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

Note: to get the "shp" data, please visit at http://zevross.com/blog/2018/10/02/creating-beautiful-demographic-maps-in-r-with-the-tidycensus-and-tmap-packages/#part-2-creating-beautiful-maps-with-tmap

A) The easiest possible map, just the geography:

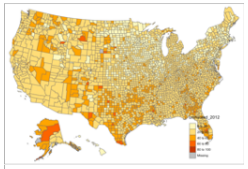
Define the shape and the layer elements

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

B) Add a variable to your map:

Get a map of the 2012 data using all of the tmap defults

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



C) Change the shape:

Use bubbles in place of polygons

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

D) Include multiple layers:

Add location of Empire State Building to the map

(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)

E)Projecting data on-the-fly (Winkel-Tripel example):

Make the map on the view of on-the-fly. Use the "projection" argument in the "tm_shape" function

(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()

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.

* Some Style options: quantile, jenks, pretty, equal, sd
* Some Palette options: BuPu, OrRd, PuBuGn, YlOrRd

Note: With "shiny" and "shinyjs" package, run "display.brewer.all()" to view the Color Brewer Plattes.

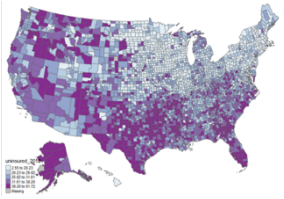
Example: BuPu color scheme with quantile classification

(Code): var <- "uninsured_2012"

tm_shape(shp, projection = 2163)+

tm_polygons(var, style = "quantile", palette = "BuPu")

+ tm_legend(legend.position = c("left", "bottom"))



B) User-defined classification:

For controlling the cut points, drop the "style" argument and use breaks.

Note: we changed the color of the county outlines and added a little transparency for not as overwhelming.

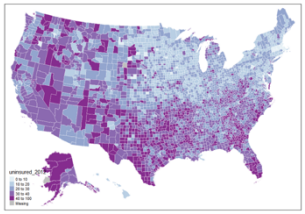
(code): cuts <- c(0, 10, 20, 30, 40, 100)

tm_shape(shp, projection = 2163) +

tm_polygons(var, breaks = cuts,

palette = "BuPu", border.col = "white",

border.alpha = 0.5) + tm_legend(legend.position = c("left", "bottom"))



C) Additional color option:

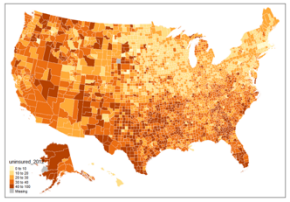
Example 1: Apply type of palette instead of palette scheme

If you don't know exactly which color scheme to use but want to apply a *sequential* palette, use palette = "seq". This will apply colors from the first *sequential* set of colors in the RColorBrewer color schemes

(code): tm_shape(shp, projection = 2163)

+ tm_polygons(var, breaks = cuts, palette = "seq",

border.col = "white", border.alpha = 0.5) + tm_legend(legend.position = c("left", "bottom"))



Example 2: Reverse the color scheme

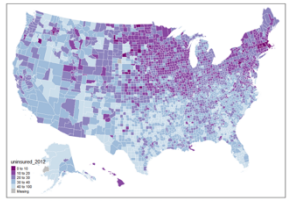
Reverse the "BuPu" color schemes with a simple "-".

(code): tm_shape(shp, projection = 2163) +

tm_polygons(var, breaks = cuts,

palette = "-BuPu", border.col = "white",

border.alpha = 0.5) + tm_legend(legend.position = c("left", "bottom"))

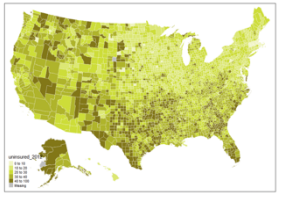


Example 3: Choose custom colors :

Assign colors outside of "RColorBrewer": create a vector of HEX and apply to the "palette" argument.

(code): mycols <- c("#0f4c3", "#dce775", "#cddc39", "#afb42b", "#827717")

tm_shape(shp, projection = 2163) + tm_polygons(var, breaks = cuts, palette = mycols, border.col = "white", border.alpha = 0.5) + tm_legend(legend.position = c("left", "bottom"))



Customizing layout features and adding attributes

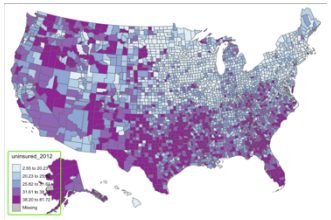
A) Add titles to the map

The **main title** is controlled by the "title" argument in "tm_layout".

The **legend title** is controlled by the "title" argument in the layer.

(Code): mymap <- tm_shape(shp, projection = 2163) + tm_polygons(var, breaks = cuts, palette = "BuPu", border.col = "white", border.alpha = 0.5, title = "Uninsured (%)") + tm_legend(legend.position = c("left", "bottom")) + tm_layout(title = "Uninsured adults ages 18-34 by county, 2012", title.size = 1.1, title.position = c("center", "top"))

mymap

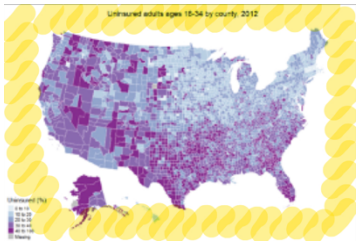


B) Increase the map margins (margins inside the frame)

The default value for the inner margins = 0.02.

Note: The order of inner.margins inputs is bottom, left, top and right. Values can be between 0 and 1.

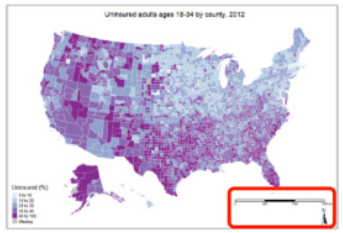
(Code): mymap + tm_layout(inner.margins = c(0.06, 0.10, 0.10, 0.08))



C) Add a scalebar and north arrow: the defaults

The default location for both the scalebar and north arrow is the bottom-right corner.

(Code): mymap + tm_scale_bar() + tm_compass()

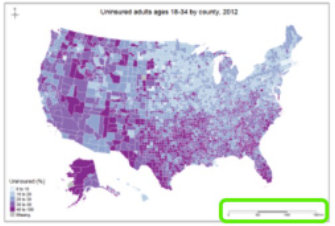


C) Add a scalebar and north arrow: customized

Make the scalebar to show units in miles, not kilometers. To do this we'll need to add the "unit" argument to the "tm_shape" function (not the "tm_compass" function).

(Code):

# Add unit argument to tm_shape
tm_shape(shp, projection = 2163, unit = "mi")
# Customize scale bar, north arrow
mymap + tm_scale_bar(color.dark = "gray60", position = c("right", "bottom")) + tm_compass(type = "4star", size = 2.5, fontsize = 0.5, color.dark = "gray60", text.color = "gray60", position = c("left", "top"))



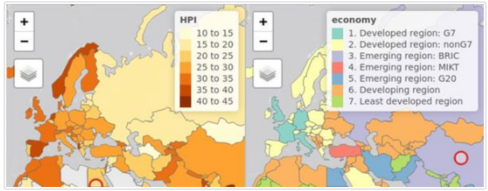
Working with Facets

Facets can be created in three ways:

A) By assigning multiple variable names to one aesthetic

(Code): tmap_mode(" view")

tm_shape(World)+tm_polygons(x(" HPI","economy"))+ tm_facets(sync=T,ncol=2)



B) By splitting the spatial data with the "by" argument of "tm_facets"

(Code): tmap_mode("plot"); data(NLD_muni)

NLD_muni$perc_men <-NLD_muni$pop_men /NLD_muni$population *100

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

C) By using the "tmap_arrange" function

tm1 <-tm_shape(NLD_muni) +tm_polygons("population", convert2density =T)

tm2 <-tm_shape(NLD_muni) +tm_bubbles(size ="population")

tmap_arrange(tm1, tm2)

BaseMaps and overlay tile maps

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".

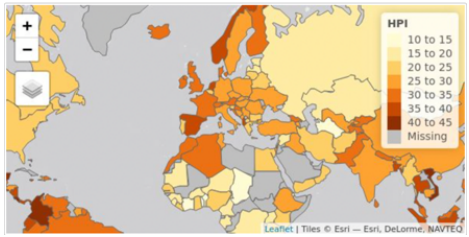
(Code): tmap_mode("view")

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

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.

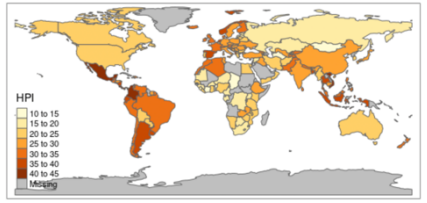
(Code): tmap_mode("view") tm_shape(World) + tm_polygons("HPI")



Quick Thematic Map

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

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



Exporting Maps

tm <- tm_shape(word)+tm_polygons("HPI", legend.title ="Happy Planet Index")

A) Save an image ("plot" mode)

(Code): tmap_save(tm, filenamefilename= "world_map.png")

B) Save as stand-alone HTML file("view" mode)

(Code): tmap_save(tm, filename ="world_map.html")