QCB 408/508 – Homework 1

Shannon Keenan

Instructions

Homework 1 is due by **11:59pm on March 5th.** See the syllabus for further details on homeworks. Please submit both a `.Rmd` and `.pdf` file on Blackboard. We will not accept additional files.

Make sure you have your file structure arranged such that we can compile your `.Rmd` in a folder with the provided data files files. **You are not allowed to use the function `setwd()`. If it is used, we will count the homework as unable to compile and you will lose a significant amount of points.** Unzip the homework directory and double-click the `hw1.Rmd` file. This will start RStudio and set the working directory to the homework directory, regardless of where it is on your computer.

Be sure to follow the collaboration policy outlined in the syllabus. **For this homework, there are no problems where collaborations are allowed.**

We will ask questions using bolded headings and provide further details for these questions using the boxed-quote formatting. You will be expected to answer with R code and normal text.

For all parts of this assignment, you MUST use R commands to print the output as part of your R Markdown file. You are not permitted to find the answer in the R interactive session and then copy-paste the answer into this document. Further, you are required to use LaTeX to answer questions involving math equations in the R Markdown file, when applicable.

For this assignment, we ask that you try to use functions from the libraries that are discussed in *YARP, Yet Another R Primer.*

Lastly, we advise that you compile your R Markdown file early and often, so you can catch problems before they get too tangled up with your work. Don't forget to look at the compiled PDF as well! R will not warn you when you create and submit a 5,000 page PDF because of unexpected output.

For this homework, students enrolled in QCB 408 should answer all questions.

For undergraduates: Please type your name below the honor pledge to serve as a digital signature.

I pledge my honor that I have not violated the honor code when completing this assignment.

Digitally signed:

Problem 1. Basics of R and R Markdown [10 pts]

For this section, we will briefly review a selection of the basic R skills needed for this course. This is by no means comprehensive! We assume you have read and understood Part I of *YARP, Yet Another R Primer* before completing this problem.

An example: What is 159083 divided by 619?

If we needed to provide additional information for a problem, we would do it in a boxed block like this sentence.

```
> 159083/619
[1] 257
```

We ask that you answer R questions inside chunks using the R output. If you need to provide explanations, please do it in a paragraph of text outside of the chunks (like this one) instead of in comments in the code. The reason is that the paragraph text has formatting when the R Markdown file is compiled, while long columns will run off the page when compiled. Obviously, you should feel free to use short comments to point to details of the code when needed. You can also use LaTeX to answer math questions in paragraphs outside chunks, for example $\frac{159083}{619} = 257$ or:

$$\frac{159083}{619} = 257$$

(a) Basic data types: populate a vector and a square matrix with the consecutive integers between 1 and 25. Then, use the same integers to populate a column named `N` in a data frame that also includes a character column named `A` (two columns total, you can fill the character column with whatever you want).

Please use some function that R provides to generate the integers between 1 and 25. Do not add elements manually!

```
>
> v <- seq(1,25,1)
>
> m <- matrix(v, length(v), length(v))
>
> df <- data.frame('N' = v, 'A' = rep('Hello',length(v)))
```

(b) Accessing elements: use the `[` operator to access an element of the vector you made and a row of the matrix you made. Use the `$` operator to access (by name) a column of the data frame you made.

```
>
> v[8]
[1] 8
> m[8,]
 [1] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
> df$N
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
```

(c) Dimensions: check the length of the vector you made and the dimensions of the matrix you made.

```
>
> length(v)
[1] 25
> dim(m)
[1] 25 25
```

(d) Checking classes: use the `class()` function on the three variables you made.

```
>
> class(v)
[1] "numeric"
> class(m)
[1] "matrix"
> class(df)
[1] "data.frame"
```

(e) Functions: write a function to check if an integer is a prime number, then apply this function to the first row of your matrix.

Since the numbers are small, feel free to check for prime-ness any way you'd like.

```
>
> is.prime <- function(number) {
+   if (number < 1) {
+     print('This number is not prime')
+   } else if (number %in% c(1,2)) {
+       print('This number is prime')
+   } else {
+       re <- NULL
+       for (i in 2:(number-1)) {
+         re[i] <- number %% i
+       }
+     if (0 %in% re) {
+         print('This number is not prime')
+     } else {
+         print('This number is prime')
+     }
+   }
+ }
>
> #Apply function to first row of the matrix
>
> p <- sapply(m[1,],is.prime, simplify = TRUE)
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
```

```
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
[1] "This number is prime"
> p
 [1] "This number is prime" "This number is prime" "This number is prime"
 [4] "This number is prime" "This number is prime" "This number is prime"
 [7] "This number is prime" "This number is prime" "This number is prime"
[10] "This number is prime" "This number is prime" "This number is prime"
[13] "This number is prime" "This number is prime" "This number is prime"
[16] "This number is prime" "This number is prime" "This number is prime"
[19] "This number is prime" "This number is prime" "This number is prime"
[22] "This number is prime" "This number is prime" "This number is prime"
[25] "This number is prime"
```

(f) Random numbers: use `set.seed()` to set a seed. Then, use `sample()` on the vector you created to shuffle it.

For homeworks, you will want to use `set.seed` to keep random numbers reproducible. It usually suffices to just do it once at the beginning of the homework.

```
>
> set.seed(8)
>
> v.shuffled <- v[sample(v)]
```

Problem 2. Exploratory data analysis of gene expression in breast cancer tumors. [16 pts]

We assume you have read and understood Part III of *YARP, Yet Another R Primer* before completing this problem.

Biological background. Mutations in the BRCA1 and BRCA2 genes are a hereditary risk factor in breast cancer. In this problem, we will consider gene expression data from tumors of cancer patients with these two mutations as well as "sporadic" breast cancer. Gene expression is the biological process in which genetic information encoded in DNA is turned into gene product, usually proteins. Gene products accomplish many tasks in biological systems and in general exist in different abundances. By studying genome-wide gene expression, researchers hope to catch a glimpse of the complex biological systems underlying phenotypes such as disease. This dataset is sourced from Hedenfalk, et al. (2001).

Read in the gene expression data `brca.txt` and sample information `brca_samples.txt`.

The gene expression dataset is generated from a two-channel microarray experiment with a reference sample. Gene expression in this experiment is measured from RNA purified from the tumors and reference. The values of the gene expression data are base-2 log of the gene expression ratio. More positive corresponds to more gene expression in the tumor sample. A value of 1 means there is twice as much RNA from the tumor sample when compared to RNA from the reference sample. Similarly, 2 corresponds to four times as much, -1 corresponds to one half as much, -2 corresponds to one quarter as much, etc. Note that the `gene_id` column consists of integers that serve as placeholders and do not directly map to human genes. There are 22 tumors analyzed, and the labels and mutations are documented in `samples.txt`.

```
>
> brca_expression <- read.csv('~/Desktop/QCB 508/HW1/hw1_update/brca.txt',sep='\t')
> brca_samples <- read.csv('~/Desktop/QCB 508/HW1/hw1_update/brca_samples.txt', sep='\t')
```

(a) Identify how many genes have outlier values, using the $1.5 \times \mathrm{IQR}$ definition. Pool the gene expression data among all individuals and genes to compute the IQR. Compute the IQR using `quantile` or `summary`.

```
>
> #Compute IQR
> pooled_exp <- brca_expression[,2:dim(brca_expression)[2]] %>% gather()
>
> splits <- quantile(pooled_exp$value)
>
> Q1 <- splits[2]
> Q3 <- splits[4]
>
> IQR <- Q3-Q1
>
> #Compute how many genes have outliers
>
> ng <- dim(brca_expression)[1]
> ns <- dim(brca_expression)[2]
>
> otl <- matrix(,nrow=ng, ncol=ns-1)
>
> for (i in 1:ng) {
```

```
+   otl[i,] <- brca_expression[i,2:ns]< Q1-1.5*IQR | brca_expression[i,2:ns]> Q3 +1.5*IQR
+ }
>
> g_out <- rowSums(otl)
>
> #Number of genes with outliers
> sum(g_out>0)
[1] 742
>
> #Percentage of genes with outliers
> sum(g_out>0)/ng*100
[1] 23.00062
```

742 genes, or about 23% of genes measure, have outliers.

(b) How many genes have more than 15 outliers? What does this suggest about pooling the gene expression data?

```
>
> sum(g_out>15)
[1] 32
```

32 genes have more than 15 outliers. This suggests that pooling the gene expression data may not be the best method of determining IQR?????

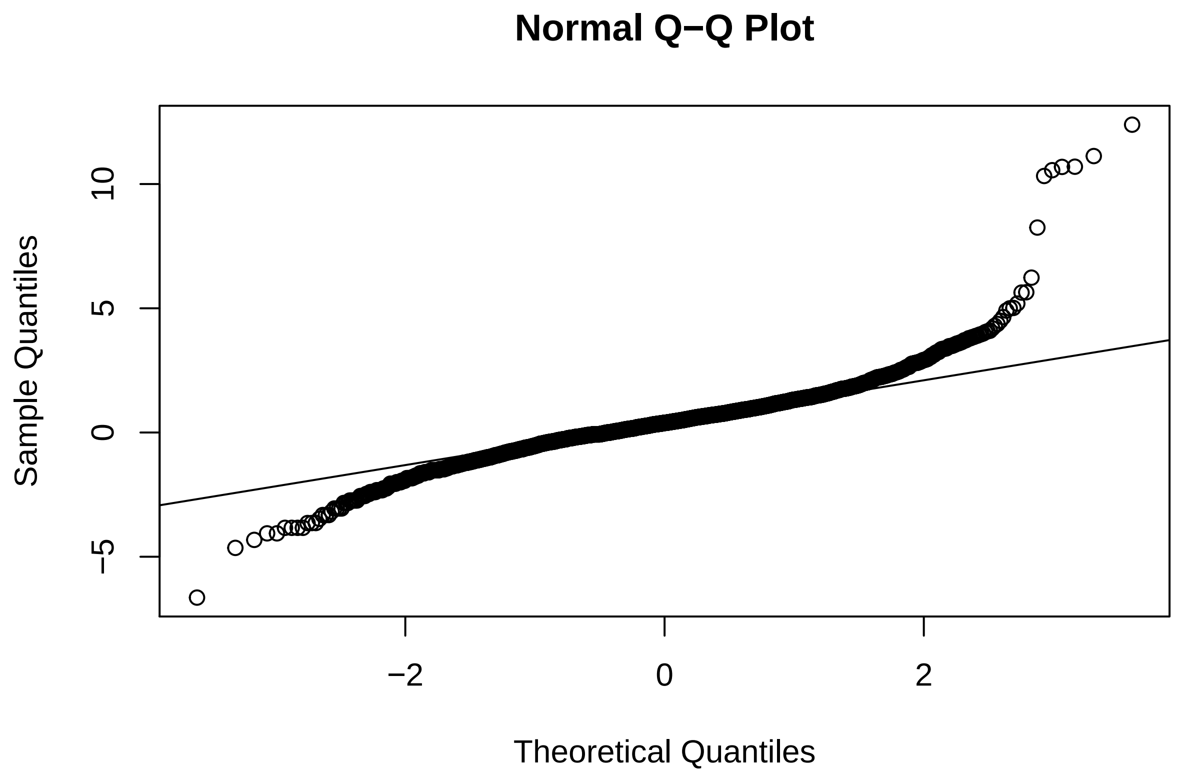
(c) Now, make the outlier calculation except on a gene-by-gene basis.

```
>
> ng <- dim(brca_expression)[1]
> ns <- dim(brca_expression)[2]
>
> otlg <- matrix(,nrow=ng, ncol=ns-1)
>
> for (i in 1:ng) {
+   splits <- as.numeric(quantile(brca_expression[i,2:ns]))
+   Q1 <- splits[2]
+   Q3 <- splits[4]
+   IQR <- Q3-Q1
+   otlg[i,] <- brca_expression[i,2:ns]< Q1-1.5*IQR | brca_expression[i,2:ns]> Q3 +1.5*IQR
+ }
>
> gg_out <- rowSums(otlg)
>
> #Number of genes with outliers
> sum(gg_out>0)
[1] 1275
>
> #Percentage of genes with outliers
> sum(gg_out>0)/ng*100
[1] 39.52263
```

Now, there are 1275 genes, or about 40%, with outliers.

(d) Use a quantile-quantile plot to assess whether or not sample s1721's gene expression vector is approximately Normal-distributed.

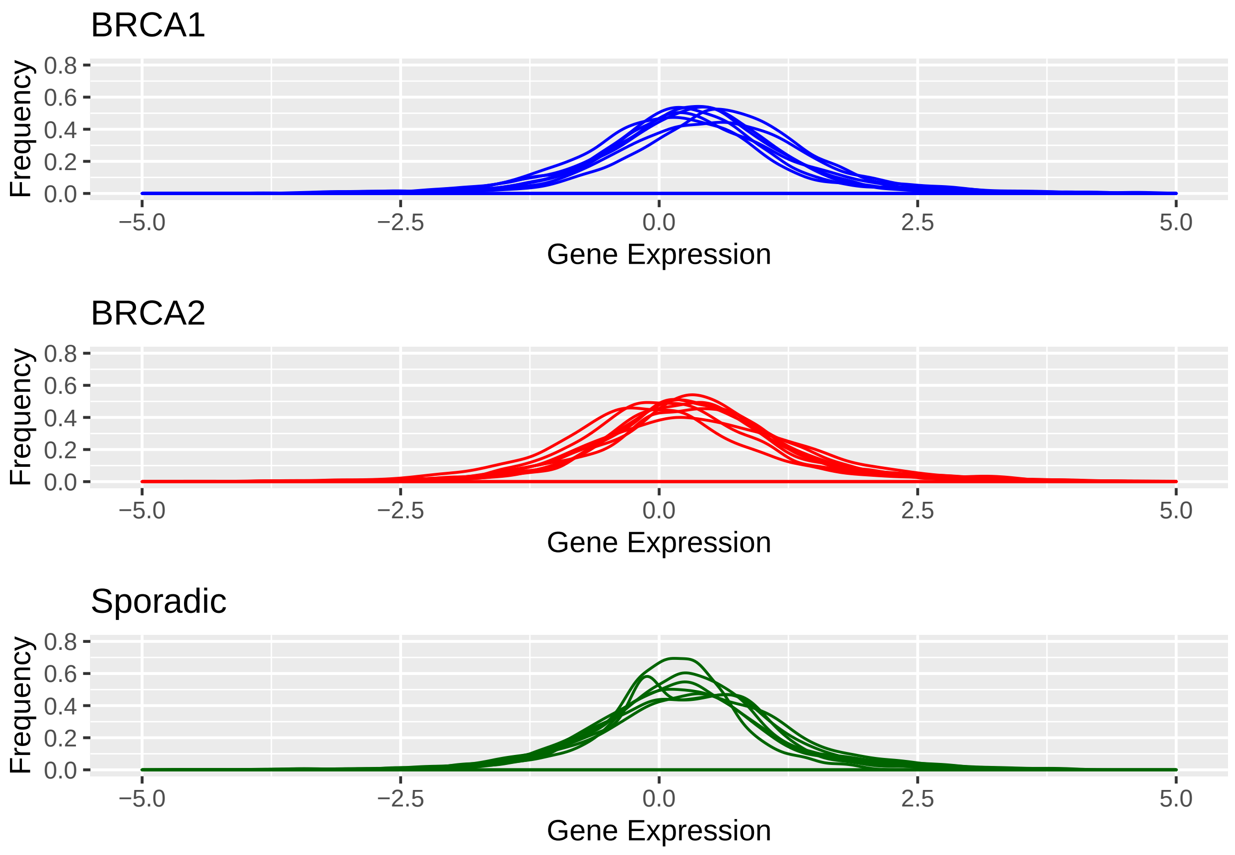
```
>
> qqnorm(brca_expression$s1721); qqline(brca_expression$s1721)
```



This gene expression data does not quite follow a normal distribution. There are some values that are more extreme that what would be expected if sampling from a Normal.

(e) Plot the distribution of gene expression values for each individual, on the same plot. Order and color the plots by mutation type.

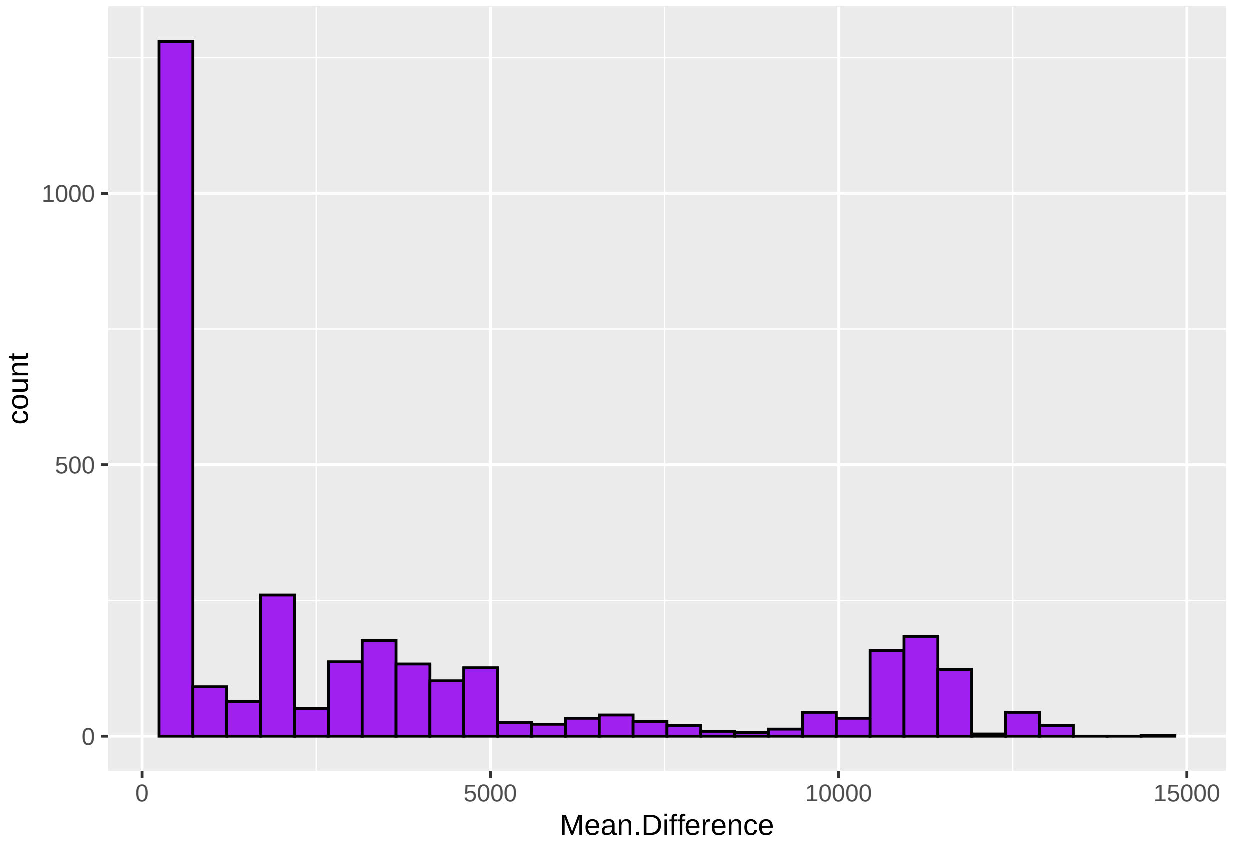
```
>
> m <- melt(brca_expression, c('gene_id'))
> m <- cbind(m, 'mutation'=brca_samples$mutation[match(m$variable,brca_samples$sample_id)])
>
> p1 <- ggplot(filter(m, mutation == 'BRCA1'), aes(x=value, category =variable)) +
+   geom_density(color='blue') +
+   xlab('Gene Expression') + ylab('Frequency')+
+   ggtitle('BRCA1')+
+   xlim(-5, 5)+ylim(0,.8)
>
> p2 <- ggplot(filter(m, mutation == 'BRCA2'), aes(x=value, category =variable)) +
+   geom_density(color='red') +
+   xlab('Gene Expression') + ylab('Frequency')+
+   ggtitle('BRCA2')+
+   xlim(-5, 5)+ylim(0,.8)
>
> p3 <- ggplot(filter(m, mutation == 'Sporadic'), aes(x=value, category =variable)) +
+   geom_density(color='darkgreen') +
+   xlab('Gene Expression') + ylab('Frequency')+
+   ggtitle('Sporadic')+
+   xlim(-5, 5)+ylim(0,.8)
>
> grid.arrange(p1,p2,p3)
Warning: Removed 2 rows containing non-finite values (stat_density).
Warning: Removed 38 rows containing non-finite values (stat_density).
Warning: Removed 1 rows containing non-finite values (stat_density).
```



(f) Effect sizes: compute the mean difference in gene expression per gene between BRCA1 and BRCA2. In other words, for each gene, compute the difference of means between patients in BRCA1 and BRCA2. Then, compute the mean, skewness, and excess kurtosis of these effect sizes. Finally, plot the histogram of effect sizes.

```
>
> #First find the mean difference in gene expression between BRCA1 and BRCA2
> BRCA1_samples <- dcast(filter(m, mutation=='BRCA1'), gene_id~variable, value.var='value')
> BRCA2_samples <- dcast(filter(m, mutation=='BRCA2'), gene_id~variable, value.var='value')
>
> B1m <- rowMeans(BRCA1_samples)
> B2m <- rowMeans(BRCA2_samples)
>
> mean_diff <- data.frame('gene_id' = BRCA1_samples$gene_id, 'Mean Difference' = abs(B2m-B1m))
>
> #Comput mean, skewness, and excess Kurtosis of effect sizes
> mean(mean_diff$Mean.Difference)
```

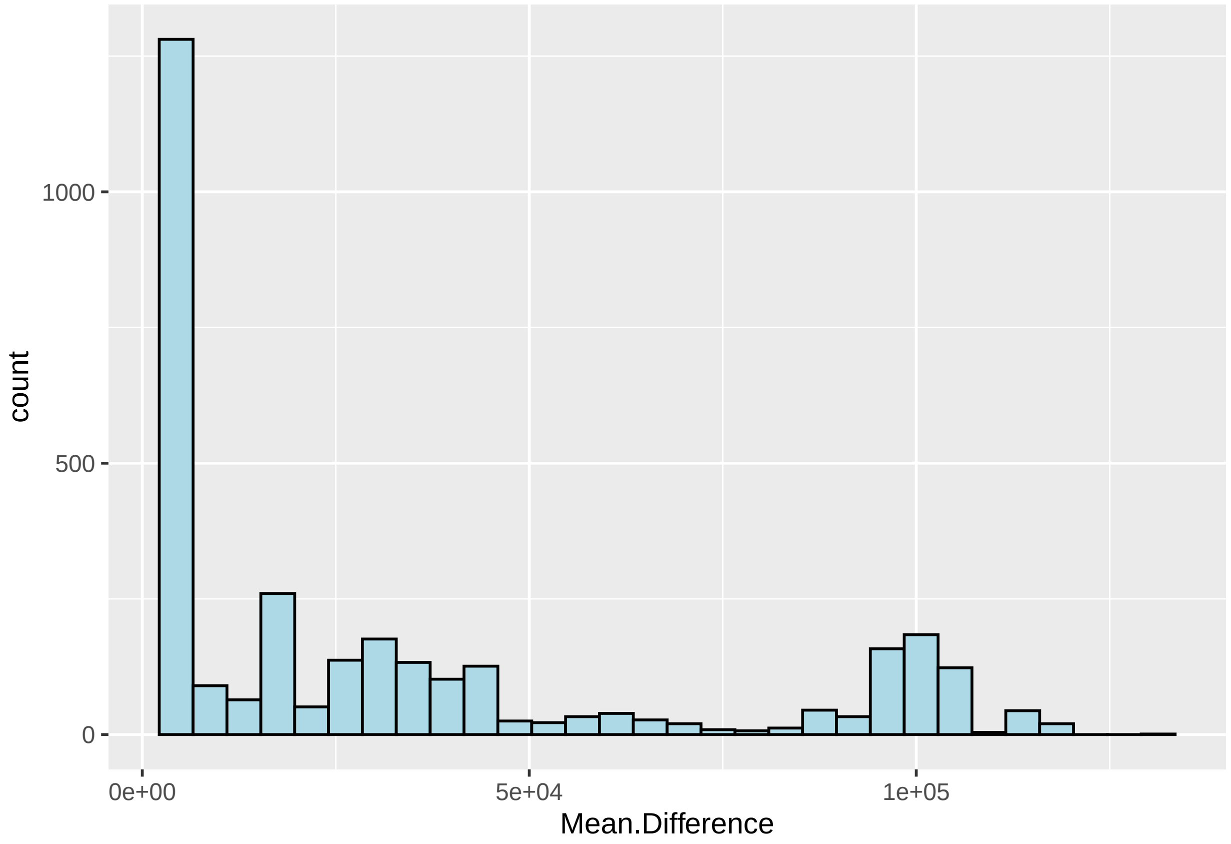
```
[1] 3738.294
> skewness(mean_diff$Mean.Difference)
[1] 1.003978
> kurtosis(mean_diff$Mean.Difference)-3
[1] -0.4726983
>
> #Plot histogram of effect sizes
> ggplot(mean_diff, aes(Mean.Difference))+geom_histogram(color='black',fill='purple')
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



(g) Permuted effect sizes: repeat the effect size analysis from the previous question, except this time, shuffle (i.e., randomly permute) the BRCA1 and BRCA2 labels. What can you infer from this plot when compared to the previous one?

```
>
> BRCA_random <- cbind(BRCA1_samples, BRCA2_samples[2:ncol(BRCA2_samples)])
>
> set.seed(8)
> B1 <- sample(1:(ncol(BRCA_random)-1),(ncol(BRCA1_samples)-1), replace=FALSE)
> B2 <- which(!(1:(ncol(BRCA_random)-1) %in% B1))
>
> B1rm <- rowMeans(BRCA_random[,B1])
> B2rm <- rowMeans(BRCA_random[,B2])
>
> mean_diff_shuff <- data.frame('gene_id' = BRCA1_samples$gene_id, 'Mean Difference' = abs(B2rm-B1rm))
>
```

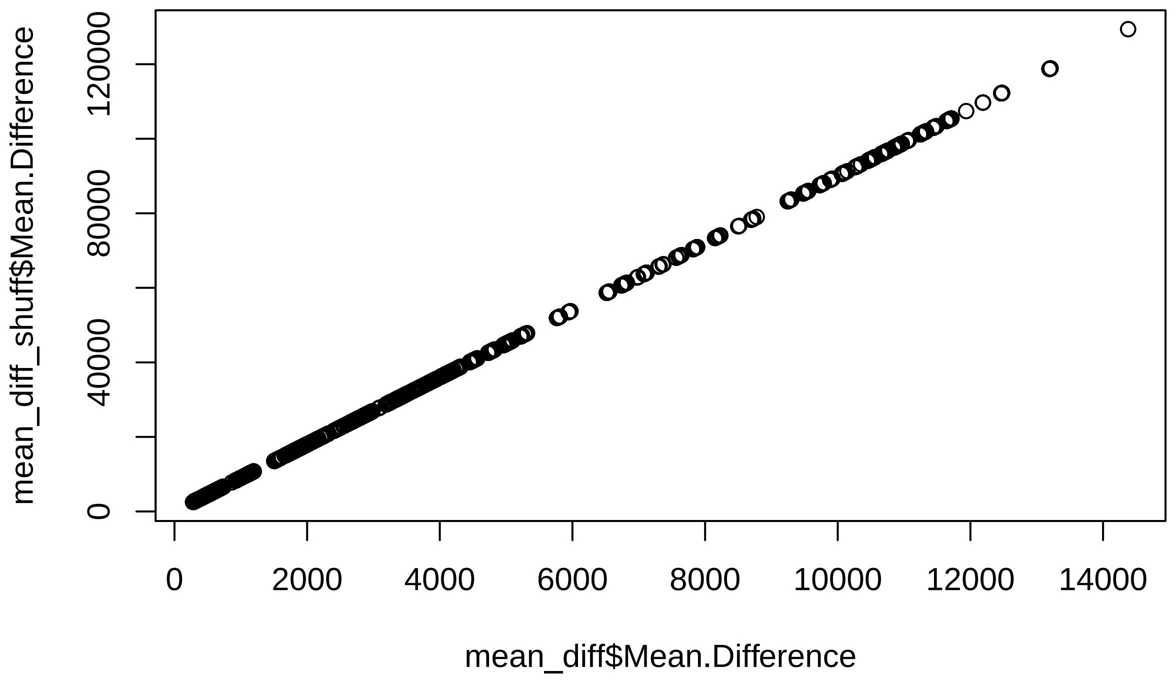
```
> #Comput mean, skewness, and excess Kurtosis of effect sizes
> mean(mean_diff_shuff$Mean.Difference)
[1] 33644.18
> skewness(mean_diff_shuff$Mean.Difference)
[1] 1.003988
> kurtosis(mean_diff_shuff$Mean.Difference)-3
[1] -0.4726769
>
> #Plot histogram of effect sizes
> ggplot(mean_diff_shuff, aes(Mean.Difference))+geom_histogram(color='black',fill='light blue')
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Not sure what this tells us about the data…

(h) Make a quantile-quantile plot of the observed effect sizes versus the shuffled label effect sizes. Describe what you observe. Does this agree with the above analyses?

```
>
> qqplot(mean_diff$Mean.Difference,mean_diff_shuff$Mean.Difference);
```



The plot shows a straight line, meaning the data is distributed the same way, whether the BRCA labels are shifted or not.

Problem 3. Random mating with a finite population size. [18 pts]

Suppose that the assumptions of Hardy-Weinberg equilibrium are satisfied in a population, except that (1) the population size is finite and (2) there may be genetic drift. Otherwise, mating is completely random, there is no mutation, migration, or selection, and each generation is discrete and non-overlapping.

There are n total diploid individuals in the population. At a particular locus, there are alleles A and B. At generation 0, there are x_0 B alleles ($2n - x_0$ A alleles) and the proportion of B alleles is $p_0 = \frac{x_0}{2n}$. The values x_0 and p_0 are non-random quantities, so they can be treated as parameters. For each subsequent generation, there are exactly n individuals produced, implying each generation continues to have exactly $2n$ alleles. For $t = 0, 1, 2, \ldots$, we will let X_t be the random number of B alleles and P_t be the random proportion of B alleles.

(a) Determine the distribution of X_1 in terms of x_0 (and n, of course). For any $t > 0$, determine the distribution of $X_{t+1}|X_t$.

The chances of an individual in generation 0 passing along a B allele is p_0 or $\frac{x_0}{2n}$. That means that the next generation will recieve 2 independent alleles that each have a p_0 change of being allele B. So the offspring in generation 1 has 2 independant instances of a Bernoulli, or is distributed as Binomial(2, p_0):

$$Z_1 \sim \text{Binomial}(2, p_0)$$

That means that:

$$Pr(X_1 = B|Z = k) = \begin{cases} 0 & k = 0 \\ 0.5 & k = 1 \\ 1 & k = 2 \end{cases}$$

So then:

$$Pr(X_1) = \sum_{k=0}^{k=2} Pr(X_1 = B|Z = k)Pr(Z = k) = 0 * (1 - p_0)^2 + 0.5 * 2p_0(1 - p_0) + 1 * p_0^2 = \mathbf{p_0}$$

This is just a Bernoulli with the same rate as success as in the 0 generation. Therefore,

$$X_1 \sim \text{Bernoulli}\left(\frac{x_o}{2n}\right)$$

Since this will be the same for every generation:

$$X_{t+1} \sim \text{Bernoulli}\left(\frac{x_t}{2n}\right)$$

(b) Calculate $\text{E}[X_t]$ in terms of x_0, and calculate $\text{E}[P_t]$ in terms of p_0.

(c) Calculate $\text{Var}[X_1]$ in terms of p_0 and calculate $\text{Var}[X_{t+1}|X_t]$ for $t > 0$ in terms of P_t. Finally, calculate $\text{Var}[X_t]$ and $\text{Var}[P_t]$ in terms of p_0.

(d) Determine $\lim_{t\to\infty} \text{Var}[P_t]$.

(e) It is a fact that under this model, with probability 1 it is eventually that case that $P_t = 1$ or $P_t = 0$. It can be calculated that $P_t = 1$ eventually with probability p_0 and $P_t = 0$ eventually with probability $1 - p_0$. Explain why the results from parts (b) and (d) makes sense given these fixation probabilities.

(f) Carry out a simulation where P_t is simulated for $t = 0, 1, \ldots, 200$ for 40 replications. Plot t on the x-axis versus P_t on the y-axis, where the results for all 40 replcations are shown in a single plot. Do this for all combinations of $n = 50, 200, 1000$ and $p_0 = 0.01, 0.1, 0.5$ (for a total of 9 plots, which you can arrange in a 3 × 3 grid, if you want). Summarize your observations about the relationship between n and the trajectories of P_t, and also about the relationship between p_0 and the trajectories of P_t.

Problem 4. Models of RNA-Seq data. [16 pts]

In this problem, we will explore RNA-Seq gene expression data. The file `rna_seq.txt` contains a matrix of counts, where the rows are genes and the columns are observations. The observations are sampled from the same biological condition. One strategy to analyze RNA-Seq data is to transform the counts y_{ij} as follows:

$$x_{ij} = \log_2 \left(\frac{y_{ij} + 0.5}{d_j} \times 10^6 \right),$$

where d_j is the read depth (total number of counts) for observation j.

(a) Load `rna_seq.txt`. Filter genes with fewer than 25 total counts. Transform the data as above, and report d_j for all observed gene-count vectors. Why do we divide the per-gene read counts y_{ij} by d_j?

(b) Calculate the gene-wise sample means and sample variances of both the original counts y_{ij} and the log-transformed counts x_{ij}. Create five plots: (1) the gene-wise sample means versus the sample variances of the original counts; (2) log-transformed gene-wise sample means versus the log-transformed sample variances of the original counts; (3) the gene-wise sample means versus the sample variances of the transformed counts; (4) a histogram of the y_{ij} values; and (5) a histogram of the x_{ij} values. What do you observe?

(c) In class we found that the variance of the untransformed counts are quadratic with respect to the mean, $\mathrm{E}[Y_{ij}] = \mu_{ij}$:

$$\mathrm{Var}[Y_{ij}] = \mu_{ij} + \mu_{ij}^2 \phi_i.$$

Suppose we want to (approximately) calculate $\mathrm{Var}[X_{ij}]$. One strategy is to apply a first order Taylor expansion:

$$X_{ij} = g(Y_{ij}) \approx g(\mu_{ij}) + g'(\mu_{ij})(Y_{ij} - \mu_{ij}).$$

Based on this, calculate an approximation of $\mathrm{Var}[X_{ij}]$ in terms of the above $\mathrm{Var}[Y_{ij}]$.

(d) Finally, approximate the squared biological coefficient of variation in the study, ϕ. Here, assume that it is the same for every gene, i.e. $\phi_1 = \phi_2 = \ldots = \phi_m$ where there are m genes in the dataset. Note: You may also assume that for large y, $\log(y + c) \approx \log(y)$.

Problem 5. Central Limit Theorem (CLT) [10 pts]

(a) CLT shown through simulation.

Suppose that $X_1, X_2, \ldots, X_{70} \sim \text{Exponential}(3)$. Show through simulation and visualizations that

$$\frac{\overline{X} - 1/3}{\sqrt{1/9 \times 1/70}}$$

is distributed approximately Normal$(0, 1)$, where $\overline{X} = \frac{1}{70}\sum_{i=1}^{70} X_i$.

(b) Poisson approximation to Binomial.

Suppose that $X \sim \text{Binomial}(1000, 0.01)$. Show through simulation that X is distributed approximately Poisson(λ) where $\lambda = 1000 \times 0.01$. Explain why this approximation works.

Problem 6. Random Variables [15 pts]

(a) Standardized random variable.

Suppose X is a random variable with population mean $\mathrm{E}[X]$ and population variance $\mathrm{Var}[X]$. Let

$$Z = \frac{X - \mathrm{E}[X]}{\sqrt{\mathrm{Var}[X]}}.$$

Show that $\mathrm{E}[Z] = 0$ and $\mathrm{Var}[Z] = 1$.

(b) Covariance under independence.

Suppose that X and Y are independent random variables. Show that $\mathrm{Cov}(X, Y) = 0$.

(c) Expected value of the sample variance.

Suppose $X_1, X_2, \ldots, X_n$ are i.i.d. random variables with $\mathrm{E}[X_i] = \mu$ and $\mathrm{Var}[X_i] = \sigma^2$. Let

$$S^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n-1},$$

where $\overline{X} = \sum_{i=1}^{n} X_i / n$. Show that $\mathrm{E}[S^2] = \sigma^2$.

Problem 7. EFDs and Fisher Information [15 pts]

(a) Write the $\text{Binomial}(n, p)$ **distribution in exponential family form. Identify the natural parameter,** η**, sufficient statistic,** $T(x)$**, and cumulant generating function,** $A(\eta)$**.**

(b) Compute the Fisher information for $X \sim \text{Binomial}(n, p)$ **in terms of** p**, and then compute the Fisher information in terms of its natural parameter,** η**. Determine whether they are equal or not.**

(c) For each value $n = 1, 2, 4, 8$**, calculate the Fisher information for** $p = 0.05, 0.10, \ldots, 0.95$**. Plot** p **versus the Fisher information for each value of** n **(color each value of** n **differently) and explain what you observe.**

Always include the session information.

```
> sessionInfo()
R version 3.6.2 (2019-12-12)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS Mojave 10.14.6

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] moments_0.14   gridExtra_2.3  ggplot2_3.2.1  reshape2_1.4.3 dplyr_0.8.4
[6] tidyr_1.0.2    knitr_1.28

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.3       magrittr_1.5     munsell_0.5.0    tidyselect_1.0.0
 [5] colorspace_1.4-1 R6_2.4.1         rlang_0.4.4      stringr_1.4.0
 [9] plyr_1.8.5       tools_3.6.2      grid_3.6.2       gtable_0.3.0
[13] xfun_0.12        withr_2.1.2      ellipsis_0.3.0   htmltools_0.4.0
[17] lazyeval_0.2.2   yaml_2.2.1       digest_0.6.24    assertthat_0.2.1
[21] tibble_2.1.3     lifecycle_0.1.0  crayon_1.3.4     farver_2.0.3
[25] purrr_0.3.3      vctrs_0.2.2      glue_1.3.1       evaluate_0.14
[29] rmarkdown_2.1    labeling_0.3     stringi_1.4.6    compiler_3.6.2
[33] pillar_1.4.3     scales_1.1.0     pkgconfig_2.0.3
```