified, empty rows and columns are dropped.

drop.NA.facets logical. If the by argument is specified, and all values of the defined aesthetic variables (e.g. col from tm_fill) for specific facets, should these facets be dropped? FALSE by default.

sync logical. Should the navigation in view mode (zooming and panning) be synchronized? By default TRUE if the facets have the same bounding box. This is generally the case when rasters are plotted, or when free.coords is FALSE.

showNA If the by argument is specified, should missing values of the by-variable be shown in a facet? If two by-variables are specified, should missing values be shown in an additional row and column? If NA, missing values only are shown if they exist. Similar to the useNA argument of table, where TRUE, FALSE, and NA correspond to "always", "no", and "ifany" respectively.

textNA text used for facets of missing values.

free.scales logical. Should all scales of the plotted data variables be free, i.e. independent of each other? Possible data variables are color from tm_fill, color and size from tm_symbols and line color from tm_lines.

free.scales.fill
logical. Should the color scale for the choropleth be free?

free.scales.symbol.size
logical. Should the symbol size scale for the symbol map be free?

free.scales.symbol.col
logical. Should the color scale for the symbol map be free?

free.scales.symbol.shape
logical. Should the symbol shape scale for the symbol map be free?

free.scales.text.size
logical. Should the text size scale be free?

free.scales.text.col
logical. Should the text color scale be free?

free.scales.line.col
Should the line color scale be free?

free.scales.line.lwd
Should the line width scale be free?

`free.scales.raster`
: Should the color scale for raster layers be free?

`inside.original.bbox`
: If `free.coords`, should the bounding box of each small multiple be inside the original bounding box?

`scale.factor`
: Number that determines how the elements (e.g. font sizes, symbol sizes, line widths) of the small multiples are scaled in relation to the scaling factor of the shapes. The elements are scaled to the `scale.factor`th root of the scaling factor of the shapes. So, for `scale.factor=1`, they are scaled proportional to the scaling of the shapes. Since elements, especially text, are often too small to read, a higher value is recommended. By default, `scale.factor=2`.

`drop.shapes`
: deprecated: renamed to `drop.units`

Details

The global option `limits` controls the limit of the number of facets that are plotted. By default, `tmap_options(limits=c(facets.plot=64, facets.view=4))`. The maximum number of interactive facets is set to four since otherwise it may become very slow.

Value

tmap-element

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

vignette("tmap-getstarted")

Examples

```
data(World, NLD_muni, NLD_prov, land, metro)

current.mode <- tmap_mode("plot")

# CASE 1: Facets defined by constant values
tm_shape(World) +
    tm_fill(c("forestgreen", "goldenrod")) +
tm_format("World", title=c("A green world", "A dry world"), bg.color="lightskyblue2",
    title.position=c("left", "bottom"))

# CASE 2: Facets defined by multiple variables
tm_shape(World) +
    tm_polygons(c("well_being", "life_exp"),
     style=c("pretty", "fixed"), breaks=list(NULL, c(65,70,75,80,85)),
     palette=list("Oranges", "Purples"),
     border.col = "black",
        title=c("Well-Being Index", "Life Expectancy")) +
```

```
tm_format("World")

## Not run:
tm_shape(NLD_muni) +
    tm_fill(c("pop_0_14", "pop_15_24", "pop_25_44", "pop_45_64", "pop_65plus"),
        style="kmeans",
        palette=list("Oranges", "Greens", "Blues", "Purples", "Greys"),
        title=c("Population 0 to 14", "Population 15 to 24", "Population 25 to 44",
            "Population 45 to 64", "Population 65 and older")) +
tm_shape(NLD_prov) +
    tm_borders() +
tm_format("NLD", frame = TRUE, asp=0)

## End(Not run)

# CASE 3: Facets defined by group-by variable(s)
# A group-by variable that divides the objects spatially
tm_shape(NLD_prov) +
    tm_polygons("gold2") +
    tm_facets(by="name")

## Not run:
tm_shape(NLD_muni) +
    tm_borders() +
    tm_facets(by="province") +
    tm_fill("population", style="kmeans", convert2density = TRUE) +
tm_shape(NLD_prov) +
    tm_borders(lwd=4) +
    tm_facets(by="name")

## End(Not run)

# The objects are divided by a non-spatial variable (e.g. date/time)
if (require(dplyr) && require(tidyr)) {
metro_long <- metro %>%
gather(year, population, -name, -name_long, -iso_a3, -geometry) %>%
mutate(year = as.integer(substr(year, 4, 7)))

tm_shape(metro_long) +
tm_bubbles("population") +
tm_facets(by = "year")
}
## Not run:
tm_shape(land) +
tm_raster("black") +
tm_facets(by="cover_cls")

## End(Not run)

# Facets defined by two group-by variables
## Not run:
World$HPI3 <- cut(World$HPI, breaks = c(20, 35, 50, 65),
    labels = c("HPI low", "HPI medium", "HPI high"))
```

```
World$GDP3 <- cut(World$gdp_cap_est, breaks = c(0, 5000, 20000, Inf),
    labels = c("GDP low", "GDP medium", "GDP high"))

tm_shape(World) +
tm_fill("HPI3", palette="Dark2", colorNA="grey90", legend.show = FALSE) +
tm_facets(c("HPI3", "GDP3"), showNA=FALSE, free.coords = FALSE)

metro_edited <- metro %>%
mutate(pop1950cat = cut(pop1950, breaks=c(0.5, 1, 1.5, 2, 3, 5, 10, 40)*1e6),
   pop2020cat = cut(pop2020, breaks=c(0.5, 1, 1.5, 2, 3, 5, 10, 40)*1e6))

tm_shape(World) +
tm_fill() +
tm_shape(metro_edited) +
tm_dots("red", size = .5) +
tm_facets(c("pop1950cat", "pop2020cat"), free.coords = FALSE) +
tm_layout(panel.label.rot = c(0, 90), panel.label.size = 2)

## End(Not run)

# restore current mode
tmap_mode(current.mode)
```

`tm_fill` *Draw polygons*

Description

Creates a tmap-element that draws the polygons. `tm_fill` fills the polygons. Either a fixed color is used, or a color palette is mapped to a data variable. `tm_borders` draws the borders of the polygons. `tm_polygons` fills the polygons and draws the polygon borders.

Usage

```
tm_fill(col = NA, alpha = NA, palette = NULL,
  convert2density = FALSE, area = NULL, n = 5,
  style = ifelse(is.null(breaks), "pretty", "fixed"), breaks = NULL,
  interval.closure = "left", labels = NULL, midpoint = NULL,
  stretch.palette = TRUE, contrast = NA, colorNA = NA,
  textNA = "Missing", showNA = NA, colorNULL = NA, thres.poly = 0,
  title = NA, legend.show = TRUE, legend.format = list(),
  legend.is.portrait = TRUE, legend.reverse = FALSE,
  legend.hist = FALSE, legend.hist.title = NA, legend.z = NA,
  legend.hist.z = NA, id = NA, popup.vars = NA,
  popup.format = list(), group = NA, auto.palette.mapping = NULL,
  max.categories = NULL, ...)

tm_borders(col = NA, lwd = 1, lty = "solid", alpha = NA,
  group = NA)
```

```
tm_polygons(col = NA, alpha = NA, border.col = NA,
  border.alpha = NA, group = NA, ...)
```

Arguments

col	For tm_fill, it is one of • a single color value • the name of a data variable that is contained in shp. Either the data variable contains color values, or values (numeric or categorical) that will be depicted by a color palette (see palette. In the latter case, a choropleth is drawn. • "MAP_COLORS". In this case polygons will be colored such that adjacent polygons do not get the same color. See the underlying function map_coloring for details. For tm_borders, it is a single color value that specifies the border line color. If multiple values are specified, small multiples are drawn (see details).
alpha	transparency number between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the col is used (normally 1).
palette	a palette name or a vector of colors. See tmaptools::palette_explorer() for the named palettes. Use a "-" as prefix to reverse the palette. The default palette is taken from tm_layout's argument aes.palette, which typically depends on the style. The type of palette from aes.palette is automatically determined, but can be overwritten: use "seq" for sequential, "div" for diverging, and "cat" for categorical.
convert2density	boolean that determines whether col is converted to a density variable. Should be TRUE when col consists of absolute numbers. The area size is either approximated from the shape object, or given by the argument area.
area	Name of the data variable that contains the area sizes in squared kilometer.
n	preferred number of classes (in case col is a numeric variable).
style	method to process the color scale when col is a numeric variable. Discrete options are "cat", "fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", and "jenks". A numeric variable is processed as a categorical variable when using "cat", i.e. each unique value will correspond to a distinct category. For the other discrete options, see the details in classIntervals. Continuous options are "cont" and "order". The former maps the values of col to a smooth gradient, whereas the latter maps the order of values of col to a smooth gradient. They are the continuous variants of respectively the discrete methods "equal" and quantile".
breaks	in case style=="fixed", breaks should be specified. The breaks argument can also be used when style="cont". In that case, the breaks are mapped evenly to the sequential or diverging color palette.
interval.closure	value that determines whether where the intervals are closed: "left" or "right". Only applicable if col is a numeric variable.

labels — labels of the classes.

midpoint — The value mapped to the middle color of a diverging palette. By default it is set to 0 if negative and positive values are present. In that case, the two sides of the color palette are assigned to negative respectively positive values. If all values are positive or all values are negative, then the midpoint is set to NA, which means that the value that corresponds to the middle color class (see style) is mapped to the middle color. Only applies when col is a numeric variable. If it is specified for sequential color palettes (e.g. "Blues"), then this color palette will be treated as a diverging color palette.

stretch.palette — Logical that determines whether the categorical color palette should be stretched if there are more categories than colors. If TRUE (default), interpolated colors are used (like a rainbow). If FALSE, the palette is repeated.

contrast — vector of two numbers that determine the range that is used for sequential and diverging palettes (applicable when auto.palette.mapping=TRUE). Both numbers should be between 0 and 1. The first number determines where the palette begins, and the second number where it ends. For sequential palettes, 0 means the brightest color, and 1 the darkest color. For diverging palettes, 0 means the middle color, and 1 both extremes. If only one number is provided, this number is interpreted as the endpoint (with 0 taken as the start).

colorNA — color used for missing values. Use NULL for transparency.

textNA — text used for missing values.

showNA — logical that determines whether missing values are named in the legend. By default (NA), this depends on the presence of missing values.

colorNULL — colour for polygons that are shown on the map that are out of scope

thres.poly — number that specifies the threshold at which polygons are taken into account. The number itself corresponds to the proportion of the area sizes of the polygons to the total polygon size. By default, all polygons are drawn. To ignore polygons that are not visible in a normal plot, a value like 1e-05 is recommended.

title — title of the legend element

legend.show — logical that determines whether the legend is shown

legend.format — list of formatting options for the legend numbers. Only applicable if labels is undefined. Parameters are:

fun Function to specify the labels. It should take a numeric vector, and should return a character vector of the same size. By default it is not specified. If specified, the list items scientific, format, and digits (see below) are not used.

scientific Should the labels be formatted scientifically? If so, square brackets are used, and the format of the numbers is "g". Otherwise, format="f", and text.separator, text.less.than, and text.or.more are used. Also, the numbers are automatically rounded to millions or billions if applicable.

format By default, "f", i.e. the standard notation xxx.xxx, is used. If scientific=TRUE then "g", which means that numbers are formatted scientifically, i.e. n.dddE+nn if needed to save space.

digits Number of digits after the decimal point if `format="f"`, and the number of significant digits otherwise.

text.separator Character string to use to separate numbers in the legend (default: "to").

text.less.than Character value(s) to use to translate "Less than". When a character vector of length 2 is specified, one for each word, these words are aligned when `text.to.columns = TRUE`

text.or.more Character value(s) to use to translate "or more". When a character vector of length 2 is specified, one for each word, these words are aligned when `text.to.columns = TRUE`

text.align Value that determines how the numbers are aligned, `"left"`, `"center"` or `"right"`. By default `"left"` for legends in portrait format (`legend.is.protrait = TRUE`), and `"center"` otherwise.

text.to.columns Logical that determines whether the text is aligned to three columns (from, text.separator, to). By default `FALSE`.

... Other arguments passed on to `formatC`

`legend.is.portrait`
: logical that determines whether the legend is in portrait mode (`TRUE`) or landscape (`FALSE`)

`legend.reverse`
: logical that determines whether the items are shown in reverse order, i.e. from bottom to top when `legend.is.portrait = TRUE` and from right to left when `legend.is.portrait = FALSE`

`legend.hist`
: logical that determines whether a histogram is shown

`legend.hist.title`
: title for the histogram. By default, one title is used for both the histogram and the normal legend.

`legend.z`
: index value that determines the position of the legend element with respect to other legend elements. The legend elements are stacked according to their z values. The legend element with the lowest z value is placed on top.

`legend.hist.z`
: index value that determines the position of the histogram legend element

`id`
: name of the data variable that specifies the indices of the polygons. Only used for `"view"` mode (see `tmap_mode`).

`popup.vars`
: names of data variables that are shown in the popups in `"view"` mode. If `convert2density=TRUE`, the derived density variable name is suffixed with `_density`. If `NA` (default), only aesthetic variables (i.e. specified by `col` and `lwd`) are shown). If they are not specified, all variables are shown. Set popup.vars to `FALSE` to disable popups. When a vector of variable names is provided, the names (if specified) are printed in the popups.

`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.

`group`
: name of the group to which this layer belongs in view mode. Each group can be selected or deselected in the layer control item. Groups can either be

	specified as base or overlay groups in tm_view (arguments base.groups and overlay.groups).
auto.palette.mapping	deprecated. It has been replaced by midpoint for numeric variables and stretch.palette for categorical variables.
max.categories	deprecated. It has moved to tmap_options.
...	for tm_polygons, these arguments passed to either tm_fill or tm_borders. For tm_fill, these arguments are passed on to map_coloring.
lwd	border line width (see par)
lty	border line type (see par)
border.col	border line color
border.alpha	transparency number between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the col is used (normally 1).

Details

Small multiples can be drawn in two ways: either by specifying the by argument in tm_facets, or by defining multiple variables in the aesthetic arguments. The aesthetic argument of tm_fill (and tm_polygons) is col. In the latter case, the arguments, except for thres.poly, and the ones starting with legend., can be specified for small multiples as follows. If the argument normally only takes a single value, such as n, then a vector of those values can be specified, one for each small multiple. If the argument normally can take a vector, such as palette, then a list of those vectors (or values) can be specified, one for each small multiple.

Value

tmap-element

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

vignette("tmap-getstarted")

Examples

```
data(World)

# Constant fill
tm_shape(World) + tm_fill("darkolivegreen3") + tm_format("World", title="A green World")

# Borders only
tm_shape(World) + tm_borders()

# Data variable containing colours values
World$isNLD <- ifelse(World$name=="Netherlands", "darkorange", "darkolivegreen3")
```

```
tm_shape(World) +
    tm_fill("isNLD") +
tm_layout("Find the Netherlands!")

tm_shape(World) +
tm_polygons("economy", title="Economy", id="name") +
tm_text("iso_a3", size="AREA", scale=1.5) +
tm_format("World")

# Numeric data variable
tm_shape(World) +
tm_polygons("HPI", palette="RdYlGn", style="cont", n=8,
title="Happy Planet Index", id="name") +
tm_text("iso_a3", size="AREA", scale=1.5) +
tm_style("grey") +
tm_format("World")

## Not run:
data(NLD_prov, NLD_muni)
# Map coloring algorithm
tm_shape(NLD_prov) +
    tm_fill("name", legend.show = FALSE) +
tm_shape(NLD_muni) +
    tm_polygons("MAP_COLORS", palette="Greys", alpha = .25) +
tm_shape(NLD_prov) +
    tm_borders(lwd=2) +
    tm_text("name", shadow=TRUE) +
tm_format("NLD", title="Dutch provinces and\nmunicipalities", bg.color="white")

# Cartogram
if (require(cartogram)) {
NLD_prov_pop <- cartogram(NLD_prov, "population")
tm_shape(NLD_prov_pop) +
tm_polygons("origin_non_west", title = "Non-western origin (%)")
}

## End(Not run)

# TIP: check out these examples in view mode, enabled with tmap_mode("view")
```

`tm_grid` *Coordinate grid lines*

Description

Creates a tmap-element that draws coordinate grid lines. It serves as a layer that can be drawn anywhere between other layers. By default the coordinate system of the (master) shape object is used, which results in horizontal and vertical lines. Alternatively, grid lines can be reprojected, for instance to latitude longitude coordinates, and hence be curved.

Usage

```
tm_grid(x = NA, y = NA, n.x = NA, n.y = NA, projection = NA,
  col = NA, lwd = 1, alpha = NA, labels.size = 0.6,
  labels.col = NA, labels.rot = c(0, 0),
  labels.format = list(big.mark = ","), labels.margin.x = 0,
  labels.margin.y = 0, labels.inside.frame = TRUE)
```

Arguments

x	x coordinates for vertical grid lines. If NA, it is specified with a pretty scale and n.x.
y	y coordinates for horizontal grid lines. If NA, it is specified with a pretty scale and n.y.
n.x	preferred number of grid lines for the x axis.
n.y	preferred number of grid lines for the y axis.
projection	projection character. If specified, the grid lines are projected accordingly. See set_projection for projection details. Many world maps are projected, but still have latitude longitude ("longlat") grid lines.
col	color of the grid lines.
lwd	line width of the grid lines
alpha	alpha transparency of the grid lines. Number between 0 and 1. By default, the alpha transparency of col is taken.
labels.size	font size of the tick labels
labels.col	font color of the tick labels
labels.rot	Rotation angles of the labels. Vector of two values: the first is the rotation angle (in degrees) of the tick labels on the x axis and the second is the rotation angle of the tick labels on the y axis. Only 0, 90, 180, and 270 are valid values.
labels.format	list of formatting options for the grid labels. Parameters are: **fun** Function to specify the labels. It should take a numeric vector, and should return a character vector of the same size. By default it is not specified. If specified, the list items scientific, format, and digits (see below) are not used. **scientific** Should the labels be formatted scientifically? If so, square brackets are used, and the format of the numbers is "g". Otherwise, format="f", and text.separator, text.less.than, and text.or.more are used. Also, the numbers are automatically rounded to millions or billions if applicable. **format** By default, "f", i.e. the standard notation xxx.xxx, is used. If scientific=TRUE then "g", which means that numbers are formatted scientifically, i.e. n.dddE+nn if needed to save space. **digits** Number of digits after the decimal point if format="f", and the number of significant digits otherwise. **...** Other arguments passed on to formatC
labels.margin.x	margin between tick labels of x axis and the frame

`labels.margin.y`
: margin between tick labels of y axis and the frame

`labels.inside.frame`
: Show labels inside the frame?

`tm_iso` *Draw iso (contour) lines with labels*

Description

This function is a wrapper of `tm_lines` and `tm_text` aimed to draw isopleths, which can be created with `smooth_map`.

Usage

```
tm_iso(col = NA, text = "level", size = 0.5, remove.overlap = TRUE,
  along.lines = TRUE, overwrite.lines = TRUE, group = NA, ...)
```

Arguments

`col`	line color. See `tm_lines`.
`text`	text to display. By default, it is the variable named `"level"` of the shape that is created with `smooth_map`
`size`	text size (see `tm_text`)
`remove.overlap`	see `tm_text`
`along.lines`	see `tm_text`
`overwrite.lines`	see `tm_text`
`group`	name of the group to which this layer belongs in view mode. Each group can be selected or deselected in the layer control item. Groups can either be specified as base or overlay groups in `tm_view` (arguments `base.groups` and `overlay.groups`).
`...`	arguments passed on to `tm_lines` or `tm_text`

See Also

`smooth_map`

tm_layout *Layout of cartographic maps*

Description

This element specifies the map layout. The main function tm_layout controls title, margins, aspect ratio, colors, frame, legend, among many other things. The function tm_legend is a shortcut to access all legend. arguments without this prefix. The other functions are wrappers for two purposes: tm_format specifies position related layout settings such as margins, and tm_style specifies general styling related layout settings such as colors and font. Typically, the former functions are shape dependent, and the latter functions are shape independent. See details for predefined styles and formats. With tmap.style, a default style can be specified. Multiple tm_layout elements (or wrapper functions) can be stacked: called arguments will be overwritten.

Usage

```
tm_layout(title, scale, title.size, bg.color, aes.color, aes.palette,
  attr.color, sepia.intensity, saturation, frame, frame.lwd,
  frame.double.line, asp, outer.margins, inner.margins, between.margin,
  outer.bg.color, fontface, fontfamily, compass.type, earth.boundary,
  earth.boundary.color, earth.boundary.lwd, earth.datum, space.color,
  legend.show, legend.only, legend.outside, legend.outside.position,
  legend.outside.size, legend.position, legend.stack, legend.just,
  legend.width, legend.height, legend.hist.height, legend.hist.width,
  legend.title.size, legend.title.fontface, legend.title.fontfamily,
  legend.text.size, legend.text.fontface, legend.text.fontfamily,
  legend.hist.size, legend.format, legend.frame, legend.frame.lwd,
  legend.text.color, legend.bg.color, legend.bg.alpha,
  legend.hist.bg.color, legend.hist.bg.alpha, title.snap.to.legend,
  title.position, title.color, title.fontface, title.fontfamily,
  title.bg.color, title.bg.alpha, panel.show, panel.labels,
  panel.label.size, panel.label.color, panel.label.fontface,
  panel.label.fontfamily, panel.label.bg.color, panel.label.height,
  panel.label.rot, main.title, main.title.size, main.title.color,
  main.title.fontface, main.title.fontfamily, main.title.position,
  attr.outside, attr.outside.position, attr.outside.size, attr.position,
  attr.just, design.mode, basemaps, basemaps.alpha, bg.overlay,
  bg.overlay.alpha)

tm_legend(...)

tm_style(style, ...)

tm_format(format, ...)
```

Arguments

title	Global title of the map. For small multiples, multiple titles can be specified. The title is drawn inside the map. Alternatively, use panel.labels to print the map as a panel, with the title inside the panel header (especially useful for small multiples). Another alternative is the main.title which prints a title above the map. Titles for the legend items are specified at the layer functions (e.g. tm_fill).
scale	numeric value that serves as the global scale parameter. All font sizes, symbol sizes, border widths, and line widths are controlled by this value. Each of these elements can be scaled independently with the scale, lwd, or size arguments provided by the tmap-elements.
title.size	Relative size of the title
bg.color	Background color. By default it is "white". A recommended alternative for choropleths is light grey (e.g., "grey85").
aes.color	Default color values for the aesthetics layers. Should be a named vector with the names chosen from: fill, borders, symbols, dots, lines, text, na. Use "#00000000" for transparency.
aes.palette	Default color palettes for the aesthetics. It takes a list of three items: seq for sequential palettes, div for diverging palettes, and cat for categorical palettes. By default, Color Brewer palettes (see (see tmaptools::palette_explorer())) are used. It is also possible provide a vector of colors for any of these items.
attr.color	Default color value for map attributes
sepia.intensity	Number between 0 and 1 that defines the amount of sepia effect, which gives the map a brown/yellowish flavour. By default this effect is disabled (sepia.intensity=0). All colored used in the map are adjusted with this effect.
saturation	Number that determines how much saturation (also known as chroma) is used: saturation=0 is greyscale and saturation=1 is normal. A number larger than 1 results in very saturated maps. All colored used in the map are adjusted with this effect. Hacking tip: use a negative number.
frame	Either a boolean that determines whether a frame is drawn, or a color value that specifies the color of the frame.
frame.lwd	width of the frame
frame.double.line	draw a double frame line border?
asp	Aspect ratio. The aspect ratio of the map (width/height). If NA, it is determined by the bounding box (see argument bbox of tm_shape), the outer.margins, and the inner.margins. If 0, then the aspect ratio is adjusted to the aspect ratio of the device.
outer.margins	Relative margins between device and frame. Vector of four values specifying the bottom, left, top, and right margin. Values are between 0 and 1. When facets are created, the outer margins are the margins between the outer panels and the device borders (see also between.margin)

`inner.margins` Relative margins inside the frame. Vector of four values specifying the bottom, left, top, and right margin. Values are between 0 and 1. By default, 0 for each side if master shape is a raster, otherwise 0.02.

`between.margin` Margin between facets (small multiples) in number of text line heights. The height of a text line is automatically scaled down based on the number of facets.

`outer.bg.color` Background color outside the frame.

`fontface` global font face for the text in the map. It can also be set locally per element (see e.g. `title.fontface`).

`fontfamily` global font family for the text in the map. It can also be set locally per (see e.g. `title.fontfamily`).

`compass.type` type of compass, one of: `"arrow"`, `"4star"`, `"8star"`, `"radar"`, `"rose"`. Of course, only applicable if a compass is shown. The compass type can also be set within `tm_compass`.

`earth.boundary` Logical that determines whether the boundaries of the earth are shown or an object that specifies the boundaries. This object can be a vector of size four, a 2 by 2 matrix (bounding box), or an `extent` object. By default, the boundaries are `c(-180, -90, 180, 90)`. Useful for projected world maps. Often, it is useful to crop both poles (e.g., with `c(-180, -88, 180, 88)`).

`earth.boundary.color`
Color of the earth boundary.

`earth.boundary.lwd`
Line width of the earth boundary.

`earth.datum` Geodetic datum to determine the earth boundary. By default `"WGS84"`, other frequently used datums are `"NAD83"` and `"NAD27"`. Any other `PROJ.4` character string can be used.

`space.color` Color of the space, i.e. the region inside the frame, and outside the earth boundary.

`legend.show` Logical that determines whether the legend is shown.

`legend.only` logical. Only draw the legend (without map)? Particularly useful for small multiples with a common legend.

`legend.outside` Logical that determines whether the legend is plot outside of the map/facets. Especially useful when using facets that have a common legend (i.e. with `free.scales=FALSE`).

`legend.outside.position`
Character that determines the outside position of the legend. Only applicable when `legend.outside=TRUE`. One of: `"right"`, `"left"`, `"top"`, or `"bottom"`.

`legend.outside.size`
Numeric value that determines the relative size of the legend, when `legend.outside=TRUE`. If the first value of `legend.outside.position` is `"top"` or `"bottom"`, then it is the width of the legend, else it is the height of the legend.

`legend.position`
Position of the legend. Vector of two values, specifying the x and y coordinates. Either this vector contains `"left"`, `"LEFT"`, `"center"`, `"right"`, or `"RIGHT"` for the first value and `"top"`, `"TOP"`, `"center"`, `"bottom"`, or `"BOTTOM"` for the

second value, or this vector contains two numeric values between 0 and 1 that specifies the x and y coordinates of the left bottom corner of the legend. The uppercase values correspond to the position without margins (so tighter to the frame). By default, it is automatically placed in the corner with most space based on the (first) shape object. If `legend.outside=TRUE`, this argument specifies the legend position within the outside panel.

`legend.stack` Value that determines how the legend items are stacked: `"vertical"` or `"horizontal"`.

`legend.just` Justification of the legend relative to the point coordinates. The first value specifies horizontal and the second value vertical justification. Possible values are: `"left"` , `"right"`, `"center"`, `"bottom"`, and `"top"`. Numeric values of 0 specify left/bottom alignment and 1 right/top alignment. This option is only used, if `legend.position` is specified by numeric coordinates.

`legend.width` width of the legend. If it is a negative number, it will be the exact legend width. If it is a positive number (by default), it will be the maximum legend width; the actual legend width will be decreased automatically based on the legend content and font sizes.

`legend.height` height of the legend. If it is a negative number, it will be the exact legend height. If it is a positive number (by default), it will be the maximum legend height; the actual legend height will be decreased automatically based on the legend content and font sizes.

`legend.hist.height`
height of the histogram. This height is initial. If the total legend is downscaled to `legend.height`, the histogram is downscaled as well.

`legend.hist.width`
width of the histogram. By default, it is equal to the `legend.width`.

`legend.title.size`
Relative font size for the legend title

`legend.title.fontface`
font face for the legend title. By default, set to the global parameter `fontface`.

`legend.title.fontfamily`
font family for the legend title. By default, set to the global parameter `fontfamily`.

`legend.text.size`
Relative font size for the legend text elements

`legend.text.fontface`
font face for the legend text labels. By default, set to the global parameter `fontface`.

`legend.text.fontfamily`
font family for the legend text labels. By default, set to the global parameter `fontfamily`.

`legend.hist.size`
Relative font size for the choropleth histogram

`legend.format` list of formatting options for the legend numbers. Only applicable for layer functions (such as `tm_fill`) where `labels` is undefined. Parameters are:

fun Function to specify the labels. It should take a numeric vector, and should return a character vector of the same size. By default it is not specified. If specified, the list items `scientific`, `format`, and `digits` (see below) are not used.

scientific Should the labels be formatted scientifically? If so, square brackets are used, and the `format` of the numbers is `"g"`. Otherwise, `format="f"`, and `text.separator`, `text.less.than`, and `text.or.more` are used. Also, the numbers are automatically rounded to millions or billions if applicable.

format By default, `"f"`, i.e. the standard notation `xxx.xxx`, is used. If `scientific=TRUE` then `"g"`, which means that numbers are formatted scientifically, i.e. `n.dddE+nn` if needed to save space.

digits Number of digits after the decimal point if `format="f"`, and the number of significant digits otherwise.

text.separator Character string to use to separate numbers in the legend (default: "to").

text.less.than Character value(s) to use to translate "Less than". When a character vector of length 2 is specified, one for each word, these words are aligned when `text.to.columns = TRUE`

text.or.more Character value(s) to use to translate "or more". When a character vector of length 2 is specified, one for each word, these words are aligned when `text.to.columns = TRUE`

text.align Value that determines how the numbers are aligned, `"left"`, `"center"` or `"right"`. By default `"left"` for legends in portrait format (`legend.is.protrait = TRUE`), and `"center"` otherwise.

text.to.columns Logical that determines whether the text is aligned to three columns (from, text.separator, to). By default `FALSE`.

text.align Value that determines how the numbers are aligned, `"left"`, `"center"` or `"right"`. By default `"left"` for legends in portrait format (`legend.is.protrait = TRUE`), and `"center"` otherwise.

text.to.columns Logical that determines whether the text is aligned to three columns (from, text.separator, to). By default `FALSE`.

... Other arguments passed on to `formatC`

`legend.frame` either a logical that determines whether the legend is placed inside a frame, or a color that directly specifies the frame border color.

`legend.frame.lwd`
line width of the legend frame (applicable if `legend.frame` is `TRUE` or a color)

`legend.text.color`
color of the legend text

`legend.bg.color`
Background color of the legend. Use `TRUE` to match with the overall background color `bg.color`.

`legend.bg.alpha`
Transparency number between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the `legend.bg.color` is used (normally 1).

`legend.hist.bg.color`
Background color of the histogram

`legend.hist.bg.alpha`
Transparency number between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the `legend.hist.bg.color` is used (normally 1).

`title.snap.to.legend`
: Logical that determines whether the title is part of the legend. By default `FALSE`, unless the legend is drawn outside the map (see `legend.outside`).

`title.position`
: Position of the title. Vector of two values, specifying the x and y coordinates. Either this vector contains "left", "LEFT", "center", "right", or "RIGHT" for the first value and "top", "TOP", "center", "bottom", or "BOTTOM" for the second value, or this vector contains two numeric values between 0 and 1 that specifies the x and y coordinates of the tile. The uppercase values correspond to the position without margins (so tighter to the frame). By default the title is placed on top of the legend (determined by `legend.position`).

`title.color`
: color of the title

`title.fontface`
: font face for the title. By default, set to the global parameter `fontface`.

`title.fontfamily`
: font family for the title. By default, set to the global parameter `fontfamily`.

`title.bg.color`
: background color of the title. Use `TRUE` to match with the overall background color `bg.color`. By default, it is `TRUE` if `legend.frame` is `TRUE` or a color.

`title.bg.alpha`
: Transparency number between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the `title.bg.color` is used (normally 1).

`panel.show`
: Logical that determines if the map(s) are shown as panels. If `TRUE`, the title will be placed in the panel header instead of inside the map. By default, it is `TRUE` when small multiples are created with the `by` variable. (See `tm_facets`)

`panel.labels`
: Panel labels. Only applicable when `panel.show` is `TRUE`. For cross tables facets, it should be a list containing the row names in the first, and column names in the second item.

`panel.label.size`
: Relative font size of the panel labels

`panel.label.color`
: Font color of the panel labels

`panel.label.fontface`
: font face for the panel labels. By default, set to the global parameter `fontface`.

`panel.label.fontfamily`
: font family for the panel labels. By default, set to the global parameter `fontfamily`.

`panel.label.bg.color`
: Background color of the panel labels

`panel.label.height`
: Height of the labels in number of text line heights.

`panel.label.rot`
: Rotation angles of the panel labels. Vector of two values: the first is the rotation angle (in degrees) of the row panels, which are only used in cross-table facets (when `tm_facets`'s `by` is specified with two variables). The second is the rotation angle of the column panels.

`main.title`
: Title that is printed above the map (or small multiples). When multiple pages are generated (see `along` argument of `tm_facets`), a vector can be provided. By default, the main title is only printed when this `along` argument is specified.

`main.title.size`
: Size of the main title

`main.title.color`
: Color of the main title

`main.title.fontface`
: font face for the main title. By default, set to the global parameter `fontface`.

`main.title.fontfamily`
: font family for the main title. By default, set to the global parameter `fontfamily`.

`main.title.position`
: Position of the main title. Either a numeric value between 0 (left) and 1 (right), or a character value: `"left"`, `"center"`, or `"right"`.

`attr.outside`
: Logical that determines whether the attributes are plot outside of the map/facets.

`attr.outside.position`
: Character that determines the outside position of the attributes: `"top"` or `"bottom"`. Only applicable when `attr.outside=TRUE`. If the legend is also drawn outside (with `legend.outside=TRUE`) and on the same side of the map (e.g. also `"top"` or `"bottom"`), the attributes are placed between the map and the legend. This can be changed by setting `attr.outside.position` to `"TOP"` or `"BOTTOM"`: in this case, the attributes are placed above respectively below the legend.

`attr.outside.size`
: Numeric value that determines the relative height of the attribute viewport, when `attr.outside=TRUE`.

`attr.position`
: Position of the map attributes, which are `tm_credits`, `tm_scale_bar`, `tm_compass`, and `tm_minimap`. Vector of two values, specifying the x and y coordinates. The first value is `"left"`, `"LEFT"`, `"center"`, `"right"`, or `"RIGHT"`, and the second value `"top"`, `"TOP"`, `"center"`, `"bottom"`, or `"BOTTOM"`. The uppercase values correspond to the position without margins (so tighter to the frame). Positions can also be set separately in the map attribute functions. If `attr.outside=TRUE`, this argument specifies the position of the attributes within the outside panel.

`attr.just`
: Justification of the attributes relative to the point coordinates. The first value specifies horizontal and the second value vertical justification. Possible values are: `"left"` , `"right"`, `"center"`, `"bottom"`, and `"top"`. Numeric values of 0 specify left/bottom alignment and 1 right/top alignment. This option is only used, if `attr.position` is specified by numeric coordinates. It can also be specified per attribute function.

`design.mode`
: Logical that enables the design mode. If `TRUE`, inner and outer margins, legend position, aspect ratio are explicitly shown. Also, feedback text in the console is given.

`basemaps`
: Migrated to `tm_view`. vector of one or more names of baselayer maps used in the interactive view mode. See `tm_view`.

`basemaps.alpha`
: Migrated to `tm_view`. alpha transparency (opacity) of the basemaps. See `tm_view`.

`bg.overlay`
: Not used anymore. See `tm_view`.

`bg.overlay.alpha`
: Not used anymore. See `tm_view`.

`...`
: other arguments from `tm_layout`

`style`
: name of the style

`format`
: name of the format

Details

Predefined styles:

"white"	White background, commonly used colors (default)
"gray"/"grey"	Grey background, useful to highlight sequential palettes (e.g. in choropleths)
"natural"	Emulation of natural view: blue waters and green land
"bw"	Greyscale, obviously useful for greyscale printing
"classic"	Classic styled maps (recommended)
"cobalt"	Inspired by latex beamer style cobalt
"albatross"	Inspired by latex beamer style albatross
"beaver"	Inspired by latex beamer style beaver

Predefined formats

"World"	Format specified for world maps
"World_wide"	Format specified for world maps with more space for the legend
"NLD"	Format specified for maps of the Netherlands
"NLD_wide"	Format specified for maps of the Netherlands with more space for the legend

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

vignette("tmap-getstarted")

Examples

```
data(World, land)

tm_shape(World) +
    tm_fill("pop_est_dens", style="kmeans", title="Population density") +
tm_style("albatross", frame.lwd=10) + tm_format("World", title="The World")

## Not run:
tm_shape(land) +
tm_raster("elevation", breaks=c(-Inf, 250, 500, 1000, 1500, 2000, 2500, 3000, 4000, Inf),
palette = terrain.colors(9), title="Elevation", midpoint = NA) +
tm_shape(World, is.master=TRUE) +
tm_borders("grey20") +
tm_grid(projection="longlat", labels.size = .5) +
tm_text("name", size="AREA") +
tm_compass(position = c(.65, .15), color.light = "grey90") +
tm_credits("Eckert IV projection", position = c("right", "BOTTOM")) +
tm_style("classic") +
```

```
tm_layout(bg.color="lightblue",
inner.margins=c(.04,.03, .02, .01),
earth.boundary = TRUE,
space.color="grey90") +
tm_legend(position = c("left", "bottom"),
frame = TRUE,
bg.color="lightblue")

## End(Not run)

tm_shape(World, projection="robin") +
tm_polygons("HPI", palette="div", n=7,
title = "Happy Planet Index") +
tm_credits("Winkel Tripel projection", position = c("right", "BOTTOM")) +
tm_style("natural", earth.boundary = c(-180, -87, 180, 87), inner.margins = .05) +
tm_legend(position=c("left", "bottom"), bg.color="grey95", frame=TRUE)

# Example to illustrate the type of titles
tm_shape(World) +
tm_polygons(c("income_grp", "economy"), title = c("Legend Title 1", "Legend Title 2")) +
tm_layout(main.title = "Main Title",
main.title.position = "center",
main.title.color = "blue",
title = c("Title 1", "Title 2"),
title.color = "red",
panel.labels = c("Panel Label 1", "Panel Label 2"),
panel.label.color = "purple",
legend.text.color = "brown")

## Not run:
# global option tmap.style demo

# get current style
current.style <- tmap_style()

qtm(World, fill = "economy", format = "World")

tmap_style("col_blind")
qtm(World, fill = "economy", format = "World")

tmap_style("cobalt")
qtm(World, fill = "economy", format = "World")

# set to current style
tmap_style(current.style)

## End(Not run)

# TIP: check out these examples in view mode, enabled with tmap_mode("view")
```

---

tm_lines *Draw spatial lines*

Description

Creates a tmap-element that draw spatial lines.

Usage

```
tm_lines(col = NA, lwd = 1, lty = "solid", alpha = NA, scale = 1,
  lwd.legend = NULL, lwd.legend.labels = NULL, n = 5,
  style = ifelse(is.null(breaks), "pretty", "fixed"), breaks = NULL,
  interval.closure = "left", palette = NULL, labels = NULL,
  midpoint = NULL, stretch.palette = TRUE, contrast = NA,
  colorNA = NA, textNA = "Missing", showNA = NA, colorNULL = NA,
  title.col = NA, title.lwd = NA, legend.col.show = TRUE,
  legend.lwd.show = TRUE, legend.format = list(),
  legend.col.is.portrait = TRUE, legend.lwd.is.portrait = FALSE,
  legend.col.reverse = FALSE, legend.lwd.reverse = FALSE,
  legend.hist = FALSE, legend.hist.title = NA, legend.col.z = NA,
  legend.lwd.z = NA, legend.hist.z = NA, id = NA, popup.vars = NA,
  popup.format = list(), group = NA, auto.palette.mapping = NULL,
  max.categories = NULL)
```

Arguments

col	color of the lines. Either a color value or a data variable name. If multiple values are specified, small multiples are drawn (see details).
lwd	line width. Either a numeric value or a data variable. In the latter case, the class of the highest values (see style) will get the line width defined by scale. If multiple values are specified, small multiples are drawn (see details).
lty	line type.
alpha	transparency number between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the col is used (normally 1).
scale	line width multiplier number.
lwd.legend	vector of line widths that are shown in the legend. By default, this is determined automatically.
lwd.legend.labels	vector of labels for that correspond to lwd.legend.
n	preferred number of color scale classes. Only applicable when lwd is the name of a numeric variable.
style	method to process the color scale when col is a numeric variable. Discrete options are "cat", "fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", and "jenks". A numeric variable is processed as a categorical variable when using "cat", i.e. each unique value will correspond to a distinct category. For the other discrete options, see the details in classIntervals. Continuous options are "cont" and "order". The former maps the values of col to a smooth gradient, whereas the latter maps the order of values of col to a smooth gradient. They are the continuous variants of respectively the discrete methods "equal" and quantile".

breaks in case style=="fixed", breaks should be specified. The breaks argument can also be used when style="cont". In that case, the breaks are mapped evenly to the sequential or diverging color palette.

interval.closure value that determines whether where the intervals are closed: "left" or "right". Only applicable if col is a numeric variable.

palette a palette name or a vector of colors. See tmaptools::palette_explorer() for the named palettes. Use a "-" as prefix to reverse the palette. The default palette is taken from tm_layout's argument aes.palette, which typically depends on the style. The type of palette from aes.palette is automatically determined, but can be overwritten: use "seq" for sequential, "div" for diverging, and "cat" for categorical.

labels labels of the classes

midpoint The value mapped to the middle color of a diverging palette. By default it is set to 0 if negative and positive values are present. In that case, the two sides of the color palette are assigned to negative respectively positive values. If all values are positive or all values are negative, then the midpoint is set to NA, which means that the value that corresponds to the middle color class (see style) is mapped to the middle color. Only applies when col is a numeric variable. If it is specified for sequential color palettes (e.g. "Blues"), then this color palette will be treated as a diverging color palette.

stretch.palette Logical that determines whether the categorical color palette should be stretched if there are more categories than colors. If TRUE (default), interpolated colors are used (like a rainbow). If FALSE, the palette is repeated.

contrast vector of two numbers that determine the range that is used for sequential and diverging palettes (applicable when auto.palette.mapping=TRUE). Both numbers should be between 0 and 1. The first number determines where the palette begins, and the second number where it ends. For sequential palettes, 0 means the brightest color, and 1 the darkest color. For diverging palettes, 0 means the middle color, and 1 both extremes. If only one number is provided, this number is interpreted as the endpoint (with 0 taken as the start).

colorNA color used for missing values. Use NULL for transparency.

textNA text used for missing values.

showNA logical that determines whether missing values are named in the legend. By default (NA), this depends on the presence of missing values.

colorNULL colour for polygons that are shown on the map that are out of scope

title.col title of the legend element regarding the line colors

title.lwd title of the legend element regarding the line widths

legend.col.show logical that determines whether the legend for the line colors is shown

legend.lwd.show logical that determines whether the legend for the line widths is shown

legend.format list of formatting options for the legend numbers. Only applicable if labels is undefined. Parameters are:

fun Function to specify the labels. It should take a numeric vector, and should return a character vector of the same size. By default it is not specified. If specified, the list items `scientific`, `format`, and `digits` (see below) are not used.

scientific Should the labels be formatted scientifically? If so, square brackets are used, and the `format` of the numbers is `"g"`. Otherwise, `format="f"`, and `text.separator`, `text.less.than`, and `text.or.more` are used. Also, the numbers are automatically rounded to millions or billions if applicable.

format By default, `"f"`, i.e. the standard notation `xxx.xxx`, is used. If `scientific=TRUE` then `"g"`, which means that numbers are formatted scientifically, i.e. `n.dddE+nn` if needed to save space.

digits Number of digits after the decimal point if `format="f"`, and the number of significant digits otherwise.

text.separator Character string to use to separate numbers in the legend (default: "to").

text.less.than Character value(s) to use to translate "Less than". When a character vector of length 2 is specified, one for each word, these words are aligned when `text.to.columns = TRUE`

text.or.more Character value(s) to use to translate "or more". When a character vector of length 2 is specified, one for each word, these words are aligned when `text.to.columns = TRUE`

text.align Value that determines how the numbers are aligned, `"left"`, `"center"` or `"right"`. By default `"left"` for legends in portrait format (`legend.is.protrait = TRUE`), and `"center"` otherwise.

text.to.columns Logical that determines whether the text is aligned to three columns (from, text.separator, to). By default `FALSE`.

... Other arguments passed on to `formatC`

`legend.col.is.portrait`
: logical that determines whether the legend element regarding the line colors is in portrait mode (`TRUE`) or landscape (`FALSE`)

`legend.lwd.is.portrait`
: logical that determines whether the legend element regarding the line widths is in portrait mode (`TRUE`) or landscape (`FALSE`)

`legend.col.reverse`
: logical that determines whether the items of the legend regarding the line colors sizes are shown in reverse order, i.e. from bottom to top when `legend.col.is.portrait = TRUE` and from right to left when `legend.col.is.portrait = FALSE`

`legend.lwd.reverse`
: logical that determines whether the items of the legend regarding the line widths are shown in reverse order, i.e. from bottom to top when `legend.lwd.is.portrait = TRUE` and from right to left when `legend.lwd.is.portrait = FALSE`

`legend.hist` logical that determines whether a histogram is shown regarding the line colors

`legend.hist.title`
: title for the histogram. By default, one title is used for both the histogram and the normal legend for line colors.

legend.col.z	index value that determines the position of the legend element regarding the line colors with respect to other legend elements. The legend elements are stacked according to their z values. The legend element with the lowest z value is placed on top.
legend.lwd.z	index value that determines the position of the legend element regarding the line widths. (See legend.col.z)
legend.hist.z	index value that determines the position of the legend element regarding the histogram. (See legend.col.z)
id	name of the data variable that specifies the indices of the lines. Only used for "view" mode (see tmap_mode).
popup.vars	names of data variables that are shown in the popups in "view" mode. If NA (default), only aesthetic variables (i.e. specified by col and lwd) are shown). If they are not specified, all variables are shown. Set popup.vars to FALSE to disable popups. When a vector of variable names is provided, the names (if specified) are printed in the popups.
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.
group	name of the group to which this layer belongs in view mode. Each group can be selected or deselected in the layer control item. Groups can either be specified as base or overlay groups in tm_view (arguments base.groups and overlay.groups).
auto.palette.mapping	deprecated. It has been replaced by midpoint for numeric variables and stretch.palette for categorical variables.
max.categories	deprecated. It has moved to tmap_options.

Details

Small multiples can be drawn in two ways: either by specifying the by argument in tm_facets, or by defining multiple variables in the aesthetic arguments. The aesthetic arguments of tm_lines are col and lwd. In the latter case, the arguments, except for the ones starting with legend., can be specified for small multiples as follows. If the argument normally only takes a single value, such as n, then a vector of those values can be specified, one for each small multiple. If the argument normally can take a vector, such as palette, then a list of those vectors (or values) can be specified, one for each small multiple.

Value

tmap-element

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

vignette("tmap-getstarted")

Examples

```
data(World, rivers)

qtm(rivers)

## Not run:
tm_shape(World) +
    tm_fill() +
tm_shape(rivers) +
    tm_lines(col="black", lwd="scalerank", scale=2, legend.lwd.show = FALSE) +
tm_style("cobalt", title = "Rivers of the World") +
tm_format("World")

## End(Not run)
```

---

tm_logo *Logo*

Description

Creates a map logo. Multiple logos can be specified which are shown next to each other. Logos placed on top of each other can be specified with stacking tm_logo elements.

Usage

```
tm_logo(file, height = 3, halign = "center", margin = 0.2,
  position = NA, just = NA)
```

Arguments

file	either a filename or url of a png image. If multiple files/urls are provided with a character vector, the logos are placed near each other. To specify logos for small multiples use a list of character values/vectors. In order to stack logos vertically, multiple tm_logo elements can be stacked.
height	height of the logo in number of text line heights. The width is scaled based the height and the aspect ratio of the logo. If multiple logos are specified by a vector or list, the heights can be specified accordingly.
halign	if logos in one row have different heights, halign specifies the vertical alignment. Possible values are "top", "center" and "bottom".
margin	margin around the logo in number of text line heights.

position — position of the logo. Vector of two values, specifying the x and y coordinates. Either this vector contains "left", "LEFT", "center", "right", or "RIGHT" for the first value and "top", "TOP", "center", "bottom", or "BOTTOM" for the second value, or this vector contains two numeric values between 0 and 1 that specifies the x and y value of the center of the text. The uppercase values correspond to the position without margins (so tighter to the frame). The default value is controlled by the argument "attr.position" of tm_layout.

just — Justification of the attribute relative to the point coordinates. The first value specifies horizontal and the second value vertical justification. Possible values are: "left" , "right", "center", "bottom", and "top". Numeric values of 0 specify left/bottom alignment and 1 right/top alignment. This option is only used, if position is specified by numeric coordinates. The default value is controlled by the argument "attr.just" of tm_layout.

Examples

```
## Not run:
data(NLD_muni, NLD_prov)

tm_shape(NLD_muni) +
tm_polygons("origin_native", border.alpha=0.5, style="cont", title="Native Dutch (%)") +
tm_logo("http://statline.cbs.nl/Statweb/Images/cbs_logo.png",
        position=c("left", "bottom"), height = 2) +
tm_layout(bg.color="gray98")

data(World)

tm_shape(World) +
tm_polygons("HPI", palette="RdYlGn") +
tm_logo(c("https://www.r-project.org/logo/Rlogo.png",
        system.file("img/tmap.png", package="tmap"))) +
tm_logo("http://blog.kulikulifoods.com/wp-content/uploads/2014/10/logo.png",
        height=5, position = c("left", "top")) +
tm_format("World")

## End(Not run)
```

---

tm_minimap *Minimap*

Description

Creates a minimap in view mode. See addMiniMap.

Usage

```
tm_minimap(server = NA, position = c("left", "bottom"),
  toggle = TRUE, ...)
```

Arguments

server	name of the provider or an URL (see tm_tiles). By default, it shows the same map as the basemap, and moreover, it will automatically change when the user switches basemaps. Note the latter does not happen when server is specified.
position	position of the scale bar Vector of two values, specifying the x and y coordinates. The first is either "left" or "right", the second either "top" or "bottom".
toggle	should the minimap have a button to minimise it? By default TRUE.
...	arguments passed on to addMiniMap.

See Also

addMiniMap

---

tm_raster *Draw a raster*

Description

Creates a tmap-element that draws a raster. For coloring, there are three options: 1) a fixed color is used, 2) a color palette is mapped to a data variable, 3) RGB values are used. The function tm_raster is designed for options 1 and 2, while tm_rgb is used for option 3.

Usage

```
tm_raster(col = NA, alpha = NA, palette = NULL, n = 5,
  style = ifelse(is.null(breaks), "pretty", "fixed"), breaks = NULL,
  interval.closure = "left", labels = NULL, midpoint = NULL,
  stretch.palette = TRUE, contrast = NA, saturation = 1,
  interpolate = NA, colorNA = NULL, textNA = "Missing",
  showNA = NA, colorNULL = NULL, title = NA, legend.show = TRUE,
  legend.format = list(), legend.is.portrait = TRUE,
  legend.reverse = FALSE, legend.hist = FALSE,
  legend.hist.title = NA, legend.z = NA, legend.hist.z = NA,
  group = NA, auto.palette.mapping = NULL, max.categories = NULL)

tm_rgb(alpha = NA, saturation = 1, interpolate = TRUE, ...)
```

Arguments

col	three options: a single color value, the name of a data variable that is contained in shp, or the name of a variable in shp that contain color values. In the second case the values (numeric or categorical) that will be depicted by a color palette (see palette. If multiple values are specified, small multiples are drawn (see details). By default, it is a vector of the names of all data variables unless the by argument of tm_facets is defined (in that case, the default color of dots is taken from the tmap option aes.color). Note that the number of small multiples is limited by tmap_options("limits")).

alpha — transparency number between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the col is used (normally 1).

palette — a palette name or a vector of colors. See tmaptools::palette_explorer() for the named palettes. Use a "-" as prefix to reverse the palette. The default palette is taken from tm_layout's argument aes.palette, which typically depends on the style. The type of palette from aes.palette is automatically determined, but can be overwritten: use "seq" for sequential, "div" for diverging, and "cat" for categorical.

n — preferred number of classes (in case col is a numeric variable)

style — method to process the color scale when col is a numeric variable. Discrete options are "cat", "fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", and "jenks". A numeric variable is processed as a categorical variable when using "cat", i.e. each unique value will correspond to a distinct category. For the other discrete options, see the details in classIntervals. Continuous options are "cont" and "order". The former maps the values of col to a smooth gradient, whereas the latter maps the order of values of col to a smooth gradient. They are the continuous variants of respectively the discrete methods "equal" and quantile".

breaks — in case style=="fixed", breaks should be specified. The breaks argument can also be used when style="cont". In that case, the breaks are mapped evenly to the sequential or diverging color palette.

interval.closure — value that determines whether where the intervals are closed: "left" or "right". Only applicable if col is a numeric variable.

labels — labels of the classes

midpoint — The value mapped to the middle color of a diverging palette. By default it is set to 0 if negative and positive values are present. In that case, the two sides of the color palette are assigned to negative respectively positive values. If all values are positive or all values are negative, then the midpoint is set to NA, which means that the value that corresponds to the middle color class (see style) is mapped to the middle color. Only applies when col is a numeric variable. If it is specified for sequential color palettes (e.g. "Blues"), then this color palette will be treated as a diverging color palette.

stretch.palette — Logical that determines whether the categorical color palette should be stretched if there are more categories than colors. If TRUE (default), interpolated colors are used (like a rainbow). If FALSE, the palette is repeated.

contrast — vector of two numbers that determine the range that is used for sequential and diverging palettes (applicable when auto.palette.mapping=TRUE). Both numbers should be between 0 and 1. The first number determines where the palette begins, and the second number where it ends. For sequential palettes, 0 means the brightest color, and 1 the darkest color. For diverging palettes, 0 means the middle color, and 1 both extremes. If only one number is provided, this number is interpreted as the endpoint (with 0 taken as the start).

saturation — Number that determines how much saturation (also known as chroma) is used: saturation=0 is greyscale and saturation=1 is normal. This saturation value is multiplied by the overall saturation of the map (see tm_layout).

`interpolate` Should the raster image be interpolated? By default `FALSE` for `tm_raster` and `TRUE` for `tm_rgb`.

`colorNA` color used for missing values. Use `NULL` for transparency.

`textNA` text used for missing values.

`showNA` logical that determines whether missing values are named in the legend. By default (`NA`), this depends on the presence of missing values.

`colorNULL` colour for polygons that are shown on the map that are out of scope

`title` title of the legend element

`legend.show` logical that determines whether the legend is shown

`legend.format` list of formatting options for the legend numbers. Only applicable if `labels` is undefined. Parameters are:

fun Function to specify the labels. It should take a numeric vector, and should return a character vector of the same size. By default it is not specified. If specified, the list items `scientific`, `format`, and `digits` (see below) are not used.

scientific Should the labels be formatted scientifically? If so, square brackets are used, and the `format` of the numbers is `"g"`. Otherwise, `format="f"`, and `text.separator`, `text.less.than`, and `text.or.more` are used. Also, the numbers are automatically rounded to millions or billions if applicable.

format By default, `"f"`, i.e. the standard notation `xxx.xxx`, is used. If `scientific=TRUE` then `"g"`, which means that numbers are formatted scientifically, i.e. `n.dddE+nn` if needed to save space.

digits Number of digits after the decimal point if `format="f"`, and the number of significant digits otherwise.

text.separator Character string to use to separate numbers in the legend (default: "to").

text.less.than Character value(s) to use to translate "Less than". When a character vector of length 2 is specified, one for each word, these words are aligned when `text.to.columns = TRUE`

text.or.more Character value(s) to use to translate "or more". When a character vector of length 2 is specified, one for each word, these words are aligned when `text.to.columns = TRUE`

text.align Value that determines how the numbers are aligned, `"left"`, `"center"` or `"right"`. By default `"left"` for legends in portrait format (`legend.is.protrait = TRUE`), and `"center"` otherwise.

text.to.columns Logical that determines whether the text is aligned to three columns (from, text.separator, to). By default `FALSE`.

... Other arguments passed on to `formatC`

`legend.is.portrait` logical that determines whether the legend is in portrait mode (`TRUE`) or landscape (`FALSE`)

`legend.reverse` logical that determines whether the items of the legend regarding the text sizes are shown in reverse order, i.e. from bottom to top when `legend.is.portrait = TRUE` and from right to left when `legend.is.portrait = FALSE`

legend.hist	logical that determines whether a histogram is shown
legend.hist.title	title for the histogram. By default, one title is used for both the histogram and the normal legend.
legend.z	index value that determines the position of the legend element with respect to other legend elements. The legend elements are stacked according to their z values. The legend element with the lowest z value is placed on top.
legend.hist.z	index value that determines the position of the histogram legend element
group	name of the group to which this layer belongs in view mode. Each group can be selected or deselected in the layer control item. Groups can either be specified as base or overlay groups in tm_view (arguments base.groups and overlay.groups).
auto.palette.mapping	deprecated. It has been replaced by midpoint for numeric variables and stretch.palette for categorical variables.
max.categories	deprecated. It has moved to tmap_options.
...	arguments passed on from tm_raster to tm_rgb

Details

Small multiples can be drawn in two ways: either by specifying the by argument in tm_facets, or by defining multiple variables in the aesthetic arguments. The aesthetic argument of tm_raster is col. In the latter case, the arguments, except for the ones starting with legend., can be specified for small multiples as follows. If the argument normally only takes a single value, such as n, then a vector of those values can be specified, one for each small multiple. If the argument normally can take a vector, such as palette, then a list of those vectors (or values) can be specified, one for each small multiple.

Value

tmap-element

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

vignette("tmap-getstarted")

Examples

```
data(World, land, metro)

pal8 <- c("#33A02C", "#B2DF8A", "#FDBF6F", "#1F78B4", "#999999", "#E31A1C", "#E6E6E6", "#A6CEE3")
tm_shape(land, ylim = c(-88,88)) +
    tm_raster("cover_cls", palette = pal8, title = "Global Land Cover") +
tm_shape(metro) + tm_dots(col = "#E31A1C") +
```

```
tm_shape(World) +
    tm_borders(col = "black") +
tm_layout(scale = .8,
legend.position = c("left","bottom"),
    legend.bg.color = "white", legend.bg.alpha = .2,
    legend.frame = "gray50")

## Not run:
pal20 <- c("#003200", "#3C9600", "#006E00", "#556E19", "#00C800", "#8CBE8C",
   "#467864", "#B4E664", "#9BC832", "#EBFF64", "#F06432", "#9132E6",
   "#E664E6", "#9B82E6", "#B4FEF0", "#646464", "#C8C8C8", "#FF0000",
   "#FFFFFF", "#5ADCDC")
tm_shape(land) +
tm_raster("cover", palette = pal20, title = "Global Land Cover") +
tm_layout(scale=.8, legend.position = c("left","bottom"))

## End(Not run)


tm_shape(land, ylim = c(-88,88)) +
    tm_raster("trees", palette = "Greens", title = "Percent Tree Cover") +
tm_shape(World) +
    tm_borders() +
tm_layout(legend.position = c("left", "bottom"), bg.color = "lightblue")

## Not run:
tm_shape(land) +
tm_raster("black") +
tm_facets(by="cover_cls")

## End(Not run)

# TIP: check out these examples in view mode, enabled with tmap_mode("view")
```

`tm_scale_bar` *Scale bar*

Description

Creates a scale bar. By default, the coordinate units are assumed to be meters, and the map units in kilometers. This can be changed in `tm_shape`.

Usage

```
tm_scale_bar(breaks = NULL, width = NA, size = 0.5,
  text.color = NA, color.dark = "black", color.light = "white",
  lwd = 1, position = NA, just = NA)
```

Arguments

breaks	breaks of the scale bar. If not specified, breaks will be automatically be chosen given the prefered width of the scale bar. Not available for view mode.
width	(preferred) width of the scale bar. Only applicable when breaks=NULL. In plot mode, it corresponds the relative width; the default is 0.25 so one fourth of the map width. In view mode, it corresponds to the width in pixels; the default is 100.
size	relative text size (which is upperbound by the available label width)
text.color	color of the text. By default equal to the argument attr.color of tm_layout.
color.dark	color of the dark parts of the scale bar, typically (and by default) black.
color.light	color of the light parts of the scale bar, typically (and by default) white.
lwd	line width of the scale bar
position	position of the scale bar Vector of two values, specifying the x and y coordinates. Either this vector contains "left", "LEFT", "center", "right", or "RIGHT" for the first value and "top", "TOP", "center", "bottom", or "BOTTOM" for the second value, or this vector contains two numeric values between 0 and 1 that specifies the x and y value of the left bottom corner of the scale bar. The uppercase values correspond to the position without margins (so tighter to the frame). The default value is controlled by the argument "attr.position" of tm_layout.
just	Justification of the attribute relative to the point coordinates. The first value specifies horizontal and the second value vertical justification. Possible values are: "left" , "right", "center", "bottom", and "top". Numeric values of 0 specify left/bottom alignment and 1 right/top alignment. This option is only used, if position is specified by numeric coordinates. The default value is controlled by the argument "attr.just" of tm_layout.

Examples

```
current.mode <- tmap_mode("plot")

data(NLD_muni)
qtm(NLD_muni, theme = "NLD") + tm_scale_bar(position=c("left", "bottom"))

# restore current mode
tmap_mode(current.mode)
```

---

tm_sf *Draw simple features*

Description

Creates a tmap-element that draws simple features. Basically, it is a stack of tm_polygons, tm_lines and tm_dots. In other words, polygons are plotted as polygons, lines as lines and points as dots.

Usage

```
tm_sf(col = NA, size = 0.02, shape = 16, lwd = 1, lty = "solid",
  alpha = NA, palette = NULL, border.col = NA, border.lwd = 1,
  border.lty = "solid", border.alpha = NA, group = NA, ...)
```

Arguments

col	color of the simple features. See the col argument of tm_polygons, tm_lines and tm_symbols.
size	size of the dots. See the size argument tm_symbols. By default, the size is similar to dot size (see tm_dots)
shape	shape of the dots. See the shape argument tm_symbols. By default, dots are shown.
lwd	width of the lines. See the lwd argument of tm_lines
lty	type of the lines. See the lty argument of tm_lines
alpha	transparency number. See alpha argument of tm_polygons, tm_lines and tm_symbols
palette	palette. See palette argument of tm_polygons, tm_lines and tm_symbols
border.col	color of the borders. See border.col argument of tm_polygons and tm_symbols.
border.lwd	line width of the borders. See border.lwd argument of tm_polygons and tm_symbols.
border.lty	line type of the borders. See border.lwd argument of tm_polygons and tm_symbols.
border.alpha	transparency of the borders. See border.alpha argument of tm_polygons and tm_symbols.
group	name of the group to which this layer belongs in view mode. Each group can be selected or deselected in the layer control item. Groups can either be specified as base or overlay groups in tm_view (arguments base.groups and overlay.groups).
...	other arguments passed on to tm_polygons, tm_lines and tm_symbols

Value

tmap-element

See Also

vignette("tmap-getstarted")

Examples

```
data(World)

World$geometry[World$continent == "Africa"] <-
  sf::st_centroid(World$geometry[World$continent == "Africa"])
World$geometry[World$continent == "South America"] <-
  sf::st_cast(World$geometry[World$continent == "South America"],
```

```
    "MULTILINESTRING", group_or_split = FALSE)

tm_shape(World) +
tm_sf()
```

---

tm_shape *Specify the shape object*

Description

Creates a tmap-element that specifies a spatial data object, which we refer to as shape. Also the projection and covered area (bounding box) can be set. It is possible to use multiple shape objects within one plot (see tmap-element).

Usage

```
tm_shape(shp, name = NULL, is.master = NA, projection = NULL,
  bbox = NULL, unit = NULL, simplify = 1, point.per = NA,
  line.center = "midpoint", filter = NULL, ...)
```

Arguments

shp	shape object, which is an object from a class defined by the sf, sp, or raster package. For instance, an sf object, an SpatialPolygons(DataFrame), or a RasterBrick.
name	name of the shape object (character) as it appears in the legend in "view" mode. Default value is the name of shp.
is.master	logical that determines whether this tm_shape is the master shape element. The bounding box, projection settings, and the unit specifications of the resulting thematic map are taken from the tm_shape element of the master shape object. By default, the first master shape element with a raster shape is the master, and if there are no raster shapes used, then the first tm_shape is the master shape element.
projection	Either a crs object or a character value. If it is a character, it can either be a PROJ.4 character string or a shortcut. See get_proj4 for a list of shortcut values. By default, the projection is used that is defined in the shp object itself, which can be obtained with get_projection.
bbox	bounding box. One of the following: • A bounding box (an sf bbox object, see st_bbox, a 2 by 2 matrix (used by the sp package), or an Extent object used by the raster package). • Open Street Map search query. The bounding is automatically generated by querying q from Open Street Map Nominatim. See http://wiki.openstreetmap.org/wiki/Nominatim. • Another shape object, from which the bounding box is extracted.

	If unspecified, the current bounding box of shp is taken. The bounding box is feed to bb (as argument x. The other arguments of bb can be specified directly as well (see . .).
unit	desired units of the map. One of "metric" (default), "imperial", "km", "m", "mi" and "ft". Used to specify the scale bar (see tm_scale_bar) and to calculate densities for choropleths (see argument convert2density in tm_fill).
simplify	simplification factor for spatial polygons and spatial lines. A number between 0 and 1 that indicates how many coordinates are kept. See the underlying function simplify_shape, from which the arguments keep.units and keep.subunits can be passed on (see . . .). This requires the suggested package rmapshaper.
point.per	specification of how points or text labels are plotted when the geometry is a multi line or a multi polygon. One of "feature", "segment" or "largest". The first generates a point/label 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. By default, it is set to "segment" if the geometry of shp is a (multi)points geometry or a geometrycollection, and "feature" otherwise.
line.center	specification of where points are placed for (multi)line geometries. Either "midpoint" or "centroid". The former places a point at the middle of the line, the latter at the controid.
filter	logical vector which indicated per feature whether it should be included. Features for which filter is FALSE will be colored light gray (see the colorNULL argument in the layer functions)
...	Arguments passed on to bb (e.g. ext can be used to enlarge or shrinke a bounding box), and simplify_shape (the arguments keep.units and keep.subunits)

Value

tmap-element

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

read_shape to read ESRI shape files, set_projection, vignette("tmap-getstarted")

Examples

```
current.mode <- tmap_mode("plot")

data(World, metro, rivers)

tm_shape(World, projection="longlat") +
    tm_polygons() +
tm_layout("Long lat coordinates (WGS84)", inner.margins=c(0,0,.1,0), title.size=.8)
```

```
World$highlighted <- ifelse(World$iso_a3 %in% c("GRL", "AUS"), "gold", "gray75")
tm_shape(World, projection="merc", ylim=c(.1, 1), relative = TRUE) +
    tm_polygons("highlighted") +
tm_layout("Web Mercator projection. Although widely used, it is discouraged for
statistical purposes. In reality, Australia is 3 times larger than Greenland!",
    inner.margins=c(0,0,.1,0), title.size=.6)

tm_shape(World, projection="robin") +
    tm_polygons() +
tm_layout(
"Winkel-Tripel projection, adapted as default by the National Geographic Society for world maps.",
    inner.margins=c(0,0,.1,0), title.size=.8)

tm_shape(World) +
    tm_polygons() +
tm_layout("Eckhart IV projection. Recommended in statistical maps for its equal-area property.",
    inner.margins=c(0,0,.1,0), title.size=.8)


# different levels of simplification
## Not run:
tm1 <- tm_shape(World, simplify = 0.05) + tm_polygons() + tm_layout("Simplification: 0.05")
tm2 <- tm_shape(World, simplify = 0.1) + tm_polygons() + tm_layout("Simplification: 0.1")
tm3 <- tm_shape(World, simplify = 0.25) + tm_polygons() + tm_layout("Simplification: 0.25")
tm4 <- tm_shape(World, simplify = 0.5) + tm_polygons() + tm_layout("Simplification: 0.5")

require(tmaptools)
tmap_arrange(tm1, tm2, tm3, tm4)

## End(Not run)

# three groups of layers, each starting with tm_shape
## Not run:
tm_shape(World) +
    tm_fill("darkolivegreen3") +
tm_shape(metro) +
    tm_bubbles("pop2010", col = "grey30", scale=.5) +
tm_shape(rivers) +
    tm_lines("lightcyan1") +
tm_layout(bg.color="lightcyan1", inner.margins=c(0,0,.02,0), legend.show = FALSE)

## End(Not run)

# restore current mode
tmap_mode(current.mode)
```

---

tm_symbols *Draw symbols*

Description

Creates a tmap-element that draws symbols, including symbols and dots. The color, size, and shape of the symbols can be mapped to data variables.

Usage

```
tm_symbols(size = 1, col = NA, shape = 21, alpha = NA,
  border.col = NA, border.lwd = 1, border.alpha = NA, scale = 1,
  perceptual = FALSE, clustering = FALSE, size.max = NA,
  size.lim = NA, sizes.legend = NULL, sizes.legend.labels = NULL,
  n = 5, style = ifelse(is.null(breaks), "pretty", "fixed"),
  breaks = NULL, interval.closure = "left", palette = NULL,
  labels = NULL, midpoint = NULL, stretch.palette = TRUE,
  contrast = NA, colorNA = NA, textNA = "Missing", showNA = NA,
  colorNULL = NA, shapes = 21:25, shapes.legend = NULL,
  shapes.legend.fill = NA, shapes.labels = NULL, shapeNA = 4,
  shape.textNA = "Missing", shape.showNA = NA, shapes.n = 5,
  shapes.style = ifelse(is.null(shapes.breaks), "pretty", "fixed"),
  shapes.breaks = NULL, shapes.interval.closure = "left",
  legend.max.symbol.size = 0.8, just = NA, jitter = 0, xmod = 0,
  ymod = 0, icon.scale = 3, grob.dim = c(width = 48, height = 48,
  render.width = 256, render.height = 256), title.size = NA,
  title.col = NA, title.shape = NA, legend.size.show = TRUE,
  legend.col.show = TRUE, legend.shape.show = TRUE,
  legend.format = list(), legend.size.is.portrait = FALSE,
  legend.col.is.portrait = TRUE, legend.shape.is.portrait = TRUE,
  legend.size.reverse = FALSE, legend.col.reverse = FALSE,
  legend.shape.reverse = FALSE, legend.hist = FALSE,
  legend.hist.title = NA, legend.size.z = NA, legend.col.z = NA,
  legend.shape.z = NA, legend.hist.z = NA, id = NA,
  popup.vars = NA, popup.format = list(), group = NA,
  auto.palette.mapping = NULL, max.categories = NULL)

tm_squares(size = 1, col = NA, shape = 22, scale = 4/3, ...)

tm_bubbles(size = 1, col = NA, shape = 21, scale = 4/3,
  legend.max.symbol.size = 1, ...)

tm_dots(col = NA, size = 0.02, shape = 16, title = NA,
  legend.show = TRUE, legend.is.portrait = TRUE, legend.z = NA, ...)

tm_markers(shape = marker_icon(), col = NA, border.col = NULL,
  clustering = TRUE, text = NULL, text.just = "top",
  markers.on.top.of.text = TRUE, group = NA, ...)
```

Arguments

size	a single value or a shp data variable that determines the symbol sizes. The reference value size=1 corresponds to the area of symbols that have the same height as one line of text. If a data variable is provided, the symbol sizes are scaled proportionally (or perceptually, see perceptual) where by default the symbol with the largest data value will get size=1 (see also size.max). If multiple values are specified, small multiples are drawn (see details).
col	color(s) of the symbol. Either a color (vector), or categorical variable name(s). If multiple values are specified, small multiples are drawn (see details).
shape	shape(s) of the symbol. Either direct shape specification(s) or a data variable name(s) that is mapped to the symbols specified by the shapes argument. See details for the shape specification.
alpha	transparency number between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the col is used (normally 1).
border.col	color of the symbol borders.
border.lwd	line width of the symbol borders. If NA, no symbol borders are drawn.
border.alpha	transparency number, regarding the symbol borders, between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the col is used (normally 1).
scale	symbol size multiplier number.
perceptual	logical that determines whether symbols are scales with a perceptually (TRUE) or mathematically (FALSE, default value). The perceived area of larger symbols is often underestimated. Flannery (1971) experimentally derived a method to compensate this for symbols, which is enabled by this argument.
clustering	value that determines whether the symbols are clustered in "view" mode. It does not work proportional bubbles (i.e. tm_bubbles). One of: TRUE, FALSE, or the output of markerClusterOptions.
size.max	value that is mapped to size=1. By default (NA), the maximum data value is chosen. Only applicable when size is the name of a numeric variable of shp
size.lim	vector of two limit values of the size variable. Only symbols are drawn whose value is greater than or equal to the first value. Symbols whose values exceed the second value are drawn at the size of the second value. Only applicable when size is the name of a numeric variable of shp
sizes.legend	vector of symbol sizes that are shown in the legend. By default, this is determined automatically.
sizes.legend.labels	vector of labels for that correspond to sizes.legend.
n	preferred number of color scale classes. Only applicable when col is a numeric variable name.
style	method to process the color scale when col is a numeric variable. Discrete options are "cat", "fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", and "jenks". A numeric variable is processed as a categorical variable when using "cat", i.e. each unique value will correspond to a distinct category. For the other discrete options, see the details in

classIntervals. Continuous options are "cont" and "order". The former maps the values of col to a smooth gradient, whereas the latter maps the order of values of col to a smooth gradient. They are the continuous variants of respectively the discrete methods "equal" and quantile".

breaks in case style=="fixed", breaks should be specified. The breaks argument can also be used when style="cont". In that case, the breaks are mapped evenly to the sequential or diverging color palette.

interval.closure
value that determines whether where the intervals are closed: "left" or "right". Only applicable if col is a numeric variable.

palette a palette name or a vector of colors. See tmaptools::palette_explorer() for the named palettes. Use a "-" as prefix to reverse the palette. The default palette is taken from tm_layout's argument aes.palette, which typically depends on the style. The type of palette from aes.palette is automatically determined, but can be overwritten: use "seq" for sequential, "div" for diverging, and "cat" for categorical.

labels labels of the classes

midpoint The value mapped to the middle color of a diverging palette. By default it is set to 0 if negative and positive values are present. In that case, the two sides of the color palette are assigned to negative respectively positive values. If all values are positive or all values are negative, then the midpoint is set to NA, which means that the value that corresponds to the middle color class (see style) is mapped to the middle color. Only applies when col is a numeric variable. If it is specified for sequential color palettes (e.g. "Blues"), then this color palette will be treated as a diverging color palette.

stretch.palette
Logical that determines whether the categorical color palette should be stretched if there are more categories than colors. If TRUE (default), interpolated colors are used (like a rainbow). If FALSE, the palette is repeated.

contrast vector of two numbers that determine the range that is used for sequential and diverging palettes (applicable when auto.palette.mapping=TRUE). Both numbers should be between 0 and 1. The first number determines where the palette begins, and the second number where it ends. For sequential palettes, 0 means the brightest color, and 1 the darkest color. For diverging palettes, 0 means the middle color, and 1 both extremes. If only one number is provided, this number is interpreted as the endpoint (with 0 taken as the start).

colorNA colour for missing values. Use NULL for transparency.

textNA text used for missing values of the color variable.

showNA logical that determines whether missing values are named in the legend. By default (NA), this depends on the presence of missing values.

colorNULL colour for polygons that are shown on the map that are out of scope

shapes palette of symbol shapes. Only applicable if shape is a (vector of) categorical variable(s). See details for the shape specification. By default, the filled symbols 21 to 25 are taken.

shapes.legend — symbol shapes that are used in the legend (instead of the symbols specified with shape. Especially useful when shapes consist of grobs that have to be represented by neutrally colored shapes (see also shapes.legend.fill.

shapes.legend.fill — Fill color of legend shapes (see shapes.legend)

shapes.labels — Legend labels for the symbol shapes

shapeNA — the shape (a number or grob) for missing values. By default a cross (number 4). Set to NA to hide symbols for missing values.

shape.textNA — text used for missing values of the shape variable.

shape.showNA — logical that determines whether missing values are named in the legend. By default (NA), this depends on the presence of missing values.

shapes.n — preferred number of shape classes. Only applicable when shape is a numeric variable name.

shapes.style — method to process the shape scale when shape is a numeric variable. See style argument for options

shapes.breaks — in case shapes.style=="fixed", breaks should be specified

shapes.interval.closure — value that determines whether where the intervals are closed: "left" or "right". Only applicable if shape is a numeric variable.

legend.max.symbol.size — Maximum size of the symbols that are drawn in the legend. For circles and bubbles, a value larger than one is recommended (and used for tm_bubbles)

just — justification of the symbols relative to the point coordinates. The first value specifies horizontal and the second value vertical justification. Possible values are: "left" , "right", "center", "bottom", and "top". Numeric values of 0 specify left alignment and 1 right alignment. The default value is c("center", "center"). For icons, this value may already be speficied (see tmap_icons). The just, if specified, will overrides this.

jitter — number that determines the amount of jittering, i.e. the random noise added to the position of the symbols. 0 means no jittering is applied, any positive number means that the random noise has a standard deviation of jitter times the height of one line of text line.

xmod — horizontal position modification of the symbols, in terms of the height of one line of text. Either a single number for all polygons, or a numeric variable in the shape data specifying a number for each polygon. Together with ymod, it determines position modification of the symbols. See also jitter for random position modifications. In most coordinate systems (projections), the origin is located at the bottom left, so negative xmod move the symbols to the left, and negative ymod values to the bottom.

ymod — vertical position modification. See xmod.

icon.scale — scaling number that determines how large the icons (or grobs) are in plot mode in comparison to proportional symbols (such as bubbles). In view mode, the size is determined by the icon specification (see tmap_icons) or, if grobs are specified by grob.width and grob.heigth

`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 ggplot2 graphic successfully. Only needed for the view mode.

`title.size` title of the legend element regarding the symbol sizes

`title.col` title of the legend element regarding the symbol colors

`title.shape` title of the legend element regarding the symbol shapes

`legend.size.show` logical that determines whether the legend for the symbol sizes is shown

`legend.col.show` logical that determines whether the legend for the symbol colors is shown

`legend.shape.show` logical that determines whether the legend for the symbol shapes is shown

`legend.format` list of formatting options for the legend numbers. Only applicable if `labels` is undefined. Parameters are:

fun Function to specify the labels. It should take a numeric vector, and should return a character vector of the same size. By default it is not specified. If specified, the list items `scientific`, `format`, and `digits` (see below) are not used.

scientific Should the labels be formatted scientifically? If so, square brackets are used, and the `format` of the numbers is `"g"`. Otherwise, `format="f"`, and `text.separator`, `text.less.than`, and `text.or.more` are used. Also, the numbers are automatically rounded to millions or billions if applicable.

format By default, `"f"`, i.e. the standard notation `xxx.xxx`, is used. If `scientific=TRUE` then `"g"`, which means that numbers are formatted scientifically, i.e. `n.dddE+nn` if needed to save space.

digits Number of digits after the decimal point if `format="f"`, and the number of significant digits otherwise.

text.separator Character string to use to separate numbers in the legend (default: "to").

text.less.than Character value(s) to use to translate "Less than". When a character vector of length 2 is specified, one for each word, these words are aligned when `text.to.columns = TRUE`

text.or.more Character value(s) to use to translate "or more". When a character vector of length 2 is specified, one for each word, these words are aligned when `text.to.columns = TRUE`

text.align Value that determines how the numbers are aligned, `"left"`, `"center"` or `"right"`. By default `"left"` for legends in portrait format (`legend.is.protrait = TRUE`), and `"center"` otherwise.

text.to.columns Logical that determines whether the text is aligned to three columns (from, text.separator, to). By default `FALSE`.

... Other arguments passed on to `formatC`

`legend.size.is.portrait`
: logical that determines whether the legend element regarding the symbol sizes is in portrait mode (`TRUE`) or landscape (`FALSE`)

`legend.col.is.portrait`
: logical that determines whether the legend element regarding the symbol colors is in portrait mode (`TRUE`) or landscape (`FALSE`)

`legend.shape.is.portrait`
: logical that determines whether the legend element regarding the symbol shapes is in portrait mode (`TRUE`) or landscape (`FALSE`)

`legend.size.reverse`
: logical that determines whether the items of the legend regarding the symbol sizes are shown in reverse order, i.e. from bottom to top when `legend.size.is.portrait = TRUE` and from right to left when `legend.size.is.portrait = FALSE`

`legend.col.reverse`
: logical that determines whether the items of the legend regarding the symbol colors are shown in reverse order, i.e. from bottom to top when `legend.col.is.portrait = TRUE` and from right to left when `legend.col.is.portrait = FALSE`

`legend.shape.reverse`
: logical that determines whether the items of the legend regarding the symbol shapes are shown in reverse order, i.e. from bottom to top when `legend.shape.is.portrait = TRUE` and from right to left when `legend.shape.is.portrait = FALSE`

`legend.hist`
: logical that determines whether a histogram is shown regarding the symbol colors

`legend.hist.title`
: title for the histogram. By default, one title is used for both the histogram and the normal legend for symbol colors.

`legend.size.z`
: index value that determines the position of the legend element regarding the symbol sizes with respect to other legend elements. The legend elements are stacked according to their z values. The legend element with the lowest z value is placed on top.

`legend.col.z`
: index value that determines the position of the legend element regarding the symbol colors. (See `legend.size.z`)

`legend.shape.z`
: index value that determines the position of the legend element regarding the symbol shapes. (See `legend.size.z`)

`legend.hist.z`
: index value that determines the position of the histogram legend element. (See `legend.size.z`)

`id`
: name of the data variable that specifies the indices of the symbols. Only used for `"view"` mode (see `tmap_mode`).

`popup.vars`
: names of data variables that are shown in the popups in `"view"` mode. If `NA` (default), only aesthetic variables (i.e. specified by `col` and `lwd`) are shown). If they are not specified, all variables are shown. Set popup.vars to `FALSE` to disable popups. When a vector of variable names is provided, the names (if specified) are printed in the popups.

`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.

group — name of the group to which this layer belongs in view mode. Each group can be selected or deselected in the layer control item. Groups can either be specified as base or overlay groups in tm_view (arguments base.groups and overlay.groups).

auto.palette.mapping — deprecated. It has been replaced by midpoint for numeric variables and stretch.palette for categorical variables.

max.categories — deprecated. It has moved to tmap_options.

... — arguments passed on to tm_symbols. For tm_markers, arguments can also be passed on to tm_text. In that case, they have to be prefixed with text., e.g. the col argument should be names text.col

title — shortcut for title.col for tm_dots

legend.show — shortcut for legend.col.show for tm_dots

legend.is.portrait — shortcut for legend.col.is.portrait for tm_dots

legend.z — shortcut for legend.col.z shortcut for tm_dots

text — text of the markers. Shown in plot mode, and as popup text in view mode.

text.just — justification of marker text (see just argument of tm_text). Only applicable in plot mode.

markers.on.top.of.text — For tm_markers, should the markers be drawn on top of the text labels?

Details

Small multiples can be drawn in two ways: either by specifying the by argument in tm_facets, or by defining multiple variables in the aesthetic arguments, which are size, col, and shape. In the latter case, the arguments, except for the ones starting with legend., can be specified for small multiples as follows. If the argument normally only takes a single value, such as n, then a vector of those values can be specified, one for each small multiple. If the argument normally can take a vector, such as palette, then a list of those vectors (or values) can be specified, one for each small multiple.

A shape specification is one of the following three options. To specify multiple shapes (needed for the shapes argument), a vector or list of these shape specification is required. The shape specification options can also be mixed. For the shapes argument, it is possible to use a named vector or list, where the names correspond to the value of the variable specified by the shape argument.

1. A numeric value that specifies the plotting character of the symbol. See parameter pch of points and the last example to create a plot with all options.
2. A grob object, which can be a ggplot2 plot object created with ggplotGrob. To specify multiple shapes, a list of grob objects is required. See example of a proportional symbol map with ggplot2 plots.
3. An icon specification, which can be created with tmap_icons.

For small multiples, a list of these shape specification(s) should be provided.

Value

tmap-element

References

Flannery J (1971). The Relative Effectiveness of Some Common Graduated Point Symbols in the Presentation of Quantitative Data. Canadian Cartographer, 8(2), 96-109.

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

vignette("tmap-getstarted")

Examples

```
data(World, metro)
metro$growth <- (metro$pop2020 - metro$pop2010) / (metro$pop2010 * 10) * 100

tm_shape(World) +
    tm_fill("grey70") +
tm_shape(metro) +
    tm_bubbles("pop2010", col = "growth",
        border.col = "black", border.alpha = .5,
        style="fixed", breaks=c(-Inf, seq(0, 6, by=2), Inf),
        palette="-RdYlBu", contrast=1,
        title.size="Metro population",
        title.col="Growth rate (%)") +
tm_format("World")

tm_shape(metro) +
tm_symbols(size = "pop2010", col="pop2010", shape="pop2010",
legend.format = list(text.align="right", text.to.columns = TRUE)) +
tm_legend(outside = TRUE, outside.position = "bottom", stack = "horizontal")


if (require(ggplot2) && require(dplyr) && require(tidyr) && require(tmaptools) && require(sf)) {
data(NLD_prov)

origin_data <- NLD_prov %>%
st_set_geometry(NULL) %>%
mutate(FID= factor(1:n())) %>%
select(FID, origin_native, origin_west, origin_non_west) %>%
gather(key=origin, value=perc, origin_native, origin_west, origin_non_west, factor_key=TRUE)

origin_cols <- get_brewer_pal("Dark2", 3)

grobs <- lapply(split(origin_data, origin_data$FID), function(x) {
ggplotGrob(ggplot(x, aes(x="", y=-perc, fill=origin)) +
geom_bar(width=1, stat="identity") +
scale_y_continuous(expand=c(0,0)) +
scale_fill_manual(values=origin_cols) +
```

```
theme_ps(plot.axes = FALSE))
})

names(grobs) <- NLD_prov$name

tm_shape(NLD_prov) +
tm_polygons() +
tm_symbols(size="population", shape="name",
shapes=grobs,
sizes.legend=c(.5, 1,3)*1e6,
scale=1,
legend.shape.show = FALSE,
legend.size.is.portrait = TRUE,
shapes.legend = 22,
title.size = "Population",
id = "name",
popup.vars = c("population", "origin_native",
   "origin_west", "origin_non_west")) +
tm_add_legend(type="fill",
col=origin_cols,
labels=c("Native", "Western", "Non-western"),
title="Origin") +
tm_format("NLD")
}



# TIP: check out these examples in view mode, enabled with tmap_mode("view")

## Not run:
if (require(rnaturalearth)) {

airports <- ne_download(scale=10, type="airports")
airplane <- tmap_icons(system.file("img/airplane.png", package = "tmap"))

current.mode <- tmap_mode("view")
tm_shape(airports) +
tm_symbols(shape=airplane, size="natlscale",
      legend.size.show = FALSE, scale=1, border.col = NULL, id="name", popup.vars = TRUE) +
tm_view(set.view = c(lon = 15, lat = 48, zoom = 4))
tmap_mode(current.mode)
}

## End(Not run)

##############################################################################################

## Not run:
# plot all available symbol shapes:
if (require(ggplot2)) {
ggplot(data.frame(p=c(0:25,32:127))) +
geom_point(aes(x=p%%16, y=-(p%/%16), shape=p), size=5, fill="red") +
geom_text(mapping=aes(x=p%%16, y=-(p%/%16+0.25), label=p), size=3) +
```

```
scale_shape_identity() +
theme(axis.title=element_blank(),
  axis.text=element_blank(),
  axis.ticks=element_blank(),
  panel.background=element_blank())
}

## End(Not run)
```

---

tm_text *Add text labels*

Description

Creates a tmap-element that adds text labels.

Usage

```
tm_text(text, size = 1, col = NA, root = 3, clustering = FALSE,
  size.lim = NA, sizes.legend = NULL, sizes.legend.labels = NULL,
  sizes.legend.text = "Abc", n = 5, style = ifelse(is.null(breaks),
  "pretty", "fixed"), breaks = NULL, interval.closure = "left",
  palette = NULL, labels = NULL, labels.text = NA, midpoint = NULL,
  stretch.palette = TRUE, contrast = NA, colorNA = NA,
  textNA = "Missing", showNA = NA, colorNULL = NA, fontface = NA,
  fontfamily = NA, alpha = NA, case = NA, shadow = FALSE,
  bg.color = NA, bg.alpha = NA, size.lowerbound = 0.4,
  print.tiny = FALSE, scale = 1, auto.placement = FALSE,
  remove.overlap = FALSE, along.lines = FALSE,
  overwrite.lines = FALSE, just = "center", xmod = 0, ymod = 0,
  title.size = NA, title.col = NA, legend.size.show = TRUE,
  legend.col.show = TRUE, legend.format = list(),
  legend.size.is.portrait = FALSE, legend.col.is.portrait = TRUE,
  legend.size.reverse = FALSE, legend.col.reverse = FALSE,
  legend.hist = FALSE, legend.hist.title = NA, legend.size.z = NA,
  legend.col.z = NA, legend.hist.z = NA, group = NA,
  auto.palette.mapping = NULL, max.categories = NULL)
```

Arguments

text	name of the variable in the shape object that contains the text labels
size	relative size of the text labels (see note). Either one number, a name of a numeric variable in the shape data that is used to scale the sizes proportionally, or the value "AREA", where the text size is proportional to the area size of the polygons.
col	color of the text labels. Either a color value or a data variable name. If multiple values are specified, small multiples are drawn (see details).

root — root number to which the font sizes are scaled. Only applicable if size is a variable name or "AREA". If root=2, the square root is taken, if root=3, the cube root etc.

clustering — value that determines whether the text labels are clustered in "view" mode. One of: TRUE, FALSE, or the output of markerClusterOptions.

size.lim — vector of two limit values of the size variable. Only text labels are drawn whose value is greater than or equal to the first value. Text labels whose values exceed the second value are drawn at the size of the second value. Only applicable when size is the name of a numeric variable of shp. See also size.lowerbound which is a threshold of the relative font size.

sizes.legend — vector of text sizes that are shown in the legend. By default, this is determined automatically.

sizes.legend.labels — vector of labels for that correspond to sizes.legend.

sizes.legend.text — vector of example text to show in the legend next to sizes.legend.labels. By default "Abc". When NA, examples from the data variable whose sizes are close to the sizes.legend are taken and "NA" for classes where no match is found.

n — preferred number of color scale classes. Only applicable when col is a numeric variable name.

style — method to process the color scale when col is a numeric variable. Discrete options are "cat", "fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", and "jenks". A numeric variable is processed as a categorical variable when using "cat", i.e. each unique value will correspond to a distinct category. For the other discrete options, see the details in classIntervals. Continuous options are "cont" and "order". The former maps the values of col to a smooth gradient, whereas the latter maps the order of values of col to a smooth gradient. They are the continuous variants of respectively the discrete methods "equal" and quantile".

breaks — in case style=="fixed", breaks should be specified. The breaks argument can also be used when style="cont". In that case, the breaks are mapped evenly to the sequential or diverging color palette.

interval.closure — value that determines whether where the intervals are closed: "left" or "right". Only applicable if col is a numeric variable.

palette — a palette name or a vector of colors. See tmaptools::palette_explorer() for the named palettes. Use a "-" as prefix to reverse the palette. The default palette is taken from tm_layout's argument aes.palette, which typically depends on the style. The type of palette from aes.palette is automatically determined, but can be overwritten: use "seq" for sequential, "div" for diverging, and "cat" for categorical.

labels — labels of the color classes, applicable if col is a data variable name

labels.text — Example text to show in the legend next to the labels. When NA (default), examples from the data variable are taken and "NA" for classes where they don't exist.

midpoint — The value mapped to the middle color of a diverging palette. By default it is set to 0 if negative and positive values are present. In that case, the two sides of the color palette are assigned to negative respectively positive values. If all values are positive or all values are negative, then the midpoint is set to NA, which means that the value that corresponds to the middle color class (see style) is mapped to the middle color. Only applies when col is a numeric variable. If it is specified for sequential color palettes (e.g. "Blues"), then this color palette will be treated as a diverging color palette.

stretch.palette — Logical that determines whether the categorical color palette should be stretched if there are more categories than colors. If TRUE (default), interpolated colors are used (like a rainbow). If FALSE, the palette is repeated.

contrast — vector of two numbers that determine the range that is used for sequential and diverging palettes (applicable when auto.palette.mapping=TRUE). Both numbers should be between 0 and 1. The first number determines where the palette begins, and the second number where it ends. For sequential palettes, 0 means the brightest color, and 1 the darkest color. For diverging palettes, 0 means the middle color, and 1 both extremes. If only one number is provided, this number is interpreted as the endpoint (with 0 taken as the start).

colorNA — colour for missing values. Use NULL for transparency.

textNA — text used for missing values.

showNA — logical that determines whether missing values are named in the legend. By default (NA), this depends on the presence of missing values.

colorNULL — colour for polygons that are shown on the map that are out of scope

fontface — font face of the text labels. By default, determined by the fontface argument of tm_layout.

fontfamily — font family of the text labels. By default, determined by the fontfamily argument of tm_layout.

alpha — transparency number between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the fontcolor is used (normally 1).

case — case of the font. Use "upper" to generate upper-case text, "lower" to generate lower-case text, and NA to leave the text as is.

shadow — logical that determines whether a shadow is depicted behind the text. The color of the shadow is either white or yellow, depending of the fontcolor.

bg.color — background color of the text labels. By default, bg.color=NA, so no background is drawn.

bg.alpha — number between 0 and 1 that specifies the transparency of the text background (0 is totally transparent, 1 is solid background).

size.lowerbound — lowerbound for size. Only applicable when size is not a constant. If print.tiny is TRUE, then all text labels which relative text is smaller than size.lowerbound are depicted at relative size size.lowerbound. If print.tiny is FALSE, then text labels are only depicted if their relative sizes are at least size.lowerbound (in other words, tiny labels are omitted).

`print.tiny` boolean, see `size.lowerbound`

`scale` text size multiplier, useful in case `size` is variable or `"AREA"`.

`auto.placement` logical (or numeric) that determines whether the labels are placed automatically. If `TRUE`, the labels are placed next to the coordinate points with as little overlap as possible using the simulated annealing algorithm. Therefore, it is recommended for labeling spatial dots or symbols. If a numeric value is provided, this value acts as a parameter that specifies the distance between the coordinate points and the text labels in terms of text line heights.

`remove.overlap` logical that determines whether the overlapping labels are removed

`along.lines` logical that determines whether labels are rotated along the spatial lines. Only applicable if a spatial lines shape is used.

`overwrite.lines`
logical that determines whether the part of the lines below the text labels is removed. Only applicable if a spatial lines shape is used.

`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.

`xmod` horizontal position modification of the text (relatively): 0 means no modification, and 1 corresponds to the height of one line of text. Either a single number for all polygons, or a numeric variable in the shape data specifying a number for each polygon. Together with `ymod`, it determines position modification of the text labels. In most coordinate systems (projections), the origin is located at the bottom left, so negative `xmod` move the text to the left, and negative `ymod` values to the bottom.

`ymod` vertical position modification. See xmod.

`title.size` title of the legend element regarding the text sizes

`title.col` title of the legend element regarding the text colors

`legend.size.show`
logical that determines whether the legend for the text sizes is shown

`legend.col.show`
logical that determines whether the legend for the text colors is shown

`legend.format` list of formatting options for the legend numbers. Only applicable if `labels` is undefined. Parameters are:

fun Function to specify the labels. It should take a numeric vector, and should return a character vector of the same size. By default it is not specified. If specified, the list items `scientific`, `format`, and `digits` (see below) are not used.

scientific Should the labels be formatted scientifically? If so, square brackets are used, and the `format` of the numbers is `"g"`. Otherwise, `format="f"`, and `text.separator`, `text.less.than`, and `text.or.more` are used. Also, the numbers are automatically rounded to millions or billions if applicable.

format By default, "f", i.e. the standard notation xxx.xxx, is used. If scientific=TRUE then "g", which means that numbers are formatted scientifically, i.e. n.dddE+nn if needed to save space.

digits Number of digits after the decimal point if format="f", and the number of significant digits otherwise.

text.separator Character string to use to separate numbers in the legend (default: "to").

text.less.than Character value(s) to use to translate "Less than". When a character vector of length 2 is specified, one for each word, these words are aligned when text.to.columns = TRUE

text.or.more Character value(s) to use to translate "or more". When a character vector of length 2 is specified, one for each word, these words are aligned when text.to.columns = TRUE

text.align Value that determines how the numbers are aligned, "left", "center" or "right". By default "left" for legends in portrait format (legend.is.portrait = TRUE), and "center" otherwise.

text.to.columns Logical that determines whether the text is aligned to three columns (from, text.separator, to). By default FALSE.

... Other arguments passed on to formatC

legend.size.is.portrait
: logical that determines whether the legend element regarding the text sizes is in portrait mode (TRUE) or landscape (FALSE)

legend.col.is.portrait
: logical that determines whether the legend element regarding the text colors is in portrait mode (TRUE) or landscape (FALSE)

legend.size.reverse
: logical that determines whether the items of the legend regarding the text sizes are shown in reverse order, i.e. from bottom to top when legend.size.is.portrait = TRUE and from right to left when legend.size.is.portrait = FALSE

legend.col.reverse
: logical that determines whether the items of the legend regarding the text colors are shown in reverse order, i.e. from bottom to top when legend.col.is.portrait = TRUE and from right to left when legend.col.is.portrait = FALSE

legend.hist
: logical that determines whether a histogram is shown regarding the text colors

legend.hist.title
: title for the histogram. By default, one title is used for both the histogram and the normal legend for text colors.

legend.size.z
: index value that determines the position of the legend element regarding the text sizes with respect to other legend elements. The legend elements are stacked according to their z values. The legend element with the lowest z value is placed on top.

legend.col.z
: index value that determines the position of the legend element regarding the text colors. (See legend.size.z)

legend.hist.z
: index value that determines the position of the histogram legend element. (See legend.size.z)

group — name of the group to which this layer belongs in view mode. Each group can be selected or deselected in the layer control item. Groups can either be specified as base or overlay groups in tm_view (arguments base.groups and overlay.groups).

auto.palette.mapping — deprecated. It has been replaced by midpoint for numeric variables and stretch.palette for categorical variables.

max.categories — deprecated. It has moved to tmap_options.

Value

tmap-element

Note

The absolute fontsize (in points) is determined by the (ROOT) viewport, which may depend on the graphics device.

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

vignette("tmap-getstarted")

Examples

```
current.mode <- tmap_mode("plot")

data(World, metro)

tm_shape(World) +
    tm_text("name", size="AREA")

tm_shape(World) +
tm_text("name", size="pop_est", col="continent", palette="Dark2",
title.size = "Population", title.col="Continent") +
tm_legend(outside = TRUE)

tmap_mode("view")

## Not run:
require(tmaptools)
metro_aus <- crop_shape(metro, bb("Australia"))

tm_shape(metro_aus) +
tm_dots() +
tm_text("name", just = "top")

# alternative
```

```
tm_shape(metro_aus) +
tm_markers(text = "name")

## End(Not run)

# restore current mode
tmap_mode(current.mode)
```

`tm_view` *Options for the interactive tmap viewer*

Description

Set the options for the interactive tmap viewer. Some of these options can also be set with `tm_layout`, since they are style dependent (e.g., the choice of basemaps). The function `tm_view` overrides these options when specified.

Usage

```
tm_view(alpha, colorNA, projection, symbol.size.fixed, dot.size.fixed,
  text.size.variable, bbox, set.bounds, set.view, set.zoom.limits,
  view.legend.position, control.position, popup.all.data, legend.position,
  basemaps = NULL, basemaps.alpha = NULL)
```

Arguments

`alpha`	transparency (opacity) parameter applied to whole map. By default, it is set to `0.7` if basemaps are used, and `1` otherwise.
`colorNA`	default color for missing values in interactive mode. If the color of missing values is not defined in the layer functions (e.g. `tm_fill`), then the default color is taken from the `na` value of the `aes.color` argument in `tm_layout`. This `colorNA` argument (if not `NA` itself) overrides that default value. For interactive maps, it can be useful to set `colorNA` to `NULL`, which means transparent.
`projection`	projection. Either a EPSG number, or a `leaflet_crs` object created with `leafletCRS`. By default, the Web Mercator (3857) is used, since the vast majority of basemaps are rendered accordingly. Other standards are EPSG numbers 4326 (WGS84) and 3395 (Mercator). If set to 0, the projection of the master shape is used (see `tm_shape`) provided that a EPSG number can be extracted.
`symbol.size.fixed`	should symbol sizes be fixed while zooming?
`dot.size.fixed`	should dot sizes be fixed while zooming?
`text.size.variable`	should text size variables be allowed in view mode? By default `FALSE`, since in many applications, the main reason to vary text size is to prevent occlusion in plot mode, which is often not a problem in view mode due to the ability to zoom in.

bbox — bounding box. One of the following:

- A bounding box (an sf bbox object, see st_bbox, a 2 by 2 matrix (used by the sp package), or an Extent object used by the raster package).
- Open Street Map search query. The bounding is automatically generated by querying q from Open Street Map Nominatim. See http://wiki.openstreetmap.org/wiki/Nominatim.

If set, it overrides set.view and all bbox arguments of tm_shape.

set.bounds — logical that determines whether maximum bounds are set, or a numeric vector of four values that specify the lng1, lat1, lng2, and lat2 coordinates (see setMaxBounds).

set.view — numeric vector that determines the view. Either a vector of three: lng, lat, and zoom, or a single value: zoom. See setView. Only applicable if bbox is not specified

set.zoom.limits — numeric vector of two that set the minimum and maximum zoom levels (see tileOptions).

view.legend.position — Character vector of two values, specifying the position of the legend. Use "left" or "right" for the first value and "top" or "bottom" for the second value. It overrides the value of legend.position of tm_layout, unless set to NA.

control.position — Character vector of two values, specifying the position of the layer control UI. Use "left" or "right" for the first value and "top" or "bottom" for the second value.

popup.all.data — not used anymore. As of version 1.6, the popups are specified by the argument popup.vars in the layer functions tm_fill, tm_symbols, and tm_lines.

legend.position — not used anymore, renamed to view.legend.position

basemaps — Deprecated. Use tm_basemap instead, or set the default basemaps in tmap_options

basemaps.alpha — Deprecated. See basemaps

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI

See Also

vignette("tmap-getstarted")

Examples

```
# world choropleth/bubble map of the world
data(World, metro)
metro$growth <- (metro$pop2020 - metro$pop2010) / (metro$pop2010 * 10) * 100

map1 <- tm_shape(metro) +
tm_bubbles("pop2010", col = "growth",
```

```
    border.col = "black", border.alpha = .5,
    style="fixed", breaks=c(-Inf, seq(0, 6, by=2), Inf),
    palette="-RdYlBu", contrast=1,
    title.size="Metro population",
    title.col="Growth rate (%)", id="name",
        popup.vars=c("pop2010", "pop2020", "growth")) +
    tm_legend(outside=TRUE)

    current.mode <- tmap_mode("plot")

    # plot map
    map1

    # view map with default view options
    tmap_mode("view")
    map1

    # view map with changed view options
    map1 + tm_view(alpha = 1, basemaps = "Stamen.Watercolor")

    # restore current mode
    tmap_mode(current.mode)
```

`tm_xlab` *Axis labels*

Description

Add axis labels

Usage

```
tm_xlab(text, size = 0.8, rotation = 0, space = NA)

tm_ylab(text, size = 0.8, rotation = 90, space = NA)
```

Arguments

`text`	text for the axis
`size`	fontsize, by default 0.8
`rotation`	rotation angle in degrees. By default, 0 for the x axis label and 90 for the y axis label.
`space`	space between labels and the map in numbers of line heights. By default, it is 0, unless grid labels are plotted outside the frame (i.e., `tm_grid` is called with `labels.inside.frame = FALSE`). In that case, `space` corresponds to the height of one line, taking the grid label size into account.

Examples

```
data(World)

qtm(World, fill="#FFF8DC", projection="longlat", inner.margins=0) +
tm_grid(x = seq(-180, 180, by=20), y=seq(-90,90,by=10), col = "gray70") +
tm_xlab("Longitude") +
tm_ylab("Latitude")
```

World *World and Netherlands map*

Description

Maps of the world and the Netherlands (province and municipality level), class sf

Usage

```
data(World)

data(NLD_prov)

data(NLD_muni)
```

Details

The default projections for these maps are Eckhart IV (World) and Rijksdriehoekstelsel (Netherlands). See below. To change the projection, use set_projection. Alternatively, the projection can be changed temporarily for plotting purposes by using the projection argument of tm_shape (or qtm).

World World map. The default projection for this world map is Eckhart IV since area sizes are preserved, which is a very important property for choropleths.

NLD_prov and NLD_muni, maps of the Netherlands at province and municipality level of 2013. The used projection is the Rijksdriehoekstelsel projection. **Important:** publication of these maps is only allowed when cited to Statistics Netherlands (CBS) and Kadaster Nederland as source.

Source

http://www.naturalearthdata.com for World

http://www.happyplanetindex.org for World

http://www.cbs.nl for NLD_prov and NLD_muni.

References

Statistics Netherlands (2014), The Hague/Heerlen, Netherlands, http://www.cbs.nl.

Kadaster, the Netherlands' Cadastre, Land Registry, and Mapping Agency (2014), Apeldoorn, Netherlands, http://www.kadaster.nl.

Index

∗Topic **GIS,**
tmap-package, 3
∗Topic **animation**
tmap_animation, 14
∗Topic **bubble**
tmap-package, 3
∗Topic **choropleth,**
tmap-package, 3
∗Topic **choropleth**
tm_fill, 39
∗Topic **features**
tm_sf, 67
∗Topic **maps,**
tmap-package, 3
∗Topic **map**
tm_symbols, 71
tmap-package, 3
∗Topic **simple**
tm_sf, 67
∗Topic **statistical**
tmap-package, 3
∗Topic **symbol**
tm_symbols, 71
∗Topic **thematic**
tmap-package, 3
+.tmap, 6

addMiniMap, *61, 62*

bb, *70*

classIntervals, *40*, *56*, *63*, *74*, *82*
crs, *9*, *69*

Extent, *69*, *88*
extent, *49*

formatC, *42*, *45*, *51*, *58*, *64*, *76*, *85*

get_proj4, *9*, *69*
get_projection, *9*, *69*
ggplotGrob, *78*
grid.newpage(), *8*
grob, *78*

icons, *18*

knit_print, *8*, *16*
knit_print.tmap *(*print.tmap*)*, 7
knit_print.tmap_arrange *(*tmap_arrange*)*, 15

land, *5*, 6
last_plot, *18*
leaflet, *8*, *19*, *20*
leafletCRS, *87*

map_coloring, *40*, *43*
marker_icon *(*tmap_icons*)*, 17
markerClusterOptions, *73*, *82*
metro, *5*, 7

NLD_muni, *5*
NLD_muni *(*World*)*, 90
NLD_prov, *5*
NLD_prov *(*World*)*, 90

options, *22*, *23*

par, *43*
points, *78*
print, *5*, *8*, *16*
print.tmap, 7, *19*
print.tmap_arrange *(*tmap_arrange*)*, 15

qtm, *3*, *8*, 8, *19*, *23*, *90*

raster, *8*, *36*, *69*
RasterBrick, *5*, *6*, *8*, *69*
read_osm, *31*
read_shape, *70*
rivers, *5*, 11

saveWidget, *27*
set_projection, *45*, *70*, *90*
setMaxBounds, *88*
setView, *88*
sf, *5*, *7, 8*, *11*, *69*, *88*, *90*
simplify_shape, *70*
smooth_map, *46*
sp, *8*, *69*
SpatialPolygons(DataFrame), *8*, *69*
st_bbox, *69*, *88*

table, *36*
theme_ps, 12
tileOptions, *88*
tm_add_legend, 30
tm_basemap, *3*, *10*, *13*, *23*, 31, *88*
tm_borders, *3*, *13*
tm_borders *(*tm_fill*)*, 39
tm_bubbles, *4*, *13*
tm_bubbles *(*tm_symbols*)*, 71
tm_compass, *4*, *13*, 32, *49*, *53*
tm_credits, *4*, *13*, 33, *53*
tm_dots, *4*, *13*, *67, 68*
tm_dots *(*tm_symbols*)*, 71
tm_facets, *4*, *9*, *14, 15*, *20*, 35, *43*, *52*, *59*, *62*, *65*, *78*
tm_fill, *3*, *13*, *35, 36*, 39, *48*, *50*, *70*, *87, 88*
tm_format, *4*, *13*
tm_format *(*tm_layout*)*, 47
tm_grid, *4*, *13*, 44, *89*
tm_iso, *4*, *13*, 46
tm_layout, *4*, *10*, *13*, *15–17*, *20, 21*, *23, 24*, *26–28*, *32–34*, *40*, 47, *57*, *61*, *63*, *67*, *74*, *82, 83*, *87, 88*
tm_legend, *4*, *13*
tm_legend *(*tm_layout*)*, 47
tm_lines, *3*, *12*, *36*, *46*, 55, *67, 68*, *88*
tm_logo, *4*, *13*, 60
tm_markers, *4*, *13*
tm_markers *(*tm_symbols*)*, 71
tm_minimap, *4*, *13*, *53*, 61
tm_polygons, *3*, *10*, *12*, *23*, *67, 68*
tm_polygons *(*tm_fill*)*, 39
tm_raster, *3*, *13*, 62
tm_rgb, *4*, *13*
tm_rgb *(*tm_raster*)*, 62
tm_scale_bar, *4*, *13*, *53*, 66, *70*
tm_sf, 67
tm_shape, *3*, *10*, *12, 13*, *21*, *23*, *48*, *66*, 69, *87, 88*, *90*
tm_squares, *4*, *13*
tm_squares *(*tm_symbols*)*, 71
tm_style, *4*, *13*, *28*
tm_style *(*tm_layout*)*, 47
tm_symbols, *3*, *9*, *12*, *18*, *30*, *36*, *68*, 71, *88*
tm_text, *3*, *13*, *46*, *78*, 81
tm_tiles, *3*, *10*, *13*, *20*, *62*
tm_tiles *(*tm_basemap*)*, 31
tm_view, *4*, *13*, *16*, *19–21*, *23, 24*, *31*, *43*, *46*, *53*, *59*, *65*, *68*, *78*, *86*, 87
tm_xlab, *4*, *13*, *34*, 89
tm_ylab, *4*, *13*
tm_ylab *(*tm_xlab*)*, 89
tmap, *15*
tmap *(*tmap-package*)*, 3
tmap-element, 12
tmap-package, 3
tmap_animation, *5*, 14, *35*
tmap_arrange, *5*, 15
tmap_format, *10*, 16
tmap_format_add *(*tmap_format*)*, 16
tmap_icons, *5*, 17, *75*, *78*
tmap_last, *5*, 18, *20, 21*
tmap_leaflet, *5*, *8*, 19, *20, 21*
tmap_mode, *4*, *7, 8*, *19*, 20, *42*, *59*, *77*
tmap_options, *4*, *10*, *17*, *20, 21*, 22, *27, 28*, *31*, *43*, *59*, *65*, *78*, *86*, *88*
tmap_options_diff, *28*
tmap_options_diff *(*tmap_options*)*, 22
tmap_options_reset *(*tmap_options*)*, 22
tmap_options_save *(*tmap_options*)*, 22
tmap_save, *5*, *8*, *18*, *20*, 26
tmap_style, *4*, *10*, *23, 24*, 28, *29*
tmap_style_catalog *(*tmap_style_catalogue*)*, 29
tmap_style_catalogue, *17*, *28*, 29
tmap_tip, 29
ttm, *4*
ttm *(*tmap_mode*)*, 20

viewport, *8*, *27*

World, *5*, 90