BME 590 Homework 1

In this homework, the objectives are to

1. Use R to calculate summary statistics and create basic plots of biomedical data.
2. Run bash commands in RMardown to answer questions.

Assignments will only be accepted in electronic format in RMarkdown (.rmd) files and PDF files. Commented versions of any code to produce analyses will be required. RMarkdown and PDF homework files should be uploaded to Gradescope with the naming convention date_lastname_firstname_HW[X].Rmd. For example, Dr.Dunn's first homework assignments would be named 20210129_Dunn_Jessilyn_HW1.Rmd and 20210129Dunn_Jessilyn_HW1.pdf. It is important to note that **5 points** will be deducted for every assignment that is named improperly.

```
library(ggplot2)
library(tidyverse)
library(knitr)
```

Dataset Summary and Plotting

R Markdown shortcuts: https://rmd4sci.njtierney.com/keyboard-shortcuts

1. Load the dataset named "HW1_dataset.csv" that you can download from Sakai HW1 folder. The column names are self-explanatory. Display the first 10 lines of the dataset.

```
# Set the working directory
setwd("E:/BME 590 01/homework 1")

# Load the dataset
I <- read.csv("HW1_dataset.csv")
```

```
# Display the first 10 lines of the dataset
head(I, 10)
```

```
##    age       work_class  education     marital_status      occupation  race
## 1   38          Private  Bachelors      Never-married   Other-service White
## 2   63          Private  Bachelors      Never-married           Sales White
## 3   44      Federal-gov Assoc-acdm           Divorced Exec-managerial White
## 4   50 Self-emp-not-inc  Bachelors Married-civ-spouse           Sales White
## 5   37          Private  Bachelors      Never-married Exec-managerial White
## 6   27        Local-gov  Bachelors      Never-married  Prof-specialty White
## 7   27     Self-emp-inc    HS-grad      Never-married           Sales White
## 8   52          Private    HS-grad           Divorced           Sales White
## 9   60 Self-emp-not-inc    HS-grad Married-civ-spouse           Sales White
## 10  37          Private    HS-grad Married-civ-spouse           Sales White
##       sex hours_per_week income_class
```

```
## 1    Male             20        <=50K
## 2    Male             27        <=50K
## 3  Female             40        <=50K
## 4    Male              8        <=50K
## 5    Male             50        <=50K
## 6    Male             35        <=50K
## 7    Male             50        <=50K
## 8  Female             40        <=50K
## 9    Male             40        <=50K
## 10   Male             45        <=50K
```

2. What's the number of rows in this dataset? What's the number of columns in this dataset?

```
# The number of rows in the data set
paste(" The number of rows is: ", nrow(I))
```

```
## [1] " The number of rows is:  5000"
```

```
# The number of columns in the data set
paste(" The number of columns is: ", ncol(I))
```

```
## [1] " The number of columns is:  9"
```

3. Add a column named ">=50" that has TRUE for "income_class" indicating income greater than 50K and FALSE otherwise. Show the first 10 lines of this dataset.

```
# Create a vector that has the same number as the observations
v <- vector(mode = "logical", length = 5000)

# Make all elements in the vector become TRUE if income_class is greater than
# 50 at the same position
v[I$income_class == ">50K"] <- TRUE

# Append the vector to the data frame, use a second backslash to prevent the
# escape problem
I["\\>=50"] <- v

# Show the first 10 lines of this data set
head(I, 10)
```

```
##    age       work_class  education     marital_status      occupation  race
## 1   38          Private  Bachelors      Never-married   Other-service White
## 2   63          Private  Bachelors      Never-married           Sales White
## 3   44      Federal-gov Assoc-acdm           Divorced Exec-managerial White
## 4   50 Self-emp-not-inc  Bachelors Married-civ-spouse           Sales White
## 5   37          Private  Bachelors      Never-married Exec-managerial White
## 6   27        Local-gov  Bachelors      Never-married  Prof-specialty White
## 7   27     Self-emp-inc    HS-grad      Never-married           Sales White
## 8   52          Private    HS-grad           Divorced           Sales White
## 9   60 Self-emp-not-inc    HS-grad Married-civ-spouse           Sales White
## 10  37          Private    HS-grad Married-civ-spouse           Sales White
```

```
##       sex hours_per_week income_class \\>=50
## 1    Male             20        <=50K  FALSE
## 2    Male             27        <=50K  FALSE
## 3  Female             40        <=50K  FALSE
## 4    Male              8        <=50K  FALSE
## 5    Male             50        <=50K  FALSE
## 6    Male             35        <=50K  FALSE
## 7    Male             50        <=50K  FALSE
## 8  Female             40        <=50K  FALSE
## 9    Male             40        <=50K  FALSE
## 10   Male             45        <=50K  FALSE
```

Reference:
* https://stackoverflow.com/questions/10605485/an-error-is-an-unrecognized-escape-in-character-string-starting-while
* https://rstudio-education.github.io/hopr/modify.html
* http://uc-r.github.io/creating_vectors

4. How many entries came from people who are black, male and received more than 50K US dollars per year? How many entries came from people who are white, female and received less **or equal** than 50K US dollars per year? You can try either the filter() function or the count() function, or any other function as you please.
   (1)

```
# Call the count function
# First find black
temp = I[I$race == "Black", ]
# Then find Male
temp = temp[temp$sex == "Male", ]
# At last count by group of income_class
count(temp, income_class)
```

```
##   income_class  n
## 1        <=50K 89
## 2         >50K 76
```

```
# Call the filter function
count1 <- filter(.data = I, race == "Black" & sex == "Male" & income_class == ">50K")

# Calculate the number of row of filtered dataframe
nrow(count1)
```

```
## [1] 76
```

```
print("76 entries came from people who are black, male and received more than 50K US dollars per year.")
```

```
## [1] "76 entries came from people who are black, male and received more than 50K US dollars per year."
```

(2)

```
# Call the filter function
count2 <- filter(.data = I, race == "White" & sex == "Female" & income_class == "<=50K")

# Calculate the number of row of filtered dataframe
nrow(count2)
```

```
## [1] 1077
```

```
print(" 1077 entries came from people who are white, female and received less or equal than 50K US dolla
```

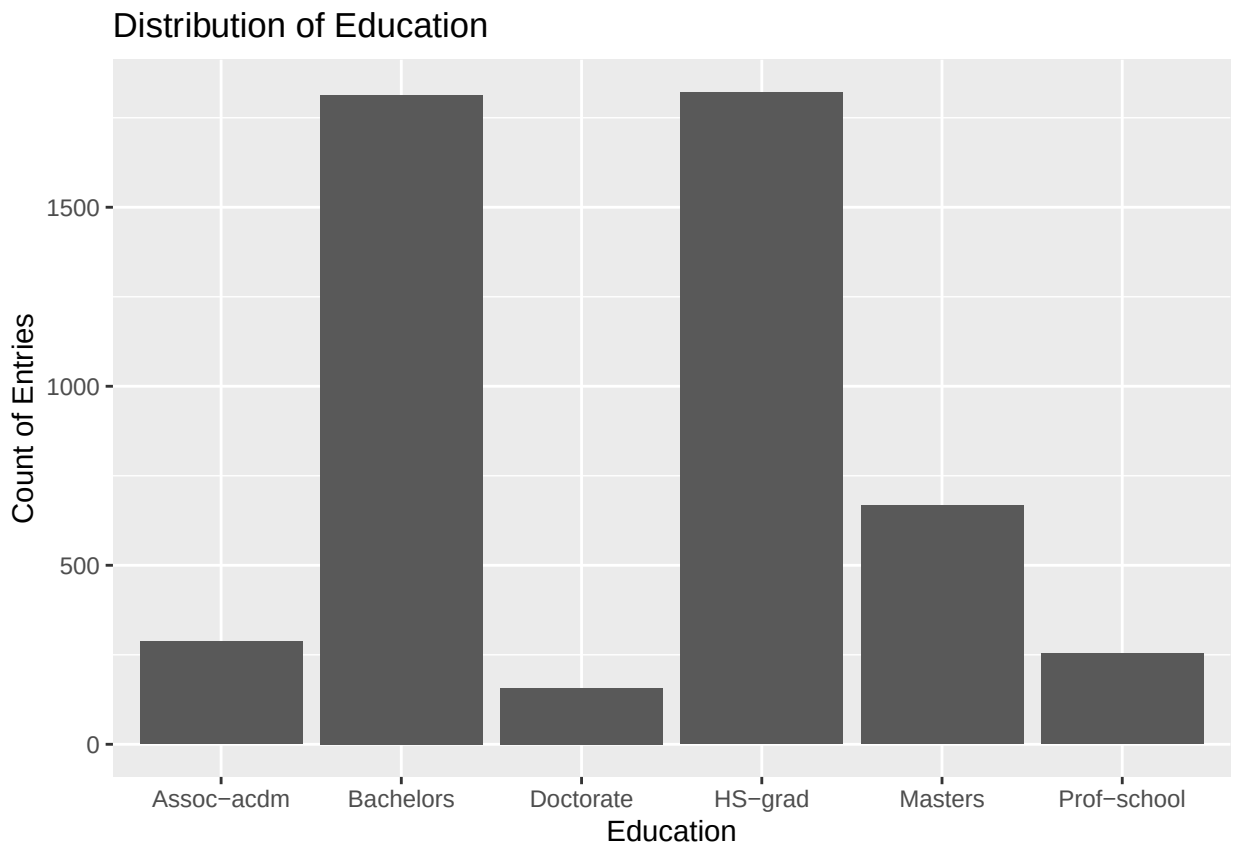
```
## [1] " 1077 entries came from people who are white, female and received less or equal than 50K US doll
```

Reference: https://www.rdocumentation.org/packages/dplyr/versions/0.7.8/topics/filter

5. Use ggplot and plot a histogram of the education predictor in this dataset.

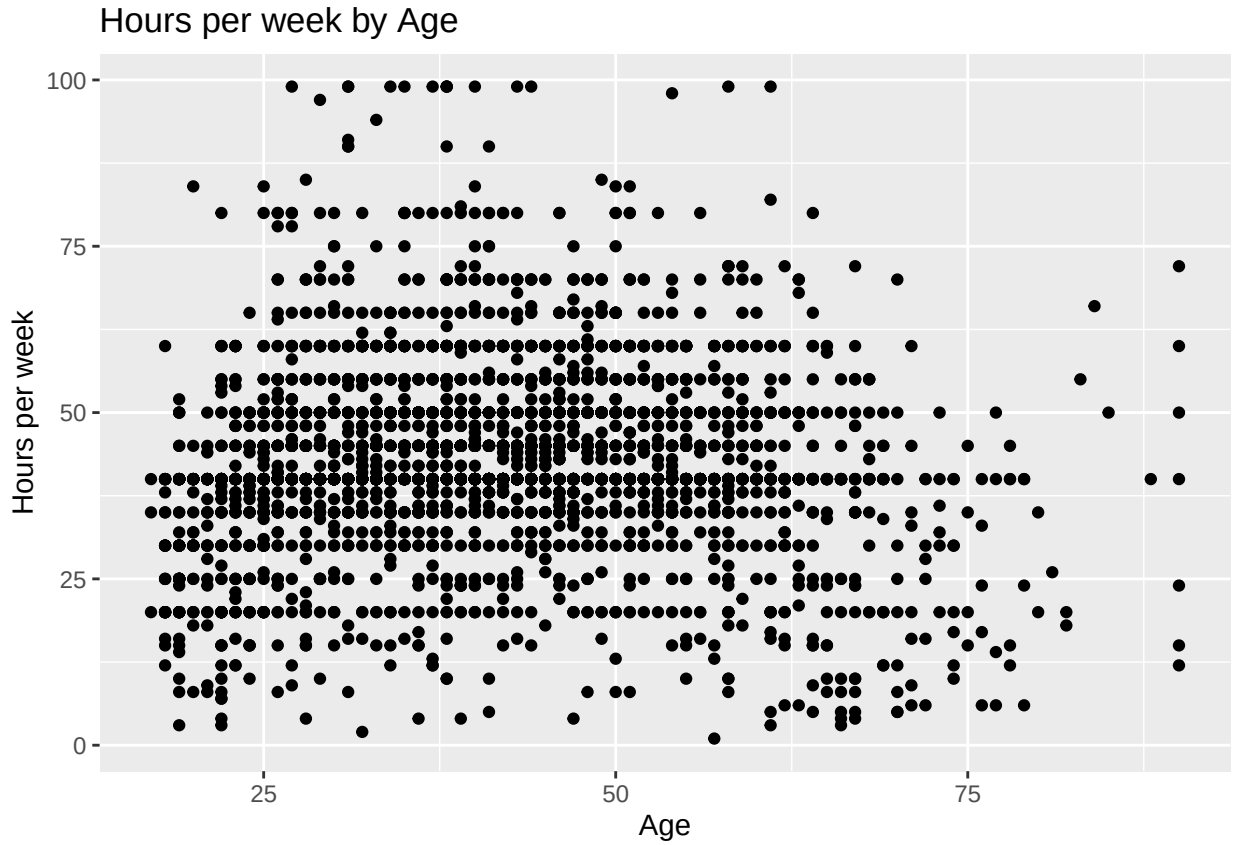
```
ggplot(
    data = I,
    aes(x = education)) +
    geom_histogram(
        stat="count") +
    ggtitle("Distribution of Education") +
    xlab("Education") +
    ylab("Count of Entries")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



6. Use ggplot and plot a scatter plot of hours_per_week over age. Does there seem to be a pattern?

```
ggplot(
    data = I,
    aes(
        x = age,
        y = hours_per_week)) +
    geom_point() +
    ggtitle("Hours per week by Age") +
    xlab("Age") +
    ylab("Hours per week")
```

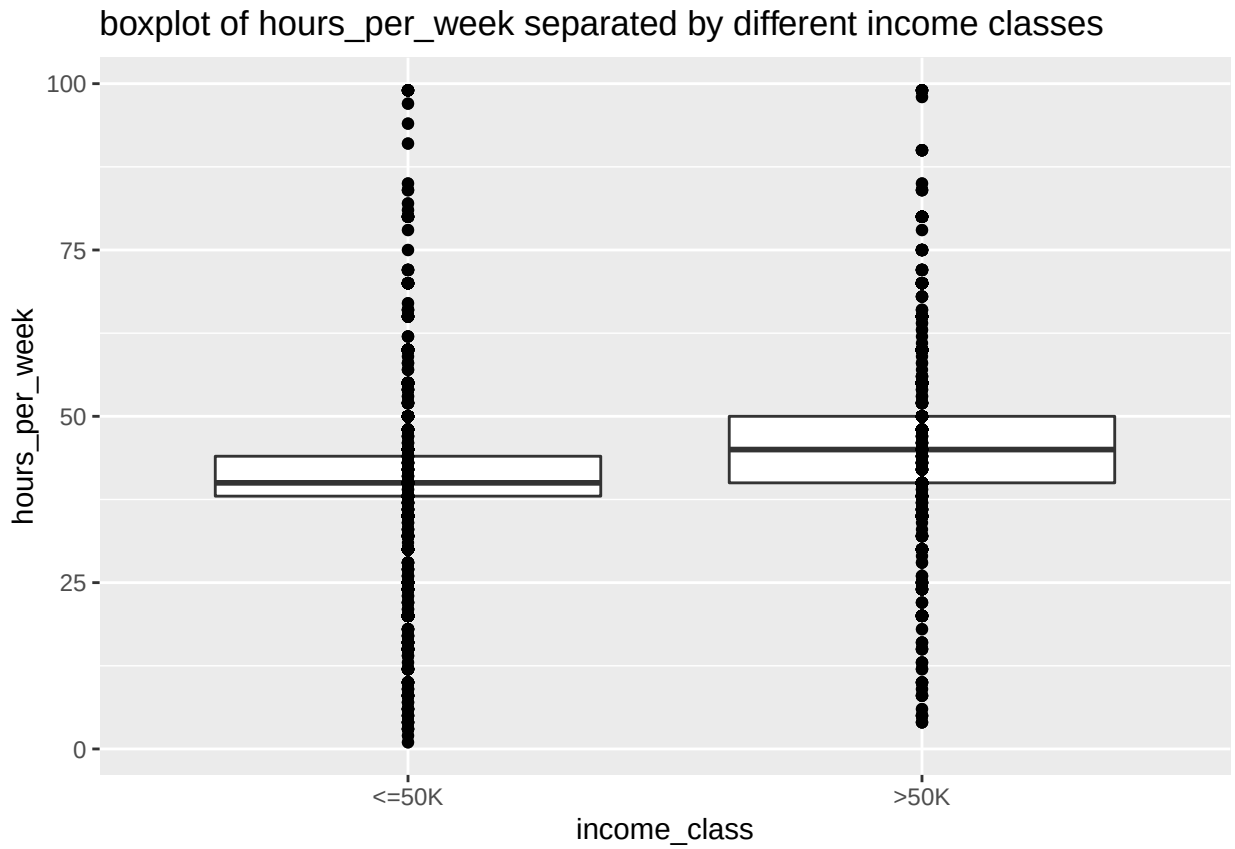


There does not seem to be a pattern.

7. Use ggplot to plot a boxplot of hours_per_week separated by different income classes.

- income class should be at the x-axis
- y-axis should be the values in the hours_per_week column

```
ggplot(I, aes(x=income_class, y=hours_per_week)) + geom_boxplot()+
  geom_point() +
  ggtitle("boxplot of hours_per_week separated by different income classes")
```



Write a Function in R

1. Download Heart Disease data set from UCI Machine Learning Repository using the following bash command. Bash chunk can be inserted by clicking on the drop-down menu beside the **Insert** at the top-left of the editor. **wget** is a bash tool that can be easily downloaded and installed on your computer. Depending on your operating system and package management preferences, you will have to install wget through different methods. Please search on Google on how to install wget.

```
wget https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data
```

2. Import the file into R and label the column names properly. Note you can get some necessary information from the archive. Print the first few lines of the dataframe including your header row. How many rows and columns are there in this dataset? *Reference:* https://stackoverflow.com/questions/6081439/changing-column-names-of-a-data-frame

```
# Read Table
heart <- read.table(
  file = "processed.cleveland.data",
  sep = ","
)
```

```r
# Rename column names according to the Archive using colnames()
colnames(heart) <- c("age", "sex", "cp", "trestbps", "chol", "fbs", "restecg",
                     "thalach", "exang", "oldpeak", "slope", "ca", "thal",
                     "num")
```

```r
# Peek the data
head(heart)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope  ca thal num
## 1  63   1  1      145  233   1       2     150     0     2.3     3 0.0  6.0   0
## 2  67   1  4      160  286   0       2     108     1     1.5     2 3.0  3.0   2
## 3  67   1  4      120  229   0       2     129     1     2.6     2 2.0  7.0   1
## 4  37   1  3      130  250   0       0     187     0     3.5     3 0.0  3.0   0
## 5  41   0  2      130  204   0       2     172     0     1.4     1 0.0  3.0   0
## 6  56   1  2      120  236   0       0     178     0     0.8     1 0.0  3.0   0
```

```r
paste("The number of rows in this data set is: ", nrow(heart))
```

```
## [1] "The number of rows in this data set is:  303"
```

```r
paste("The number of columns in this data set is: ", ncol(heart))
```

```
## [1] "The number of columns in this data set is:  14"
```

3. The last column of the data contains diagnosis information ranging from 0 to 4. The "0" label indicates that the subject is healthy, whereas the subjects with non-zero value in this field are positively diagnosed with heart disease. Using dplyr, add a new column named "diagnosed" that contains binary information about whether the subject has heart disease or not (e.g. true/false, 0/1, etc). Print the first few rows of your dataframe to confirm you have successfully created the new column.

```r
# Make a copy of the data frame
heart_copy <- heart

# Add a new column by calling the muate function
heart <-mutate(heart_copy, "diagnosed" = (num >= 1))

head(heart)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope  ca thal num
## 1  63   1  1      145  233   1       2     150     0     2.3     3 0.0  6.0   0
## 2  67   1  4      160  286   0       2     108     1     1.5     2 3.0  3.0   2
## 3  67   1  4      120  229   0       2     129     1     2.6     2 2.0  7.0   1
## 4  37   1  3      130  250   0       0     187     0     3.5     3 0.0  3.0   0
## 5  41   0  2      130  204   0       2     172     0     1.4     1 0.0  3.0   0
## 6  56   1  2      120  236   0       0     178     0     0.8     1 0.0  3.0   0
##   diagnosed
## 1     FALSE
## 2      TRUE
## 3      TRUE
## 4     FALSE
## 5     FALSE
## 6     FALSE
```

4. Write a function that takes in a dataframe and a column name of the dataframe and prints the summary statistics of values in that dataframe. You should follow this logic with potential improvements:

- If the values are numeric, print that the values are numeric and then print the maximum value, minimum value, mean and median.
- If the values are characters, categorical or logical, print the data type and then print the counts of each existing values.
- You should also check that the column name provided by the user of this function is indeed a column of the dataframe provided by the user as well. Check this and output appropriate error message.

Here is an example simple function that add two numbers (or vectors).

```
add_two_numbers <- function(a,b) {
  if (!is.numeric(a) || !is.numeric(b)) {
    stop("Invalid input(s): inputs must all be atomic numerical values.")
  }

  return(a+b)
}
add_two_numbers(c(1,2),2)
```

```
## [1] 3 4
```

```
str(heart)
```

```
## 'data.frame':    303 obs. of  15 variables:
##  $ age      : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex      : num  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp       : num  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps : num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol     : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs      : num  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg  : num  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach  : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang    : num  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak  : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope    : num  3 2 2 3 1 1 3 1 2 3 ...
##  $ ca       : chr  "0.0" "3.0" "2.0" "0.0" ...
##  $ thal     : chr  "6.0" "3.0" "7.0" "3.0" ...
##  $ num      : int  0 2 1 0 0 0 3 0 2 1 ...
##  $ diagnosed: logi  FALSE TRUE TRUE FALSE FALSE FALSE ...
```

Your function here:

```
summarize_column <- function(df, column_name) {
  if (column_name %in% colnames(df)) {
    if (is.numeric(df[, column_name])) {
      cat(paste("The Maximum value is: ", max(df[, column_name])))
      cat(paste("\nThe Minimum value is: ", min(df[, column_name])))
      cat(paste("\nThe Mean value is: ", mean(df[, column_name])))
      cat(paste("\nThe Median value is: ", median(df[, column_name])))
    } else if (is.logical(df[, column_name]) ||
```

```
               is.character(df[, column_name])||
               is.categorical(df[, column_name])){
      cat("The data type of this column is: ", typeof(df[, column_name]))
      cat("\nThe numbers of each values are: ")
      table(df[, column_name])
    } else {
      stop("Error! Unknown data type!")
    }
  } else {
    stop("Error! The coulmn_name you input does not exist!")
  }

}
```

```
summarize_column(heart, "ca")
```

```
## The data type of this column is:  character
## The numbers of each values are:

##
##   ? 0.0 1.0 2.0 3.0
##   4 176  65  38  20
```

5. Test the function using the dataframe we have above, both when it works and it outputs an error messages. You can add the clause "error = TRUE" at the r chunk declaration so that your knitting is not stopped by this error. Test it on both the heart disease dataset as well as the Chromosome 20 dataset.

```
summarize_column(heart, "sex")
```

```
## The Maximum value is:  1
## The Minimum value is:  0
## The Mean value is:  0.67986798679868
## The Median value is:  1
```

```
summarize_column(heart, "aa")
```

```
## Error in summarize_column(heart, "aa"): Error! The coulmn_name you input does not exist!
```

Shell Scripting

Working in R is sometimes limited by the amount of data that R can load and allow the user to efficiently work with. Hence, sometimes we interact with datasets using shell scripting, or bash commands. To complete this homework, you will need to access DCC and use the shell environment directly available from there. Follow the these steps to access DCC:

- Connect to DukeVPN following the instructions in this website: https://oit.duke.edu/what-we-do/services/vpn

- Log into Duke Compute Cluster (DCC) by typing below in your terminal:

```
ssh mh511@dcc-slogin-03.oit.duke.edu
```

To receive full credits for each of the questions, you should copy the shell command you used to achieve the answer and also the printed answer from the shell terminal after each question.

We will work with a dataset that comes from this Our World in Data. It shows many metrics such as the new case count, the total death count and death count per million population from every region and country since the end of last year.

1. Upload the folder named "owid-covid-data.csv.gz" that you can download from Sakai HW1 folder to your DCC home directory.

```
scp /henry/home/owid-covid-data.csv.gz mh511@dcc-login-03.oit.duke.edu/hpc/home/mh511
```

2. Look at the first 5 lines and last 5 lines of the data file owid-covid-data.csv.gz without unzipping it (This is not a very large dataset per se, but is used to simulate a working with a large dataset). Find out what "zcat" does by running "man zcat" and use "zcat" to complete this task.

```
zcat owid-covid-data.csv.gz | head -5
```

iso_code,continent,location,date,total_cases,new_cases,new_cases_smoothed,total_deaths,new_deaths,new_deaths_smoo
ABW,North America,Aruba,2020-03-13,2.0,2.0„0.0,0.0„18.733,18.733„0.0,0.0„„„„„„,0.0,106766.0,584.8,41.2,13.085,7.452,3597
ABW,North America,Aruba,2020-03-19„,0.286„,0.0„,2.676„,0.0„„„„„33.33,106766.0,584.8,41.2,13.085,7.452,35973.781„,11.6
ABW,North America,Aruba,2020-03-20,4.0,2.0,0.286,0.0,0.0,0.0,37.465,18.733,2.676,0.0,0.0,0.0„„„„„33.33,106766.0,584.8,41.
ABW,North America,Aruba,2020-03-21„,0.286„,0.0„,2.676„,0.0„„„„„44.44,106766.0,584.8,41.2,13.085,7.452,35973.781„,11.6

```
zcat owid-covid-data.csv.gz | tail -5
```

„International,2020-08-20,696.0„,7.0„„„„„„„„„„„„„„„„„ „International,2020-08-21,696.0„,7.0„„„„„„„„„„„„„„„„„
„International,2020-08-22,696.0„,7.0„„„„„„„„„„„„„„„„„ „International,2020-08-23,696.0„,7.0„„„„„„„„„„„„„„„„„
„International,2020-08-24,696.0„,7.0„„„„„„„„„„„„„„„„„

3. How many lines are there in this file? Find out without unzipping the file.

```
zcat owid-covid-data.csv.gz | wc -l
```

39272

4. Unzip owid-covid-data.csv.gz without deleting the original zipped file. Use "man gunzip" to find out how to do that.

```
gunzip -k owid-covid-data.csv.gz

# Two other options
# gunzip < file.gz > file
# zcat file.gz > file
```

5. Find out how many days of COVID-19 data *Italy* has in this dataset. Use "grep" to find the lines that contain the work "Italy" and count the lines.

```
# Get the Number of Italy
grep -o -i Italy owid-covid-data.csv | wc -l
```

238

```
# To find the lines that contain the word
# Print both the line number and the line
grep -n Italy owid-covid-data.csv

# Only print out the line number
grep -n Italy owid-covid-data.csv | cut -f1 -d: | sort -u
```

18178 18179 18180 18181 18182 18183 18184 18185 18186 18187 18188 18189 18190 18191 18192 18193 18194
18195 18196 18197 18198 18199 18200 18201 18202 18203 18204 18205 18206 18207 18208 18209 18210 18211
18212 18213 18214 18215 18216 18217 18218 18219 18220 18221 18222 18223 18224 18225 18226 18227 18228
18229 18230 18231 18232 18233 18234 18235 18236 18237 18238 18239 18240 18241 18242 18243 18244 18245
18246 18247 18248 18249 18250 18251 18252 18253 18254 18255 18256 18257 18258 18259 18260 18261 18262
18263 18264 18265 18266 18267 18268 18269 18270 18271 18272 18273 18274 18275 18276 18277 18278 18279
18280 18281 18282 18283 18284 18285 18286 18287 18288 18289 18290 18291 18292 18293 18294 18295 18296
18297 18298 18299 18300 18301 18302 18303 18304 18305 18306 18307 18308 18309 18310 18311 18312 18313
18314 18315 18316 18317 18318 18319 18320 18321 18322 18323 18324 18325 18326 18327 18328 18329 18330
18331 18332 18333 18334 18335 18336 18337 18338 18339 18340 18341 18342 18343 18344 18345 18346 18347
18348 18349 18350 18351 18352 18353 18354 18355 18356 18357 18358 18359 18360 18361 18362 18363 18364
18365 18366 18367 18368 18369 18370 18371 18372 18373 18374 18375 18376 18377 18378 18379 18380 18381
18382 18383 18384 18385 18386 18387 18388 18389 18390 18391 18392 18393 18394 18395 18396 18397 18398
18399 18400 18401 18402 18403 18404 18405 18406 18407 18408 18409 18410 18411 18412 18413 18414 18415

6. Look at "owid_covid_codebook.csv", which is a data dictionary for "owid_covid_data.csv". Each row is a COVID-19 case count entry. How many columns and how many rows are there in this dataset? Find out which column indicates the country from the codebook. Use bash command to determine how many unique countries have data in this file. I recommend using "awk" to parse the file.

```
# To find the rows and columns
# First Load the R module
module load R/3.2.5

R
```

```
# Read csv file
I <- read.csv("owid-covid-data.csv")

# Row number
nrow(I)

# Column Number
ncol(I)
```

Row number is 39271 (39272 if the title(head) is included) column number is 40

```
# Can also use the bash command
# Count the row number
wc -l owid-covid-data.csv
```

```
# Count the column number
awk "{print NF}" < owid-covid-data.csv | uniq
```

```
# TO find out which column indicate the contry
which(colnames(I) == "location")
```

3

```
# To see all the unique countries
tail -n +2 owid-covid-data.csv | awk -F , '{ a[$3]++ } END { for (b in a) { print b } }'
```

Malaysia Eritrea Angola Bolivia Norway Greenland Estonia Thailand Benin Aruba Haiti Finland Denmark Taiwan Maldives Ghana Iran Falkland Islands Dominican Republic Ethiopia Netherlands South Sudan Pakistan Iraq Australia Montenegro Belize Sint Maarten (Dutch part) Uganda Equatorial Guinea Mauritius Dominica Kuwait Ukraine France Azerbaijan New Zealand Western Sahara Albania Bosnia and Herzegovina Malta Mexico Algeria Cape Verde International Togo Poland Guernsey Czech Republic Seychelles Serbia Palestine Oman Fiji New Caledonia Liberia Hong Kong Guyana Hungary South Africa Uruguay Papua New Guinea Chile Namibia Comoros Turks and Caicos Islands Laos United Arab Emirates China Vietnam Saint Lucia Cyprus Central African Republic Macedonia Latvia Kazakhstan Egypt Cameroon Senegal Portugal Djibouti Saudi Arabia Ireland Ecuador Philippines Peru Guinea Georgia Bahamas United States Switzerland Trinidad and Tobago Sweden Slovakia Sao Tome and Principe Singapore Guinea-Bissau Syria Paraguay Democratic Republic of Congo Mongolia French Polynesia Luxembourg Zimbabwe Anguilla Cote d'Ivoire Indonesia Afghanistan Mozambique Bermuda Lithuania Honduras Niger Libya Chad India Nicaragua Kenya Burundi Uzbekistan Mali Slovenia Qatar Iceland Yemen Kyrgyzstan Greece Cayman Islands Curacao Suriname Gibraltar Bulgaria Sierra Leone Guatemala British Virgin Islands Tunisia Antigua and Barbuda Brunei Tanzania Bonaire Sint Eustatius and Saba El Salvador Jordan Croatia Faeroe Islands Bhutan Botswana Saint Vincent and the Grenadines Belarus Cambodia Jamaica Barbados Rwanda Panama Nepal Mauritania Gabon Northern Mariana Islands Sudan Armenia Kosovo Zambia Nigeria Canada Somalia Romania Japan Colombia Burkina Faso Timor United Kingdom Bahrain Austria World Tajikistan Swaziland Morocco Venezuela Puerto Rico Israel Montserrat Saint Kitts and Nevis Grenada Argentina Congo Jersey Germany United States Virgin Islands Myanmar Monaco Cuba San Marino Malawi Sri Lanka Italy Costa Rica Andorra Vatican South Korea Guam Gambia Liechtenstein Bangladesh Russia Moldova Turkey Belgium Spain Madagascar Lesotho Lebanon Brazil Isle of Man

7. Use "awk" to filter by country "iso_code" so that we are only looking at COVID data from the USA. Find out the date that has the largest number of single day new cases in the USA using "sort". https://www.joeldare.com/wiki/using_awk_on_csv_files/ https://www.timdennis.com/data/tech/2016/08/09/using-awk-filter-rows.html/ https://unix.stackexchange.com/questions/316867/how-to-identify-sort-descending-and-display-the-top-10-blocks-of-text-by-categ/ https://stackoverflow.com/questions/17048188/how-to-use-awk-sort-by-column-3

```
# Filter and then pipe
# Sort by descending order and save the result in a new file
awk -F "\"*,\"*" '{ if ($1 == "USA") { print } }' owid-covid-data.csv | sort -t, -rnk6 | head -2 > sort
```

USA,North America,United States,2020-07-25,4112529.0,78427.0,66402.0,145546.0,1304.0,897.143,12424.46,236.938,200.609,4
unclear (incl. non-PCR),68.98,331002647.0,35.608,38.3,15.413,9.732,54225.446,1.2,151.089,10.79,19.1,24.6,,2.77,78.86
USA,North America,United States,2020-07-17,3576221.0,76930.0,65459.0,138358.0,939.0,723.857,10804.207,232.415,197.76,41
unclear (incl. non-PCR),68.98,331002647.0,35.608,38.3,15.413,9.732,54225.446,1.2,151.089,10.79,19.1,24.6,,2.77,78.86

The largest number of single day new cases is: 78427