Package ‘tmap’

February 20, 2015

Maintainer  Martijn Tennekes <mtennekes@gmail.com>
License  GPL-3
Title  Thematic Maps
Type  Package
LazyLoad  yes
Author  Martijn Tennekes
Description  Thematic maps are geographical maps in which spatial data distributions are visualized. This package offers a flexible, layer-based, way to create all kinds of thematic maps, such as choropleths and bubble maps.
Version  0.6
Date  2014-06-30
Depends  R (>= 3.0)
Imports  rgeos, grid, gridBase, sp, rgdal, classInt, RColorBrewer, igraph, fields, vegan
Suggests  knitr
VignetteBuilder  knitr
NeedsCompilation  no
Repository  CRAN
Date/Publication  2014-07-30 18:25:21

R topics documented:

  tmap-package ......................................................... 2
  +.tmap .......................................................... 3
  animation_tmap .................................................. 3
  append_data .................................................... 4
  approx_areas .................................................... 5
  calc_densities .................................................. 6
  convert_shape_data .............................................. 6
  crop_shape ..................................................... 7
double_line .......................... 7
fit_polylines ........................................ 8
get_IDs ........................................ 8
get_polygon_ranges ......................................... 9
print.tmap ....................................... 9
qtm ........................................... 10
read_shape ...................................... 11
rivers .......................................... 12
set_projection .................................... 12
split_lines_equal .................................... 14
split_lines_poly .................................... 14
tmap-element ........................................ 15
tm_borders ....................................... 15
tm_bubbles ....................................... 16
tm_facets ......................................... 18
tm_fill ........................................... 20
tm_grid ........................................... 22
tm_layout ......................................... 23
tm_lines .......................................... 25
tm_shape .......................................... 27
tm_text ........................................... 29
World ............................................ 31
write_shape ....................................... 32

Index 33

---

tmap-package

Thematic Maps

Description

Package: tmap
Type: Package
Version: 0.6
Date: 2014-07-30
License: GPL-3
LazyLoad: yes

Details

Thematic maps are geographical maps in which spatial data distributions are visualized. This package offers a flexible, layer-based, way to create all kinds of thematic maps, such as choropleths and bubble maps. To get started, see vignette("tmap-nutshell").


+.tmap

Stacking of tmap elements

Description

The plus operator allows you to stack tmap-elements.

Usage

### S3 method for class 'tmap'

```r
e1 + e2
```

Arguments

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

See Also

vignette("tmap-nutshell")

---

animation_tmap

Create animations

Description

This function creates a gif or mpeg animation from a tm plot. The free tool ImageMagick is required.

Usage

```r
animation_tmap(expr, width = 1000, height = 1000, delay = 40,
filename = "animation.gif")
```

Arguments

- `expr`: R expression to create series of tm plots. In order to create a series of plots, which are combined to an animation, it is important to set nrow and ncol in `tm_facets` such that `nrow * ncol < [number of small multiples]`. In most situations, where one map is shown, both nrow and ncol are set to 1.
- `width`: width of the animation file (in pixels)
- `height`: height of the animation file (in pixels)
- `delay`: delay time between images
- `filename`: filename of the video (should be a .gif or .mpg file)
Examples

```r
## Not run:
data(Europe)

animation_tmap({
  tm_shape(Europe) +
  tm_fill("yellow") +
  tm_borders() +
  tm_facets(by = "name", nrow=1,ncol=1) +
  tm_layout(scale=2)
}, width=1200, height=800, filename="my_animation.gif"

## End(Not run)
```

---

**append_data**

### Append data

Append data.frame to shape object

#### Usage

```r
append_data(shp, data, key.data = NULL, key.shp = NULL, 
  ignore.duplicates = FALSE, fixed.order = is.null(key.data) && 
  is.null(key.shp))
```

#### Arguments

- **shp** shape object
- **data** data.frame
- **key.data** variable name of data to be matched with key.shp. If not specified, and fixed.order is FALSE, the row names of data are taken.
- **key.shp** variable name of shp map data to be matched with key.data. If not specified, and fixed.order is FALSE, the polygon ID’s are taken.
- **ignore.duplicates** should duplicated keys in data be ignored? (FALSE by default)
- **fixed.order** should the data be append in the same order as the shapes in shp?

#### Value

shape object with appended data
approx_areas

**approx_areas**  
*Approximate area sizes of the shapes.*

**Description**

This function approximates the area sizes of the polygons either in 1) squared kilometers, 2) absolute numbers based on the polygon coordinates, 3) proportional numbers, 4) normalized numbers.

**Usage**

`approx_areas(shp, total.area.km2 = NA, units = NULL)`

**Arguments**

- `shp`: shape object, i.e. a `SpatialPolygons(DataFrame)`
- `total.area.km2`: total area size of `shp` in number of squared kilometers. If NA, and `unit = "km2"`, then the polygon coordinates are assumed to be in meters.
- `units`: one of
  - "km2": Squared kilometers. For this method, `total.area.km2` is required.
  - "abs": Absolute numbers based on polygon coordinates. Only useful if the projection satisfies the equal-area property. Note: these are equal to the `area` slots of the polygons, where the `area` slots of the holes are subtracted. Also note that for many projections, the coordinate units are meters, so the area sizes correspond to squared meters (rather than squared kilometers).
  - "prop": Proportional numbers. In other words, the total of the area sizes equals one.
  - "norm": Normalized numbers. All area sizes are normalized to the largest area, of which the area size equals one.

The default method is "abs", unless `total.area.km2` is specified (in that case, it is "km2").

**Details**

To approximate the sizes in squared kilometer, `total.area.km2` is required. Note that this method is an approximation, since it depends on the used projection and the level of detail of the `SpatialPolygons` object. Projections with equal-area property are highly recommended.

**Value**

numeric vector of area sizes
calc_densities    Calculate densities

Description
This function transposes quantity variables to density variables, which are needed for choropleths.

Usage
calc_densities(shp, var, total.area.km2 = NA, drop = TRUE)

Arguments
- shp: a shape object
- var: name(s) of a quality variable name contained in the map data
- total.area.km2: total area size of shp in number of squared kilometers. If NA, and unit="km2", then the polygon coordinates are assumed to be in meters.
- drop: boolean that determines whether an one-column data-frame should be returned as a vector

Value
vector or data.frame (depending on whether length(var)==1 with density values. This can be appended directly to the shape file.

convert_shape_data    Convert shape data

Description
Convert data from one polygon shape to another. It uses an intersection matrix, which stores the intersection ratios of the two shape objects per polygon

Usage
convert_shape_data(shp.from, shp.to, variables.from = NULL, variables.to = NULL)

Arguments
- shp.from: the shape object to be converted. It should contain data.
- shp.to: the shape object to be converted into.
- variables.from: names of the variable of shp.from to be converted. If missing, all numeric variables are taken.
- variables.to: variable names to be used. Should be the same number of variable names as variables.from
**crop_shape**

Value

shape object shp.to with converted data from shp.from

---

**Crop shape file**

---

Description

Crops a shape object, to a rectangle which is by default its bounding box.

Usage

crop_shape(shp, bbox = shp@bbox)

Arguments

- **shp** shape object.
- **bbox** rectangle to crop the shp with. It is represented by a 2x2 matrix in which the x and y coordinates are respectively row 1 and 2, and the minimum and maximum values are respectively column 1 and 2. By default the bounding box of shp is taken.

Value

a cropped shape object. Its bounding box is set to bb. Data is retained in case shp has data. A vector of matched ID’s is stored as an attribute matchID. This vector contains for each polygon in the returned shape object the number of its original polygon in shp.

---

**double_line**

Create a double line

---

Description

Create a double line from a single line. (Experimental)

Usage

double_line(shp, width)

Arguments

- **shp** The shape object that contains the lines (SpatialLinesDataFrame)
- **width** Width between the double lines

Value

SpatialLinesDataFrame
**fit_polylines**  
*Fit polylines through a set of spatial points*

**Description**  
This function fits one or more smooth polylines through a set of spatial points. (Experimental)

**Usage**

```r
fit_polylines(..., id = NULL, min.dist = 10, max.opt.dist = 250, sep.dist = 5000, verbose = TRUE)
```

**Arguments**

- `...`: Shape objects that contain spatial points.
- `id`: Name of the data variable that determines the classes of the points. For each class, a polyline is fit. If omitted, a polyline is fit through all points.
- `min.dist`: Minimum distance. Points that are closer than `min.dist` from any other point are omitted in the fitting method (see details below).
- `max.opt.dist`: Maximal optimized distance. Between any two points that lie closer than `max.opt.dist` to each other, an edge is created in the fitting method (see details below).
- `sep.dist`: Separation distance. If the distance between two groups of points is larger than `sep.dist`, two separate polylines are created.
- `verbose`: Print logging text

**Value**

SpatialLinesDataFrame

---

**get_IDs**  
*Get ID's of the shape items*

**Description**  
Get ID's of the shape items. For polygons and lines, the ID attribute is used. For points, the coordinates are used.

**Usage**

```r
get_IDs(shp)
```

**Arguments**

- `shp`: Shape object
get_polygon_ranges

Value
   vector of ID's

Description
   Get the ranges of the polygons in the current projection. Also the total range of the shape object is returned.

Usage
   get_polygon_ranges(shp, key = NULL)

Arguments
   shp     shape object
   key     variable name in the shape data that identifies the polygons. If not specified, the default ID is used (see get_IDs).

Value
   list containing a data.frame of ranges per polygon, and a range vector of the total shape object

print.tmap

Print tm object

Description
   Print tm object

Usage
   ## S3 method for class 'tmap'
   print(x, ...)

Arguments
   x   tm object
   ... not used
qtm

Quick thematic map plot

Description

This function is a convenient wrapper for drawing thematic maps quickly.

Usage

```r
qtm(shp, fill = "grey90", bubble.size = NULL, bubble.col = NULL,
    text = NULL, text.cex = 1, line.lwd = NULL, line.col = NULL,
    borders = "grey40", theme = NULL, scale = 1, ...)
```

Arguments

- **shp**: shape object. For `tm_fill` and `tm_bubbles`, a `SpatialPolygonsDataFrame` or a `SpatialPointsDataFrame` is required. `SpatialPoints` and `SpatialPointsDataFrame` are only used for `tm_bubbles`.
- **fill**: either a color to fill the polygons, or name of the data variable in `shp` to draw a choropleth.
- **bubble.size**: name of the data variable in `shp` for the bubblemap that specifies the sizes of the bubbles. If neither `bubble.size` nor `bubble.col` is specified, no bubblemap is drawn.
- **bubble.col**: name of the data variable in `shp` for the bubblemap that specifies the colors of the bubbles. If neither `bubble.size` nor `bubble.col` is specified, no bubblemap is drawn.
- **text**: Name of the data variable that contains the text labels.
- **text.cex**: Font size of the text labels. Either a constant value, or the name of a numeric data variable.
- **line.lwd**: either a line width or a name of the data variable that specifies the line width. Only applicable if `shp` is a `SpatialLines` or `SpatialLinesDataFrame`.
- **line.col**: either a line color or a name of the data variable that specifies the line colors. Only applicable if `shp` is a `SpatialLines` or `SpatialLinesDataFrame`.
- **borders**: color of the polygon borders. Use NA to omit the borders.
- **theme**: one of "World", "Europe", or "NLD"
- **scale**: numeric value that serves as the global scale parameter. All font sizes, bubble sizes, border widths, and line widths are controlled by this value. The parameters `bubble.size`, `text.cex`, and `line.lwd` can be scaled separately with respectively `bubble.scale`, `text.scale`, and `line.scale`.
- **...**: parameters passed on to the `tm_*` functions.

Value

`tmap-element`
read_shape

See Also

vignette("tmap-nutshell")

Examples

# load shape objects
data(Europe)
data(World)

# just the map
qtm(Europe)

# choropleth
qtm(Europe, fill="gdp_cap_est", text="iso_a3", text.cex="pop_est", title="GDP per capita", textNA="Non-European countries")
qtm(World, fill="pop_est_dens", theme="World", style="kmeans", title="Population per km")

# bubble map
qtm(Europe, bubble.size="pop_est", bubble.col="part", theme="Europe")

---

read_shape Read shape file

Description

This function reads an ESRI shape file.

Usage

read_shape(file)

Arguments

file a shape file name (including directory).

Details

This function is a convenient wrapper of readOGR of the package sp. It also fixes a common bug when reading shape files that are in the Dutch rijksdriehoekstelsel.

Value

shape object
rivers  Spatial data of cities and rivers

Description
Spatial data of main world cities and rivers.

Source
http://www.naturalearthdata.com/

set_projection  Set and get the map projection

Description
The function set_projection sets the projection of a shape file. It is a convenient wrapper of spTransform with shortcuts for commonly used projections. The projection can also be set directly in the plot call with tm_shape. This function is also used to set the current projection information without transformation of the shape object, which is useful when this information is missing in the shape object. The function get_projection is used to get the projection information.

Usage

set_projection(shp, projection = NULL, current.projection = NULL,
transform = !is.null(projection), overwrite.current.projection = FALSE)

get_projection(shp)

Arguments

shp  shape object, which is one of
• "1" SpatialPolygons(DataFrame)
• "2" SpatialPoints(DataFrame)
• "3" SpatialLines(DataFrame)

projection  character that determines the projection. Either a PROJ.4 character string (see http://trac.osgeo.org/proj/), of one of the following shortcuts:
"longlat" Not really a projection, but a plot of the longitude-latitude coordinates (WGS84 datum).
"wintri" Winkel Tripel (1921). Popular projection that is useful in world maps. It is the standard of world maps made by the National Geographic Society. Type: compromise
"robin" Robinson (1963). Another popular projection for world maps. Type: compromise
"eck4" Eckert IV (1906). Projection useful for world maps. Area sizes are preserved, which makes it particularly useful for truthful choropleths. Type: equal-area

"hd" Hobo-Dyer (2002). Another projection useful for world maps in which area sizes are preserved. Type: equal-area

"gall" Gall (Peters) (1855). Another projection useful for world maps in which area sizes are preserved. Type: equal-area

"merc" Mercator (1569). Projection in which shapes are locally preserved. However, areas close to the poles are inflated. Google Maps uses a close variant of the Mercator. Type: conformal

"utmXX(s)" Universal Transverse Mercator. Set of 60 projections where each projection is a traverse mercator optimized for a 6 degree longitude range. These ranges are called UTM zones. Zone 01 covers -180 to -174 degrees (West) and zone 60 174 to 180 east. Replace XX in the character string with the zone number. For southern hemisphere, add "s". So, for instance, the Netherlands is "utm31" and New Zealand is "utm59s"

"m11" Miller (1942). Projetion based on Mercator, in which poles are displayed. Type: compromise

"eqc8" Equirectangular (120). Projection in which distances along meridians are conserved. The equator is the standard parallel. Also known as Plate Carrée. Type: equidistant

"eqc30" Equirectangular (120). Projection in which distances along meridians are conserved. The latitude of 30 is the standard parallel. Type: equidistant

"eqc45" Equirectangular (120). Projection in which distances along meridians are conserved. The latitude of 45 is the standard parallel. Also known as Gall isographic. Type: equidistant

"rd" Rijksdriehoekstelsel. Triangulation coordinate system used in the Netherlands.

See http://en.wikipedia.org/wiki/List_of_map_projections for a overview of projections. By default, the projection is used that is defined in the shp object itself.

current.projection

the current projection of shp. Only use this if the current projection is missing.

transform

Logical that determines whether to transform the shape file into the specified projection. By default TRUE. If the current shape projection is missing, longitude latitude coordinates (WGS84) are assumed. If FALSE, then the specified projection is simply written to the shape file without transforming it (use this at your own risk!).

overwrite.current.projection

logical that determines whether the current projection is overwritten if it already has a projection that is different.

Value

set_projection returns a (transformed) shape object with updated projection information. get_projection returns the PROJ.4 character string of shp.
### split_lines_equal

**Description**

Split lines in segments of equal length. (Experimental)

**Usage**

```r
split_lines_equal(shp, dist = 1000, include.last = FALSE)
```

**Arguments**

- `shp`: The shape object that contains the lines
- `dist`: Distance per segment
- `include.last`: Include last point, even though the distance is less than `dist` from the previous point?

**Value**

`SpatialLinesDataFrame`

### split_lines_poly

**Description**

Split lines by polygons. Data of the corresponding polygons is appended to the line segments (Experimental)

**Usage**

```r
split_lines_poly(shp.lines, shp.poly, variables.lines, variables.poly)
```

**Arguments**

- `shp.lines`: The shape object that contains the lines
- `shp.poly`: The shape object that contains the polygons
- `variables.lines`: Names of the variables of `shp.lines` that are appended to the split lines shape object.
- `variables.poly`: Names of the variables of `shp.poly` that are appended to the split lines shape object.

**Value**

`shape object with splitted lines`
**Description**

Building block to draw thematic maps.

**Details**

The only fundamental, and hence required element is

- `tm_shape` that specifies the shape object, and also controls the projection and bounding box

The elements that serve as drawing layers are

- `tm_borders` to draw polygon borders
- `tm_fill` to color the polygons
- `tm_bubbles` to draw bubbles
- `tm_lines` to draw lines

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`.

The layout elements are

- `tm_layout` to change the appearance of the map, for instance titles and legend positions. Predefined themes for the example shape files are `tm_layout_World`, `tm_layout_Europe`, and `tm_layout_NLD`.
- `tm_facets` that specifies how small multiples are created, i.e. how many rows and columns, and whether the statistical data variables have free scales or not.
- `tm_grid` that specifies grid lines

**tm_borders**

*Draw polygon borders*

**Description**

This layer defines the borders of the polygons. Color, line width and line type can be set.

**Usage**

```r
tm_borders(col = "grey40", lwd = 1, lty = "solid")
```
**Arguments**

- `col` line color
- `lwd` line width (see `par`)
- `lty` line type (see `par`)

**Value**

tmap-element

**See Also**

vignette("tmap-nutshell")

**Examples**

```r
## Europe example
data(Europe)
tm_shape(Europe) + tm_borders()

## Netherlands example
data(NLD_prov)
data(NLD_muni)

tm_shape(NLD_prov) +
  tm_fill("name") +
tm_shape(NLD_muni) +
  tm_borders() +
tm_shape(NLD_prov) +
  tm_borders(lwd=2) +
tm_text("name") +
tm_layout_NLD("Provinces and municipalities", legend.show=FALSE)
```

---

**tm_bubbles**

*Draw bubble map*

**Description**

This layer specifies a bubble map. Both colors and sizes of the bubbles can be mapped to data variables.

**Usage**

```r
tm_bubbles(size = 1, col = "blueviolet", border.lwd = NA,
  border.col = "black", scale = 1, size.lim = NA, n = 5,
  style = "pretty", breaks = NULL, palette = NULL, labels = NULL,
  auto.palette.mapping = TRUE, contrast = 1, max.categories = 12,
  colorNA = "#FF1414", textNA = "Missing", xmod = 0, ymod = 0)
```
Arguments

size  shp data variable that determines the bubble sizes. Multiple variable names create small multiples

col  color(s) of the bubble. Either a color (vector), or categorical variable name(s). Multiple variable names create small multiples

border.lwd  line width of the bubble borders. If NA (default), no bubble borders are drawn.

border.col  color of the bubble borders.

scale  bubble size multiplier number.

size.lim  vector of two limit values of the size variable. Only bubbles are drawn whose value is greater than or equal to the first value. Bubbles 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

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

style  method to cut the color scale: "fixed", "equal", "pretty", "quantile", "kmeans". Only applicable when col is a numeric variable name.

breaks  in case style=="fixed", breaks should be specified

palette  color palette (see RColorBrewer::display.brewer.all) for the bubbles. Only when col is set to a variable.

labels  labels of the classes

auto.palette.mapping

When diverging colour palettes are used (i.e. "RdBu") this method automatically maps colors to values such that the middle colors (mostly white or yellow) are assigned to values of 0, and the two sides of the color palette are assigned to negative respectively positive values.

contrast  number between 0 and 1 (default) that determines the contrast of the palette. Only applicable when auto.palette.mapping=TRUE and col is a numeric variable name.

max.categories  in case col is the name of a categorical variable, 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.

colorNA  colour for missing values

textNA  text used for missing values. Use NA to omit text for missing values in the legend

xmod  horizontal position modification of the bubbles, relatively where 0 means no modification, and 1 means the total width of the frame. 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 bubbles. In most coordinate systems (projections), the origin is located at the bottom left, so negative xmod move the bubbles to the left, and negative ymod values to the bottom.

ymod  vertical position modification. See xmod.
**tm_facets**

Small multiples grid

**Description**

This element specifies how small multiples are placed in a grid. Either the argument `by` should be specified, i.e. the name of a variable by which the data is grouped, or multiple variable names should.
be provided with \texttt{tm\_fill}, \texttt{tm\_lines}, or \texttt{tm\_bubbles}. In this function, the number of rows and columns can be specified, as well as whether the scales are free (i.e. independent of each other).

Usage

\begin{verbatim}
tm_facets(by = NULL, ncol = NULL, nrow = NULL, 
free.scales = is.null(by), free.scales.fill = free.scales, 
free.scales.bubble.size = free.scales, 
free.scales.bubble.col = free.scales, free.scales.line.col = free.scales, 
free.scales.line.lwd = free.scales)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{by} data variable name by which the data is split
  \item \texttt{ncol} number of columns of the small multiples grid
  \item \texttt{nrow} number of rows of the small multiples grid
  \item \texttt{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 \texttt{tm\_fill}, color and size from \texttt{tm\_bubbles} and line color from \texttt{tm\_lines}.
  \item \texttt{free.scales.fill} logical. Should the color scale for the choropleth be free?
  \item \texttt{free.scales.bubble.size} logical. Should the bubble size scale for the bubble map be free?
  \item \texttt{free.scales.bubble.col} logical. Should the color scale for the bubble map be free?
  \item \texttt{free.scales.line.col} logical. Should the line color scale be free?
  \item \texttt{free.scales.line.lwd} Should the line width scale be free?
\end{itemize}

Value

\texttt{tmap-element}

See Also

\texttt{vignette("tmap-nutshell")}

Examples

\begin{verbatim}
## World example
data(World)
tm_shape(World) +
  tm_fill(c("green", "blue")) +
  tm_layout_World(c("A green world", "A blue world"))

## Europe example
data(Europe)
tm_shape(Europe) +
\end{verbatim}
tm_fill(c("gdp_cap_est", "pop_est_dens"), style="kmeans") +
  tm_layout_Europe(c("GDP per capita", "Population density"), scale=2)

## Netherlands example

data(NLD_muni)
data(NLD_prov)

tm_shape(NLD_muni) +
  tm_fill(c("pop_0_14", "pop_15_24", "pop_25_44", "pop_45_64", "pop_65plus"),
          convert2density=TRUE, style="kmeans") +

tm_shape(NLD_prov) +
  tm_borders() +
  tm_facets(free.scales=FALSE) +
  tm_layout_NLD(c("Population 0 to 14", "Population 15 to 24", "Population 25 to 44",
                   "Population 45 to 64", "Population 65 and older"), scale=4, draw.frame = TRUE)

---

**tm_fill**  
*Draw choropleth*

**Description**

This layer specifies a choropleth. A color palette is mapped to a data variable. By default, a diverging color palette is used for numeric variables and a qualitative palette for categorical variables.

**Usage**

```r
  tm_fill(col = "grey90", palette = NULL, convert2density = FALSE,
          area = NULL, n = 5, style = "pretty", breaks = NULL, labels = NULL,
          auto.palette.mapping = TRUE, contrast = 1, max.categories = 12,
          colorNA = "grey65", textNA = "Missing", thres.poly = 1e-05)
```

**Arguments**

- **col**: either a single color value or a name of the data variable that is contained in `shp`. In the latter case, a choropleth is drawn.
- **palette**: palette name. See `RColorBrewer::display.brewer.all()` for options. Use a "-" as prefix to reverse the palette. By default, "RdYlGn" is taken for numeric variables and "Dark2" for categorical variables.
- **n**: preferred number of classes (in case col is a numeric variable)
- **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.
style
method to cut the color scale (in case col is a numeric variable): "fixed", "equal", "pretty", "quantile", "kmeans"

breaks
in case style=="fixed", breaks should be specified

labels
labels of the classes

auto.palette.mapping
When diverging colour palettes are used (i.e. "RdBu") this method automatically maps colors to values such that the middle colors (mostly white or yellow) are assigned to values of 0, and the two sides of the color palette are assigned to negative respectively positive values.

contrast
number between 0 and 1 (default) that determines the contrast of the palette. Only applicable when auto.palette.mapping=TRUE

max.categories
in case col is the name of a categorical variable, 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.

colorNA
color used for missing values

textNA
text used for missing values. Use NA to omit text for missing values in the legend

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.

Value
tmap-element

See Also
vignette("tmap-nutshell")

Examples

# World examples
data(World)

tm_shape(World) + tm_fill("green3") + tm_layout_World(title="A green World")

World$highlighted <- ifelse(World$iso_a3 %in% c("GRL", "AUS"), "gold", "gray75")
tm_shape(World, projection="merc") +
tm_fill("highlighted") +
tm_borders() +
tm_layout("Mercator projection. Although used in Google Maps, it is discouraged for statistical purposes. In reality, Australia is 3 times larger than Greenland!", inner.margins=c(0,0,.1,0), title.cex=.6)

tm_shape(World) +
  tm_fill("pop_est_dens", style="kmeans", palette="YlOrRd") +
  tm_borders() +
  tm_text("iso_a3", cex="AREA", cex.lowerbound=.4, bg.alpha=0) +
  tm_layout_World(title="Population density per km^2")
tm_shape(World) +
  tm_fill("income_grp", palette="-Blues") +
  tm_borders() +
  tm_text("iso_a3", cex="AREA", scale=1.5, bg.alpha=0) +
  tm_layout_world("Income classification")

# Europe example
data(Europe) +
  tm_shape(Europe) +
  tm_fill("gdp_cap_est", style="kmeans", textNA = "Non-European countries") +
  tm_borders() +
  tm_text("iso_a3", cex="AREA", scale=2, bg.alpha=0) +
  tm_layout_Europe("GDP per capita")

# Netherlands examples
data(NLD_muni)
data(NLD_prov)

  tm_shape(NLD_prov) +
    tm_fill("name") +
  tm_shape(NLD_muni) +
    tm_borders() +
  tm_shape(NLD_prov) +
    tm_borders(lwd=2) +
    tm_text("name") +
  tm_layout_NLD("Provinces and municipalities", legend.show=FALSE)

  tm_shape(NLD_muni) +
    tm_fill(col="population", convert2density=TRUE, style="kmeans") +
    tm_borders() +
  tm_shape(NLD_prov) +
    tm_borders(lwd=2) +
  tm_layout_NLD(title="Population (per km2)", legend.digits=0)

---

**tm_grid**

**Coordinate grid lines**

**Description**

This element draws coordinate grid lines.

**Usage**

```
TM_grid(n.x = 8, n.y = 8, col = "grey50", labels.cex = 0.75,
  labels.col = "grey20", on.top = TRUE)
```

**Arguments**

- `n.x`: Preferred number of grid lines for the x axis.
- `n.y`: Preferred number of grid lines for the y axis.
tm_layout

Layout elements of cartographic maps

Description

This element specifies layout options for the maps. The main function `tm_layout` is used by default as general layout theme. The functions `tm_layout_World`, `tm_layout_Europe`, and `tm_layout_NLD` are layout themes for World, Europe, and Netherlands maps (which are contained in this package).

Usage

```r
tm_layout(title = NA, scale = 1, title.cex = 1.5, bg.color = NULL,
          draw.frame = TRUE, title.position = c("left", "top"),
          title.bg.color = NA, asp = NA, frame.lwd = 1,
          outer.margins = rep(0.02, 4), inner.margins = rep(0.02, 4),
          outer.bg.color = "white", legend.show = TRUE, legend.hist.show = FALSE,
          legend.only = FALSE, legend.titles = c(fill = NA, bubble.size = NA,
          bubble.col = NA, line.col = NA, line.lwd = NA), legend.position = c("left",
          "top"), legend.is.portrait = c(fill = TRUE, bubble.size = FALSE, bubble.col
          = TRUE, line.col = TRUE, line.lwd = FALSE), legend.width = 0.3,
          legend.height = 0.9, legend.hist.height = 0.3,
          legend.config = c("fill_hist", "fill", "bubble.size", "bubble.col",
          "line.col", "line.lwd"), legend.title.cex = 1, legend.text.cex = 0.7,
          legend.hist.cex = 0.7, legend.digits = 2L, legend.bg.color = NA)

tm_layout_World(title = NA, scale = 0.85, title.position = c("left",
"bottom"), title.bg.color = TRUE, outer.margins = rep(0.02, 4),
inner.margins = c(0, 0.02, 0.02, 0.02), legend.position = c("left",
"bottom"), legend.width = 0.2, legend.height = 0.5,
legend.bg.color = TRUE, ...)

tm_layout_Europe(title = NA, legend.position = c("left", "top"),
outer.margins = rep(0.02, 4), inner.margins = c(0, 0.25, 0, 0), ...)

tm_layout_NLD(title = NA, draw.frame = FALSE, inner.margins = c(0.05, 0.3,
0.05, 0.05), legend.position = c("left", "top"), legend.width = 0.3, ...)
```
Arguments

title
Title(s). By default, the name of the statistical variable of which the legend is
drawn at the top (see legend.config) is used as a title.

scale
numeric value that serves as the global scale parameter. All font sizes, bubble
sizes, border widths, and line widths are controled by this value. Each of these
elements can be scaled independantly with the scale, lwd, or cex arguments
provided by the tmap-elements.

title.cex
Relative size of the title

bg.color
Background color. By default it is light grey (grey85) for choropleths and white
for other maps.

draw.frame
Boolean that determines whether a frana is drawn.

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

title.bg.color
background color of the title. Use TRUE to match with the overall background
color bg.color.

asp
Aspect ratio. The aspect ratio of the map (width/height). If NA, it is deter-
mimed by the bounding box (see argument bbox of tm_shape) and the argument
frame.margins. If 0, then the aspect ratio is adjusted to the aspect ratio of the
device.

frame.lwd
Width of the frame

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.

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.

outer.bg.color
Background color outside the frame.

legend.show
Logical that determines whether the legend is shown. Use legend.config to
configure which legend elements are shown.

legend.hist.show
Logical that determines whether to show a histogram for the choropleth fill vari-
able.

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

legend.titles
titles of the legend elements. Named character vector, where the names cor-
respond to the legend elements and the value to the titles of those elements.
Possible legend element names are: "fill", "bubble.size", "bubble.col",
"line.col", and "line.lwd". For small multiples, a list of character vectors
can be provided, where the list names correspond to the legend elements, and
the character vectors to the legend titles of the small multiples per legend ele-
ment. By default, the names of the corresponding statistical variables are used.
For the legend element at the top, no legend title is used since the main title is
used for this. A legend title for this element can be speficied.
**legend.position**
Position of the legend. Vector of two values, specifying the x and y coordinates. Either this vector contains "left", "center" or "right" for the first value and "top", "center", or "right" 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 legend.

**legend.is.portrait**
Logical vector that determines whether the orientation of the legend elements are portrait (TRUE) or landscape (FALSE). The vector should be named with the corresponding elements, which are "fill", "bubble.size", "bubble.col", "line.col", and "line.lwd".

**legend.width**
Maximum width of the legend

**legend.height**
Maximum height of the legend.

**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.config**
Character vector that specifies which legend elements are drawn and at what position. The legend elements are called "fill", "fill_hist", "bubble.size", "bubble.col", "line.col", and "line.lwd". The **legend.config** vector should only contain these elements (it can also be a subset). The order corresponds to the order in which the legend elements are stacked from top to bottom.

**legend.title.cex**
Relative font size for the legend title

**legend.text.cex**
Relative font size for the legend text elements

**legend.hist.cex**
Relative font size for the choropleth histogram

**legend.digits**
Number of digits for the legend labels

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

... other arguments from **tm_layout**

**See Also**

vignette("tmap-nutshell")

---

**tm_lines** *Draw spatial lines*

**Description**

This layer draw spatial lines.
Usage

```r
tm_lines(col = "red", lwd = 1, lty = "solid", scale = 1, n = 5,
        style = "pretty", breaks = NULL, palette = NULL, labels = NULL,
        auto.palette.mapping = TRUE, contrast = 1, max.categories = 12,
        colorNA = "grey65", textNA = "Missing")
```

Arguments

- **col**: color of the lines. Either a color value or a data variable name.
- **lwd**: line width
- **lty**: line type
- **scale**: line width multiplier number.
- **n**: preferred number of color scale classes. Only applicable when `lwd` is the name of a numeric variable.
- **style**: method to cut the color scale: "fixed", "equal", "pretty", "quantile", "kmeans". Only applicable when `lwd` is the name of a numeric variable.
- **breaks**: in case `style`="fixed", breaks should be specified
- **palette**: color palette (see `RColorBrewer::display.brewer.all`) for the lines. Only when `col` is set to a variable.
- **labels**: labels of the classes
- **auto.palette.mapping**: When diverging colour palettes are used (i.e. "RdBu") this method automatically maps colors to values such that the middle colors (mostly white or yellow) are assigned to values of 0, and the two sides of the color palette are assigned to negative respectively positive values. In this case of line widths, obviously only the positive side is used.
- **contrast**: number between 0 and 1 (default) that determines the contrast of the palette. Only applicable when `auto.palette.mapping=TRUE`
- **max.categories**: in case `col` is the name of a categorical variable, 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.
- **colorNA**: color used for missing values
- **textNA**: text used for missing values. Use NA to omit text for missing values in the legend

Value

- `tmap-element`

See Also

- `vignette("tmap-nutshell")`
Examples

data(rivers)

data(Europe)
  tm_shape(Europe) +
  tm_fill("darkolivegreen3") +
  tm_borders("white") +
  tm_shape(rivers) +
  tm_lines(col="navy", lwd="scalerank", scale=2) +
  tm_layout("Rivers in Europe", legend.show=FALSE)

data(World)
  tm_shape(World) +
  tm_fill("darkolivegreen3") +
  tm_shape(rivers) +
  tm_lines(col="navy") +
  tm_layout_World("Rivers in the World")

---

**tm_shape**

*Specify the shape object*

Description

This element specifies the shape object. Also the used projection and covered area (bounding box) can be set.

Usage

```r
tm_shape(shp, projection = NULL, xlim = NULL, ylim = NULL,
  relative = TRUE, bbox = NULL)
```

Arguments

- **shp**
  - shape object, which is one of
    - "1"`SpatialPolygons(DataFrame)`
    - "2"`SpatialPoints(DataFrame)`
    - "3"`SpatialLines(DataFrame)`

- **projection**
  - character that determines the projection. Either a PROJ.4 character string (see [http://trac.ostm.org/proj/](http://trac.ostm.org/proj/)), of one of the following shortcuts:
    - "longlat" Not really a projection, but a plot of the longitude-latitude coordinates.
    - "wintri" Winkel Tripel (1921). Popular projection that is useful in world maps. It is the standard of world maps made by the National tmngraphic Society. Type: compromise
"robin" Robinson (1963). Another popular projection for world maps. Type: compromise
"eck4" Eckert IV (1906). Projection useful for world maps. Area sizes are preserved, which makes it particularly useful for truthful choropleths. Type: equal-area
"hd" Hobo-Dyer (2002). Another projection useful for world maps in which area sizes are preserved. Type: equal-area
"gall" Gall (Peters) (1855). Another projection useful for world maps in which area sizes are preserved. Type: equal-area
"merc" Mercator (1569). Projection in which shapes are locally preserved. However, areas close to the poles are inflated. Google Maps uses a close variant of the Mercator. Type: conformal
"mill" Miller (1942). Projection based on Mercator, in which poles are displayed. Type: compromise
"eqc0" Equirectangular (120). Projection in which distances along meridians are conserved. The equator is the standard parallel. Also known as Plate Carrée. Type: equidistant
"eqc30" Equirectangular (120). Projection in which distances along meridians are conserved. The latitude of 30 is the standard parallel. Type: equidistant
"eqc45". Equirectangular (120). Projection in which distances along meridians are conserved. The latitude of 45 is the standard parallel. Also known as Gall isographic. Type: equidistant
"rd" Rijksdriehoekstelsel. Triangulation coordinate system used in the Netherlands.

See [http://en.wikipedia.org/wiki/List_of_map_projections](http://en.wikipedia.org/wiki/List_of_map_projections) for an overview of projections. By default, the projection is used that is defined in the shp object itself.

- `xlim` limits of the x-axis
- `ylim` limits of the y-axis
- `relative` boolean that determines whether relative values are used for `xlim` and `ylim` or absolute. Note: relative values will depend on the current bounding box (bbox) of the first shape object.
- `bbox` bounding box, which is a 2x2 matrix that consists absolute `xlim` and `ylim` values. If specified, it overrides the `xlim` and `ylim` parameters.

**Value**

`tmap-element`

**See Also**

`set_projection`, `vignette("tmap-nutshell")`
Examples

```r
data(World)

tm_shape(World, projection="longlat") +
  tm_fill() +
  tm_borders() +
  tm_layout("Long lat coordinates (WGS84)", inner.margins=c(0,0,.1,0), title.cex=.8)

World$highlighted <- ifelse(World$iso_a3 %in% c("GRL", "AUS"), "gold", "gray75")

tm_shape(World, projection="merc") +
  tm_fill("highlighted") +
  tm_borders() +
  tm_layout("Mercator projection. Although used in Google Maps, it is discouraged for statistical purposes. In reality, Australia is 3 times larger than Greenland!", inner.margins=c(0,0,.1,0), title.cex=.6)

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

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

---

**tm_text**

Add text labels

**Description**

This layer adds text labels

**Usage**

```r
tm_text(text, cex = 1, root = 3, fontcolor = NA, fontface = "plain",
      fontfamily = "sans", case = NA, bg.color = NA, bg.alpha = 100,
      cex.lowerbound = 0.4, print.tiny = FALSE, scale = 1, xmod = 0,
      ymod = 0)
```

**Arguments**

- `text` name of the variable in the shape object that contains the text labels
cex  relative size of the text labels. Either one number, a name of a numeric variable in the shape data that is used to scale the sizes proportionally, or AREA where the text size is proportional to the the area size of the polygons.

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

fontcolor  relative size of the text labels

fontface  font face of the text labels

fontfamily  font family of the text labels

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

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

bg.alpha  number between 0 and 255 that specifies the transparency of the text background (0 is totally transparent, 255 is solid background). The default value is 100.

cex.lowerbound  lowerbound for cex. Needed to ignore the tiny labels in case cex is a variable.

print.tiny  boolean that determines if tiny labels (which size is smaller than cex.lowerbound) are print at size cex.lowerbound

scale  scalar needed in case cex is based

xmod  horizontal position modification of the text, relatively where 0 means no modification, and 1 means the total width of the frame. 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.

Value
tmap-element

See Also
vignette("tmap-nutshell")

Examples

# Europe example
data(Europe)
tm_shape(Europe) +
  tm_fill("gdp_cap_est", style="kmeans", textNA = "Non-European countries") +
  tm_borders() +
  tm_text("iso_a3", cex="AREA", scale=2, bg.alpha=0) +
  tm_layout_Europe("GDP per capita")
data(rivers)
data(cities)

tm_shape(Europe) +
  tm_fill("pop_est_dens", style="kmeans", textNA="Non-European countries") +
  tm_borders() +
tm_shape(rivers) +
  tm_lines("dodgerblue3") +
tm_shape(cities) +
  tm_text("name", cex="pop_max", scale=1, ymod=-.02, root=4, cex.lowerbound = .60,
           bg.color="yellow", bg.alpha = 150) +
  tm_bubbles("pop_max", "red", border.col = "black", border.lwd=1, size.lim = c(0, 2e7)) +
tm_shape(Europe) +
  tm_text("name", cex="area", scale=1.5, root=8, cex.lowerbound = .40,
           fontface="bold", case=NA, fontcolor = "gray35") +
tm_layout_Europe("Map of Europe",
                legend.titles = c(fill="Country population density (people per km2)",
                                 bubble.size="City Population"))
write_shape

Write shape file

Description
This function writes an ESRI shape file.

Usage
write_shape(shp, file)

Arguments
- shp: a shape object.
- file: file name (including directory)

Details
This function is a convenient wrapper of writeOGR of the package sp.
Index

∗Topic GIS, tmap-package, 2
∗Topic animation
    animation_tmap, 3
∗Topic bubble
tmap-package, 2
∗Topic choropleth,
tmap-package, 2
∗Topic densities
calc_densities, 6
∗Topic maps,
tmap-package, 2
∗Topic map
tmap-package, 2
∗Topic statistical
tmap-package, 2
∗Topic thematic
tmap-package, 2
+. tmap, 3

animation_tmap, 3
append_data, 4
approx_areas, 5
calc_densities, 6
cities (rivers), 12
convert_shape_data, 6
crop_shape, 7
double_line, 7
Europe (World), 31
fit_polygons, 8
get_IDs, 8, 9
get_polygon_ranges, 9
get_projection (set_projection), 12
NLD_muni (World), 31
NLD_prov (World), 31
par, 16
print.tmap, 9
qt, 10
read_shape, 11
rivers, 12
set_projection, 12, 28
SpatialLines, 10
SpatialLines(DataFrame), 12, 27
SpatialLinesDataFrame, 7, 10
SpatialPoints, 10
SpatialPoints(DataFrame), 12, 27
SpatialPointsDataFrame, 10
SpatialPolygons(DataFrame), 12, 27
SpatialPolygonsDataFrame, 10
split_lines_equal, 14
split_lines_poly, 14
spTransform, 12
tm_borders, 15, 15, 27
tm_bubbles, 10, 15, 16, 19, 27
tm_facets, 3, 15, 18
tm_fill, 10, 15, 19, 20, 27
tm_grid, 15, 22
tm_layout, 15, 23
tm_layout_Europe, 15
tm_layout_Europe (tm_layout), 23
tm_layout_NLD, 15
tm_layout_NLD (tm_layout), 23
tm_layout_World, 15
tm_layout_World (tm_layout), 23
tm_lines, 15, 19, 25, 27
tm_shape, 12, 15, 24, 27
tm_text, 27, 29
tmap (tmap-package), 2
tmap-element, 15
tmap-package, 2

World, 31
write_shape, 32