STAT115 Homework 1 By MACAULAY

Friday June 16, 2023

Part I: Introduction to R

Problem 1: Installation

Please install the following R/Bioconductor packages

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install()
BiocManager::install("sva")

install.packages(c("ggplot2", "dplyr", "tidyr", "HistData", "mvtnorm",
                   "reticulate"))
```

Please run this command (use `eval=TRUE`) to see if Bioconductor can work fine.

```
BiocManager::valid()
```

```
# these packages are needed for HW2
# affy and affyPLM are needed to read the microarray data and run RMA
#library(sva) # for batch effect correction. Contains ComBat and sva.
library(ggplot2) # for plotting
library(dplyr) # for data manipulation
library(reticulate) # needed to run python in Rstudio
# these next two are not essential to this course
library(mvtnorm) # need this to simulate data from multivariate normal
library(HistData) # need this for data
```

Problem 2: Getting help

You can use the `mean()` function to compute the mean of a vector like so:

```
x1 <- c(1:10, 50)
mean(x1)
```

```
## [1] 9.545455
```

```
x1
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 50
```

However, this does not work if the vector contains NAs:

```
x1_na <- c(1:10, 50, NA)
mean(x1_na)
```

```
## [1] NA
```

Please use R documentation to find the mean after excluding NA's (hint: `?mean`)

```
# your code here
mean(x1_na, na.rm = TRUE)
```

```
## [1] 9.545455
```

Grading: Grade on correctness.

- 0.5pt

Part II: Data Manipulation

Problem 3: Basic Selection

In this question, we will practice data manipulation using a dataset collected by Francis Galton in 1886 on the heights of parents and their children. This is a very famous dataset, and Galton used it to come up with regression and correlation.

The data is available as `GaltonFamilies` in the `HistData` package. Here, we load the data and show the first few rows. To find out more information about the dataset, use `?GaltonFamilies` .

```
#install.packages("HistData")
library(HistData)
data(GaltonFamilies)
head(GaltonFamilies)
```

```
##   family father mother midparentHeight children childNum gender childHeight
## 1    001   78.5   67.0           75.43        4        1   male        73.2
## 2    001   78.5   67.0           75.43        4        2 female        69.2
## 3    001   78.5   67.0           75.43        4        3 female        69.0
## 4    001   78.5   67.0           75.43        4        4 female        69.0
## 5    002   75.5   66.5           73.66        4        1   male        73.5
## 6    002   75.5   66.5           73.66        4        2   male        72.5
```

a. **Please report the height of the 10th child in the dataset.**

```
# your code here
(tenth_child_height <- GaltonFamilies$childHeight[10])
```

```
## [1] 68
```

```
#tenth_child_height
```

b. **What is the breakdown of male and female children in the dataset?**

```
# your code here
(breakdown <- summary(GaltonFamilies$gender))
```

```
## female   male
##    453    481
```

c. **How many observations (number of rows) are in Galton's dataset? Please answer this question without consulting the R help.**

```
(observations <- nrow(GaltonFamilies))
```

```
## [1] 934
```

```
# your code here
```

d. **What is the mean height for the 1st child in each family?**

```
# your code here
first_childs <- subset(GaltonFamilies, childNum == 1)
(mean_height <- mean(first_childs$childHeight))
```

```
## [1] 69.82098
```

e. **Create a table showing the mean height for male and female children.**

```
# your code here
(mean_gener_height <- aggregate(childHeight ~ gender, data = GaltonFamilies, FUN = mea
n))
```

```
##   gender childHeight
## 1 female    64.10397
## 2   male    69.23410
```

f. **What was the average number of children each family had?**

```
# your code here
(average_childrennum_family <- aggregate(children ~ family, data = GaltonFamilies, FUN =
function(x) mean(x[1])))
```

```
##     family children
## 1      001        4
## 2      002        4
## 3      003        2
## 4      004        5
## 5      005        6
## 6      006        1
## 7      007        6
## 8      008        3
## 9      009        1
## 10     010        1
## 11     011        8
## 12     012        1
## 13     013        2
## 14     014        2
## 15     015        3
## 16     016        9
## 17     017        6
## 18     018        3
## 19     019        1
## 20     020        8
## 21     021        3
## 22     022        3
## 23     023        7
## 24     024        1
## 25     025        2
## 26     026        5
## 27     027        3
## 28     028        6
## 29     029        3
## 30     030        1
## 31     031        6
## 32     032        5
## 33     033        5
## 34     034        1
## 35     035        5
## 36     036        4
## 37     037        4
## 38     038        6
## 39     039        2
## 40     040        5
## 41     041        1
## 42     042        6
## 43     043        2
## 44     044        2
## 45     045        3
## 46     046        8
## 47     047        4
## 48     048        3
## 49     049        7
## 50     050        2
## 51     051        2
```

```
## 52     052        5
## 53     053        9
## 54     054        4
## 55     055        5
## 56     056        5
## 57     057        5
## 58     058        7
## 59     059        1
## 60     060        2
## 61     061        4
## 62     062        6
## 63     063        1
## 64     064        5
## 65     065        1
## 66     066       11
## 67     067        4
## 68     068        5
## 69     069        8
## 70     070        5
## 71     071        6
## 72     072        7
## 73     073        3
## 74     074        2
## 75     075        7
## 76     076        7
## 77     077        4
## 78     078        5
## 79     079        8
## 80     080        1
## 81     081        4
## 82     082        9
## 83     083        8
## 84     084        4
## 85     085        5
## 86     086        4
## 87     087        4
## 88     088        4
## 89     089        8
## 90     090        7
## 91     091        3
## 92     092        2
## 93     093        4
## 94     094        2
## 95     095        3
## 96     096        5
## 97     097       10
## 98     098        1
## 99     099        8
## 100    100        3
## 101    101        4
## 102    102        6
## 103    103        7
```

```
## 104    104        4
## 105    105        6
## 106    106        7
## 107    107        9
## 108    108        7
## 109    109        7
## 110    110        4
## 111    111        1
## 112    112        3
## 113    113        1
## 114    114        6
## 115    115        7
## 116    116        3
## 117    117        1
## 118    118        3
## 119    119        5
## 120    120       11
## 121    121        8
## 122    122        4
## 123    123        5
## 124    124        9
## 125    125        3
## 126    126        4
## 127    127        1
## 128    128        2
## 129    129        3
## 130    130       11
## 131    131        2
## 132    132        2
## 133    133        7
## 134    134        4
## 135    135        8
## 136    136       10
## 137   136A        8
## 138    137        4
## 139    138        5
## 140    139        1
## 141    140       10
## 142    141        8
## 143    142        4
## 144    143        1
## 145    144        4
## 146    145        8
## 147    146        6
## 148    147        1
## 149    148        1
## 150    149        5
## 151    150        1
## 152    151        2
## 153    152        1
## 154    153        5
## 155    154        1
```

```
## 156    155         7
## 157    156         4
## 158    157         1
## 159    158        10
## 160    159         5
## 161    160         1
## 162    161         8
## 163    162         6
## 164    163         5
## 165    164         4
## 166    165         3
## 167    166        11
## 168    167         4
## 169    168         8
## 170    169         3
## 171    170         5
## 172    171         1
## 173    172         8
## 174    173         9
## 175    174         5
## 176    175         6
## 177    176         8
## 178    177         5
## 179    178         1
## 180    179         2
## 181    180         6
## 182    181         7
## 183    182         1
## 184    183         4
## 185    184         1
## 186    185        15
## 187    186         4
## 188    187         1
## 189    188         4
## 190    189         5
## 191    190         9
## 192    191         2
## 193    192         6
## 194    193         6
## 195    194         2
## 196    195         3
## 197    196         4
## 198    197         5
## 199    198         7
## 200    199         7
## 201    200         1
## 202    201         2
## 203    202         2
## 204    203         3
## 205    204         2
```

g. **Convert the children's heights from inches to centimeters and store it in a column called `childHeight_cm` in the `GaltonFamilies` dataset. Show the first few rows of this dataset.**

```
# your code here
GaltonFamilies$childHeight_cm <- GaltonFamilies$childHeight * 2.54
head(GaltonFamilies)
```

```
##   family father mother midparentHeight children childNum gender childHeight
## 1    001   78.5   67.0           75.43        4        1   male        73.2
## 2    001   78.5   67.0           75.43        4        2 female        69.2
## 3    001   78.5   67.0           75.43        4        3 female        69.0
## 4    001   78.5   67.0           75.43        4        4 female        69.0
## 5    002   75.5   66.5           73.66        4        1   male        73.5
## 6    002   75.5   66.5           73.66        4        2   male        72.5
##   childHeight_cm
## 1        185.928
## 2        175.768
## 3        175.260
## 4        175.260
## 5        186.690
## 6        184.150
```

Problem 4: Spurious Correlation

```
# set seed for reproducibility
set.seed(1234)
N <- 25
ngroups <- 100000
sim_data <- data.frame(group = rep(1:ngroups, each = N),
                       X = rnorm(N * ngroups),
                       Y = rnorm(N * ngroups))
head(sim_data)
```

```
##   group          X           Y
## 1     1 -1.2070657 -0.01702494
## 2     1  0.2774292  0.40812082
## 3     1  1.0844412 -1.56429132
## 4     1 -2.3456977  1.13073145
## 5     1  0.4291247  0.82286009
## 6     1  0.5060559 -1.95833239
```

In the code above, we generate 10^{5} groups of 25 observations each. In each group, we have X and Y, where X and Y are independent normally distributed data and have 0 correlation.

a. **Find the correlation between X and Y for each group, and display the highest correlations.**

Hint: since the data is quite large and your code might take a few moments to run, you can test your code on a subset of the data first (e.g. you can take the first 100 groups like so):

```
sim_data_first_100 <- sim_data[1:100, ]
sim_data_first_100$correlation <- by(sim_data_first_100, sim_data_first_100$group, FUN =
function(x) cor(x$X, x$Y))
head(sim_data_first_100[order(sim_data_first_100$correlation, decreasing = TRUE), ])
```

```
##    group          X          Y correlation
## 4      1 -2.3456977  1.1307314  0.04792116
## 8      1 -0.5466319  0.8969592  0.04792116
## 12     1 -0.9983864  0.9420667  0.04792116
## 16     1 -0.1102855 -1.5405679  0.04792116
## 20     1  2.4158352  0.9731538  0.04792116
## 24     1  0.4595894  0.2673287  0.04792116
```

In general, this is good practice whenever you have a large dataset: If you are writing new code and it takes a while to run on the whole dataset, get it to work on a subset first. By running on a subset, you can iterate faster.

However, please do run your final code on the whole dataset.

```
# your code here
sim_data$correlation <- by(sim_data, sim_data$group, FUN = function(x) cor(x$X, x$Y))
head(sim_data[order(sim_data$correlation, decreasing = TRUE), ])
```

```
##        group          X          Y correlation
## 99655   3987  0.8891285 -1.2684142   0.8014245
## 199655  7987 -0.8480355 -1.0288452   0.8014245
## 299655 11987 -1.0732217  0.5216807   0.8014245
## 399655 15987 -0.1682145 -0.2705579   0.8014245
## 499655 19987 -1.7055112 -0.5573700   0.8014245
## 599655 23987  0.5994902  0.9540217   0.8014245
```

b. **The highest correlation is around 0.8. Can you explain why we see such a high correlation when X and Y are supposed to be independent and thus uncorrelated?**

Because we cherrypicked the highest correlations among 100,000 correlations, it is just by chance that we found a few with such a high correlation. We can see in the histogram below that most of the correlations are around the expected value of 0.

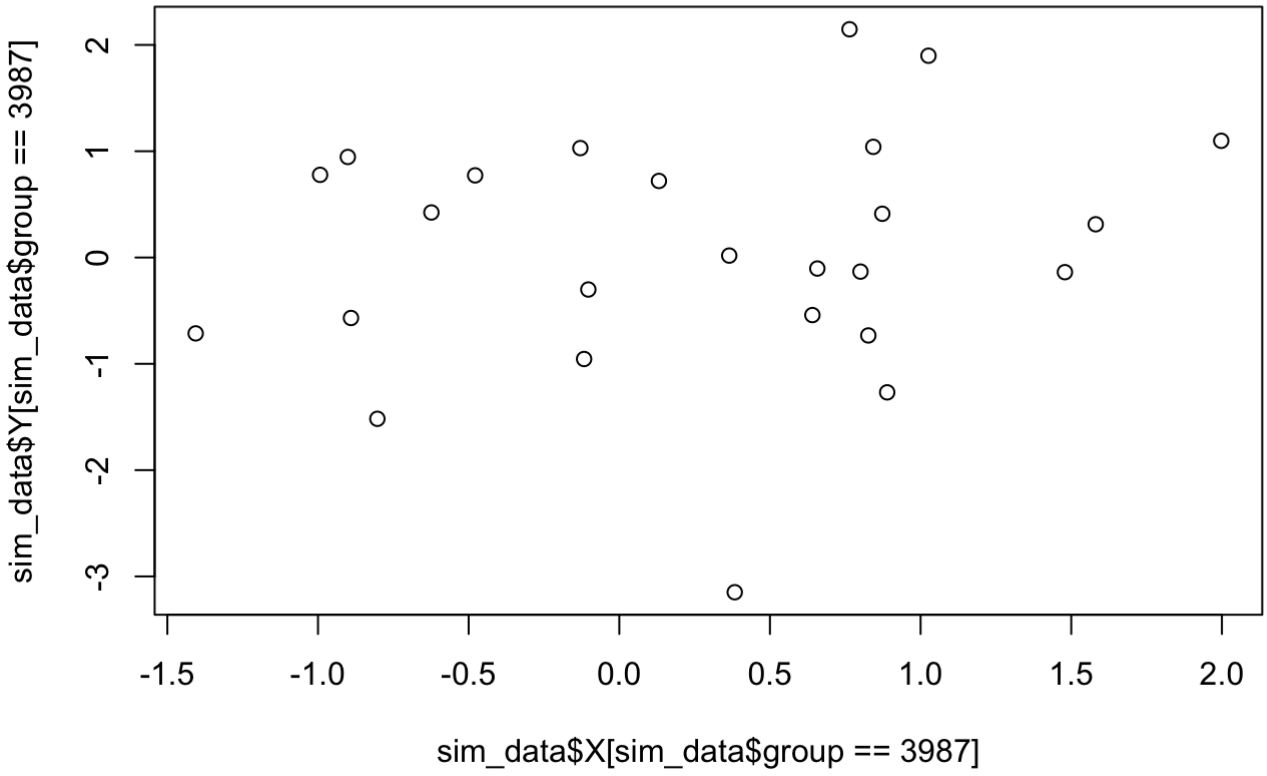
```
head(sim_data)
```

Part III: Plotting

Problem 5

Show a plot of the data for the group that had the highest correlation you found in Problem 4.

```
# your code here
plot(sim_data$X[sim_data$group == 3987], sim_data$Y[sim_data$group == 3987])
```



Grading: 1pt.

Problem 6

We generate some sample data below. The data is numeric, and has 3 columns: X, Y, Z.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
N <- 100
Sigma <- matrix(c(1, 0.75, 0.75, 1), nrow = 2, ncol = 2) * 1.5
means <- list(c(11, 3), c(9, 5), c(7, 7), c(5, 9), c(3, 11))
dat <- lapply(means, function(mu)
  rmvnorm(N, mu, Sigma))
dat <- as.data.frame(Reduce(rbind, dat)) %>%
  mutate(Z = as.character(rep(seq_along(means), each = N)))
names(dat) <- c("X", "Y", "Z")
```

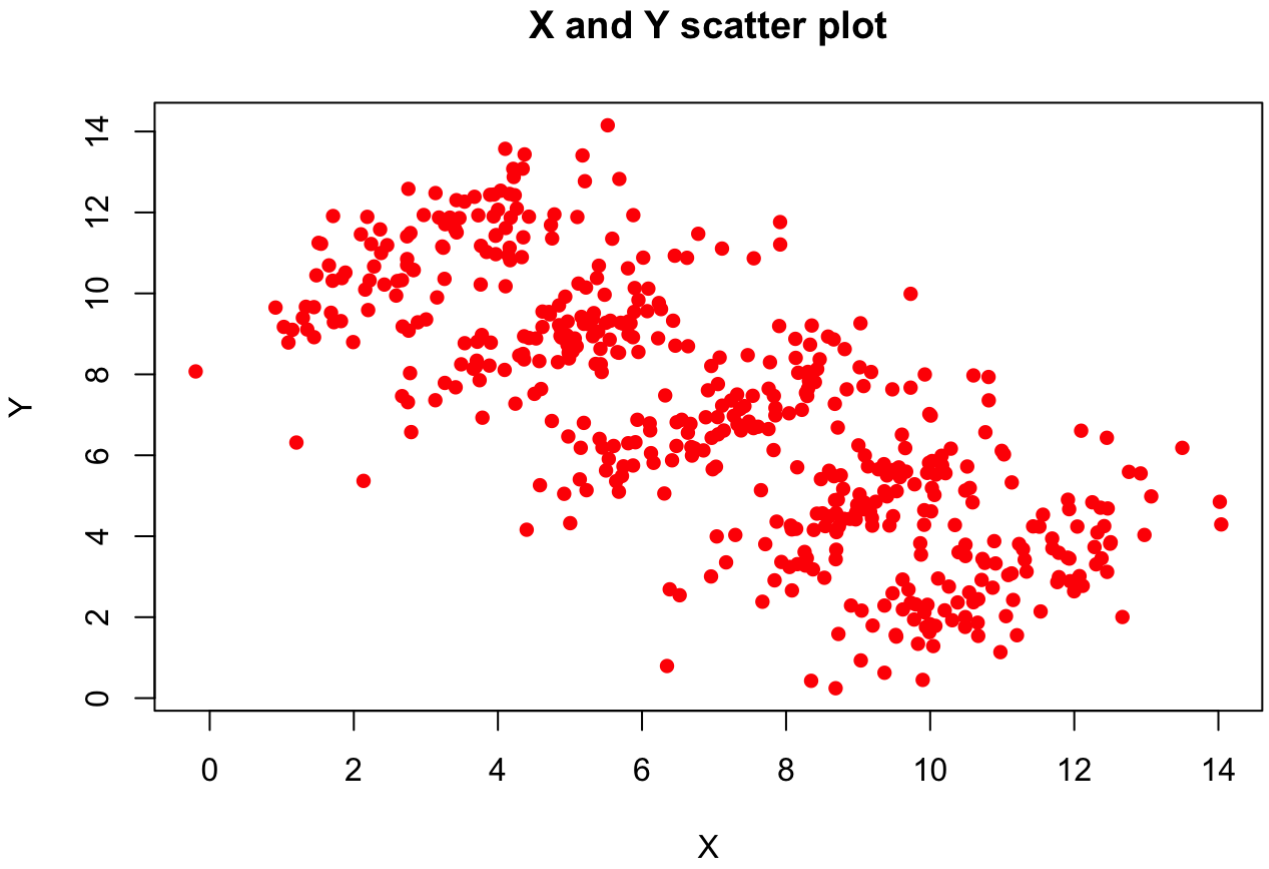
a. **Compute the overall correlation between X and Y.**

```
# your code here
(correlation <- cor(dat$X, dat$Y))
```

```
## [1] -0.7362015
```

b. **Make a plot showing the relationship between X and Y. Comment on the correlation that you see.**

```
# your code here
plot(dat$X, dat$Y, col = "red", xlab = "X", ylab = "Y",pch = 16, main = "X and Y scatter
plot")
```



Your text answer here. The correlation between X and Y is negative, meaning that as the values of X increases, the values of Y tend to decrease, and vice versa. The correlation between X and Y is negative.

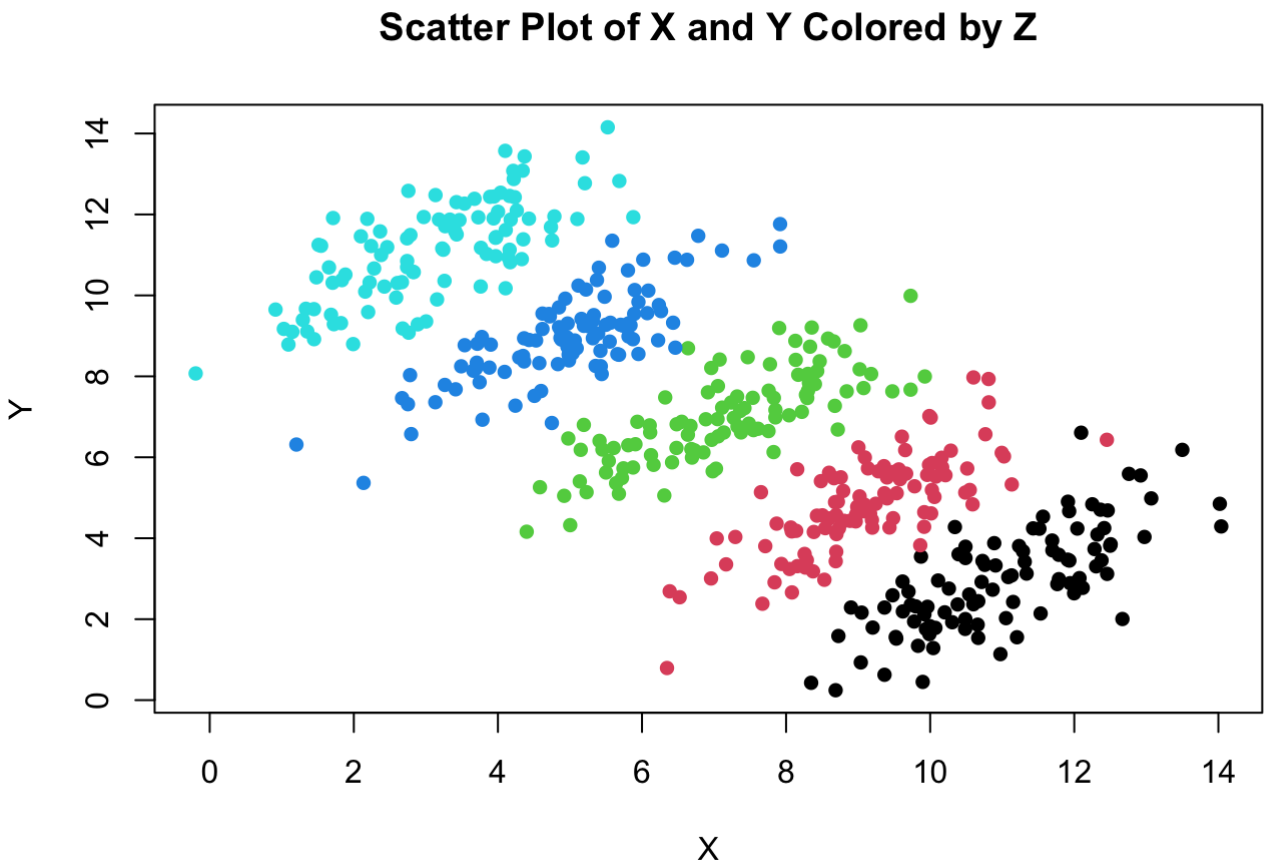
c. **Compute the correlations between X and Y for each level of Z.**

```
# your code here
(each_z_correlation <- by(dat, dat$Z, function(subset) cor(subset$X, subset$Y)))
```

```
## dat$Z: 1
## [1] 0.7431233
## ------------------------------------------------------------
## dat$Z: 2
## [1] 0.7562614
## ------------------------------------------------------------
## dat$Z: 3
## [1] 0.77199
## ------------------------------------------------------------
## dat$Z: 4
## [1] 0.784718
## ------------------------------------------------------------
## dat$Z: 5
## [1] 0.7298214
```

d. **Make a plot showing the relationship between X and Y, but this time, color the points using the value of Z. Comment on the result, especially any differences between this plot and the previous plot.**

```
# your code here
plot(dat$X, dat$Y,  col = dat$Z, xlab = "X", ylab = "Y", pch = 16, main = "Scatter Plot
of X and Y Colored by Z")
```



Your text answer here. For each level of Z, there is a positive correlation between X and Y, while the previous graph showed an overall negative correlation between X and Y

Part IV: Bash practices

Problem 7: Bash practices on Odyessy

Please answer the following question using bash commands and include those in your answer. Data are available at `/n/stat115/2020/HW1/public_MC3.maf`

Mutation Annotation Format (MAF (https://docs.gdc.cancer.gov/Data/File_Formats/MAF_Format/)) is a tab-delimited text file with aggregated mutation information. MC3.maf `/n/stat115/2020/HW1/public_MC3.maf` is a curated list of somatic mutation (https://www.britannica.com/science/somatic-mutation) occured in many patients with different types of cancers from TCGA.

Since a complete MAF file contains far more information than we need, in this problem we will focus on part of it.

```
Chromosome  Start_Position  Hugo_Symbol Variant_Classification
10  123810032   TACC2   Missense_Mutation
10  133967449   JAKMIP3 Silent
11  124489539   PANX3   Missense_Mutation
11  47380512    SPI1    Missense_Mutation
11  89868837    NAALAD2 Missense_Mutation
11  92570936    FAT3    Silent
12  107371855   MTERFD3 Missense_Mutation
12  108012011   BTBD11  Missense_Mutation
12  117768962   NOS1    5'Flank
```

In `/n/stats115/2020/HW1/MC3/public_MC3.maf`, `Chromosome` and `Start_Position` together specifies the genomics location where a location has happened. `Hogo_symbol` is the overlapping gene of that location, and `Variant_Classification` specifies how it influences downstream biological processes, e.g. transcription and translation.

Please include your bash commands and the full output from bash console with text answer to the questions.

a. How many lines are there in this file? How many times “KRAS” gene has emerged?

```
# your bash code here
file_path="/homes/asahu/liulab_home/data/stat115/2020/HW1/public_MC3.maf"
line_count=$(wc -l < "$file_path")
echo "Line count is: $line_count"

kras_count=$(grep -w -c "KRAS" "$file_path")
echo "KRAS emerged $kras_count times"
```

```
your bash output here
Line count is: 3600964
KRAS emerged 921 times
```

b. How many unique `Variant_Classification` are there in the MAF? Please count occurence of each type and sort them. Which one is the most frequent?

```
# your bash code here
file_path="/homes/asahu/liulab_home/data/stat115/2020/HW1/public_MC3.maf"
counts=$(cut -f 4 "$file_path" | sort | uniq -c | sort -nr)
count_num=$(echo "$counts" | wc -l)
most_frequent=$(echo "$counts" | head -n 1)
echo -e "There are $count_num unique Variant_Classification in the MAF, and they include
\n$counts"
echo "The most frequent is$most_frequent"
```

```
your bash output here

There are 17 unique Variant_Classification in the MAF, and they include
1921979 Missense_Mutation
 782687 Silent
 282636 3'UTR
 157232 Nonsense_Mutation
 108104 Intron
  87013 Frame_Shift_Del
  81323 5'UTR
  50617 Splice_Site
  49540 RNA
  27128 Frame_Shift_Ins
  21060 3'Flank
  15726 5'Flank
  10254 In_Frame_Del
   2723 Translation_Start_Site
   2042 Nonstop_Mutation
    899 In_Frame_Ins
      1 Variant_Classification


The most frequent is 1921979 Missense_Mutation
```

Your text answer: There are 17 unique Variant_Classification in the MAF, and they include 1921979 Missense_Mutation 782687 Silent 282636 3’UTR 157232 Nonsense_Mutation 108104 Intron 87013 Frame_Shift_Del 81323 5’UTR 50617 Splice_Site 49540 RNA 27128 Frame_Shift_Ins 21060 3’Flank 15726 5’Flank 10254 In_Frame_Del 2723 Translation_Start_Site 2042 Nonstop_Mutation 899 In_Frame_Ins 1 Variant_Classification

The most frequent is 1921979 Missense_Mutation

c. What are the top FIVE most frequent genes? Please provide the bash command and equivalent Python command. If you are a PI looking for a gene to investigate (you need to find a gene with potentially better biological significance), which gene out of the top 5 would you choose? Why?

```
# your bash code here
file_path="/homes/asahu/liulab_home/data/stat115/2020/HW1/public_MC3.maf"
gene_counts=$(cut -f 3 "$file_path" | sort | uniq -c | sort -nr)
echo -e "The top FIVE most frequent genes are: \n$gene_counts" | head -n 5
```

```
your bash output here

The top FIVE most frequent genes are:
  15171 TTN
   6875 MUC16
   4601 TP53
   3198 CSMD3
```

Equivalent python command:

```
# your python command here
```

```
your python output here
```

Yor text answer:

d. Write a bash program that determines whether a user-input year ([YYYY]) is a leap year or not (all years that are multiples of four. If the year is centennial and not divisible by 400, then it is not a leap year). The user input can be either positional or interactive. Please include the content of your shell script here and test on 1900/2000/2002, does your code run as expected?

```
# your bash code here
if [ $# -eq 1 ]; then
    year=$1
else
    # Prompt the user for input interactively
    read -p "Enter a year [YYYY]: " year
fi

# Check if the year is a leap year
if [ $((year % 4)) -eq 0 ] && [ $((year % 100)) -ne 0 ] || [ $((year % 400)) -eq 0 ]; th
en
    echo "$year is a leap year."
else
    echo "$year is not a leap year."
fi
```

Part V. High throughput sequencing read mapping

We will give you a simple example to test high throughput sequencing alignment for RNA-seq data. Normally for paired-end sequencing data, each sample will have two separate FASTQ files, with line-by-line correspondence to the two reads from the same fragment. Read mapping could take a long time, so we have created just two FASTQ files of one RNA-seq sample with only 3M fragments (2 * 3M reads) for you to run STAR instead of the full data. The files are located at `/n/stat115/2020/HW1` . The mapping will generate one single output file. Make sure to use the right parameters for single-end (SE) vs paired-end (PE) modes in BWA and STAR.

Please include the commands that you used to run BWA and STAR in your answers.

Problem 8: BWA

1. Use BWA (Li & Durbin, Bioinformatics 2009) to map the reads to the Hg38 version of the reference genome, available on Odyssey at `/n/stat115/HW2_2019/bwa_hg38_index/hg38.fasta`. In `/n/stat115/HW1_2020/BWA/loop`, you are provided with three `.fastq` files with following structure (`A_1` and `A_r` are paired sequencing reads from sample_A). Write a for loop in bash to align reads to the reference using BWA PE mode and geneterate output in SAM format.

How many rows are in each output `.sam` files? Use SAMTools on the output to find out how many reads are mappable and uniquely mappable (please also calculate the ratio). Please include full samtools output and text answer.

```
# please provide the content of your batch script (including the header)
#!/bin/bash

reference="/liulab_home/data/stat115/2020/HW1/bwa_hg38_index/hg38_GATK.fa"

for set in A B C; do
    # Input files
    left_file="${set}_1.fastq"
    right_file="${set}_r.fastq"

    # Output file
    output_file="output_${set}.sam"

    # Align reads using BWA in PE mode
    bwa mem -t 4 -R "${reference}" "${left_file}" "${right_file}" > "${output_file}"

    echo "Alignment for ${set} is complete."
done



#SAMTOOL EXECUTION

#!/bin/bash

# Path to the SAMTools executable
samtools=$(which samtools)

for set in A B C; do
    output_file="output_${set}.sam"
    num_lines=$(wc -l < "${output_file}")

    num_mapped_reads=$("${samtools}" view -c -F 4 "${output_file}")

    num_uniquely_mapped_reads=$("${samtools}" view -c -F 4 -q 30 "${output_file}")

    ratio=$(awk "BEGIN {printf \"%.2f\", ${num_uniquely_mapped_reads} / ${num_mapped_rea
ds} * 100}")

    echo "Results for output_${set}:"
    echo "Total number of lines rows in the SAM file: ${num_lines}"
    echo "Number of mapped reads: ${num_mapped_reads}"
    echo "Number of uniquely mapped reads: ${num_uniquely_mapped_reads}"
    echo "Ratio of uniquely mapped reads to mapped reads: ${ratio}%"
done
```

```
samtools output
Results for output_A:
Total number of lines rows in the SAM file: 2000054
Number of mapped reads: 1961651
Number of uniquely mapped reads: 1265864
Ratio of uniquely mapped reads to mapped reads: 64.53%

Results for output_B:
Total number of lines rows in the SAM file: 2000045
Number of mapped reads: 1961226
Number of uniquely mapped reads: 1264915
Ratio of uniquely mapped reads to mapped reads: 64.50%

Results for output_C:
Total number of lines rows in the SAM file: 2000052
Number of mapped reads: 1961725
Number of uniquely mapped reads: 1265520
Ratio of uniquely mapped reads to mapped reads: 64.51%
```

You text answer

Problem 9: STAR alignment

1. Use STAR (Dobin et al, Bioinformatics 2012) to map the reads to the reference genome, available on Odyssey at `/n/stat115/HW1_2020/STARIndex` . Use the paired-end alignment mode and generate the output in SAM format. Please include full STAR report.
   How many reads are mappable and how many are uniquely mappable?

```
# please provide the content of your sbatch script (including the header)

#Use the code below to check the readlength of the genome

seqtk sample -s100 /liulab/asahu/macaulay/code/R_codes/datas/HW1/bwa_hg38_index/hg38_GAT
K.fasta 1 | awk '{if(NR%4==2) print length($1);}' | sort -rn | head -n 1

#Use the code below to generate the STARindex files


STAR --runThreadN 8 \
     --runMode genomeGenerate \
     --genomeDir STAR_result \
     --genomeFastaFiles /homes/asahu/liulab_home/data/stat115/2020/HW1/index/hg38_raw \
     --sjdbOverhang 248956421



#Solution to Problem 9
for set in A B C; do

    left_file="/homes/asahu/liulab_home/data/stat115/2020/HW1/loop/${set}_l.fastq"
    right_file="/homes/asahu/liulab_home/data/stat115/2020/HW1/loop/${set}_r.fastq"
    output_dir="/liulab/asahu/macaulay/code/R_codes/datas/HW1/loop/STAR_result_${set}"
    reference_index="/liulab_home/data/stat115/2020/HW1/STARIndex"

    STAR --runMode alignReads \
         --genomeDir "${reference_index}" \
         --readFilesIn "${left_file}" "${right_file}" \
         --outSAMtype SAM \
         --outFileNamePrefix "${output_dir}/"


    mapped_reads=$(grep "Number of input reads" "${output_dir}/Log.final.out" | cut -f
2)
    uniquely_mapped_reads=$(grep "Uniquely mapped reads number" "${output_dir}/Log.fina
l.out" | cut -f 2)
```

```
Log file from STAR
```

Yor text answer here.

2. If you are getting a different number of mappable fragments between BWA and STAR on the same data, why?

Your text answer here.

Part VII: Dynamic programming with Python

Problem 10

Given a list of finite integer numbers, Write a python script to maximize the Z where Z is the sum of the numbers from location X to location Y on this list. Be aware, your algorithm should look at each number ONLY ONCE from left to right. Your script should return three values: the starting index location X, the ending index location Y, and Z, the sum of numbers between index X and Y (inclusive).

For example, if A=[-2, 1, 7, -4, 5, 2, -3, -6, 4, 3, -8, -1, 6, -7, -9, -5], your program should return (start_index = 1, end_index = 5, sum = 11) corresponding to [1, 7, -4, 5, 2].

Please test your program with this example and see if you can get the correct numbers.

Hint: Consider dynamic programming.

```
#Kadane's algorithm

def maximum_subarray(nums):
    maximum_sum = float('-inf')
    current_sum = 0
    start_index = 0
    end_index = 0
    temp_index = 0

    for i in range(len(nums)):
        current_sum += nums[i]

        if current_sum > maximum_sum:
            maximum_sum = current_sum
            start_index = temp_index
            end_index = i

        if current_sum < 0:
            current_sum = 0
            temp_index = i + 1

    return start_index, end_index, maximum_sum

# Test case
A = [-2, 1, 7, -4, 5, 2, -3, -6, 4, 3, -8, -1, 6, -7, -9, -5]

start_index, end_index, subarray_sum = maximum_subarray(A)

print("start_index =", start_index)
```

```
## start_index = 1
```

```
print("end_index =", end_index)
```

```
## end_index = 5
```

```
print("sum =", subarray_sum)
```

```
## sum = 11
```

OUTPUT

start_index = 1 end_index = 5 sum = 11