# ps6

*A.J. Torre*

*November 6, 2018*

**Comments: I am turning this in late due to a family emergency that caused me to be out of town from 10/22 to 10/31. I did not look at the solutions handed out in class or at any solutions of my classmates. I did ask some classmates for help (Malvika and Andrea) on the last question.**

## Questions

### Question 1: Read the first 3 pages and Section 3 of the Lo et al article.

#### a) What are the goals of their simulation study and what are the metrics that they consider in assessing their method?

This paper's goal is to determine number of components in a mixture distributio, which is an important but difficult problem. They propose their own method based on an extension of Vuong's theorem and use an empirical power study to look at their results.

#### b) What choices did the authors make in designing their simulation study? What are the key aspects of the data generating mechanism that likely affect the statistical power of the test? Are there data-generating scenarios that the authors did not consider that would be useful to consider?

In this paper, they discussed using an EM algorithm to get MLEs of the parameters with 100 sets of random starting values for the parameters.The p values were computed with both approximation error and truncation error, and they considered two cases with null and alternative hypothesis. Several different sample sizes considered. In terms of other ways to generate data, I looked online and found this article on how to generate data in R using the simstudy package https://www.r-bloggers.com/it-can-be-easy-to-explore-data-generating-mechanisms-with-the-simstudy-package/.

#### c) Interpret their tables on power (Tables 2 and 4)- do the results make sense in terms of how the power varies as a function of the data generating mechanism?

Table 2 is titled: Simulated powers, in percentages, of the adjusted and unadjusted tests for testing a single normal vs a two-component normal mixture based on 1000 replications under each alternative. Table 4 is titled: Simulated powers, in percentages, of the adjusted and unadjusted tests for testing a mixture of two normals versus a mixture of three normals based on 1000 replications under each alternative with the mixing proportions 0.3 and 0.4. So, both tables calculate the simulated powers, but the first one is two-component based on a 1000 replications and the second is two-components normals versus three-component normals. From Table 2, they found that there is no strong evidence that the power depends on the mixing proportion; power is very low for $n<200$ when the two components are not well separated. Overall, I do think the results make sense of based on their simulation/ data-generating mechanism.

#### d) Do their tables do a good job of presenting the simulation results and do you have any alternative suggestions for how to do this?

Overall, their tables do give extensive results, but also can be hard to understand exactly all that's going on since they used quite a few different sample sizes as well as different comparisons. They could perhaps graph these results so readers can more easily understand the results of this study. I'm personally somewhat unfamiliar with papers on this topic so I had to look up some outside/ online resources on data-generating mechanisms in order to understand this paper better.

### Question 2: SQL Question about StackOverflow

To solve this question, I first started by setting the database up, following the in-class notes from Unit 7.

```
# setting up the SQLite database
drv <- dbDriver("SQLite")
# set working directory here manually
dbFilename <- "stackoverflow-2016.db"
db <- dbConnect(drv, dbname = file.path(dbFilename))
```

To find how many users had questions about apache-spark but not about Python, I first joined the questions, users, and questions_tags tables together through creating a view:

```
# making a query to create a view to join the different
# tables together
dbGetQuery(db, "create view try_this_4 as select q.questionid, q.ownerid, u.userid, t.questionid,
           t.tag from questions q join users u on q.ownerid=u.userid join questions_tags t on
           q.questionid=t.questionid")
```

Next, to get which users asked about apache-spark but not Python, I used the wildcard symbol % for both apache-spark and Python and then used the "except" command in SQL to say that I want all the users who have asked a question about apache-spark except those that also have the Python tag. To get the number of how many users this is, we can just take the length the list we created from out query, which ends up being 4,647.

```
# code for overlapping SQL queries
final_results <- dbGetQuery(db, "select distinct userid from try_this_5 where tag like '%apache_spark%'
                          except select distinct userid from try_this_5 where tag like '%python%'")

# to get the number of users
length(final_results[[1]])
```

### Question 3: SPARK Question (opt out)

### Question 4: Wikipedia Traffic Data

#### a) Parallelized R code:

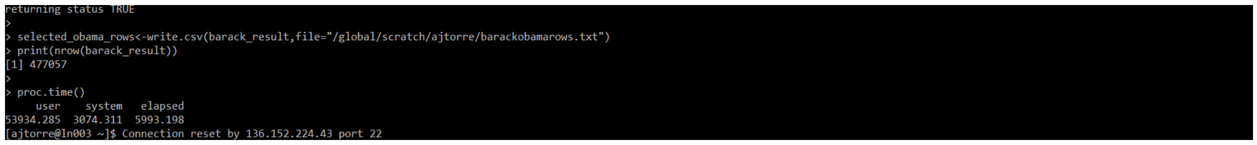
First, since we need to submit this job to Savio, I made a bash file using the vi editor in bash and followed in-lab example of how to create a .sh file to submit jobs.

```
#!/bin/bash
# Job name:
#SBATCH --job-name=obamajob
#
# Account:
#SBATCH --account=ic_stat243
#
```

```
# Partition:
#SBATCH --partition=savio2
#
# Number of tasks needed for use case:
#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 obamajob.R obamajob.Rout
```

To create the R code to run (the file obamajob.R), I tried to follow what was shown in lab 8, where the same question was asked, but just for a smaller data set of files. The idea is using list.files to create a character vector with all the names of files in the wikistats directory, then to make a function that takes in a file as input and looks to see in what articles Barack Obama is mentioned in, and then we use foreach (and dopar) to run the function on all of the files in the wikistats directory. Lastly, to get how many times he's mentioned, we just count the number of rows in our output.

```
# using the R wikistats data
pathf <- "/global/scratch/paciorek/wikistats_full/dated_for_R"

# list.files will give us a character vector of all the names
# of the files, which we want in order to use our foreach
# function later use regex for 'part' in order to collect all
# the wiki files
file_names <- list.files(pathf, pattern = "part*", full.names = TRUE)

# following in class example on how to set up libraries and
# cores
library(readr)
library(parallel)
library(doParallel)

nCores <- as.integer(Sys.getenv("SLURM_CPUS_ON_NODE"))
registerDoParallel(nCores)

# this function takes in a file name, uses read delim, and
# grep to get only the data where Barack_Obama is mentioned
get_obama_mentions <- function(file) {
    data <- read.delim(file, sep = "", header = FALSE, stringsAsFactors = FALSE)
    subset(data, grepl("Barack_Obama", data[, 4], ignore.case = TRUE))
    # use grepl bc we only care about the number of files that
    # mention Barack Obama so it's ok that this will only return
    # TRUE
}

# applying foreach to apply the above function to each file
# use .combine=rbind to avoid getting a list of lists as
```



```
returning status TRUE
>
> selected_obama_rows<-write.csv(barack_result,file="/global/scratch/ajtorre/barackobamarows.txt")
> print(nrow(barack_result))
[1] 477057
>
> proc.time()
    user    system   elapsed
53934.285  3074.311  5993.198
[ajtorre@ln003 ~]$ Connection reset by 136.152.224.43 port 22
```

Figure 1: Output of obamajob.Rout

```r
# output; use packages is 'readr' since our get_obama_mention
# function needs it in order to do read.delim
barack_result <- foreach(f = file_names, .packages = c("readr"),
    .combine = rbind, .multicombine = FALSE, .verbose = TRUE) %dopar%
    {
        output <- get_obama_mentions(f)
        output
    }

selected_obama_rows <- write.csv(barack_result, file = "/global/scratch/ajtorre/barackobamarows.txt")

# final step to return the number of results where Barack
# Obama is mentioned
print(nrow(barack_result))
```

Since I'm on a Windows device, I also used some scp commands in bash to copy my files from the Windows side to the Linux side and then again to my dtn.brc.berkeley.edu account. Then, from there, I used the below commands to submit and monitor my job.

```
sbatch obamajob.sh

squeue -j obamajob.sh

wwall -j obamajob.sh -t
```

It took this code about an hour and a half(more details in part b) to run. My final answer is that it Barack Obama is mentioned in 477,057 articles.

### b) Time of Code:

The process time that it took to run the above code is shown in Figure 1, using SLURM CPUS ON NODE to get the number of cores. So, the elapsed time is 5993.198/60, which is about 100 minutes so about an hour and a half. Discussing the timing results from classmates, they said their code took somewhere between 45 minutes and 2 hours to run, possibly depending on the number of cores or how the code was written. If we used 96 cores, where 12 (the number of SLURM CPUS ON NODE) times 8 is 96, we would expect it to go 8 times as fast as our previous code. So, 100/8 is 12.5. So, based on these results, I would say using parallelized R code might be more efficent in this case as it's a couple minutes faster than using Spark.