Package ‘tmap’

March 27, 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 statistical data are visualized. This package offers a flexible, layer-based, way to create thematic maps, such as choropleths and bubble maps.

Version 0.7
Date 2015-03-27
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 2015-03-27 20:58:30

R topics documented:

tmap-package .......................................................... 2
+.tmap ................................................................. 3
animation_tmap ....................................................... 3
append_data .......................................................... 4
approx_areas ......................................................... 5
calc_densities ....................................................... 7
convert_shape_data ................................................... 8
crop_shape .......................................................... 9
double_line .......................................................... 9
Description

Package: tmap
Type: Package
Version: 0.7
Date: 2015-03-27
License: GPL-3
LazyLoad: yes
Details

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

Author(s)

Martijn Tennekes <mtennekes@gmail.com>

### +.tmap

**Stacking of tmap elements**

Description

The plus operator allows you to stack tmap elements.

Usage

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

Arguments

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

See Also

`vignette("tmap-nutshell")`

### animation_tmap

**Create animations**

Description

Create a gif or mpeg animation from a tmap 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 tmap plots. 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.

width  width of the animation file (in pixels)

height  height of the animation file (in pixels)

delay  delay time between images (in 1/100th of a second)

filename  filename of the video (should be a .gif or .mpg file)

Note

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

Examples

```r
## Not run:
data(europeI
animation_tmap({
  tm_shape(europeI +
  tm_fill("yellow") +
  tm_borders() +
  tm_facets(by = "name", nrow=1, ncol=1) +
  tm_layout(scale=2)
}, width=1200, height=800, delay=100, filename="European countries.gif")
## End(Not run)
```

append_data  Append data to a shape object

Description

Data, in the format of a data.frame, is appended to a shape object. This is either done by a right join where keys are specified for both data and shape, or by fixed order.

Usage

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

Arguments

shp        shape object, which is one of
1. SpatialPolygons(DataFrame)
2. SpatialPoints(DataFrame)
3. SpatialLines(DataFrame)
data       data.frame
key.shp    variable name of shp map data to be matched with key.data. If not specified, and fixed.order is FALSE, the ID’s of the polygons/lines/points are taken.
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.
ignore.duplicates
should duplicated keys in data be ignored? (FALSE by default)
ignore.na  should NA values in key.data and key.shp 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.

Examples

## Not run:
data(Europe)
f <- tempfile()
domain_codes <- read.table(f, header=TRUE, sep="|")unlink(f)
domain_codes <- subset(domain_codes, select = c("Alpha3Code", "TopLevelDomain"))
domain_codes$Alpha3Code <- toupper(domain_codes$Alpha3Code)

Europe <- append_data(Europe, domain_codes, key.shp = "iso_a3", key.data = "Alpha3Code", ignore.na = TRUE)

qtm(Europe, text="TopLevelDomain")

## End(Not run)

approx_areas

Approximate area sizes of the shapes

Description

Approximate 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.
approx_areas

Usage

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

Arguments

shp shape object, i.e. a SpatialPolygonsDataFrame

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. In this case, the area sizes are also returned in squared kilometers.

"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.

Examples

data(World)

data(NLD_muni)

NLD_muni$area <- approx_areas(NLD_muni, total.area.km2 = 33893)

tm_shape(NLD_muni) +
tm_bubbles(size="area") +
tm_layout("Area in km2")
calc_densities  Calculate densities

Description

Transpose quantitative variables to density variables, which are often needed for choropleths. For example, the colors of a population density map should correspond population density counts rather than absolute population numbers.

Usage

calc_densities(shp, var, total.area.km2 = NA, units = "km2", suffix = "", drop = TRUE)

Arguments

shp  
a shape object, i.e. a SpatialPolygons(DataFrame)

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.

units  
the units of the area sizes, either "m2" or "km2". In other words, either var/m2 or var/km2 density values are calculated.

suffix  
character that is appended to the variable names. The resulting names are used as column names of the returned data.frame.

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 with append_data with fixed.order=TRUE.

Examples

data(NLD_muni)

NLD_muni_pop_per_km2 <- calc_densities(NLD_muni, var = c("pop_men", "pop_women"), suffix = "_km2")
NLD_muni <- append_data(NLD_muni, NLD_muni_pop_per_km2, fixed=TRUE)

qt(NLD_muni, fill=c("pop_men_km2", "pop_women_km2"), nrow=1)
convert_shape_data  

Convert shape data

Description

Convert numeric data from one polygon shape to another. It uses an intersection matrix, which stores the intersection ratios of the two shape objects per polygon (see `intersection_shapes`).

Usage

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

Arguments

- `shp.from`: the shape object, i.e. a `SpatialPolygonsDataFrame`, to be converted. It should contain data.
- `shp.to`: the shape object, i.e. a `SpatialPolygonsDataFrame`, to be converted into.
- `variables.from`: names of the numeric variables 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`

Value

Shape object `shp.to` with converted data from `shp.from`.

Note

Running time may be pretty slow.

Examples

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

## Compare to the original province population data
tm(NLD_muni, fill="population", convert2density = TRUE)
tm(NLD_prov, fill="population", convert2density = TRUE)

## For illustration, the population of provinces is derived from the municipality data
NLD_prov <- convert_shape_data(NLD_muni, NLD_prov, variables.from = "population")
tm(NLD_prov, fill="population.data", convert2density = TRUE)

## Now, the population of province level is spread equally over the municipalities
NLD_muni <- convert_shape_data(NLD_prov, NLD_muni, variables.from = "population")
tm(NLD_muni, fill="population.data", convert2density = TRUE)
```
crop_shape

## Description

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

## Usage

crop_shape(shp, bbox = shp@bbox, set.bbox = TRUE)

### Arguments

- **shp**: shape object, which is one of
  1. `SpatialPolygons(DataFrame)`
  2. `SpatialPoints(DataFrame)`
  3. `SpatialLines(DataFrame)`
- **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.
- **set.bbox**: logical that determines whether the bounding box of shp is set to bbox. If FALSE, the bounding box is determined by shp after cropping.

### 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 a attribute matchID. This vector contains for each polygon in the returned shape object the number of its original polygon in shp.

double_line

## Description

Create a double line from a single line. (Experimental, see note)

## Usage

double_line(shp, width)
**Arguments**

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

**Value**

SpatialLinesDataFrame

**Note**

This function is still in experimental phase, which means that it may not be stable and it may be changed significantly in future versions. Moreover, it is unsure if it will stay in tmap; instead, it may be put in a different package, along with functions of similar tasks.

---

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

**Description**

Fit one or more smooth polylines through a set of spatial points. (Experimental, see note)

**Usage**

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

**Arguments**

- **...**: Shape object(s) from which spatial points are taken. A shape object is one of
  1. SpatialPolygons(DataFrame)
  2. SpatialPoints(DataFrame)
  3. SpatialLines(DataFrame)
- **id**: Name of the data variable that determines the classes of the points. For each class a polyline is fit. Is omitted, a polyline is fit through all points.
- **min.dist**: Minimum distance. Points that are closer than min.dist from any other point are clustered (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
**get_IDs**

**Details**

Method: the coordinates from all shape objects are collected per category (defined by `id`). Per category, the following steps are taken: 1) coordinates that are closer than `min.dist` from each other are grouped, where the group means are considered as coordination points subsequently; 2) the coordination points are clustered by fitting a minimum spanning tree where edges longer than `sep.dist` are removed; 3) for each cluster, a graph is created by adding an edge for any two coordinate points that are at least `max.opt.dist` far away from each other. Also, the edges of a fitted minimum spanning tree are added to the graph; 4) for each cluster, the shortest path is found within the constructed graph between the two coordinate points that are farthest away from each other.

**Value**

*SpatialLinesDataframe*

**Note**

This function is still in experimental phase, which means that it may not be stable and it may be changed significantly in future versions. Moreover, it is unsure if it will stay in tmap; instead, it may be put in a different package, along with functions of similar tasks.

---

**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**

`get_IDs(shp)`

**Arguments**

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

**Value**

vector of ID’s
get_polygon_ranges

Get and set polygon direction

Description
Get and set the direction of polygons (clockwise or counterclockwise). (Experimental, see note)

Usage
get_polygon_directions(shp, y@_bottom = TRUE)
set_polygon_directions(shp, y@_bottom = TRUE, clockwise = TRUE)

Arguments
- shp: shape object, i.e. SpatialPolygons(DataFrame)
- y@_bottom: logical: is the y=0 coordinate at the bottom of the screen?
- clockwise: logical: should the direction of the polygons be set to clockwise?

Value
List of logical vectors, each corresponding to a Polygons object. The values of each vector correspond to the Polygon objects. TRUE means clockwise.

Note
This function is still in experimental phase, which means that it may not be stable and it may be changed significantly in future versions. Moreover, it is unsure if it will stay in tmap; instead, it may be put in a different package, along with functions of similar tasks.

generate_polygon

get_polygon_ranges

Get ranges of the polygons

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)
**intersection_shapes**

**Arguments**

- `shp`  
  shape object, i.e. a `SpatialPolygons(DataFrame)`

- `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.

**Description**

The value of row i and column j in the intersection matrix corresponds to the proportion of the area of the i’th polygon of the first shape object that intersects the j’th polygon of the second shape object.

**Usage**

```r
intersection_shapes(shp.from, shp.to, id.from = NULL, id.to = NULL, absolute = FALSE)
```

**Arguments**

- `shp.from`  
  the first shape object

- `shp.to`  
  the second shape object

- `id.from`  
  name of the data variable of `shp.from` that contains identification names of the polygons of `shp.from`. These are used as row names of the intersection matrix.

- `id.to`  
  name of the data variable of `shp.to` that contains identification names of the polygons of `shp.to`. These are used as column names of the intersection matrix.

- `absolute`  
  should the intersection matrix contain intersection area sizes rather than proportions?

**Value**

Intersection matrix with the number of rows equal to the number of polygons of `shp.from` and the number of columns equal to the number of polygons of `shp.to`.

**Note**

Running time may be pretty slow.
### Examples

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

m <- intersection_shapes(NLD_muni, NLD_prov, id.from="name", id.to="name")

## End(Not run)
```

### Description

Map points to a polyline. For each point, the closest point on the line and the corresponding distance is determined. At the moment, it only considers the begin- joint-, and endpoint coordinates of a line, not their interpolations. To take the interpolations into account, use `split_lines_equal`.

(Experimental, see note)

### Usage

```r
map_points_to_line(shp.points, shp.lines, by.key = FALSE, key.points = NULL, key.lines = NULL)
```

### Arguments

- `shp.points`: shape object containing the points, i.e. a `SpatialPoints(DataFrame)`
- `shp.lines`: shape object containing the polylines, i.e. a `SpatialLines(DataFrame)`
- `by.key`: should the points be mapped to the polylines by class (determined by `key.points` and `key.lines`)?
- `key.points`: name of a variable contained in `shp.points` that determined the classes of the points. The points are mapped to the polylines class-wise.
- `key.lines`: name of a variable contained in `shp.lines` that determined the classes of the polylines. The points are mapped to the polylines class-wise.

### Value

`data.frame`, where the rows correspond to the points. The columns are: 1) `id1`, which is the Lines identifier in which the coordination point is contained, 2) `id2`, which is the Line identifier in which the coordination point is contained, 3) `id3`, which is the number of the closest coordination point in the polyline, 4) the distance with the closest coordination point.

### Note

This function is still in experimental phase, which means that it may not be stable and it may be changed significantly in future versions. Moreover, it is unsure if it will stay in `tmap`; instead, it may be put in a different package, along with functions of similar tasks.
**print.tmap**

*Draw tmap plot*

**Description**

Draw tmap plot on current graphics device

**Usage**

```r
## S3 method for class 'tmap'
print(x, vp = NULL, ...)
```

**Arguments**

- `x` tmap object. A tmap object is created with `qtm` or by stacking `tmap-elements`.
- `vp` viewport to draw the plot in
- `...` not used

---

**qtm**

*Quick thematic map plot*

**Description**

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

**Usage**

```r
qtm(shp, fill = "grey85", bubble.size = NULL, bubble.col = NULL,
    text = NULL, text.cex = 1, line.lwd = NULL, line.col = NULL,
    borders = "grey40", theme = NULL, scale = NA, ...)
```

**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 controled by this value. The parameters bubble.size, text.cex, and line.lwd can be scaled seperately with respectively bubble.scale, text.scale, and line.scale.
...  arguments passed on to the tm_* functions. If an argument name is not unique for a particular tm_ function, then it should be prefixed with the function name without "tm_". For instance, style is an argument of tm_fill, tm_bubbles, and tm_lines. Therefore, in order to define the style for a choropleth, its argument name should be fill.style.

Value
tmap-element

See Also
vignette("tmap-nutshell")

Examples
data(Europe)
data(World)
data(cities)

# just the map
qtm(Europe)

# choropleth
qtm(World, fill = "economy", text = "iso_a3", text.cex = "AREA",
fill.palette="-Blues", theme = "World", title="Economy")

qtm(Europe, fill="gdp_cap_est", text="iso_a3", text.cex="pop_est",
title="GDP per capita", fill.textNA="Non-European countries")

qtm(World, fill="pop_est_dens", theme="World", fill.style="kmeans", title="Population per km")

# bubble map
qtm(World, borders = NA) + qtm(cities, bubble.size = "pop_max", bubble.col="purple",
title="Cities of the World", theme = "World", bubble.scale=.5)
Description
Read an ESRI shape file. Optionally, set the current projection if it is missing.

Usage
read_shape(file, current.projection = NULL)

Arguments
file a shape file name (including directory).
current.projection
the current projection of the shape object, if it is missing in the shape file.
It should be 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 characte string with the zone number. For southern hemisphere, add "s". So, for instance, the Netherlands is "utm31" and New Zealand is "utm59s"
"mill" Miller (1942). Projection based on Mercator, in which poles are displayed. Type: compromise
"eqc" 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.


Details

This function is a convenient wrapper of readOGR of the package rgdal. It is possible to set the current projection, if it is undefined in the shape file. If a reprojection is required, use set_projection.

Value

shape object

Note

For the Netherlands: often, the Dutch Rijksdriehoekstelsel (Dutch National Grid) projection is provided in the shape file without proper datum shift parameters to wgs84. This functions automatically adds these parameters. See http://www.qgis.nl/2011/12/05/epsg28992-of-rijksdriehoekstelsel-verschuiving/ (in Dutch) for details.

---

rivers | Spatial data of cities and rivers

Description

Spatial data of main world cities and rivers.

Usage

data(rivers)

data(cities)

Source

http://www.naturalearthdata.com
sbind

Combine shape objects

Description

Combine shape objects into one shape object. It works analogous to `rbind`.

Usage

sbind(...)

Arguments

... shape objects. Each shape object is one of

1. `SpatialPolygons(DataFrame)`
2. `SpatialPoints(DataFrame)`
3. `SpatialLines(DataFrame)`

Value

shape object

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

```r
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 or 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 68 174 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"
  "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.

**split.SpatialPolygonsDataFrame**

**make.EPSG.** 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.SpatialPolygonsDataFrame**

*Divide into multiple shape objects*

**Description**

Divide shape object into multiple objects.

**Usage**

```r
## S3 method for class 'SpatialPolygonsDataFrame'
split(x, f, drop = FALSE, ...)
```

```r
## S3 method for class 'SpatialPointsDataFrame'
split(x, f, drop = FALSE, ...)
```

```r
## S3 method for class 'SpatialLinesDataFrame'
split(x, f, drop = FALSE, ...)
```

**Arguments**

- **x**: shape object, which is one of
  1. `SpatialPolygons(DataFrame)`
  2. `SpatialPoints(DataFrame)`
  3. `SpatialLines(DataFrame)`
- **f**: factor to split x
- **drop**: unused factor levels are dropped
- **...**: other arguments (not used)
split_lines_equal

*Split lines in segments of equal length*

**Description**

Split lines in segments of equal length. (Experimental, see note)

**Usage**

\[
\text{split_lines_equal}(\text{shp}, \text{dist} = 1000, \text{include.last} = \text{FALSE})
\]

**Arguments**

- **shp**: The shape object that contains the lines, i.e. \text{SpatialLinesDataFrame}
- **dist**: Distance per segment
- **include.last**: Include last point, even though the distance is less than dist from the previous point?

**Value**

Shape object, i.e. \text{SpatialPolygons(DataFrame)}.

**Note**

This function is still in experimental phase, which means that it may not be stable and it may be changed significantly in future versions. Moreover, it is unsure if it will stay in tmap; instead, it may be put in a different package, along with functions of similar tasks.

split_lines_poly

*Split lines by polygons*

**Description**

Split a lines shape object by a polygon shape object. Data of the corresponding polygons is appended to the line segments (Experimental, see note)

**Usage**

\[
\text{split_lines_poly}(\text{shp.lines, shp.poly, variables.lines, variables.poly})
\]
Arguments

shp.lines The shape object that contains the lines, i.e. a SpatialLinesDataFrame
shp.poly The shape object that contains the polygons, i.e. a SpatialPolygonsDataFrame
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, a SpatialLinesDataFrame

Note

This function is still in experimental phase, which means that it may not be stable and it may be changed significantly in future versions. Moreover, it is unsure if it will stay in tmap; instead, it may be put in a different package, along with functions of similar tasks.

tmap-element     tmap element

Description

Building block for drawing thematic maps.

Details

The only fundamental, and hence required element is

• \texttt{tm\_shape} that specifies the shape object, and also controls the projection and bounding box.

The elements that serve as drawing layers are

• \texttt{tm\_borders} to draw polygon borders
• \texttt{tm\_fill} to fill the polygons
• \texttt{tm\_bubbles} to draw bubbles
• \texttt{tm\_lines} to draw lines
• \texttt{tm\_text} to print text

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 \texttt{tm\_shape} (see examples below).

The layout elements are
• **tm_layout** to change the appearance of the map, for instance titles, legend positions, and margins. 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 coordinate grid lines

### Examples

```r
data(World)
data(Europe)
data(NLD_muni)
data(NLD_prov)
data(cities)
qtm(Europe)

qtm(World, fill = "economy", text = "iso_a3", text.cex = "AREA", fill.palette = "Blues", theme = "World", title = "Economy")

tm_shape(World) +
  tm_fill("pop_est_dens", style = "kmeans", palette = "YlOrRd") +
  tm_borders() +
  tm_text("iso_a3", cex = "AREA", cex.lowerbound = .4) +
  tm_layout_World(title = "Population density per km2")

tm_shape(Europe) +
  tm_fill() +
  tm_shape(cities) +
  tm_bubbles(size = "pop_max", col = "capital", size.lim = c(0, 2e7)) +
  tm_text("name", cex = "pop_max", scale = 2, root = 3, ymod = -.015, bg.alpha = 0) +
  tm_layout_Europe("Metropolitan population", legend.titles = c("bubble.col = "Capital"))

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", free.coords = TRUE, drop.shapes = TRUE) +
  tm_layout(legend.show = FALSE)
```

---

**tmBorders**

**Draw polygon borders**

### Description

Creates a tmap-element that defines the borders of the polygons. Line color, width, and type can be set.
tm_bubbles

Usage

tm_borders(col = "grey40", lwd = 1, lty = "solid", alpha = NA)

Arguments

- **col**: line color
- **lwd**: line width (see `par`)
- **lty**: line type (see `par`)
- **alpha**: transparency number between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the `col` is used (normally 1).

Value

`tmap-element`

See Also

`vignette("tmap-nutshell")`

Examples

```r
data(Europe)
data(NLD_prov)
data(NLD_muni)

tm_shape(Europe) + tm_borders()

tm_shape(NLD_muni) +
   tm_fill(col="population", convert2density=TRUE, style="kmeans") +
   tm_borders("grey25", alpha=.5) +
   tm_shape(NLD_prov) +
   tm_borders("grey40", lwd=2) +
   tm_layout_NLD(title="Population (per km2)", bg.color="white", draw.frame = FALSE)
```

---

tm_bubbles

**Draw bubbles**

Description

Creates a `tmap-element` that draws bubbles. Both colors and sizes of the bubbles can be mapped to data variables.
Usage

```r
tm_bubbles(size = 1, col = "blueviolet", alpha = NA, border.col = NA, border.lwd = 1, border.alpha = NA, scale = 1, size.lim = NA, sizes.legend = NULL, sizes.legend.labels = NULL, 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.
- `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 bubble borders.
- `border.lwd` line width of the bubble borders. If NA (default), no bubble borders are drawn.
- `border.alpha` transparency number, regarding the bubble borders, between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the `col` is used (normally 1).
- `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`.
- `sizes.legend` vector of bubble 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 cut the color scale: e.g. "fixed", "equal", "pretty", "quantile", or "kmeans". See the details in `classIntervals`. 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
A 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` move them to the bottom.

### ymod
Vertical position modification. See `xmod`.

### Value

- `tmap_element`

### See Also

- `vignette("tmap-nutshell")`

### Examples

```r
data(World)
data(Europe)
data(cities)

tm_shape(World) +
  tm_fill() +
  tm_shape(cities) +
  tm_bubbles("pop_max", scale=0.5) +
  tm_layout_World("Cities of the World")

tm_shape(Europe) +
  tm_borders() +
  tm_fill() +
  tm_shape(cities) +
  tm_bubbles(size="pop_max", col="capital", size.lim=c(0, 2e7)) +
  tm_text("name", cex="pop_max", scale=2, root=3, ymod=-.015, bg.alpha=0) +
  tm_layout_Europe("Metropolitan population", legend.titles=c(bubble.col="Capital"))
```
tm_facets

Small multiples

Description

Creates a tmap-element that specifies how small multiples are placed in a facet 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 tm_fill, tm_lines, or 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

tm_facets(by = NULL, ncol = NULL, nrow = NULL, free.coords = FALSE, drop.shapes = FALSE, 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, scale.factor = 2)

Arguments

by data variable name by which the data is split
ncol number of columns of the small multiples grid
nrow number of rows of the small multiples grid
free.coords logical. If the by argument is specified, should each map has its own coordinate ranges?
drop.shapes logical. If the by argument is specified, should all non-selected shapes be dropped?
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_bubbles and line color from tm_lines.
free.scales.fill logical. Should the color scale for the choropleth be free?
free.scales.bubble.size logical. Should the bubble size scale for the bubble map be free?
free.scales.bubble.col logical. Should the color scale for the bubble map be free?
free.scales.line.col Should the line color scale be free?
free.scales.line.lwd Should the line width scale be free?
scale.factor Number that determines how the elements (e.g. font sizes, bubble sizes, line widths) of the small multiples are scaled in relation to the scaling factor of the shapes. The elements are scaled to the scale.factor*th root of the scaling factor of the shapes. So, for scale.factor=1, they are scaled proportional to the scaling of the shapes. Since elements, especially text, are often too small to read, a higher value is recommended. By default, scale.factor=2.
Value
tmap-element

See Also
vignette("tmap-nutshell")

Examples
data(World)
data(Europe)
data(NLD_muni)
data(NLD_prov)

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

# Facets defined by multiple variables
tm_shape(Europe) +
   tm_borders() +
   tm_fill(c("gdp_cap_est", "pop_est_dens"), style="kmeans") +
tm_layout_Europe(c("GDP per capita", "Population density"))

tm_shape(NLD_muni) +
   tm_fill(c("pop_0_14", "pop_15_24", "pop_25_44", "pop_45_64", "pop_65plus"),
          style="kmeans") +
tm_shape(NLD_prov) +
   tm_borders() +
tm_layout_NLD(c("Population 0 to 14", "Population 15 to 24", "Population 25 to 44",
                  "Population 45 to 64", "Population 65 and older"), draw.frame = TRUE, asp=0)

# Facets defined by groupings
tm_shape(NLD_prov) +
   tm_borders() +
   tm_fill("gold2") +
tm_facets(by="name") +
tm_layout()

tm_shape(NLD_prov) +
   tm_fill("gold2") + tm_borders() +
tm_facets(by="name", free.coords = TRUE, drop.shapes=TRUE) +
tm_layout()

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", free.coords=TRUE, drop.shapes=TRUE) +
tm_layout(legend.show = FALSE)

---

### tm_fill

**Fill polygons**

#### Description

Creates a tmap-element that fills polygons. Either a fixed color is used, or a color palette is mapped to a data variable. By default, a divering color palette is used for numeric variables and a qualitative palette for categorical variables.

#### Usage

```r
tm_fill(col = "grey85", alpha = NA, palette = NULL,
convert2density = FALSE, area = NULL, n = 5, style = "pretty",
breaks = NULL, labels = NULL, auto.palette.mapping = TRUE,
contrast = 1, max.categories = 12, colorNA = "grey60",
textNA = "Missing", thres.poly = 1e-05)
```

#### Arguments

- **col**: either a single color value or the name of a data variable that is contained in `shp`. In the latter case, 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.
- **alpha**: transparency number between 0 (totally transparent) and 1 (not transparent). By default, the alpha value of the `col` is used (normally 1).
- **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.
- **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 cut the color scale (in case `col` is a numeric variable): e.g. "fixed", "equal", "pretty", "quantile", or "kmeans". See the details in classIntervals.
- **breaks**: in case `style="fixed"`, breaks should be specified
- **labels**: labels of the classes
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**

data(World)
data(Europe)
data(NLD_muni)
data(NLD_prov)

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

# Data variable containing colours values
Europe$isNL <- ifelse(Europe$name=="Netherlands", "darkorange", "darkolivegreen3")
tm_shape(Europe) +
 tm_fill("isNL") +
 tm_layout("Find the Netherlands!")

# Numeric data variable
tm_shape(Europe) +
 tm_fill("gdp_cap_est", style="kmeans", textNA = "Non-European countries") +
 tm_borders() +
 tm_text("iso_a3", cex="AREA", root=4, scale=2) +
 tm_layout_Europe("GDP per capita")

tm_shape(World) +
 tm_fill("pop_est_dens", style="kmeans", palette="YlOrRd") +
tm_borders() +
  tm_text("iso_a3", cex="AREA", cex.lowerbound=.4) +
tm_layout_World(title="Population density per km2")

# Categorical data variable
tm_shape(World) +
  tm_fill("income_grp", palette="-Blues") +
  tm_borders() +
  tm_text("iso_a3", cex="AREA", scale=1.5) +
tm_layout_World("Income classification")

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

---

**tm_grid**

*Coordinate grid lines*

**Description**

Creates a **tmap-element** that draws coordinate grid lines.

**Usage**

```r
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.
- **col**  
  Color for the grid lines.
- **labels.cex**  
  Font size of the tick labels
- **labels.col**  
  Font color for the tick labels
- **on.top**  
  Boolean that determines whether the grid lines are drawn on top of the map (TRUE) or under the map (FALSE).
tm_layout

Layout elements of cartographic maps

Description

This element specifies layout options for the maps. The main function `tm_layout` can be seen as a general layout theme. The functions `tm_layout_World`, `tm_layout_Europe`, and `tm_layout_NLD` are layout themes specified for the world, Europe, and Netherlands maps (which are contained in this package). Tip: create a layout theme for your own map (see example below).

Usage

```
# Default settings
tm_layout(title = NA, scale = 1, title_cex = 1.5, bg_color = NULL, 
          draw_frame = TRUE, title_position = NULL, 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 = NULL, 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_scientific = FALSE, 
          legend_digits = NA, legend_bg_color = NA)

# World

# Europe

# Netherlands
```

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 frame 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 "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 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 determined 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 variable.

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 correspond 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 element. 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 specified.

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 "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 legend. By default, it is automatically placed in the corner with most space based on the (first) shape object.


```
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 po-
  sition. 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 corre-
  sponds 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.scientific
  logical. Should the numeric legend labels be formatted scientific?
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")

Examples

data(World)

  tm_shape(World) + tm_fill("pop_est_dens", style="kmeans") +
  tm_layout_World("World Population")

  # A custom layout wrapper for Africa
  tm_layout_Africa <- function(title=NA,
    inner.margins = c(.02, .4, .02, .02),
    draw.frame = FALSE,
    title.position="left", "bottom")t
  legend.position = c("left", "bottom"),
  bg.color = "lightskyblue2", ...)
```
### tm_lines

Draw spatial lines

**Description**

Creates a tmap-element that draw spatial lines.

**Usage**

```r
tm_lines(col = "red", lwd = 1, lty = "solid", alpha = NA, 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
- **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.
- **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: e.g. "fixed", "equal", "pretty", "quantile", or "kmeans". See the details in classIntervals. 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.
**tm_shape**

 Specifies the shape object

**Description**

Creates a tmap-element that specifies the shape object. 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**

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

**Arguments**

- `shp`: Shape object
- `projection`: Projection
- `xlim`, `ylim`: Limits
- `relative`: Relative
- `bbox`: Bounding box

**Value**

 tmap-element

**See Also**

 vignette("tmap-nutshell")

**Examples**

```r
data(rivers)
data(World)
data(Europe)

tm_shape(World) +
tm_fill("darkolivegreen3") +
tm_shape(rivers) +
tm_lines(col="navy") +
tm_layout_World("Rivers of the World", inner.margins = c(0,0,.1,0))

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

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

For drawing layers `tm_fill` and `tm_borders`, 1 is required. For drawing layer `tm_lines`, 3 is required. Layers `tm_bubbles` and `tm_text` accept any of them.

projection  character that determines the projection. Either a PROJ.4 character string or 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
  "utmX(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"
  "m111" 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 a overview of projections. See [http://trac.osgeo.org/proj/](http://trac.osgeo.org/proj/) for the PROJ.4 project home page. An extensive list of PROJ.4 codes can be created with rgdal's `make_EPSG`. By default, the projection is used that is defined in the shp object itself, which can be obtained with `get_projection`.

### xlim

limits of the x-axis. These are either absolute or relative (depending on the argument `relative`). Alternatively, the argument `bbox` can be set to set absolute values.

### ylim

limits of the y-axis. See `xlim`.

### relative

boolean that determines whether relative values are used for `xlim` and `ylim` or absolute. Note: in case multiple shape objects are used within one plot, the 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 both `xlim` and `ylim`.

### Value

tmap-element

### See Also

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

### Examples

```r
data(World)
data(cities)
data(rivers)

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)

# three groups of layers, each starting with tm_shape
tm_shape(World) +
tm_fill("darkolivegreen3") +
tm_shape(cities) +
tm_bubbles("pop_max", col = "grey30", scale=.5) +
tm_shape(rivers) +
tm_lines("lightcyan1") +
tm_layout(World(title = "", bg.color="lightcyan1", legend.show = FALSE)

---

**tm_text**

*Add text labels*

**Description**

Creates a tmap-element that adds text labels.

**Usage**

```r
tm_text(text, cex = 1, root = 3, fontcolor = NA, fontface = "plain",
     fontfamily = "sans", alpha = NA, case = NA, shadow = FALSE,
     bg.color = NA, bg.alpha = NA, 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 (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.*

- **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**  
  *color of the text labels*

- **fontface**  
  *font face of the text labels*

- **fontfamily**  
  *font family of the text labels*

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

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

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

text

bg.color

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

text

bg.alpha

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

text

cex.lowerbound

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

text

print.tiny

boolean, see cex.lowerbound

text

scale

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

text

xmod

horizontal position modification of the text (relatively): 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.

text

ymod

vertical position modification. See xmod.

text

Value


tmap-element

Note

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

See Also

vignette("tmap-nutshell")

Examples

# World example
data(World)
tm_shape(World) +
tm_text("name", cex="AREA")

# Europe example
data(Europe)
tm_shape(Europe) +
  tm_fill() +
  tm_borders() +
  tm_text("iso_a3", cex="AREA", root=4, shadow = TRUE, fontcolor = "grey20", scale=2,
cex.lowerbound = .1) +
tm_shape(Europe) +
tm_text("name", cex="AREA", root=4, scale=1,
  ymod=-.04 * approx_areas(Europe, units = "norm")^(1/4))

<table>
<thead>
<tr>
<th>World</th>
<th>World, Europe, and Netherlands map</th>
</tr>
</thead>
</table>

Description
Maps of the world, Europe, and the Netherlands (province and municipality level).

Usage
- data(World)
- data(Europe)
- data(NLD_prov)
- data(NLD_muni)

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

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

Europe Europe map. Lambert azimuthal equal-area projection is used by default for this map. Several countries are transcontinental and are partly located in Asia. From these countries, only Russia and Turkey have been included in this map as part of Europe since they are widely considered as European countries. Other transcontinental countries Azerbaijan, Georgia, and Kazakhstan, are also included in the map, but only as background (so without data). From the other surrounding countries, only Greenland is removed from the map, since it interferes with the preferred map title at the lop left.

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

Source
- http://www.naturalearthdata.com for World and Europe
- Statistics Netherlands (http://www.cbs.nl) for NLD_prov and NLD_muni.
write_shape

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write a shape object to an ESRI shape file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>write_shape(shp, file)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>This function is a convenient wrapper of writeOGR from the package rgdal.</td>
</tr>
</tbody>
</table>
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, 7
*Topic maps, tmap-package, 2
*Topic map
  tmap-package, 2
*Topic statistical
  tmap-package, 2
*Topic thematic
  tmap-package, 2
+ Ntmap
  animation_tmap, 3
append_data, 4, 7
approx_areas, 5
calc_densities, 7
cities (rivers), 18
classIntervals, 26, 30, 36
convert_shape_data, 8
crop_shape, 9
double_line, 9
Europe (World), 42
fit_polylines, 10
get_IDs, 11, 13
get_polygon_directions, 12
get_polygon_ranges, 12
get_projection, 39
get_projection (set_projection), 19
intersection_shapes, 8, 13
make_EPSG, 21, 39
map_points_to_line, 14
NLD_muni (World), 42
NLD_prov (World), 42
par, 25
print.tmap, 15
qtm, 15, 15, 42
rbind, 19
read_shape, 17, 39
rivers, 18
sbind, 19
set_polygon_directions
  (get_polygon_directions), 12
set_projection, 18, 19, 39, 42
SpatialLines, 16
SpatialLines(DataFrame), 5, 9–11, 14, 19–21, 38
SpatialLinesDataFrame, 10, 16, 22, 23
SpatialPoints, 15
SpatialPoints(DataFrame), 5, 9–11, 14, 19–21, 38
SpatialPointsDataFrame, 15
SpatialPolygons(DataFrame), 5–13, 19–22, 38
SpatialPolygonsDataFrame, 15, 23
split.SpatialLinesDataFrame
  (split.SpatialPolygonsDataFrame), 21
split.SpatialPointsDataFrame
  (split.SpatialPolygonsDataFrame), 21
split.SpatialPolygonsDataFrame, 21
split_lines_equal, 14, 22
split_lines_poly, 22
spTransform, \textit{19}

tm\_borders, \textit{23, 24, 38}
tm\_bubbles, \textit{15, 16, 23, 25, 28, 38}
tm\_facets, \textit{4, 24, 28}
tm\_fill, \textit{15, 16, 23, 28, 30, 38}
tm\_grid, \textit{24, 32}
tm\_layout, \textit{24, 33}
tm\_layout\_Europe, \textit{24}
tm\_layout\_Europe (tm\_layout), \textit{33}
tm\_layout\_NLD, \textit{24}
tm\_layout\_NLD (tm\_layout), \textit{33}
tm\_layout\_World, \textit{24}
tm\_layout\_World (tm\_layout), \textit{33}
tm\_lines, \textit{16, 23, 28, 36, 38}
tm\_shape, \textit{19, 23, 34, 37, 42}
tm\_text, \textit{23, 38, 40}
tmap (tmap-package), \textit{2}
tmap-element, \textit{23}
tmap-package, \textit{2}

viewport, \textit{15}

World, \textit{42}
write\_shape, \textit{43}