Package ‘tmap’

July 18, 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 2.3

Date 2019-07-17

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), leafsync, stats

Suggests lwgeom (>= 0.1-4), rmapshaper, rmarkdown, knitr, png,
cartogram, osmdata, ggplot2, dplyr, tidyr, shiny, 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  2019-07-18 06:42:17 UTC

R topics documented:

tmap-package ........................................... 2
+.tmap .................................................. 5
land ..................................................... 5
metro .................................................... 6
print.tmap ............................................. 7
qtm ...................................................... 8
renderTmap ............................................ 11
rivers .................................................. 13
theme_ps ............................................... 13
tmap-element ......................................... 14
tmap_animation ....................................... 15
tmap_arrange ......................................... 16
tmap_format .......................................... 18
tmap_icons ........................................... 19
tmap_last ............................................. 20
tmap_leaflet ......................................... 20
tmap_mode ............................................ 21
tmap_options ......................................... 24
tmap_save ............................................ 28
tmap_style ........................................... 30
tmap_style_catalogue ................................. 31
tmap_tip ............................................... 31
tm_add_legend ......................................... 32
tm_baseemap .......................................... 33
tm_compass .......................................... 35
tm_credits ........................................... 36
tm_facets ............................................ 37
tm_fill ............................................... 42
tm_grid ............................................... 48
tm_iso .................................................. 50
tm_layout ............................................. 51
tm_lines .............................................. 60
tm_logo ............................................... 65
tm_minimap ........................................... 66
tm_raster ............................................. 67
tm_scale_bar ......................................... 72
tm_sf .................................................. 73
tm_shape .............................................. 75
tm_symbols ........................................... 77
tm_text ............................................... 87
tm_view ............................................... 94
**tmap-package**

---

**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")`\(^1\) 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

\(^1\)../doc/tmap-getstarted.html
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:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tmap_icons</td>
<td>Specify icons for markers or proportional symbols</td>
</tr>
</tbody>
</table>

Output functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>print</td>
<td>Plot in graphics device or view interactively in web browser or RStudio’s viewer pane</td>
</tr>
<tr>
<td>tmap_last</td>
<td>Redraw the last map</td>
</tr>
<tr>
<td>tmap_leaflet</td>
<td>Obtain a leaflet widget object</td>
</tr>
<tr>
<td>tmap_animation</td>
<td>Create an animation</td>
</tr>
<tr>
<td>tmap_arrange</td>
<td>Create small multiples of separate maps</td>
</tr>
<tr>
<td>tmap_save</td>
<td>Save thematic maps (either as image or HTML file)</td>
</tr>
</tbody>
</table>

Spatial datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>World</td>
<td>World country data (sf object of polygons)</td>
</tr>
<tr>
<td>NLD_prov</td>
<td>Netherlands province data (sf object of polygons)</td>
</tr>
<tr>
<td>NLD_muni</td>
<td>Netherlands municipal data (sf object of polygons)</td>
</tr>
<tr>
<td>metro</td>
<td>Metropolitan areas (sf object of points)</td>
</tr>
<tr>
<td>rivers</td>
<td>Rivers (sf object of lines)</td>
</tr>
<tr>
<td>land</td>
<td>Global land cover (RasterBrick object)</td>
</tr>
</tbody>
</table>

Author(s)

Martijn Tennekes <mtennekes@gmail.com>

References


See Also

vignette("tmap-getstarted")³

²https://doi.org/10.18637/jss.v084.i06
³../doc/tmap-getstarted.html
+ .tmap  

Stacking of tmap elements

Description

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

Usage

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

Arguments

- `e1`: first tmap-element
- `e2`: second tmap-element

References


See Also

tmap-element and vignette("tmap-getstarted")^5

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.

---

^4https://doi.org/10.18637/jss.v084.i06

^5../doc/tmap-getstarted.html
**metro**

**Source**

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

**References**


---

**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**


---

**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**

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

```r
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` argument 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` argument 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 specifies the raster colors. Only applicable when `shp` is a spatial raster.

**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. Argument 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 or with a search term. This option can be set with tmap_options (argument qtm.minimap).

Value

tmap-element

References


See Also

vignette("tmap-getstarted")

https://doi.org/10.18637/jss.v084.i06
../doc/tmap-getstarted.html
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)
renderTmap

Wrapper functions for using tmap in shiny

Description

Use tmapOutput to create a UI element, and renderTmap to render the tmap map. To update the map (more specifically, to add and remove layers) use tmapProxy. Adding layers is as usual, removing layers can be done with the function tm_remove_layer.

Usage

renderTmap(expr, env = parent.frame(), quoted = FALSE)
tmapOutput(outputId, width = "100\%", height = 400)
tmapProxy(mapId, session = shiny::getDefaultReactiveDomain(), x)

Arguments

expr A tmap object. A tmap object is created with qtm or by stacking tmap-elements.
env The environment in which to evaluate expr
quoted Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable
outputId Output variable to read from
width, height the width and height of the map
mapId single-element character vector indicating the output ID of the map to modify (if invoked from a Shiny module, the namespace will be added automatically)
session the Shiny session object to which the map belongs; usually the default value will suffice
x the tmap object that specifies the added and removed layers.
zindex the z index of the pane in which the layer is contained that is going to be removed. It is recommended to specify the zindex for this layer when creating the map (inside renderTmap).

Details

Two features from tmap are not (yet) supported in Shiny: small multiples (facets) and colored backgrounds (argument bg.color of tm_layout). Workarounds for small multiples: create multiple independent maps or specify as.layers = TRUE in tm_facets.
Examples

```r
if (require("shiny")) {
  data(World)
  world_vars <- setdiff(names(World), c("iso_a3", "name", "sovereignty", "geometry"))
  ui <- fluidPage(
    tmapOutput("map"),
    selectInput("var", "Variable", world_vars)
  )
  server <- function(input, output, session) {
    output$map <- renderTmap({
      tm_shape(World) +
      tm_polygons(world_vars[1], zindex = 401)
    })
    observe({
      var <- input$var
      tmapProxy("map", session, {
        tm_remove_layer(401) +
        tm_shape(World) +
        tm_polygons(var, zindex = 401)
      })
    })
  }
  app <- shinyApp(ui, server)
  if (interactive()) app
}
```

rivers  

Spatial data of rivers

Description

Spatial data of rivers, of class `sf`

Usage

```r
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?

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\(^8\)

See Also

vignette("tmap-getstarted")\(^9\)

The examples in each of the element functions

\(^8\)https://doi.org/10.18637/jss.v084.i06
\(^9\)https://doi.org/10.18637/jss.v084.i06
\(../doc/tmap-getstarted.html\)
tmap_animation  

Create animation

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

Usage

```r
tmap_animation(tm, filename = "animation.gif", width = NA,
               height = NA, dpi = 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)
- `dpi`: dots per inch. Only applicable for raster graphics. By default 300, but this can be set with the option `output.dpi` in `tmap_options`.
- `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

```r
## 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

```r

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

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

```r
# 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
- `widths` vector of column widths. It should add up to 1 and the length should be equal to `ncol`
- `heights` vector of row heights. It should add up to 1 and the length should be equal to `nrow`
- `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, widths = c(.25, .75))
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)
**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**

```r
marker_icon()
```

**Arguments**

- `file` name of the format. Run `tmap_format()` to see the choices.
- `width` options from `tm_layout` or `tm_view`. Can also be a list of those options.
- `height` 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**

```r
# 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_last

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**

Retrieves 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

```r
tmap_last()
```

Value

- call

See Also

- `tmap_save`
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, in.shiny = FALSE, ...)

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
in.shiny: is the leaflet output going to be used in shiny? If so, two features are not supported and therefore disabled: facets and colored backgrounds.
...: arguments passed on to print.tmap

Value

leaflet object

See Also

tmapOutput for tmap in Shiny, 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)
if <- tmap_leaflet(map1)

# show leaflet widget
if

# 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 de-
fault mode, and supports all tmap’s features, such as small multiples (see tm_facets) and extensive layout settings (see tm_layout). It is rec-
ommended 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

See Also
vignette("tmap-getstarted") 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

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

map1 <- tm_shape(Worl +
tm_polygons("income_grp", palette="-Blues", contrast=.7, id="name", title="Income group") +
tm_shape(meter) +
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",
```

10https://doi.org/10.18637/jss.v084.i06
11../doc/tmap-getstarted.html
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, output.format, output.size, output.dpi)

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 = 64)

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 pre-
  viewed at http://leaflet-extras.github.io/leaflet-providers/
  preview. Also supports URL's for tile servers, such as
  "http://{s}.tile.openstreetmap.org/
  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?

output.format  The format of the static maps saved with tmap_save without specification of the filename. The default is "png".

output.size  The size of the static maps saved with tmap_save without specification of width and height. The unit is squared inch and the default is 49. This means that square maps (so with aspect ratio 1) will be saved as 7 by 7 inch images and a map with aspect ratio 2 (e.g. most world maps) will be saved as approximately 10 by 5 inch.

output.dpi  The default number of dots per inch for tmap_save and tmap_animation.

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("myStyle") 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**

```r
# 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 to a file. This can be either a static plot (e.g. png) or an interactive map (html).
usage

tmap_save(tm = NULL, filename = NA, width = NA, height = NA,
units = NA, dpi = NA, 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, tiff, and html are supported. If the extension is missing, the file will be saved as a static plot in "plot" mode and as an interactive map (html) in "view" mode. The default format for static plots is png, but this can be changed using the option "output.format" in tmap_options.

height, width
  The width and height of the plot (not applicable for html files). Units are set with the argument units. If one of them is not specified, this is calculated using the formula asp = width / height, where asp is the estimated aspect ratio of the map. If both are missing, they are set such that width * height is equal to the option "output.size" in tmap_options. This is by default 49, meaning that is the map is a square (so aspect ratio of 1) both width and height are set to 7.

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. By default it is set to 300, but this can be changed using the option "output.dpi" in tmap_options.

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

```r
## 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

```r
tmap_style(style)
```
tmap_style_catalogue

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.

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()

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, group = NULL)
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
- **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. By default NULL, which means that the legend will not be shown in the layer control item.

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

```r
tm_basemap(server = NA, group = NA, alpha = NA, tms = FALSE)
```

```r
tm_tiles(server, group = NA, alpha = 1, zindex = NA, tms = FALSE)
```

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. Set `group=NULL` to hide the layer in the layer control item. By default, it will be set to the name of the shape (specified in `tm_shape`). Tile layers generated with `tm_basemap` will be base groups whereas tile layers generated with `tm_tiles` will be overlay groups.

- **alpha**: alpha

- **tms**: is the provided tile server defined according to the TMS protocol? By default `FALSE`.

- **zindex**: zindex of the pane in view mode. By default, it is set to the layer number plus 400. By default, the tmap layers will therefore be placed in the custom panes "tmap401", "tmap402", etc., except for the base tile layers, which are placed in the standard "tile". This parameter determines both the name of the pane and the z-index, which determines the pane order from bottom to top. For instance, if `zindex` is set to 500, the pane will be named "tmap500".

Details

When `tm_basemap` is not specified, the default basemaps are shown, which can be configured by the `basemaps` argument 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 `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

```r
## 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/",
```
tm_compass

"World_Light_Gray_Reference/MapServer/tile/{z}/{y}/{x}"), group = "Labels")

# Use tmap options 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**

```r
tm_compass(north = 0, type = NA, text.size = 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, fontsize = NULL)
```

**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)

- `text.size`  
  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`.
fontsize  deprecated: renamed to `text.size`

Examples

```r
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 = TRUE)

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

### Description

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

### Usage

```r
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_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 \textit{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 \texttt{col} in \texttt{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

\begin{verbatim}
\begin{verbatim}
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)
\end{verbatim}
\end{verbatim}

Arguments

- \textbf{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).
- \textbf{along} data variable name by which the data is split and plotted on separate pages. This is especially useful for animations made with \texttt{tmap_animation}. The \texttt{along} argument can be used in combination with the \texttt{by} argument. It is only supported in "plot" mode (so not in "view" mode).
- \textbf{as.layers} logical that determines whether facets are shown as different layers in "view" mode. By default \texttt{FALSE}, i.e. facets are drawn as small multiples.
tm_facets

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 have 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.factorth 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


See Also

vignette("tmap-getstarted")13

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"),

12https://doi.org/10.18637/jss.v084.i06
13../doc/tmap-getstarted.html
```r
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"),
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

```r
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"))
```

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

```r
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))
```

```r
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**

```r
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(), zindex = 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", "jenks", and "log10_pretty". 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 (except "log10_pretty"), see the details in `classIntervals`. Continuous options are "cont", "order", and "log10". The first maps the values of `col` to a smooth gradient, the second maps the order of values of `col` to a smooth gradient, and the third uses a logarithmic transformation.

**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 \texttt{xxx.xxx}, is used. If \texttt{scientific=TRUE} then "g", which means that numbers are formatted scientifically, i.e. \texttt{n.dddE+nn} if needed to save space.

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

\texttt{big.num.abbr}  Vector that defines whether and which abbreviations are used for large numbers. It is a named numeric vector, where the name indicated the abbreviation, and the number the magnitude (in terms on numbers of zero). Numbers are only abbreviation when they are large enough. Set it to \texttt{NA} to disable abbreviations. The default is \texttt{c\("mln\"=6,\"bln\"=9\)}. For layers where \texttt{style} is set to \texttt{log10} or \texttt{log10_pretty}, the default is \texttt{NA}.

prefix  Prefix of each number

\texttt{suffix}  Suffix of each number

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

\texttt{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 \texttt{text.to.columns=TRUE}

\texttt{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 \texttt{text.to.columns=TRUE}

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

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

...  Other arguments passed on to \texttt{formatC}

\texttt{legend.is.portrait}  \texttt{logical} that determines whether the legend is in portrait mode (\texttt{TRUE}) or landscape (\texttt{FALSE})

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

\texttt{legend.hist}  \texttt{logical} that determines whether a histogram is shown

\texttt{legend.hist.title}  \texttt{title} for the histogram. By default, one title is used for both the histogram and the normal legend.

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

\texttt{legend.hist.z}  \texttt{index value} that determines the position of the histogram legend element

\texttt{id}  \texttt{name} of the data variable that specifies the indices of the polygons. Only used for \texttt{\texttt{\"view\"}} mode (see \texttt{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 format-
ing 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.

zindex  zindex of the pane in view mode. By default, it is set to the layer number
plus 400. By default, the tmap layers will therefore be placed in the custom
panes "tmap401", "tmap402", etc., except for the base tile layers, which
are placed in the standard "tile". This parameter determines both the name
of the pane and the z-index, which determines the pane order from bottom to top.
For instance, if zindex is set to 500, the pane will be named "tmap500".

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. Set group = NULL to hide
the layer in the layer control item. By default, it will be set to the name of the shape
(specified in tm_shape).

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
tm_fill

References

Tennekes, M., 2018, tmap: Thematic Maps in R, Journal of Statistical Software, 84(6), 1-39, DOI\textsuperscript{14}

See Also

vignette("tmap-getstarted")\textsuperscript{15}

Examples

data(World)

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

# Borders only
tm_shape(World) + tmBorders()

# 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

\textsuperscript{14}https://doi.org/10.18637/jss.v084.i06
\textsuperscript{15}../doc/tmap-getstarted.html
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 / graticule 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, tm_grid draws horizontal and vertical lines according to the coordinate system of the (master) shape object. Latitude and longitude graticules are drawn with tm_graticules.

**Usage**

```r
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.cardinal = FALSE,
labels.margin.x = 0, labels.margin.y = 0, labels.space.x = NA,
labels.space.y = NA, labels.inside.frame = FALSE,
ticks = !labels.inside.frame, lines = TRUE, zindex = NA)
tm_graticules(x = NA, y = NA, n.x = NA, n.y = NA,
projection = "longlat", labels.format = list(suffix =
intToUtf8(176)), labels.cardinal = 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. For the labels, a pretty sequence is used, so the number of actual labels may be different than `n.x`.
- `n.y` preferred number of grid lines for the y axis. For the labels, a pretty sequence is used, so the number of actual labels may be different than `n.y`.
- `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.cardinal add the four cardinal directions (N, E, S, W) to the labels, instead of using negative coordinates for west and south (so it assumes that the coordinates are positive in the north-east direction).
labels.margin.x margin between tick labels of x axis and the frame. Note that when labels.inside.frame == FALSE and ticks == TRUE, the ticks will be adjusted accordingly.
labels.margin.y margin between tick labels of y axis and the frame. Note that when labels.inside.frame == FALSE and ticks == TRUE, the ticks will be adjusted accordingly.
labels.space.x space that is used for the labels and ticks for the x-axis when labels.inside.frame == FALSE. By default, it is determined automatically using the widths and heights of the tick labels. The unit of this parameter is text line height.
labels.space.y space that is used for the labels and ticks for the y-axis when labels.inside.frame == FALSE. By default, it is determined automatically using the widths and heights of the tick labels. The unit of this parameter is text line height.
labels.inside.frame
  Show labels inside the frame? By default FALSE

ticks
  If labels.inside.frame = FALSE, should ticks can be drawn between the labels and the frame?

lines
  If labels.inside.frame = FALSE, should grid lines can be drawn?

zindex
  zindex of the pane in view mode. By default, it is set to the layer number plus 400. By default, the tmap layers will therefore be placed in the custom panes "tmap401", "tmap402", etc., except for the base tile layers, which are placed in the standard "tile". This parameter determines both the name of the pane and the z-index, which determines the pane order from bottom to top. For instance, if zindex is set to 500, the pane will be named "tmap500".

... arguments passed on to tm_grid

Examples

  current.mode <- tmap_mode("plot")
  data(NLD_muni, World)
  tmap_arrange(
    qtm(NLD_muni, borders = NULL) + tm_grid(),
    qtm(NLD_muni, borders = NULL) + tm_graticules()
  )
  qtm(World, shape.projection = "robin", style = "natural") +
  tm_graticules(ticks = FALSE) +
  tm_layout(frame=FALSE)
  tmap_mode(current.mode)

---

**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. Set group=NA to hide the layer in the layer control item. By default, it will be set to the name of the shape (specified in tm_shape).
...
arguments passed on to tm_lines or tm_text

See Also

smooth_map

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.color, legend.title.size, legend.title.fontface,
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 [tmaptools::palette_explorer()](#)) are used. It is also possible to 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 colors 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. Note that the actual height or width of the legend is determined by the content of the legend (and the used font sizes). This argument specifies the upperbound of the width or height.

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 different legends are stacked: "vertical" or "horizontal". To stack items within a same legend, look at "legend.is.portrait" in the specific layer calls.

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.color

Color of the legend titles

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.color

Color of the legend text

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, text.or.more, and big.num.abbr 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.
**big.num.abbr** Vector that defines whether and which abbreviations are used for large numbers. It is a named numeric vector, where the name indicated the abbreviation, and the number the magnitude (in terms on numbers of zero). Numbers are only abbreviation when they are large enough. Set it to NA to disable abbreviations. The default is `c("mln" = 6,"bln" = 9)`. For layers where `style` is set to `log10` or `log10_pretty`, the default is NA.

**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.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.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 plotted 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.
...
other arguments from `tm_layout`
stylename of the style
formatname of the format
**Details**

Predefined styles:
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


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) +
  tmBorders("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"),
```
\begin{verbatim}
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")
\end{verbatim}
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.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(), zindex = NA, 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", "jenks", and "log10_pretty". 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 (except "log10_pretty"), see the details in classIntervals. Continuous options are "cont", "order", and "log10". The first maps the values of col to a smooth gradient, the second maps the order of values of col to a smooth gradient, and the third uses a logarithmic transformation.
in case \texttt{style="fixed"}, breaks should be specified. The \texttt{breaks} argument can also be used when \texttt{style="cont"}. In that case, the breaks are mapped evenly to the sequential or diverging color palette.

\texttt{interval.closure} 

value that determines whether where the intervals are closed: "left" or "right". Only applicable if \texttt{col} is a numeric variable.

\texttt{palette} 

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

\texttt{labels} 

labels of the classes

\texttt{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 \texttt{NA}, which means that the value that corresponds to the middle color class (see \texttt{style}) is mapped to the middle color. Only applies when \texttt{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.

\texttt{stretch.palette} 

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

\texttt{contrast} 

vector of two numbers that determine the range that is used for sequential and diverging palettes (applicable when \texttt{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).

\texttt{colorNA} 

color used for missing values. Use \texttt{NULL} for transparency.

\texttt{textNA} 

text used for missing values.

\texttt{showNA} 

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

\texttt{colorNULL} 

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

\texttt{title.col} 

title of the legend element regarding the line colors

\texttt{title.lwd} 

title of the legend element regarding the line widths

\texttt{legend.col.show} 

logical that determines whether the legend for the line colors is shown

\texttt{legend.lwd.show} 

logical that determines whether the legend for the line widths is shown

\texttt{legend.format} 

list of formatting options for the legend numbers. Only applicable if \texttt{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.

big.num.abbr  Vector that defines whether and which abbreviations are used for
large numbers. It is a named numeric vector, where the name indicated the
abbreviation, and the number the magnitude (in terms on numbers of zero).
Numbers are only abbreviation when they are large enough. Set it to NA
to disable abbreviations. The default is c("mln" = 6, "bln" = 9). For
layers where style is set to log10 or log10_pretty, the default is
NA.

prefix  Prefix of each number

suffix  Suffix of each number

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

text.less.than  Character value(s) to use to translate "Less than". When a char-
acter 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.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
tm_lines

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.

zindex
  zindex of the pane in view mode. By default, it is set to the layer number plus 400. By default, the tmap layers will therefore be placed in the custom panes "tmap401", "tmap402", etc., except for the base tile layers, which are placed in the standard "tile". This parameter determines both the name of the pane and the z-index, which determines the pane order from bottom to top. For instance, if zindex is set to 500, the pane will be named "tmap500".

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. Set group = NULL to hide the layer in the layer control item. By default, it will be set to the name of the shape (specified in tm_shape).

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


See Also

vignette("tmap-getstarted")^19

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.

^18https://doi.org/10.18637/jss.v084.i06
^19../doc/tmap-getstarted.html
Usage

```r
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 on 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

```r
## 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",
  systemctl.file("img/tmap.png", package="tmap"))) +
```
tm_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

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, 
zindex = NA, group = NA, auto.palette.mapping = NULL, 
max.categories = NULL, max.value = 255)

tm_rgb(r = 1, g = 2, b = 3, alpha = NA, saturation = 1, 
interpolate = TRUE, max.value = 255, ...)

tm_rgba(r = 1, g = 2, b = 3, a = 4, alpha = NA, saturation = 1, 
interpolate = TRUE, max.value = 255, ...)

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 
figure 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 tmap::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", "jenks", and "log10_pretty". 
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 (except "log10_pretty"), see the details in classIntervals. 
Continuous options are "cont", "order", and "log10". The first maps the 
values of col to a smooth gradient, the second maps the order of values of col 
to a smooth gradient, and the third uses a logarithmic transformation.
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".

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.

big.num.abbr  Vector that defines whether and which abbreviations are used for large numbers. It is a named numeric vector, where the name indicated the abbreviation, and the number the magnitude (in terms on numbers of zero). Numbers are only abbreviation when they are large enough. Set it to NA to disable abbreviations. The default is c("mln" = 6,"bln" = 9). For layers where style is set to log10 or log10_pretty, the default is NA.

prefix  Prefix of each number

suffix  Suffix of each number

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.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

zindex

zindex of the pane in view mode. By default, it is set to the layer number plus 400. By default, the tmap layers will therefore be placed in the custom panes "tmap401", "tmap402", etc., except for the base tile layers, which are placed in the standard "tile". This parameter determines both the name of the pane and the z-index, which determines the pane order from bottom to top. For instance, if zindex is set to 500, the pane will be named "tmap500".

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. Set group = NULL to hide the layer in the layer control item. By default, it will be set to the name of the shape (specified in tm_shape).

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.

max.value

for tm_rgb, what is the maximum value per layer? By default 255.

r

raster band for the red channel. It should be an integer between 1 and the number of raster layers.

g

raster band for the green channel. It should be an integer between 1 and the number of raster layers.

b

raster band for the blue channel. It should be an integer between 1 and the number of raster layers.

...

arguments passed on from tm_raster to tm_rgb

a

raster band for the alpha channel. It should be an integer between 1 and the number of raster layers.

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\textsuperscript{20}

\textsuperscript{20}https://doi.org/10.18637/jss.v084.i06
See Also

vignette("tmap-getstarted")

Examples

data(World, land, metro)

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",
            "#67864", "#B4E664", "#9BC832", "#EBFF64", "#F06432", "#9132E6",
            "#E664E6", "#9B82E6", "#B4FEF0", "#644644", "#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)

## 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")
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, text.size = 0.5, text.color = NA, color.dark = "black", color.light = "white", lwd = 1, position = NA, just = NA, size = NULL)

Arguments

breaks  breaks of the scale bar. If not specified, breaks will be automatically be chosen given the preferred 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.
text.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.
size  deprecated: renamed to text.size

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**

```r
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 of `tm_symbols`. By default, the size is similar to dot size (see `tm_dots`).
- **shape**
  - Shape of the dots. See the `shape` argument of `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 the `alpha` argument of `tm_polygons`, `tm_lines` and `tm_symbols`.
- **palette**
  - Palette. See the `palette` argument of `tm_polygons`, `tm_lines` and `tm_symbols`.
- **border.col**
  - Color of the borders. See the `border.col` argument of `tm_polygons` and `tm_symbols`.
- **border.lwd**
  - Line width of the borders. See the `border.lwd` argument of `tm_polygons` and `tm_symbols`.
- **border.lty**
  - Line type of the borders. See the `border.lty` argument of `tm_polygons` and `tm_symbols`.
- **border.alpha**
  - Transparency of the borders. See the `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. Set `group = NULL` to hide the layer in the layer control item. By default, it will be set to the name of the shape (specified in `tm_shape`).
- **...**
  - Other arguments passed on to `tm_polygons`, `tm_lines` and `tm_symbols`.
tm_shape

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

<table>
<thead>
<tr>
<th>shp</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>name of the shape object (character) as it appears in the legend in &quot;view&quot; mode. Default value is the name of shp.</td>
</tr>
</tbody>
</table>

---

22 ../doc/tmap-getstarted.html
is.master logical that determines whether this \texttt{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 \texttt{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 \texttt{tm_shape} is the master shape element.

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

bbox bounding box. One of the following:

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

If unspecified, the current bounding box of \texttt{shp} is taken. The bounding box is feed to \texttt{bb} (as argument \texttt{x}). The other arguments of \texttt{bb} can be specified directly as well (see \ldots).

unit desired units of the map. One of "metric" (default), "imperial", "km", "m", "mi" and "ft". Used to specify the scale bar (see \texttt{tm_scale_bar}) and to calculate densities for choropleths (see argument \texttt{convert2density} in \texttt{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 \texttt{simplify_shape}, from which the arguments \texttt{keep.units} and \texttt{keep.subunits} can be passed on (see \ldots). This requires the suggested package \texttt{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 \texttt{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 centroid.

filter logical vector which indicated per feature whether it should be included. Features for which filter is \texttt{FALSE} will be colored light gray (see the \texttt{colorNULL} argument in the layer functions)

Arguments passed on to \texttt{bb} (e.g. \texttt{ext} can be used to enlarge or shrink a bounding box), and \texttt{simplify_shape} (the arguments \texttt{keep.units} and \texttt{keep.subunits})
Value

tmap-element

References


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)

---

23 https://doi.org/10.18637/jss.v084.i06
24 ../doc/tmap-getstarted.html
tm_symbols

## 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**

```r
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.close = "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.close = "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,
```
tm_symbols

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(), zindex = NA, 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.
Note that the default shapes (specified by shapes) is not supported in "view"
mode. 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 trans-
parent) 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.
clusterings 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", "jenks", and "log10_pretty". 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 (except "log10_pretty"), see the details in classIntervals. Continuous options are "cont", "order", and "log10". The first maps the values of col to a smooth gradient, the second maps the order of values of col to a smooth gradient, and the third uses a logarithmic transformation.

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)
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 specified (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.height`.

**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.

big.num.abbr Vector that defines whether and which abbreviations are used for large numbers. It is a named numeric vector, where the name indicated the abbreviation, and the number the magnitude (in terms on numbers of zero). Numbers are only abbreviation when they are large enough. Set it to NA to disable abbreviations. The default is c(“mln” = 6,”bln” = 9). For layers where style is set to log10 or log10_pretty, the default is NA.

prefix Prefix of each number

suffix Suffix of each number

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 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
tm_symbols

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.

zindex
   zindex of the pane in view mode. By default, it is set to the layer number plus 400. By default, the tmap layers will therefore be placed in the custom panes "tmap401", "tmap402", etc., except for the base tile layers, which are placed in the standard "tile". This parameter determines both the name of the pane and the z-index, which determines the pane order from bottom to top. For instance, if zindex is set to 500, the pane will be named "tmap500".

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. Set group = NULL to hide the
layer in the layer control item. By default, it will be set to the name of the shape (specified in \texttt{tm\_shape}).

\texttt{auto\_palette\_mapping}

deprecated. It has been replaced by \texttt{midpoint} for numeric variables and \texttt{stretch\_palette} for categorical variables.

\texttt{max\_categories}

deprecated. It has moved to \texttt{tmap\_options}.

\ldots arguments passed on to \texttt{tm\_symbols}. For \texttt{tm\_markers}, arguments can also be passed on to \texttt{tm\_text}. In that case, they have to be prefixed with \texttt{text.}, e.g. the \texttt{col} argument should be names \texttt{text.col}

\texttt{title}

shortcut for \texttt{title.col} for \texttt{tm\_dots}

\texttt{legend\_show}

shortcut for \texttt{legend.col.show} for \texttt{tm\_dots}

\texttt{legend\_is\_portrait}

shortcut for \texttt{legend.col.is.portrait} for \texttt{tm\_dots}

\texttt{legend\_z}

shortcut for \texttt{legend.col.z} shortcut for \texttt{tm\_dots}

\texttt{text}

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

\texttt{text\_just}

justification of marker text (see \texttt{just} argument of \texttt{tm\_text}). Only applicable in plot mode.

\texttt{markers\_on\_top\_of\_text}

For \texttt{tm\_markers}, should the markers be drawn on top of the text labels?

\textbf{Details}

Small multiples can be drawn in two ways: either by specifying the \texttt{by} argument in \texttt{tm\_facets}, or by defining multiple variables in the aesthetic arguments, which are \texttt{size}, \texttt{col}, and \texttt{shape}. In the latter case, the arguments, except for the ones starting with \texttt{legend.}, can be specified for small multiples as follows. If the argument normally only takes a single value, such as \texttt{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 \texttt{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.

1. A numeric value that specifies the plotting character of the symbol. See parameter \texttt{pch} of \texttt{points} and the last example to create a plot with all options. Note that this is not supported for the "view" mode.

2. A \texttt{grob} object, which can be a \texttt{ggplot2} plot object created with \texttt{ggplotGrob}. To specify multiple shapes, a list of \texttt{grob} objects is required. See example of a proportional symbol map with \texttt{ggplot2} plots.

3. An icon specification, which can be created with \texttt{tmap\_icons}.

To specify multiple shapes (needed for the \texttt{shapes} argument), a vector or list of these shape specification is required. The shape specification options can also be mixed. For the \texttt{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 \texttt{shape} argument. For small multiples, a list of these shape specification(s) should be provided.
tm_symbols

Value
tmap-element

References

See Also
vignette("tmap-getstarted")

Examples
data(World, metro)
m立法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")

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") +
      theme_classic())

  if (!require(tmaptools)) {
    message("tm_shape() requires tmaptools to be installed.")
  } else {
    tm_shape(NLD_prov) +
      tm_fill("#808080") +
      tm_bubbles("perc", col=origin_cols, shape=origin_cols,
        size=.3, border.col=origin_cols, border.alpha=.5,
        title="Origin")
  }
}

https://doi.org/10.18637/jss.v084.i06
../doc/tmap-getstarted.html
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(group = "Provinces") +
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",
group = "Charts",
id = "name",
popup.vars = c("population", "origin_native",
  "origin_west", "origin_non_west")) +
tm_add_legend(type="fill",
group = "Charts",
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(naturalearth)) {
  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**

```r
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, zindex = 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", "jenks", and "log10_pretty". 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 (except "log10_pretty"), see the details in classIntervals. Continuous options are "cont", "order", and "log10". The first maps the values of col to a smooth gradient, the second maps the order of values of col to a smooth gradient, and the third uses a logarithmic transformation.

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.
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.

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.

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.

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).

colour for missing values. Use NULL for transparency.

text used for missing values.

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

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

font face of the text labels. By default, determined by the fontface argument of \texttt{tm_layout}.

font family of the text labels. By default, determined by the fontfamily argument of \texttt{tm_layout}.

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

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.

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

background color of the text labels. By default, \texttt{bg.color}=NA, so no background is drawn.

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

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=NULL` 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.

**big.num.abbr** Vector that defines whether and which abbreviations are used for large numbers. It is a named numeric vector, where the name indicated the abbreviation, and the number the magnitude (in terms on numbers of zero). Numbers are only abbreviation when they are large enough. Set it to `NA` to disable abbreviations. The default is `c(\"mln\"=6,\"bln\"=9)`. For layers where `style` is set to `log10` or `log10.pretty`, the default is `NA`.

**prefix** Prefix of each number

**suffix** Suffix of each number

**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`)

zindex

zindex of the pane in view mode. By default, it is set to the layer number plus 400. By default, the tmap layers will therefore be placed in the custom panes "tmap401", "tmap402", etc., except for the base tile layers, which are placed in the standard "tile". This parameter determines both the name of the pane and the z-index, which determines the pane order from bottom to top. For instance, if `zindex` is set to 500, the pane will be named "tmap500".

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. Set `group = NULL` to hide the layer in the layer control item. By default, it will be set to the name of the shape (specified in `tm_shape`).

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


²⁷https://doi.org/10.18637/jss.v084.i06
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.

---

28../doc/tmap-getstarted.html
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 an 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


See Also

vignette("tmap-getstarted")

Examples

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

map1 <- tm_shape(meter) +
  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")

https://doi.org/10.18637/jss.v084.i06
../doc/tmap-getstarted.html
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**

```r
tm_xlab(text, size = 0.8, rotation = 0, space = 0)
tm_ylab(text, size = 0.8, rotation = 90, space = 0)
```

**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**

```r
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")
```
Description

Maps of the world and the Netherlands (province and municipality level), class \texttt{sf}

Usage

\begin{verbatim}
data(World)
data(NLD_prov)
data(NLD_muni)
\end{verbatim}

Details

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

\texttt{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.

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

Source

\begin{verbatim}
http://www.naturalearthdata.com for World
http://www.happyplanetindex.org for World
http://www.cbs.nl for NLD_prov and NLD_muni.
\end{verbatim}

References

\begin{verbatim}
\end{verbatim}