Thematic Map Visualization: tmap

Thematic maps are geographical maps in which spatial data distributions are visualized.

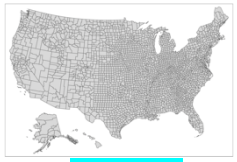
Quick Plotting method:	
qtm:	plot a thematic map
Main Plotting method:	
Aesthetics derived layers:	
tm_shape:	specify a shape object
Aesthetics derived layers:	
tm_polygons:	create polygon layer(with borders)
tm_symbols:	create a layer of symbols
tm_lines:	create a layer of lines
tm_raster:	create a layer of text labels
tm_text:	create a layer of text labels
tm_basemap:	create a layer of basemap tiles
tm_tiles:	create a layer of overlay tiles
Aesthetics derived layers:	
tm_fill:	create a polygon layer (without borders)
tm_borders:	create polygon borders
tm_bubbles:	create a layer of bubbles
tm_squares:	create a layer of squares
tm_dots:	create a layer of dots
tm_markers:	create a layer of markers
tm_iso:	create a layer of iso/contour lines
tm_rgb:	create a raster layer of an image
Faceting (small multiples)	
tm_facets:	define facets
Attributes	
tm_grid:	create grid lines
tm_scale_bar:	create a scale bar
tm_compass:	create a map compass
tm_credits:	create a text for credits
tm_logo :	create a logo
tm_xlab and tm_ylab:	create axis labels
tm_minimap:	create a minimap (view mode only)
Layout element:	
tm_layout:	Adjust the layout (main function)
tm_legend:	Adjust the legend
tm_view:	Configure the interactive view mode
tm_style:	Apply a predefined style
tm_format:	Apply a predefined format
Change options:	
tmap_mode	Set the tmap mode: "plot" or "view"
ttm	Toggle between the modes
tmap_options	Set global tmap options (from tm_layout, tm_view, and a couple of others)
tmap_style	Set the default style
Create icons:	
tmap_icons	Specify icons for markers or proportional symbols
Ouput functions	
print	Plot in graphics device or view interactively in web browser or RStudio's viewer pane
tmap_last	Redraw the last map
tmap_leaflet	Obtain a leaflet widget object
tmap_animation	Create an animation
tmap_arrange	Create small multiples of separate maps
tmap_save	Save thematic maps (either as image or HTML file)
Spatial datasets	
World	World country data (sf object of polygons)
NLD_prov	Netherlands province data (sf object of polygons)
NLD_muni	Netherlands municipal data (sf object of polygons)
metro	Metropolitan areas (sf object of points)
rivers	Rivers (sf object of lines)
land	Global land cover (stars object)

Practical Examples:

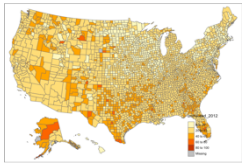
Super easy mapping

A) The easiest possible map, just the geography:

: tm_shape(shp) + tm_polygons()



B) Add a variable to your map:



(Code): tm_shape(shp) + tm_polygons("uninsured_2012")

C) Change the shape: The package provides a lot of flexibility in terms of different approaches to mapping your data. Here we use bubbles in place of polygons and note that no additional data processing is required.

(Code): tm_shape(shp) + tm_bubbles("uninsured_2012")

D) Include multiple layers:

(Code): dat <- data.frame(c("Empire State Building"), lat = c(40.748595), long = c(-73.985718))
sites <- sf::st_as_sf(dat, coords = c("long", "lat"), crs = 4326, agr = "identity")
tm_shape(shp) + tm_polygons() + tm_shape(sites) + tm_dots(size = 2)

Projecting data on-the-fly: Probably one of the nicest features of tmap is the ability to project data on-the-fly. Simply use the projection argument in the tm_shape function to assign one of the following:

(Code): wintri = "+proj=utm +zone=12 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs: NAD83/UTM zone 12N"
tm_shape(shp, projection = wintri) +
tm_polygons()

Simplify polygons or lines: The simplify_shape function is incredibly useful, particularly if you work with detailed linework or polygons which can be cumbersome to work with and time consuming to map. The function is from the tmaptools package and uses the argument fact, the simplification factor, to adjust the number of coordinates in the given spatial object. If you do a lot of simplication you can also review the st_simplify() function in sf and, for the greatest control, the functions in the package rmapshaper.

(Code): library(rmapshaper)
# Simplify shapes with tmap function
shpsimp <- simplify_shape(shp, fact = 0.05); shpsimp

Working with colors and cuts

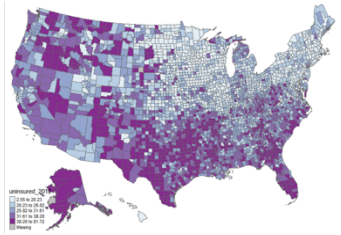
A) Built-in colors and cuts: The tmap package makes it very easy to color and classify our data using the style and palette arguments.

- Style option: quantile, jenks, pretty, equal, sd
- Palette option: BuPu, OrRd, PuBuGn, YlOrRd

EXAMPLE:

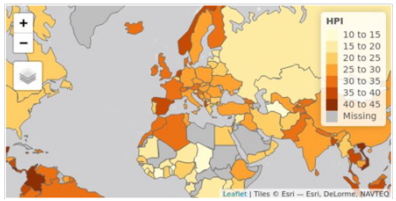
(Code): var <- "uninsured_2012"

tm_shape(shp, projection = 2163) +
tm_polygons(var,
style = "quantile",
palette = "BuPu") +
tm_legend(legend.position = c("left", "bottom"))



Interactive maps:

Each map can be plotted as a static image or viewed interactively using "plot" and "view" modes, respectively. The mode can be set with the function tmap_mode, and toggling between the modes can be done with the 'switch' ttm() (which stands for toggle thematic map.

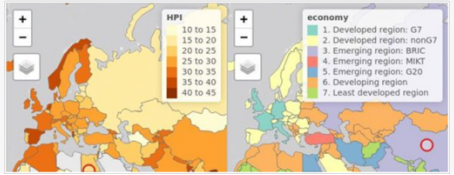


(Code):
tmap_mode("view")
tm_shape(World) +
tm_polygons("HPI")

Working with Facets

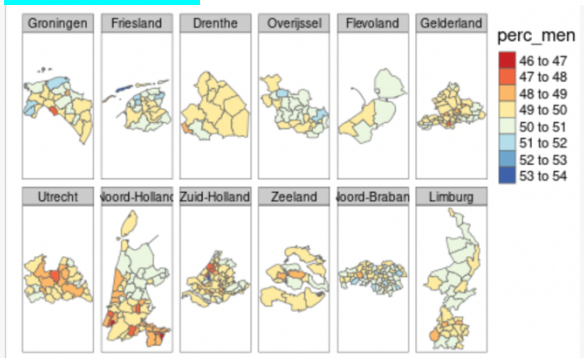
1. By assigning multiple variable names to one aesthetic (in this example the first argument of tm_polygons:

(Code): tmap_mode("view")
tm_shape(World) +tm_polygons(c("HPI", "economy")) +tm_facets(sync = TRUE, ncol = 2)



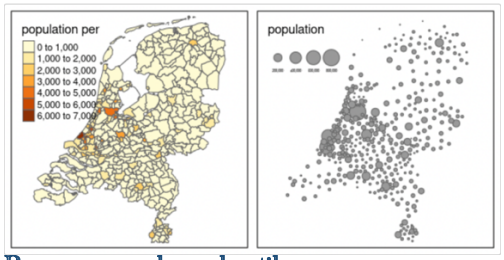
2.By splitting the spatial data with the by argument of tm_facets:

(Code): tm_shape(NLD_muni) +
tm_polygons("perc_men", palette = "RdYlBu") +
tm_facets(by = "province")



3.By using the tmap_arrange function:

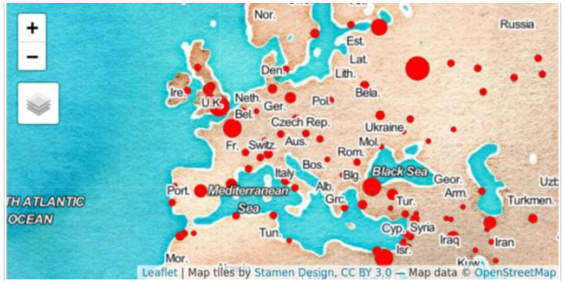
(Code): tm1 <- tm_shape(NLD_muni) + tm_polygons("population", convert2density = TRUE)
tm2 <- tm_shape(NLD_muni) + tm_bubbles(size = "population")
tmap_arrange(tm1, tm2)



Basemaps and overlay tile maps

1. Tiled basemaps can be added with the layer function tm_basemap. Semi-transparent overlay maps (for example annotation labels) can be added with tm_tiles.

(Code): tmap_mode("view")
tm_basemap("Stamen.Watercolor")+ tm_shape(metro) + tm_bubbles(size = "pop2020", col = "red") + tm_tiles("Stamen.TonerLabels")



2. Quick thematic map

Maps can also be made with one function call: `qtm function`

```
qtm(World, fill = "HPI", fill.pallete = "RdYlGn")
```