Visualization of State and City Specific Maps using R

Andrew Hooyman

2024-04-13

Introduction

As rehabilitation science begins to move out of the lab and into the home, generating a visual understanding how where patients/participants come from can be informative. The purpose of this vignette to make the process of visualizing these data easier. Data and code used here can be found at https://github.com/hooymana/easyMap

The following libraries are needed to run this code.

```
library(usmap) #for making the actual map and converting longitude and latitude to map coordinates (x,y
library(ggplot2) #for plotting participant/patient location
library(maps) #to generate city data
library(zipcodeR) #to convert zip code to longitude and latitude
library(ggrepel) #so when plotting city labels there is no overlap
```

```
#Set working directory
#setwd("")

#Data needed to create all visualizations
sample.data=read.csv("Sample.Map.Data.csv") #example participant data
city_xy=read.csv("City.Data.csv") #city data that was previously handled
```

The sample.data data frame represents the variables needed to plot participant/patient data. These data span 41 states but we will be focused on indexing these data so only data from specific states are visualized.

The x and y columns are the coordinated specific location to allow rendering of location in the usmap visualization function: plot_usmap.

```
head(sample.data)
```

```
##   X state         x          y
## 1 1    MN  541704.1   23092.39
## 2 2    RI 2307655.1   49860.64
## 3 3    WI  972454.4 -150677.24
## 4 4    WV 1703098.4 -391854.26
## 5 5    PA 2080809.4 -244374.20
## 6 6    TX  296374.3 -1330848.67
```

Quick side note: It's unlikely you will get participant data that are already with the correct coordinate data to be visualized using plot_usmap. You may either get 5 digit zip code or longitude or latitude. Below are functions that can be used to convert 5 digit zip code to longitude and latitude and then longitude and latitude to the plot_usmap specific coordinates.

```r
#To convert zip code to longitude and latitude, from zipcodeR library
zip_to_longlat=reverse_zipcode(c("90210","10001"))

zip_to_longlat[,c("state","lat","lng")]
```

```
## # A tibble: 2 x 3
##   state   lat    lng
##   <chr> <dbl>  <dbl>
## 1 CA     34.1 -118.
## 2 NY     40.8  -74.0
```

```r
#To convert longitude and latitude to plot_usmap coordinates, from usmap library
cities_with_xy = usmap_transform(zip_to_longlat,input_names = c("lng", "lat"))

#You just need the following variables for mapping
#country.etc is state
#x is x coordinate for plot_usmap
#y is y coordinate for plot_usmap
cities_with_xy[,c("state","x","y")]
```

```
##   state        x        y
## 1    CA -1687364 -1026000
## 2    NY  2147355  -128329
```

Hopefully outlining the initial process will help clarify how the data have been transformed previously for these next steps.

Mapping Method with plot_usmap and ggplot

You can see how these data can be seen in the most basic way using the plot_usmap function

```r
plot_usmap()+
  geom_point(cities_with_xy,mapping=aes(x=x,y=y),color="red")
```

The great thing about plot_usmap is that you can choose just one state. Let's do California

```
plot_usmap(include = "CA")
```

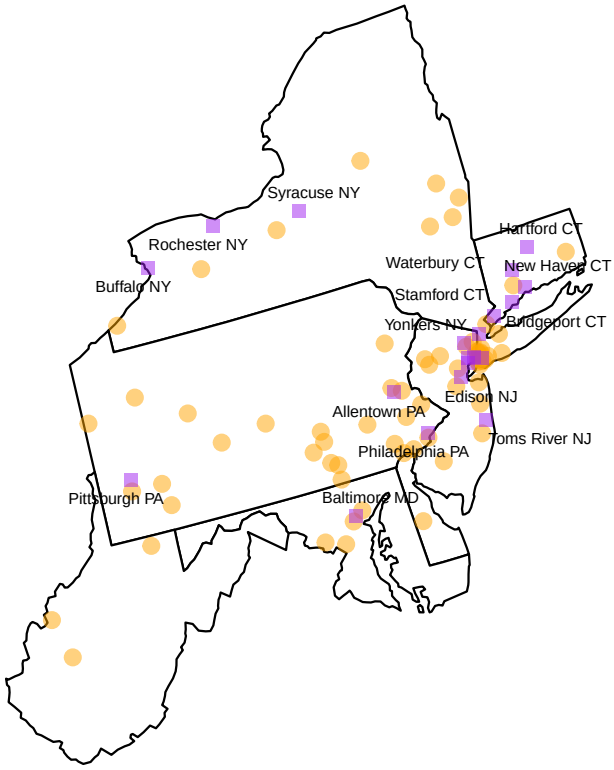
This is a good introduction, but a good standard by which we can really understand contextually where people reside is to also render nearby major metropolitan areas. This way we can see visually how far people are from macro and micro metropolitan areas. Below one can first list the state abbreviations for which specific states to render and the select the population size of the references cities to render.

```
#METHOD 1: RENDER CITIES BASED ON POPULATION SIZE
states.included=c("PA","NY","WV","NJ","DE","CT","MD") #Pick states you want to render
min.population.size=100000 #Select minimum population city size you want to render

#For only cities with a specific population size
plot_usmap(include = states.included) +#Renders just the states you want to include
  #Render points for participant location in orange
  geom_point(sample.data[sample.data$state %in% states.included, ],
             mapping = aes(x = x, y = y),
             color = "orange",
             size = 2.5,
             alpha = 0.5) +
  #Render points for cities you want to include based on population size in purple
  #For this particular area of the map there are a lot of cities with fairly high pop close to one anot
  #If labels get too close then they won't be rendered.
  #Selecting just the cities you want to render may be the better option
  geom_point(data = city_xy[city_xy$country.etc %in% states.included &
                              city_xy$pop>min.population.size ,],
             aes(x = x, y = y),size=2,shape=15,
             alpha = 0.5,show.legend = F,color="purple")+
  geom_text_repel(data=city_xy[city_xy$country.etc %in% states.included &
                                 city_xy$pop>min.population.size,],
```

```
                  aes(x = x, y = y,label=name),size=2.2,
                  point.padding = .2)
```

```
## Warning: ggrepel: 5 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



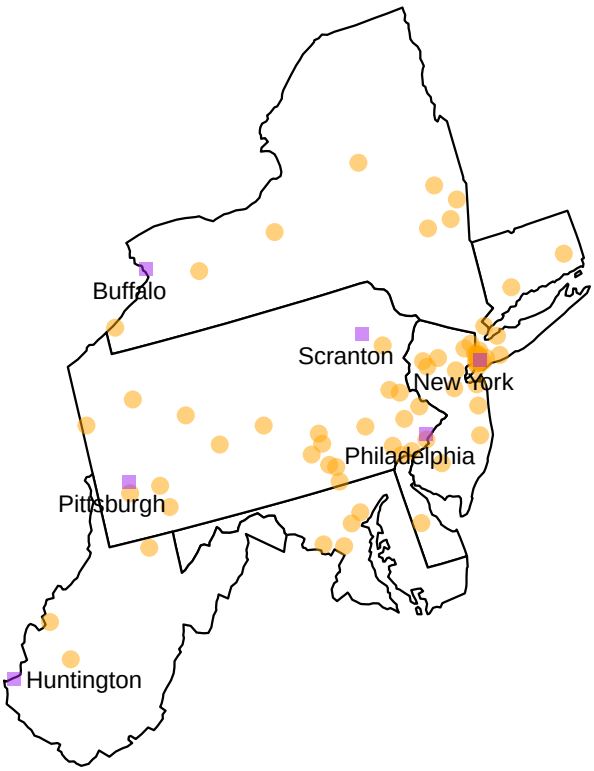
In this graph we have our participant data in orange and the reference cities that have a population of greater than 100000 in purple. We use geom_text_repel to provide labels of the reference cities. On caveat to this approach is that if there are too many cities close to one another, leading to too much label overlap, some labels are not rendered.

Alternatively, instead of plotting reference cities by population size the following code allows for plotting of specific cities entered by the researcher.

```
#METHOD 2: RENDER SPECIFIC CITIES BY NAME
states.included=c("PA","NY","WV","NJ","DE","CT","MD") #Pick states you want to render
city.included=c("Scranton","New York","Philadelphia","Pittsburgh","Huntington","Buffalo") #Pick specifi

plot_usmap(include = states.included) + #Renders just the states you want to include
  #Render points for participant location in orange
  geom_point(sample.data[sample.data$state %in% states.included, ],
             mapping = aes(x = x, y = y),
             color = "orange",
             size = 2.5,
             alpha = 0.5) +
  #Render points for cities you want to include in purple
```

```
  geom_point(data = city_xy[city_xy$country.etc %in% states.included &
                              city_xy$city %in% city.included ,],
             aes(x = x, y = y),size=2,shape=15,
             alpha = 0.5,color="purple")+
  #Render labels for cities you want to include
  geom_text_repel(data=city_xy[city_xy$country.etc %in% states.included &
                                 city_xy$city %in% city.included,],
                  aes(x = x, y = y,label=city),
                  size=3, #Change size of label here
                  point.padding = .2)
```



With this method we can just list specific reference cities so that we can eliminate too much clutter in the graph, especially as the number of labels may by large.

Alternative Mapping Method with ggplot and map_data

There is a way to also look at patient/participant location at the county level using the map_data function that comes with ggplot2. This process is even a little easier to handle that plot_usmap because it just takes the longitude and latitude directly and doesn't need an x and y specific conversion.

Example below

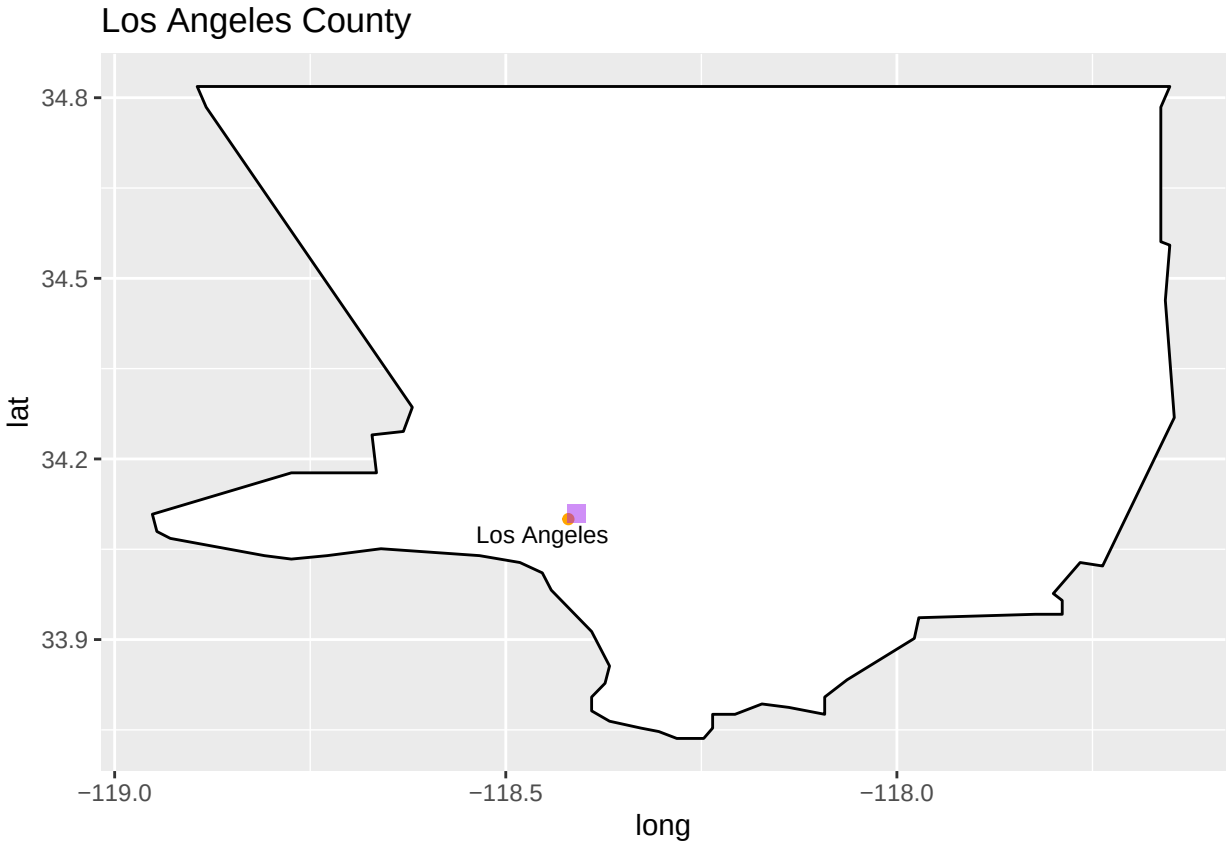
```
counties=map_data("county")
```

```
BH=reverse_zipcode(c("90210"))
BH[,c("state","lat","lng")]
```

```
## # A tibble: 1 x 3
##   state   lat   lng
##   <chr> <dbl> <dbl>
## 1 CA     34.1 -118.
```

```
counties.included="los angeles"
city.included2="Los Angeles"

#Fit polygons for county
ggplot(counties[counties$subregion %in% counties.included,],aes(x=long,y=lat,group=group))+
  geom_polygon(show.legend = F,fill="white",color="black")+
  #participant data
  geom_point(BH,mapping=aes(x=lng,lat),inherit.aes = F, color="orange")+
  #reference cities
  geom_point(city_xy[city_xy$city %in% city.included2,],mapping=aes(x=long,y=lat),
             inherit.aes = F,size=3,shape=15,color="purple",alpha=.5)+
  geom_text_repel(data=city_xy[city_xy$city %in% city.included2,],
                  aes(x = long, y = lat,label=city),
                  size=3, #Change size of label here
                  point.padding = 1,inherit.aes = F)+
  ggtitle("Los Angeles County")
```



You can also do state level data using this method as well but you will have to create a states variable using map_data("states")

Wrap-up

The goal of this vignette is to provide a straightforward and, hopefully, easier method to handle participant location data for visualization at the country, state, and county level.