Abstract

The purpose of this group project is to gain an understanding of how to use the RGA package to extract data from the Google Analytics platform and to analyze it using the functionality provided by R.

R GOOGLE ANALYTICS

Windows User

# Executive Summary

Google Analytics is a popular web-based analytics platform which gives website owners insights into their web audience by allowing owners to track website traffic – number of unique visitors per unit time, sources of the website traffic (i.e. websites which led to them “landing” on the website being analyzed), location of users and many other factors over time.

The website owner may also monitor users’ activity on the website to determine which areas of the website are most popular, which links receive the most clicks, how users typically spend their time on the website, if/when users abandon their shopping carts, the keywords used in Search Engine queries which generally lead users to the website, the “bounce rate”, etc.

Web analytics platforms like Google Analytics are critical for conducting hypothesis-based testing related to website features, e.g. A|B Testing.

Google Analytics allows its users two options: the first option is free and has core functionality, while the premium model has more expansive features. In this project, we have explored only the features of the free model. We focused our analysis on the functionality supported by the ***rga*** package in R.

Table of Contents

[Executive Summary 2](#_Toc468230467)

[Setting up 4](#_Toc468230468)

[Importing Google Analytics Data into R 5](#_Toc468230469)

[Part 1: Installing the Required Packages. 5](#_Toc468230470)

[Part 2: Accessing Google Analytics 6](#_Toc468230471)

[Part 3: Querying Google Analytics 7](#_Toc468230472)

[Query 1: Users and Pageviews Over Time 7](#_Toc468230473)

[Query 2: Mobile Traffic 9](#_Toc468230474)

[Query 3: Source Analysis 11](#_Toc468230475)

[Query 4: New vs Returning Sessions 12](#_Toc468230476)

[Query 5: Sessions by Country 13](#_Toc468230477)

[Query 6: Browser and Operating System 14](#_Toc468230478)

[Query 7: Exit Rate per page 16](#_Toc468230479)

[Query 8: Search Engines 17](#_Toc468230480)

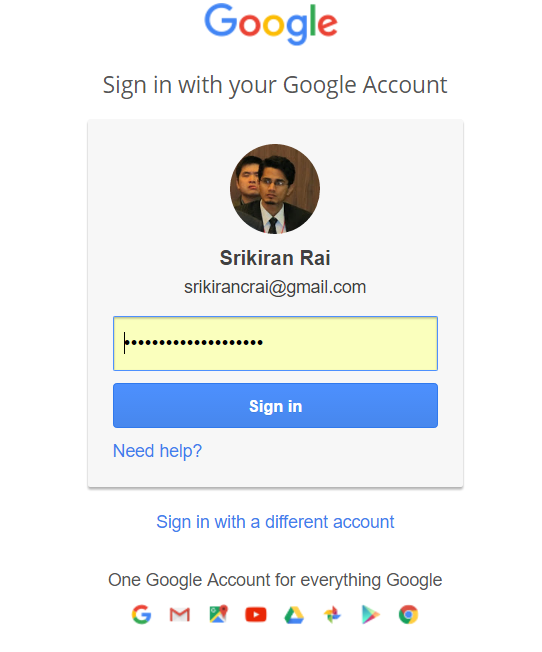
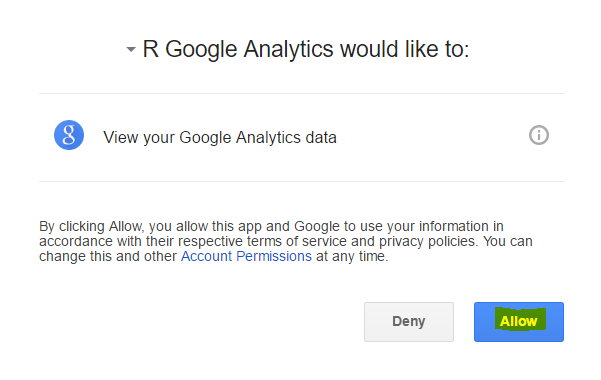
[Query 9: Keywords 18](#_Toc468230481)

[Query 10: Top Content 19](#_Toc468230482)

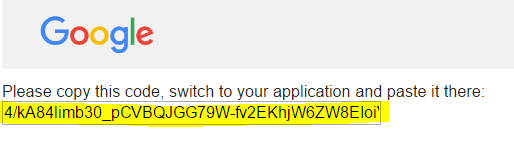
[References 21](#_Toc468230483)

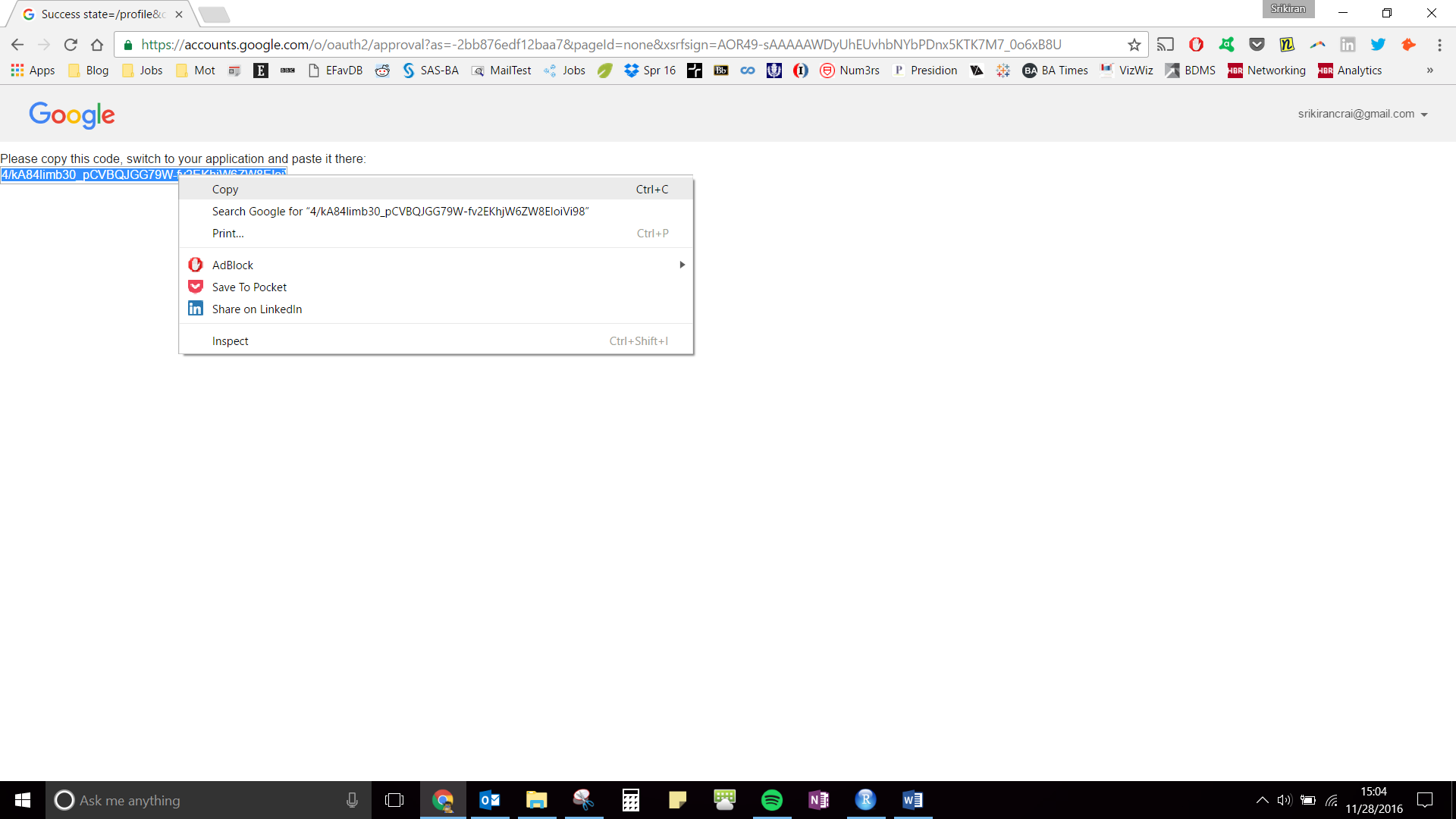
# Setting up

After installing the relevant packages and loading the libraries, the following code will be generated in R, and the user will be redirected to their Google account login page and be prompted to provide the RGA package access to the user’s Google Analytics data.

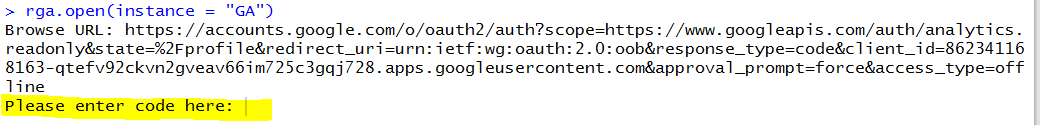
 

Upon signing into the account, and choosing “Allow”, a code string similar to that highlighted below is generated:

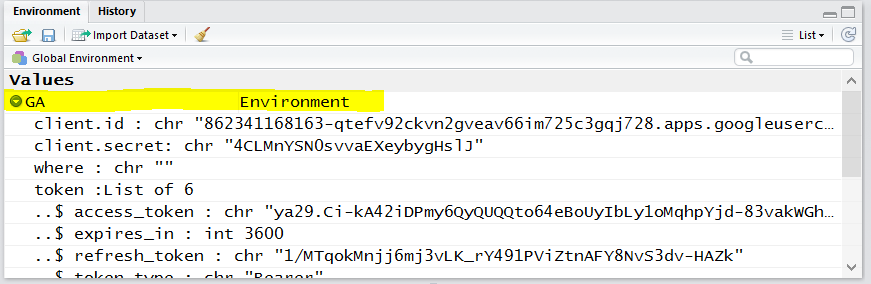




This code is then copied and pasted into the console next to the instruction “Please enter code here:”. The user then must hit the “Enter” button to execute.



After execution, the client ID, client secret and other relevant information will be packaged as an environment instance and will be imported in into the R-Studio global environment.



# Importing Google Analytics Data into R

## Part 1: Installing the Required Packages.

#### Install the packages & load the libraries

install.packages("devtools")

## Installing package into 'C:/Users/tly15101/Documents/R/win-library/3.3'  
## (as 'lib' is unspecified)

## Warning in install.packages :  
## package 'devtools' is in use and will not be installed

library(devtools)  
#In order to extract data from Google Analytics, we are going to use the "rga" package.  
#Package Information:  
 # Package name: rga  
 # Title: R Google Analytics  
 # Description: A package for seemless API connection to Google Analytics  
 # Url: https://github.com/skardhamar/rga  
 # BugReports: https://github.com/skardhamar/rga/issues  
 # Version: 0.8  
 # Date: 2012-14-11  
 # Maintainer: Bror Skardhamar <skardhamar@gmail.com>  
 # Author: Bror Skardhamar <skardhamar@gmail.com>  
#We use the install\_github function to install from github  
install\_github("skardhamar/rga", force = T)

## Downloading GitHub repo skardhamar/rga@master  
## from URL https://api.github.com/repos/skardhamar/rga/zipball/master

## Installing rga

## "C:/PROGRA~1/R/R-33~1.1/bin/x64/R" --no-site-file --no-environ --no-save \  
## --no-restore --quiet CMD INSTALL \  
## "C:/Users/tly15101/AppData/Local/Temp/RtmpqcnyYN/devtools1c02e274218/skardhamar-rga-3386acf" \  
## --library="C:/Users/tly15101/Documents/R/win-library/3.3" \  
## --install-tests

##

## Reloading installed rga

library(rga)  
  
#We will be using ggplot to create some of the charts in our analysis, so install that as well  
install.packages("ggplot2")

## Error in install.packages : Updating loaded packages

library(ggplot2)

## Part 2: Accessing Google Analytics

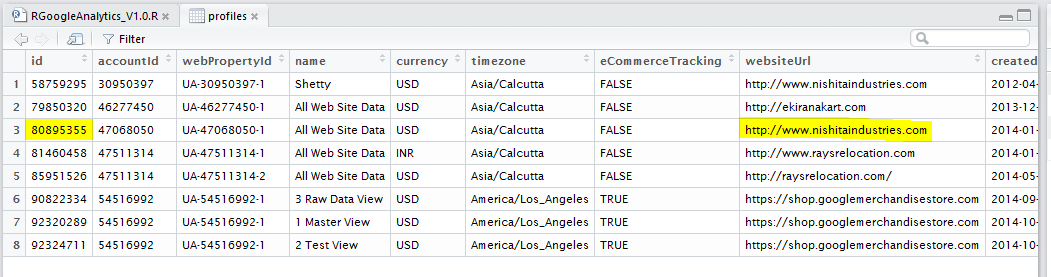
In order to open an instance of *Google Analytics* in RStudio, we use the **rga.open** function. The name of the instance is called "GA". Upon executing this code, the browser window will open and you will be prompted to login to your Google account. You have to enter your gmail ID and password which then gives the client access code.

rga.open(instance = "GA")

Once the Google Analytics instance has been created (see “Setting Up” above), you have access to all the profiles that have been created under your *Google Analytics* account. In order to see which profiles you have access to, run the below command to import profile information.

profiles = GA$getProfiles()  
#Now view all the profiles as a dataframe and choose the profile that you want to access.   
View(profiles)

If the user has access to multiple websites, they will all be listed when the profiles are viewed. The user must retain the ID only for the website whose data he wishes to view. In our case, we are using the Nishita Industries website, so the ID we retain is 80895355.



The user may now use this ID to run all the queries that extract data from *Google Analytics*.

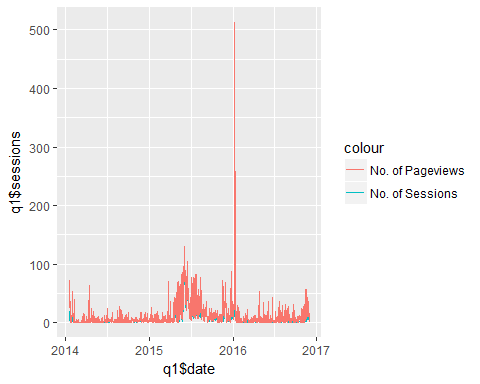
## Part 3: Querying Google Analytics

Google Analytics’ web platform for analyzing user data provides a dashboard interface to the user and a querying tool which helps user get custom reports based on their requirements. Google provides many ‘tags’ that can be used to slice and dice the data. The *rga* package provides the same functionality in the form of the *getData* variable in the R environment that was imported in the last step. This project attempts to replicate few of the most common queries executed by Google Analytics professionals in their goal of optimizing their website for most visits and conversions. (Google, 2016)

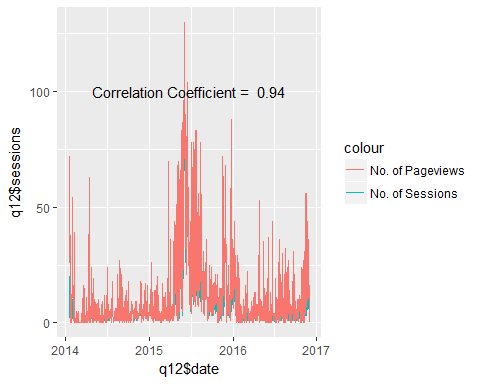
### Query 1: Users and Pageviews Over Time

This query returns the total users and pageviews for the specified time period.

q1 = GA$getData(ids = 80895355,   
 start.date = "2014-01-15",   
 end.date = "2016-11-30",  
 metrics = "ga:sessions,ga:pageviews",   
 sort = "-ga:date",   
 filters = "",   
 segment = "",  
 start = 1, max = 10000)  
View(q1)  
  
#Plot the data using ggplot.  
ggplot(q1, aes(q1$date)) +   
 geom\_line(aes(y = q1$sessions, color = "No. of Sessions")) +   
 geom\_line(aes(y = q1$pageviews, color = "No. of Pageviews"))



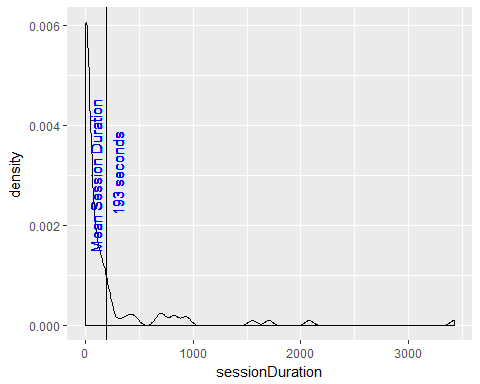
#We see that there is a significant outlier on row 327 for pageviews  
#We try to find out the source of all the clicks on that day.  
#To extract the source of the pageviews we have to add another dimension to the query  
#And restrict the date range  
q11 = GA$getData(ids = 80895355,   
 start.date = "2016-01-09",   
 end.date = "2016-01-09",  
 dimensions = "ga:date,ga:source,ga:medium",  
 metrics = "ga:sessions,ga:pageviews",   
 sort = "-ga:date",   
 filters = "",   
 segment = "",  
 start = 1, max = 10000)  
View(q11)  
#This website turns out to be a website analyzer for search engine optimization.  
#This has contributed to the spurious accesses to the website  
#Hence we get rid of that record to better understand the pageviews and sessions over time.  
q12 = q1[-327,] #to remove outlier  
#We now try and visualize the change in sessions and pageviews over time  
  
p1 = ggplot(q12, aes(q12$date)) +   
 geom\_line(aes(y = q12$sessions, color = "No. of Sessions")) +   
 geom\_line(aes(y = q12$pageviews, color = "No. of Pageviews"))  
  
p1 + annotate("text", x = mean(q12$date), y = 100, label = paste("Correlation Coefficient = ", round(cor(q12$sessions, q12$pageviews),2)))



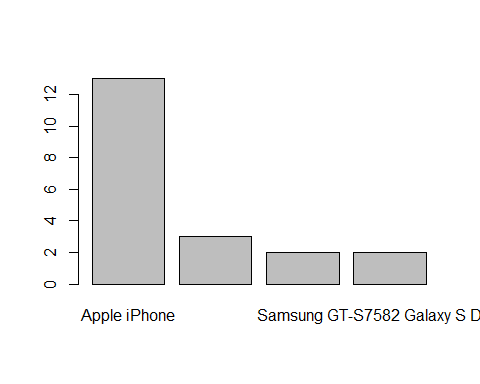
### Query 2: Mobile Traffic

This query returns some information about sessions which occurred from mobile devices. Note that "Mobile Traffic" is defined using the default segment ID -14.

q2 = GA$getData(ids = 80895355,   
 start.date = "2014-01-12",   
 end.date = "2016-11-30",   
 dimensions="ga:mobileDeviceInfo,ga:source",  
 metrics="ga:sessions,ga:pageviews,ga:sessionDuration",  
 filters = "",   
 segment = "gaid::-14",  
 start = 1, max = 10000)  
View(q2)  
  
  
p2 = ggplot(q2, aes(sessionDuration)) + geom\_density() + geom\_vline(xintercept = mean(q2$sessionDuration))  
p2 + geom\_text(aes(x=mean(q2$sessionDuration), label=paste("Mean Session Duration\n", as.character(round(mean(q2$sessionDuration)),2), "seconds"), y=0.003), colour="blue", angle=90)



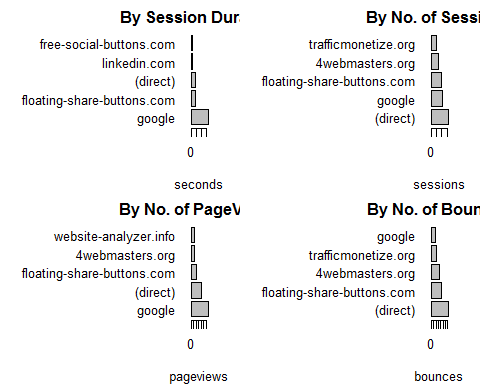
p22 = with(q2, barplot((sort(table(q2$mobileDeviceInfo[q2$mobileDeviceInfo!="(not set)"]), decreasing = T))[1:4]))



### Query 3: Source Analysis

This query returns information about page views, sessions, session duration and bounces from different sources such as Google, Facebook, etc.

q3 = GA$getData(ids = 80895355,   
 start.date = "2014-01-12",   
 end.date = "2016-11-30",  
 dimensions="ga:source,ga:medium", metrics="ga:sessions,ga:pageviews,ga:sessionDuration,ga:bounces",  
 filters = "",   
 start = 1, max = 10000)  
View(q3)  
  
#Trying to extract top 5 sources for each of the metrics  
opar = par()  
par(mfrow = c(2,2), mar=c(4,12,2,2))  
  
#First showing major sources by Session Duration  
q311 = q3[q3$sessionDuration > 2, ] #Only considering those sources where the session lasted longer than 2 seconds  
q312 = q311[order(-q311$sessionDuration), ] #Sorting the sources by decreasing order of session duration  
q313 = q312[1:5, ] #Subsetting the top 5  
View(q313)  
  
with(q313, barplot(height = q313$sessionDuration, names.arg = q313$source, las = 1, horiz = T, main = "By Session Duration", xlab = "seconds"))  
  
#Now considering major sources of sessions  
q321 = q3[q3$sessions > 10, ]#Only considering those sources which came more than 10 times  
q322 = q321[order(-q321$sessions), ]#Sorting the sources by decreasing order of no. of sessions  
q323 = q322[1:5, ] #Subsetting the top 5  
View(q323)  
  
with(q323, barplot(height = q323$sessions, names.arg = q323$source, las = 1, horiz = T, main = "By No. of Sessions", xlab = "sessions"))  
  
#Now considering major sources of pageviews  
q331 = q3[q3$pageviews > 50, ]#Only considering those sources which resulted in more than 50 pageviews  
q332 = q331[order(-q331$pageviews), ]#Sorting the sources by decreasing order of no. of pageviews  
q333 = q332[1:5, ] #Subsetting the top 5  
View(q333)  
  
with(q333, barplot(height = q333$pageviews, names.arg = q333$source, las = 1, horiz = T, main = "By No. of PageViews", xlab = "pageviews"))  
  
#Now considering the sources by most number of "bounces"  
q341 = q3[q3$bounces > 10, ]#Only considering those sources which came AND LEFT more than 10 times  
q342 = q341[order(-q341$bounces), ]#Sorting the sources by decreasing order of no. of sessions  
q343 = q342[1:5, ] #Subsetting the top 5  
View(q343)  
  
with(q343, barplot(height = q343$bounces, names.arg = q343$source, las = 1, horiz = T, main = "By No. of Bounces", xlab = "bounces"))

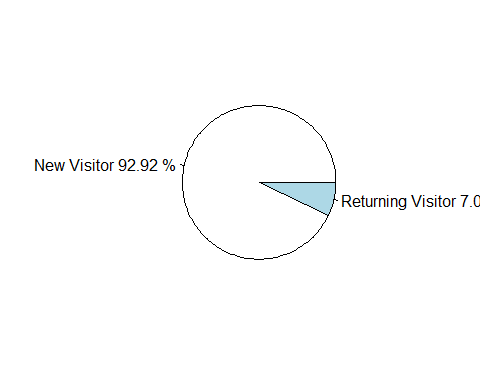


par(opar)

### Query 4: New vs Returning Sessions

This query returns the number of new sessions vs returning sessions.

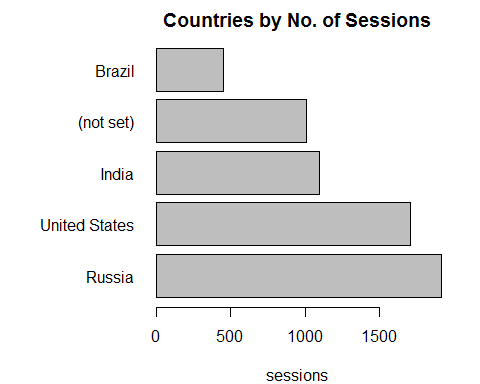
q4 = GA$getData(ids = 80895355,   
 start.date = "2014-01-12",  
 end.date = "2016-11-30",  
 dimensions="ga:userType",#The userType tag shows which kind of user  
 metrics="ga:sessions",  
 filters = "",   
 start = 1, max = 10000)  
View(q4)  
attach(q4)  
#Find percentage of sessions  
percentages = round(sessions/sum(sessions)\*100, 2)  
#Create a pie chart to view the proportions  
pie(x = sessions, labels = paste(userType, percentages, "%"))



### Query 5: Sessions by Country

This query returns a breakdown of your sessions by country, sorted by number of sessions.

q5 = GA$getData(ids = 80895355,   
 start.date = "2014-01-12",   
 end.date = "2016-11-30",  
 dimensions="ga:country",#The country tag provides the source country  
 metrics="ga:sessions",  
 sort = "-ga:sessions",  
 filters = "",   
 start = 1, max = 10000)  
View(q5)  
  
q51 = q5[1:5, ] #Subsetting the top 5  
View(q51)  
opar = par()  
par(mar=c(4,8,2,2))  
with(q51, barplot(height = q51$sessions, names.arg = q51$country, las = 1, horiz = T, main = "Countries by No. of Sessions", xlab = "sessions"))

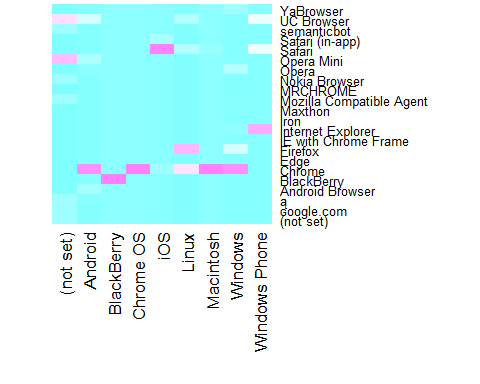


par(opar)

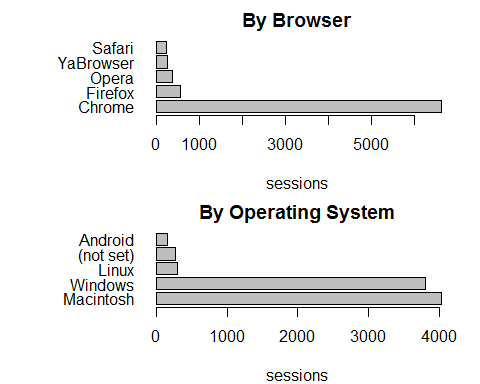
### Query 6: Browser and Operating System

This query returns a breakdown of sessions by the Operating System, web browser, and browser version used.

q6 = GA$getData(ids = 80895355,   
 start.date = "2014-01-12",   
 end.date = "2016-11-30",  
 dimensions="ga:operatingSystem,ga:operatingSystemVersion,ga:browser,ga:browserVersion",  
 #The above 4 tags provide the information on operating system, version of the operating system  
 #the browser type and the browser version  
 metrics="ga:sessions",  
 sort = "-ga:sessions",  
 filters = "",   
 start = 1, max = 10000)  
View(q6)  
  
#Now we want to look at Operating system and browser combinations that are the most common.  
#This is done because a browser can be used on multiple operating systems and multiple operating systems   
#can be using the same browser  
q61 = as.matrix(table(x = q6$browser, y = q6$operatingSystem), decreasing = T)  
heatmap(q61, col = cm.colors(256), Rowv=NA, Colv=NA, scale="column", margins=c(10,10))



#Now let us look at the top operating systems and browser by number of sessions  
#For that we will first subset only the required information  
q62 = q6[, c("operatingSystem", "browser", "sessions")]  
  
#We now have to add all the sessions for browsers. For that we use the aggregate function  
q63 = aggregate(q62$sessions, by=list(Category=q62$browser), FUN=sum)  
#The FUN = sum adds all the sessions grouped by the category browser  
q631 = q63[order(-q63$x), ]#Sorting the sources by decreasing order of no. of sessions  
q632 = q631[1:5, ] #Subsetting the top 5  
View(q632)  
  
#We now have to add all the sessions for operating systems.  
q64 = aggregate(q62$sessions, by=list(Category=q62$operatingSystem), FUN=sum)  
#The FUN = sum adds all the sessions grouped by the category Operating System  
q641 = q64[order(-q64$x), ]#Sorting the sources by decreasing order of no. of sessions  
q642 = q641[1:5, ] #Subsetting the top 5  
View(q642)  
  
#Now create two bar plots to represent these findings graphically  
opar = par()  
par(mar=c(4,8,2,2), mfrow = c(2,1))  
with(q632, barplot(height = q632$x, names.arg = q632$Category, las = 1, horiz = T, main = "By Browser", xlab = "sessions"))  
with(q642, barplot(height = q642$x, names.arg = q642$Category, las = 1, horiz = T, main = "By Operating System", xlab = "sessions"))

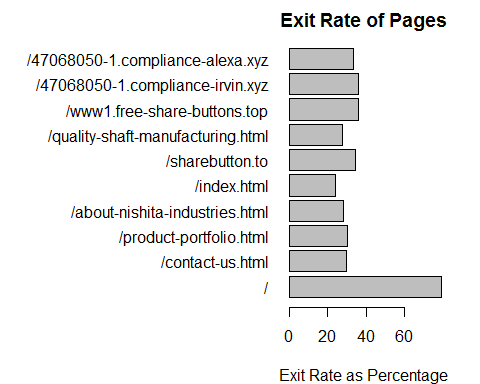


par(opar)

### Query 7: Exit Rate per page

This query returns the site usage data broken down by source and medium, sorted by sessions in descending order.

q7 = GA$getData(ids = 80895355,   
 start.date = "2014-01-12",   
 end.date = "2016-11-30",  
 dimensions="ga:exitPagePath", #This provides the path of the page from which a user exited  
 metrics="ga:pageviews,ga:exits", #Gives the number of pageviews and exits  
 sort="-ga:exits",  
 filters = "",   
 start = 1, max = 10000)  
View(q7)  
#The two metrics - pageviews and exits have to seen in relation to each other.   
#Hence we need to find the exit rate, that is the percentage of people who are leaving that particular page  
q7$exitrate = q7$exits/q7$pageviews\*100  
View(q7)  
#As we can see, there are some pages which have an exit rate of 100%  
#These pages are seen to be mostly coming in from spam websites  
#Hence we filter those out  
q7 = q7[q7$exitrate < 100, ]  
View(q7)  
#Now selecting only the top 10  
q71 = q7[1:10, ]  
opar = par()  
par(mar=c(4,15,2,2))  
with(q71, barplot(height = q71$exitrate, names.arg = q71$exitPagePath, las = 1, horiz = T, main = "Exit Rate of Pages", xlab = "Exit Rate as Percentage"))



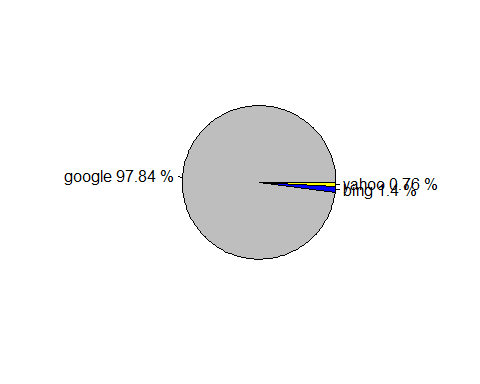
par(opar)

## Warning in par(opar): graphical parameter "cin" cannot be set

### Query 8: Search Engines

This query returns site usage data for all traffic by search engine, sorted by pageviews in descending order.

q8 = GA$getData(ids = 80895355,   
 start.date = "2014-01-12",   
 end.date = "2016-11-30",  
 dimensions="ga:source",  
 metrics="ga:pageviews,ga:sessionDuration,ga:exits",  
 sort="-ga:pageviews",  
 filters = "ga:medium==cpa,ga:medium==cpc,ga:medium==cpm,ga:medium==cpp,ga:medium==cpv,ga:medium==organic,ga:medium==ppc",  
 #the above filters are used to filter the pages which are search engines  
 #Source: Google's official documentation for tags  
 start = 1, max = 10000)  
View(q8)  
#Now display the 3 search engines as a pie-chart  
percentages1 = round(q8$pageviews/sum(q8$pageviews)\*100, 2)  
colors = c("grey", "blue", "yellow")  
pie(x = q8$pageviews, labels = paste(q8$source, percentages1, "%"), col = colors)



### Query 9: Keywords

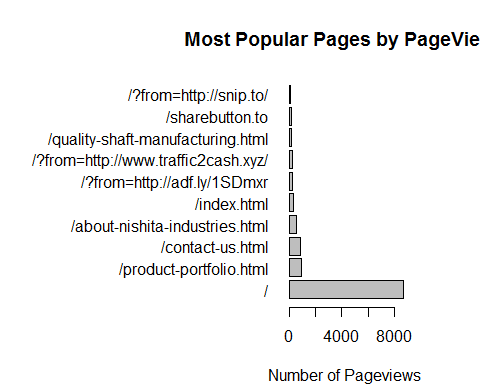
This query returns sessions broken down by search engine keywords used, sorted by sessions in descending order.

q9 = GA$getData(ids = 80895355,   
 start.date = "2014-01-12",   
 end.date = "2016-11-30",  
 dimensions="ga:keyword",#this tag gives the keywords which are used to lead to the site  
 #for example, "nishita industries goa" as a result for keyword would mean that   
 #this particular phrase was entered into the search engine to find nishitaindustries.com  
 metrics="ga:sessions",  
 sort="-ga:sessions",  
 filters = "",   
 start = 1, max = 10000)  
View(q9)  
#From the above table, we can see that most sessions have been triggered by spam websites  
#These websites aren't giving us a lot of information about relevant keywords that people google  
#to get to nishitaindustries.com. Hence we need to further filter these based on what we want.   
#However, that would be another topic, and hence we do not explore that here.

### Query 10: Top Content

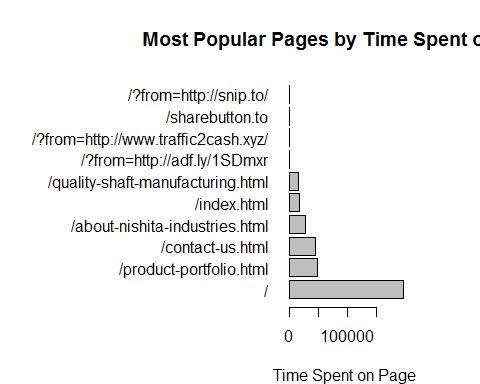
This query returns your most popular content, sorted by most pageviews.

q10 = GA$getData(ids = 80895355,   
 start.date = "2014-01-12",   
 end.date = "2016-11-30",  
 dimensions="ga:pagePath", #This gives us the   
 metrics="ga:pageviews,ga:uniquePageviews,ga:timeOnPage,ga:bounces,ga:entrances,ga:exits",  
 sort="-ga:pageviews",  
 filters = "",   
 start = 1, max = 10000)  
View(q10)  
#Let us try and analyze what are the most popular pages on nishitaindustries.com by number of pageviews  
q101 = q10[1:10, ] #Since the query extracts this sorted by pageviews, we just have to select the top 10 for our analysis  
View(q101)  
#Now let us create a bar chart to understand the magnitudes better.   
opar = par()  
par(mar=c(4,15,4,4))  
with(q101, barplot(height = q101$pageviews, names.arg = q101$pagePath, las = 1, horiz = T, main = "Most Popular Pages by PageViews", xlab = "Number of Pageviews"))



par(opar)

#We can see that many pageviews are coming from spam websites  
#However, we would be more interested in the pages on which the most time was spent.   
#For this we need to order the dataframe by time on page  
q102 = q10[order(-q10$timeOnPage), ]  
q102 = q102[1:10, ]#Extract the top 10 and view  
View(q102)  
  
opar = par()  
par(mar=c(4,15,4,4))  
with(q102, barplot(height = q102$timeOnPage, names.arg = q102$pagePath, las = 1, horiz = T, main = "Most Popular Pages by Time Spent on Page", xlab = "Time Spent on Page"))



par(opar)

#We see that the quality of shafts page features higher on this graph than the previous one

# References

Google. (2016). *Core Reporting API - Common Queries | Analytics Core Reporting API | Google Developers*. Retrieved from Google Analytics: https://developers.google.com/analytics/devguides/reporting/core/v3/common-queries

Pasin, M. (2015, January). *Google Analytics Dashboard with R and Shiny*. Retrieved from R-Bloggers: https://www.r-bloggers.com/google-analytics-dashboards-with-r-shiny/

Pasin, M. (2016, August). *Google Analytics Makes Demo Account Available to All*. Retrieved from R-Bloggers: https://www.r-bloggers.com/google-analytics-makes-demo-account-available-to-all/

Tatvic. (2014, August). *RGoogleAnalytics - Data Extraction Demo*. Retrieved from Github Repository: https://github.com/Tatvic/RGoogleAnalytics/blob/master/demo/data\_extraction\_demo.R

Weisberg, D. (2014, March). *Visualizing Google Analytics with R : Tutorial*. Retrieved from online-behavior.com: http://online-behavior.com/analytics/r