STAT 243 PS 6

Junyuan Gao(SID:26484653)

November 1, 2017

1 Q1

1. Goal: Simulation studies were conducted to investigate the finite sample properties of the test.(More specifically, to evaluate the rate of convergence of 2LR to its limiting distribution)

Metrics: p-values and powers computed from formula (14) and its adjusted form for different sample sizes

2. (a)Authors have to make choices about sample size, initial values of parameters, and some other configurations. Moreover, they have to decide what is null hypothesis and what is alternative hypothesis.

(b)The key aspect of data generating mechanism are values of parameters and the sample size. If his initial parameters in hypothesis are very different from parameters of true distribution, it will not lead to a good conclusion and the statistical power will be highly affected. If the sample size is too small, the result may not converge to that in the true distribution.

(c)One of the useful data generating scenarios that the author neglected is the set of random seed. It's better and more affective for him to set a fixed seed for multiple experiments since it will reduce the randomness.

3. Yes, I think their tables did a very good job in simulation results presentation. The tables are neat and clear, and makes authors easy to find a power value under different mixing proportion, sample size, α level and D value.

4. First, introduce the meaning of each entry in table 2 and 4:
For example, the left upper entry 1.8 of table 2, it means that when D=1, sample size=50, mixing proportion= 0.5 and nominal level=0.01, the power is 1.8% (i.e. say 1000 draw from mixed distribution, 18 of 1000 test statistics are rejected under alternative hypothesis.)

(a)for single normal v.s. two component normal mixture(Table 2):
There is no strong evidence that the power depends on the mixing proportion. Generally speaking, the simulated power of unadjusted test is larger than that of adjusted tests
simulated power increase as D increase/alpha increase/sample size increase
simulated power decrease as D decrease/alpha decrease/sample size decrease

(b)for mixture of two normals v.s. mixture of three normals(Table 4):
There is no strong correlation between sample size and power
Generally speaking, the simulated power of unadjusted test is larger than that of adjusted tests
simulated power increase as D1+D2 increase/alpha increase
simulated power decrease as D1+D2 decrease/alpha decrease

5. (a)I think 1000 simulation is appropriate:not computational intensive, and the sample size will not cause huge bias and variability. However, in today's situation, 10000 simulation sounds better since it's also computational available and will result in a better approximation to the true distribution.

(b)10 simulation is not enough since it will cause variablity, unconverge bias and voluntary response bias.

(c) Method 1: Use the formula of calculating sample size: $n = (\sigma * Z(\alpha/2)/d)^2$, where σ is standard error, α=Pr(Type I error), d= 2*(length of Confidence Interval).

Method 2: Increase the sample size and draw a curve of changing rate of parameters versus sample size. When the changing rate tends to be very slow, then stop and choose that sample size.

2 Q2

In this question, I use two left outer joins to generate the tables together, and use a except statement to satisfy the requirements(only R related questions and no Python related questions)

```
library(RSQLite)
drv <- dbDriver("SQLite")
dir <- 'D:/exam/STAT243/ps/ps6' # relative or absolute path to where the .db file
dbFilename <- 'stackoverflow-2016.db'
db <- dbConnect(drv, dbname = file.path(dir, dbFilename))

result <- dbGetQuery(db, "select distinct userid, displayname
           from questions_tags QT left outer join questions Q
           on QT.questionid= Q.questionid left outer join users U
           on Q.ownerid= U.userid
           where QT.tag= 'r' except
           select distinct userid, displayname
           from questions_tags QT left outer join questions Q
           on QT.questionid= Q.questionid left outer join users U
           on Q.ownerid= U.userid
           where QT.tag= 'python'
           ")
dim(result)

## [1] 18611     2
```

```
head(result, 10)
```

```
##    userid    displayname
## 1     357         Seibar
## 2     740      SQLMenace
## 3    1428       lindelof
## 4    1968 Konrad Rudolph
## 5    6632  Craig Francis
## 6    6722   Steve Cooper
## 7    9435        oneself
## 8   13271            kes
## 9   17216        Mikhail
## 10  23929  Dan Goldstein
```

3 Q3

In this question, I used a similar methodology to the Obama analysis, and my question is: What is the trend of Number of hits per day of Taylor Swift from 10/01/2008 to 1/01/2009?

```
srun -A ic_stat243 -p savio2 --nodes=4 -t 2:00:00 --pty bash
module load java spark
source /global/home/groups/allhands/bin/spark_helper.sh
spark-start
## note the environment variables created
env | grep SPARK

# PySpark using Python 2.6.6 (default Python on Savio)
module unload python
pyspark --master $SPARK_URL --executor-memory 60G

################################################################################
dir = /global/scratch/paciorek/wikistats_full
### read data and do some checks ###
lines = sc.textFile(dir + / + dated)
lines.getNumPartitions() # 16800 (480 input files) for full dataset
# note delayed evaluation
lines.count()


### filter to sites of interest ###
import re
from operator import add
def find(line, regex = "Taylor_Swift", 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)


taylor = lines.filter(find).repartition(480) #  18 minutes for full dataset (but remember la
taylor.count() #  observations for full dataset

################################################################################

### map-reduce step to sum hits across date-time-language triplets
###
def stratify(line):
# create key-value pairs where:
# key = date-time-language
# value = number of website hits
  vals = line.split( )
  return(vals[0] + - + vals[1] + - + vals[2], int(vals[4]))

# sum number of hits for each date-time-language value
counts = taylor.map(stratify).reduceByKey(add) # 5 minutes
# 128889 for full dataset

### 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], str(vals[1]))))

### output to file ###
# have one partition because one file per partition is written out
outputDir = '/global/home/users/junyuangao/Taylor'
counts.map(transform).repartition(1).saveAsTextFile(outputDir) # 5 sec.

################################################################################
scp junyuangao@dtn.brc.berkeley.edu:/global/home/users/junyuangao/Taylor/part-00000 /mnt/d/e
```

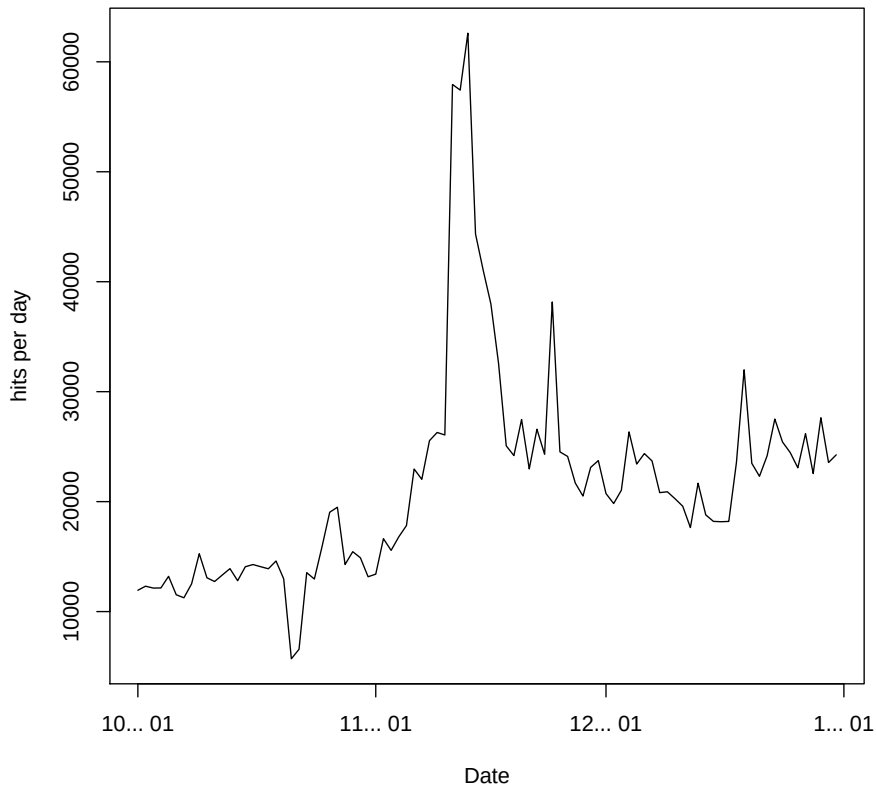
After download the data, group by date to count the hits of Taylor Swift per day, and make a plot of it.

```r
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

Taylor_data <- read.csv("part-00000", header = F, na.strings = "",stringsAsFactors = F)
names(Taylor_data) <- c("Date", "Time", "Language", "Number_of_Hits")
Taylor_data$Date <- as.Date(as.character(Taylor_data$Date),"%Y%m%d")
dates <- Taylor_data %>% group_by(Date) %>% summarize(hits_per_day = sum(Number_of_Hits))

plot(dates$Date, dates$hits_per_day, type="l", xlab="Date", ylab="hits per day")
```



4 Q4

Corporated with Shan Gao and Yachuan Liu

4.1 a

Commands to be executed on Savio terminal

```
scp /mnt/d/exam/STAT243/ps/ps6/ps6q4.R junyuangao@dtn.brc.berkeley.edu:
  /global/home/users/junyuangao/ps6q4.R
scp /mnt/d/exam/STAT243/ps/ps6/ps6q4.sh junyuangao@dtn.brc.berkeley.edu:
  /global/home/users/junyuangao/ps6q4.sh

module load r/3.2.5
module load stringr dplyr foreach doParallel
```

```
sbatch ps6q4.sh
squeue -j
squeue --start -j
squeue -u junyuangao
squeue -A ic_stat243
```

R script to be run on savio

```
library(parallel)
library(doParallel)
library(foreach)
# Windows users have to convert to integer or else it thinks this variable
# is vector of host names
ncores <- as.integer(Sys.getenv("SLURM_CPUS_ON_NODE"))
registerDoParallel(ncores)

nSub <- 959

obama <- foreach(i=0:nSub,
                 # libraries to load onto each worker
                 .packages = c("readr", "stringr", "dplyr"),
                 .combine= rbind,
                 .verbose=TRUE) %dopar% { # print statuses of each job
filepath= paste("/global/scratch/paciorek/wikistats_full/dated_for_R/part-",
        str_pad(i, width=5, side="left", pad="0"), sep="")
                data = readr::read_delim(filepath, delim=" ", col_names = FALSE)
                data= as.data.frame(data)
                # extract lines with Barack_Obama
                index= grep("Barack_Obama", data$X4)
                clean_data = data[index, ]
                 }
#some sanity check and write results to txt files
#then use scp to download it to local PC
dimension <- dim(obama)
first_lines <- head(obama)
write.table(dimension, file='/global/home/users/junyuangao/dimension.txt')
write.table(first_lines, file='/global/home/users/junyuangao/firstlines.txt')
write.table(obama, file='/global/home/users/junyuangao/obama.txt')
```

Shell script(ps6q4.sh) to process the job

```
#!/bin/bash
# Job name:
#SBATCH --job-name=OBAMA_q4
#
```

```
# Account:
#SBATCH --account=ic_stat243
#
# Partition:
#SBATCH --partition=savio
#
# Wall clock limit (2 hr here):
#SBATCH --time=02:00:00
#
## Command(s) to run:
module load r/3.2.5
R CMD BATCH --no-save ps6q4.R p6q4_output.out
```

Since the result txt file is too large(about 6GB), I download the information and first several lines about the result.

```
dimension<- read.table("dimension.txt", header = TRUE)
firstlines <- read.table("firstlines.txt", header=TRUE)
dimension
```

```
##        x
## 1 430160
## 2      6
```

```
firstlines
```

```
##              X1     X2 X3
## 14046  20081129 210000 pt
## 68106  20081014 190000 en
## 77377  20081108 190000 no
## 90408  20081128 190001 en
## 117199 20081110 160000 et
## 160067 20081101 110000 fr
##
## 14046
## 68106  Special:AllPages/I_ran_Project_Vote_voter_registration_drive_in_Illinois,_ACORN_wa
## 77377
## 90408
## 117199
## 160067
##        X5      X6
## 14046  86 2032215
## 68106   2   25520
## 77377   1    7825
## 90408  16  760462
## 117199  4   55875
## 160067  1   20922
```

4.2 b

Download .out file from Savio and find the processing time at the end of the file

```
> proc.time()
    user   system  elapsed
51404.21 56905.97  6191.53
```

From the above proc.time(), if the we can run code in part (a) 4 times faster, this will take about 25.8 minutes, which is 10 minutes slower than using PySpark(about 15 minutes). So using PySpark is more effective.

5 Q5

5.1 a

In the following chunk, we can see the operation chunks in each step(i.e. for each i, j= 1,...,n).

```
1. U_{11} = \sqrt{A_{11}}  # this step have no operation chunk
2. for j = 2:n
     U_{1j} = A_{1j}/U_{11}     #each step produce 1 operation count
3. for i = 2:n
     #each step produce i-1 operation counts
     U_{ii}= \sqrt{A_{ii}- \sum_{k=1}^{i-1} U_{ki}^2}
     for j= i+1: n
         # each step produce i-1+1= i operation counts
         U_{ij}= (A_{ij}- \sum_{k=1}^{i-1} U_{ki}U_{kj})/U_{ii}
```

From above illustration, we can generate a formula of total operation counts:

$$\text{total operation counts} = \sum_{i=2}^{n}((\sum_{j=i+1}^{n} i)+i-1)+1\times(n-1) = \sum_{i=2}^{n}(-i^2+i(n+1)-1)+n-1$$

$$= \frac{1}{6}(n^3+3n^2-10n+6)+n-1 = \frac{n^3+3n^2-4n}{6}$$

The result operation count is consistent to the order $n^3/6 + O(n^2)$.

5.2 b

You can store Cholesky upper triangular matrix U in the block of memory that is used for the original matrix since from the algorithm, we can see that when computing each U_{ij}, the only information we used from matrix A is entry A_{ij}. Thus, when we get U_{ij} value, we can store it in the memory that is used to store A_{ij} since this will not interupt the computing process.