Egypt's Slowest ISP's during November 2009 - September 2011

Monday, April 06, 2015

The Data:

The data set used in this project were downloaded from Measurement Lab's Google BigQuery dataset using the BigQuery graphic user interface and were saved as `.csv`files. Each file's name is of the form `Egypt[YearMo]RTTComplete.csv`, where `Year` $\in$ `{2009, 2010, 2011 }` and `Mo` $\in$ `{Jan, Feb, Mar, Apr,May,June,July,Aug, Sept, Oct, Nov, Dec}`.

I used regular expressions to find the files whose names had the words "Egypt" "RTTComplete" and "csv" in them and saved them into the list, `file_list_Egypt`. I read these files in as data frames and I immediately discarded all the unwanted information, keeping only the relevant (for this project) variables. I accomplished this by applying the function `dataConverter`, which I have written specifically for this purpose. Then I stored these files as data frames into the list egyptDFList, where the data frames' names are of the form: `Egypt[YearMO]`. These names are stored in the vector `dfNames`.

```
setwd("~/M_LabProject")
library(dplyr)

source("dataConverterFunction.R")
source("ipAddressLocatorFunction.R")
#used to change the class of lat. and long. later on
as.numeric.factor <- function(x) {as.numeric(levels(x))[x]}
file_list_Egypt <- list.files(pattern="Egypt.*RTTComplete.*csv")

monthCount_Egypt <- length(file_list_Egypt) #number of files
egyptDFList <- list()
dfNames <- c()
for (i in file_list_Egypt){
        x <- substring(i, 1, nchar(i)-15)
        dummy<- read.csv(i, colClasses="character")
        egyptDFList[[x]] <- dataConverter(dummy)
        dfNames<-c(dfNames, x)
}
```

Finding the Extreme RTT Values and the corresponding ISP's

Next, I found the monthly five number summaries and the inter quartile ranges for the roundtrip times, `RTT`'s. I also kept track of the top (right) outlier boundary for each data frame, i.e. each month, where an `RTT` would be classified as an outlier if it is greater than or equal to that months `median + 1.5*IQR`. I then split up each monthly data frame into two parts, one with the outlier `RTT` values, the other with the rest of the data, and I saved the resulting data frames into the two lists: `outlierDFList` and `normalDFList`.

```
topOutlierBndry <-rep(0,monthCount_Egypt)
outlierDFList<-list()
normalDFList <- list()
for (i in dfNames){
        x<-fivenum(egyptDFList[[i]]$RTT)
        iqr<-IQR(egyptDFList[[i]]$RTT)
        topOutlierBndry[i]<- x[4] + 1.5*iqr
        the_condition <- egyptDFList[[i]]$RTT >= topOutlierBndry[i]
        outlierDFList[[i]]<- egyptDFList[[i]][the_condition, ]
        normalDFList[[i]]<-egyptDFList[[i]][-the_condition, ]
        normalDFList[[i]]$date <- as.character(normalDFList[[i]]$date)
        outlierDFList[[i]]$date <- as.character(outlierDFList[[i]]$date)
}
```

Next, I took the each "outlier" and "normal" data frame and I grouped each by server IP addressess (`serverIP`) then summarized the mean, median and the number of entries of each group and saved them as data frames in the `byServerIPSummaryLIstand` `byServerNormalIPSummaryList` respectively.

```
byServerIPSummaryList<-list()
byServerNormalIPSummaryList <- list()
for (i in dfNames){
        x <- group_by(outlierDFList[[i]], serverIP)
        byServerIPSummaryList[[i]]<- data.frame(summarise(x, count=n(),
meanRTT = mean(RTT),
                                                medianRTT = median(RTT)))
        y <- group_by(normalDFList[[i]], serverIP)
        byServerNormalIPSummaryList[[i]]<-data.frame(summarise(y, count=n(),
                                meanRTT=mean(RTT), medianRTT=median(RTT)))
}
```

Locating the ISP's on the Map

Getting all the Data Ready

Now, in order to be able to examine the resulting summaries, I wanted to plot each ISP on the map.
Therefore I needed to figure out their locations given and IP address, since the data I got from Measurement Labs was often incomplete when it came to specifying the locations. I ended up using Heuristic Andrew's script for locating IP addresses using R, as it can be found at **https://heuristically.wordpress.com/2013/05/20/geolocate-ip-addresses-in-r/**. That is the scirpt now saved and sourced as `ipAddressLocatorFunction.R`.

Then I applied the address locator function to the distinct `serverIP` values of data frames in the lists `byServerIPSummaryLIstand` `byServerNormalIPSummaryList` and saved the resulting locations in the `outlierServerIPLocationList` and `normalServerIPLocationList` lists. By the way the output of the `ipAddressLocatorFunction` is a data frame with the following column names: `ip` ,

country_code, country_name, region_code, region_name, city
zipcode,latitude,longitude, metro_code, areacode.

```
outlierServerIPLocationList <- list()
normalServerIPLocationList <- list()
for (i in dfNames){
        x <- unique(byServerIPSummaryList[[i]]$serverIP)

        outlierServerIPLocationList[[i]]<- freegeoip(x)

}

for (i in dfNames){
        y <- unique(byServerNormalIPSummaryList[[i]]$serverIP)
        normalServerIPLocationList[[i]]<-freegeoip(y)

}
```

Since all I wanted is the locations of all the servers, irrespectively (for now) of which month's data they came from, I collected the monthly informations per list into two distinct data frames, one for the "outlier" servers, one for the rest.

```
aggregateOutlierServerLocationDF <- data.frame()
aggregateNormalServerLocationDF <- data.frame()
for (i in dfNames){
    aggregateOutlierServerLocationDF <-
rbind(aggregateOutlierServerLocationDF,
outlierServerIPLocationList[[i]] )
    aggregateNormalServerLocationDF <- rbind(aggregateNormalServerLocationDF,
normalServerIPLocationList[[i]])
}
```

In order to not have to replot the same locations over and over again, I grouped the aggregated data by longitude and latitude and then summarized them and found the count per given longitude and latitude.

```
temp <- group_by(aggregateOutlierServerLocationDF, longitude, latitude)
byLongLatOutlierServerDF <- data.frame(summarise(temp, count=n()))
temp <- group_by(aggregateNormalServerLocationDF, longitude, latitude)
byLongLatNormalServerDF <- data.frame(summarise(temp, count=n()))
```

Plotting the Locations on the Map

After a cursory examination of the locations, I found that the ISP's were located either in Europe or in the US (oh and one was located in Australia I beleive, but I did not graph that one.) The locations were not easily discernable on a word map so I chose to plot the European and US servers on separate maps.

Plotting the European Servers

First, I had to find all the ISP's whose longitude and latitude landed them in Europe. After I examined a world map with longitudes and latitudes I decided on the cutoff values for Europe (which later I modified so that Greece would be included too, but then I did not expect to get it perfect on the first try...)

```
#finding unique longitudes and latitudes  in Europe
outlierEuropeCondition <-
as.numeric.factor(byLongLatOutlierServerDF$longitude) <= 30 &
        as.numeric.factor(byLongLatOutlierServerDF$longitude)> -50
normalEuropeCondition <- as.numeric.factor(byLongLatNormalServerDF$longitude)
<= 30 &
        as.numeric.factor(byLongLatNormalServerDF$longitude)> -50

EuropeOutlierDF <- byLongLatOutlierServerDF[outlierEuropeCondition,]

EuropeNormalDF <- byLongLatNormalServerDF [normalEuropeCondition,]
```

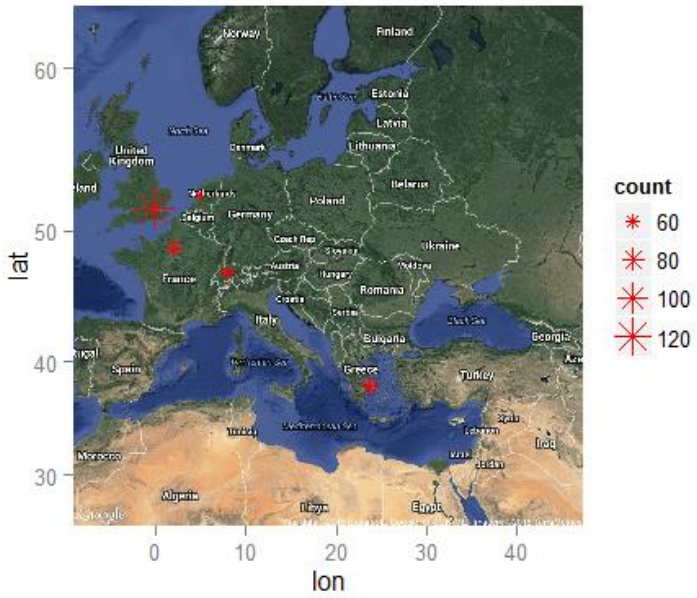
Once I decided on the list of serves that I wanted plotted on the map of Europe, I had to pick a center point for my map (using longitudes and latitudes). For this task I used the website found at *http://www.mapcoordinates.net/en*, where a single click on the map would give me the longitude and latitude of the point of interest. So I pointed and clicked and saved the values in the `Eu_lat` and `Eu_long` variables, then I used the `get_googlemap()` function to create the map. Oh, and for my maps I needed the `ggmap` and `ggplot2` packages, so I loaded those as well.

```
library(ggplot2);library(ggmap)

## Warning: package 'ggmap' was built under R version 3.1.3

Eu_lat<- 47.5172007  #generic Europe latitude
Eu_lon<- 19.07226563 #generic Europe longitude
googMap <- get_googlemap(center = c(lon = Eu_lon , lat = Eu_lat ), zoom =4,
                         maptype='hybrid', scale=2)

## Map from URL :
http://maps.googleapis.com/maps/api/staticmap?center=47.517201,19.072266&zoom
=4&size=640x640&scale=2&maptype=hybrid&sensor=false
```

Now, all is left is to actually plot the maps:

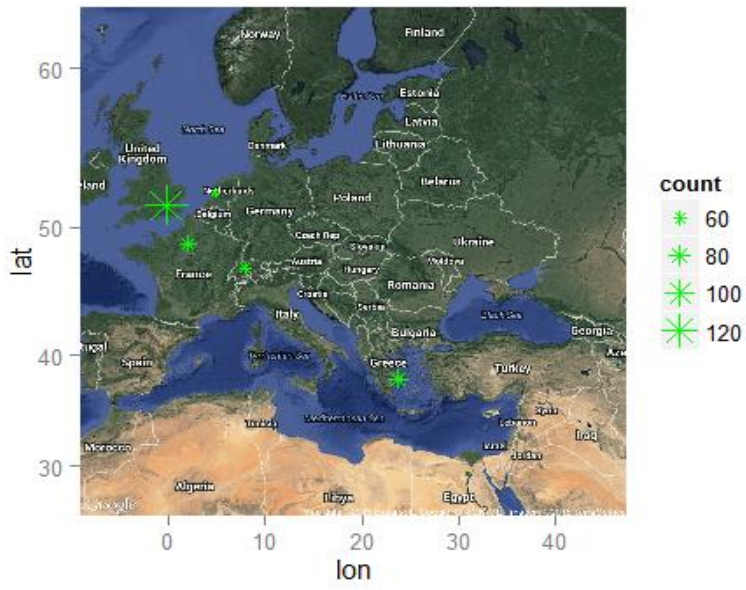
These are the locations of the "outlier" servers in Europe, where `count` represents the number of times the server has appeared in the data frame - more or less...

```
#plotting servers form Europe
ggmap(googMap)+geom_point(
        aes(x =as.numeric.factor(longitude), y =as.numeric.factor(latitude),
            size = count),pch=8, data=EuropeOutlierDF , colour= "red")
```



And these are the locations of the "normal" ISP's in Europe:

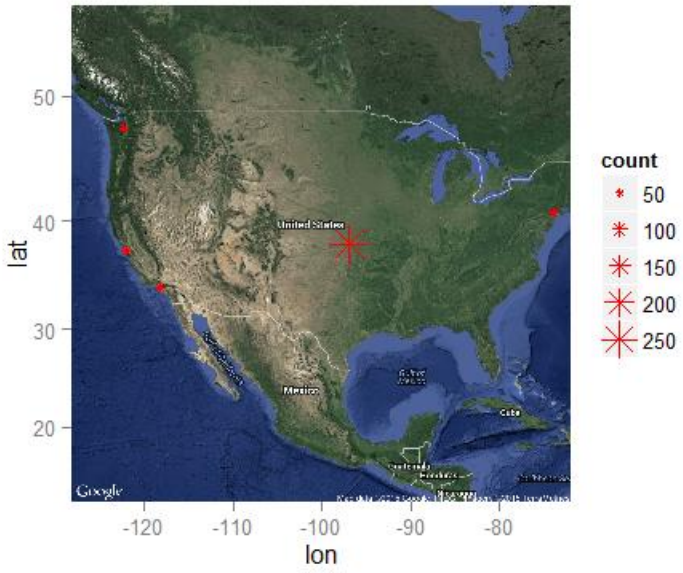
```
ggmap(googMap) +geom_point(
        aes(x =as.numeric.factor(longitude), y =as.numeric.factor(latitude),
            size = count),pch=8, data=EuropeNormalDF , colour= "green")
```



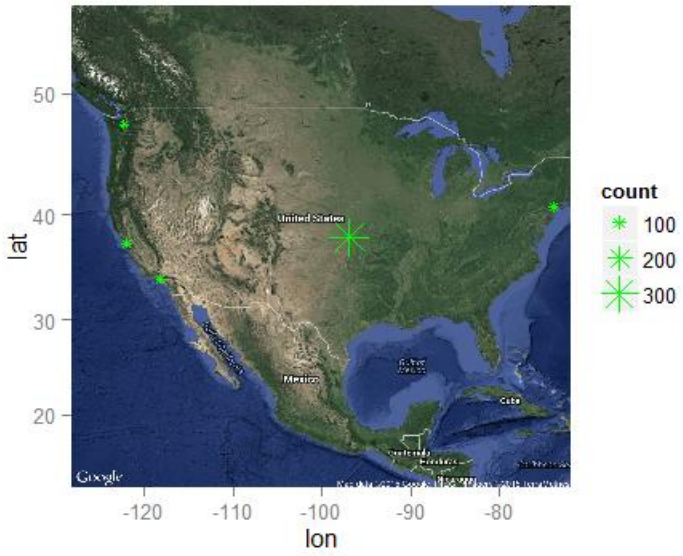
Plotting the US Servers

```
#centering map on US
googMapUS <- get_googlemap(center=c(lon=-100.1953125, lat=37.16031655),
zoom=4,
                           maptype="hybrid", scale=2)
```

```
## Map from URL :
http://maps.googleapis.com/maps/api/staticmap?center=37.160317,-
100.195312&zoom=4&size=640x640&scale=2&maptype=hybrid&sensor=false

#finding unique longitudes and latitudes not in Europe
USOutlierDF <- byLongLatOutlierServerDF[!outlierEuropeCondition,]
USNormalDF <- byLongLatNormalServerDF [!normalEuropeCondition,]

#plotting server locations from the US

#the servers locations include one ISP not in north America, which will be
#ignored by ggmap - fortunately for me

#Plotting the "outlier" servers in the US
ggmap(googMapUS)+geom_point(
        aes(x =as.numeric.factor(longitude), y =as.numeric.factor(latitude),
size=count),
        pch=8, data=USOutlierDF , colour="red")

##Warning: Removed 1 rows containing missing values (geom_point).
```




```
#plotting the "normal" ISP's in the US
ggmap(googMapUS) +geom_point(
        aes(x =as.numeric.factor(longitude), y =as.numeric.factor(latitude),
            size = count),pch=8, data=USNormalDF , colour= "green")

## Warning: Removed 1 rows containing missing values (geom_point).
```



Conclusions

All the ISP's that have served Egypt during the time period of November 2009 and September 2011 have, at one point or another, had `RTT`'s that were in the outlier range of the `RTT`'s of the month during which the measurement was taken since the locations of the "outlier" servers (in red) match up perfectly with the locations of the "normal" servers (in green.)

The `count`'s on the maps are ambigous at this point, since I did not carefully keep track of the number of times a particular ISP has appeared in the "outlier" list.
And, at any rate, I would need to use percentages out of the total number of measurements as opposed to the actual counts to get a reasonable picture of what was happening.