Problem Set #6

Todd Faulkenberry

10/25/2018

Problem 1

Part A

The goals of the simulation study are to determine both the statistical and substantive significance of how quickly the test statistic of a sample drawn from a normal mixture distribution converges to asymptotic distribution. They use the test statistics from two different simulation studies – one vs two-component normal mixture and two vs three-component normal mixture – as their metrics.

Part B

The authors had to make many choices, among them sample size, number of starting components, separation distance, mixing proportion, number of repetitions, and nominal vs actual levels. Some of these aspects that likely impact the power are the sample size, number of components and separation. The authors did not find strong evidence that mixing proportion impacted power.

Part C

In Table 2, we have simulated powers for both unadjusted and adjusted tests for a normal distribution vs a two-component normal mixture. Each test has been replicated 1000 times. The numbers in the table represents the proportion of the simulation test statistics you rejected (e.g. 13.4 would be 13.4% of 134 of the 1000 replications.) Some of the results here make sense. For example, we see a very strong impact on power from separation distance. We do not see a large impact of power from sample size, however, which is something we'd expect to see.

In Table 4, we have a Table similar to Table 2 but with an added distance level. Here, we once again see the strong impact from separation distance. We also see, however, a more pronounced impact of sample size here as compared to Table 2.

Part D

I think the tables do a fine job of communication the data for those well versed in reading these types of statistics, but I would prefer some type of visualization to convey it. A visualization would communicate the important trends just as effectively and much more efficiently.

Problem 2

Here, we are asked to find how many unique users have asked Spark-related questions but not Python-related questions on StackOverflow. In order to do this, I connected to the database and then ran two separate queries to get unique ownerids for users that have ask, respective, spark and python questions. I then ran an anti-join on the two databases, which effectively gives me the users who have asked spark questions but not python ones. I then returned the number of rows of the resulting database.

```
## Connecting to database
drive <- dbDriver('SQLite')
dir <- '~/Downloads'
database <- 'stackoverflow-2016.db'

db <- dbConnect(drive, dbname = file.path(dir, database))

## Exploring tables in database
dbListTables(db)
```

```
## [1] "answers"        "questions"      "questions_tags" "users"
```

```
dbListFields(db, 'users')
```

```
##  [1] "userid"         "creationdate"   "lastaccessdate" "location"
##  [5] "reputation"     "displayname"    "upvotes"        "downvotes"
##  [9] "age"            "accountid"
```

```
dbListFields(db, 'questions')
```

```
## [1] "questionid"   "creationdate" "score"        "viewcount"
## [5] "title"        "ownerid"
```

```
dbListFields(db, 'questions_tags')
```

```
## [1] "questionid" "tag"
```

```
## Grabbing list of unique ownerids of those who have asked apache spark questions
apache_spark <- dbGetQuery(db, 'select DISTINCT ownerid from
questions Q, questions_tags T, users U WHERE
U.userid = Q.ownerid AND
T.questionid = Q.questionid AND
T.tag LIKE "apache%spark"')

## Grabbing list of unique ownerids of those who have asked python questions
python <- dbGetQuery(db, 'select DISTINCT ownerid from
questions Q, questions_tags T, users U WHERE
                          U.userid = Q.ownerid AND
                          T.questionid = Q.questionid AND
                          T.tag = "python"')

## Anti-join of apache-spark and python tables (returns values in table A that do not appear in table b
apache_no_python <- anti_join(apache_spark, python, by = 'ownerid')
nrow(apache_no_python)
```

```
## [1] 4583
```

The above code shows that 4583 unique users have asked spark questions but not python question on Stack Overflow.

Question 3

For this question, I decided to extend the searching for numbers of hits on the wikipedia page for the Second Amendment. While many people (myself included) were very excited for Obama's election victory, many people were not. I wanted to see if these people were as actively searching for information on Wikipedia as those who were excited about the election. I believe searches for Second Amendment is a good proxy for this

because many people who were afraid of Obama's election had consumed a lot of accusations around him being a tyrant who would "take your guns." My theory before doing the analysis was that searches of Second Amendment would also significantly increase around this time.

```
## Bash code

# Running interactive session
srun -A ic_stat243 -p savio2 --nodes=4 -t 3:00:00 --pty bash
module load java spark/2.1.0 python/3.5
source /global/home/groups/allhands/bin/spark_helper.sh
spark-start


spark-submit --master $SPARK_URL  $SPARK_DIR/examples/src/main/python/pi.py

# PySpark using Python 3.5 (Spark 2.1.0 doesn't support Python 3.6)
pyspark --master $SPARK_URL --executor-memory 60G \
        --conf "spark.executorEnv.PATH=${PATH}" \
        --conf "spark.executorEnv.LD_LIBRARY_PATH=${LD_LIBRARY_PATH}" \
        --conf "spark.executorEnv.PYTHONPATH=${PYTHONPATH}" \
        --conf "spark.executorEnv.PYTHONHASHSEED=321"
```

```
## Python code
# Directory to wikipedia data
dir = '/global/scratch/paciorek/wikistats_full'
# Read data
lines = sc.textFile(dir + '/' + 'dated')
import re
from operator import add
import numpy as np
import pandas as pd
# Function to find data
def find(line, regex = "Second_Amendment_to_the_United_States_Constitution", 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)
# Filter to sites on interest
second_amd = lines.filter(find)
# Functions to create DataFrame
def remove_partial_lines(line):
    vals = line.split(' ')
    if len(vals) < 6:
        return(False)
    else:
        return(True)
def create_df_row(line):
    p = line.split(' ')
    return(int(p[0]), int(p[1]), p[2], p[3], int(p[4]), int(p[5]))
tmp = second_amd.filter(remove_partial_lines).map(create_df_row)
## Creating dataframe
```

```
df = sqlContext.createDataFrame(tmp, schema = ["date", "hour", "lang", "site", "hits", "size"])
## Grouping and printing 100 rows, which I analyzed in R
df.groupBy('date').sum('hits').show(100)
```

I had trouble actually exporting my data, so I copied the output into a text file, which I then saved onto my desktopI then loaded that data into R, cleaned it, and analyzed it via the code below.

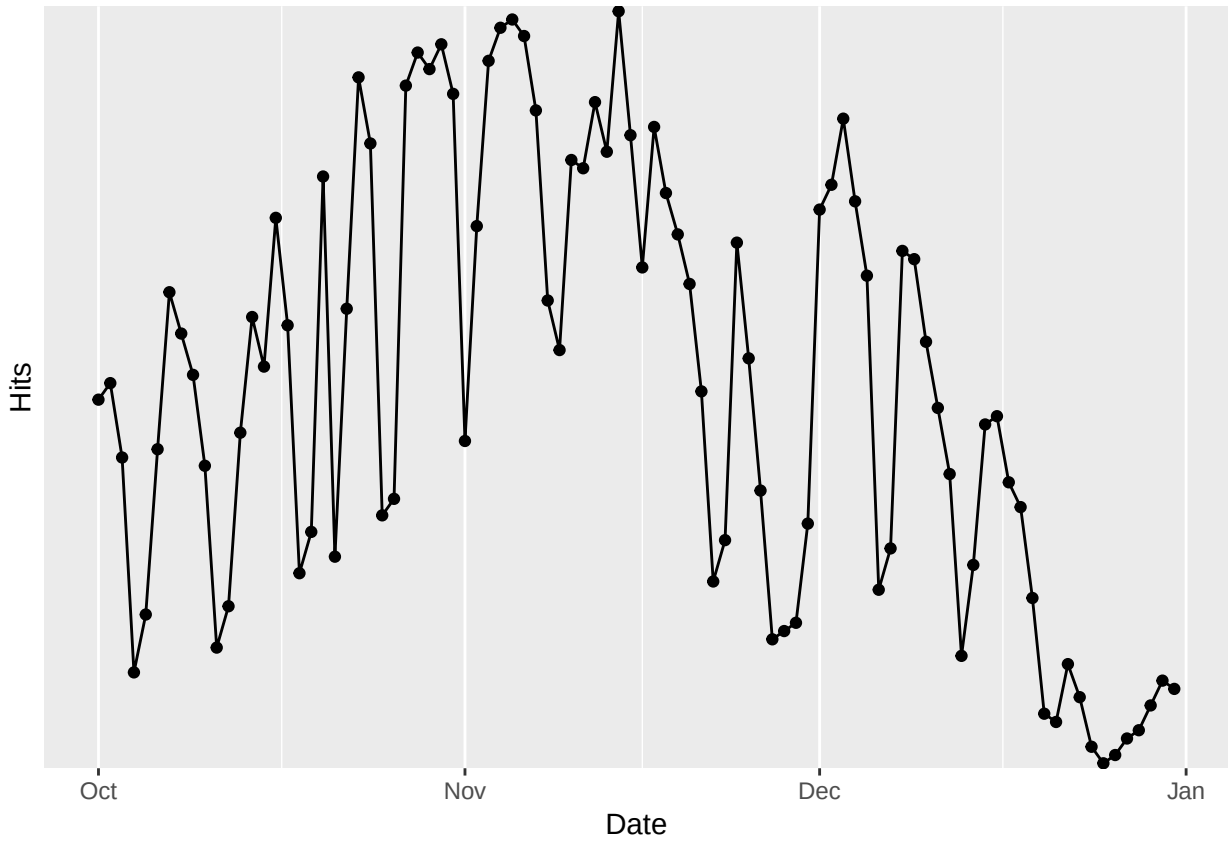
```
## R code
## Reading in txt file from desktop that I grabbed from Spark
Lines <- read_table("~/Desktop/results.txt")
```

```
## Parsed with column specification:
## cols(
##   `|    date|sum(hits)|` = col_character()
## )
```

```
Lines <- Lines[-1,]

# Separating dataframe into two columns
second_amd_df <- Lines %>%
  separate(1, c('Date', 'Hits'), sep = "\\| ")

# Cleaning up unnecessary characters from txt file
second_amd_df$Date <- gsub('\\|', '', second_amd_df$Date)
second_amd_df$Hits <- gsub('\\|', '', second_amd_df$Hits)

# Converting column to correct classes
second_amd_df$Date <- ymd(second_amd_df$Date)
as.integer(second_amd_df$Hits)
```

```
##  [1]  4205  4094  5125  2738  4821  7186  6427  6846  2644  5575  5226
## [12]  2319  5524  4530  4263  3530  2188  6441  6704  3194  1959  5600
## [23]  7809  5533  7340  4890  5511 17296 10828  1883  4914  1589  4970
## [34]  5500  1628  2759  3022  4944  9571  7569  1602  4644  2826  2205
## [45]  1415  3847  1994  4999  5312  5357  1430  5244  4243  5728  3374
## [56]  4494  4014  3082  8006  4600  9120  4749  4137  1040  6733  3659
## [67]  8507  6717  5242  3823  3998  2140  5247  4646  2642  6140  5015
## [78]  1180  2979  9117  5420  5986  7527 15822  1394  8850  2912  5942
## [89]  6810  7238  6371  3139
```

```
# Line graph of Hits over time
ggplot(data=second_amd_df, aes(x=Date, y=Hits, group = 1)) +
  geom_point() +
  geom_line() +
  scale_y_discrete(breaks = seq(0,20000,1000))
```




```
# Not sure why my y-axis isn't displaying correctly,
# but without scale_y_discrete, it displayed so many
# values that you couldn't even interpret the axis.
```

The y-axis in the above graph ranges from 1,000 to ~20,000. As we see in the graph above, there certainly was a surge in wikipedia searches around the Second Amendment around the time of the election. The peaks, however, are not sustained, and the the number of hits steadily decrease until a number in late December that is consistenly lower than the number of hits before the election. This suggests to me that the general had very little worry about Obama "taking their guns," and perhaps that the sentiment only developed later in his presidency, after a sustained message by conservative brass and media.

Problem 4

In the below code, I had trouble connecting to Spark via the spark_connect() call in my chunk of R code. I kept receiving the Java errors that Chris warned about, even after adjusting my memory across a wide range. It simply wasn't working. So, I never actually ran the code below that (meaning it may not be perfect.) I did write the code that I would have implemented, however, had I been able to connect to Spark.

```
# Running interactive session
srun -A ic_stat243 -p savio2 --nodes=4 -t 3:00:00 --pty bash
module load java spark/2.1.0 python/3.5
source /global/home/groups/allhands/bin/spark_helper.sh
spark-start

module load r r-packages
R
```

```
conf <- spark_config()
conf$spark.driver.memory <- "8G"
conf$spark.executor.memory <- "50G"

## Still working on getting connected to Spark (wouldn't work b/c of Java errors.)
sc <- spark_connect(master = Sys.getenv("SPARK_URL"), config = conf)

## How I would have formatted the data had I been able to connect to Spark.
cols <- c(date = 'numeric', hour = 'numeric', lang = 'character',
          page = 'character', hits = 'numeric', size = 'numeric')


## Turning into csv.
wiki <- spark_read_csv(sc, "wikistats",
                       "/global/scratch/paciorek/wikistats/dated",
                       header = FALSE, delimiter = ' ',
                       columns = cols, infer_schema = FALSE)

## Parallelized code to run through Wiki and grab all the pages that mention Barack Obama.

nCores <- 4
registerDoParallel(nCores)

wiki_plus <- foreach(i = nrow(wiki)) %dopar% {
                       spark_apply(wiki, function(data) {
                         data$obama = stringr::str_detect(data$page, "Barack_Obama")
                         data
}, columns = c(colnames(wiki), 'obama'))
}

obama <- collect(wiki_plus %>% filter(obama))
```