Homework 1

Warlon Zeng

October 31, 2016

Question 1

Part A

Manually removed this piece of text from data.txt (downloaded data.txt @ http://www.google.org/flutrends/about/data/flu/us/data.txt). Removed introductory paragraphs at beginning. We answer the task by summary() because descriptive statistics are mean, median, mode, etc. things of that nature. We also answer the task by plotting a visual, where it may be better to look at where the data spikes in the event of outliers. One observation is that around the start of every year, when it is coldest, search queries spike. Region 10 generally queries more than region 1 overall.

```
df <- read.csv("data.txt") # read in data.txt stored locally in same folder as hw.R. df is short for da

region1 <- df$HHS.Region.1..CT..ME..MA..NH..RI..VT. # $ sign is an operator to access column. After $hh
region10 <- df$HHS.Region.10..AK..ID..OR..WA. # After $hhs I have auto completion to find Region.
summary(region1) # print descriptive statistics
```

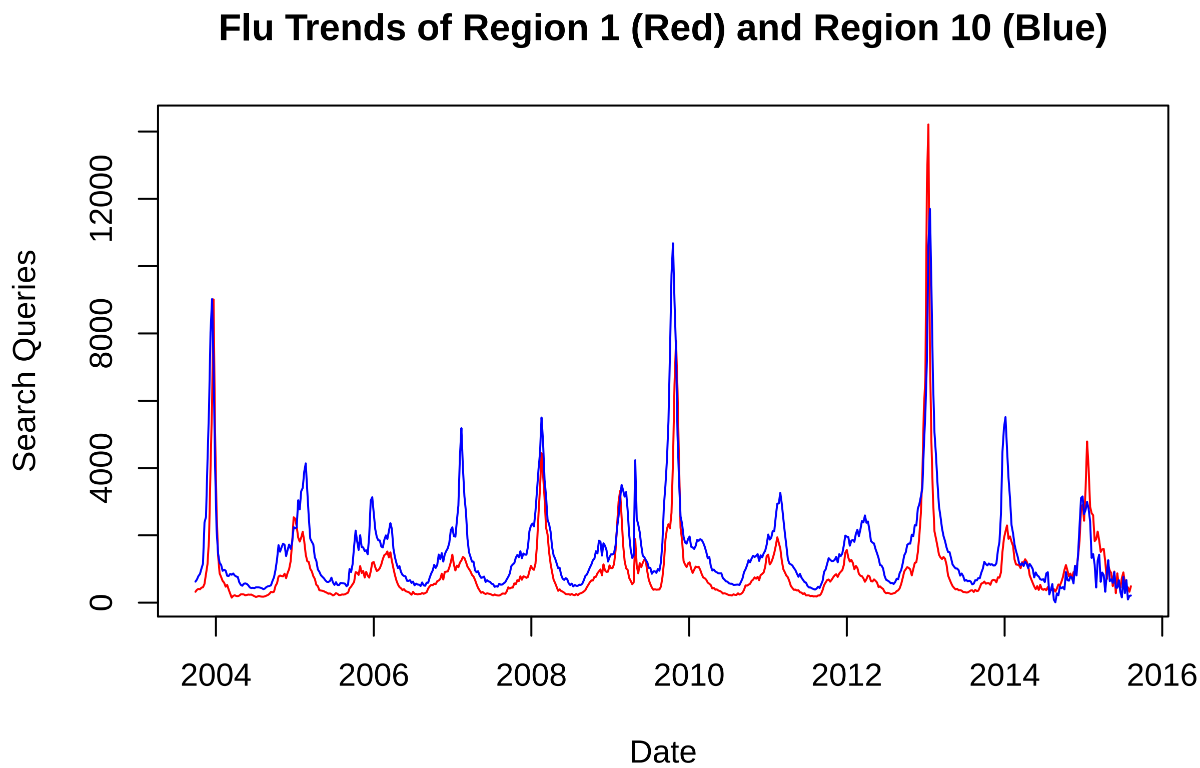
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   149.0   361.5   682.5   986.4  1115.0 14210.0
```

```
summary(region10) # print descriptive statistics
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    10.0   691.8  1179.0  1589.0  1868.0 11700.0
```

```
date <- as.Date(df$Date) # get date to numeric

plot(date, region1, type="l", col = "red", main="Flu Trends of Region 1 (Red) and Region 10 (Blue)", xla
lines(date, region10, col="blue") # multiple lines in one graph here: http://stackoverflow.com/question
```



Part B

We already know there are missing data so we will use zoo package to deal with it. Zoo has spline, a polynomial interpolation. Good for smooth curves. Not perfect, but suitable. Only mesa and scottsdale are missing values, so we will simply fill them in with comparable data from those who have it (phoenix, tempe, tuscon). Plotting the graph indicates little error in cleaning data, but can be further fine tuned. I used the mean to restore/guess previous peak values and their rise and fall. I used linear instead of polynomial because the data spikes in a spikey fashion, not smoothy down.

```
# if you don't have zoo package, perform install.packages('zoo') first.
require(zoo)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

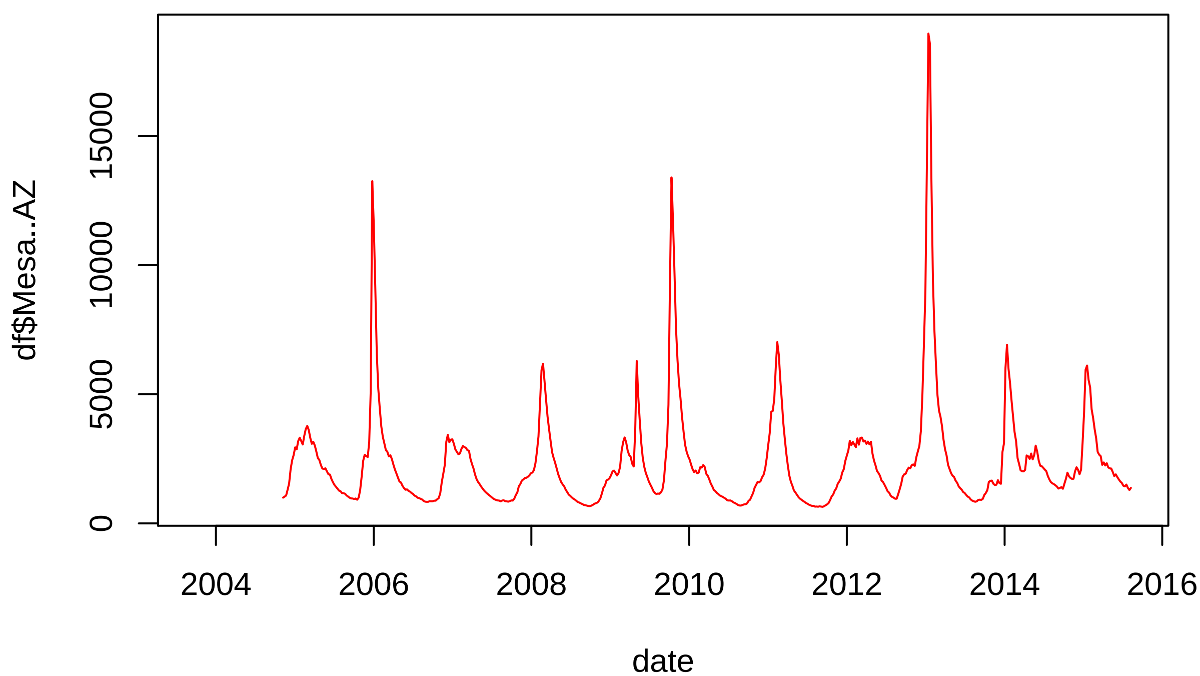
```
phoenix <- df$Phoenix..AZ # phoenix has no NA's so we do not need to clean it
df_phoenix <- data.frame(phoenix)
```

```r
#plot(date, df_phoenix$phoenix, type = 'l', col = 'green')

tempe <- df$Tempe..AZ # tempe has no NA's so we do not need to clean it
df_tempe <- data.frame(tempe)
#plot(date, df_tempe$tempe, type = 'l', col = 'blue')

tucson <- df$Tucson..AZ # tempe has no NA's so we do not need to clean it
df_tucson <- data.frame(tucson)
#plot(date, df_tucson$tucson, type = 'l', col = 'purple')


df_mesa <- na.spline(df$Date, df$Mesa..AZ)
plot(date, df$Mesa..AZ, type = 'l', col = 'red')
```

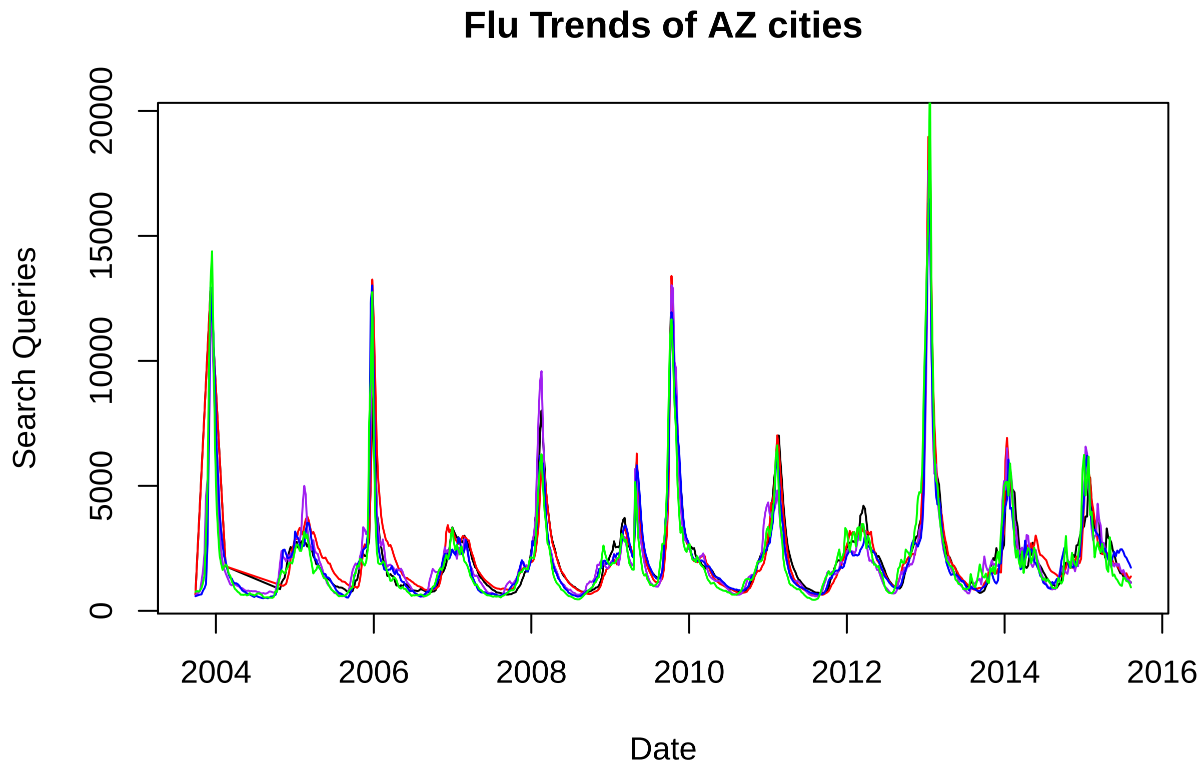



```r
mesa <- df$Mesa..AZ # visually means nothing in table
df_mesa <- data.frame(mesa) # data frame with NA's
df_mesa$mesa[1] <- mean(c(df_phoenix$phoenix[1], df_tempe$tempe[1], df_tucson$tucson[1]))
df_mesa$mesa[11] <- mean(c(df_phoenix$phoenix[11], df_tempe$tempe[11], df_tucson$tucson[11]))
df_mesa$mesa[21] <- mean(c(df_phoenix$phoenix[21], df_tempe$tempe[21], df_tucson$tucson[21]))
df_mesa <- na.approx(df_mesa) # i am using linear interpolation instead of polynomial interpolation to

scottsdale <- df$Scottsdale..AZ
df_scottsdale <- data.frame(scottsdale)
df_scottsdale$scottsdale[1] <- mean(c(df_phoenix$phoenix[1], df_tempe$tempe[1], df_tucson$tucson[1])) #
df_scottsdale$scottsdale[11] <- mean(c(df_phoenix$phoenix[11], df_tempe$tempe[11], df_tucson$tucson[11]
```

```r
df_scottsdale$scottsdale[21] <- mean(c(df_phoenix$phoenix[21], df_tempe$tempe[21], df_tucson$tucson[21]
df_scottsdale <- na.approx(df_scottsdale) # i am using linear interpolation instead of polynomial inter

plot(date, df_scottsdale, type = 'l', col = 'black', main="Flu Trends of AZ cities", xlab = "Date", yla
lines(date, df_mesa, type = 'l', col = 'red')
lines(date, df_tucson$tucson, type = 'l', col = 'purple')
lines(date, df_tempe$tempe, type = 'l', col = 'blue')
lines(date, df_phoenix$phoenix, type = 'l', col = 'green')
```



Part C

Source: http://www.infoplease.com/us/states/population-by-rank.html. The most recent year is 2015. 2016 did not finish yet. The population for one year remains the same throughout that year. So the population recorded in 2015 will be 1 number. In conclusion: there is little evidence to support population having a strong relationship with flu queries in the year 2015. Points are all spread and no clear pattern. Data is dependent on categorical since we are using single data points in an attempt to explain flu queries over a year We'd have to resort to binning and such smooth the data.

```r
#install.packages("XML")
require(XML)
```

```
## Loading required package: XML
```

```r
#install.packages("RCurl")
require(RCurl)
```

```
## Loading required package: RCurl
```

```
## Loading required package: bitops
```

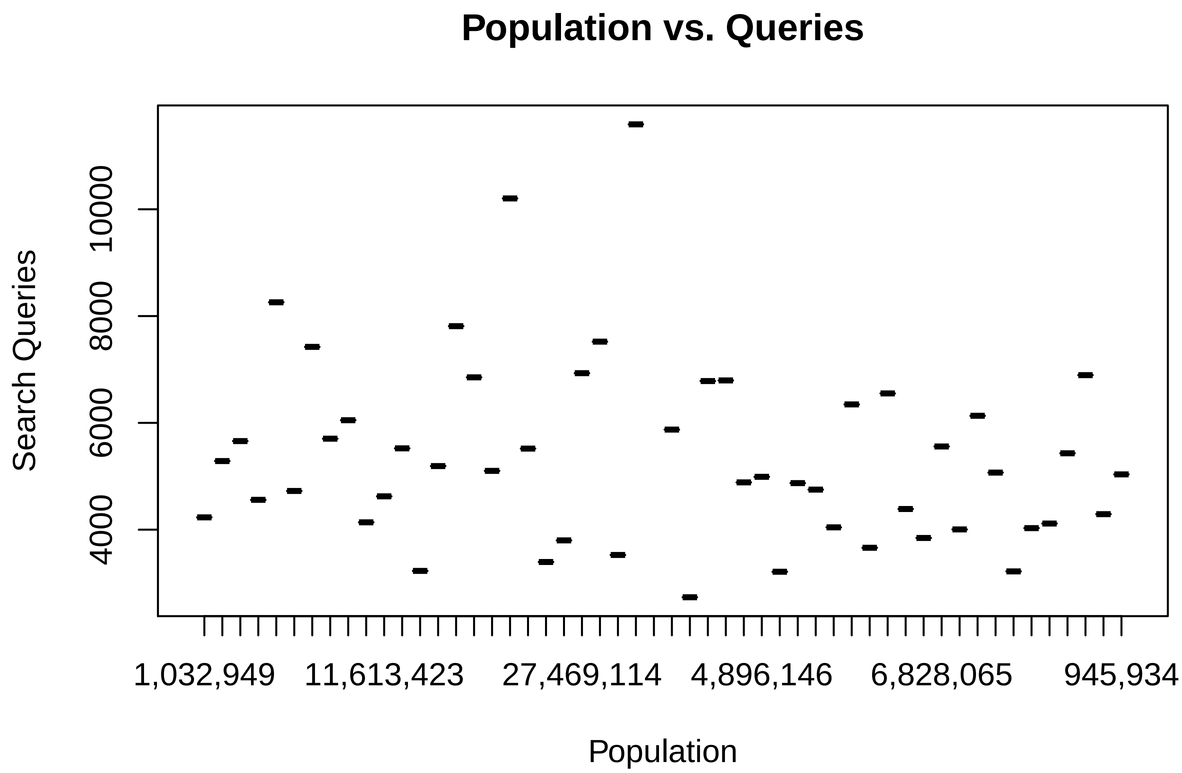
```r
url = getURL("http://www.infoplease.com/us/states/population-by-rank.html") # http://www.infoplease.com
pop2015 <- readHTMLTable(url, header = TRUE, which = 1)
colnames(pop2015) <- c("State", "July 2015. pop") # rename the columns, can be manual -- not computatio
pop2015 <- pop2015[-52,] # delete the total population row, don't need it.

popQuery <- data.frame("population" = integer(0), "queries" = integer(0), "state" = character(0), strin

for (i in 1:51) { # all 50 states
  state <- pop2015$State[i]
  state <- sub(" ", ".", toString(state))
  if (state == "DC") {
    state = "District.of.Columbia" # matching
  }
  queryMax <- max(df[589:620, state]) # 2015JAN-DEC, MAX
  newRow <- data.frame("population" = pop2015$`July 2015. pop`[i], "queries" = queryMax, "state" = stat
  popQuery <- rbind(popQuery, newRow)
}

sortedPop <- data.frame("population" = integer(0), "queries" = integer(0), "state" = character(0), stri
# sort into ascending order
for (i in 51:1) {
  newRow <- data.frame("population" = popQuery$population[i], "queries" = popQuery$queries[i], "state" =
  #sortedPop <- rbind(sortedPop, newRow1)
  #newRow2 <- data.frame("queries" = popQuery$queries[i])
  #sortedPop <- rbind(sortedPop, newRow2)
  #newRow3 <- data.frame("state" = popQuery$state[i])
  sortedPop <- rbind(sortedPop, newRow)
}

plot(sortedPop$population, sortedPop$queries, type = 'l', col = 'black', main="Population vs. Queries",
```




```
#model <- lm(sortedPop$population ~ sortedPop$queries)
#summary(model)

#summary(sortedPop)
#summary(sortedPop$population)
#summary(sortedPop$queries)
```

Part D

I used http://developers.google.com/public-data/docs/canonical/countries_csv for the latitudes. The data set was small so I manually transferred the data. There is absolutely no relationalship with latitudes and search queries for their countries. The multiple R-squared value comes out 0.03837, and even if I had a typo in typing the data in, there is nothing in this. Data set, visually or statistically, defining a relationship.

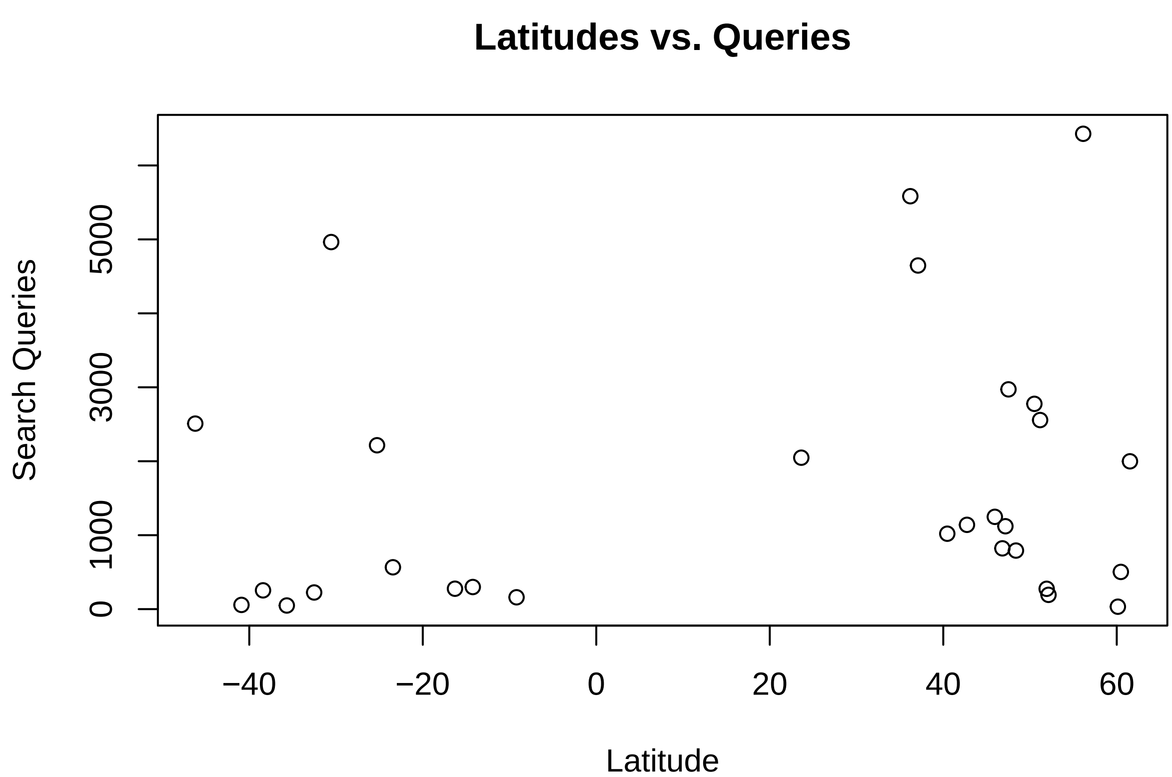
```
df2 <- read.csv("data2.txt") # read in data.txt stored locally in same folder as hw.R. df is short for
lats <- data.frame("latitude" = numeric(0), "country" = character(0))
newRow <- data.frame("latitude" = -38.416097, country = "Argentina")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = -25.274398, country = "Australia")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 47.516231, country = "Austria")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 50.503887, country = "Belgium")
lats <- rbind(lats, newRow)
```

```r
newRow <- data.frame("latitude" = -16.290154, country = "Bolivia")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = -14.235004, country = "Brazil")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 42.733883, country = "Bulgaria")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 56.130366, country = "Canada")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = -35.675147, country = "Chile")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = -46.227638, country = "France")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 51.165691, country = "Germany")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 47.162494, country = "Hungary")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 36.204824, country = "Japan")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 23.634501, country = "Mexico")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 52.132633, country = "Netherlands")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = -40.900557, country = "New Zealand")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 60.472024, country = "Norway")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = -23.442503, country = "Paraguay")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = -9.189967, country = "Peru")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 51.919438, country = "Poland")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 45.943161, country = "Romania")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 61.52401, country = "Russia")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = -30.559482, country = "South Africa")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 40.463667, country = "Spain")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 60.128161, country = "Sweden")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 46.818188, country = "Switzerland")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 48.379433, country = "Ukraine")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = 37.09024, country = "United States")
lats <- rbind(lats, newRow)
newRow <- data.frame("latitude" = -32.522779, country = "Uruguay")
lats <- rbind(lats, newRow)

latQuery <- data.frame("latitude" = numeric(0), "queries" = integer(0), "country" = character(0))
```

```r
for (i in 1:29) {# 29 countries
  country <- lats$country[i]
  country <- sub(" ", ".", toString(country))
  queryMax2 <- max(df2[628:659, country]) # 2015JAN-DEC, MAX
  newRow <- data.frame("latitude" = lats$latitude[i], "queries" = queryMax2, "country" = country)
  latQuery <- rbind(latQuery, newRow)
}

plot(latQuery$latitude, latQuery$queries, type = 'p', col = 'black', main="Latitudes vs. Queries", xlab
```




```r
model <- lm(latQuery$latitude ~ latQuery$queries)
summary(model)
```

```
##
## Call:
## lm(formula = latQuery$latitude ~ latQuery$queries)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -68.77 -37.71  16.90  30.53  48.13
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      11.858686   9.895353   1.198    0.241
## latQuery$queries  0.004259   0.004103   1.038    0.309
```

```
##
## Residual standard error: 38.95 on 27 degrees of freedom
## Multiple R-squared:  0.03837,    Adjusted R-squared:  0.002754
## F-statistic: 1.077 on 1 and 27 DF,  p-value: 0.3085
```

Question 2

Based on the graph, there is not much difference. The sample size was cut short to 155 (n=4) but the general trend and shape remains the same. Statistics are also similar.

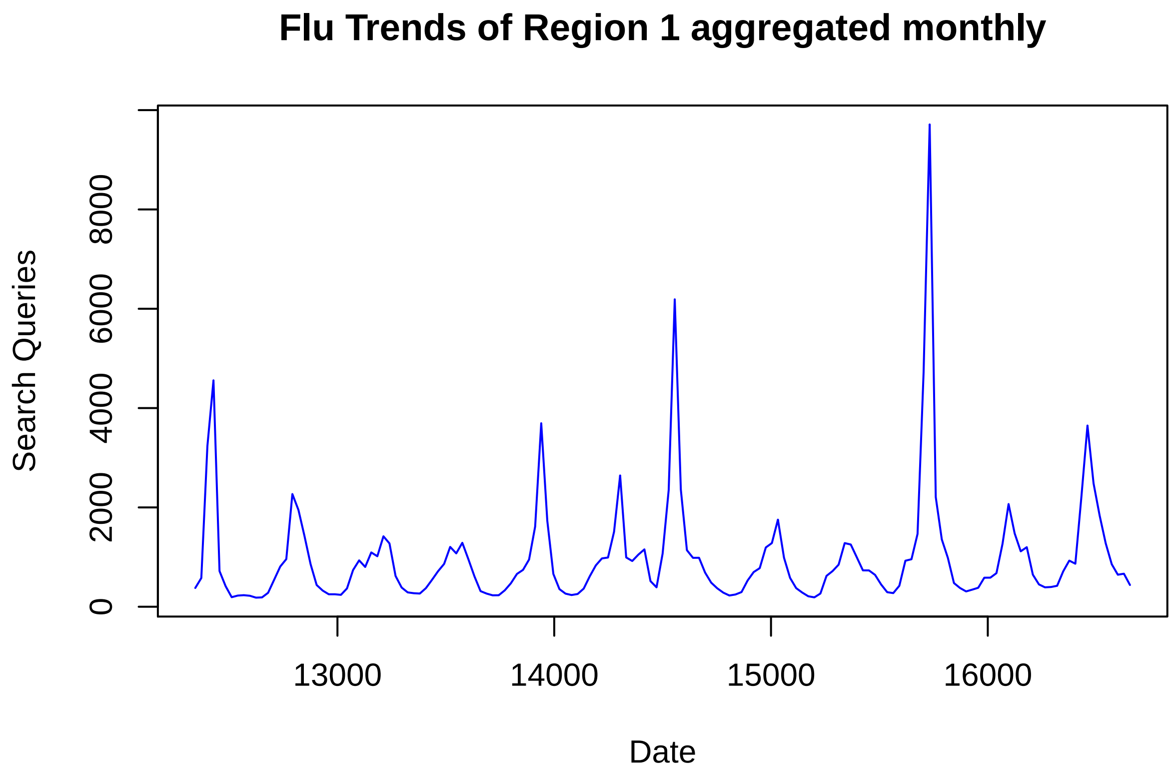
```
df3 <- df
n <- 4
# aggregate averages of rows in every n specified by 1:nrow(..) and in fun = mean
df4 <- aggregate(x = df3, by = list(gl(ceiling(nrow(df3)/n), n)[1:nrow(df3)]), FUN = mean)

df4["Group.1"] = NULL

k <- 1
l <- 1
for (j in 1:620) { # j goes from 1 to 620
  if (k == 4){
    #df4$Date[] <- df3$Date[i]

    df4$Date[l] <- as.Date(df3$Date[j]) # i converted into numerical for graphicability
    #df4$Date[l] <- date[j] # same thing
    #x <- data.frame("Date" = df3$Date[j]) # this is how it should look like
    #df4$Date[l] <- data.frame("Date" = df3$Date[j]) # formatting gets weird here
    k <- 1 # k is just a resetter for n=4
    l <- l + 1 # l goes from 1 to 155
  }
  else {
    k <- k + 1
  }
}

plot(df4$Date, df4$HHS.Region.1..CT..ME..MA..NH..RI..VT., type="l", col = "blue", main="Flu Trends of R
```




```r
model <- lm(df4$Date ~ df4$HHS.Region.1..CT..ME..MA..NH..RI..VT.)
summary(model)
```

```
##
## Call:
## lm(formula = df4$Date ~ df4$HHS.Region.1..CT..ME..MA..NH..RI..VT.)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2525.93 -1010.94   -90.73  1090.57  2225.72
##
## Coefficients:
##                                            Estimate Std. Error t value
## (Intercept)                               1.437e+04  1.334e+02  107.74
## df4$HHS.Region.1..CT..ME..MA..NH..RI..VT. 1.271e-01  8.883e-02    1.43
##                                           Pr(>|t|)
## (Intercept)                                 <2e-16 ***
## df4$HHS.Region.1..CT..ME..MA..NH..RI..VT.    0.155
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1253 on 153 degrees of freedom
## Multiple R-squared:  0.0132, Adjusted R-squared:  0.006747
## F-statistic: 2.046 on 1 and 153 DF,  p-value: 0.1546
```

Question 3

Part A

Recall that I used require() to get XML and RCurl packages for getting 2015 year census population of states. I already installed them earlier.

```
url2 <- getURL("http://www.cdc.gov/mmwr/preview/mmwrhtml/mm6401a4.htm?s_cid=mm6401a4_w") # from assignm
tables1 <- readHTMLTable(url2, header = TRUE, which = 1, stringASFactors=F) # 1st table in the html
tables2 <- readHTMLTable(url2, header = TRUE, which = 2, stringASFactors=F) # 2nd table in the hmtl
```

Part B

I webscrapped an unique page on the web… that uniquely gives webscrapping examples in its own unique way.

```
url3 <- getURL("http://example.webscraping.com/view/Sweden-219")
tables3 <- readHTMLTable(url3, header = TRUE, which = 1, stringASFactors=F) # 1st table in the html
```

Question 4

Did GoViral study