Problem Set 6

Dongyu Lang
SID: 24174288

October, 29, 2017

Question 1

Question 1
The goal of the simulation is to test that the likelihood ratio statistic based on the Kullback-Leibler information criterion of the null hypothesis that a random sample is drawn from a ko-comnponent normal mixture distribution against the alternative hypothesis that the sample is drawn from a k1-component normal mixture distribution is asymptotically distributed as a weighted sum of independent chi-squared random variables with one degree of freedom, under general regularity conditions.[1]
The metric that they consider is to compare 2LR and the adjusted 2LR versus the nominal level under normal distribution.
Question 2
The choices that the authors have are 2LR method and the adjusted 2LR method, mixing porportion, sample size and the value of D.
The key aspects that are likely affect the statistical power of the test are mixing porportion, sample size and the value of D.
The authors fail to consider the tests with two different mixing porportions.
Question 3
The tables do a good job of presenting the simulation study because they show all the results they want to present and are easy to read and compare between different parameters.
However, my alternative suggestion is that since that the Nominal levels are 0.01 and 0.05, but 2LR and adjusted 2LR are in percentage, it is better to make the nominal levels to be consistent with 2LR and the adjusted 2LR.
Question 4
Table 2 shows that the powers are low for D equals 1 and 2, and powers are reasonable for sample size that is greater than 100. In addition, the there is no obvious evidence that the results differ with different mixing proportions.
Table 4 shows that the test exhibits low power for sample size less than 200 with $D_1 \leq 4$ and $D_2 \leq 2$ and with $D_1 \leq 2$ and $D_2 \leq 4$. Thus the two tables have some similarity. Thus, the results make sense.
Question 5

1000 simulations are enough in my opinion, while 10 sre not enough, because under 1000 simulations, the results make sense and are consistent according to the tables.

Reference
1. Lo, Yungtai, Nancy R. Mendell, and Donald B. Rubin. "Testing the Number of Components in a Normal Mixture." Biometrika 88, no. 3 (2001): 767-78. http://www.jstor.org/stable/2673445.

Question 2

```
## Read the database
drv <- dbDriver("SQLite")
db <- dbConnect(drv, dbname = file.path("2016.db"))
r_not_python = dbGetQuery(db, "select distinct Q.ownerid, U.displayname
          from questions Q, questions_tags T, users U where
          Q.questionid = T.questionid and Q.ownerid =
          U.userid and T.tag = 'r' and Q.ownerid not in
          (select distinct Q1.ownerid from
          questions Q1,questions_tags T1 where
          Q1.questionid = T1.questionid and T1.tag = 'python'
          and Q1.ownerid is not NULL)")
head(r_not_python)
```

```
##   ownerid     displayname
## 1  575952      Thalecress
## 2 5738949         AMedina
## 3 4802680 Dhwani Dholakia
## 4 3507767     user3507767
## 5 2670641   Reuben Mathew
## 6 4148256            Jake
```

```
## Number of unique users
dim(r_not_python)[1]
```

```
## [1] 18611
```

Question 3

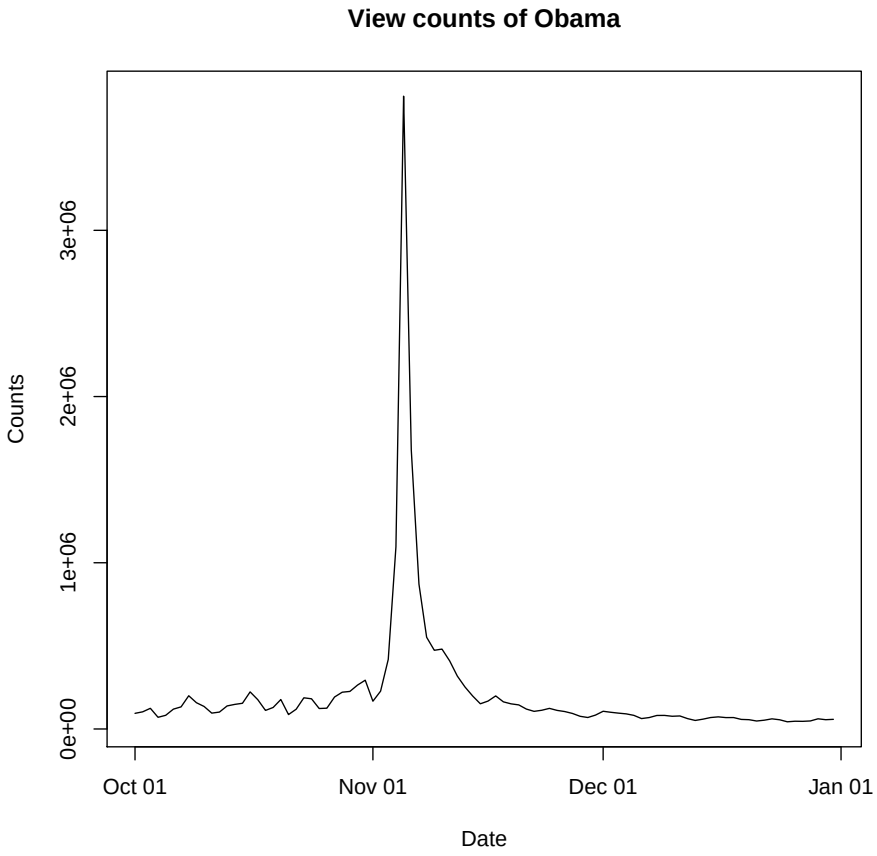
I first use the same method presented in class to get the data file that contains all the information about view counts of Obama using Savio.

```
dir = "/global/scratch/paciorek/wikistats_full/dated"
lines = sc.textFile(dir)
```

```
import re
from operator import add
def find(line, regex = "Barack_Obama", 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)
obama = lines.filter(find).repartition(480)
obama.count()

def stratify(line):
  vals = line.split( )
  return(vals[0] + - + vals[1] + - + vals[2], int(vals[4]))

counts = obama.map(stratify).reduceByKey(add)

def transform(vals):
  key = vals[0].split(-)
  return(",".join((key[0], key[1], key[2], str(vals[1]))))

outputDir = "/global/home/users/yaleldy"
counts.map(transform).repartition(1).saveAsTextFile(outputDir)
```

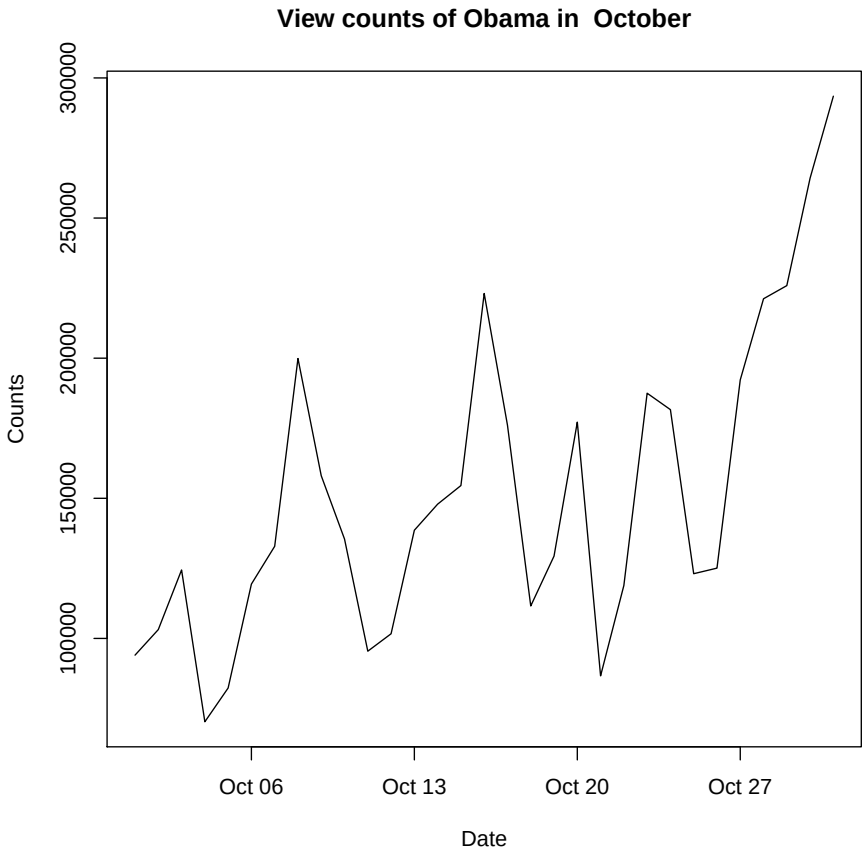
And I want to extend the Obama analysis, and take a look at the difference of view counts between English speaking countries and non-English speaking countries.
Let us first take a look at the overall counts of all the people.

```
## Read the data
rawdata = read.csv(file = "part-00000",header = F)
## Aggregate the data by Date
group_by_date = aggregate(rawdata[c("V4")], by = list(rawdata$V1),FUN = sum)
group_by_date$Group.1 = as.Date(as.character(group_by_date[,1]),format = "%Y%m%d")
## Plot the the view counts vs. Date
plot(group_by_date$Group.1,group_by_date$V4,type = "l",
     ylab="Counts",xlab = "Date", main = "View counts of Obama")
```



The graph is interesting that the counts are relatively stable except for the peak between Nov. 04 to Nov. 06. And, Obama was elected as the President on Nov. 04. Thus, it explains why there is a peak in that period.
Let's also analyze the counts in only October.

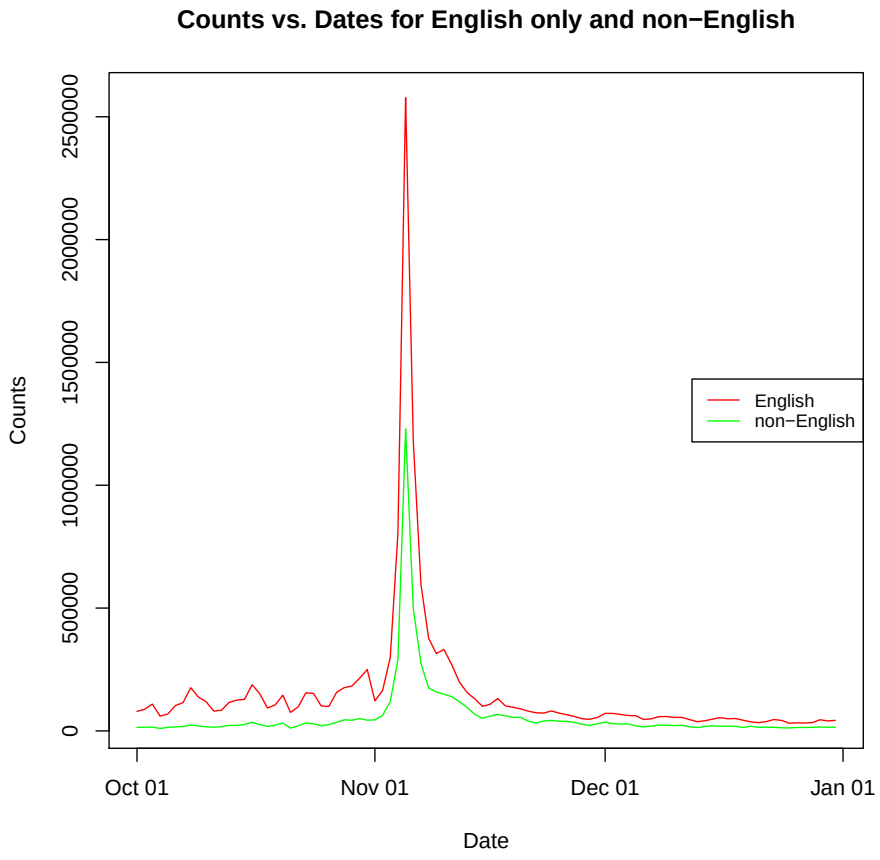
```
plot(group_by_date$Group.1[1:31],group_by_date$V4[1:31],type = "l",
     ylab="Counts",xlab = "Date", main = "View counts of Obama in  October")
```



From the graph, we can notice that there are three peaks before the mid October, these are around Oct 02, Oct 07 and Oct 15. And these are the dates for the vice-presidential debate, second presidential debate and final presidential debate repectively. And after the mid October, we can notice that there is a steady increase in counts, which might be the results of the debates.
Now let's take a look at the counts between that the language is English and non-English.

```
## Aggregate the data by languages and dates
by_lang = aggregate(rawdata[c("V4")], by = list(rawdata$V3,rawdata$V1),FUN = sum)
by_lang$Group.2 = as.Date(as.character(by_lang[,2]),format = "%Y%m%d")
## Select the data that the language is English
eng = by_lang[by_lang$Group.1 == "en",]
## Select the data that the language is non-English and aggregate them.
non_eng = by_lang[by_lang$Group.1 != "en",]
non_eng = aggregate(non_eng[c("V4")], by = list(non_eng$Group.2), FUN = sum)
plot(eng$Group.2,eng$V4,type = "l", col="red",xlab="Date",
```

```
     ylab="Counts",main = "Counts vs. Dates for English only and non-English")
lines(non_eng$Group.1,non_eng$V4, col="green")
legend("right", legend=c("English", "non-English"),
       col=c("red", "green"), lty=1:1, cex=0.8)
```



We can notice that Counts that the language is English is consistently above the non-English counts. The reason might be that English speaking countries care more about the election, especially for U.S citizens. In addition, some non-English speaking countries do not use Wikipedia, for example China, which has the world most population. Thus, in reality, for those who do not speak English, the counts might be higher.

Question 4(a)

I run the code in Savio interactively, and save and transfer the results to my local machine.

```
library(readr)
library(parallel)
library(doParallel)
library(plyr)

## Use 24 cores
nCores = 24
registerDoParallel(nCores)

## Calculate the run time for the parallel.
runtime = system.time(
  result <- foreach(i=0:239) %dopar% {
    ## Create the file names
    filename = paste("/global/scratch/paciorek/wikistats_full/dated_for_R/part-",
                  formatC(000+i,width = 5,flag = "0"),sep = "")
    ## Read the file
    rawdata = read_delim(file = filename, delim = " ", col_names = F)
    ## Get the indices that contain Barack Obama
    index = grep(".*(B|b)(A|a)(R|r)(A|a)(C|c)(K|k).*(O|o)(B|b)(A|a)(M|m)(A|a).*",
                 rawdata$X4)
    ## Return a dataframe that contains only Barack Obama
    obama_rawdata=rawdata[index,][c("X1","X2","X3","X5")]
    obama_rawdata
  })
## Combine a list of dataframes into a dataframe
result_combine = ldply(result,data.frame)
## Save the result
save(result_combine,file="/global/home/users/yaleldy/rparallel.Rda")
save(runtime,file="/global/home/users/yaleldy/runtime.Rda")
```

```
## Read the results saved from Savio
load("rparallel.rda")
head(result_combine)

##         X1     X2 X3 X5
## 1 20081129 210000 pt 86
## 2 20081014 190000 en  2
## 3 20081108 190000 no  1
## 4 20081128 190001 en 16
## 5 20081106 090000 en  1
## 6 20081110 160000 et  4

dim(result_combine)

## [1] 145653      4
```

```
load("runtime.rda")
head(runtime)

##  user.self   sys.self    elapsed user.child  sys.child
##      0.274      0.134   1041.434  12913.944   8380.350
```

The time for processing a quarter of the files, that is 240 files, is 1041 seconds; thus, the time to run full data set is four times of 1041 seconds, that is 4164 seconds, which is about 70 minutes.

Question 4(b)

Suppose we can assume perfect scalability, then we can save the time by four times, that is we only need to use $\frac{4164}{4} = 1041 seconds = 17.35 minutes$. Thus, it takes a bit longer than using Spark.

Question 4(c)

I used basically the same method as part a, except that I added a preschedule in the foreach.

```
library(readr)
library(parallel)
library(doParallel)
library(plyr)
library(doMC)

nCores = 24
registerDoMC(nCores)

mcoptions <- list(preschedule=TRUE)
runtime_pre = system.time(
  ## Add a preschedule in the foreach
  result <- foreach(i=0:239, .options.multicore=mcoptions) %dopar% {
    filename = paste("/global/scratch/paciorek/wikistats_full/dated_for_R/part-",
                     formatC(000+i,width = 5,flag = "0"),sep = "")
    rawdata = read_delim(file = filename, delim = " ", col_names = F)
    index = grep(".*(B|b)(A|a)(R|r)(A|a)(C|c)(K|k).*(O|o)(B|b)(A|a)(M|m)(A|a).*",
                 rawdata$X4)
    obama_rawdata=rawdata[index,][c("X1","X2","X3","X5")]
    obama_rawdata
  })
result_combine_pre = ldply(result,data.frame)
```

```
save(result_combine_pre,file="/global/home/users/yaleldy/rparallelpre.Rda")
save(runtime_pre,file="/global/home/users/yaleldy/runtimepre.Rda")
```

```
load("rparallelpre.rda")
head(result_combine_pre)

##         X1     X2 X3 X5
## 1 20081129 210000 pt 86
## 2 20081014 190000 en  2
## 3 20081108 190000 no  1
## 4 20081128 190001 en 16
## 5 20081106 090000 en  1
## 6 20081110 160000 et  4

dim(result_combine_pre)

## [1] 145653      4

load("runtimepre.rda")
head(runtime_pre)

##  user.self   sys.self    elapsed user.child  sys.child
##      0.251      0.210   1473.027  14204.053  17362.531
```

Interestingly, The time when using prescheduling is higher than without it. It takes the job 1473 seconds to finish conpared with 1041 seconds when not using prescheduling.

Question 5(a)

First, we consider the divisions. we have $(n-1)$ divisions in the first row. For i from row 2 to row n, we have (n-i) divisions in each row; thus , the total is $(n-1)+\sum_{i=2}^{n}(n-i)=\frac{n^2-n}{2}$.
Now, we consider the multiplication, for i in each row from 2 to n, the multiplications have $(i-1)+(n-i)(i-1)$; thus $\sum_{i=2}^{n}((i-1)+(n-i)(i-1))=\frac{n^3-n}{6}$
Thus, in total, the computational complexity is $\frac{n^3+3n^2-4n}{6}$

Question 5(b)

I think we are safe to store the Cholesky upper triangular matrix, because we can always get our original matrix by using $U^TU=A$ as long as A is positive definite.

Collaboration

In this problem set, I worked with Gaowei Chen.