Geog 205 Exercise 7: Mapping with tmap

1. You can produce map data relatively easily in R using packages like ggplot2 and tmap. To get started with this exercise you should read through the tmap vignette in Resources page. More detail is provided in the tmaptools paper. You can find the vignette here

https://cran.r-project.org/web/packages/tmap/vignettes/tmap-getstarted.html

2. tmap reads in various kinds of map data (primarily shapefiles and other R data objects) and allows you create professional looking maps relatively quickly and easily.

3. To get started, open R in RStudio and install the following packages

> install.packages("tmap")

> install.packages("tmaptools")

> install.packages ("raster")

> install.packages("sf")

4. And then load all these libraries

> library(tmap) ……..etc

If you have not downloaded a recent version of R, especially those of you using Macs, you could encounter some problems loading the tmap library. If you are having trouble and R suggests that you don't have the libraries "terra" or "raster", from the R console, try and re-install those libraries from Github, using the following two commands:

> remotes:: install_github("rspatial/terra")

> remotes:: install_github("rspatial/raster")

5. Under the Datasets page, there is a .zip (compressed) file of asthma map data for the city of Los Angeles. Download the file to a folder for your R data, unzip it and you should see the individual components of the asthma shape file.

6. Set your working directory in R to point to the folder containing the asthma data and read in the shape file data

> asthma_map <- st_read("asthma.shp")

(Note that the original read_shape command might not work anymore. Depending on what libraries are loaded you might also see some warning messages from R. You don't need to worry about these, so long as your code runs.)

7. Look at the structure of your new object

> str(asthma_map)

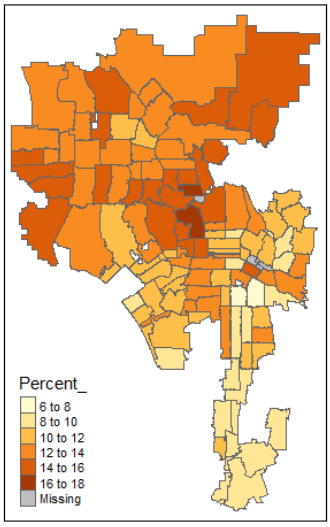
The variable “Percent_” contains numeric data showing the proportion of adults with asthma for each of the 124 polygons comprising the city of Los Angeles.

8. You can generate a quick map of the asthma data as follows

```
> qtm(asthma_map, fill = "Percent_")
```

9. And you can do the same with tmap

```
> tm_shape(asthma_map) +tm_fill("Percent_")
```



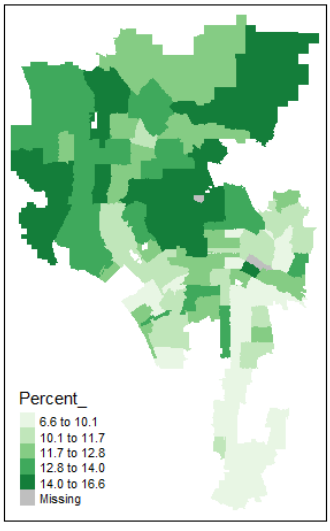
The advantage of tmap over qtm is that it provides a range of customization options for your maps

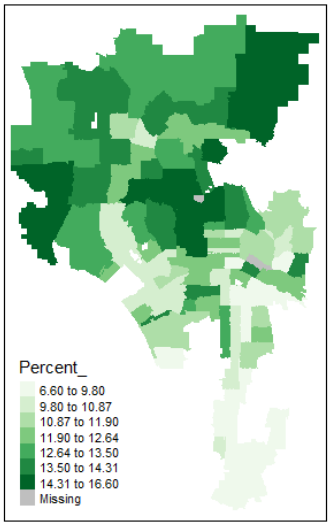
It is easy to change colors, to set the type of intervals and the number of intervals that you map, to add titles, & other information

10. Experiment with tmap – more of the options are shown in the tmaptools.pdf under the Resources page.

```
> tm_shape(asthma_map) + tm_fill("Percent_", style = "quantile", palette = "Greens")
```

```
> tm_shape(asthma_map) + tm_fill("Percent_", style = "quantile", n=7, palette = "Greens")
```



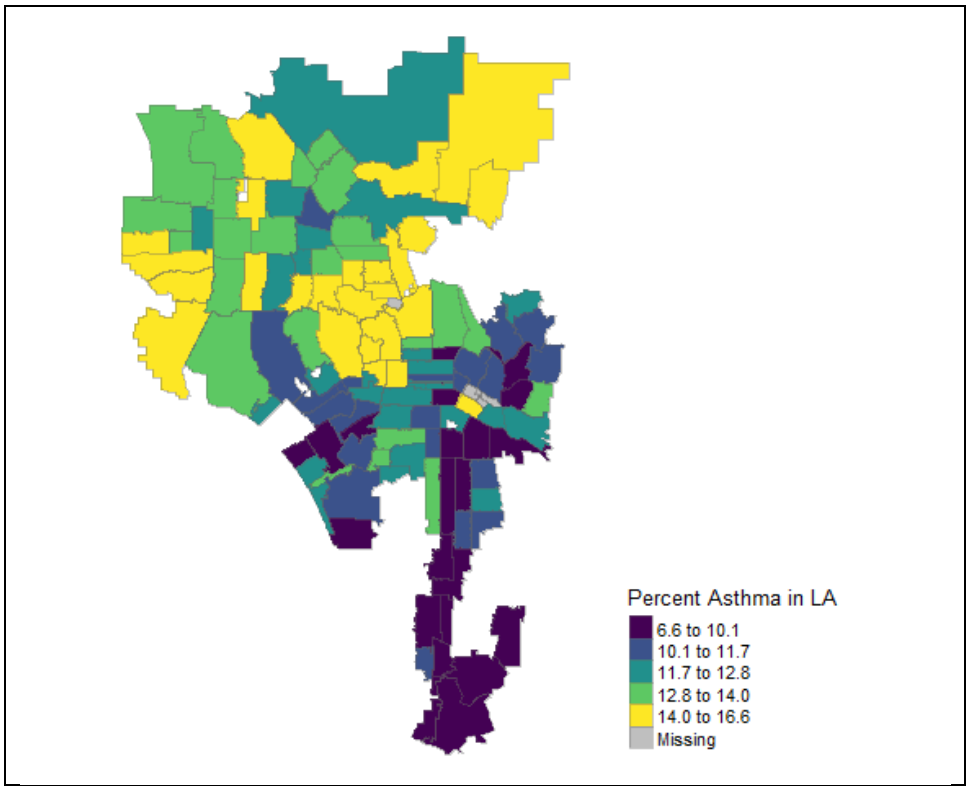


11. You can add a lot more to your map including polygon names, a scale bar and compass (check out the tmap vignette and the tmap tools pdf):

```
> tm_shape(asthma_map) +
+     tm_fill("Percent_", palette = "viridis") +
+     tm_layout(legend.outside = TRUE, frame = FALSE) +
+     tm_text("ZIPCODE", size = "Shape__Are",  auto.placement = F, legend.size.show = FALSE)
```

12. You can edit the map layout – adding borders (alpha determines border transparency) and the position of map objects

```
> tm_shape(asthma) +
+     tm_fill("Percent_",
+             palette = "viridis",
+             style = "quantile",
+             title = "Percent Asthma in LA"
+     ) +
+     tm_borders(alpha=.4) +
+     tm_layout(
+         legend.text.size = 0.7,
+         legend.title.size = 1,
+         legend.position = c("right", "bottom"),
+         legend.outside = TRUE,
+         frame = FALSE
+     )
```



12. If you add data to your shape file, you can save your new shape file using

```
> st_write(asthma_map, "asthma2_map.shp")
```

**Note that you will see a warning message. Ignore that.

13. What about some point data? Download the zip files LAbasemap.7z and May2018.7z from the Datasets page. (Unzip these files and store them in your R data directory.)

14. Set your working directory in R to point to the folder containing the new data files and read in the polygon (area) data from the Labasemap.shp and point data from the May2018.shp file

```
> lamap <- st_read("LAbasemap.shp")
> lacrime <- st_read("May2018.shp")
```

15. You can generate an overview of the polygons for LA again

```
> qtm(lamap)
```

And you can have a quick look at the location of all crimes on LA for the month of May 2018

```
> qtm(lacrime)
```

Checkout these quick maps.

16. Have a quick look at the lacrime object and note the Latitude and Longitude variables that give spatial coordinates for each location of a crime

```
> str(lacrime)
```

17. We can also overlay the crime point pattern onto the labasemap in tmap

```
> tm_shape(lamap) + tm_polygons() + tm_shape(lacrime) + tm_dots()
```

18. This point pattern map plots the locations of all crimes in the city (above left). Different types of crimes are separated by the variable crimecode and the codes for that variable are described by the variable descript. Check out some of the codes.

```
> View(lacrime)
```

19. Now what if you just wanted to examine vehicle thefts or crimecode = 510. You need to subset your data. One way of doing that, is to generate a new shape file object focused only on crimecode 510

```
> lacrime510<- lacrime[ which(lacrime$crimecode==510),]
```

Then to produce the map of car thefts (above right)

```
> tm_shape(lamap) + tm_polygons() + tm_shape(lacrime510) + tm_dots()
```

20. You can customize this map in different ways – check out the tmap literature.

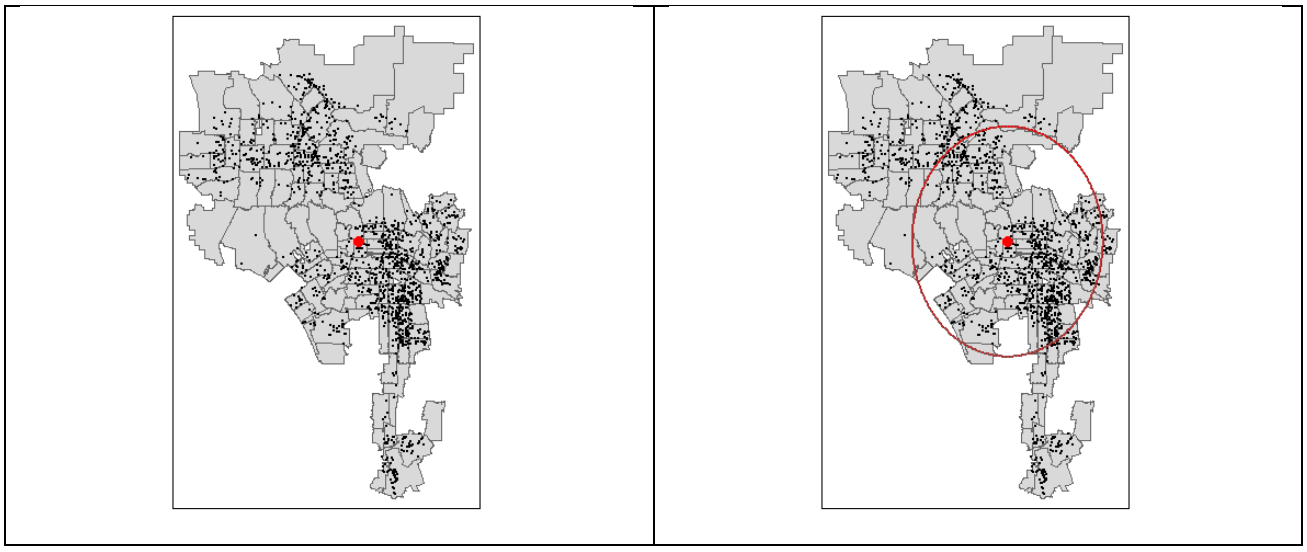
21. Now you have a point pattern layered on top of a base map. You can extract the coordinates of your points and do all the point pattern analysis that you want. What about plotting a mean center or the standard distance? Once you have created your mean center
(mc510) you have to turn it into a spatial point object using the st_sfc() function

```
> xy <- cbind(lacrime510$Longitude, lacrime510$Latitude)
> mc <-apply(xy, 2, mean)
> mc510 <- st_sfc(st_point(c(mc[1],mc[2])))
```

To plot this

```
> tm_shape(lamap) + tm_polygons() +
+ tm_shape(lacrime510) + tm_dots() +
+ tm_shape(mc510) + tm_dots(col='red',size=.35)
```

See left panel (below)



20. For standard distance, follow this and think about what you are telling R to do as you go….

```
> sd <- sqrt(sum((xy[,1] - mc[1])^2 + (xy[,2] - mc[2])^2) / nrow(xy))
> bearing <- 1:360 * pi/180
> cx <- mc[1] + sd * cos(bearing)
> cy <- mc[2] + sd * sin(bearing)
> circle <- cbind(cx, cy)
> mean_circle<-st_sfc(st_multipoint(circle))
```

To plot: (see right panel above)

```
> tm_shape(lamap) + tm_polygons() +
+ tm_shape(lacrime510) + tm_dots() +
+ tm_shape(mc510) + tm_dots(col='red',size=.4) +
+ tm_shape(mean_circle) +tm_dots(col='red',shape=21)
```

More hints for the last question on Assignment 2 ...

1. When you read in the state population csv files, you need to drop the rows for Alaska and Hawaii. Read in the population data with something like the following:

```
> stpop <- read.csv("us_state_popln_history.csv", sep=",", header=T)
```

Note that the variable for state names is “state”. To omit Alaska and Hawaii, try

```
> stpop2 <- stpop[which(stpop$state !="Alaska" & stpop$state !="Hawaii"),]
```

There are many ways to do the same thing in R. Even more efficient, after noting that Alaska is row 2 and Hawaii row 12 in the stpop file:

```
> stpop2 <- stpop[-c(2,12),]
```

Check the results with

```
> View(stpop2)
```

2. What about the NA values in the state population data? Removing Alaska and Hawaii solves most of your problems. That leaves a couple of NA <-observations in Oklahoma. What do you think you should do? Think about alternatives and explain your choice.

3. Let us assume that you have read into R a state.shp file (labelled stshape) along with your state population data (stpop2). How do you link these two objects so that you can map the population data? Looking at the data, the variable name for states in the stshape object is “NAME”. In the stpop2 object, states are listed under the variable “state”. You can merge these objects in different ways. One way is

```
> stpop3 <- merge(stshape, stpop2, by.x="NAME", by.y="state")
```

Check this

```
>View(stpop3)
```

4. You need the coordinates for the center of each state. We call these centroids. How to find the centroids?

a. One way is to extract them from your shape file

```
> centroids <- st_centroid(stpop3)     *ignore warning message and proceed
```

```
>cxy <- as.data.frame(st_coordinates(centroids))
```

And check that you have the longitude and latitude

```
> cxy
```

b. The other solution is to use the centroids that I provide in the .csv file of Module 2 and merge them with your existing data. Just use the commands you know. Let us assume that you have merged your centroids to stpop3 and stored the result in the object stpop4. You can extract your state coordinates easily as

```
>  x <- stpop4$lon
```

```
>  y <- stpop4$lat
```

6. Now you can generate some weighted means for different years using the latitude and longitude coordinates and your state population values. You need to end up with a points object storing you're your weighted means for different years. The following code might be helpful (you need to expand this a little for other years)

```
> wmx_1900<-weighted.mean(x,stpop4$pop1900)
```

```
> wmy_1900<-weighted.mean(y,stpop4$pop1900)
```

```
> wmx_2000<-weighted.mean(x,stpop4$pop2000)
```

```
> wmy_2000<-weighted.mean(y,stpop4$pop2000)
```

```
> wm_x <- c(wmx_1900, wmx_2000)
```

```
> wm_y<-c(wmy_1900, wmy_2000)
```

```
> wm_xy<-cbind(wm_x,wm_y)
```

```
> df_wmxy <- data.frame(wm_xy)
```

```
> uspoints <- st_as_sf(x=df_wmxy, coords=c("wm_x", "wm_y"), crs=4269)
```

```
> years <- c("1900", "2000")
```

```
> uspoints&years <-years
```

And using the commands you know, now plot your map of the US with the different weighted means showing the movement of the population center of the US.