STAT 243: Problem Set 1

Jin Rou New

September 11, 2014

1 Problem 1

Complete.

2 Problem 2

(a) The number of (virtual) processors is 0.

```
# Check number of (virtual) processors
grep processor /proc/cpuinfo
```

(b) The RAM is 1017868 kB.

```
# Check RAM
grep MemTotal /proc/meminfo
```

3 Problem 3

(a) (i) I used the fact that values in each column in the CSV file begin and end with double quotation marks to extract rows with the relevant values using the `grep` command.

```
mkdir data
# Download CSV file
candidate_filepath="http://www.fec.gov/data/CandidateSummary.do?format=csv"
wget -O data/candidate_summary.csv $candidate_filepath

# Output data filtered for Senate and Republic/Democratic candidates
grep '"S",' data/candidate_summary.csv | grep '"REP",' > data/candidates_rep.csv
grep '"S",' data/candidate_summary.csv | grep '"DEM",' > data/candidates_dem.csv
```

(a) (ii) After pre-processing the CSV files, I first used `sort` to sort each of them by the general numeric reverse order based on field 16 `ind_con`, which gives the total contributions from individuals. I then used `head` to select the first 5 rows of the data, before using `cut` to extract select columns to print to screen.

```
function process_file() {
  local data_filepath="$1" ##<< Data file path

  # (Note: -e to execute multiple sed commands)
  sed -e 's/\([^",]\),/\1/g' \
    -e 's/[$"]//g' $data_filepath
}

process_file data/candidates_rep.csv > data/candidates_rep_final.csv
process_file data/candidates_dem.csv > data/candidates_dem_final.csv

function get_largest_contributions() {
 local data_filepath="$1" ##<< Data file path
 local num="$2" ##<< Number of candidates to print

 echo "The $num candidates with the largest total contributions from individuals are:"
 echo "Candidate ID,Name,Office State,Type,Total contributions from individuals ($)"
 # Sort file by contributions, select first num rows, then select columns to output
 sort -gr -t',' -k16 $data_filepath \
   | head -n $num \
  | cut -d',' -f2,3,5,8,16
}

# Output info to screen
get_largest_contributions data/candidates_rep_final.csv 5
get_largest_contributions data/candidates_dem_final.csv 5
```

```
## The 5 candidates with the largest total contributions from individuals are:
## Candidate ID,Name,Office State,Type,Total contributions from individuals ($)
## S2KY00012,MCCONNELL MITCH,KY,INCUMBENT,8991643.00
## S2TX00106,CORNYN JOHN,TX,INCUMBENT,6903804.00
## S4AR00103,COTTON THOMAS,AR,CHALLENGER,5558956.11
## S4LA00107,CASSIDY WILLIAM,LA,CHALLENGER,5186351.00
## S0SC00149,GRAHAM LINDSEY OLIN,SC,INCUMBENT,5104768.00
## The 5 candidates with the largest total contributions from individuals are:
## Candidate ID,Name,Office State,Type,Total contributions from individuals ($)
## S4MA00028,MARKEY EDWARD JOHN MR,MA,INCUMBENT,14668778.00
## S4NJ00185,BOOKER CORY A,NJ,INCUMBENT,14128197.00
## S8NC00239,HAGAN KAY R,NC,INCUMBENT,10456106.00
## S4KY00091,GRIMES ALISON  LUNDERGAN,KY,CHALLENGER,10290296.00
## S4GA11277,NUNN MARY MICHELLE,GA,OPEN,8016437.00
```

(b) and (c) I wrote a function to get the principal committee ID of a candidate given either just his/her last name, or his/her last name, followed by a comma and part or all of his/her first name. The latter serves to distinguish between candidates with the same last name but different first names. I first check if the user input is of the form [lastname] or [lastname, firstname]. For user input of the form [lastname], I add a comma before using `grep` to find the row containing that candidate's info. This step is necessary because if we omit it, we run the risk of selecting rows for candidates from Cookeville when we are searching for a candidate with last name Cooke, for instance.

Using `wc` allows me to count the number of rows with names matching the user input. If there is a unique match, the function will print the principal committee ID of the candidate. Otherwise, it will print an error message. If there is more than 1 match, all the matched candidate names will be printed to allow the user to check the first name of the candidate he/she is interested in. The user can then input [lastname, firstname] into the function to try once more to find the principal committee ID.

I used an `awk` function to increment the starting count of 0 if a row has field 1 `CMTE_ID` that matches the candidate's principal committee ID from the `get_PCN` function (and field 10 `STATE` with value `CA` for California).

(d) I used the same code as in 3(c), but added an extra condition to extract only rows with values in field 15 `TRANSACTION_AMT` of more than $200.

```
# Download FEC data on individual contributions to candidates and unzip files
commid_filepath="ftp://ftp.fec.gov/FEC/2014/cn14.zip"
indcont_filepath="ftp://ftp.fec.gov/FEC/2014/indiv14.zip"

wget -O data/cn14.zip $commid_filepath
unzip data/cn14.zip
mv cn.txt data/cn.txt
wget -O data/indiv14.zip $indcont_filepath
unzip data/indiv14.zip
mv itcont.txt data/itcont.txt
```

```
## Archive:  data/cn14.zip
##   inflating: cn.txt
## Archive:  data/indiv14.zip
##   inflating: itcont.txt
```

```
# Get principal committee ID/number given [last name] or
# [last name, part or all of first name] of candidate.
# Function outputs an message if there is no match
# for the candidate or if there are multiple matches.
function get_PCN() {
  local last_name="$1" ##<< String of form [lastname] or [lastname, firstname]
  local last_name_input

  # Check user input
  # Check if there is a comma in the user input,
  # indicating that the input is of the form [lastname, firstname]
  echo "$last_name" | grep -q ","
  if [ $? -eq 0 ] # Note: $? holds exit status of the previously executed command
  then
    last_name_input="|$last_name" # grep for input
  else
    last_name_input="|$last_name," # grep for input followed by a comma
  fi

  # Get the number of matches by counting the number
  # of lines containing last_name_input
  num_matches=$(grep -i "$last_name_input" data/cn.txt \
    | wc -l)

  # Output the principal committee ID if a unique match
  # is found, else output an error message
  if [ $num_matches -eq 1 ]
  then
    PCN=$(grep -i "$last_name_input" data/cn.txt \
    | cut -d'|' -f10)
  if [ $(echo "$PCN" | wc -w) -ne 0 ]
   then
    echo $PCN
    else
   echo "No principal committee ID available for candidate $last_name."
  fi
   elif [ $num_matches -eq 0 ]
   then
     echo "No matches found."
   else
     echo "More than 1 match found. Please input [lastname, firstname] to obtain a unique match\n"\
       "if your input was of the form [lastname]."
     grep -i "$last_name_input" data/cn.txt \
       | cut -d'|' -f2 # Display all matched names
   fi
}

# (b) and (c)
# Get number of individual contributions to candidate
# above $200 nationwide and for California given [last name] or
# [last name, part or all of first name]
function get_contributions_above200() {
  local last_name="$1" ##<< String of form [lastname] or [lastname, firstname]
  local PCN
  PCN=$(get_PCN "$last_name")

if [ $(echo "$PCN" | wc -w) -ne 1 ]
then
 # Print out error messages from get_PCN function if there is no unique match
 # In such a case, number of words returned from the get_PCN function will be
 # more than 1
  echo "$PCN"
else
   awk -v last_name="$last_name" -v PCN="$PCN" 'BEGIN {
   count = 0;
   count_CA = 0;
  }
  {
   if ($1 == PCN) {
   count++;
   if ($10 == "CA") {
   count_CA++;
   }
   }
   }
   END {
    print "Candidate "last_name, "with principal committee ID "PCN,\
       "has "count, "contributions \nfrom individuals above $200 nationwide and "count_CA,\
       "such contributions from California.";
   }' FS='|' data/itcont.txt
 fi
}

echo "(b)"
get_contributions_above200 "McConnell, Mitch"
get_contributions_above200 "Grimes, Alison"
```

```
echo "(c)"
# Modify internal field separater IFS
IFS=:
candidates="MCCONNELL, MITCH":"GRIMES, ALISON":"CORNYN, JOHN":"MARKEY, EDWARD JOHN MR"

for candidate in $candidates
do
   get_contributions_above200 $candidate
done
# Restore default IFS
IFS=$' \t\n'

# (d)
# Get number of individual contributions to candidate
# above $200 nationwide and for California given [last name] or
# [last name, part or all of first name]
function get_contributions_above200_v2() {
  local last_name="$1" ##<< String of form [lastname] or [lastname, firstname]
  local PCN
  PCN=$(get_PCN "$last_name")

if [ $(echo "$PCN" | wc -w) -ne 1 ]
then
echo "$PCN"
else
   awk -v last_name="$last_name" -v PCN="$PCN" 'BEGIN {
   count = 0;
   count_CA = 0;
  }
  {
   if ($1 == PCN && $15 > 200) {
   count++;
   if ($10 == "CA") {
   count_CA++;
   }
   }
   }
   END {
      print "Candidate "last_name, "with principal committee ID "PCN,\
       "has "count, "contributions \nfrom individuals above $200 nationwide and "count_CA,\
       "such contributions from California.";
   }' FS='|' data/itcont.txt
 fi
}
echo "(d)"
get_contributions_above200_v2 "McConnell, Mitch"
get_contributions_above200_v2 "Grimes, Alison"
```

```
## (b)
## Candidate McConnell, Mitch with principal committee ID C00193342 has 6751 contributions
## from individuals above $200 nationwide and 288 such contributions from California.
## Candidate Grimes, Alison with principal committee ID C00547083 has 6452 contributions
## from individuals above $200 nationwide and 1026 such contributions from California.
## (c)
## Candidate MCCONNELL, MITCH with principal committee ID C00193342 has 6751 contributions
## from individuals above $200 nationwide and 288 such contributions from California.
## Candidate GRIMES, ALISON with principal committee ID C00547083 has 6452 contributions
## from individuals above $200 nationwide and 1026 such contributions from California.
## Candidate CORNYN, JOHN with principal committee ID C00369033 has 5475 contributions
## from individuals above $200 nationwide and 106 such contributions from California.
## Candidate MARKEY, EDWARD JOHN MR with principal committee ID C00196774 has 11017 contributions
## from individuals above $200 nationwide and 986 such contributions from California.
## (d)
## Candidate McConnell, Mitch with principal committee ID C00193342 has 6502 contributions
## from individuals above $200 nationwide and 275 such contributions from California.
## Candidate Grimes, Alison with principal committee ID C00547083 has 6078 contributions
## from individuals above $200 nationwide and 978 such contributions from California.
```

4 Problem 4

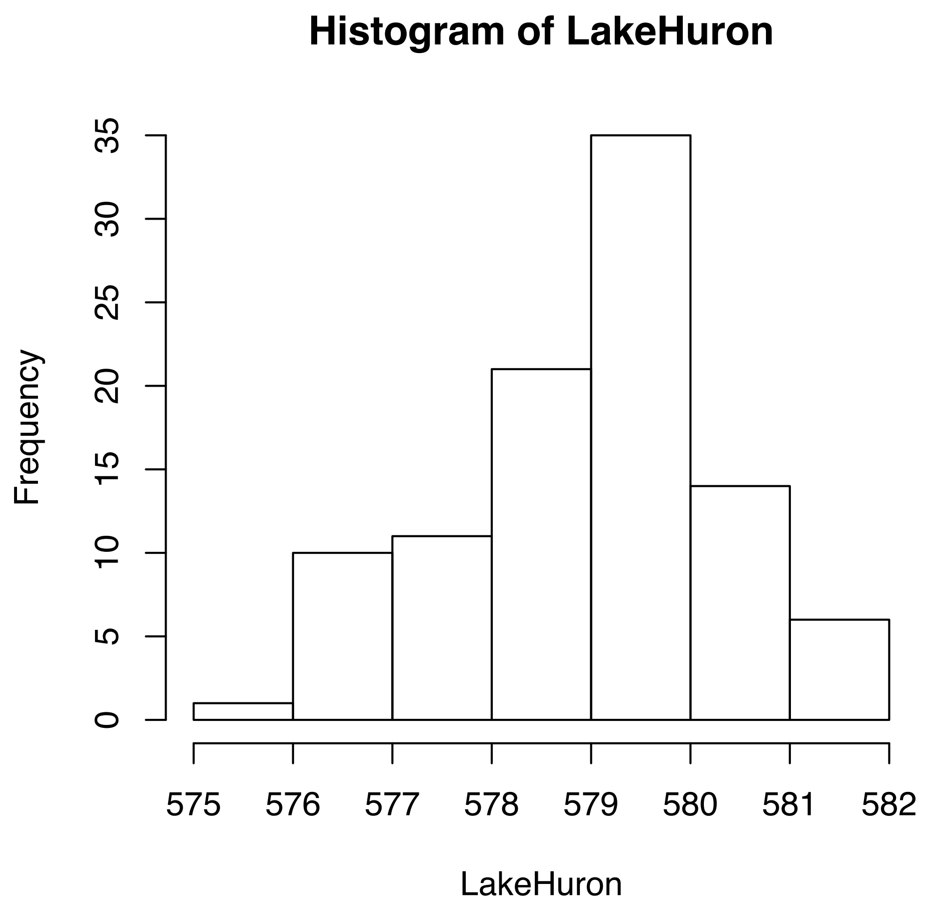
I downloaded the HTML index file and used `grep` to select only text in the rows that contain the string `txt`. Next, I used `cut` to extract only the file names after noticing that the file name is always found after a double quotation mark. I then used a `for` loop to download each file and print a status message.

```
 # Download HTML index file
 html_filepath="http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/"
 wget -O data/index.html $html_filepath

 # Extract file names of .txt files
 txt_files=$(grep -o "href=\".*.txt\"" data/index.html | cut -d'"' -f2)
 # Alternatively:
 # txt_files=$(grep -o "href=\".*.txt\"" data/index.html | sed -e 's/href=\"//' -e 's/\"//')

 for txt_file in $txt_files
 do
   wget -O data/${txt_file} $html_filepath${txt_file}
   echo "Downloaded $html_filepath${txt_file}"
 done

## Downloaded http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-countries.txt
## Downloaded http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-inventory.txt
## Downloaded http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-states.txt
## Downloaded http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-stations.txt
## Downloaded http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-version.txt
## Downloaded http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/readme.txt
## Downloaded http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/status.txt
```

5 Problem 5

The height of the water level in Lake Huron fluctuates over time. Here I ‘analyze’ the variable using R. I show a histogram of the lake levels for the period 1875 to 1972.

```
hist(LakeHuron)
```



```
lowHi <- c(which.min(LakeHuron), which.max(LakeHuron))
yearExtrema <- attributes(LakeHuron)$tsp[1] - 1 + lowHi
```