DATA ANALYSIS

COLLECT, STORE, RETRIEVE DATA

(DA5020)

TERM PROJECT REPORT

Professor- Kathleen Durant

Report by:

Dipika Mishra

Gandhar Kothari

Introduction:

Founded in 2008, Stack Overflow is the largest, most trusted online community for developers to learn, share their knowledge, and build their careers.

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's built and run *by people* as part of the Stack Exchange network of Q&A sites.

More than 50 million professional and aspiring programmers USE Stack Overflow each month to help solve coding problems, develop new skills, and find job opportunities.

Stack Overflow partners with businesses to help them understand, hire, engage, and enable the world's developers.

Stack Overflow Statistics:

In 2017 over 50,000 developers shared their work, what they build and who they are.

Every month, 40 million people visit Stack Overflow. In January, those visitors submitted 2.2 million feedback events (1.7 million votes plus 540 thousand anonymous votes).

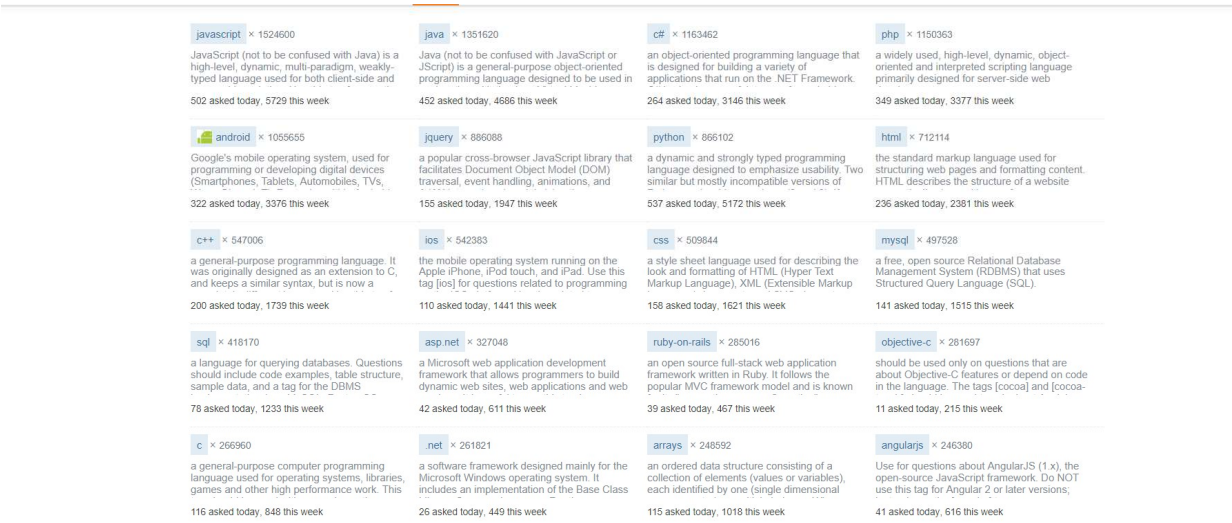
Every 8 secs or so a user asks a question or so.

There are more than 3500 active tags which people are talking about.

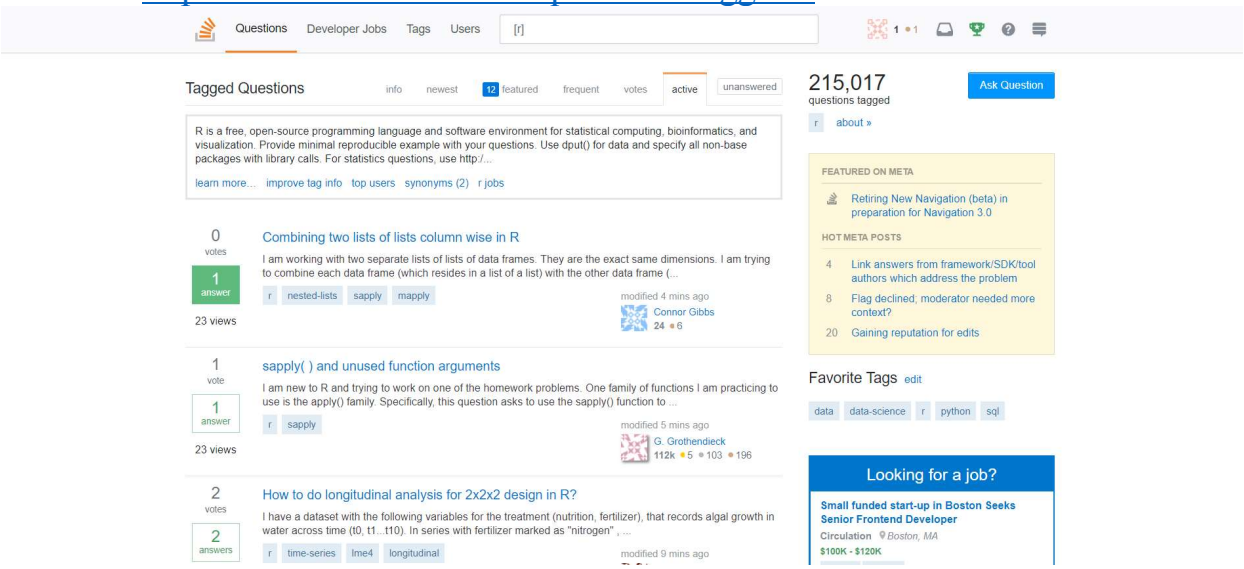
The clear majority of developers use Stack Overflow to get help for their job. Most also use Stack Overflow because they love to learn.

Tour to Stack Overflow website:

1. Clicking on Tags Tab gives us the numerous tags in the Stack Overflow https://stackoverflow.com/tags

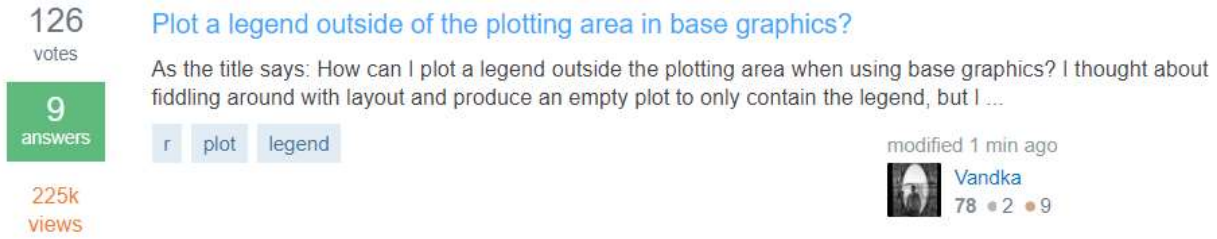


2. Clicking on the R tag the page that open is shown below. And the Url for this page is - https://stackoverflow.com/questions/tagged/r



On the right-hand side of the page there are related tags-

Every question will have a number of answers, votes, views count, users information and tagged associated with that question as seen below.



Collection: -

Our aim is to collect data for R and python tags.

A custom function was written using “RCurl” package and “Rvest” Package to scrape the data from the url.

The data for both python and R tags was stored in separate dataframe.

Cleansing: -

The data was cleaned as per the analysis required. The packages used were “stringr”, stringi” and “tmap”.

Analysis and Visualization :

The analysis was conducted to answer the questions which can be seen in the rmd attached below.

The data was visualized using graphs from the “ggplot” pacakge and “wordcloud” package for answering different questions.

Storage:-

The data collected from r and python tags was stored in SQL server using RODBC package.

R Data

SQLQuery5.sql - L...VK9TRM\mishr (57)) | SQLQuery4.sql - L...VK9TRM\mishr (56))

```sql
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [X]
      ,[id]
      ,[date]
      ,[title]
      ,[tags]
      ,[url]
      ,[views]
      ,[votes]
      ,[answers]
      ,[user]
      ,[reputation]
  FROM [master].[dbo].[r_data1]
```

121 %

Results | Messages

	X	id	date	title	tags	url
1	1	47732389	2017-12-09 19:36:15	r - subsetting dataframe creates factors	dataframe, subset	http://stackoverflow.comhttps://s
2	2	47732215	2017-12-09 19:18:39	graphical representation in Rstudio	plot	http://stackoverflow.comhttps://s
3	3	47732085	2017-12-09 19:03:26	Sum of some positions in a row - R	matrix, sum	http://stackoverflow.comhttps://s
4	4	47732049	2017-12-09 18:58:45	Change home directory for R	path	http://stackoverflow.comhttps://s
5	5	47732024	2017-12-09 18:55:36	Add new column, multiply value with strings with dplyr	dataframe, dplyr	http://stackoverflow.comhttps://s
6	6	47731923	2017-12-09 18:44:07	How to change range on ACF plot	advanced-custom-fields	http://stackoverflow.comhttps://s
7	7	47731906	2017-12-09 18:43:04	googleway package produces blank plot	geospatial, googleway	http://stackoverflow.comhttps://s
8	8	47731717	2017-12-09 18:20:19	Frequency data frame for grouped bar plot	r	http://stackoverflow.comhttps://s
9	9	47731710	2017-12-09 18:19:42	Matrix with integer random rows	matrix, random, integer	http://stackoverflow.comhttps://s
10	10	47731656	2017-12-09 18:13:46	knitr, RwordPress quotations and ampersand changes	wordpress, knit rcurl	http://stackoverflow.comhttps://s
11	11	47731582	2017-12-09 18:06:01	Scrape loop multiple pages R	web-scraping, rvest	http://stackoverflow.comhttps://s
12	12	47731557	2017-12-09 18:03:26	Create tibble with all possible models	r	http://stackoverflow.comhttps://s
13	13	47731531	2017-12-09 18:01:12	Find Best Sarima Model in R	rstudio	http://stackoverflow.comhttps://s

Query executed successfully. | LAPTOP-1MVK9TRM (13.0 SP1) | LAPTOP-1MVK9TRM\mishr ... | master | 00:00:00 | 1000 rows

Python Data –

SQLQuery4.sql - L...VK9TRM\mishr (56))

```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [X]
      ,[id]
      ,[date]
      ,[title]
      ,[tags]
      ,[url]
      ,[views]
      ,[votes]
      ,[answers]
      ,[user]
      ,[reputation]
  FROM [master].[dbo].[python_data1]
```

121 %

Results | Messages

	X	id	date	title	tags	url
1	1	47732878	2017-12-09 20:29:35Z	Creating multiple dictionaries from CVS - Python 2.7	python; python-2ÃƒÂ»7; csv; dictionary	http:
2	2	47732835	2017-12-09 20:24:41Z	Finding an element with class name in beautifulsoup	python; beautifulsoup	http:
3	3	47732832	2017-12-09 20:23:45Z	How to store a pytransitions machine in a YAML c...	python; transitions	http:
4	4	47732816	2017-12-09 20:22:01Z	none returned when trying to get tag value	python; beautifulsoup	http:
5	5	47732814	2017-12-09 20:21:50Z	Read newspaper articles to extract context	python; machine-learning; nlp; topic-modeling	http:
6	6	47732772	2017-12-09 20:17:16Z	How to set max_proc_per_cpu in Scrapyd	python; scrapy; scrapyd	http:
7	7	47732734	2017-12-09 20:13:28Z	Tkinter: clicking buttons in menu is triggering functi...	python; python-3ÃƒÂ»x; user-interface; tkinter	http:
8	8	47732711	2017-12-09 20:10:55Z	Find all User's UserID in a server	python; discord	http:
9	9	47732657	2017-12-09 20:04:38Z	wave.error: # channels not specified	python; wave	http:
10	10	47732639	2017-12-09 20:02:29Z	How to run a pytthon module written from some old...	python; dependencies; jupyter-notebook; dependen...	http:
11	11	47732621	2017-12-09 20:00:51Z	Click link only when another link is available on pa...	python; selenium	http:
12	12	47732604	2017-12-09 19:59:22Z	Python game keys are not responding	python; macos; pygame; key	http:
13	13	47732592	2017-12-09 19:58:24Z	Managed loop in Python?	python; python-2ÃƒÂ»7	http:

Query executed successfully. | LAPTOP-1MVK9TRM (13.0 SP1) | LAPTOP-1MVK9TRM\mishr ... | master | 00:00:00 | 1000 rows

References-:

- R for Data Science
- https://stackoverflow.com/

CSR Project Report

Gandhar Kothari | Dipika Mishra

2017-12-10

```r
#Installing and loading required packages
#install.packages("tidytext")
#install.packages("stringr")
#install.packages("dplyr")
#install.packages("ggplot2")
#install.packages("lubridate")
#install.packages("XML")
#install.packages("RCurl")
#install.packages("wordcloud")
#install.packages("rvest")
library("RCurl")
```

```
## Loading required package: bitops
```

```r
library("XML")
library("lubridate")
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```r
library("ggplot2")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library("stringr")
```

```
## Warning: package 'stringr' was built under R version 3.4.2
```

```r
library("tidytext")
```

```
## Warning: package 'tidytext' was built under R version 3.4.3
```

```r
library("tidyverse")
```

```
## Warning: package 'tidyverse' was built under R version 3.4.2
```

```
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
```

```
## Warning: package 'readr' was built under R version 3.4.2
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## as.difftime(): lubridate, base
## complete():    tidyr, RCurl
## date():        lubridate, base
## filter():      dplyr, stats
## intersect():   lubridate, base
## lag():         dplyr, stats
## setdiff():     lubridate, base
## union():       lubridate, base
```

```
library("wordcloud")
```

```
## Warning: package 'wordcloud' was built under R version 3.4.2
```

```
## Loading required package: RColorBrewer
```

```
library("rvest")
```

```
## Warning: package 'rvest' was built under R version 3.4.2
```

```
## Loading required package: xml2
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:purrr':
##
##     pluck
```

```
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
## The following object is masked from 'package:XML':
##
##     xml
```

```
library("RODBC")
```

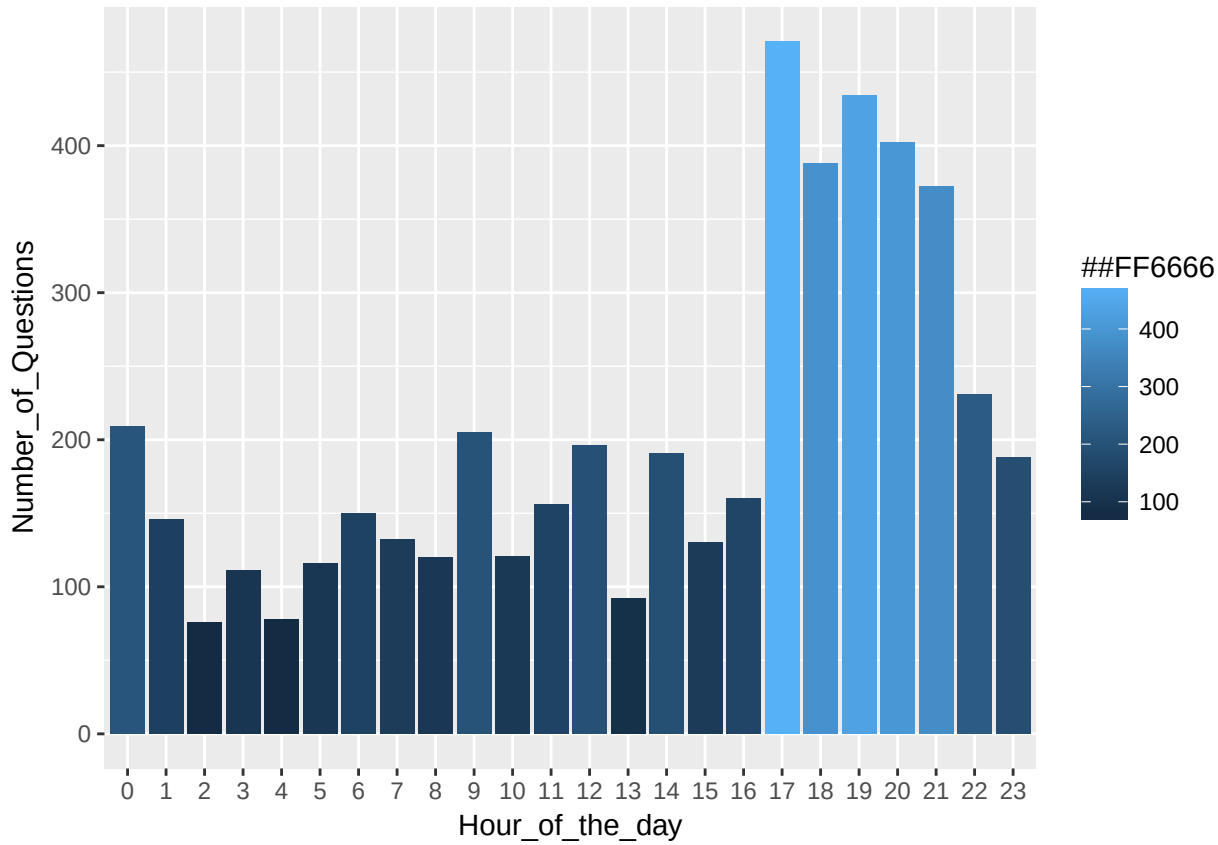
```
# Creating Function to scrape data from Stack overflow using RCurl
# stack_data = function(url, num_pages)
# {
#   scrape_data = NULL
#   #empty data frame for overall data set
#   for(i in 1:num_pages)
#   {
#     page = getURLContent(url)
#     doc = htmlParse(page, asText = TRUE)
#     # +++++ Get the posts on current page +++++
#     postspath = "//div[@class = 'question-summary']"
#     posts = getNodeSet(doc, postspath)
#     i = length(posts)
#     d = NULL
```

```
#     #empty data frame for values of a single page
#     # +++++ Process posts on current page +++++
#     i = posts
#     for(i in 1:15) {
#       p = posts[[i]]
#       # ===== ID Number =====
#       id = xpathApply(doc, postspath, xmlGetAttr, "id")[[i]]
#       id.ans = gsub("question-summary-", "", id)
#       # ===== Author =====
#       author = getNodeSet(p, ".//div[@class = 'user-details']/a")
#       author.ans = tryCatch(xmlValue(author[[1]]), error = function(e) return(NA))
#       # ===== Time Posted =====
#       path1 = ".//div[@class = 'user-action-time']/span"
#       time.ans = xpathApply(p, path1, xmlGetAttr, "title")[[1]]
#       # ===== Title of Post =====
#       title = getNodeSet(p, ".//div/h3/a[@class = 'question-hyperlink']")
#       title.ans = xmlValue(title[[1]])
#
#         # ===== Reputation Level =====
#       replevel = getNodeSet(p, ".//span[@class = 'reputation-score']")
#       rep.ans = tryCatch(xmlValue(replevel[[1]]), error = function(e) return(NA))
#       # ===== Current Views =====
#       path2 = ".//div[@class = 'views ']"
#       views = xpathApply(p, path2, xmlGetAttr, "title")[[1]]
#       views.ans = gsub(" views", "", views)
#       # ===== Current Num. of Answers =====
#       path3 = ".//div[@class = 'status unanswered']/strong | .//div[@class = 'status answered']/stron
#       answers = getNodeSet(p, path3)
#       num.ans = xmlValue(answers[[1]], trim = TRUE)
#       # ===== Votes =====
#       votes = getNodeSet(p, ".//div/span[@class = 'vote-count-post ']")
#       votes.ans = xmlValue(votes[[1]], trim = TRUE)
#       # ===== URL of Post =====
#       path4 = ".//div/h3/a[@class = 'question-hyperlink']"
#       url.ans = xpathApply(p, path4, xmlGetAttr, "href")[[1]]
#       url.ans = paste("http://stackoverflow.com", url, sep = "") # ===== Tags =====
#       path6 = ".//div[@class = 'summary']/div[2]"
#       tags = xpathApply(p, path6, xmlGetAttr, "class")[[1]]
#       tags = gsub("tags ", "", tags)
#       tags = gsub("t-", "", tags)
#       tags = gsub(" ", "; ", tags)
#       # ===== Combine all answers into a row =====
#       # id, date, title, tags, url, views, votes, answers, user, reputation & rbind it to data frame
#       d = rbind(d, data.frame(id.ans, time.ans, title.ans, tags, url.ans, views.ans, votes.ans, num.a
#     }
#     # end for loop for looping over individual posts
#     # ===== rbind individual pages to master data frame =====
#     scrape_data = rbind(scrape_data, d)
#
#
#   }
#   # end for loop for looping over individual pages
#   colnames(scrape_data) = c("id", "date", "title", "tags", "url", "views", "votes", "answers", "user"
```

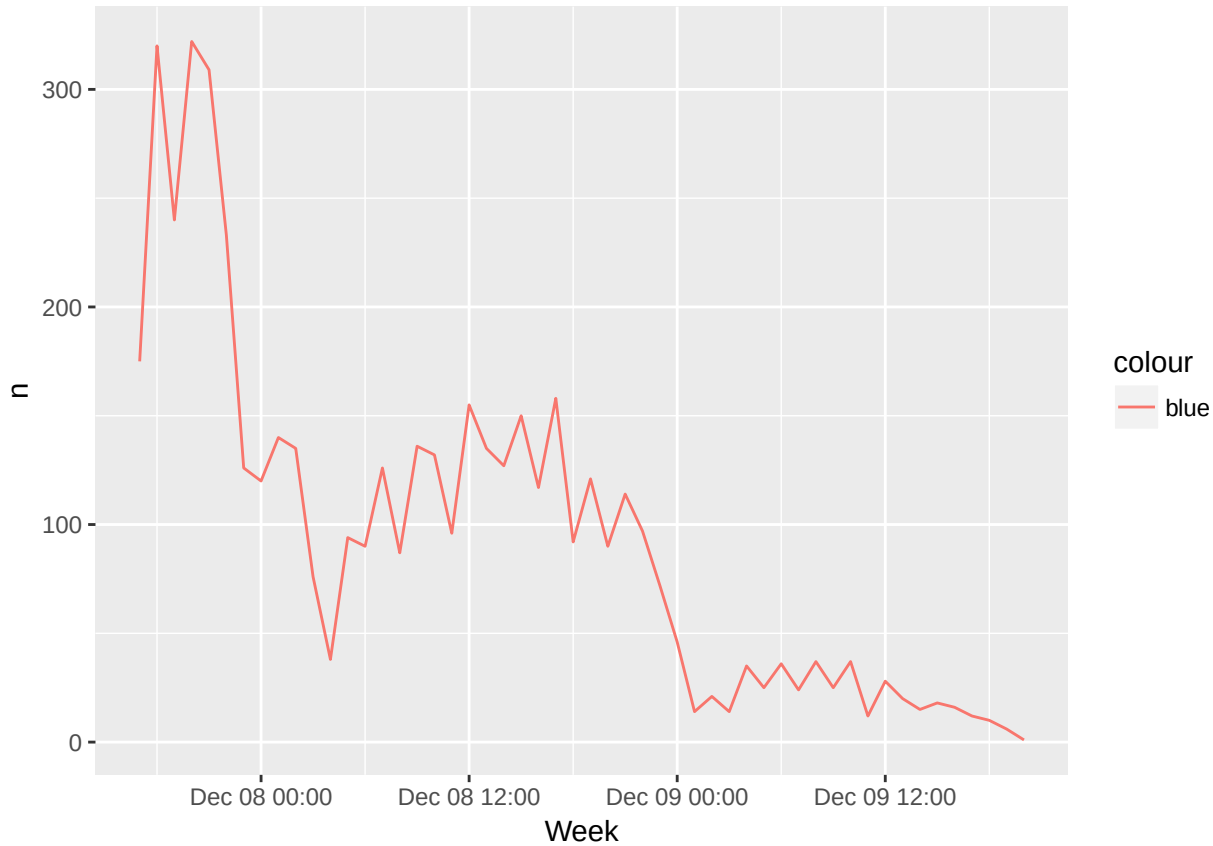
```
#   return(scrape_data)
# }
#
# # Running loop to scrape more number of pages from given URL for both R and Python
# #j = 25
# #python_data = NULL
# python_data1 = NULL
# for(j in 1:25) {
#
# u = paste("https://stackoverflow.com/questions/tagged/python?page=",j,  "" ,sep = '')
#   python_data = stack_data(u,j)
#   python_data1 = rbind(python_data1,python_data)
# }
#write.csv(r_data1,"Rdata.csv.csv")
#write.csv(python_data1, "Python_Data.csv")
```

Here we are not running the function because we have already saved the scraped data into the CSV files and using it for analysis purpose.

```
# Loading R and python data which is already scraped in above function
r_data1 <- read.csv("Rdata.csv.csv")
python_data1 <- read.csv("Python_data.csv")
# Analysing R data to answer what time of day Questions are posted
# Extracting hours from date time and creating data frame to plot a graph
hours_R <- hour(r_data1$date)
Rhours_df <- as.data.frame(table(hours_R))
colnames(Rhours_df) <- c("Hour_of_the_day", "Number_of_Questions")

# Plotting graph to show at what time frequency of questions getting posted is higher.
# From graph we can see that there is higher frequency of questions getting posted in
# the evening from 5 to 9 PM whereas frequency very low after midnight till early
# morning
ggplot(Rhours_df,aes(x= Hour_of_the_day,y= Number_of_Questions,
                     group= Number_of_Questions,fill= Number_of_Questions))+
geom_bar(stat="identity")+
scale_fill_gradient("##FF6666")
```




```
r_data1$date <- as.POSIXct(r_data1$date)

# This is trending line for questions posted over last 3 days. From graph we can see
# that more questions were posted on weekdays and frequency reduced over the weekend
r_data1 %>%
  count(Week = round_date(date, "hour")) %>%
  ggplot(aes(Week, n, colour = "blue")) +
  geom_line()
```

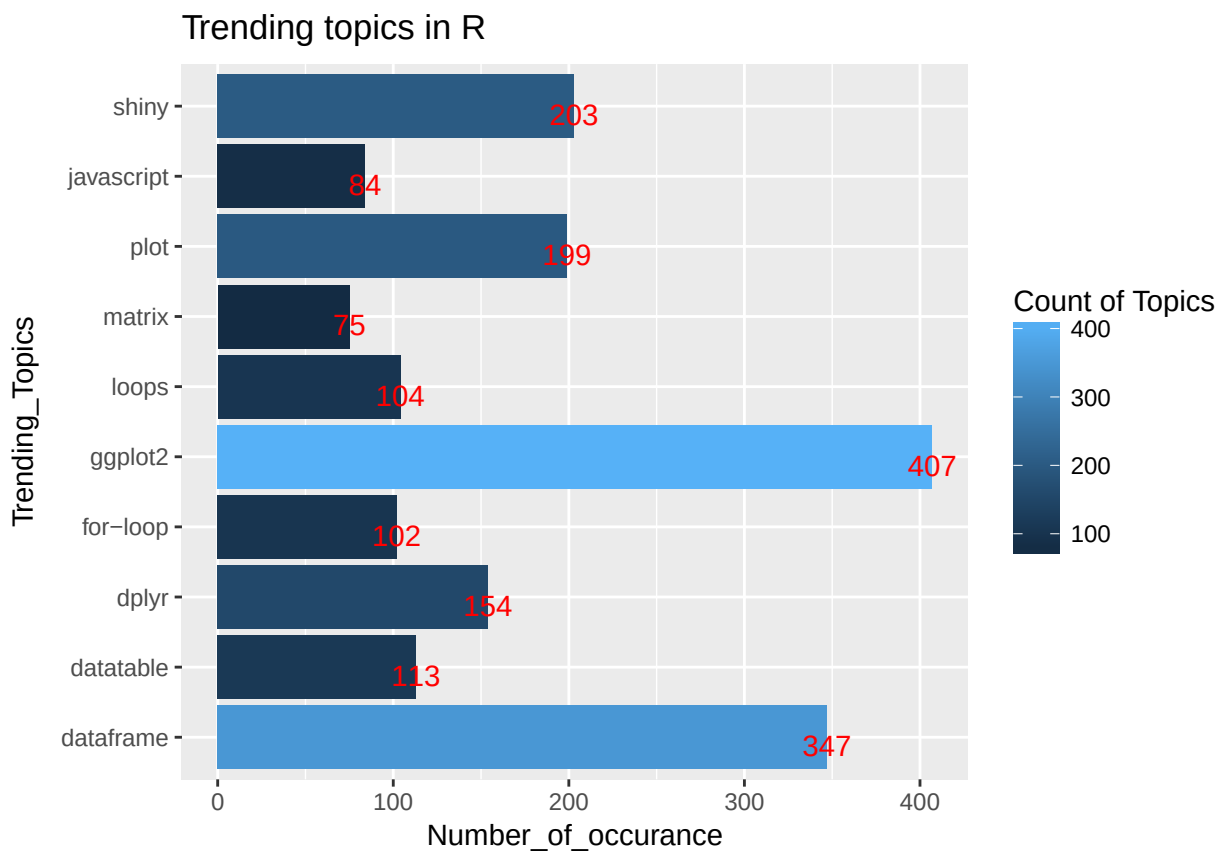



```
# Determining number of questions posted on weekdays and weekend
is_weekend <- function(d) {
  ifelse(wday(d, label = TRUE) %in% c("Sun", "Sat"), "Weekend", "Weekday")
}

questions_wday <- r_data1 %>%
  mutate(Weekend = is_weekend(date))

type_total <- questions_wday %>%
  count(Weekend) %>%
  rename(TypeTotal = n)
# Number of questions posted on weekdays are way higher than nmber of questions posted
# on weekends
type_total
```

```
## # A tibble: 2 x 2
##   Weekend TypeTotal
##     <chr>     <int>
## 1 Weekday      4447
## 2 Weekend       428
```

```
# What are trending topics in R

r_data_weekend <- questions_wday %>%  filter(Weekend == 'Weekend')
r_data_weekday <- questions_wday %>%  filter(Weekend == 'Weekday')

# Creating trending function to determine what are trending topics in R and Python
```

```r
# Cleaning scraped data using gsub and removing special characters manually
trending <- function (y) {
y <- gsub(";", ",", y)
find.listr <- list("r,", "ÃfÂ»","python,", "-3ÃfÂ»x", "-2ÃfÂ»7", "python")
find.stringr <- paste(unlist(find.listr), collapse = "|")
y <- gsub(find.stringr, replacement = "", y)
word_list_r <- strsplit(y, ",")
sep_words_r <- unlist(word_list_r)
trending_topics_r <- as.data.frame(tail(sort(table(sep_words_r)), 11))
trending_topics_r$sep_words_r <- gsub(" shiny", "shiny", trending_topics_r$sep_words_r)
trending_topics_r <- aggregate(Freq ~ sep_words_r, data = trending_topics_r, FUN = sum)
trending_topics_r <- trending_topics_r[c(-10),]
colnames(trending_topics_r) <- c("Trending_Topics", "Number_of_occurance")
return(trending_topics_r)
}

# Scraping data for R using trensing function
trending_topics_r <- trending(r_data1$tags)
# Trending topics in R
# Out of trending tags in R, ggplot and dataframe tags have higher number of questions
# posted (407 and 347 respectively) compared to other tags in R.
ggplot(trending_topics_r,aes(x= Trending_Topics,y= Number_of_occurance,
                     group= Number_of_occurance,fill= Number_of_occurance))+
geom_bar(stat="identity")+
ggtitle("Trending topics in R") +
geom_text(aes(label= Number_of_occurance), vjust = 0.9, color ='red') +
scale_fill_gradient("Count of Topics") +
coord_flip()
```

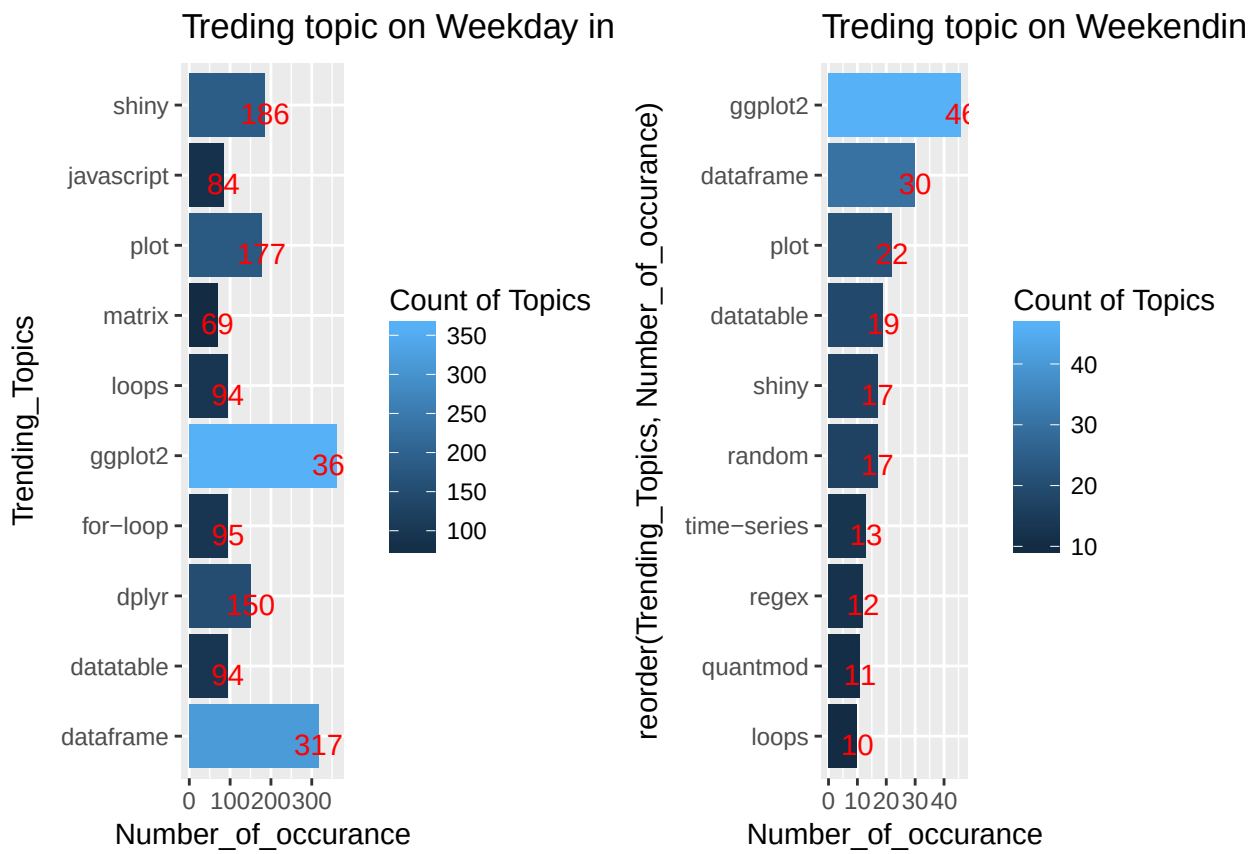



```
# Analysing trending topic in R on weekday and weekend
trending_topics_r_weekend <- trending(r_data_weekend$tags)
trending_topics_r_weekday <- trending(r_data_weekday$tags)

require(gridExtra)
```

```
## Loading required package: gridExtra

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

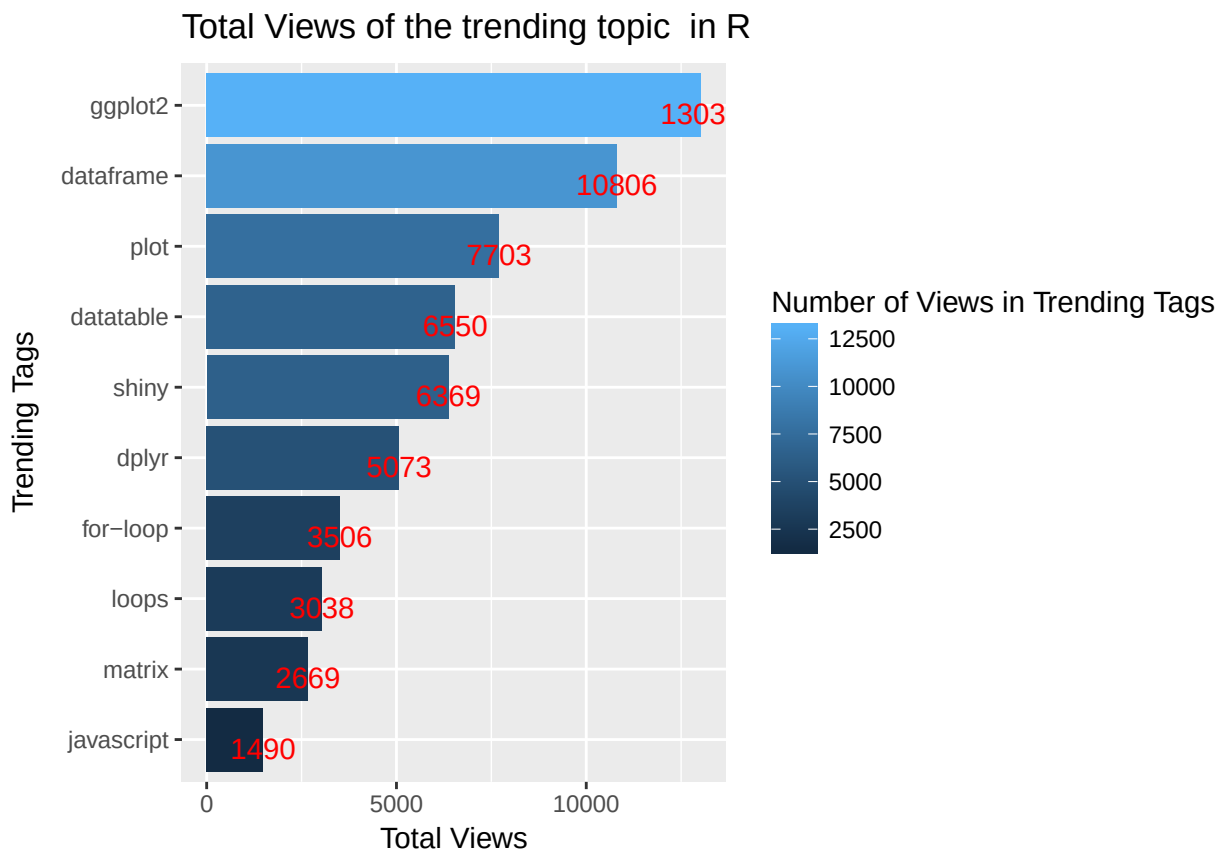
```
wday <- ggplot(trending_topics_r_weekday,aes(x= Trending_Topics,y= Number_of_occurance,   group= Number_
geom_bar(stat="identity")+
 ggtitle("Treding topic on Weekday in R") +
geom_text(aes(label= Number_of_occurance), vjust = 0.9, color ='red') +
scale_fill_gradient("Count of Topics") +
coord_flip()

wend <- ggplot(trending_topics_r_weekend,aes(x= reorder(Trending_Topics,Number_of_occurance),y= Number_
                      group= Number_of_occurance,fill= Number_of_occurance))+
geom_bar(stat="identity")+
geom_text(aes(label= Number_of_occurance), vjust = 0.9, color ='red') +
  ggtitle("Treding topic on Weekendin R") +
scale_fill_gradient("Count of Topics") +
```

```r
coord_flip()

# Here we are analysing number of questions posted in trending topics on weekdays and
# weekends. From graph it is seen that total number of questions posted on weekdays
# on each topic is higher than number of questions posted on weekedns
grid.arrange(wday, wend, ncol=2,widths=c(1.5,1.5))
```

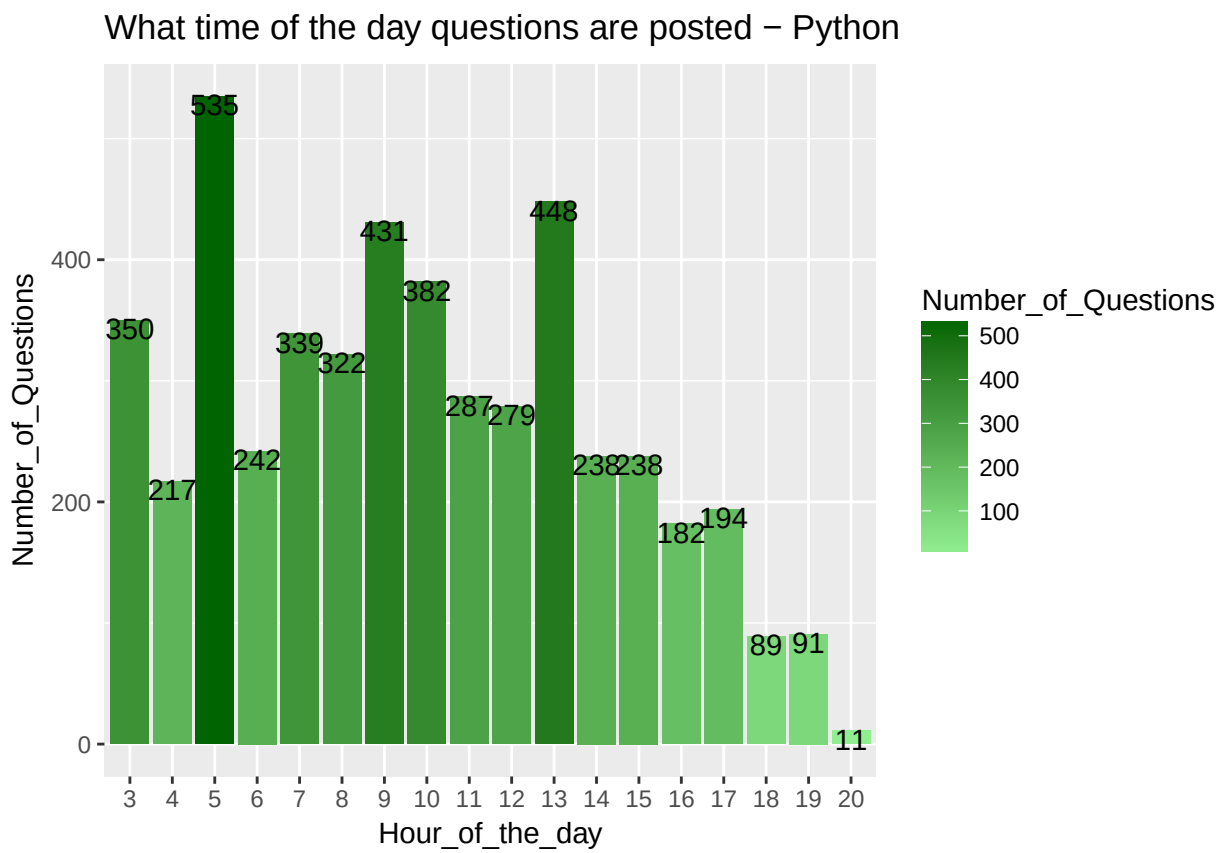



```r
#Counting views for each trending topic - R
trending_topics_rchar <- as.character(trending_topics_r$Trending_Topics)
trending_views_r <- NULL
trending_views_all_r <- NULL
k = 1
for(k in 1:10) {
  matching_r <- r_data1[grep(trending_topics_rchar[k], r_data1$tags),]
  matching_r$views <- as.numeric(matching_r$views)
  total_views_r <- as.character(sum(matching_r$views))
  trending_views_r <- data.frame(trending_topics_rchar[k], total_views_r)
  trending_views_all_r <- rbind(trending_views_all_r, trending_views_r)
}
colnames(trending_views_all_r) <- c("Trending_Topics_R", "TotalViews")

trending_views_all_r[,2] <- as.numeric(as.character(trending_views_all_r[,'TotalViews']))

# Total views for dataframe and ggplot are much higher than views of other topics.
# Whereas views for javascript and matrix have comparatively lower views. We can
# conclude that R programmers mostly face issues with ggplot and dataframe topics
```

```r
# and hence they post more number of questions and views answers related to the same.
ggplot(trending_views_all_r,aes(x= reorder(Trending_Topics_R,TotalViews),y= TotalViews  , group= TotalV
  geom_bar(stat = 'identity')+
  geom_text(aes(label= TotalViews), vjust = 0.9, color ='red') +
  labs(x = "Trending Tags", y= "Total Views") +
    ggtitle("Total Views of the trending topic  in R") +
    scale_fill_gradient("Number of Views in Trending Tags") +
  coord_flip()
```




```r
# What time of the day questions are posted - python

hours_py <- hour(python_data1$date)
pyhours_df <- as.data.frame(table(hours_py))
colnames(pyhours_df) <- c("Hour_of_the_day", "Number_of_Questions")

# Plotting graph to show at what time frequency of questions getting posted is higher.
# From graph we can see that there is higher frequency of questions getting posted at
# 5 am EST and then 1 pm EST. it might be possible because stack overflow is used
# worldwide and python programmers from other country are more active during this time
# to post or answer questions related to Python.
ggplot(pyhours_df,aes(x= Hour_of_the_day,y= Number_of_Questions,
                      group= Number_of_Questions,fill= Number_of_Questions))+
  geom_bar(stat="identity")+
  geom_text(aes(label= Number_of_Questions), vjust = 0.9, color ='black') +
  ggtitle("What time of the day questions are posted - Python") +
  scale_fill_gradient(low = "light green", high = "dark green")
```



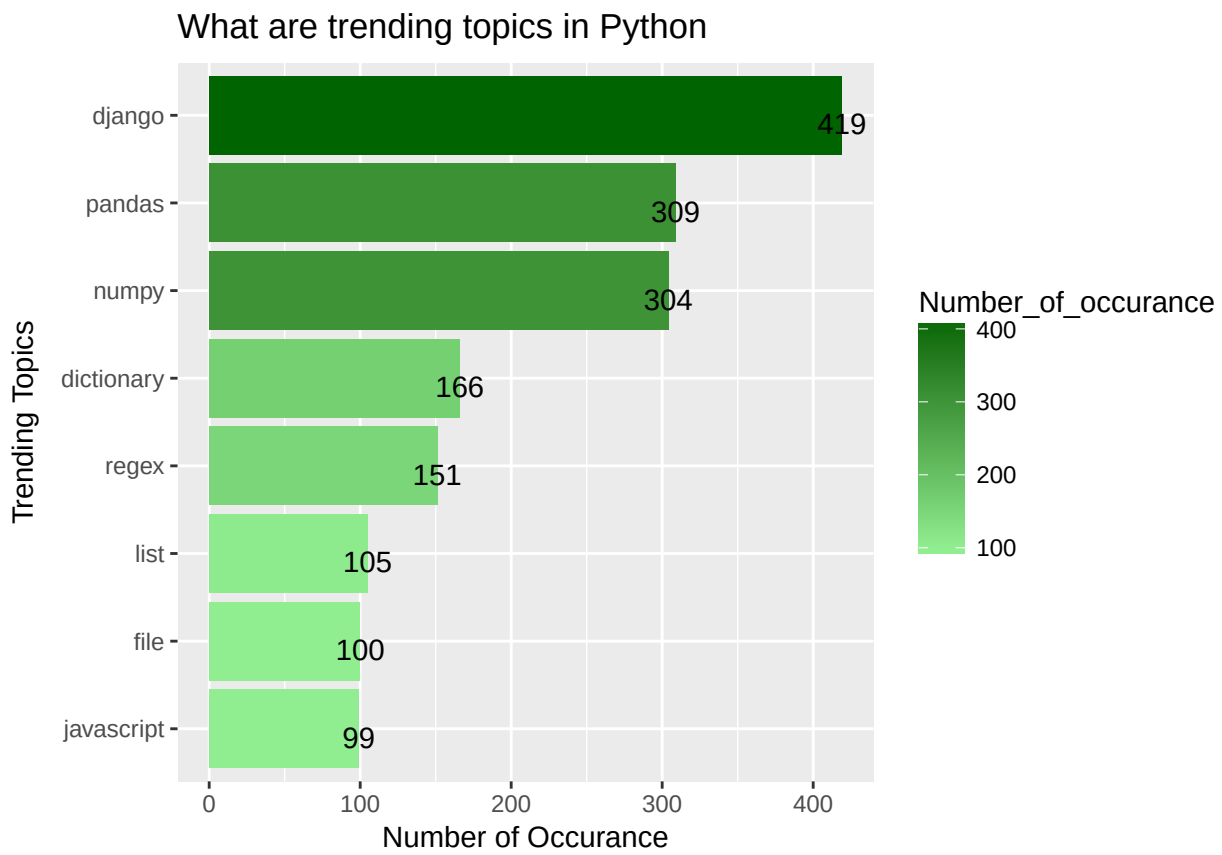

```
questions_wday_python <- python_data1 %>%
  mutate(Weekend = is_weekend(date))

type_total_py <- questions_wday_python %>%
  count(Weekend) %>%
  rename(TypeTotal = n)
type_total_py
```

```
## # A tibble: 1 x 2
##   Weekend TypeTotal
##     <chr>     <int>
## 1 Weekend      4875
```

```
# We scraped 25 pages for both R and Python data. When we analysed the data we were
# able to distinguish R data for weekdays and weekend but for python, all 25 pages
# data was generated in the weekend. Which clearly indicated that Python has more
# number of active users and more number of questions gets posted related to Python.
```

```
python_data_weekend <- questions_wday_python %>%  filter(Weekend == 'Weekend')
python_data_weekday <- questions_wday_python %>%  filter(Weekend == 'Weekday')

#What are trending topics in Python
trending_topics_py <- trending(python_data1$tags)
trending_topics_py <- trending_topics_py[-c(1:2),]

# From graph it is seen that, django, pandas and numpy have more number of questions
# posted than any other topic in python.
```

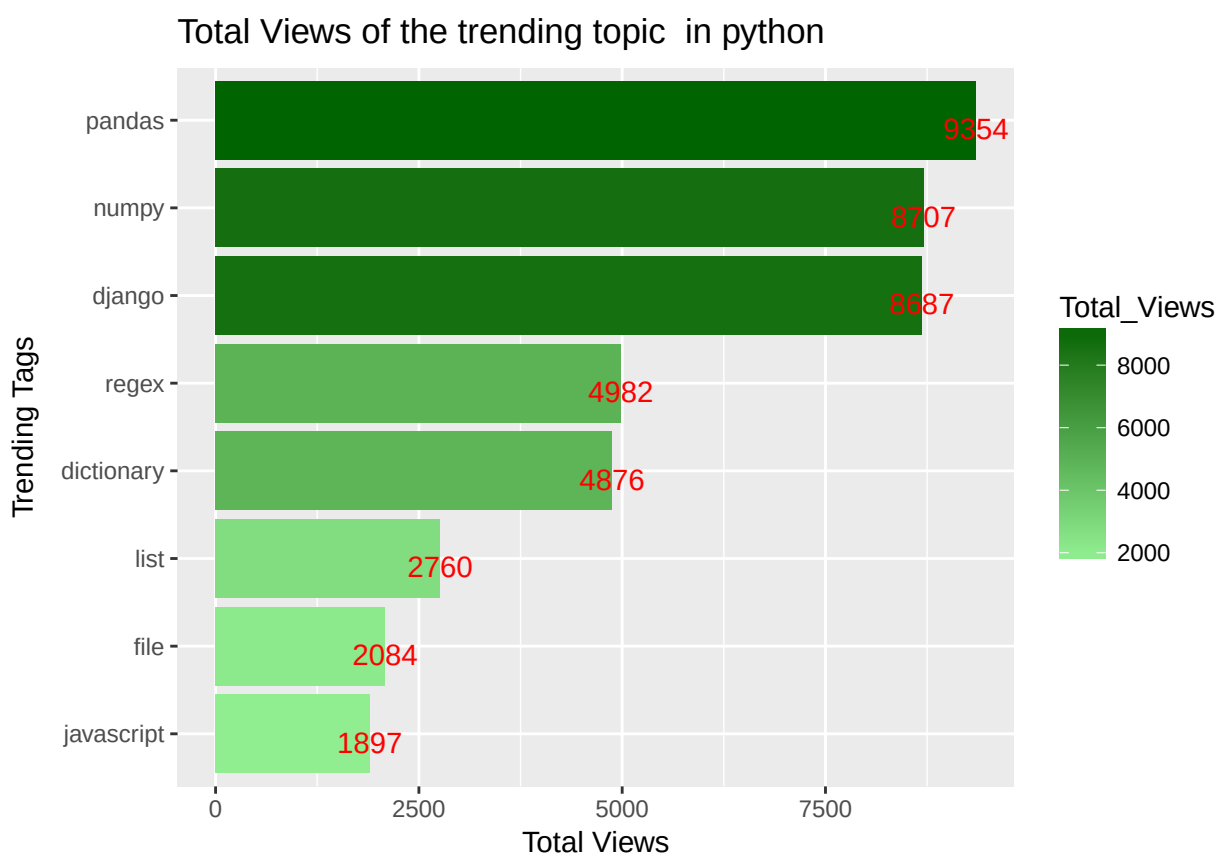
```
ggplot(trending_topics_py,aes(x= reorder(Trending_Topics, Number_of_occurance),y= Number_of_occurance, g
geom_bar(stat="identity")+
geom_text(aes(label= Number_of_occurance), vjust = 0.9, color ='black') +
labs(x = "Trending Topics", y= "Number of Occurance") +
ggtitle("What are trending topics in Python") +
scale_fill_gradient(low = "light green", high = "dark green") +
coord_flip()
```




```
#Counting views for each trending topic - Python
trending_topics_char <- as.character(trending_topics_py$Trending_Topics)
trending_views <- NULL
trending_views_all <- NULL
i = 1
for(i in 1:10) {
matching <- python_data1[grep(trending_topics_char[i], python_data1$tags),]
matching$views <- as.numeric(matching$views)
total_views <- as.character(sum(matching$views))
trending_views <- data.frame(trending_topics_char[i], total_views)
trending_views_all <- rbind(trending_views_all, trending_views)
}
colnames(trending_views_all) <- c("Trending_Topics_python", "Total_Views")

trending_views_all[,2] <- as.numeric(as.character(trending_views_all[,'Total_Views']))

trending_views_all <- na.omit(trending_views_all)
trending_views_all
```

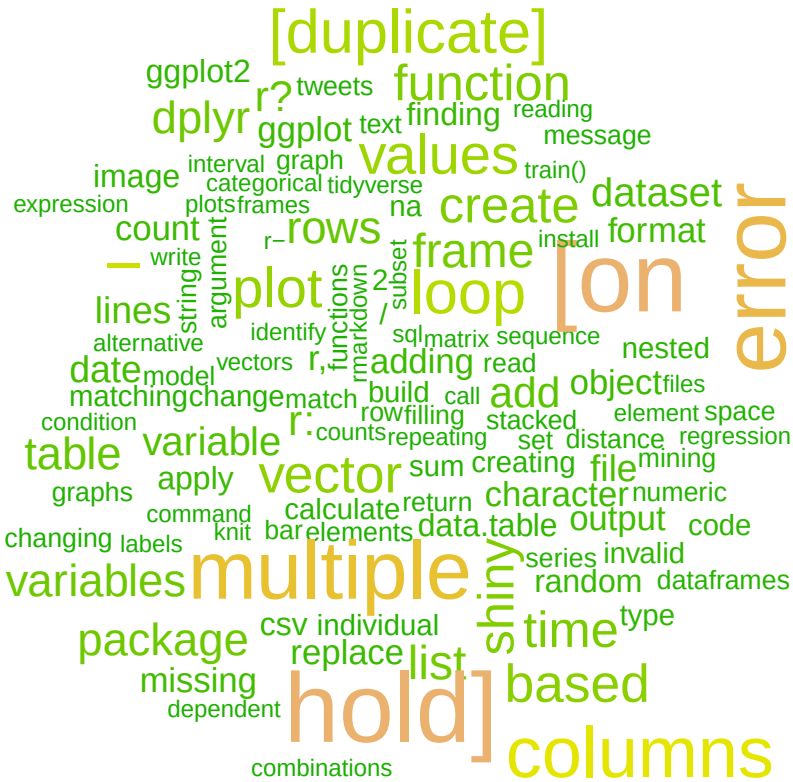
```
##   Trending_Topics_python Total_Views
## 1             dictionary        4876
## 2                 django        8687
## 3                   file        2084
## 4                   list        2760
## 5                  numpy        8707
## 6                 pandas        9354
## 7                  regex        4982
## 8             javascript        1897
```

```
# Total views for pandas, numpy and django are much higher than views of other topics # Whereas views f
# conclude that python programmers mostly face issues with pandas, numpy and django
# topics and hence they post more number of questions and views answers related to the
# same.
ggplot(trending_views_all,aes(x= reorder(Trending_Topics_python,Total_Views),y= Total_Views  , group= T
  geom_bar(stat = 'identity')+
  geom_text(aes(label= Total_Views), vjust = 0.9, color ='red') +
  labs(x = "Trending Tags", y= "Total Views") +
  ggtitle("Total Views of the trending topic  in python") +
  scale_fill_gradient(low = "light green", high = "dark green") +
  coord_flip()
```



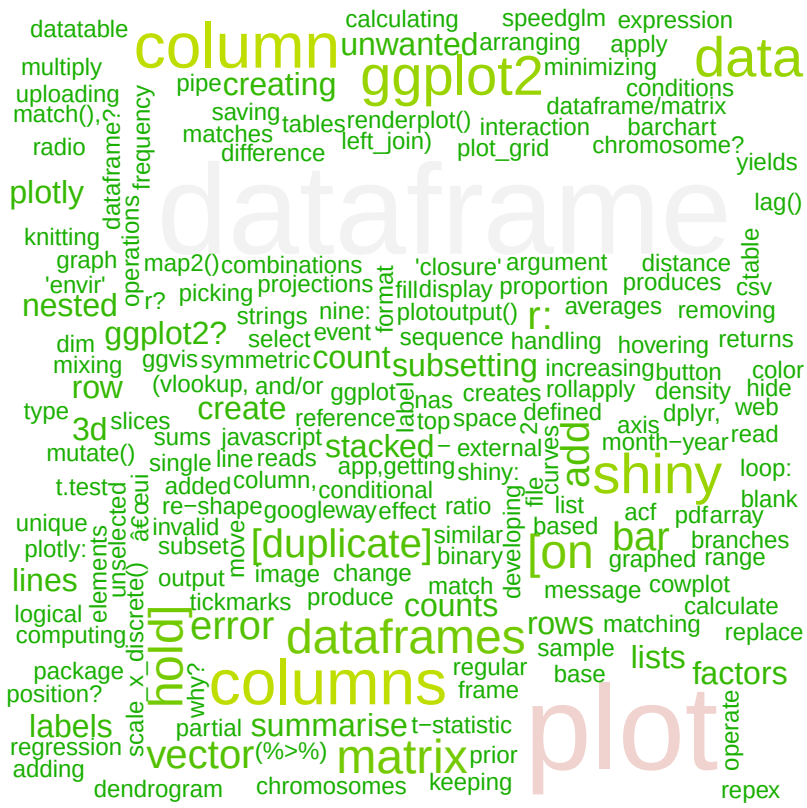

```
# What are questions asked in that trending topic in R ?
# We have used unnest_token function to seperate each word in R dataframe and count
# it's number of occurance in the dataframe.
title_words <- r_data1 %>% select(title,views)  %>%
  unnest_tokens(word, title, token = stringr::str_split, pattern = " ") %>% count(word, sort = TRUE)
```

```
title_word_counts <- title_words %>%
  anti_join(stop_words, by = c("word" = "word"))

# We have made word cloud of words that are present in question titles of R
suppressWarnings(wordcloud(title_word_counts$word,title_word_counts$n,col=terrain.colors(length(title_w
```

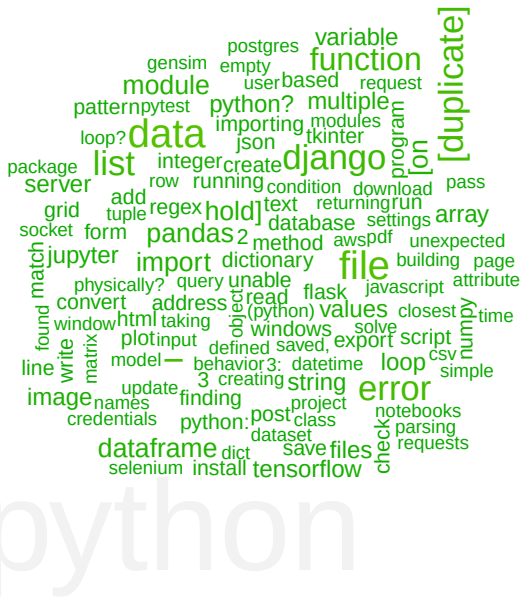



```
# questions asked in tredning topics in R
trending_topics_rchar <- as.character(trending_topics_r$Trending_Topics)

trending_questions <- NULL
trending_questions_all <- NULL
# i = 4
for(i in 1:10) {
matching <-r_data1[grep(trending_topics_rchar[i], r_data1$title),]
matching$title <- (matching$title)
if(length(matching$title)==0)  {
            next
        } else {
trending_questions <- data.frame(trending_topics_rchar[i], matching$title)
trending_questions_all <- rbind(trending_questions_all, trending_questions)
}
}
colnames(trending_questions_all) <- c("Trending_Topics", "Trending_questions")

trending_questions <- data.frame(unique(trending_questions_all$Trending_questions))
```

```
colnames(trending_questions) <- c( "Trending_questions")
trending_questions$Trending_questions <- as.character(trending_questions$Trending_questions)

title_words <- trending_questions %>% select(Trending_questions)  %>%
  unnest_tokens(word, Trending_questions, token = stringr::str_split, pattern = " ") %>% count(word, so


title_word_counts <- title_words %>%
  anti_join(stop_words, by = c("word" = "word"))

# This is word cloud for trending topics in R
suppressWarnings(wordcloud(title_word_counts$word,title_word_counts$n,col=terrain.colors(length(title_w
```




```
title_words <- python_data1 %>% select(title,views)  %>%
  unnest_tokens(word, title, token = stringr::str_split, pattern = " ") %>% count(word, sort = TRUE)

title_word_counts <- title_words %>%
  anti_join(stop_words, by = c("word" = "word"))

# This is word cloud for all questions asked related to Python
suppressWarnings(wordcloud(title_word_counts$word,title_word_counts$n,col=terrain.colors(length(title_w
```




```
trending_questions <- NULL
trending_questions_all <- NULL
#i = 1
for(i in 1:10) {
matching <- python_data1[grep(trending_topics_char[i], python_data1$title),]
matching$title <- (matching$title)
#total_views <- as.character(sum(matching$views))
trending_questions <- data.frame(trending_topics_char[i], matching$title)
trending_questions_all <- rbind(trending_questions_all, trending_questions)
}

colnames(trending_questions_all) <- c("Trending_Topics", "Trending_questions")

trending_questions <- data.frame(unique(trending_questions_all$Trending_questions))
colnames(trending_questions) <- c( "Trending_questions")
trending_questions$Trending_questions <- as.character(trending_questions$Trending_questions)

title_words <- trending_questions %>% select(Trending_questions)  %>%
  unnest_tokens(word, Trending_questions, token = stringr::str_split, pattern = " ") %>% count(word, so

title_word_counts <- title_words %>%
  anti_join(stop_words, by = c("word" = "word"))

# We have made word cloud of words that are present in trending topic in python
suppressWarnings(wordcloud(title_word_counts$word,title_word_counts$n,col=terrain.colors(length(title_w
```

```
trending_topics_rchar <- as.character(trending_topics_r$Trending_Topics)
trending_topics_rchar
```

```
##  [1] " dataframe" " datatable" " dplyr"     " for-loop"  " ggplot2"
##  [6] " loops"     " matrix"    " plot"      "javascript" "shiny"
```

```
related_tags_frame <- NULL
related_tags_data_frame<- NULL
#i
for(i in 1:10) {
  p = paste("https://stackoverflow.com/questions/tagged/r+",trending_topics_rchar[i],  "" ,sep = '')
#Reading the HTML code from the website
webpage <- read_html(p)

#Using CSS selectors to scrap the rankings section
related_tags_html <- html_nodes(webpage,'.js-gps-related-tags')
#Converting the related data to text
related_tags <- html_text(related_tags_html)
#Let's have a look at the rankings

if(length(related_tags) ==0)  {
            next
        }
else
          {

related_tags <- gsub("[\r\n]", "", related_tags)
```

```r
trim <- function( x ) {

   gsub("(^[[:space:]]+|[[:space:]]+$)", "", x)
}



related_tags <- trim(related_tags)

related_tags <- str_replace_all(related_tags, "[^[:alnum:]]", " ")
related_tags <- gsub('[0-9]+', '', related_tags)
related_tags_data <- data.frame(strsplit(gsub("[^[:alnum:] ]", "", related_tags), " +"))
related_tags_data<- related_tags_data[c(-1:-2),]

related_tags_data <- data.frame(related_tags_data)

related_tags_data <- related_tags_data[1:29,]

related_tags_data_frame <- data.frame(trending_topics_rchar[i],related_tags_data)
related_tags_frame <- rbind(related_tags_frame,related_tags_data_frame)
}
}
head(related_tags_data_frame,10)
```

```
##    trending_topics_rchar.i. related_tags_data
## 1                     shiny         shinyapps
## 2                     shiny             shiny
## 3                     shiny            server
## 4                     shiny    shinydashboard
## 5                     shiny            ggplot
## 6                     shiny           leaflet
## 7                     shiny        javascript
## 8                     shiny           rstudio
## 9                     shiny                dt
## 10                    shiny            plotly
```

```r
# We have also scraped related tags for given question. Here is the example of shiny
# tag. These are various tags which are related to shiny tags and appear on the same
# page along with questions asked related to shiny.
```

```r
# Analysis on users and views
users <- as.data.frame(tail(sort(table(r_data1$user)), 10))
users$Var1 <- as.character(users$Var1)
max_users1 <- NULL
#m = 1
for (m in 1:10) {
  max_users <- subset(r_data1, r_data1$user == users$Var1[m])
  max_users1 <- rbind(max_users1, max_users)
  }
max_users1 <- max_users1[,10:11]
users_repu <- unique(max_users1)
# Following is the list of users who answered most number of questions. Pan is the
# user who answered maximum number of times (67 times)
users[order(users$Freq, decreasing = TRUE),]
```

```
##                  Var1 Freq
## 10                Pan   67
## 9        J. McCraiton   65
## 8                Haze   53
## 7         Ashmin Kaul   48
## 6   Travers Woodward   46
## 5  Jennifer Williams   44
## 3                MAPK   42
## 4             warwcat   42
## 2               Taran   35
## 1           Mars_Tina   34
```

```
# Following is the list of users who answered most nmber of questions with their
# reputation
users_repu[order(users_repu$reputation, decreasing = TRUE),]
```

```
##                    user reputation
## 2853      J. McCraiton         65
## 547                Pan         65
## 3475  Travers Woodward         32
## 1807              Haze         17
## 1372       Ashmin Kaul        154
## 52             warwcat        119
## 228          Mars_Tina         11
## 2570 Jennifer Williams         11
## 1179             Taran        101
## 424               MAPK      1,251
```

```
##Pushing the data scraped in SQL server for saving the historical data and further analysis required i
##used RODBC package
channel <- odbcDriverConnect("driver=SQL Server;server=LAPTOP-1MVK9TRM")
 r_data1$date <- as.factor(as.POSIXct(r_data1$date))
#pushing R data
sqlSave(channel,r_data1, rownames = FALSE)
#Pushing Sql Data
sqlSave(channel,python_data1, rownames = FALSE)
```

Conclusins

. From the analysis carried out we have concluded that Python is more popular than R.

. In python, most of the people ask questions related to Numpy, Panda and Django whereas in R most of the people ask questions related to dataframe and ggplot.

. People for different countires apart from US are also very active from the time analysis carried out. We also saw that the high volume ofpython questions were asked in early morning.

. Analysis carried out helped to realize that users with high reputation have high frequency of answering questions.

.From the analysis carried we concludethat most of the questions in R and python were about visualization and numerical computation of data