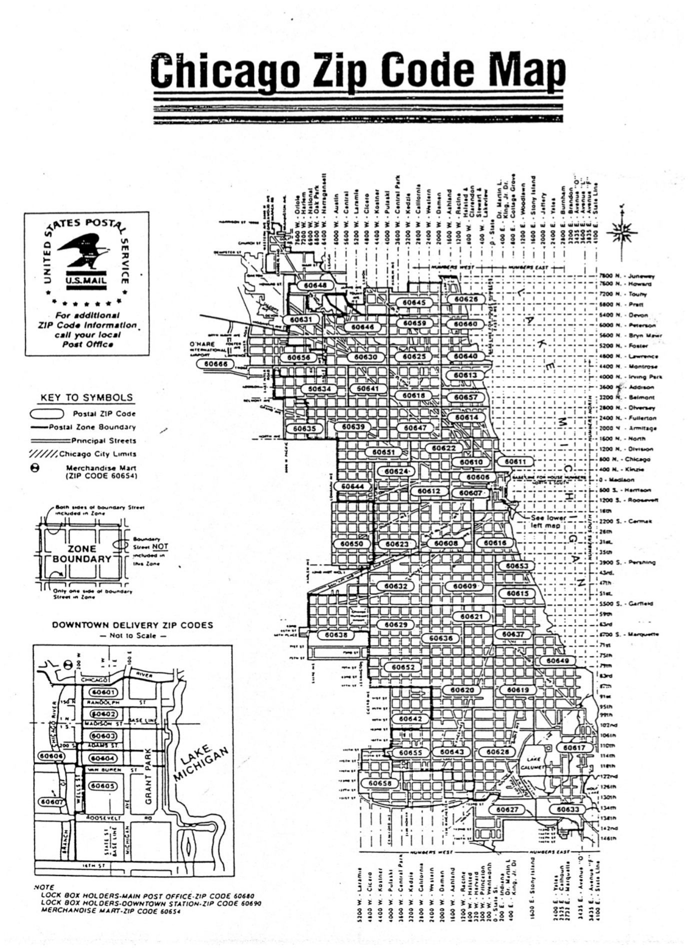
Choropleth mapping in R

Exclusion

2021-10-19

The picture below is the original map. It's difficult to be plotted by hand with different colours representing different values of data. How can we transform that to something operable to R such that we can do the data analysis and mapping jointly, and also display the map more elegantly compared to hand-drawn?


Chicago Zip Code Map
KEY TO SYMBOLS
Postal ZIP Code
Postal Zone Boundary
Principal Streets
Chicago City Limits
Merchandise Mart (ZIP CODE 60654)
ZONE BOUNDARY
DOWNTOWN DELIVERY ZIP CODES
— Not to Scale —
LAKE MICHIGAN
NOTE
LOCK BOX HOLDERS-MAIN POST OFFICE-ZIP CODE 60680
LOCK BOX HOLDERS-DOWNTOWN STATION-ZIP CODE 60690
MERCHANDISE MART-ZIP CODE 60654

First load some packages in R useful for combinatory choropleth mapping

```r
library(zipcodeR)
library(tigris)
library(sf)
library(mapsf)
library(tidyverse)
library(RColorBrewer)
```

Here we use insurance data for Chicago zip codes. The zip code serves as the connection between tabulated data frames with the spatial data (shapefile), a bit like the "JOIN" in SQL.

```
insure.mddata<-read.table("Insure.txt",header=T)
head(insure.mddata)
```

```
##     Zip Race Fire Theft  Age Volun Invol Income
## 1 60626 10.0  6.2    29 60.4   5.3   0.0  11744
## 2 60640 22.2  9.5    44 76.5   3.1   0.1   9323
## 3 60613 19.6 10.5    36 73.5   4.8   1.2   9948
## 4 60657 17.3  7.7    37 66.9   5.7   0.5  10656
## 5 60614 24.5  8.6    53 81.4   5.9   0.7   9730
## 6 60610 54.0 34.1    68 52.6   4.0   0.3   8231
```

See that it's a tabular data but with the ZIP code providing geographical information (vector data in GIS). We first need to update the ZIP codes to make sure it conforms to the current shapefile labels.

```
## Adjust the old zip codes for consistency
Zips<-insure.mddata$Zip
Zips[which(Zips==60627)]<-60827
Zips[which(Zips==60635)]<-60707
insure.mddata$Zip2019<-Zips
```

We now use the R package zipcodeR and tigris to obtain the shapefile data for the zip codes we want.The shapefile data belongs to the single feature (sf) object in R, with a suite of functions specific for this object for quick analysis.

```
## extract data corresponding to our targetd zip codes
geo.chicago<-st_as_sf(zctas(cb = T,starts_with = Zips))
```

```
## ZCTAs can take several minutes to download.  To cache the data and avoid re-downloading in future R
```

```
## extract state boundary
USstates <- st_as_sf(states(cb=TRUE))
## transform the coordinate system for both sf
USstates=st_transform(USstates,st_crs(geo.chicago))
```

Now here is an important step - to make sure the shapefile and data link correctly, we need to have consistent column names.

```
## Change the column name for zipcodes for the shapefile
names(geo.chicago)[1]<-"Zip2019"
## Now we can merge them.
sfdata.chi<-merge(geo.chicago,insure.mddata,by = "Zip2019")
```

Choropleth mapping is the most popular way to display statistical data spatially.It displays pre-defined geographical areas or regions with colours in relation to a data variable. Normally for better interpretation, we need to separate the continous variables into different classes.

Luckily, we have R packages that inherited the features from ArcGIS and allow classification by quantile, clustering and other methods.

Here we take the racial composition variable (*Race*) as the example.

```r
# package for classification in mapping
library(classInt)
## use k-means clustering to define the break
## Here I use n-3 to have a natural low-medium-high definition
breaks.race.kmeans<-classIntervals(c(-0.00001,sfdata.chi$Race),n=3,style="kmeans",rtimes=30)
breaks.race.kmeans$brks
```

```
## [1] -0.00001 29.25000 68.65000 99.70000
```

```r
## here we need to define a minimum value -0.00001 otherwise the minimum values will return NA in colou
```

Now we are ready to plot the map!

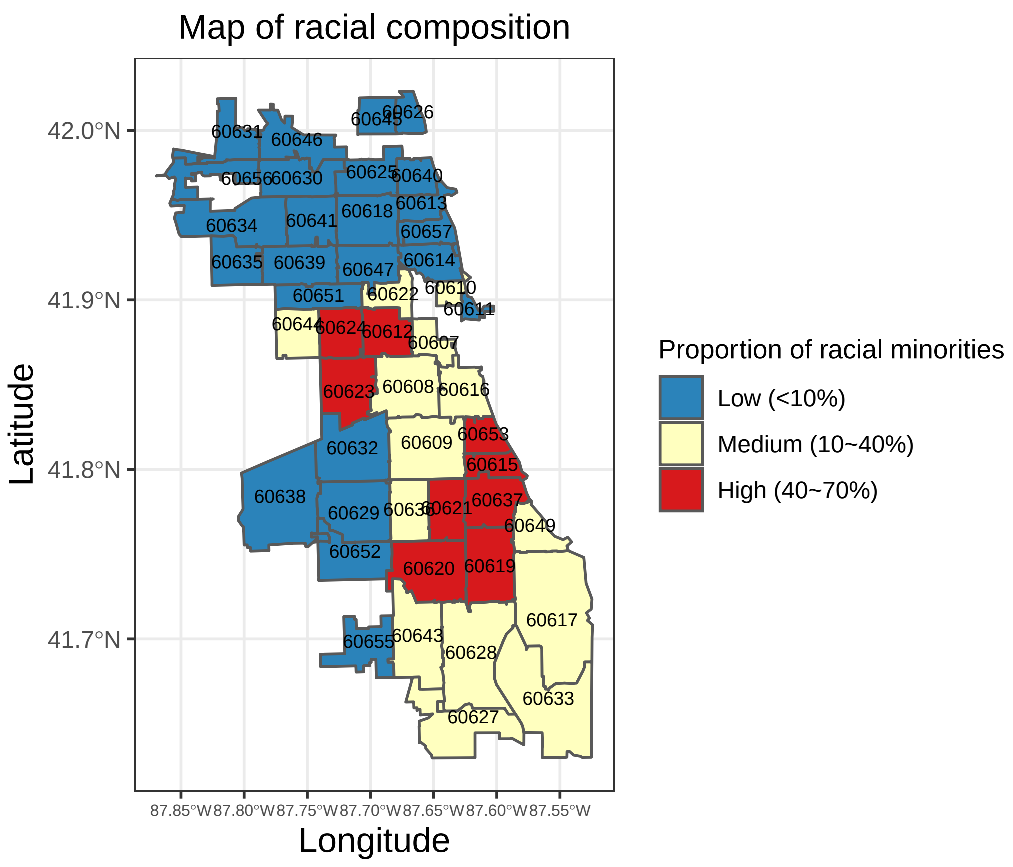
```r
## Cut the variable by the break, and then add it to the dataframe
sfdata.chi<-mutate(sfdata.chi,Race_km=cut(sfdata.chi$Race,breaks.race.kmeans$brks))
# Define the colour scheme using funciton from RcolorBrewer
mycolor<-rev(brewer.pal(5,"Spectral")) #

### use the ggplot function to plot the map
## Note that the new version of ggplot is able to handle sf object as like a dataframe

## Aesthetic set up for the graph
graph.theme.beta<-theme(axis.title=element_text(size=13,vjust=-0.4),axis.text.x = element_text(size=6),

## geom_sf_text() adds the zip code to the centroid of the polygon -
## much quicker than its precedent sp object
ggplot(sfdata.chi)+geom_sf(aes(fill=Race_km))+
  # specify colour scheme (also change the legend)
  scale_fill_manual(name="Proportion of racial minorities",labels=c("Low (<10%)","Medium (10~40%)","Hig
  theme_bw()+
   geom_sf_text(data=sfdata.chi,aes(label=Zip),colour="black",size=2.4)+ labs(title="Map of racial comp
```

```
## Warning in st_point_on_surface.sfc(sf::st_zm(x)): st_point_on_surface may not
## give correct results for longitude/latitude data
```



Now we change the blurred scanned map into a nicely looking choropleth map. We can see the low racial minority zip codes (predominantly white) is clustered in the northern parts of Chicago, while the middle parts are mostly high-minority neighbourhoods.