OSEMN Project - StackExchange Java Vs Javascript Questions

Rohan

5, December 2016

Contents

What is StackExchange:

Stack Exchange is a Q&A and discussion website for various topics including programming, science, technologies, Mathematics, Home Improvement, Statistics, and English Language and Usage. Founded in 2008 by Joel Spolsky and Jeff Atwood, the company was built on the premise that serving the developer community at large would lead to a better, smarter Internet. Since then, the Stack Exchange network has grown into a top-50 online destination, with Stack Overflow alone serving more than 40 million professional and novice programmers every month.

Like many other programming languages, Java and Javascript are two popular languages used for programming and scripting. In this project, we are going to examine the popularity for these two languages based on number of questions on Stack Exchange.

Let's get familiar with the languages first:

Java:

Originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform, Java is one of the most widely used programming languages. It is an independant, general-purpose computer programming language that is concurrent, class-based and object-oriented. With it's platform independant nature, programmers can write there code once and run anywhere they want without recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is mostly used for developing softwares for PC, mobiles or any other devices.

Javascript:

Although the name looks familiar with the earlier discussed language (Java), there are not many similarities between the Java and Javascript. Originally designed by Brendan Eich in 1995, JavaScript is one of the three core technologies of World Wide Web content production. Majority of websites are built using it, and almost all modern Web browsers support it without the need for plug-ins. It was influenced by programming languages such as *Self* and *Scheme*. Programmers also use JavaScript in video-game development, in crafting desktop and mobile applications, and in server-side network programming with run-time environments such as Node.js.

To make this project more user-friendly and reproducible, I would try to explain each and every step as we proceed.

[1]. Obtaining the Data:

For any project to proceed, we need to have a data to examine. StackExchange data could be obtained in multiple ways. For this project, I have obtained data by following steps:

[1.1] URL

Go to URL - https://data.stackexchange.com/stackoverflow/query/fork/582916

[1.2] Query

There is a readymade query on the page as given below:

```
SELECT
  DATEADD(mm, (Year(Posts.CreationDate) - 1900) * 12 + Month(Posts.CreationDate) - 1, 0) AS Month,
  Tags.TagName,
  COUNT(*) AS Questions

FROM Tags
  LEFT JOIN PostTags ON PostTags.TagId = Tags.Id
  LEFT JOIN Posts ON Posts.Id = PostTags.PostId
  LEFT JOIN PostTypes ON PostTypes.Id = Posts.PostTypeId

WHERE
  Tags.TagName IN (
  'java') AND

  PostTypes.Name = 'Question' AND
```

```
  -- Exclude the current month
  Posts.CreationDate < DATEADD(month, DATEDIFF(month, 0, GETDATE()), 0)

GROUP BY
  Year(Posts.CreationDate), Month(Posts.CreationDate), Tags.TagName

ORDER BY
Year(Posts.CreationDate), Month(Posts.CreationDate), Tags.TagName
```

[1.3] Scraping desired data

Just replace the Tag Name in *"Tags.TagName IN"* line, to Tag whose data is to be found. (In this case ***Java*** or ***Javascript***)

Once the query is ready, press "*Run Query*" button below the query.

After few verification steps, results will appear and "*Download CSV*" button will show up.

Download the CSV file and repeat the process for as many Tags you want. And upload all CSV files to Rstudio as below.

```
#Import the CSV files
#Since factors are not easy to deal with and can create problems while
#converting the data, use "stringsAsFactors = FALSE"
java = read.csv("OSEMN_Java_data.csv",stringsAsFactors = FALSE,header = TRUE)
javascript = read.csv("OSEMN_Javascript_data.csv",stringsAsFactors = FALSE,header = TRUE)
```

[2]. Scrubing the Data:

The data obtained from CSV is not always cleaned and in the format that we can directly use. Hence we need to scrub the data to make it usable.

[2.1] Cleaning and binding the files

Here, we have 2 different CSV files for different Tags. Hence, we'll have to combine them into one single file to use them. So we use "*rbind*" function for binding the data from both the files. *rbind* appends one dataframe below other, provided the column names are same in both files.

```
#Combining 2 files together
Final_data=rbind(java,javascript)

#Cheking the structure of final data
str(Final_data)
```

```
## 'data.frame':    198 obs. of  3 variables:
##  $ Month    : chr  "2008-08-01 00:00:00" "2008-09-01 00:00:00" "2008-10-01 00:00:00" "2008-11-01 00:
##  $ TagName  : chr  "java" "java" "java" "java" ...
##  $ Questions: int  223 1152 1156 966 842 1162 1219 1450 1545 1832 ...
```

```
#Ordering data in ascending order (optional)
Final_data=Final_data[order(Final_data$Month),]

#Removing the unwanted "hh:mm:ss" from the month data
```

```r
#gsub substitutes the "pattern" in data with the desired value
Final_data$Month <- gsub(x=Final_data$Month,pattern="00:00:00",replacement="")

#Converting the "Month" column in our data from character to actual Date format
#"as.Date" is used for data type conversion to date format
Final_data$Month<-as.Date(Final_data$Month, format='%Y-%m-%d')
```

[3] Exploring the Data

[3.1] Structure, Summary and Type of scrubed data

Once the data is ready to use, we can explore it for it's data types and structure. This ensures that every variable is in the format we want. For exploring our data, we'll use *str()*, *summary()* and *class()* functions.

The ***str()*** function returns the information about our dataframe which includes: ***Number of observations and variables*** , ***Name of every column***, ***Data type of each column in the dataframe***

The ***summary()*** function returns more statistical information about our dataframe including: ***mean***, ***median***, ***min*** and ***max*** of each column.

The ***Class()*** function returns the type of data file, eg: Dataframe, list, etc.

```r
str(Final_data)
```

```
## 'data.frame':    198 obs. of  3 variables:
##  $ Month    : Date, format: "2008-08-01" "2008-08-01" ...
##  $ TagName  : chr  "java" "javascript" "java" "javascript" ...
##  $ Questions: int  223 163 1152 641 1156 726 966 577 842 629 ...
```

```r
summary(Final_data)
```

```
##      Month              TagName            Questions
##  Min.   :2008-08-01   Length:198         Min.   :  163
##  1st Qu.:2010-08-08   Class :character   1st Qu.: 4492
##  Median :2012-09-01   Mode  :character   Median :12519
##  Mean   :2012-08-31                      Mean   :12106
##  3rd Qu.:2014-09-23                      3rd Qu.:18487
##  Max.   :2016-10-01                      Max.   :26895
```

```r
class(Final_data)
```

```
## [1] "data.frame"
```

[3.2] Peek into data

Since there are a so many rows in the data, it would be hard to show all the data. But, we can have a peek into the data using ***head()*** function, which returns first 6 rows of our dataframe. Here I have displayed data in tablular form using ***kable()***, which is in "***Knitr***" package.

```r
#Install "Knitr" and load the package
library(knitr)

#Display data in table format
kable(head(Final_data), caption = "JAVA Vs Javascript data from 2008 to 2016")
```

Table 1: JAVA Vs Javascript data from 2008 to 2016

	Month	TagName	Questions
1	2008-08-01	java	223
100	2008-08-01	javascript	163
2	2008-09-01	java	1152
101	2008-09-01	javascript	641
3	2008-10-01	java	1156
102	2008-10-01	javascript	726

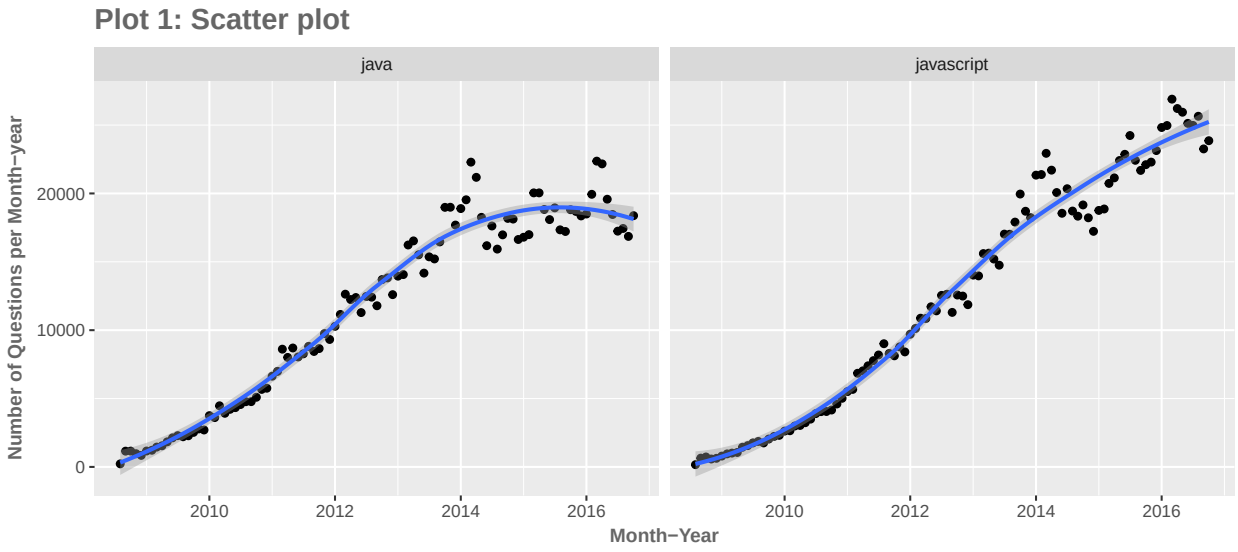
[4] Graphical representations of data

There are various ways for creating graphs in **R**, I'll use ***ggplot2*** in this project

```
#Install ggplot2 and load package
library(ggplot2)
```

[4.1] Scatterplot

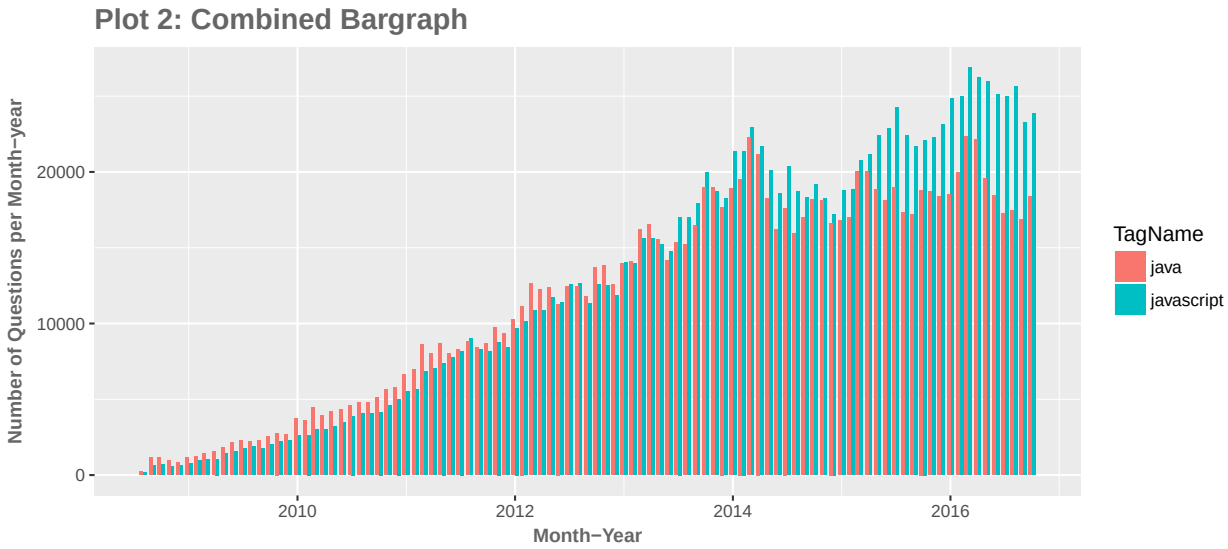
```
#Creating a variable "graph.theme" to store the color and size for
#title and axis, so it can be applied to all the graphs moving forward.
#(Optional)
graph.theme = theme(plot.title = element_text(color="#666666", face="bold", size=15))+
theme(axis.title = element_text(color="#666666", face="bold", size=10))

#Creating "scat" variable and storing "scatter plot" in it.
#Used facet_grid for seperating two graphs based on "TagName"
#geom_smooth for tracing line

scat = ggplot(data=Final_data, aes(x=Month,y = Questions)) +
geom_point()+
facet_grid(.~TagName)  +
geom_smooth()+

#Assign title for entire graph and the axes

ggtitle("Plot 1: Scatter plot") + xlab("Month-Year") +
ylab("Number of Questions per Month-year") + graph.theme

#View plot

scat
```



The above figure represents a scatter plot for the number of questions for Java and Javascript, posted on StackExchange, every month from 2008 to 2016. ***The interval on x-axis is 2 years, and Y-axis is 10,000 questions***. There are 2 distict graphs for Java and Javascript both showing the growth of each technology in terms of questions posted. We can see that Java was quite popular in start but after some time its growth has became steady and recently there is also a seen dip. On the other hand, Javascript has been popular consistently and still increasing.

[4.2] Combined Bargraph

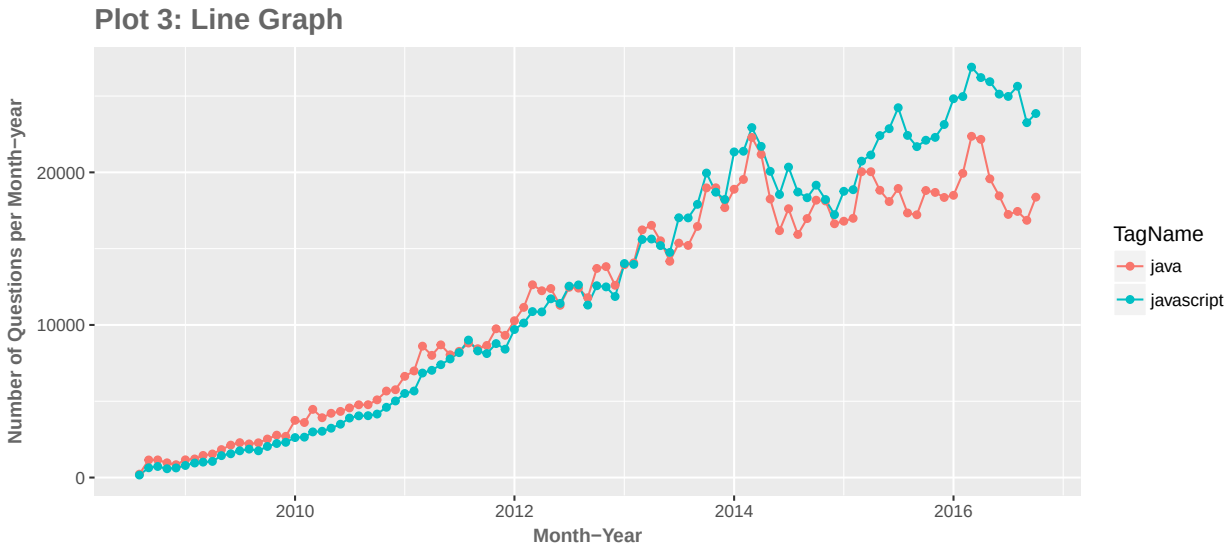
```
#Creating "bar" variable to store the graph
#Used stat="identity" as we are defining values for "Y axis" manually
#Used "position = position_dodge()" as we are displaying bars side-by-side

bar = ggplot(data=Final_data, aes(x=Month, y=Questions, fill=TagName)) +
  geom_bar(stat="identity", position=position_dodge()) +

#Assign title for entire graph and the axes

ggtitle("Plot 2: Combined Bargraph") + xlab("Month-Year") +
ylab("Number of Questions per Month-year")+ graph.theme

#View plot

bar
```



The above figure represents a Bargraph for the number of questions for Java and Javascript posted on StackExchange, every month from 2008 to 2016. ***The interval on x-axis is 2 years, and Y-axis is 10,000 questions***. The graph shows that both the technologies had almost similar number of questions till **2010**, with Java having a little edge ahead. But after **2010**, it overtook Javascript in terms of questions popularity till first few months of **2013**. But since then, Javascript has been more popular with more questions, even crossing the 20,000 mark in most of the months.

[4.3] Line Graph

```
#Creating a variable "line" to store the graph.
#Used "geom_point()" to show p  oints along the line.

line = ggplot(data=Final_data, aes(x=Month, y=Questions, colour=TagName)) +
geom_line() +
geom_point()+

#Assign title for entire graph and the axes

ggtitle("Plot 3: Line Graph") + xlab("Month-Year") +
ylab("Number of Questions per Month-year")+ graph.theme

#View plot

line
```



The above figure represents a line graph for the number of questions for Java and Javascript posted on StackExchange, every month from 2008 to 2016. ***The interval on x-axis is 2 years, and Y-axis is 10,000 questions***. Just as described in Bargraph, this graph also shows that Java has been popular in terms of questions till ***2013***. But since then, Javascript has been leading the charts with questions crossing the 20,000 mark.

[5] T-test

We can also do a Two sample hypothesis test (T-test), to verify our graphical interpretation. This test is usually done to compare whether the average difference between two groups is really significant or if it is due instead to random chance. For implementing this t-test, we compare the number of questions for our Java and Javascript dataframes.

```
#T-test comparing questions for Java and Javascript.
#"paired = False" as both are independent samples.

t.test_p=t.test(javascript$Questions,java$Questions, paired=FALSE)

#View results

t.test_p
```

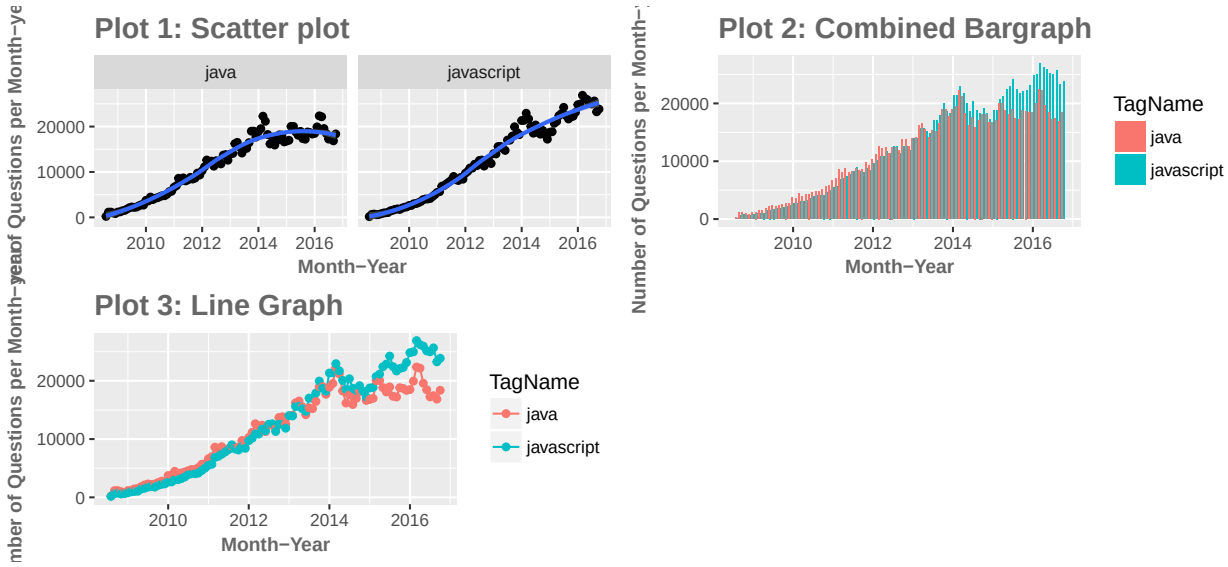
```
##
##  Welch Two Sample t-test
##
## data:  javascript$Questions and java$Questions
## t = 0.82191, df = 186.67, p-value = 0.4122
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1250.009  3035.464
## sample estimates:
## mean of x mean of y
##  12552.62  11659.89
```

The results of t-test show that the **p-value is 0.41, which is greater than 0.05** indicating that there is no significant difference between the two groups ***Java*** and ***Javascript***.

[6] Summary and Conclusion

To summarize all our observations, we can have a final look at graphs and the t-test conducted.

```
#Install gridextra and load the package
library(gridExtra)
grid.arrange(scat, bar, line, ncol=2, nrow =2)
```




```
##
##  Welch Two Sample t-test
##
## data:  javascript$Questions and java$Questions
## t = 0.82191, df = 186.67, p-value = 0.4122
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1250.009  3035.464
## sample estimates:
## mean of x mean of y
##  12552.62  11659.89
```

As per the graphs and the data collected, it can be concluded that popularity of ***Javascript*** and ***Java*** is evenly distributed in terms of the number of questions posted on ***StackExchange***. This statement is also supported by our ***t-test*** showing no significant difference since the p-value obtained is > 0.05. However, it cannot be overlooked that ***Javascript*** has overtook ***Java*** in recent times, with a remarkable difference. This might be the case due to increasing need for web applications and web content for which Javascript is more suited.