Comparing number of JavaScript and Python Repositories in Github

Nandita Ramakrishnan

April 21, 2016

Introduction

GitHub is a open source distributed revision control and source code management system. Git is a command-line tool,whereas GitHub is a Web-based application. Github allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute. It also provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project. GitHub reports having around 14 million users and more than 35 million repositories,making it the largest host of source code in the world. The repositories have codes written in different programming language. In this article we are comparing the number of JavaScript and Python repositories in Github.

Obtaining the data

The first step in analysis is to obtain the required data.The information on Github repositories can be accessed using the Github API. GitHub API is a publicly available API to interact with its huge dataset of events and interaction with hosted repositories. The API provides information on total count of repositories, repository name, size,its owner details, programming language, fork count, issues count and various other details.

The code below shows the URL for accessing information on repositories using Github API.To get information on JavaScript and Python repositories we need to pass parameters in the URL, one is programming language name and another is number of records per page. The per_page variable is set to 100, thereby limiting the number of objects retrieved per page.

```
#Github API url
githuburl <- "https://api.github.com/search/repositories"
```

```r
# Obtain data about repository from Github API
# by passing language name and per page
# object count as parameters

javascriptURL <-paste0(githuburl,"?q=language:javascript&per_page=100")

pythonURL <-paste0(githuburl,"?q=language:python&per_page=100")
```

JavaScript Repository

The Github API provides data about repositories in JSON (JavaScript Object Notation) format. We are using the fromJSON function provided by jsonlite package to convert the JSON data obtained from the URL to a list. The list is then converted to dataframe.

```r
library(jsonlite)

# Read the json data from the URL and convert to list
javascriptList <- jsonlite::fromJSON(javascriptURL)

#Converting list to dataframe
javascriptDf <- data.frame(javascriptList)
```

Python Repository

```r
# Read the json data from the URL and convert to list
pythonList <- jsonlite::fromJSON(pythonURL)

#Convert list to dataframe
pythonDf <- data.frame(pythonList)
```

Scrubing the data

The data obtained from Github API needs to be cleaned for further analysis. We need to look for any inaccurate or corrupt data and handle them. We will also be focusing on retrieving data that is required for our analysis and store them in a new dataframe.

The javascriptDf and pythonDf dataframe have a column named total_count, which indicates the total count of JavaScript and Python repositories in Github. We will be extracting that information and storing it in a vector. Then a

dataframe named repoDataframe is created with two columns, language and repository count.

```
#Get total repository count from JavaScript data frame
javascriptRepoCount <- javascriptDf[1,"total_count"]


#Get total repository count from python data frame
pythonRepoCount <- pythonDf[1,"total_count"]

#Create a vector of language
langauge <- c("JavaScript", "Python")


#Create a vector of repo count
repoCount <- c(javascriptRepoCount,pythonRepoCount)


#Dataframe is created with 2 columns -language
#and repository count

repoDataframe <- data.frame(langauge, repoCount)
```

Exploring the data

After data cleaning, we need to explore the dataframe to understand its structure and gain basic knowledge about the data that it stores.

Following are few basic functions that helps in exploring the dataset

- str() - displays the internal structure of the object
- summary() - provides statistical info of the object data
- class() - displays the data type of the object
- head() - displays first 6 rows of dataset
- tail() - displays the last 6 rows of dataset

```
str(repoDataframe)

## 'data.frame': 2 obs. of  2 variables:
##  $ langauge : Factor w/ 2 levels "JavaScript","Python": 1 2
##  $ repoCount: int  2161699 1045309
```

```
summary(repoDataframe)
```

```
##        langauge   repoCount
##  JavaScript:1   Min.   :1045309
##  Python    :1   1st Qu.:1324406
##                 Median :1603504
##                 Mean   :1603504
##                 3rd Qu.:1882602
##                 Max.   :2161699
```

```
class(repoDataframe)
```

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

We will be exploring the repoDataframe columns by identifying its data type and the value that its holds.

```
#Display datatype of language column
class(repoDataframe$langauge)
```

```
## [1] "factor"
```

```
#Display datatype of repoCount column
class(repoDataframe$repoCount)
```

```
## [1] "integer"
```

```
#Display summary of the language column
summary(repoDataframe$langauge)
```

```
## JavaScript     Python
##          1          1
```

```
# Display summary of the repoCount column
summary(repoDataframe$repoCount)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 1045000 1324000 1604000 1604000 1883000 2162000
```

For integer column repoCount, summary function provides more useful information like the minimum, maximum, average, median, 1st and 2nd quartile values.

```
# Display summary of the repoCount column
summary(repoDataframe$repoCount)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 1045000 1324000 1604000 1604000 1883000 2162000
```

Result

Displaying the complete dataset in tabular form

```
##Displays the entire dataset

repoDataframe

##     langauge repoCount
## 1 JavaScript   2161699
## 2     Python   1045309
```

The dataset has two rows and two columns. The column names are language and repoCount. The data provides information on the number of JavaScript and Python repositories in Github.

Graphs

Using the dataset we have lotted three graphs - barplot, line graph and pie chart. ggplot is a very usefule package that allows us to create various different graphs.

Barplot

```
library(ggplot2)

#Use ggplot to create a bar graph of
#language vs number of repositories


ggplot(data=repoDataframe, aes(x=langauge, y=repoCount)) +
  geom_bar(fill="red", width=.4, stat="identity") +
  xlab("Programming Language") +
  ylab("Count") +
  theme(axis.text = element_text(size = 12),
    plot.title = element_text(color="black", face="bold",size=14),
    axis.title= element_text(margin=margin(10,20,10,10)),
    plot.title = element_text(hjust = 0.5),
    axis.title.x = element_text(colour = "black",face="bold"),
    axis.title.y = element_text(colour = "black",face="bold"))
```

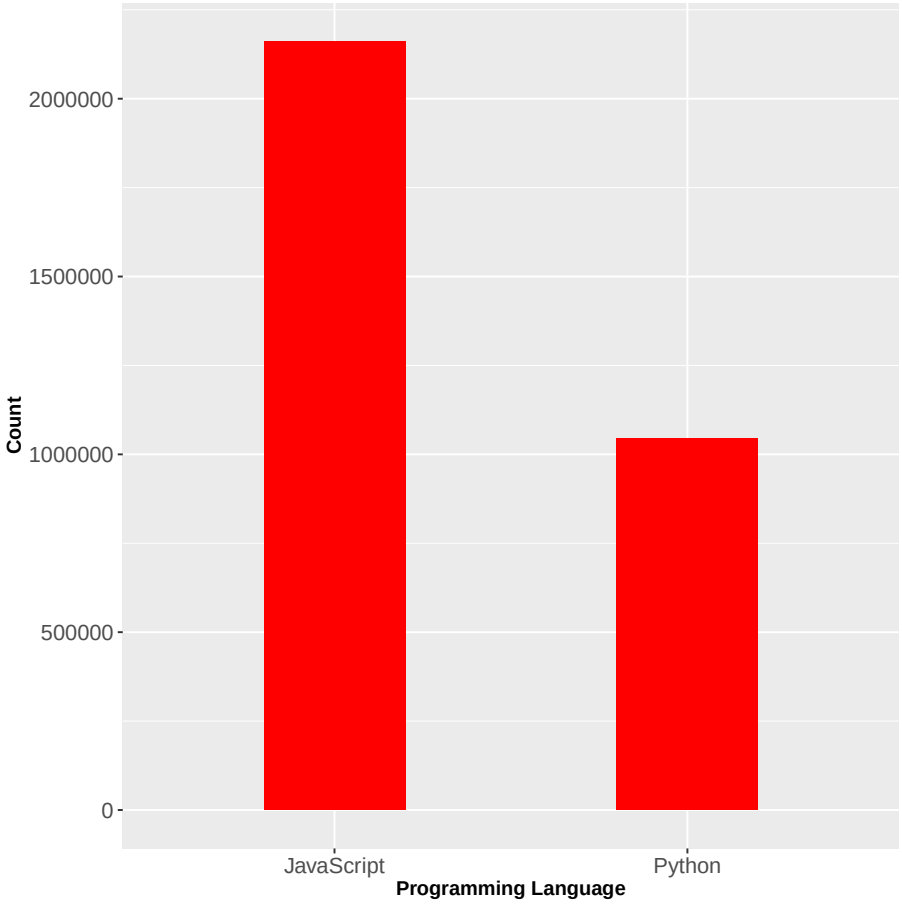

Figure1. shows the plot of programming language vs number of repositories in Github.

The graph shows that the total number of JavaScript respositories in Github is higher than the Python repository.Number of JavaScript repositories is almost double the number of Python repositories. We can conclude that JavaScript is more popular language compared to Python among the Github developers.

Line plot

```
#Plot a line graph
ggplot(repoDataframe, aes(x=langauge, y=repoCount, group=1)) +
     geom_line(colour="blue", linetype="dashed", size=1.5) +
     geom_point(colour="blue", shape=21, size=4,fill="white")+
     xlab("Programming Language") + ylab("Count")
```

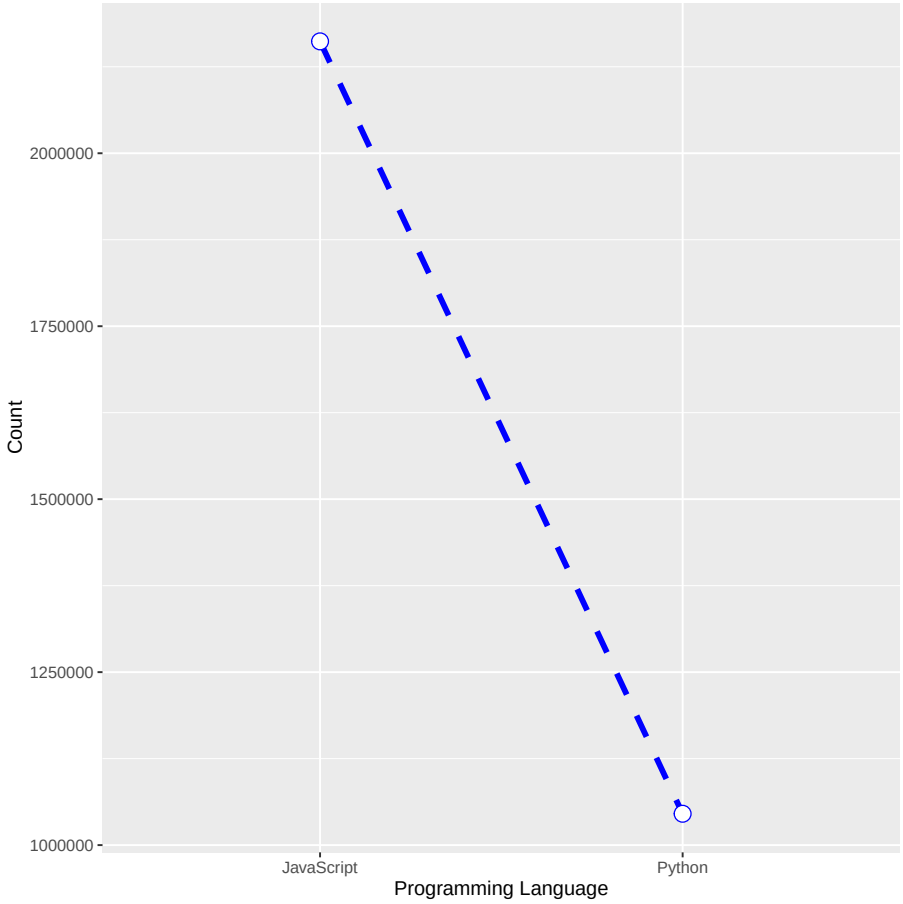

Figure2. shows the line graph of programming language vs count of repositories in Github.

It is clearly visible from both the graphs that Github has more developers contributing or developing JavaScript related project repositories than Python. Analyzing the GitHub data makes it possible to understand the popularity of programming language and can also be used to predict the trend of language among the developers.