Post 01: An Introduction to Choropleths in R

Alex Yi

October 29, 2017

Introduction

When most think of data visualizations, histograms and all sorts of graphs first come to mind. However, one can also represent data on geographic maps. A choropleth map is defined as a thematic map where areas are shaded in proportion to some metric, such as number of people living in that area. Because data is organized in a ratio, the choropleth uses a graded color series to show the data from least to most intense, in a light-dark color pattern.

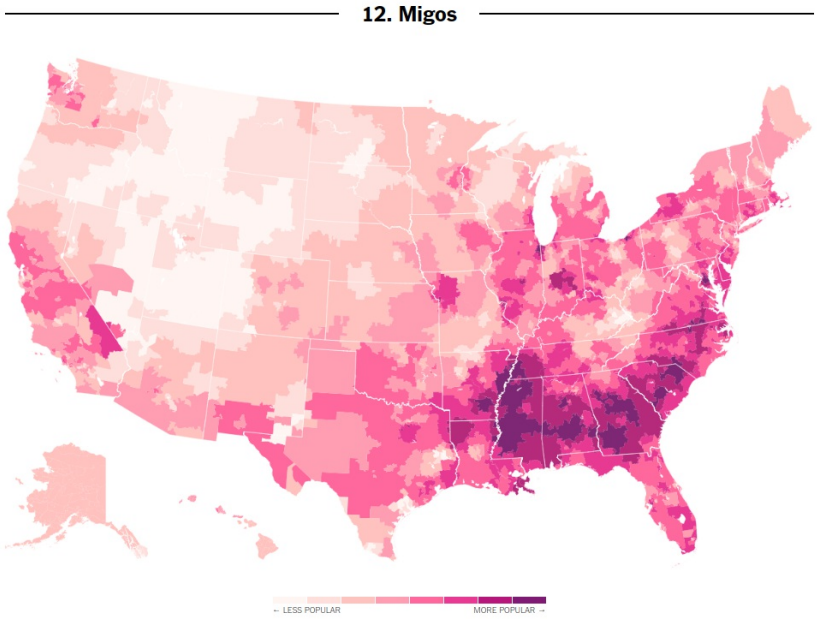
This post is meant as an introduction to choropleths, as well as the number of ways that one can use R to plot such maps, as well as provide some brief insight into how such methodologies work. I hope that you will find this informational and entertaining.

Background

As aforementioned, choropleth maps are maps where data is aggregated and summarized over previously defined regions (i.e. countries, states, census tracts). As a result, choropleths are preferred in studies where defined regions are important to the study at hand, such as election votes over a map divided into election regions.

One such example of a choropleth can be seen below. This map of the United States, from a New York Times article, uses youtube streaming data to determine where certain artists' videos receive the most play. In this case, Migos, an American rap trio, is displayed below. Given the legend that the darker shades represent higher concentrations of plays and therefore more popularity, one can observe that Migos are most popular in the Southeast region of the United States, where they hail from themselves.

```
#source: New York Times
knitr::include_graphics('migos.png')
```



Although there are indeed other tools that can be used to create choropleths, such as Mapbox, Tableau, and d3.js, this post will focus on using various libraries in R.

R Libraries

The number of ways to represent choropleths in R continues to grow with each passing day, so for the sake of this post I shall try to only focus on a few. However, one must first obtain the map and its predefined regions in order to begin. One can obtain maps by downloading map shape files through online databases such as http://gadm.org, or through libraries such as CShapes, which contains country boundaries from 1946-today. On the other hand, R also provides libraries such as ggmap, which downloads the maps from Google or Open Street Maps or Stamen Maps to use in the background of the plots.

Once the required map is loaded in one way or another, one can then fill in the choropleth through a variety of different libraries. Perhaps the most popular method would simply be through ggplot2. One can fill in the map via geom_map or geom_polygon, where coordinates are required to designate each respective region. Other libraries include sp, which uses spplot to plot geographic objects, simple features (sf), which extends the basic plot() command to be used on sf objects, and tmap, which is based on the ggplot syntax but largely takes care of most of the required styling.

Examples

In this example, I am going to create a choropleth of Alameda County, where the statistic in mind is the estimate of individuals below the 200 percent level of poverty. Furthermore, for the purpose of simplicity, I will only be using the aforemetioned ggplot2() and ggmap(). First, I have to load the shape file, as well as the relevant data. In this case, the data is the US Census data for the year 2010. I also load all relevant libraries.

```
library(rgeos)
```

```
## rgeos version: 0.3-25, (SVN revision 555)
##  GEOS runtime version: 3.5.0-CAPI-1.9.0 r4084
##  Linking to sp version: 1.2-5
##  Polygon checking: TRUE
```

```
library(maptools)
```

```
## Loading required package: sp
```

```
## Checking rgeos availability: TRUE
```

```
library(rgdal)
```

```
## rgdal: version: 1.2-13, (SVN revision 686)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 1.11.3, released 2015/09/16
##  Path to GDAL shared files: /usr/share/gdal/1.11
##  Loaded PROJ.4 runtime: Rel. 4.9.2, 08 September 2015, [PJ_VERSION: 492]
##  Path to PROJ.4 shared files: (autodetected)
##  Linking to sp version: 1.2-5
```

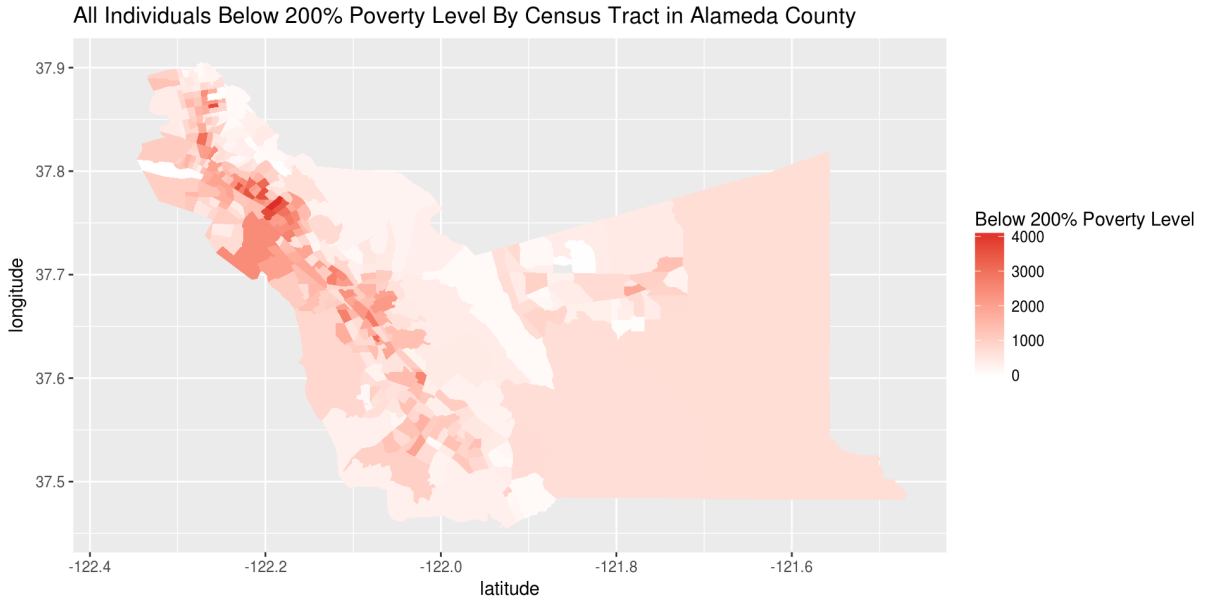
```
library(ggplot2)
#load shape and data files from directory
census_df <- read.csv("data/2010_census.csv")
alameda_sp <- readOGR("shapes/tl_2010_06001_tract10.shp")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "shapes/tl_2010_06001_tract10.shp", layer: "tl_2010_06001_tract10"
## with 361 features
## It has 12 fields
```

```
plot(alameda_sp)
```

With the shape loaded, the first chloropleth can be created, with the variable in mind.

```
library(scales)
#convert the Polygon Data Frame into a data frame based on the given region
alameda_df <- fortify(alameda_sp, region = "NAME10")
alameda_choro <- ggplot() + geom_map(data = census_df,
                    aes(map_id = CensusTract,
                        fill = All.Individuals.below.200.percent.of.poverty.level.Total.Estimate),
                    map = alameda_df) +
  expand_limits(x = alameda_df$long, y = alameda_df$lat) +
  scale_fill_gradient2(low = "#ffeda0", high = "#de2d26") +
  labs(fill = "Below 200% Poverty Level",
       title = "All Individuals Below 200% Poverty Level By Census Tract in Alameda County",
       x = "latitude", y = "longitude")
alameda_choro
```



With this first iteration of the choropleth, the code can appear rather bulky and confusing, so I will do my best to break it down. Given that the previous block read a .csv of all necessary locations, as well as a .sp file into a Spatial Data Frame (called a Large SpatialPolygonsDataFrame in this case), one must then use fortify() to convert the Spatial Data Frame into a standard data frame, which can provide to ggplot the name of the region. In this case, census tracts, represented as NAME10 on the .sp file, are used. Census tracts are the smallest metric for which population data is available, and are defined as the geographic region defined for taking a census.

Using the geom_map function on ggplot, we use the data frame (US Census data from 2010) to specify what statistic will be used to color in the map, as well as what vector will be used to match up to the shape file. The statistic in mind is represented as *fill = All.Individuals.below.200.percent.of.poverty.level.Total.Estimate,* and the vector used is *map_id = CensusTract.*

Once the variables are matched up, the map is drawn using expand_limits(), where the longitude and latitude are given in the shape file, and then colored in. In this case, the color scale is obtained from hexcodes from http://colorbrewer2.org.

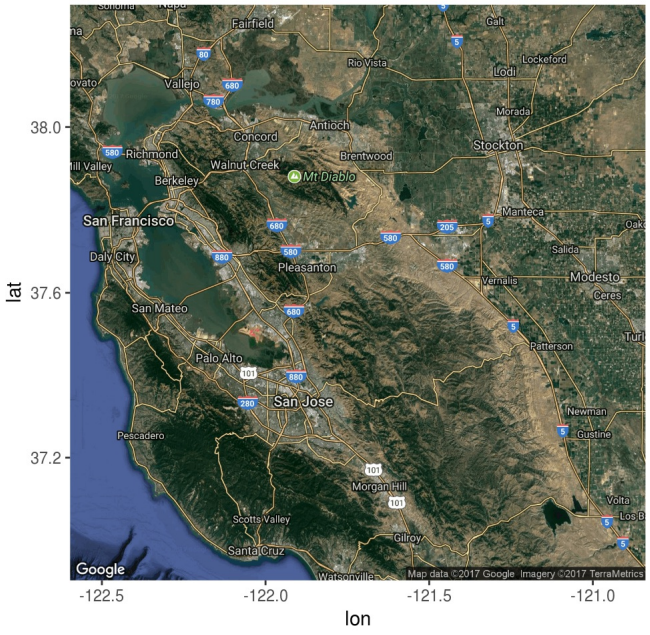
In this next iteration, I will attempt to add a background of Alameda County using ggmap. Using simply ggmap, one can search through the aforementioned online map sources to obtain a specific base image. In this case, the base image is that of Alameda County, CA as a whole.

```
library(ggmap)
#load the base ggmap for alameda county
alameda_map <- get_map(location = "Alameda County, CA", maptype = "hybrid", zoom = 9)
```

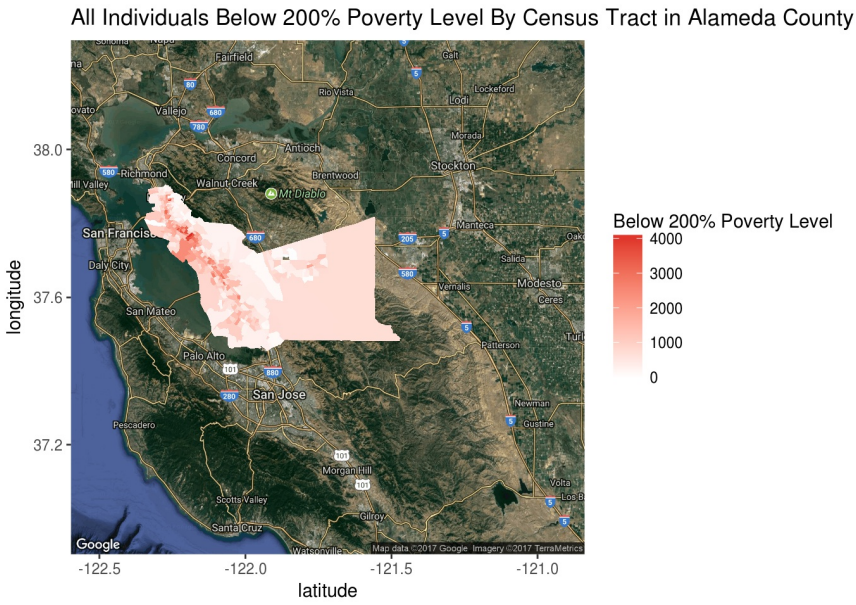
```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Alameda+County,+CA&zoom=9&size=640x640&scale=2&maptype=hybrid&language=en-EN&sensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Alameda%20County,%20CA&sensor=false
```

```
ggmap(alameda_map)
```



```
#overlay ggmap on top of given choropleth
ggmap(alameda_map) + geom_map(data = census_df,
                    aes(map_id = CensusTract,
                        fill = All.Individuals.below.200.percent.of.poverty.level.Total.Estimate),
                    map = alameda_df, inherit.aes = FALSE) +
  expand_limits(x = alameda_df$long, y = alameda_df$lat) +
  scale_fill_gradient2(low = "#ffeda0", high = "#de2d26") +
  labs(fill = "Below 200% Poverty Level",
       title = "All Individuals Below 200% Poverty Level By Census Tract in Alameda County",
       x = "latitude", y = "longitude")
```



As seen above, ggmap allows one to specify which type of map one wants according to the variable *maptype*, as well as a specified zoom. This can then be overlaid over the ggplot choropleth. However, in order for the layers to match up, one must specify *inherit.aes = FALSE*. The objects in the choropleth try to inherit the x and y mapping from the ggmap layer, but that would result in an error.

Conclusion

In this post, I strove to introduce you to the concept of a choropleth, how it can be created in R, and how it can be used to display data in a geographic manner. I obtained a shape file, plotted data onto it via census tracts, and overlaid a geographic map over the choropleth. I hope you learned more about data visualization, beyond what was covered in class. Perhaps the next time you see the opportunity to represent geographic data over a map, you will remember what a choropleth is, and think back to this post.

Sources:

http://www.milanor.net/blog/maps-in-r-choropleth-maps/ https://cengel.github.io/rspatial/4_Mapping.nb.html
http://bl.ocks.org/prabhasp/raw/5030005/ https://cran.r-project.org/web/packages/ggmap/ggmap.pdf
https://developers.google.com/maps/documentation/geocoding/start?csw=1 http://nils.weidmann.ws/projects/cshapes.html
http://my.ilstu.edu/~jrcarter/Geo204/Choro/Tom/ https://www.nytimes.com/interactive/2017/08/07/upshot/music-fandom-maps.html
https://www.opendatanetwork.com/dataset/data.acgov.org/7sbd-w6mc http://colorbrewer2.org/#type=sequential&scheme=BuPu&n=3