ps6

Malvika Rajeev

10/30/2018

Question 2

Which users have asked Spark-related questions (tags that have "apache-spark" as part of the tag) BUT not python.

```
library(RSQLite)
drv <- dbDriver("SQLite")
dir <- '~/' # relative or absolute path to where the .db file is
dbFilename <- 'stackoverflow-2016.db'
db <- dbConnect(drv, dbname = file.path(dir, dbFilename))

result <- dbGetQuery(db, "select * from users u
                          join questions q on u.userid = q.ownerid
                          join questions_tags t on q.questionid = t.questionid
                          LEFT JOIN
                          (
                          SELECT distinct q.ownerid
                          FROM questions q
                          JOIN questions_tags t
                          ON t.questionid=q.questionid
                          WHERE t.tag LIKE '%python%'
                          ) f
                          ON q.ownerid=f.ownerid
                          WHERE f.ownerid IS NULL
                          AND
                          t.tag LIKE '%apache-spark%'")

head(result["displayname"])
```

```
##        displayname
## 1    Balachandar S
## 2    Balachandar S
## 3 D. M<U+00FC>ller
## 4          karlson
## 5              pls
## 6           mtyson
```

```
cat("The number of unique users are ",length(unique(result[["userid"]])))
```

```
## The number of unique users are  4647
```

Question 3

For question 3, and given the timeline of the dataset, I was curious about the wikipedia traffic concerniing the terror attacks on Mumbai on 26th November, 2008, which was one of the deadliest terror attacks in India's

modern history. It also initiated a global discussion and the perception that the government of Pakistan was turning a blind eye to terror-related issues within its borders. So, firstly I wanted to see how many people viewed the page "terrorism in pakistan" in the page.

Suprisingly there wasn't an extremely high surge, but in general the number do increase on and after the time frame of 26th November.

```
library(dplyr)
```

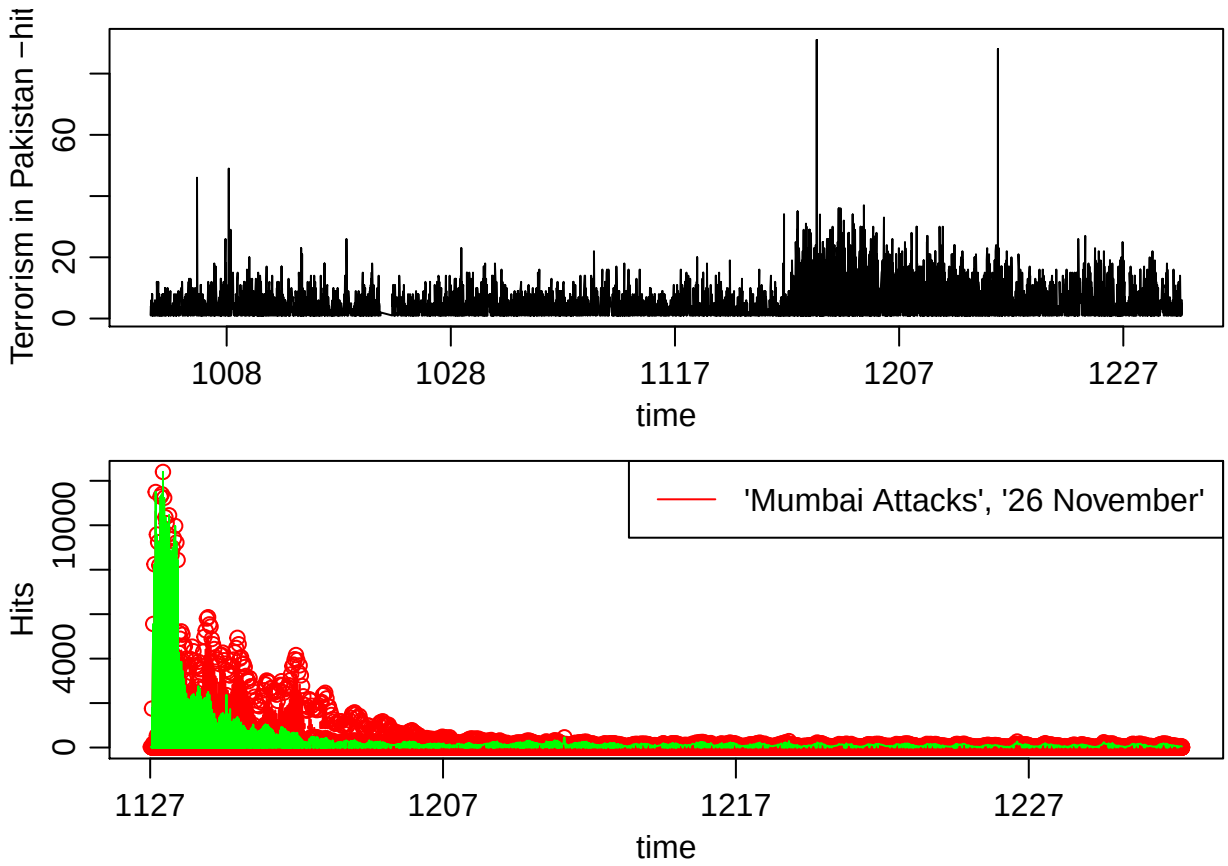
```
##
## Attaching package: 'dplyr'
```

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

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

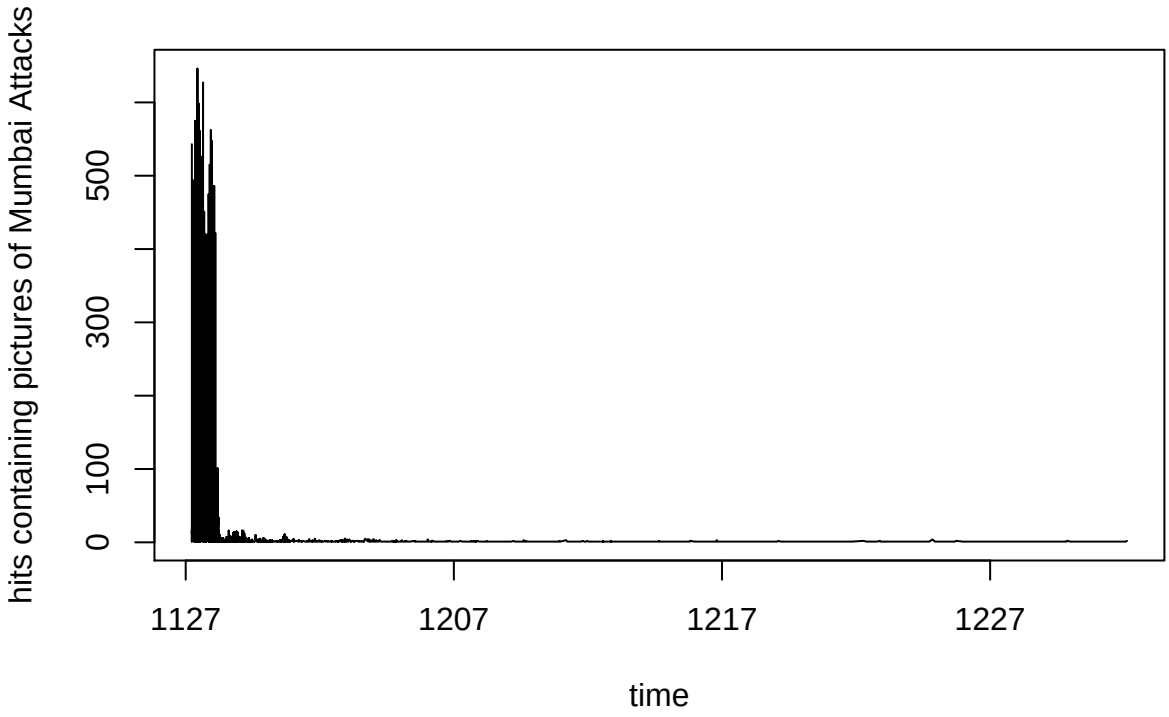
```
library(chron)
library(readr)
library(stringr)
library(ggplot2)

data <- read.csv("~/Desktop/part-00000", quote = "", header = FALSE, sep = ",", stringsAsFactors = FALSE
data = as.data.frame(data)

#Cleaning the dataset

data = as.data.frame(data)
colnames(data) <- c("date", "time", "lang", "page", "hits")
data$date <- as.character(data$date)
data$time <- as.character(data$time)
data$time[data$time %in%  c("0", "1")] <- "000000"
wh <- which(nchar(data$time) == 5)
data$time[wh] <- paste0("0", data$time[wh])
data$chron <- chron(data$date, data$time,
                    format = c(dates = 'ymd', times = "hms"))
data$chron <- data$chron + 5.5/24 ##INDIAN ST


##segregating the trends

hits_tp = subset(data, grepl("Terrorism.*Pakistan", page, ignore.case=TRUE))
#how many people searched for terrorism in pakistan over the months

trend1 = subset(data, grepl("Mumbai.*Attack", page, ignore.case=TRUE))
trend2 = subset(data, grepl("26_November", page, ignore.case=TRUE))
##the starting point of this data will be 26th november obviously

par(mfrow = c(2, 1), mgp = c(1.8, 0.7, 0), mai = c(0.6, 0.6, 0.1, 0.1))

plot(hits_tp$chron, hits_tp$hits, type = 'l', xlab = 'time', ylab = 'Terrorism in Pakistan -hits')


plot(trend1$chron, trend1$hits, type = 'b', xlab = 'time', ylab = 'Hits', col = "red")
```

```
lines(trend2$chron,trend2$hits,col="green")

legend('topright', legend = c("'Mumbai Attacks', '26 November'"), col = c("red", "green"), lty=1:2)
```

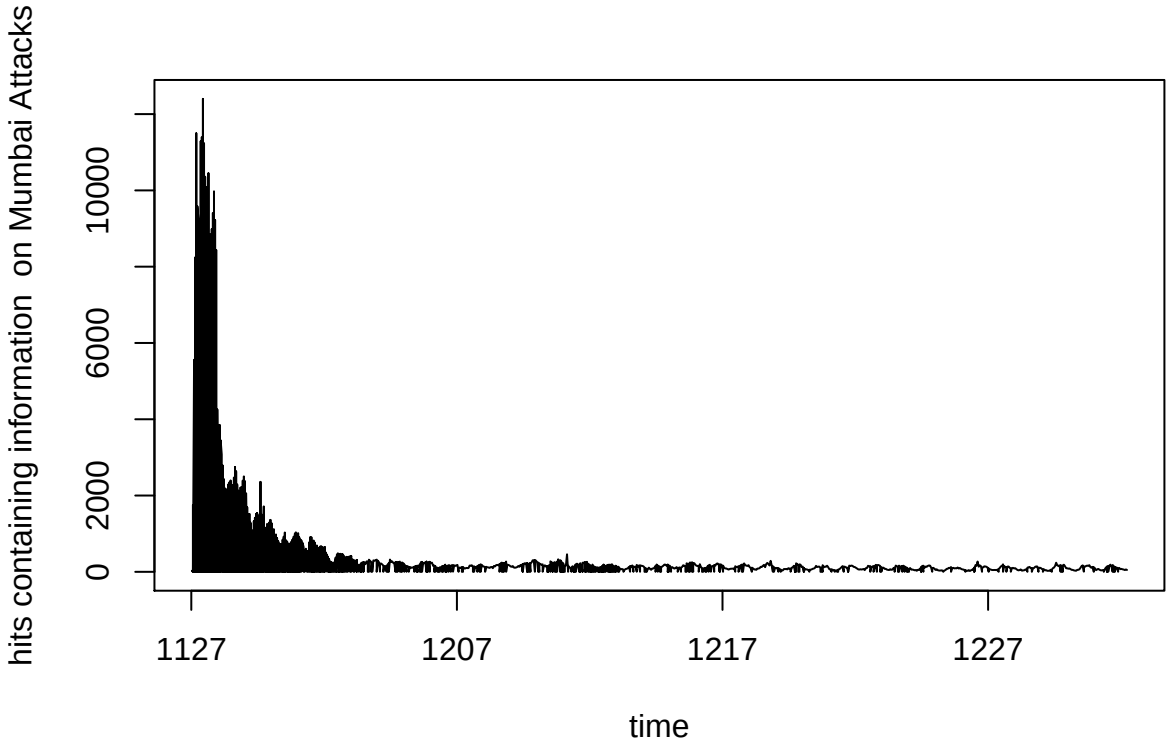


Then, I wanted to see what proportion of hits of pictures to reading up on the information. Turns out the proportion was almost equal. It does sort of make sense because there is a wide documentation of images taken by bystanders during the actual attacks.

```
pictures <- subset(trend2, grepl("\\.png", trend2[,4]))
information <- subset(trend2, (!grepl("\\.png", trend2[,4])))

plot(pictures$chron, pictures$hits, type = 'l', xlab = 'time', ylab = 'hits containing pictures of Mumba
```




```
plot(information$chron, information$hits, type = "l", xlab = 'time', ylab = 'hits containing information
```




```
proportion <- (nrow(pictures) / nrow(information))
proportion
```

```
## [1] 0.8285887
```

Also, while both curves seem to be very concentrated on the date 27-28th November, it makes sense that there were fewer and fewer hits for pictures then textual information over time.

Here is the script I used to extract information:

```
dir = '/global/scratch/paciorek/wikistats_full'
lines = sc.textFile(dir + '/' + 'dated')
import re
from operator import add
pattern = "2008_Mumbai_attacks|Operation_Water_Rat|National_Investigation_Agency|Lakshar_e_Taiba|Terror
pattern = re.compile(pattern)
def find(line, regex = pattern, language = None):
    vals = line.split(' ')
    if len(vals) < 6:
        return(False)
    tmp = re.search(regex, vals[3])
    if tmp is None or (language != None and vals[2] != language):
        return(False)
    else:
        return(True)


interest = lines.filter(find)
#interest.counts()
### map-reduce step to sum hits across date-time-language-hits quadruplets###

def stratify(line):
   vals = line.split(' ')
   return(vals[0] + '-' + vals[1] + '-' + vals[2] + '-' + vals[3], int(vals[4]))
counts = interest.map(stratify).reduceByKey(add)
### map step to prepare output ###
def transform(vals):
    # split key info back into separate fields
    key = vals[0].split('-')
    return(",".join((key[0], key[1], key[2], key[3], str(vals[1]))))
### output to file ###
# have one partition because one file per partition is written out
dir2 = '/global/scratch/malvikarajeev'
outputDir = dir2 + '/' + 'numberofhits'
counts.map(transform).repartition(1).saveAsTextFile(outputDir)
```

QUESTION 4

PART (A)

First, the shell script to submit the job:

```
#!/bin/bash
# Job name:
#SBATCH --job-name=obama
#
# Account:
#SBATCH --account=ic_stat243
#
# Partition:
#SBATCH --partition=savio2
```

```
#
# Number of tasks needed for use case (example):
#SBATCH --nodes=1
#
# Processors per task:
#SBATCH --cpus-per-task=1
#
# Wall clock limit:
#SBATCH --time=02:00:00
#
## Command(s) to run:
module load r r-packages
R CMD BATCH --no-save obama.R obama.Rout
```

As discussed, I set the number of cores to 12.

```
pathf <- '/global/scratch/paciorek/wikistats_full/dated'
file_names <- list.files(pathf, pattern="part*", full.names=TRUE)
##Get the names of the files in a list form using regex, also we know all the files are in the form par


library(readr)
library(parallel)
library(doParallel)
nCores <- 12  #otherwise as.integer(Sys.getenv("SLURM_CPUS_ON_NODE"))

registerDoParallel(nCores)

extract <- function(file){
                dat <- read.delim(file, sep = "",  header = FALSE, stringsAsFactors = FALSE)
                subset(dat, grepl("Barack_Obama", dat[, 4], ignore.case= TRUE))}


result <- foreach(f = file_names,
                  .packages = c("readr"), # libraries to load onto each worker
                  .combine = rbind,
                   .multicombine = FALSE, # how to combine results
                  .verbose = TRUE) %dopar% {
                    output <- extract(f)

                    output # this will become part of the result object
                  }
desiredRows <- write.csv(result, file = "/global/scratch/malvikarajeev/obama-counts.txt")
print(nrow(result))
```

I got 433706 rows.

PART (B)

The Spark code provided with Unit 7, it takes ~15 minutes using 96 cores to create the filtered dataset that just has the Obama-related webpage traffic using Spark. Using pararell R, I got the filtered database using 12 cores (1 node) in

proc.time() user system elapsed 45938.920 2930.044 6053.817

so

```
#Total elapsed time

t = (6053.817/60)

cat("It took ", t, "minutes on 12 cores", sep = " ")
```

```
## It took  100.897 minutes on 12 cores
```

```
new = (t*12)/ 96
cat(" it'll take approximately ", new, "minutes on 96 cores." , sep = " ")
```

```
##  it'll take approximately  12.61212 minutes on 96 cores.
```

So maybe R is more efficient in this case.