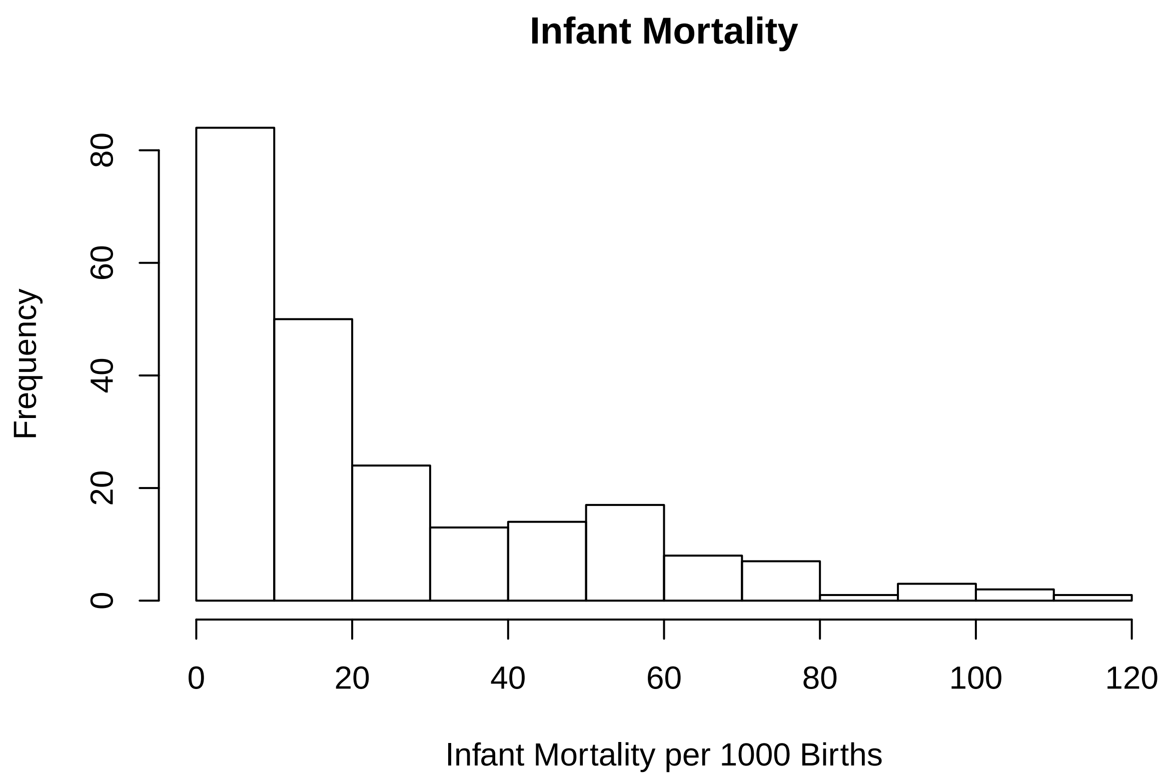
Global Demographics

Alex Bank



The histogram of infant mortality is right skewed, as we would expect. Most countries have a fairly low infant mortality rate, but the histogram is stretched to the right by a few countries with significantly higher rates.

```
##        country cia iso3166
## 242 Tajikistan  ti      TJ
```

For Tajikistan, the CIA uses the abbrieviation "ti" and the ISO uses "TJ."

After getting a CSV listing the latitude and longitude data for different countries, we needed to clean the data. Below you can see the list of country codes that had repeated values in the CSV file.

```
##  [1] "BN" "BO" "CI" "KR" "LY" "MM" "RU" "TT" "TW" "VC" "VE" "VN"
```

After removing the duplicated countries, we end up with a data frame with 243 rows.

After joining our separate data frames, some information was dropped. The process for joining the tables began by first combining all the data from the CIA factbook. Already we know we lost some rows since the factbook lists 279 countries, but only has population data for 240 countries and infant mortality data for 224 countries. When we join this table with the geolocation data, we only keep countries that have an iso3166 code, population data, and infant mortality data. Below we can see some the countries that were listed in the factbook, but dropped when joining with other tables.

```
##  [1] "Akrotiri"                       "Antarctica"
##  [3] "Ashmore and Cartier Islands"    "Baker Island"
##  [5] "Bassas da India"                "Bouvet Island"
##  [7] "British Indian Ocean Territory" "Christmas Island"
##  [9] "Clipperton Island"              "Cocos (Keeling) Islands"
```

We can now use our new table to look for specific statistics.

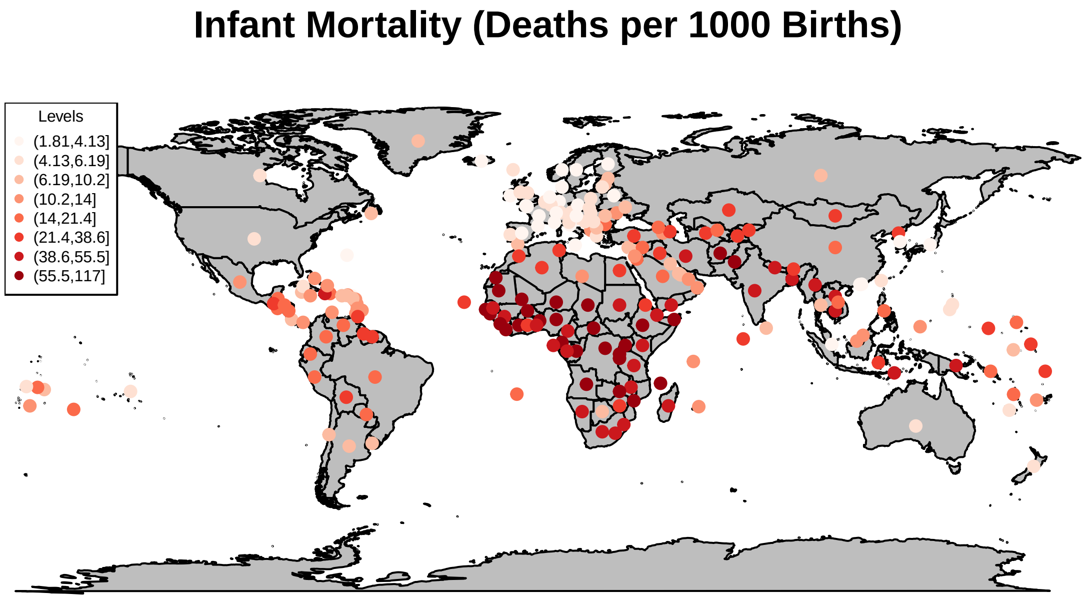
```
## The mean infant mortality rate for countries:
```

```
## with populations less than 10 million is 19.062
```

```
## with populations greater than 50 million is 26.05125
```

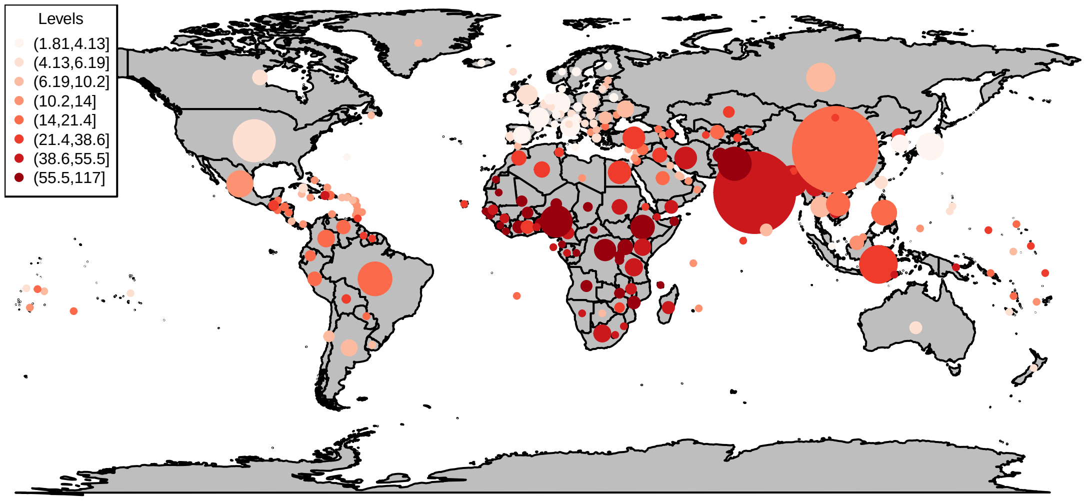
Below we can see how the countries fit into the descretized mortality rates. I used the quantile function to find eight break points for the function, which is why each factor level has about an equal number of members. I chose eight breaks because when I went to map the data, eight breaks ended up giving a nice gradient of values without being excessively granular or general.

```
## mortBreaks
## (1.81,4.13] (4.13,6.19] (6.19,10.2]   (10.2,14]   (14,21.4] (21.4,38.6]
##          28          28          27          28          27          28
## (38.6,55.5]  (55.5,117]
##          27          28
```



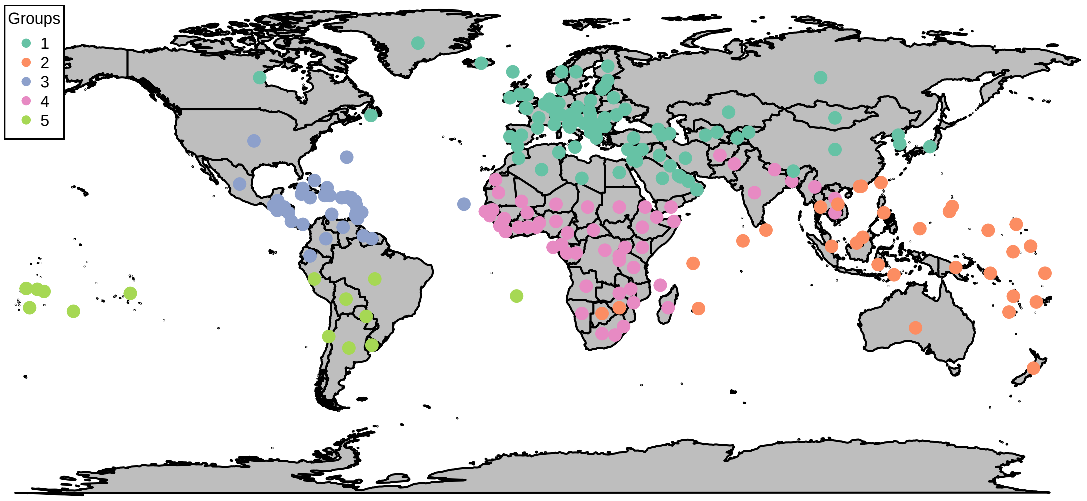
The next graphic shows the same information, but also gives some indication of each country's population. As expected, China and India dominate the global population and have middling to high infant mortality rates. The United States appears to be the country with the greatest population while still having a low infant mortality rate. Also note a minimum radius was applied to make sure all countries appear on the map.

Infant Mortality Scaled to Population



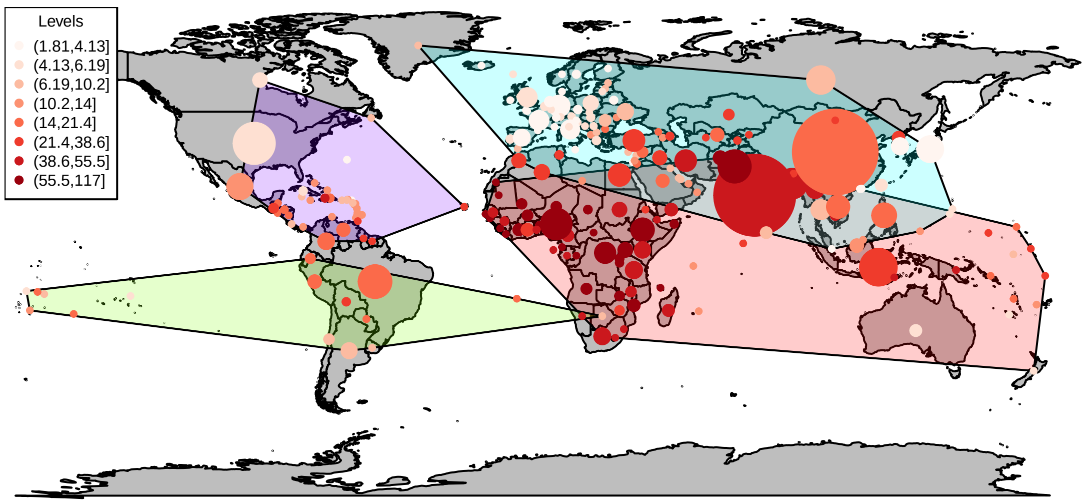
The above graphs interestingly seem to have broad trends; many countries with like infant mortality rates appear to be geographically similar. In the following map, I used *kmeans* clustering to assign each country into a group based on its location and infant mortality rate. Below we see the result of applying this method and creating five groups.

Countries in Groups based on Location and Infant Mortality



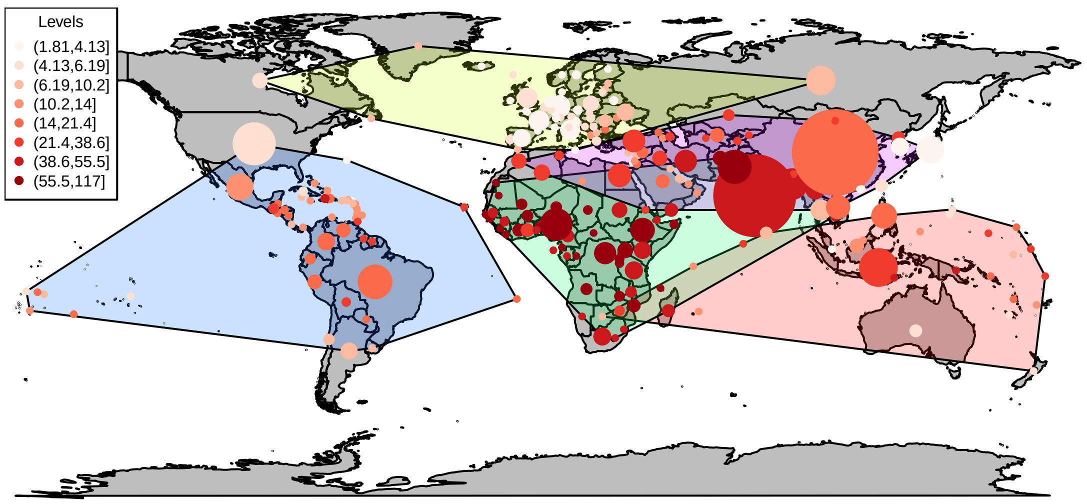
Now, just to improve the visual appeal, we graph the population and infant mortality data as before, but this time we add polygons outlining the groups of countries.

Four Groups of Countries



##

Five Groups of Countries



For my extension, go to: https://abbank92.shinyapps.io/USPopMap/

This extension involved a number of steps. The first data set I found had all the historical city population data. I wanted to combine this data with economic output, but it proved very difficult to find this data by city. What I did find, however, were two data sets: one linking every county to its Metropolitan Statistical Area, another linking Metro Area to economic output. The first data set listed the county for each city, so I was able to perform a series of left joins on the tables to get all the data I needed in one table.

From this data I constructed the interactive map. The map reveals several interesting points about population trends. Purely based on visual impression, the populations of CA and TX appear to have grow the most over the past 60 years. It was also interesting to see that the sparsely populated areas in 1900 remain relatively unpopulated in 2010.

I thought the groupings of cities based on population and economic output were incredibly interesting. The first thing to note is there is an elite group of cities that consists only of Los Angeles, New York City. Many times when I ran the clustering algorithm these two cities were in a group of their own. Not only do these places have the greatest populations, the economic output for these metro areas trump the rest by a sight. The map being shown in the app also includes Chicago and some NYC suburbs in this group (the NYC suburbs get pulled into this group because they reside in the NYC metro area, so their economic output is massive). Besides this group, the clustering did a good job of breaking cities into tiers. For example, big cities with big economies (Philly, Detroit, Indianapolis, etc.) are all put into a group. Likewise, it is interesting to see how a city lying near a city from either of these top groups tend to be placed together in either the blue or purple group.

Of course the data is not perfect. Because the data came from various sources, there were nuances I accounted for to match them as closely as possible, but perfectly alligning the data proved difficult. For example, Washington D.C. and Anchorage do not have a group on the map because the two sources listed its county as completely different things. Ideally, I would have used data that historically listed both the population and economic output of each city, and seen how the classification and prestige of different cities change over time. Regardless, this visualization is extremely interesting and easy to navigate.

Appendix of Code

```r
kmeans <- function(X, k) {
  X <- scale(X)
  centroids <- list()
  centroids <- X[sample(1:nrow(X), k), ]
  oldGroups <- numeric(nrow(X))
  newGroups <- 1:nrow(X)

  while(!identical(oldGroups, newGroups)) {
    oldGroups <- newGroups
    #make assignments based on distance
    newGroups <- sapply(1:nrow(X), function(i){
      row <- X[i, ]
      return(bestCentroid(row, centroids))
    })
    #update centroids
    for (c in 1:k) {
      group <- X[which(newGroups == c),]
      centroids[c,] <- betterCentroid(group)
    }
  }
  return(newGroups)
}

bestCentroid <- function(row, centroids) {
  distancia <- sapply(1:nrow(centroids), function(i) {
    return(dist(rbind(row, centroids[i,])))
  })
  return(which.min(distancia))
}

betterCentroid <- function(group) {
  toReturn <- numeric(ncol(group))
  toReturn <- sapply(seq_along(toReturn), function(i) mean(group[ ,i]))
  return(toReturn)
}
```

```r
regionalMap <- function(k) {
  groupos <- kmeans(threeVars, k)
  prettycolors <- rainbow(k, alpha = 0.2)

  map("world", fill = TRUE, col = "grey")

  for (i in 1:k) {
    y <- threeVars[which(groupos == i),1]
    x <- threeVars[which(groupos == i),2]
    inds <- chull(x, y)
    polygon(x[inds], y[inds], col = prettycolors[i])
  }

  symbols(countryData$longitude, countryData$latitude,
          add = TRUE, inches = FALSE, circles = radii,
          fg = mortColors, bg = mortColors)
  legend("topleft", legend = levels(mortBreaks), col = colores, pch = 19, cex = .5, title = 'Levels', bg
}

#~~~~Preparing Extension Data~~~~~

cityData <- read.csv("extensionData/1790-2010_MASTER.csv",
                     stringsAsFactors = FALSE)
countyAndMSA <- read.csv('extensionData/county_msa.csv',
                         stringsAsFactors = FALSE)
msaAndGDP <- read.csv('extensionData/gdp_metro0917.csv',
                      stringsAsFactors = FALSE)
#Get rid of cities with no population in 2010
cityData <- cityData[-which(cityData$X2010 == 0),]
#First clean the counties by adding the state
cityData$CountyName2 <- paste(cityData$County_Name,
                              cityData$ST,
                              sep = ", ")
#Left join of cities with their MSA by county
citiesWithMSA <- left_join(cityData,
                           countyAndMSA,
                           by = c("CountyName2" = "County"))
#Fix some shit
msaAndGDP$U.S..metropolitan.areas <- gsub(', .*$', '',
                                          msaAndGDP$U.S..metropolitan.areas)
citiesWithMSA$MSA <- gsub(', .*$', '',
                          citiesWithMSA$MSA)
#Left join of new data set with GDP by metro area
data <- left_join(citiesWithMSA,
                  msaAndGDP,
                  by = c("MSA" = "U.S..metropolitan.areas"))

#~~~~Making the map and app~~~~~
set.seed(1)
m <- leaflet() %>% addTiles()

maxrad <- 30
palette <- brewer.pal(5, "Blues")
```

```r
yearsToGraph <- c('X1790', 'X1850', 'X1900', 'X1950', 'X2010')
for (y in yearsToGraph) {
  i <- which(names(data) == y)

  nonzero <- data[which(data[,i]!=0),]
  nonzero <- nonzero[which(!is.na(nonzero$LAT)),]

  tiers <- quantile(nonzero[,i], probs = seq(0,1,.2))
  tiers[1] <- tiers[1] - .0005

  cityPopGroup <- cut(nonzero[,i], tiers)
  cityPopGroupColors <- palette[cityPopGroup]

  s <- sqrt(nonzero[,i])
  k <- maxrad/max(s)

  m <- m %>% addCircleMarkers(radius = s*k,
                              lng = nonzero$LON, lat = nonzero$LAT,
                              color = cityPopGroupColors, fillOpacity = 0.2,
                              group = gsub('X', '', y))
}

metros <- unique(data$MSA)
metros <- metros[!is.na(metros)]
flashyCities <- numeric(length(metros))
for (i in 1:length(metros)) {
  tro <- metros[i]
  oneMetro <- data[which(data$MSA == tro),]
  flashyCities[i] <- oneMetro$City[which.max(oneMetro$X2010)]
}
dataForGroups <- data[which(data$City %in% flashyCities), ]
dataForGroups <- dataForGroups[which(!is.na(dataForGroups$LON)),]
dataForGroups <- dataForGroups[which(!is.na(dataForGroups$X2015)),]
X <- dataForGroups[, c('X2010','X2015')]
X <- as.matrix(X)
assignments <- kmeans(X, 5)
groupPalette <- c('red', 'green', 'orange', 'purple', 'blue')
dataForGroups$grolors <- groupPalette[assignments]

icons <- awesomeIcons(icon = 'ion-cash',
                      iconColor = 'black', library = 'ion',
                      markerColor = dataForGroups$grolors)

m <- m %>% addAwesomeMarkers(lng = dataForGroups$LON, lat = dataForGroups$LAT,
                             icon = icons, label = dataForGroups$City,
                             group = 'Econ and Pop Group')

m <- m %>% addLayersControl(baseGroups = c('1790', '1850', '1900', '1950', '2010'),
                            overlayGroups = c('Econ and Pop Group'),
                            options = layersControlOptions(collapsed = TRUE))
m <- m %>% hideGroup('Econ and Pop Group')

ui <- fluidPage(
  titlePanel('Population Shifts over Time'),
  sidebarLayout(position = 'right',
                sidebarPanel(
                  h2('Using the Map', align = 'center'),
                  p("This map can be used to look at population changes over the history of the United S
                    style = "font-family: 'times'; font-si16pt"),
                  p('Toggling the "Econ and Pop Group" button will display a whole bunch of markers. Eac
                    em('kmeans'),
                    'clustering, these cities are grouped based on their population and economic output
                    style = "font-family: 'times'; font-si16pt"
                  ),
                  p('The economic output data is from 2015, and the groups are based on 2010 census popu
                ),
                mainPanel(
                  leafletOutput("map")
                )
  )
)

server <- function(input, output) {
  output$map <- renderLeaflet(m)
}

shinyApp(ui, server)
```