R Repositories on GitHub in January-2015 and February-2015

Juhi Garg
School of Information Technology
Illinois State University

April 20, 2015

Introduction

The purpose of this paper is to analyze the number of R repositories created on Github in the month of January-2015 and February-2015. Gihub is an open source code management system started in 2008 by Linus Torvalds. To understand GitHub, we must first have an understanding of Git. Git is an open-source version control system. That means that When developers are creating something (an application, for example), they are making constant changes to the code and releasing new versions, up to and after the first official (non-beta) release.Git keep these revisions straight, and store the modifications in a central repository. This 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.Git is a command-line tool, but the center around which all things involving Git revolve effectively, the Hub, is GitHub.com, where developers can store their projects and network with likeminded people. GitHub itself isnt much more than a social network like Facebook or Flickr. You build a profile, upload projects to share and connect with other users by following their accounts. And while many users store programs and code projects, theres nothing preventing you from keeping text documents or other file types in your project folders to show off. GitHut is an attempt to visualize and explore the complexity of the universe of programming languages used across the repositories hosted on GitHub.R is one such language.RStudio is an excellent integrated development environment built specifically for R. It contains version control for Git.Here we will analyze the repositories craeted in R language in the month of January-2015 and February-2015. Our OSEMN assignment will involve Obtaining, Scrubbing, Exploring, More exploring and graphiNg data from a number of different on-line resources.

Obtaining the Data

The first step towards OSEMN is to obtain the data.The data source for this report is GitHub API. GitHub provides publicly available API to interact with its huge dataset of events and interaction with the hosted repositories.All data is sent and received as JSON. As an unauthorized user GitHub's API sets the default page size to 30 records that means you can get only 30 records for a particular query. This can be increased to a maximum of 100 by giving the header perpage="100" in the URL but not more than that. For this report, I need only the count of records and not the actual record, that is the number of repositories created in a particular month and not any specific detail about each repository. So, using "api.github.com", we can access GitHub API, then for searching the repositories we have to provide the path as "search repositories" then quering for dates of creation so as we are looking for januay and february I have provided "2015-01-01 to 2015-01-31" and "2015-02-01 to 2015-02-28" respectively, and as we are searching particulary for repositories created in R so we need to pass query for language R. And to increase the per page results, I specified "perpage=100". The full data set can be accessed by having proper authorization. So the URL for searching the repositories created in R for January is as below:

January Data

```r
# Getting the january data from Github API queried
# using two parameters:
# (1)-"created" between 2015-01-01 and 2015-01-31
# (2)-"language"=R.
# The data is in the json format

JanuaryURL<-"https://api.github.com/search/repositories?q=created%3A%222015-01-01+..+2015-01
```

As the json file here is a nested data consting of two tables "items" and "owners",therefore we are using the "jsonlite" package to read data. If the data set consists of only one table, then we would have used "RJSONIO" package. The function "fromJSON" will read json data and convert it to list. The list is then converted to a dataframe using "data.frame()" function.

```r
library(jsonlite)

##
## Attaching package:  'jsonlite'
##
## The following object is masked from 'package:utils':
##
##     View
```

```
# Reading the json file and converting it to a list using the jsonlite package.
JanuaryList = jsonlite::fromJSON(JanuaryURL)
#Converting list to dataframe
JanuaryDf<-data.frame(JanuaryList)
```

The data has 86 columns,but we have taken out only the relevant columns.

```
#Taking out the relavant columns
ModifiedJanuaryDf<-JanuaryDf[ , c(1,3,4,7,64)]
```

First 6 rows of the data is shown using head() command.

```
#Taking out the 6 rows
head(ModifiedJanuaryDf)

##   total_count items.id            items.name items.private
## 1        6153 28691460                 editR         FALSE
## 2        6153 29205154               dagdata         FALSE
## 3        6153 29544018               assertr         FALSE
## 4        6153 29139502              chartist         FALSE
## 5        6153 28728332             Rlinkedin         FALSE
## 6        6153 30126776 social-media-workshop         FALSE
##   items.mirror_url
## 1               NA
## 2               NA
## 3               NA
## 4               NA
## 5               NA
## 6               NA
```

In the same way, the data for february is collected.

February Data

```
# Getting the february data from Github API queried using two parameters:
# (1)-"created" between 2015-02-01 to 2015-02-28 (2)-"language"=R.
#The data is in
# the json format
FebruaryURL<-"https://api.github.com/search/repositories?q=created%3A%222015-02-01+..+2015-0

# Reading the json file and converting it to a list using the jsonlite package.
FebruaryList = jsonlite::fromJSON(FebruaryURL)

#Converting list to dataframe
```

```
FebruaryDf<-data.frame(FebruaryList)

#Taking out the relavant columns
ModifiedFebruaryDf<-FebruaryDf[ , c(1,3,4,7,64)]

#Taking out the 6 rows
head(ModifiedFebruaryDf)

##   total_count items.id      items.name items.private items.mirror_url
## 1        5694 31300539       shinystan         FALSE               NA
## 2        5694 31203588             rio         FALSE               NA
## 3        5694 30628339 internetarchive         FALSE               NA
## 4        5694 30158790            drat         FALSE               NA
## 5        5694 30308379          stackr         FALSE               NA
## 6        5694 30709113 ggplot-tutorial         FALSE               NA
```

Scrubbing the Data

Scrubbing the data includes cleaning the data and making it useful for further analysis. As for this report we need to find the total number of repositories created in a particular month, it is given by the totalcount field of our data, therefore taking out the totalcount field from both the data set. For January, it is stored in ”JanuaryRepositoriesCount” variable For February, it is stored in ”FebruaryRepositoriesCount” variable.

```
# The variable total_count in the data will give the number of repositories.
JanuaryRepositoriesCount <-ModifiedJanuaryDf[1, ]$total_count
# The variable total_count in the data will give the number of repositories.
FebruaryRepositoriesCount <-ModifiedFebruaryDf[1, ]$total_count
```

Now, as we need to show the count of repositories against each month, so making two vectors, ”Month” vector will store the name of two months and ”NoOfRepositorie s” will show the count of repositories.

```
# Creating a vector "Month" which will store the
# name of months.
Month <-c("January", "February")

# Creating a vector "NoOfRepositories" which will store the
#count of January and February Repositories.
NoOfRepositories<-c(JanuaryRepositoriesCount, FebruaryRepositoriesCount)
```

The two vectors are then bind to form a dataframe ”df”.

```
# The two vectors are then bind into a dataframe.
df = data.frame(Month, NoOfRepositories)

# Display the dataframe.
df

##      Month NoOfRepositories
## 1  January             6153
## 2 February             5694
```

Exploring the Data

The data is explored using the following commands: (a). class() (b). str() (c). summary()

Lets explore our data frame with the class() command.It is used to detrmine the class associated with the object in R. The class of our final data is ”dataframe”. The class() funtion can be appplied independetly to each variable of our data set. The class of vector Month is ”factor”. The class of vector NoOfRepositories is ”integer”.

```
class(df)

## [1] "data.frame"

class(df$Month)

## [1] "factor"

class(df$NoOfRepositories)

## [1] "integer"
```

Lets explore our data frame with the str() command.It compactly display the internal structure of an R object. The structure of our data frame shows that it has 2 observations(rows) of 2 variables, then it displays the value of each variable with their class types. Similar to the class() command, the str() command can also be applied to each variable of the data set. The structure of variable ”month” says that it if of factor type with 2 columns and 1 row and also displays its values. The structure of variable ”NoOfRepositories” says that it is of integer type with 2 columns and 1 row and also diplays its values.

```
str(df)

## ’data.frame’: 2 obs. of  2 variables:
##  $ Month           : Factor w/ 2 levels "February","January": 2 1
##  $ NoOfRepositories: int  6153 5694
```

```
str(df$Month)

##  Factor w/ 2 levels "February","January": 2 1

str(df$NoOfRepositories)

##  int [1:2] 6153 5694
```

Finally, we ll explore our data using the summary() command. This function will give details like the minimum and the maximum allowable values, mean, median and quartiles for each variable in the data frame.

```
summary(df)

##       Month   NoOfRepositories
##  February:1   Min.   :5694
##  January :1   1st Qu.:5809
##               Median :5924
##               Mean   :5924
##               3rd Qu.:6038
##               Max.   :6153

summary(df$Month)

## February  January
##        1        1

summary(df$NoOfRepositories)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5694    5809    5924    5924    6038    6153
```

Results

```
df

##      Month NoOfRepositories
## 1  January             6153
## 2 February             5694
```

Table 1: Showing the number of repositories created in January and February-2015

To futher explain the results grphically, we have used ggplot. Two different packages were used to create these graphs. (a): ggplot2: ggplot2 allows you to create graphs, data can be represented by different color, symbol, size. (b): plotrix: Plotrix is basically used to plot the 3D pie charts.

```
library(plotrix)
library (ggplot2)
```

The results are plotted in 3 different ways:

```
# Using ggplot to plot the dataframe with bars.
ggplot(data=df, aes(x=Month, y=NoOfRepositories, fill=Month))+
geom_bar(stat="identity")+
scale_fill_manual(values=c("blue","yellow"))+
ggtitle("Number  of R Repositories created on Github in 2015")
```

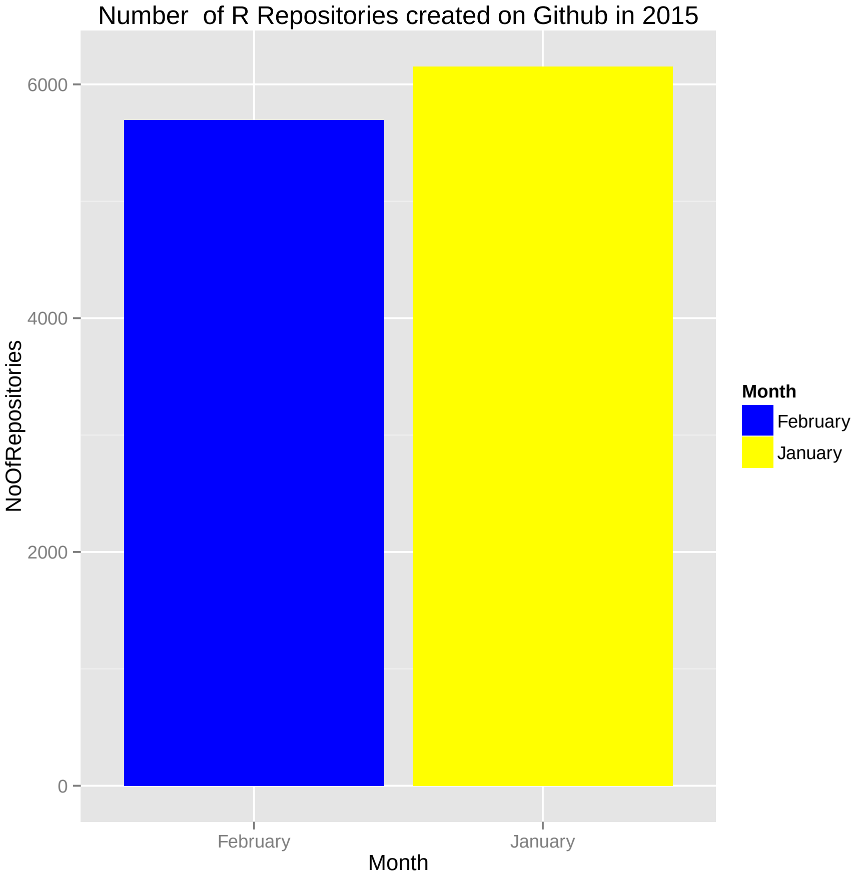

Figure1: BarPlot showing count of repositories of each month.

```
# Using ggplot to plot the dataframe with line and points.
ggplot(data=df, aes(x=Month, y=NoOfRepositories,group=1)) +
geom_line(colour="blue", linetype="dashed",size=1.5) +
geom_point(colour="red", size=4, shape=21, fill="yellow")
```

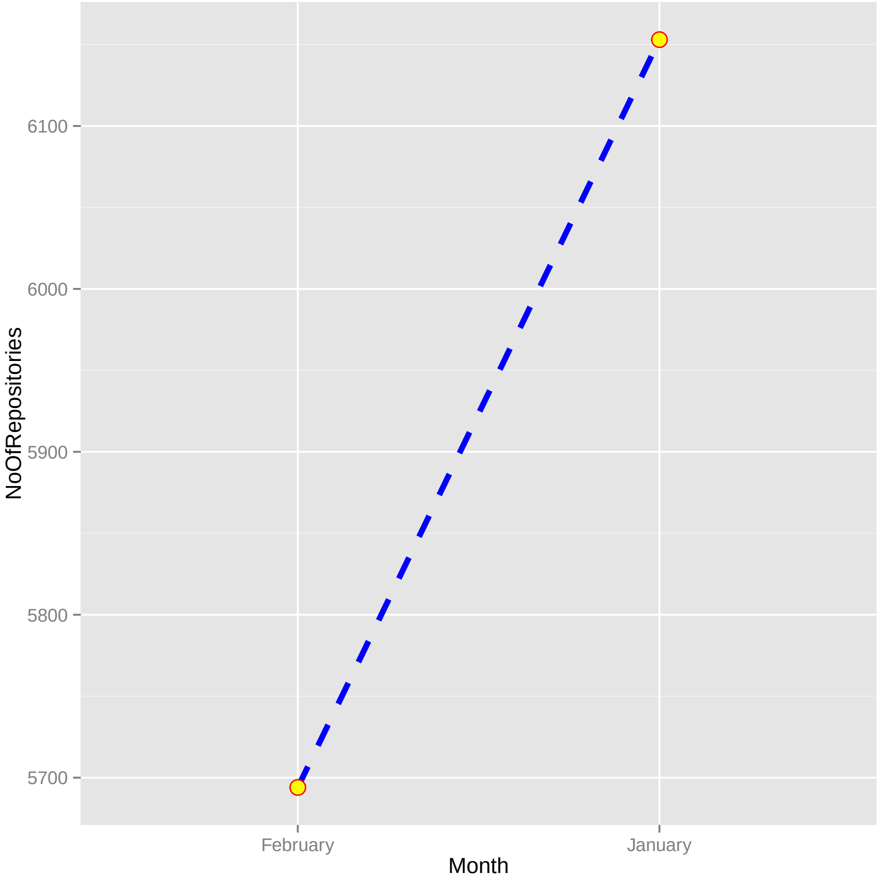

Figure2: Line plot showing count of repositories of each month.

```
# 3D Pie Chart of Plotrix package
pie3D(NoOfRepositories,labels=Month,explode=0.1,
main="Pie Chart showing R repositories created on Github in 2015 ")
```

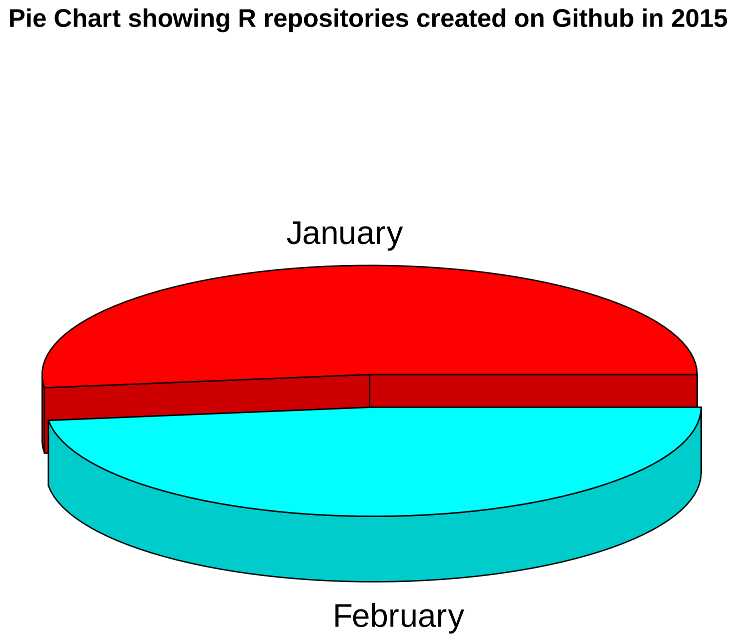

Figure3: 3D pie chart showing count of repositories of each month.