ps6

Yihuan Song

10/23/2018

question 1

(a) The goals of their simulation study are to test the null hypothesis of a k0-component normal mixture distribution is asymptotically distributed as a weighted sum of independent chi-squared random variables with one degree of freedom. The metrics that they consider in assessing their method is to test a single normal versus a two-component normal mixture, and a two-component normal mixture versus a three-component normal mixture by 1000 samples generated for each sample size.

(b) The choices the authors had to make in designing their simulation study included the number of components for the normal mixture, number of samples for each sample size, sample sizes and the significance level. The key aspects of the data generating mechanism that likely affect the statistical power of the test are the mixing proportion, sample sizes, significance level and measures of distance between the two component. There are data-generating scenarios that the authors did not consider that would be useful to consider, for example, k-component mixture with unequal variance.

(c) The results make sense, since in general, the power increases as distance increases, sample size increases, or normal level increases. Meaning that with larger distance between the two components, larger sample size in general or larger significance level, the process would yield higher percentages of correct identification of the true model. However, the mixing proportions does not show a clear pattern in terms of influence on the power.

(d) Their tables are clear in terms of presenting the simulation results, with all information and conditions for the simulation shown on the table. An alternative suggestion of presenting the table would be to separately report the results of different normal levels, so that it is easier to compare result for the same normal level.

question 2

```r
library(RSQLite)
drv <- dbDriver("SQLite")
dir <- "/Users/yihuan/Desktop"
dbFilename <- 'stackoverflow-2016.db'

db <- dbConnect(drv, dbname = file.path(dir, dbFilename))

get_tb <- dbGetQuery(db, "SELECT userid FROM
        questions Q, questions_tags T, users U
        WHERE Q.questionid = T.questionid AND
        Q.ownerid = U.userid
        AND tag LIKE '%apache-spark%'
        AND userid NOT IN (SELECT userid FROM
        questions Q, questions_tags T, users U WHERE
        Q.questionid = T.questionid AND
        Q.ownerid = U.userid
        AND tag LIKE '%python%')")

count <- dbGetQuery(db, "SELECT COUNT(DISTINCT userid) FROM
        questions Q, questions_tags T, users U
        WHERE Q.questionid = T.questionid AND
        Q.ownerid = U.userid
        AND tag LIKE '%apache-spark%'
        AND userid NOT IN (SELECT userid FROM
        questions Q, questions_tags T, users U WHERE
        Q.questionid = T.questionid AND
        Q.ownerid = U.userid
        AND tag LIKE '%python%')")
head(count)
```

```
##   COUNT(DISTINCT userid)
## 1                   4647
```

question3

By Economics knowledge, the recession began in 2008. Specifically, 2008-10-24 was famously named the "Bloody Friday" due to the recession. By wikipidia, on 2008-10-24, "many of the world's stock exchanges experienced the worst declines in their history, with drops of around 10% in most indices". Furthermore, on 2008-12-1, the National Bureau of Economic Research officially announced that "The US economy has been in recession since December 2007". Therefore, we would like to see if the number of searches appearing in wikipidia conforms with these "big days".

```
#get data from spark
ssh yihuan_song@hpc.brc.berkeley.edu
srun -A ic_stat243 -p savio2 --nodes=4 -t 1:00:00 --pty bash
module load java spark/2.1.0 python/3.5
source /global/home/groups/allhands/bin/spark_helper.sh
spark-start
pyspark --master $SPARK_URL --conf "spark.executorEnv.PYTHONHASHSEED=321"  --executor-memory 60G
dir = '/global/scratch/paciorek/wikistats_full'
lines = sc.textFile(dir + '/' + 'dated')
import re
from operator import add
# find all searches appeared as "recession", credit to Chris's demo for Spark
def find(line, regex = "Recession", 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)
recession = lines.filter(find)
# sum the number of hits by the "date-time-language" key
def stratify(line):
    vals = line.split(' ')
    return(vals[0] + '-' + vals[1] + '-' + vals[2], int(vals[4]))
counts = recession.map(stratify).reduceByKey(add)

# undo the transformation on the key
def transform(vals):
    key = vals[0].split('-')
    return(",".join((key[0], key[1], key[2], str(vals[1]))))
# get and store the result
mydir = '/global/scratch/yihuan_song'
outputDir = mydir + '/' + 'recession-counts'
counts.map(transform).repartition(1).saveAsTextFile(outputDir)
scp -r yihuan_song@dtn.brc.berkeley.edu:/global/scratch/yihuan_song/recession-counts/ ~/Desktop/
```

After we got the recession data from spark, we could then input the data into r and analyze if the "big days" (2018-10-24 and 2018-12-01) caused more searches for "recession" on wikipidia. From the analysis using r, we can see that the plot showed two peaks in dates near 2018-10-24 and 2018-12-01. Furthermore, by ordering the data by number of searches, we found that indeed, the most searches for "recession" appeared on 2018-10-24, and 2018-12-01 and the following day were also in the dates that had the top hits.

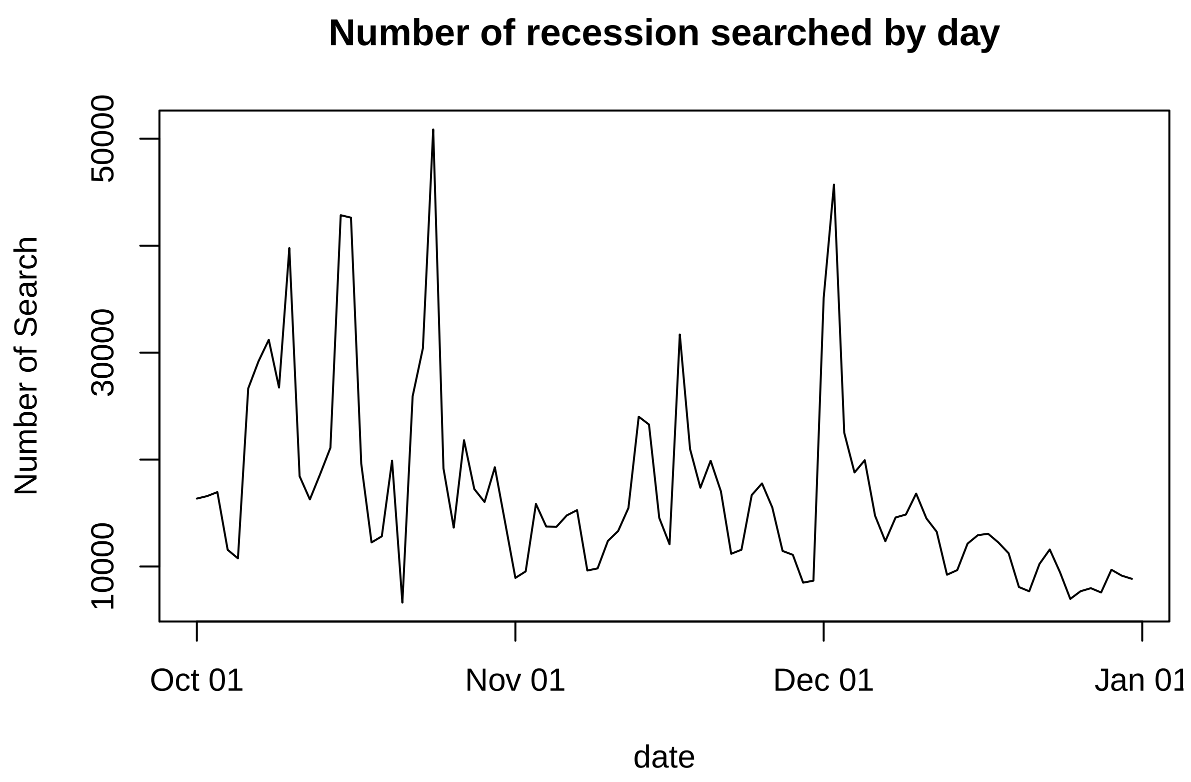
```
library(stringr)
library(dplyr)
```

```
##
## 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
```

```
#read in processed data
recession <- read.csv("~/Desktop/recession-counts/part-00000", header = FALSE)
names(recession) <- c("date", "time", "language", "search")
#sum by date
recession$date <- as.Date(as.character(recession$date), "%Y%m%d")
SearchByDay <- recession %>% group_by(date) %>% summarise(searchperday = sum(search))
#plot the searches by day to see a trend
plot(SearchByDay$date, SearchByDay$searchperday, type = "l", main = "Number of recession searched by da
     xlab = "date", ylab = "Number of Search")
```



```r
#find out the dates containing most searches
ordered <- SearchByDay %>% arrange(desc(searchperday))
head(ordered)
```

```
## # A tibble: 6 x 2
##   date       searchperday
##   <date>            <int>
## 1 2008-10-24        50862
## 2 2008-12-02        45713
## 3 2008-10-15        42846
## 4 2008-10-16        42617
## 5 2008-10-10        39772
## 6 2008-12-01        35115
```

question 4

(a) First, we use srun to set up savio such that we run our job on a single Savio node in the savio2 partition. Then, we write R code and run interactively to filter to only the rows where "Barack_Obama" appears. We used srun and run the code interactively in R, using a quarter of the files(240 files). It took approximately 30 minutes to complete the job for 240 files. Therefore, to complete the job for the full 960 files, assuming the time scales accordingly, we would need approximately 2 hours.

```
#start savio and run R interactively
ssh -Y SAVIO_USERNAME@hpc.brc.berkeley.edu
srun -A ic_stat243 -p savio2  --nodes=1 -t 1:00:00 --pty bash
module load r r-packages
R
```

R code

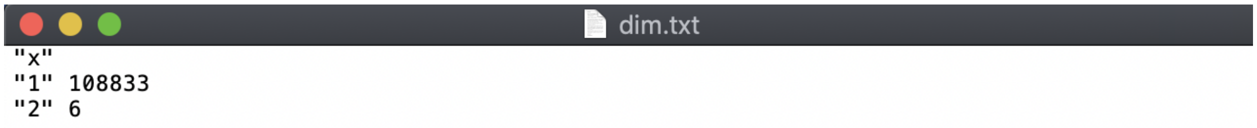
```r
library(parallel)
library(foreach)
library(doParallel)

#start cluster
registerDoParallel(as.numeric(Sys.getenv("SLURM_CPUS_ON_NODE")))
numFile <- 960/4 - 1
#filter rows containing Obama for each file
find <- foreach( f = 0:numFile,
                 .packages = c("readr","dplyr"),
                 .combine = rbind,
                 .verbose = TRUE) %dopar% {
                   path <- paste0("/global/scratch/paciorek/wikistats_full/dated_for_R/part-",
                                  formatC(f, width = 5, format = "d", flag = "0"))
                   wiki <- read_delim(path, delim = " ", col_names = FALSE)
                   obama <- wiki[grep("Barack_Obama", wiki$X4), ]
                   obama
                 }
#export the result
write.table(dim(find), file = "/global/scratch/yihuan_song/obama/dim.txt")
write.table(find, file = "/global/scratch/yihuan_song/obama/obama.txt")

stopImplicitCluster()

#scp -r yihuan_song@dtn.brc.berkeley.edu:/global/scratch/yihuan_song/obama/ ~/Desktop/
```

As shown in the screenshot, the result of our r code returned 108833 observations for 240 files. Therefore, we would get approximately 435,332 records using our R code, which is close to the result by Pyspark in the demo, which counted 433k observations for full dataset.



```
"x"
"1" 108833
"2" 6
```

(b) To get the filtered dataset that is only Obama-related on a single Savio node using parallelized R, we would use approximately 2 hours in total. Assuming perfect scalability, we would use approximately 30 minutes if we parallelize R across 4 nodes (96 cores). Since it takes about 15 minutes using 96 cores to create the filtered dataset using PySpark, using parallelize R would take approximately twice as long as using PySpark. Therefore, using Pyspark would be more efficient in terms of how long it takes to do the filtering.