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)
```

Dataset Summary and Plotting

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.

```
# Load the dataset
df <- read.csv("HW1_dataset.csv")
# Print the first 10 lines of the dataset
head(df,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?

```
# number of rows in the dataset
nrow(df)
```

```
## [1] 5000
```

```
# number of columns in the dataset
ncol(df)
```

```
## [1] 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.

```
# build a new column checking if the income is <=50K with boolean values
df$check <- ifelse(df$income_class == "<=50K", "FALSE", "TRUE")
# rename the column name as ">=50K"
names(df)[10] <- ">=50K"
# show the first 10 lines of the dataset
head(df,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 >=50K
## 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
```

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 than 50K US dollars per year? You can try either the filter() function or the count() function, or any other function as you please.

```
# count of black, male and received more than 50K US dollars per year
count(df %>% filter(race == "Black") %>% filter(sex == "Male") %>% filter(income_class == ">50K"))
```

```
##    n
## 1 76
```

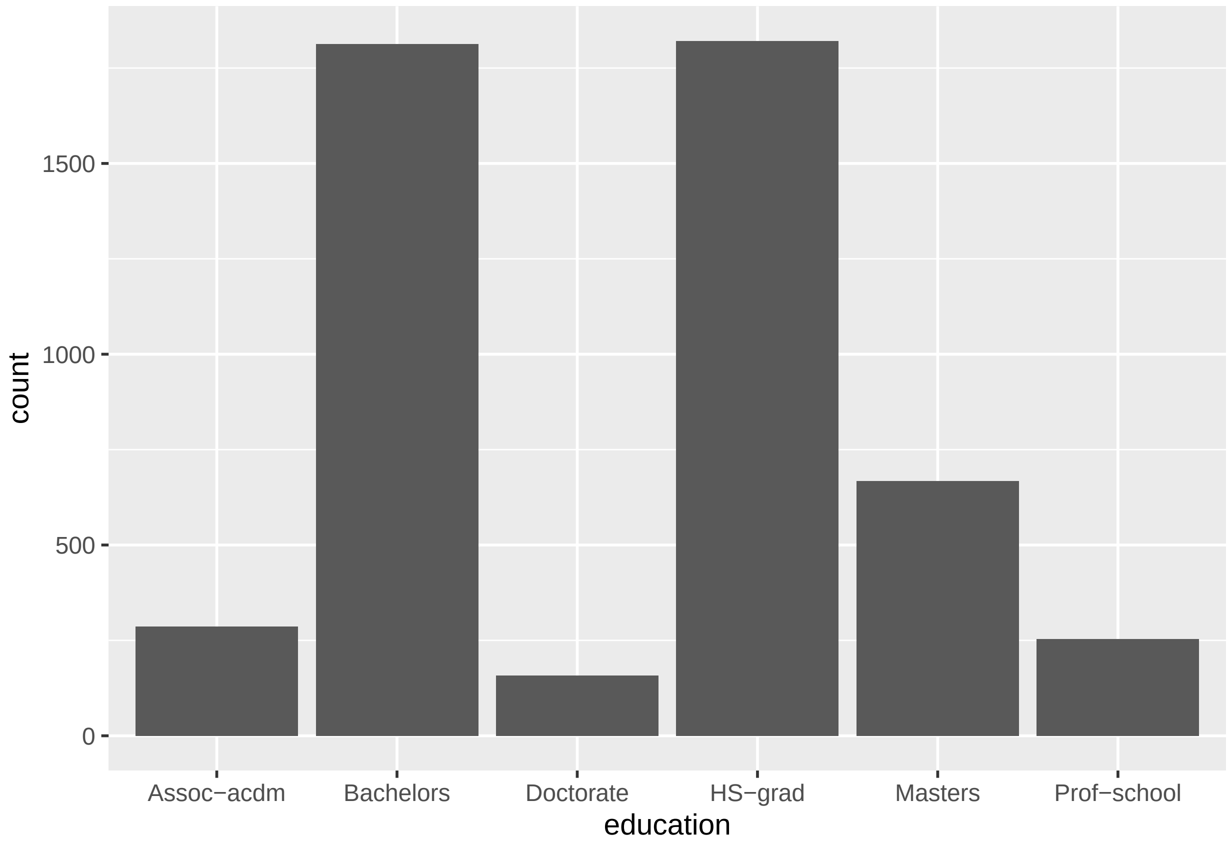
```
#count of white, female and received less than 50K US dollars per year
count(df %>% filter(race == "White") %>% filter(sex == "Female") %>% filter(income_class == "<=50K"))
```

```
##      n
## 1 1077
```

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

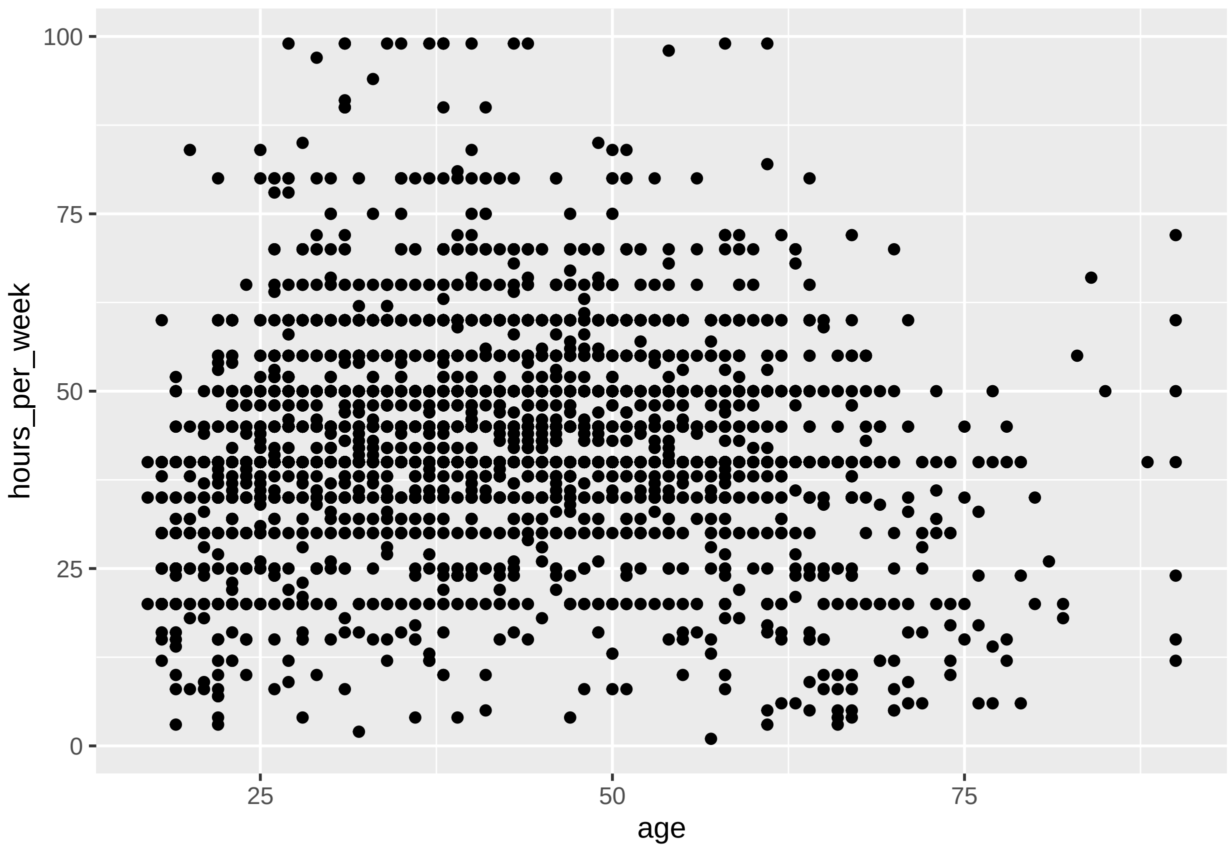
```
# Plot the histogram
ggplot(df, aes(x=education)) + geom_histogram(stat="count")
```

```
## 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?

```
# Plot the scatter plot
ggplot(df, aes(x=age, y=hours_per_week)) + geom_point()
```

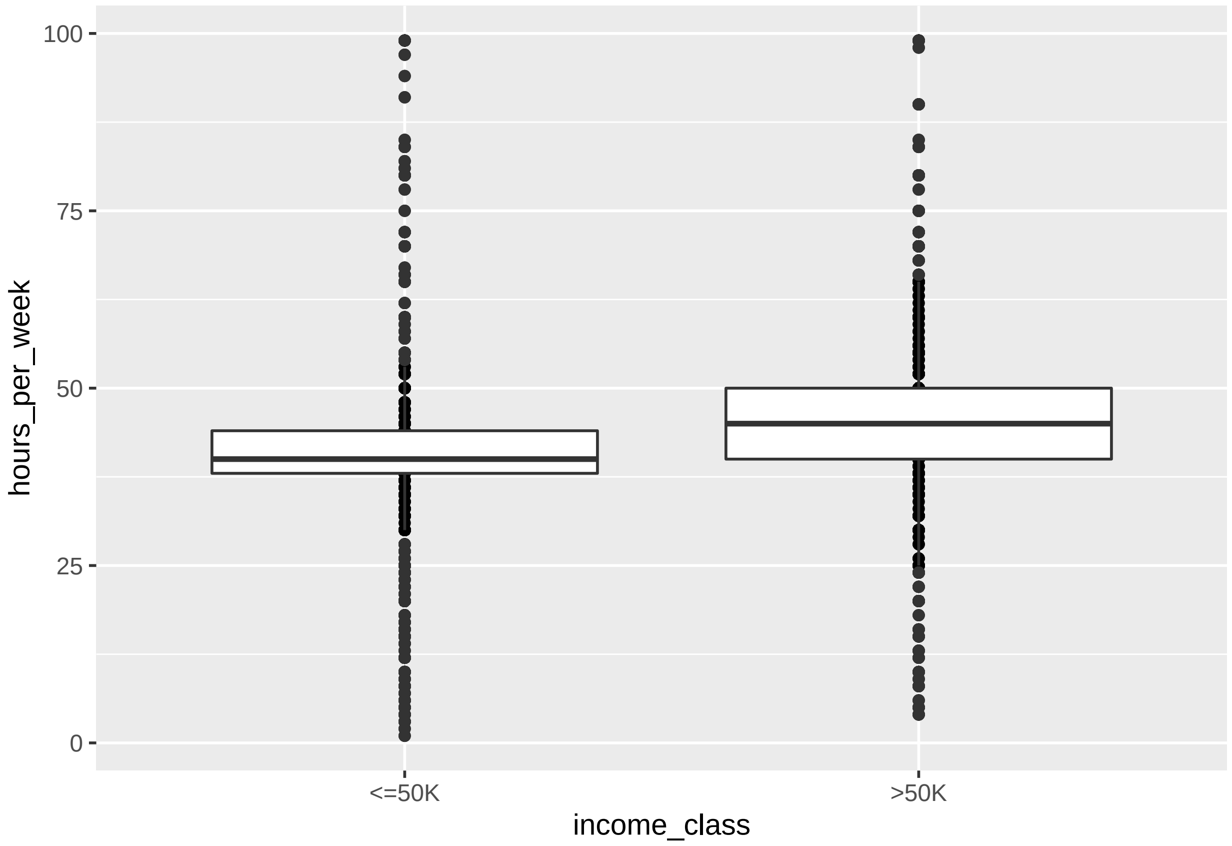


It appears that much fewer people are working as age goes up and most people in their working ages have a similar pattern of working around 25-50 hrs a week.

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

```
# plot the box plot
ggplot(df, aes(x=income_class, y=hours_per_week)) + geom_point() +geom_boxplot()
```



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?

```
# Read and load the dataset
d <- read.table("processed.cleveland.data", sep=",")
# Label the column names
colnames(d) <- c("age",
                 "sex",
                 "cp",
                 "trestbps",
                 "chol",
                 "fbs",
                 "restecg",
                 "thalach",
                 "exang",
                 "oldpeak",
                 "slope",
                 "ca",
                 "thal",
                 "num")
# show the first 10 lines of the dataset
head(d,10)
```

```
##    age sex cp trestbps chol fbs restecg thalach exang oldpeak slope  ca thal
## 1   63   1  1      145  233   1       2     150     0     2.3     3 0.0  6.0
## 2   67   1  4      160  286   0       2     108     1     1.5     2 3.0  3.0
## 3   67   1  4      120  229   0       2     129     1     2.6     2 2.0  7.0
## 4   37   1  3      130  250   0       0     187     0     3.5     3 0.0  3.0
## 5   41   0  2      130  204   0       2     172     0     1.4     1 0.0  3.0
## 6   56   1  2      120  236   0       0     178     0     0.8     1 0.0  3.0
## 7   62   0  4      140  268   0       2     160     0     3.6     3 2.0  3.0
## 8   57   0  4      120  354   0       0     163     1     0.6     1 0.0  3.0
## 9   63   1  4      130  254   0       2     147     0     1.4     2 1.0  7.0
## 10  53   1  4      140  203   1       2     155     1     3.1     3 0.0  7.0
##    num
## 1    0
## 2    2
## 3    1
## 4    0
## 5    0
## 6    0
## 7    3
## 8    0
## 9    2
## 10   1
```

```
# print the number of rows
nrow(d)
```

```
## [1] 303
```

```
# print the number of columns
ncol(d)
```

```
## [1] 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.

```
# Add a new column to the dataset called diagnosed
d <- d %>% add_column(diagnosed = (ifelse(d$num == 0, FALSE, TRUE)))
# Show the first 10 lines of the dataset
head(d,10)
```

```
##    age sex cp trestbps chol fbs restecg thalach exang oldpeak slope  ca thal
## 1   63   1  1      145  233   1       2     150     0     2.3     3 0.0  6.0
## 2   67   1  4      160  286   0       2     108     1     1.5     2 3.0  3.0
## 3   67   1  4      120  229   0       2     129     1     2.6     2 2.0  7.0
## 4   37   1  3      130  250   0       0     187     0     3.5     3 0.0  3.0
## 5   41   0  2      130  204   0       2     172     0     1.4     1 0.0  3.0
## 6   56   1  2      120  236   0       0     178     0     0.8     1 0.0  3.0
## 7   62   0  4      140  268   0       2     160     0     3.6     3 2.0  3.0
## 8   57   0  4      120  354   0       0     163     1     0.6     1 0.0  3.0
## 9   63   1  4      130  254   0       2     147     0     1.4     2 1.0  7.0
## 10  53   1  4      140  203   1       2     155     1     3.1     3 0.0  7.0
##    num diagnosed
## 1     0     FALSE
## 2     2      TRUE
## 3     1      TRUE
## 4     0     FALSE
## 5     0     FALSE
## 6     0     FALSE
## 7     3      TRUE
## 8     0     FALSE
## 9     2      TRUE
## 10    1      TRUE
```

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).

```r
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
```

Your function here:

```r
# Build the function
summarize_column <- function(data, column_name) {
  if (!is.data.frame(data) || is.null(data[column_name])) {
    stop("Invalid input(s): inputs must be a valid dataframe and given a valid column name")
  }
  this_d <- data[[column_name]]
  if(is.numeric(this_d)){
    print("The values are numeric")
    m <- matrix(c("max",max(this_d),"min",min(this_d),"mean",mean(this_d),"median",median(this_d)),nrow
    print(m)
  } else{
    print(class(this_d))
    table(this_d)
  }
}
```

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.

```
# Test summarize_column function
summarize_column(d,"age")
```

```
## [1] "The values are numeric"
##      [,1]  [,2]  [,3]               [,4]
## [1,] "max" "min" "mean"             "median"
## [2,] "77"  "29"  "54.4389438943894" "56"
```

```
summarize_column(dat,"a")
```

```
## Error in is.data.frame(data): object 'dat' not found
```

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 rh275@dcc-login.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 ~/Desktop/spring2021_Duke/BME590/owid-covid-data.csv.gz rh275@dcc-login-03.oit.duke.edu:/hpc/home/rh
```

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 "gzcat" does by running "man gzcat" and use "gzcat" to complete this task.

```
gzcat owid-covid-data.csv.gz | head -5
gzcat owid-covid-data.csv.gz | tail -5
```

```
## iso_code,continent,location,date,total_cases,new_cases,new_cases_smoothed,total_deaths,new_deaths,new
## ABW,North America,Aruba,2020-03-13,2.0,2.0,,0.0,0.0,,18.733,18.733,,0.0,0.0,,,,,,,,,,0.0,106766.0,58
## ABW,North America,Aruba,2020-03-19,,,0.286,,,0.0,,,2.676,,,0.0,,,,,,,,,,33.33,106766.0,584.8,41.2,13
## 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,,,,,,,,
## ABW,North America,Aruba,2020-03-21,,,0.286,,,0.0,,,2.676,,,0.0,,,,,,,,,,44.44,106766.0,584.8,41.2,13
## ,,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.

```
gzcat 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 -c owid-covid-data.csv.gz > owid.csv
```

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.

```
# Grep the data and show
grep "Italy" owid.csv | head -10
# Count lines
grep "Italy" owid.csv | wc -l
```

```
## ITA,Europe,Italy,2019-12-31,0.0,0.0,,0.0,0.0,,0.0,0.0,,0.0,0.0,,,,,,,,,,,,60461828.0,205.859,47.9,23
## ITA,Europe,Italy,2020-01-01,0.0,0.0,,0.0,0.0,,0.0,0.0,,0.0,0.0,,,,,,,,,,,0.0,60461828.0,205.859,47.9
## ITA,Europe,Italy,2020-01-02,0.0,0.0,,0.0,0.0,,0.0,0.0,,0.0,0.0,,,,,,,,,,,0.0,60461828.0,205.859,47.9
## ITA,Europe,Italy,2020-01-03,0.0,0.0,,0.0,0.0,,0.0,0.0,,0.0,0.0,,,,,,,,,,,0.0,60461828.0,205.859,47.9
## ITA,Europe,Italy,2020-01-04,0.0,0.0,,0.0,0.0,,0.0,0.0,,0.0,0.0,,,,,,,,,,,0.0,60461828.0,205.859,47.9
## ITA,Europe,Italy,2020-01-05,0.0,0.0,,0.0,0.0,,0.0,0.0,,0.0,0.0,,,,,,,,,,,0.0,60461828.0,205.859,47.9
## ITA,Europe,Italy,2020-01-06,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,,,,,,,,,,0.0,60461828.0,
## ITA,Europe,Italy,2020-01-07,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,,,,,,,,,,0.0,60461828.0,
## ITA,Europe,Italy,2020-01-08,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,,,,,,,,,,0.0,60461828.0,
## ITA,Europe,Italy,2020-01-09,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,,,,,,,,,,0.0,60461828.0,
##      238
```

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.

```
# number of columns
cat owid.csv | head -1 | sed 's/,/ /g' | wc -w
# number of rows
cat owid.csv | wc -l
# Determine unique countries
awk -F , '{ a[$3]++ } END { for (b in a) { print b } }' owid.csv | wc -l
```

```
##       40
##    39272
##      213
```

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”.

```
# Get the line of largest new cases day in USA
awk -F, '$1 == "USA"' owid.csv | sort -t, -nrk6 | head -1 | tr ',' ' ' > list.txt
# print out the date
awk '{print $6}' list.txt
```

```
## 2020-07-25
```