

Assessment_4_R_Data_Analysis

Natasha Ford, Kahla Finch, Devanshee Alpeshkumar Gandhi

2024-09-25

Part 1

Gene Expression Data

The file “gene_expression.tsv” contains RNA-seq count data for three samples of interest.

Download Data files

```
URL_P1="https://raw.githubusercontent.com/ghazkha/Assessment4/refs/heads/main/gene_expression.tsv"
download.file(URL_P1,destfile="Gene_Expression.csv")
```

Gene expression file was downloaded using the download.file() command

Q1

Gene Expression file read in as table.

```
Gene_Expression <- read.table("https://raw.githubusercontent.com/ghazkha/Assessment4/refs/heads/main/gene_expression.tsv",
                              header=TRUE, as.is=TRUE)
head(Gene_Expression)
```

	GTEX.1117F.0226.SM.5GZZ7	GTEX.1117F.0426.SM.5EGHI
## ENSG00000223972.5_DDX11L1	0	0
## ENSG00000227232.5_WASH7P	187	109
## ENSG00000278267.1_MIR6859-1	0	0
## ENSG00000243485.5_MIR1302-2HG	1	0
## ENSG00000237613.2_FAM138A	0	0
## ENSG00000268020.3_OR4G4P	0	1
##	GTEX.1117F.0526.SM.5EGHJ	
## ENSG00000223972.5_DDX11L1	0	
## ENSG00000227232.5_WASH7P	143	
## ENSG00000278267.1_MIR6859-1	1	
## ENSG00000243485.5_MIR1302-2HG	0	
## ENSG00000237613.2_FAM138A	0	
## ENSG00000268020.3_OR4G4P	0	

Required package, “readr” installed and loaded using install.packages() and library() command. Data read in as table with the read.table command, the row.names=1 argument creates rows with the gene identifiers as the names. The head() function displays the first 6 lines of the table.

Q2

Mean data of Gene expression added to table as new column

```
Gene_Expression$Mean_value <- rowMeans(Gene_Expression, na.rm = TRUE)
head(Gene_Expression)
```

```
##                                GTEX.1117F.0226.SM.5GZZ7 GTEX.1117F.0426.SM.5EGHI
## ENSG00000223972.5_DDX11L1                        0                        0
## ENSG00000227232.5_WASH7P                        187                      109
## ENSG00000278267.1_MIR6859-1                      0                        0
## ENSG00000243485.5_MIR1302-2HG                      1                        0
## ENSG00000237613.2_FAM138A                      0                        0
## ENSG00000268020.3_OR4G4P                      0                        1
##                                GTEX.1117F.0526.SM.5EGHJ Mean_value
## ENSG00000223972.5_DDX11L1                        0      0.0000000
## ENSG00000227232.5_WASH7P                      143 146.3333333
## ENSG00000278267.1_MIR6859-1                      1    0.3333333
## ENSG00000243485.5_MIR1302-2HG                      0    0.3333333
## ENSG00000237613.2_FAM138A                      0    0.0000000
## ENSG00000268020.3_OR4G4P                      0    0.3333333
```

rowMeans() is used to calculate the mean value of each row in the data file, those values are then subset using Gene_Expression\$Mean_value <- into the file. The head() function displays the first 6 lines of the table.

Q3

10 genes with the highest mean expression

```
Sorted_Genes <- Gene_Expression[order(-Gene_Expression$Mean_value), ]
Top10_Mean <- head(Sorted_Genes, 10)
print(Top10_Mean)
```

```
##                                GTEX.1117F.0226.SM.5GZZ7 GTEX.1117F.0426.SM.5EGHI
## ENSG00000198804.2_MT-CO1                        267250                    1101779
## ENSG00000198886.2_MT-ND4                        273188                    991891
## ENSG00000198938.2_MT-CO3                        250277                    1041376
## ENSG00000198888.2_MT-ND1                        243853                    772966
## ENSG00000198899.2_MT-ATP6                      141374                    696715
## ENSG00000198727.2_MT-CYB                      127194                    638209
## ENSG00000198763.3_MT-ND2                      159303                    543786
## ENSG00000211445.11_GPX3                      464959                    39396
## ENSG00000198712.1_MT-CO2                      128858                    545360
## ENSG00000156508.17_EEF1A1                      317642                    39573
##                                GTEX.1117F.0526.SM.5EGHJ Mean_value
## ENSG00000198804.2_MT-CO1                      218923      529317.3
## ENSG00000198886.2_MT-ND4                      277628      514235.7
## ENSG00000198938.2_MT-CO3                      223178      504943.7
## ENSG00000198888.2_MT-ND1                      194032      403617.0
## ENSG00000198899.2_MT-ATP6                      151166      329751.7
## ENSG00000198727.2_MT-CYB                      141359      302254.0
## ENSG00000198763.3_MT-ND2                      149564      284217.7
## ENSG00000211445.11_GPX3                      306070      270141.7
## ENSG00000198712.1_MT-CO2                      122816      265678.0
## ENSG00000156508.17_EEF1A1                      339347      232187.3
```

The `order()` function sorts the data based on the mean values, the `-` argument must be used to sort in descending order. `head(sorted)` subsets the top 10 rows into `Top10_Mean` which can be displayed using `print()`.

Q4

Number of genes with a mean <10

```
Low_Mean <- subset(Gene_Expression, Mean_value < 10)
nrow(Low_Mean)
```

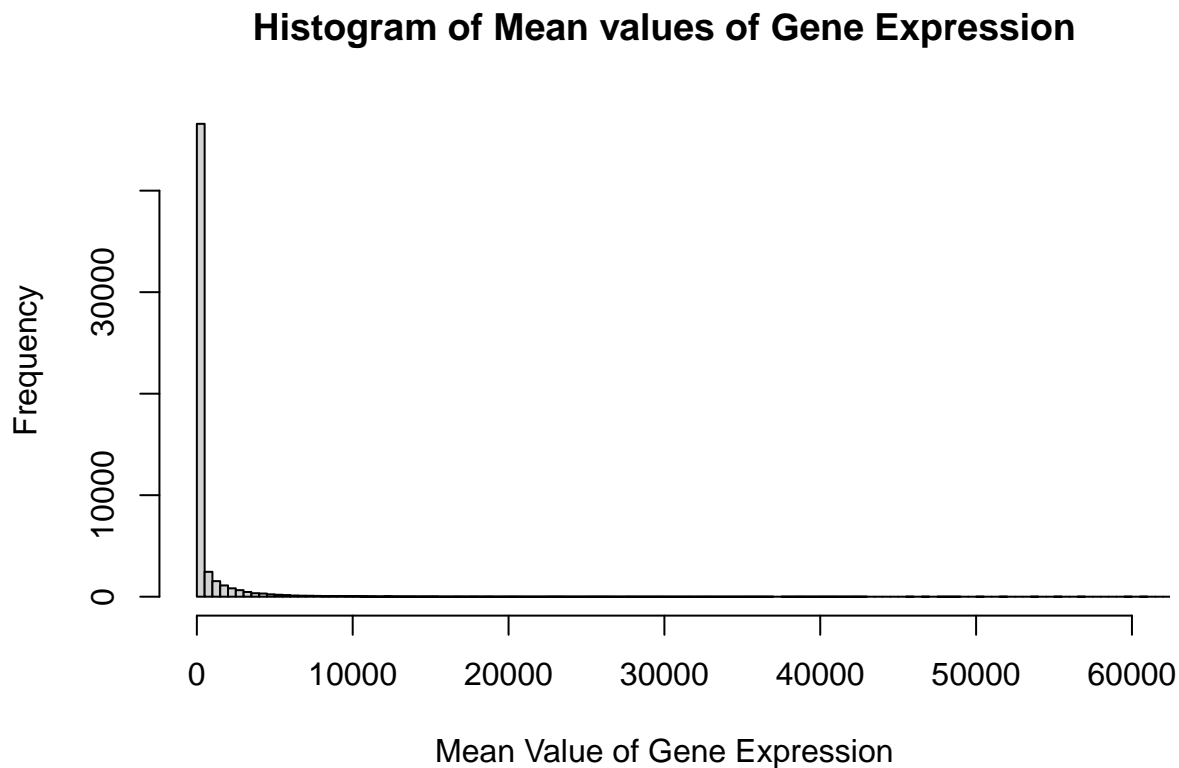
```
## [1] 35988
```

Subset taken of gene expression data with a mean value of less than 10. The rows in the resulting data are counted to get the amount of genes with a mean under 10.

Q5

Histogram of Mean Values

```
hist(Gene_Expression$Mean_value,breaks=1000, xlim = c(0,60000), xlab = "Mean Value of Gene Expression",
```



The Histogram shown in the above figure displays the mean values of the gene expression data. The histogram was created with the `hist()` command, which produces a histogram plot.

Q6

Growth Data

The file “growth_data.csv” contains tree circumference measurements for trees growing at both a control site and a treatment site. The trees were planted 20 years ago.

Importing growth data csv file into R

```
growth_data_URL <- "https://raw.githubusercontent.com/ghazkha/Assessment4/refs/heads/main/growth_data.csv"
download.file(growth_data_URL, destfile="growth_data")

growth_data <- read.csv("growth_data")
head(growth_data)
```

```
##      Site TreeID Circumf_2005_cm Circumf_2010_cm Circumf_2015_cm
## 1 northeast  A012             5.2             10.1             19.9
## 2 southwest  A039             4.9             9.6             18.9
## 3 southwest  A010             3.7             7.3             14.3
## 4 northeast  A087             3.8             6.5             10.9
## 5 southwest  A074             3.8             6.4             10.9
## 6 northeast  A008             5.9             10.0             16.8
##      Circumf_2020_cm
## 1                 38.9
## 2                 37.0
## 3                 28.1
## 4                 18.5
## 5                 18.4
## 6                 28.4
```

The growth data csv file was converted into a R object, via the download.file command, the first six outputs are shown via the head() command.

Column names of Growth Data

```
colnames(growth_data)

## [1] "Site"           "TreeID"          "Circumf_2005_cm" "Circumf_2010_cm"
## [5] "Circumf_2015_cm" "Circumf_2020_cm"
```

To determine the column names of the growth data file, the command colnames() was used, and the column names of “Site”, “TreeID”, “Circumf_2005_cm”, “Circumf_2010_cm”, “Circumf_2015_cm” and “Circumf_2020_cm” can be seen in the output above.

Q7

Mean and standard deviation at northeast and southwest sites at 2005 and 2020

```
growth_northeast_2005 <- subset(growth_data, Site == "northeast", select = c("Circumf_2005_cm"))
head(growth_northeast_2005)
```

Mean and standard deviation of northeast site at 2005

```
##      Circumf_2005_cm
## 1                 5.2
## 4                 3.8
```

```
## 6          5.9
## 7          4.4
## 8          5.3
## 12         3.5
```

```
mean_growth_northeast_2005 <- mean(growth_northeast_2005$Circumf_2005_cm)
mean_growth_northeast_2005
```

```
## [1] 5.292
```

```
SD_growth_northeast_2005 <- sd(growth_northeast_2005$Circumf_2005_cm)
SD_growth_northeast_2005
```

```
## [1] 0.9140267
```

To determine the mean of 5.292 cm and standard deviation of 0.914 cm at the northeast site at 2005, the complete growth data was subsetting into northeast 2005 and mean() and SD() commands were used.

```
growth_northeast_2020 <- subset(growth_data, Site == "northeast", select = c("Circumf_2020_cm"))
head(growth_northeast_2020)
```

Mean and standard deviation of northeast site at 2020

```
##      Circumf_2020_cm
## 1          38.9
## 4          18.5
## 6          28.4
## 7          50.0
## 8          25.8
## 12         26.0
```

```
mean_growth_northeast_2020 <- mean(growth_northeast_2020$Circumf_2020_cm)
mean_growth_northeast_2020
```

```
## [1] 54.228
```

```
SD_growth_northeast_2020 <- sd(growth_northeast_2020$Circumf_2020_cm)
SD_growth_northeast_2020
```

```
## [1] 25.22795
```

To determine the mean of 54.228 cm and standard deviation of 25.22795 cm at the northeast site at 2020, the complete growth data was subsetting into northeast 2020 and mean() and SD() commands were used.

```
growth_southwest_2005 <- subset(growth_data, Site == "southwest", select = c("Circumf_2005_cm"))
head(growth_southwest_2005)
```

Mean and standard deviation of southwest site at 2005

```
##      Circumf_2005_cm
## 2          4.9
## 3          3.7
## 5          3.8
## 9          7.1
## 10         3.8
## 11         5.4
```

```
mean_growth_southwest_2005 <- mean(growth_southwest_2005$Circumf_2005_cm)
mean_growth_southwest_2005
```

```
## [1] 4.862
```

```
SD_growth_southwest_2005 <- sd(growth_southwest_2005$Circumf_2005_cm)
SD_growth_southwest_2005
```

```
## [1] 1.147471
```

To determine the mean of 4.862 cm and standard deviation of 1.147471 cm at the southwest site at 2005, the complete growth data was subsetting into southwest 2005 and mean() and SD() commands were used.

```
growth_southwest_2020 <- subset(growth_data, Site == "southwest", select = c("Circumf_2020_cm"))
head(growth_southwest_2020)
```

Mean and standard deviation of southwest site at 2020

```
##      Circumf_2020_cm
## 2                37.0
## 3                28.1
## 5                18.4
## 9                34.2
## 10               28.4
## 11               40.5
```

```
mean_growth_southwest_2020 <- mean(growth_southwest_2020$Circumf_2020_cm)
mean_growth_southwest_2020
```

```
## [1] 45.596
```

```
SD_growth_southwest_2020 <- sd(growth_southwest_2020$Circumf_2020_cm)
SD_growth_southwest_2020
```

```
## [1] 17.87345
```

To determine the mean of 45.596 cm and standard deviation of 17.87345 cm at the southwest site at 2020, the complete growth data was subsetting into southwest 2020 and mean() and SD() commands were used.

Table summary of mean and standard deviation

```
mean_SD_table <- data.frame(col1 = c((mean_growth_northeast_2005),(SD_growth_northeast_2005)),
                             col2 = c((mean_growth_northeast_2020),(SD_growth_northeast_2020)),
                             col3 = c((mean_growth_southwest_2005),(SD_growth_southwest_2005)),
                             col4 = c((mean_growth_southwest_2020),(SD_growth_southwest_2020)))
colnames(mean_SD_table) <- c("Northeast 2005","Northeast 2020","Southwest 2005","Southwest 2020")
rownames(mean_SD_table) <- c("Mean", "Standard Deviation")
mean_SD_table
```

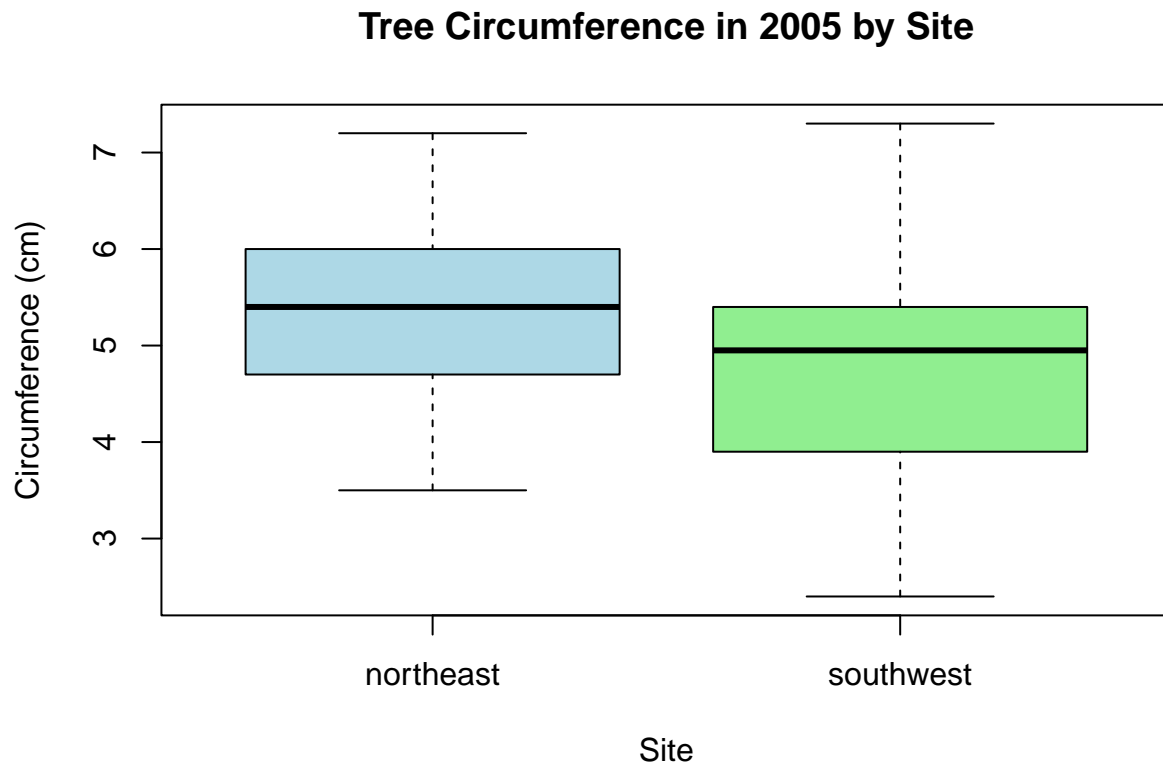
```
##      Northeast 2005 Northeast 2020 Southwest 2005 Southwest 2020
## Mean              5.2920000      54.22800      4.862000      45.59600
## Standard Deviation 0.9140267      25.22795      1.147471      17.87345
```

To provide the table above, a data frame was created with the mean and standard deviation results from the northeast and southwest sites and 2005 and 2020.

Q8

Box plot for tree circumference in 2005 (start of the study)

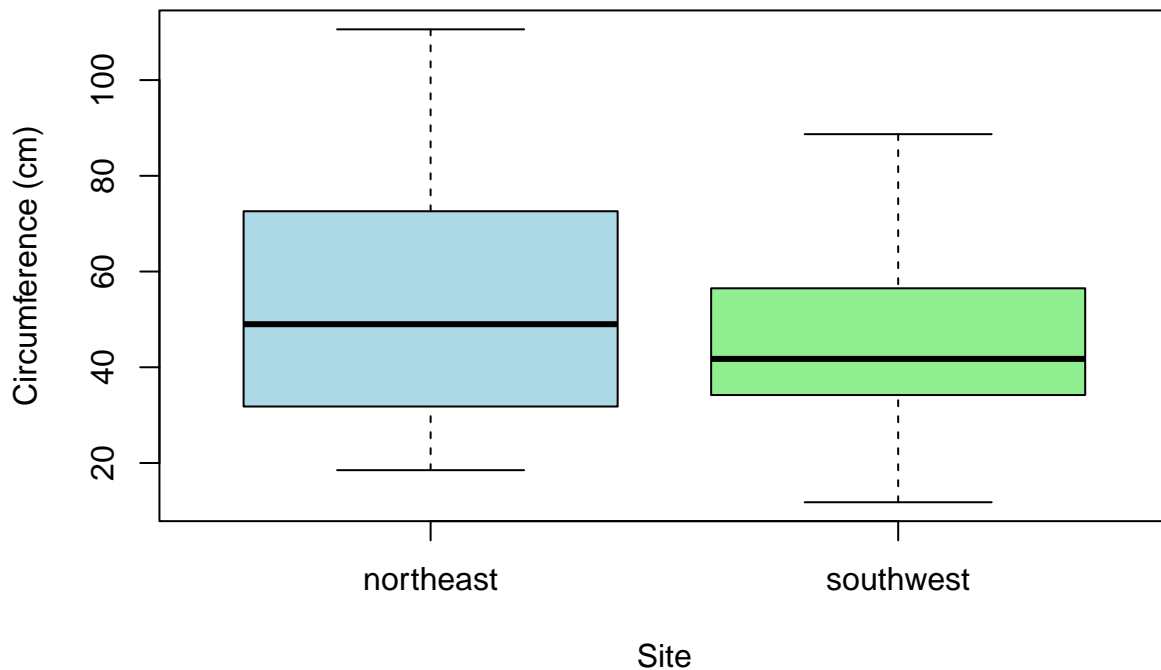
```
boxplot(growth_data$Circumf_2005_cm ~ growth_data$Site,  
        main = "Tree Circumference in 2005 by Site",  
        xlab = "Site", ylab = "Circumference (cm)",  
        col = c("lightblue", "lightgreen"))
```



Box plot for tree circumference in 2020 (end of the study)

```
boxplot(growth_data$Circumf_2020_cm ~ growth_data$Site,  
        main = "Tree Circumference in 2020 by Site",  
        xlab = "Site", ylab = "Circumference (cm)",  
        col = c("lightblue", "lightgreen"))
```

Tree Circumference in 2020 by Site



Shown above are two separate box plots. The first box will show how the tree circumference is distributed across both sites in 2005, and the second box will show the distribution in 2020. This gives us an understanding of growth over time and how it differs between the two sites. The plots above were created with the `boxplot()` command.

Q9

Calculating the growth between 2010 and 2020 for both sites

```
growth_data$Growth_10yr <- growth_data$Circumf_2020_cm - growth_data$Circumf_2010_cm
```

Subsetting the data for northeast and southwest sites

```
growth_northeast_10yr <- subset(growth_data, Site == "northeast", select = c("Growth_10yr"))  
growth_southwest_10yr <- subset(growth_data, Site == "southwest", select = c("Growth_10yr"))
```

Calculating the mean growth over the last 10 years for each site

```
mean_growth_northeast_10yr <- mean(growth_northeast_10yr$Growth_10yr)  
mean_growth_southwest_10yr <- mean(growth_southwest_10yr$Growth_10yr)
```

Outputting the mean growth values

```
mean_growth_northeast_10yr
```



```
## [1] 42.94
```

```
mean_growth_southwest_10yr
```

```
## [1] 35.49
```

This will give two numeric values:

Mean growth at the northeast site over the last 10 years is 42.94 cm Mean growth at the southwest site over the last 10 years is 35.49 cm

These values tell us how much the tree circumference increased on average at each site between 2010 and 2020.

Q10

Performing t-test to compare the 10-year growth between the two sites

```
t_test_result <- t.test(growth_data$Growth_10yr ~ growth_data$Site)
```

Outputting the p-value from the t-test

```
t_test_result$p.value
```

```
## [1] 0.06229256
```

The p-value will be a number between 0 and 1:

If the p-value < 0.05, we conclude that the 10-year growth is significantly different between the two sites. If the p-value >= 0.05, we conclude that there is no significant difference in growth between the two sites. As the p-value had a result of 0.062, we can conclude there is no significant difference in growth between the 2 sites.

Part 2

Q1

Downloading Coding DNA of *Salmonella*

```
salmonella_URL <- "https://ftp.ensemblgenomes.ebi.ac.uk/pub/bacteria/release-59/fasta/bacteria_50_collection/salmonella/fasta/Salmonella_enterica_subsp._enterica_serovar_Weltevreden.fasta.gz"
download.file(salmonella_URL, destfile = "Salmonella.fa.gz")
gunzip("Salmonella.fa.gz")
```

```
salmonella <- seqinr::read.fasta("Salmonella.fa")
```

To download the *Salmonella enterica* subsp. *enterica* serovar Weltevreden (GCA_005518735), (*Salmonella*) file, the download.file command was used, along with the gunzip command to remove the zip format from the FASTA file.

Downloading Coding DNA of *E.coli*

```
ecoli_URL <- "http://ftp.ensemblgenomes.org/pub/bacteria/release-53/fasta/bacteria_0_collection/escherichia_coli/fasta/E_coli.fasta.gz"
download.file(ecoli_URL, destfile = "ecoli.fa.gz")
gunzip("ecoli.fa.gz")
```

```
e_coli <- seqinr::read.fasta("ecoli.fa")
```

To download the *Escherichia coli* str. K-12 substr. MG1655 str. K12 (GCA_000005845) (*E.coli*) file, the download.file command was used, along with the gunzip command to remove the zip format from the FASTA file.

Table displaying Coding DNA sequence length of *Salmonella* and *E.coli*

```
Salmonella_E.coli <- as.table(rbind(c(length(salmonella),length(e_coli))))
colnames(Salmonella_E.coli) <- c("Salmonella","E-col")
rownames(Salmonella_E.coli) <- "Coding DNA sequences"
Salmonella_E.coli
```

```
##              Salmonella E-col
## Coding DNA sequences    4585    4239
```

The table shown above displays the amount of Coding DNA sequences, we can see a difference in length between *Salmonella* and *E.coli*, with *Salmonella* having 4585 coding DNA sequences and *E.coli* having 4239 coding DNA sequences.

Q2

Coding DNA in *Salmonella* and *E.coli*

```
salmonella_length <-as.numeric(summary(salmonella)[,1])
sum(salmonella_length)
```

```
## [1] 4294851
```

```
e_coli_length <- as.numeric(summary(e_coli)[,1])
sum(e_coli_length)
```

```
## [1] 3978528
```

Table presenting total coding DNA in *Salmonella* and *E.coli*

```
CodingDNA <- as.table(rbind(c(sum(salmonella_length),sum(e_coli_length))))
colnames(CodingDNA) <- c("Salmonella","E-col")
rownames(CodingDNA) <- "Total Coding DNA"
CodingDNA
```

```
##              Salmonella E-col
## Total Coding DNA    4294851 3978528
```

The table shown above displays the amount of Coding DNA in total in both *Salmonella* and *E.coli*, we can see a difference in total amount of DNA between *Salmonella* and *E.coli*, with *Salmonella* having 4294851 coding DNA base pairs and *E.coli* having 3978528 coding DNA base pairs.

Q3

Length of all coding sequences in *Salmonella*

```
salmonella_csl <- sapply(salmonella, length)
head(salmonella_csl,25)
```

```
## ENSB:rzMRPrOXj2f6n3A ENSB:x3MzlaR1iM60p6v ENSB:btfdQZt4_vWjqOV
##              417              402              2613
```

```
## ENSB:IZ64ldiL3-oOAYf ENSB:9CMZHlFso1P0uRQ ENSB:WDGo7WHsQXy-ZVK
##          1314          165          918
## ENSB:nxZOxcqpENLAhQ- ENSB:PdQ_njih_D4z76t ENSB:doMeXiedV9pXccR
##          879          2799          1080
## ENSB:kQ3Q9ymg3iP1FNt ENSB:HIcMhKuQYYbSiCI ENSB:kMzQPuEK1M6h21v
##          1335          1545          1563
## ENSB:PybEbgTbnAhpXJ4 ENSB:g7VWNp8A7d0sFEs ENSB:CqxksGF2_zG4Fwf
##          852          201          627
## ENSB:qcIxBcPJXPY3xCH ENSB:2fftfgVWM0g5gVb ENSB:asHCFULnp2Cpguy
##          438          1593          1419
## ENSB:Szs8QJYwEVWvtvU ENSB:Rw0LhdcFy6IGZRX ENSB:Pa_nz_ONaNE4x4I
##          996          189          1428
## ENSB:OK5ccU11Vz1MFBH ENSB:_-Kn39gDsVlrpn5 ENSB:iK0yimEq0tEz0Gl
##          1017          1326          324
## ENSB:821CJ4HgDYyKI6H
##          159
```

`sapply()` works as a loop function when calculating the length of each sequence in the *Salmonella* data. the `head()` function is utilised to show the first 25 sequences lengths in an effort to minimise the size of the `rmd` file. `print` can be used in it's place for a more complete view.

Length of all coding sequences in *E.coli*

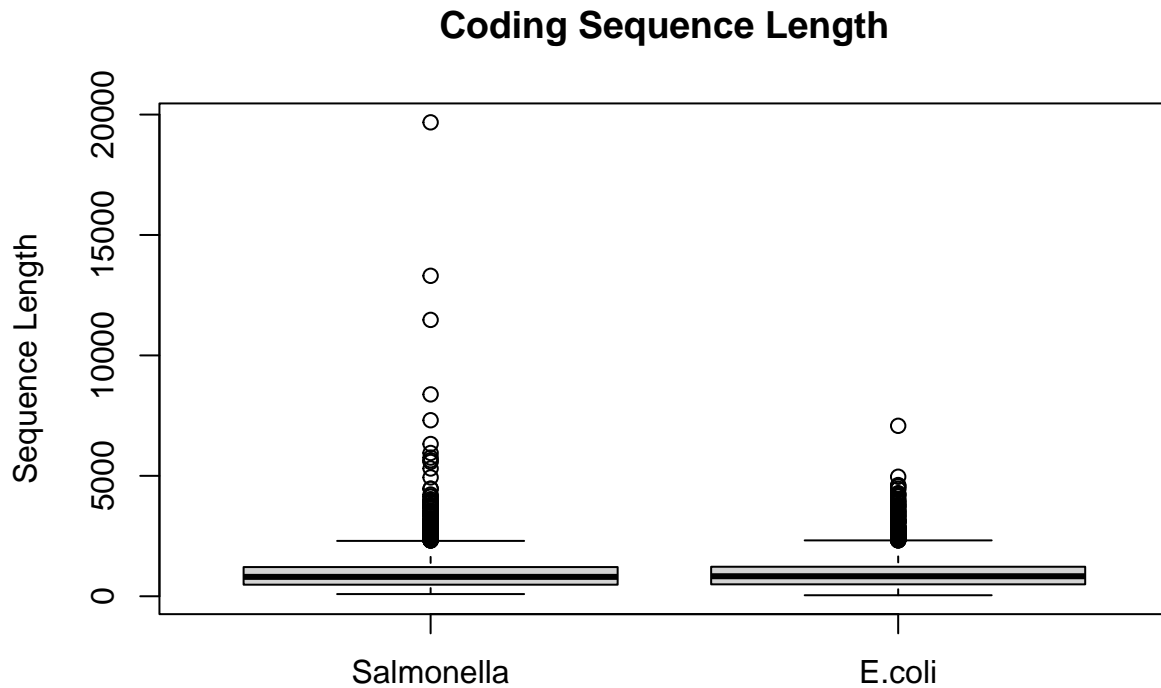
```
e_coli_csl <- sapply(e_coli, length)
head(e_coli_csl,25)
```

```
## AAC73112 AAC73113 AAC73114 AAC73115 AAC73116 AAC73117 AAC73118 AAC73119
##          66          2463          933          1287          297          777          1431          954
## AAC73120 AAC73121 AAC73122 AYC08161 AAC73124 AAC73125 AAC73126 AAC73127
##          588          567          714          486          405          1917          1131          1113
## AAC73129 AAT48122 AAC73130 AAC73131 AAC73132 AAC73133 AAC73134 AAC73135
##          210          153          1167          906          504          276          264          219
## AAC73136
##          942
```

`sapply()` works as a loop function when calculating the length of each sequence in the *E.coli* data. the `head()` function is utilised to show the first 25 sequences lengths in an effort to minimise the size of the `rmd` file. `print` can be used in it's place for a more complete view.

Boxplot of coding sequence length

```
boxplot(salmonella_csl, e_coli_csl,
        ylab="Sequence Length",
        main="Coding Sequence Length",
        names = c("Salmonella", "E.coli"))
```



The `boxplot()` function creates a boxplot of both the *Salmonella* and the *E.coli* data. The heading of the plot is changed with the 'main' argument, the y axis label with 'ylab' and the names of each individual plot with the 'names' argument.

Mean and Median coding sequence length of *Salmonella*

```
mean(salmonella_cs1)
```

```
## [1] 936.7178
```

The `mean()` command calculates the average value, in this case the length, of every sequence. The mean for the length of *Salmonella* sequences is 936.7178.

```
median(salmonella_cs1)
```

```
## [1] 804
```

The `median()` command calculates the middle value of all the sequences. The median for the length of *Salmonella* sequences is 804.

Mean and Median coding sequence length of *E.coli*

```
mean(e_coli_cs1)
```

```
## [1] 938.5534
```

The `mean()` command calculates the average value, in this case the length, of every sequence. The mean for the length of *E.coli* sequences is 938.5534.

```
median(e_coli_csl)
```

```
## [1] 831
```

The median() command calculates the middle value of all the sequences. The median for the length of *E.coli* sequences is 831.

Both organisms are quite similar with sequence lengths. *Salmonella* has a slightly lower mean value (4bp difference) and an even lower median value (27bp difference). This indicates that while on average, the two organisms have similar length sequences, however *Salmonella* has a higher frequency of longer sequences. These differences can indicate different genetic expression and functionality.

Q4

Frequency of DNA bases

```
salmonella_DNA <- unlist(salmonella)
salmonella_DNA_comp <- count(salmonella_DNA,1)
salmonella_DNA_prop <- salmonella_DNA_comp/sum(salmonella_DNA_comp)
salmonella_DNA_prop
```

```
##
##          a          c          g          t
## 0.2339450 0.2513941 0.2808617 0.2337993
```

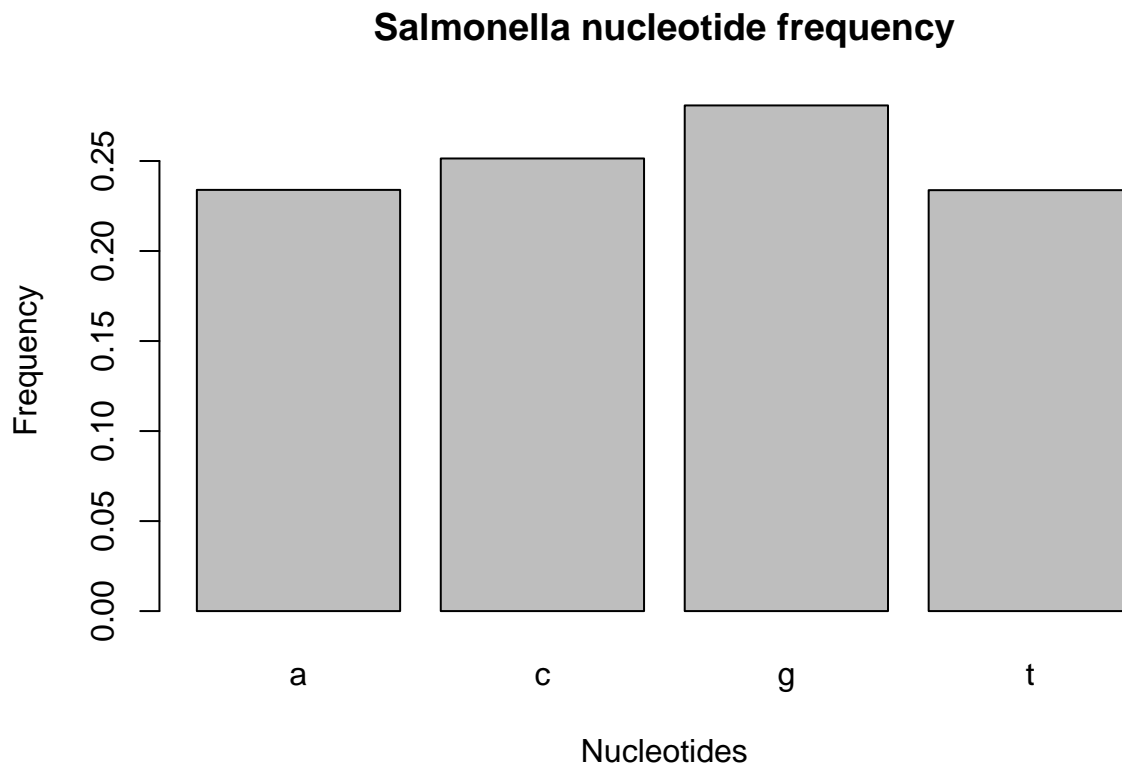
```
e_coli_DNA <- unlist(e_coli)
e_coli_DNA_comp <- count(e_coli_DNA,1)
e_coli_DNA_prop <- e_coli_DNA_comp/sum(e_coli_DNA_comp)
e_coli_DNA_prop
```

```
##
##          a          c          g          t
## 0.2402316 0.2457175 0.2735939 0.2404570
```

The unlist() command converts the list data into a vector making it easier for analysis. count() is used to calculate the amount of each nucleotide in each sequence, <- stores the results in the _comp values. Dividing the _comp data by the sum of itself determines the frequency as a percentage in decimal form.

Bar plot for *Salmonella* nucleotide frequency

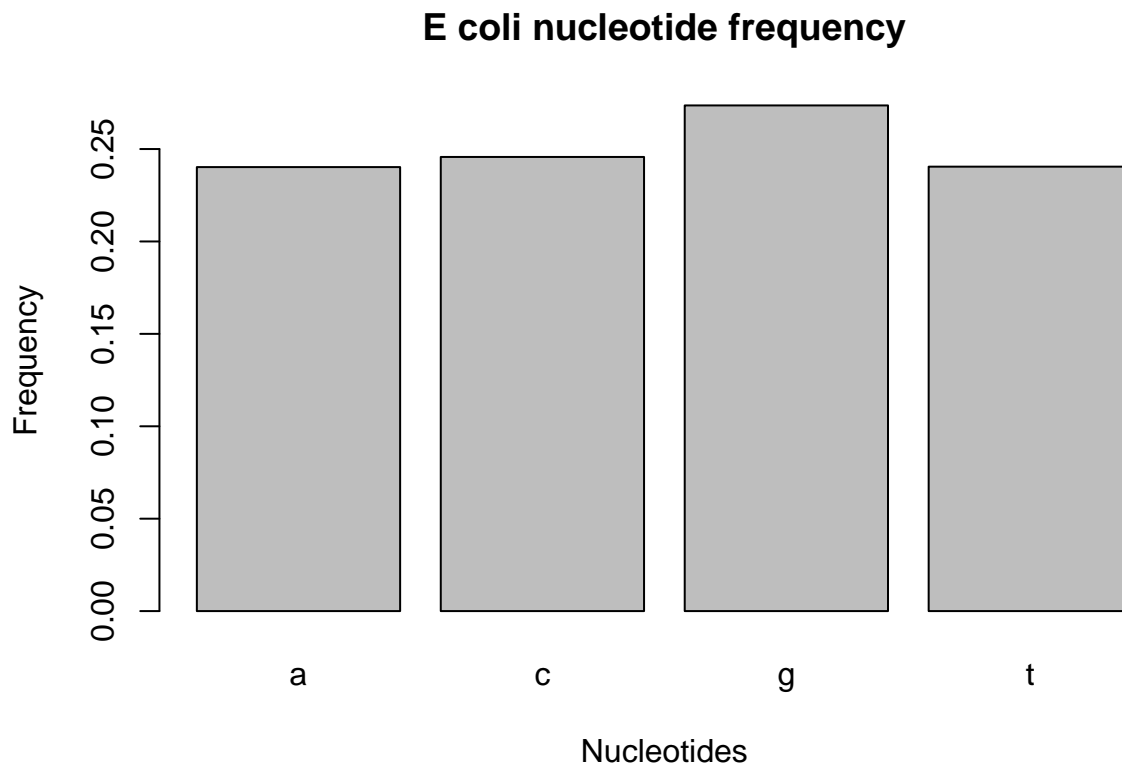
```
barplot(salmonella_DNA_prop,xlab="Nucleotides",ylab="Frequency", main="Salmonella nucleotide frequency")
```



The frequency of nucleotides is displayed as a barplot with the `barplot()` command. The plot heading is determined with “main”, the x and y axis labels are determined using “xlab” and “ylab” respectively.

Bar plot for *E.coli* nucleotide frequency

```
barplot(e_coli_DNA_prop,xlab="Nucleotides",ylab="Frequency", main="E coli nucleotide frequency")
```



The frequency of nucleotides is displayed as a barplot with the `barplot()` command. The plot heading is determined with “main”, the x and y axis labels are determined using “xlab” and “ylab” respectively.

Salmonella Protein sequence frequency

```
salmonella_prot <- lapply(salmonella, translate)

salmonella_aa <- unique(salmonella_prot[[2]])
salmonella_aa <- salmonella_aa[salmonella_aa != "*"]

salmonella_aa_len <- length(salmonella_aa)

salmonella_aa_count <- lapply(salmonella_prot, function(sequence) {
  count(sequence, wordsize = 1, alphabet = salmonella_aa)
})

salmonella_aa_table <- do.call(rbind, salmonella_aa_count)

salmonella_aa_totals <- colSums(salmonella_aa_table)

salmonella_aa_prop <- salmonella_aa_totals/sum(salmonella_aa_totals)

salmonella_aa_prop
```

```
##           A           D           E           F           G           I           K
## 0.10083505 0.05415912 0.05817653 0.04002090 0.07623562 0.06161545 0.04450283
##           L           M           N           P           Q           R           S
```

```
## 0.11030698 0.02847749 0.03952880 0.04634278 0.04553858 0.05850387 0.06015874
##          T          V          W          Y
## 0.05697239 0.07288451 0.01584825 0.02989211
```

To translate all sequences in the data to proteins sequences, the `lapply()` command is applied. To define the amino acid alphabet, `unique()` extracts unique amino acids while `salmonella_aa[salmonella_aa != "*"]` removes any unknown amino acids. This is then used as a reference when using `lapply(x,count())` to calculate the amount of each amino acid in the sequences. `do.call(rbind,x)` is usually for combining more complex amounts of data into 1 data frame, however it is still effective here. `colSums` calculates the total of each column, in this case, it add the amount of each amino acid across the sequences. The proportion of each total is calculated by dividing each total by the overall total. The results are then displayed.

E.coli Protein sequence frequency

```
e_coli_prot <- lapply(e_coli, translate)

e_coli_aa <- unique(e_coli_prot[[2]])
e_coli_aa <- e_coli_aa[e_coli_aa != "*"]

e_coli_aa_len <- length(e_coli_aa)

e_coli_aa_count <- lapply(e_coli_prot, function(sequence) {
  count(sequence, wordsize = 1, alphabet = e_coli_aa)
})

e_coli_aa_table <- do.call(rbind, e_coli_aa_count)

e_coli_aa_totals <- colSums(e_coli_aa_table)

e_coli_aa_prop <- e_coli_aa_totals/sum(e_coli_aa_totals)

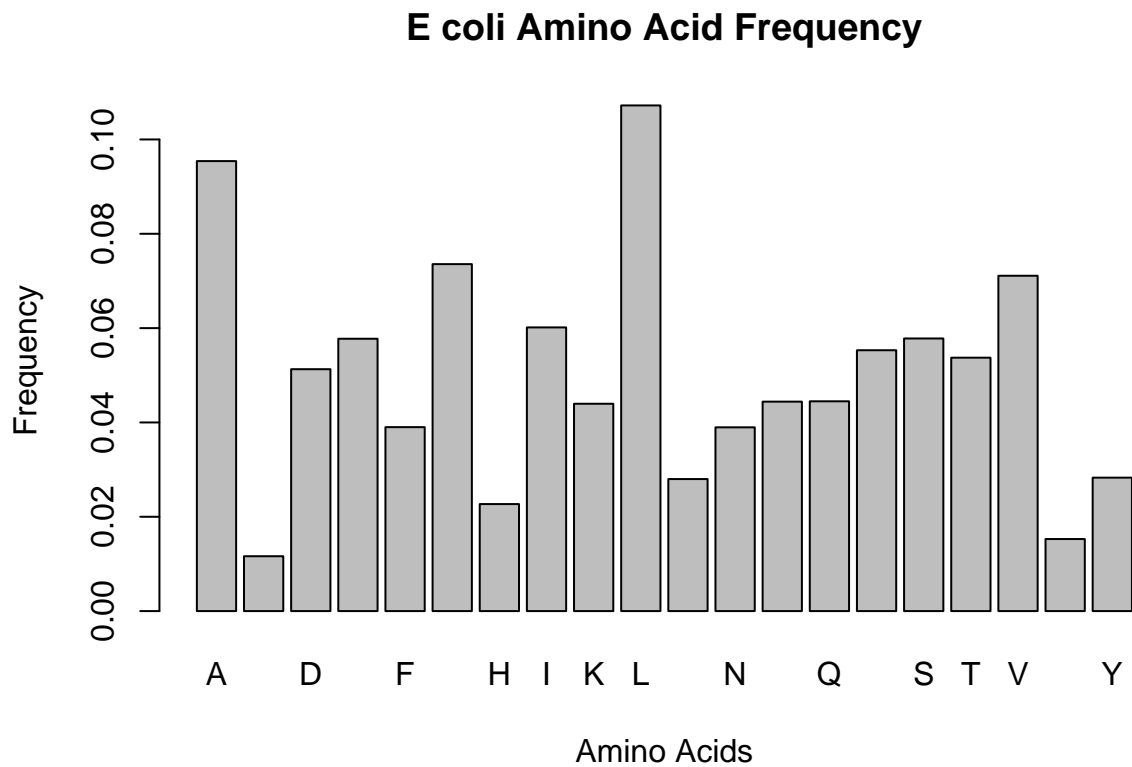
e_coli_aa_prop
```

```
##          A          C          D          E          F          G          H
## 0.09541075 0.01163142 0.05128535 0.05774708 0.03900413 0.07356326 0.02269019
##          I          K          L          M          N          P          Q
## 0.06014734 0.04396049 0.10721464 0.02799453 0.03896025 0.04440454 0.04447943
##          R          S          T          V          W          Y
## 0.05530596 0.05780306 0.05372798 0.07109945 0.01527758 0.02829257
```

To translate all sequences in the data to proteins sequences, the `lapply()` command is applied. To define the amino acid alphabet, `unique()` extracts unique amino acids while `e_coli_aa[e_coli_aa != "*"]` removes any unknown amino acids. This is then used as a reference when using `lapply(x,count())` to calculate the amount of each amino acid in the sequences. `do.call(rbind,x)` is usually for combining more complex amounts of data into 1 data frame, however it is still effective here. `colSums` calculates the total of each column, in this case, it add the amount of each amino acid across the sequences. The proportion of each total is calculated by dividing each total by the overall total. The results are then displayed.

Bar plot for *E.coli* amino acid frequency

```
barplot(e_coli_aa_prop,xlab="Amino Acids",ylab="Frequency", main="E coli Amino Acid Frequency")
```

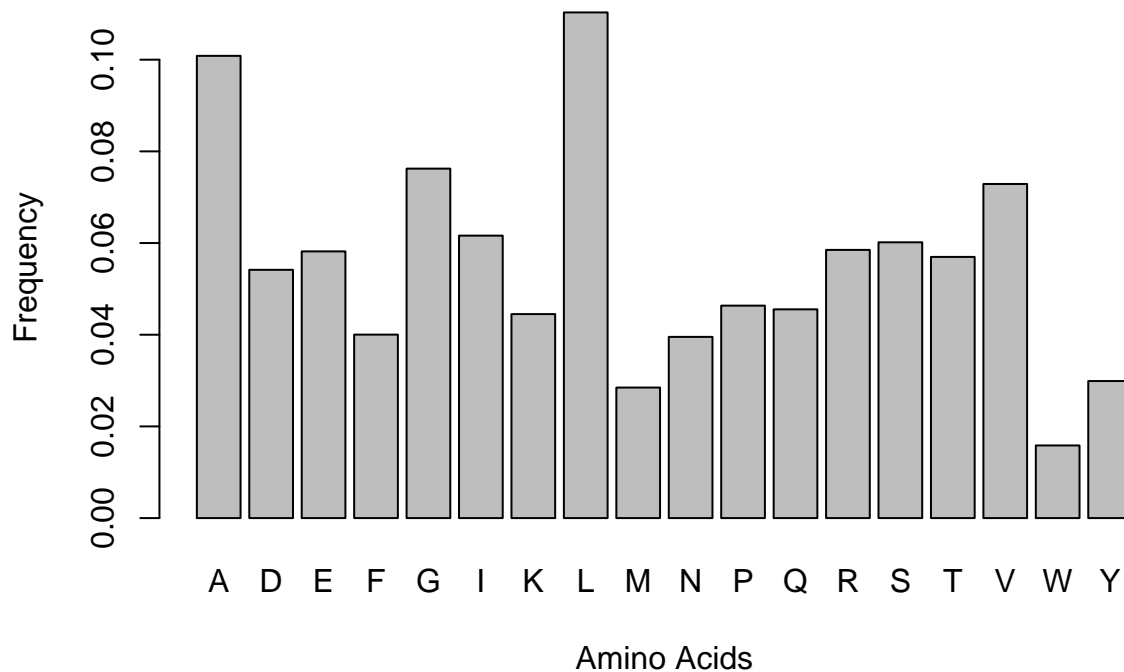



The frequency of each amino acid is displayed as a barplot with the `barplot()` command. The plot heading is determined with “main”, the x and y axis labels are determined using “xlab” and “ylab” respectively.

Bar plot for *Salmonella* amino acid frequency

```
barplot(salmonella_aa_prop,xlab="Amino Acids",ylab="Frequency", main="Salmonella Amino Acid Frequency")
```

Salmonella Amino Acid Frequency



The frequency of each amino acid is displayed as a barplot with the `barplot()` command. The plot heading is determined with “main”, the x and y axis labels are determined using “xlab” and “ylab” respectively.

Nucleotide frequency and their distributions, as shown in the barplots, are quite similar between the 2 organisms indicating functional similarities. As the amino acid barplots indicate, most amino acid frequencies across the two organisms are similar, however, *E.coli* has data for both C and H where *Salmonella* does not.

Q5

```
# Define function to calculate codon usage
codon_usage <- function(sequence_list) {
  codons <- c()
  for (seq in sequence_list) {
    seq_codons <- s2c(seq)
    codons <- c(codons, uco(seq_codons, frame = 0)) # Counts codon occurrences
  }
  codon_table <- table(codons) # Creates a frequency table of codons
  return(codon_table)
}

# Codon usage for Salmonella
salmonella_codon_usage <- codon_usage(salmonella)
```

```
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
## Warning in s2c(seq): Wrong argument type in s2c(), NA returned
e_coli_codon_usage <- prop.table(e_coli_codon_usage)

# Combine and view results
codon_usage_table <- data.frame(
  Codon = names(salmonella_codon_usage),
  Salmonella = as.numeric(salmonella_codon_usage),
  Ecoli = as.numeric(e_coli_codon_usage)
)
head(codon_usage_table)

##      Codon Salmonella Ecoli
## 1         0          1     1

### Quantifying Codon Usage Bias (RSCU)

# Function to calculate RSCU (Relative Synonymous Codon Usage)
calculate_rscu <- function(codon_usage, codon_table) {
  rscu_values <- codon_usage / mean(codon_usage[codon_table]) # Normalizing by synonymous codons
  return(rscu_values)
}

# Calculate RSCU for both organisms
salmonella_rscu <- calculate_rscu(salmonella_codon_usage, codon_usage_table$Codon)
e_coli_rscu <- calculate_rscu(e_coli_codon_usage, codon_usage_table$Codon)

# Combine RSCU results in a table
rscu_table <- data.frame(
  Codon = codon_usage_table$Codon,
  Salmonella_RSCU = salmonella_rscu,
  Ecoli_RSCU = e_coli_rscu
)
```

Combine and view results

```
codon_usage_table <- data.frame(
  Codon = names(salmonella_codon_usage),
  Salmonella = as.numeric(salmonella_codon_usage),
  Ecoli = as.numeric(e_coli_codon_usage)
)
head(codon_usage_table)
```

```
##      Codon Salmonella Ecoli
## 1      0      1      1
```

Quantifying Codon Usage Bias (RSCU)

Function to calculate RSCU (Relative Synonymous Codon Usage)

```
calculate_rscu <- function(codon_usage, codon_table) {
  rscu_values <- codon_usage / mean(codon_usage[codon_table]) # Normalizing by synonymous codons
  return(rscu_values)
}
```

Calculate RSCU for both organisms

```
salmonella_rscu <- calculate_rscu(salmonella_codon_usage, codon_usage_table$Codon)
e coli_rscu <- calculate_rscu(e coli codon usage, codon usage table$Codon)
```

```
# Combine RSCU results in a table
```

```
rscu_table <- data.frame(
  Codon = codon_usage_table$Codon,
  Salmonella_RSCU = salmonella_rscu,
  Ecoli_RSCU = e_coli_rscu
)
```

```

head(rscu_table)

##      Codon Salmonella_RSCU.codons Salmonella_RSCU.Freq Ecoli_RSCU.codons
## 1      0                      0                      1                      0
##      Ecoli_RSCU.Freq
## 1                      1

# Convert columns to numeric
rscu_table <- data.frame(lapply(rscu_table, function(x) as.numeric(as.character(x))), stringsAsFactors = FALSE)

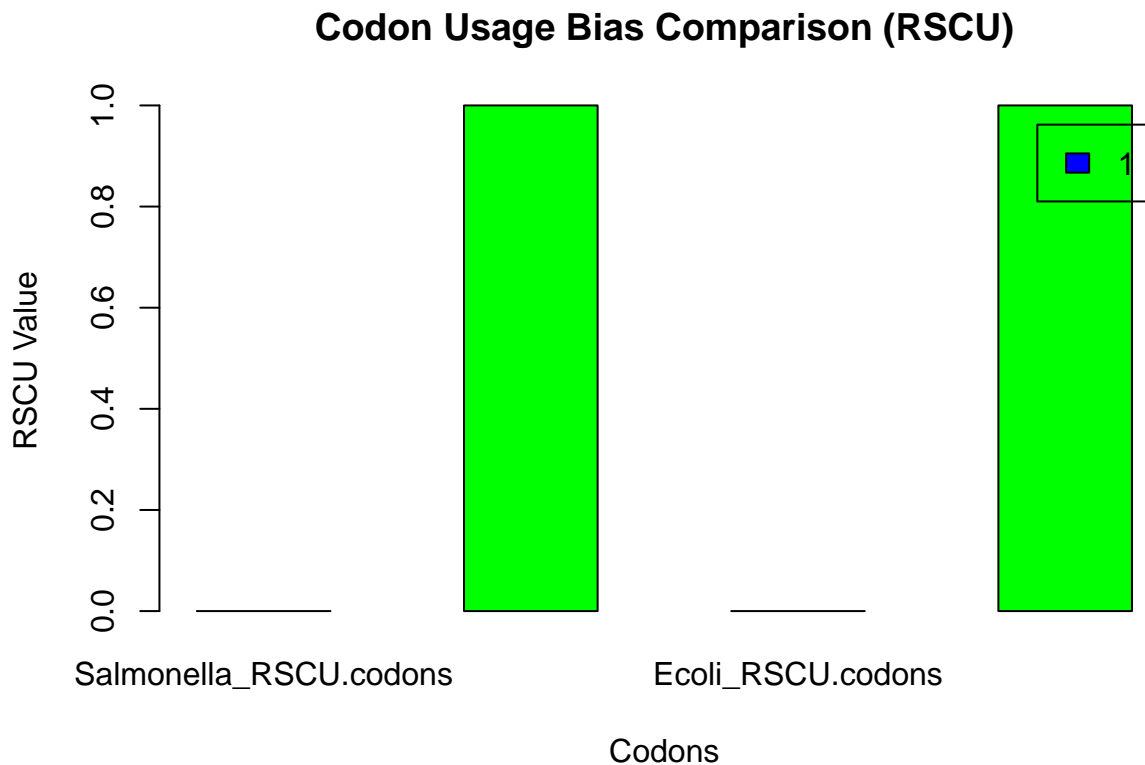
# Check for complete cases and remove non-numeric entries
rscu_table <- rscu_table[complete.cases(rscu_table), ]

# Check structure
str(rscu_table)

## 'data.frame':    1 obs. of  5 variables:
##  $ Codon          : num 0
##  $ Salmonella_RSCU.codons: num 0
##  $ Salmonella_RSCU.Freq : num 1
##  $ Ecoli_RSCU.codons   : num 0
##  $ Ecoli_RSCU.Freq     : num 1

# Create the bar plot
barplot(as.matrix(rscu_table[, -1]),
        beside = TRUE,
        col = c("blue", "green"),
        legend = rownames(rscu_table),
        xlab = "Codons",
        ylab = "RSCU Value",
        main = "Codon Usage Bias Comparison (RSCU)")

```



Q5 Attempt 2

Codon usage table for Salmonella

```
codon_salmonella <- readSet(file="https://ftp.ensemblgenomes.ebi.ac.uk/pub/bacteria/release-59/fasta/ba
```

```
## Warning: `progress_estimated()` was deprecated in dplyr 1.0.0.
## i The deprecated feature was likely used in the coRdon package.
## Please report the issue at <https://github.com/BioinfoHR/coRdon/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
codon_tab_salmonella <- codonTable(codon_salmonella)
head(codon_tab_salmonella)
```

```
## codonTable instance with codon counts from 6 sequences.
##
## 1 ENSB:rzMRPrOXj2f6n3A cds primary_assembly:ASM551873v1:contig00038:4502:4918:-1 gene:ENSB:rzMRPrOXj
## 2 ENSB:x3MzlaR1iM6Op6v cds primary_assembly:ASM551873v1:contig00003:321788:32
## 3 ENSB:btfdQZt4_vWjqOV cds primary_assembly:ASM551873v1:contig00009:90040:92652:1 ge
## 4 ENSB:IZ64ldiL3-o0AYf cds primary_assembly:ASM551873v1:contig00025:33632:34945:1 gene:ENSB:IZ64ld
## 5 ENSB:9CMZH1Fso1P0uR
## 6 ENSB:WDGo7WHsQXy-ZVK cds primary_assembly:ASM551873v1:contig00004:270379:271296:1 gene:ENSB:
## length AAA AAC AAG AAT ACA ACC ACG ACT AGA AGC AGG AGT ATA ATC ATG ATT CAA
## 1 139 4 7 0 8 0 8 4 1 2 1 0 0 0 4 4 4 4
## 2 134 9 2 2 5 0 2 2 2 1 0 1 0 0 2 7 4 2
```



```
## 3      871  17  22   5  25   6  24  23   3   1  37   3  11   6  11  10  20  17
## 4      438  14   5   1   3   0  14   7   2   0   4   0   4   5  22  23  29   3
## 5        55   3   0   0   0   1   0   2   0   0   0   0   0   2   0   4   1   0
## 6      306  12   4   4   7   1   5   5   0   2   9   0   3   2   4  16  15   5
##      CAC CAG CAT CCA CCC CCG CCT CGA CGC CGG CGT CTA CTC CTG CTT GAA GAC GAG GAT
## 1      0   6   1   2   3   0   1   0   1   0   3   0   3   2   1   2   3   0   6
## 2      0   6   0   0   0   5   1   0   6   0   4   0   1   5   2  12   2   4   4
## 3      1  19   8   1   6  20   9   4  25   9  16   3   3  34  10  14  21  11  33
## 4      4   6   2   1   6  13   0   1   5   1   7   1   6  25   3   8   5   3   8
## 5      1   0   0   1   0   1   2   0   1   0   0   0   0   3   1   0   0   3   1
## 6      3   7   5   0   0  11   2   2   8   6   5   0   5  17   3  10   4  11   8
##      GCA GCC GCG GCT GGA GGC GGG GGT GTA GTC GTG GTT TAA TAC TAG TAT TCA TCC TCG
## 1      1   1   2   4   4   2   2   4   1   1   3   1   1   0   0   2   4   1   3
## 2      3   4   3   3   1   1   1   2   0   1   0   2   1   0   0   1   1   1   0
## 3      8  23  39   4  17  47  19  11   9  15  27   5   1  26   0  25   7   7  14
## 4      4  12  17   3   6  28   9   5   4  16  15  10   1   5   0   5   1   4   3
## 5      1   1   2   3   0   0   1   0   1   1   3   1   1   0   0   0   0   1   1
## 6      5   5  14   2   3   9   3   3   2   3   6   5   1   2   0   9   0   2   4
##      TCT TGA TGC TGG TGT TTA TTC TTG TTT
## 1      0   0   0   1   0   3   2   4   7
## 2      2   0   0   1   0   4   4   2   3
## 3      6   0   0  19   5   5   9  13  22
## 4      3   0   3   4   0   8   6   8  17
## 5      0   0   1   1   3   1   0   1   4
## 6      0   0   3   2   0   5   1   6  10
```

The *Salmonella* Fasta file was read by the command `readSet ()` to create a codon table, which the first six values of each codon can be seen above.

Codon usage table for E-coli

```
codon_e_coli <- readSet(file="http://ftp.ensemblgenomes.org/pub/bacteria/release-53/fasta/bacteria_0_codons.fasta")
codon_tab_e_coli <- codonTable(codon_e_coli)
head(codon_tab_e_coli)

## codonTable instance with codon counts from 6 sequences.
##
## 1          AAC73112 cds chromosome:ASM584v2:Chromosome:190:255:1 gene:b0001 gene_biotype:protein_coding
## 2 AAC73113 cds chromosome:ASM584v2:Chromosome:337:2799:1 gene:b0002 gene_biotype:protein_coding
## 3          AAC73114 cds chromosome:ASM584v2:Chromosome:2801:3733:1 gene:b0003 gene_biotype:protein_coding
## 4          AAC73115 cds chromosome:ASM584v2:Chromosome:3734:5020:1 gene:b0004 gene_biotype:protein_coding
## 5          AAC73116 cds chromosome:ASM584v2:Chromosome:5234:5530:1 gene:b0005 gene_biotype:protein_coding
## 6          AAC73117 cds chromosome:ASM584v2:Chromosome:5683:6459:-1 gene:b0006 gene_biotype:protein_coding
##      length AAA AAC AAG AAT ACA ACC ACG ACT AGA AGC AGG AGT ATA ATC ATG ATT CAA
## 1      22    1   1   0   0   1   7   0   0   0   1   0   0   0   1   1   3   0
## 2     821   22  16  12  22   2  19   8   5   0  12   2   3   1  15  23  30  11
## 3     311    3   6   7   6   2   3   3   1   0   4   1   3   0   9  10   7   3
## 4     429   17  11   5   8   2  11   6   4   0   6   0   3   0   9   8   8   5
## 5      99    5   0   2   2   0   0   1   0   0   0   0   0   1   1   2   1   2
## 6     259   11   4   7   9   0   6   6   1   0   7   0   1   0   9   6   8   5
##      CAC CAG CAT CCA CCC CCG CCT CGA CGC CGG CGT CTA CTC CTG CTT GAA GAC GAG GAT
## 1      0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
## 2      6  19   8   2   6  18   3   3  19   4  18   2  13  43   8  40  14  13  30
## 3      3  11   3   4   1   9   2   2   4   5   5   0   5  15   2  15   5   6   8
## 4      5  16   5   9   3  11   0   1   7   0   9   1   6  29   1  19   4  10  20
```

```
## 5 7 2 6 2 1 5 1 1 4 0 2 1 1 2 1 2 1 0 4
## 6 0 9 3 2 0 7 3 2 7 2 4 0 4 15 4 7 4 9 13
## GCA GCC GCG GCT GGA GGC GGG GGT GTA GTC GTG GTT TAA TAC TAG TAT TCA TCC TCG
## 1 0 0 1 0 0 1 0 2 0 0 0 0 0 0 0 0 0 0 0
## 2 14 36 26 15 9 22 10 22 5 18 27 19 0 8 0 12 6 10 9
## 3 6 9 17 5 2 12 5 8 3 5 7 8 1 3 0 5 1 6 3
## 4 5 8 24 10 2 11 4 11 2 5 15 5 1 5 0 4 3 2 5
## 5 3 0 1 3 2 6 0 3 2 2 1 1 1 1 0 2 1 1 0
## 6 3 6 8 2 1 6 2 5 1 1 5 0 1 5 0 5 2 4 1
## TCT TGA TGC TGG TGT TTA TTC TTG TTT
## 1 0 1 0 0 0 0 0 0 0
## 2 11 1 9 4 3 10 19 13 11
## 3 0 0 6 4 3 3 4 6 6
## 4 0 0 1 3 3 5 11 12 13
## 5 1 0 0 6 0 2 0 0 0
## 6 0 0 0 2 0 2 8 6 8
```

The *E.coli* Fasta file was read by the command `readSet ()` to create a codon table, which the first six values of each codon can be seen above.

Codon bias for salmonella

```
salmonella_milc <- MILC(codon_tab_salmonella)
head(salmonella_milc)
```

```
##          self
## [1,] 0.7199963
## [2,] 0.5252359
## [3,] 0.5502128
## [4,] 0.5231461
## [5,] 0.7078510
## [6,] 0.5652358
```

The codon bias of *Salmonella's* first 6 genes is shown above, to find this the `MILC()` command was used, with the `head()` command.

Codon bias for E-coli

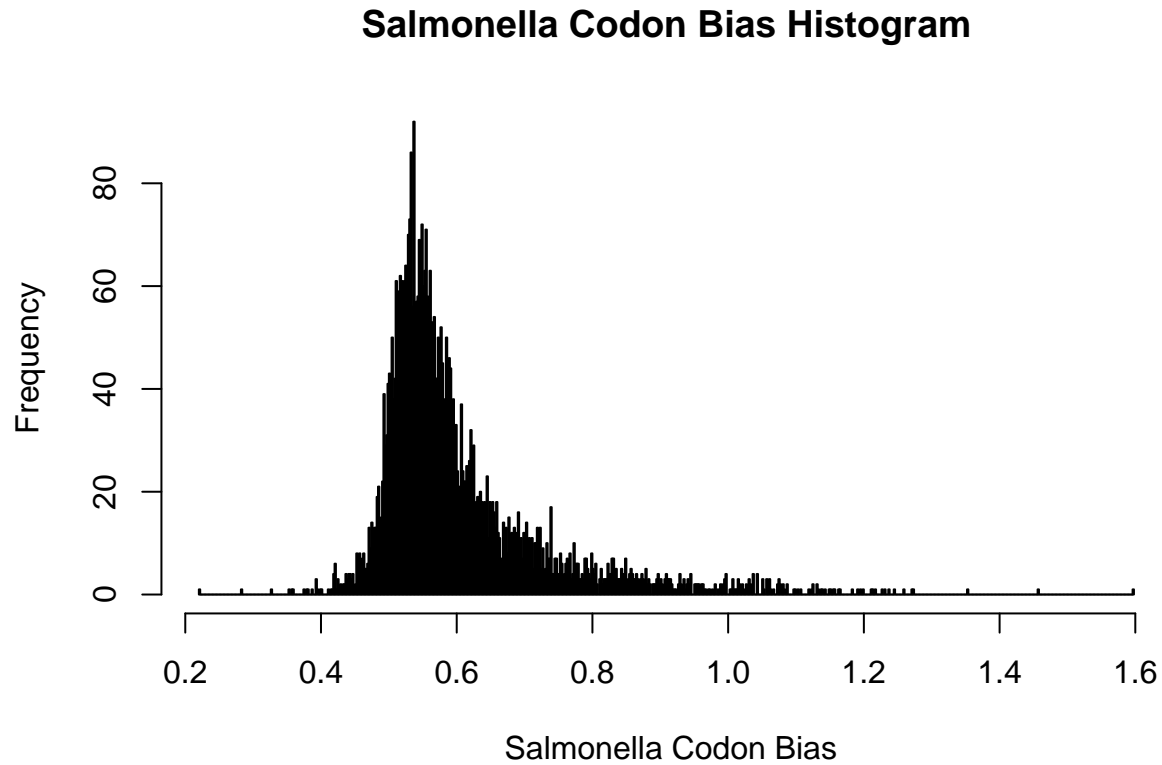
```
e_coli_milc <- MILC(codon_tab_e_coli)
head(e_coli_milc)
```

```
##          self
## [1,] 0.4690096
## [2,] 0.5160116
## [3,] 0.5034795
## [4,] 0.5419666
## [5,] 0.4711169
## [6,] 0.5398071
```

The codon bias of *E.coli's* first 6 genes is shown above, to find this the `MILC()` command was used, with the `head()` command.

Histogram of Codon bias in *Salmonella*

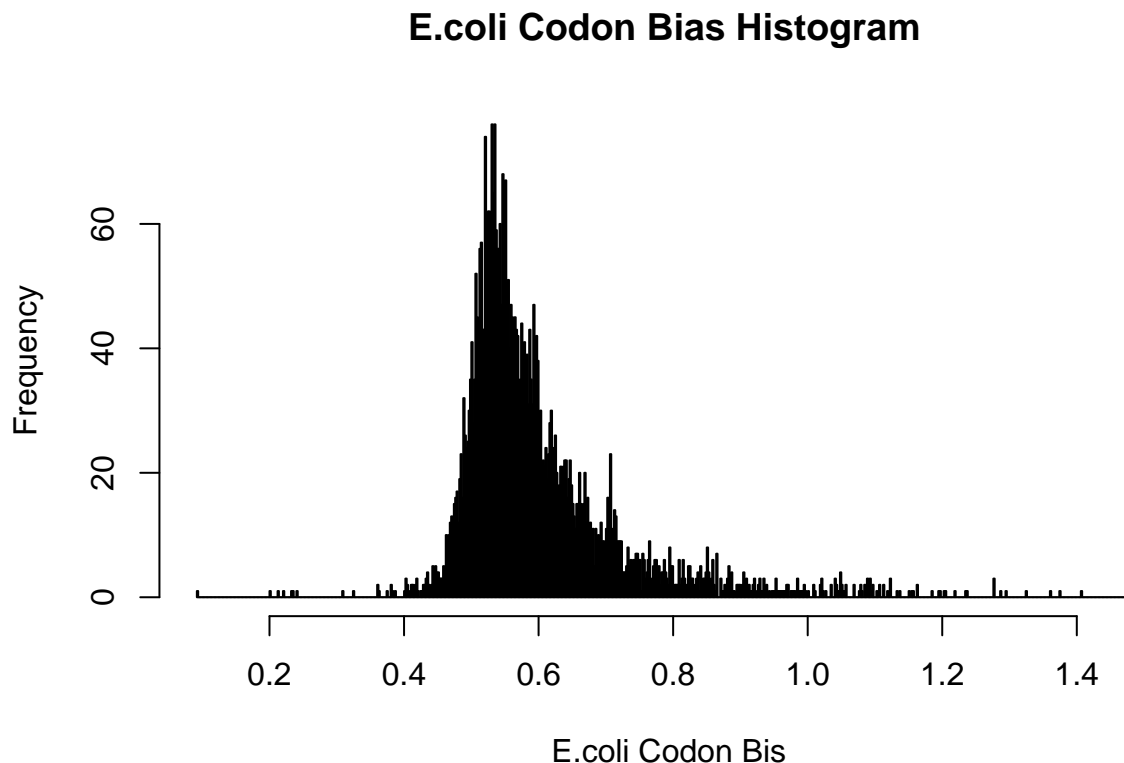
```
hist(salmonella_milc,
     main = "Salmonella Codon Bias Histogram",
     xlab = " Salmonella Codon Bias", ylab = "Frequency",breaks=500)
```



Shown above is a histogram displaying the *Salmonella* Codon Bias, this was visualised with the `hist()` command.

Histogram of Codon bias in *E.coli*

```
hist(e_coli_milc,
     main = "E.coli Codon Bias Histogram",
     xlab = " E.coli Codon Bis", ylab = "Frequency",breaks=500)
```



Shown above is a histogram displaying the *E.coli* Codon Bias, this was visualised with the `hist()` command.

The *Salmonella* histogram shows a higher frequency of codon bias around 0.5 to 0.6, with the spread of bias from approximately 0.2 to 1.6. The histogram of *E.coli* shows a higher frequency of codon bias around 0.5 to 0.6, with the spread of bias being from approximately 0.2 to 1.4. The Codon bias for both *Salmonella* and *E.coli* show a similar histogram shape, indicating a similar codon bias between both organisms.

Q6

Creating amino acid alphabet and unlisting *Salmonella* proteins

```
salmonella_prot <- lapply(salmonella, translate)
salmonella_prots <- unlist(salmonella_prot)

head(salmonella_prots)

## ENSB:rzMRPrOXj2f6n3A1 ENSB:rzMRPrOXj2f6n3A2 ENSB:rzMRPrOXj2f6n3A3
## "M" "R" "V"
## ENSB:rzMRPrOXj2f6n3A4 ENSB:rzMRPrOXj2f6n3A5 ENSB:rzMRPrOXj2f6n3A6
## "K" "H" "A"

aa <- unique(salmonella_prot[[2]])
aa <- aa[aa != "*"]
```

Creating data frame of 3 length K-mers in Salmonella

```
salmonella_counts_3 <- count(salmonella_prots, wordsize=3, alphabet=aa)
head(salmonella_counts_3)
```

```
##
##   AAA  AAD  AAE  AAF  AAG  AAI
## 1575  630  782  534 1185  893

df_salmonella_counts_3 <- as.data.frame(salmonella_counts_3)
head(df_salmonella_counts_3)
```

```
##   Var1 Freq
## 1   AAA 1575
## 2   AAD  630
## 3   AAE  782
## 4   AAF  534
## 5   AAG 1185
## 6   AAI  893
```

A data frame of 3 length K-mers was created for *Salmonella* to be able to sort over represented and underrepresented 3 length k-mers. The data frame was created with the `as.data.frame()` command.

Creating data frame of 3 length K-mers in E.coli

```
e_coli_prot <- lapply(e_coli, translate)
e_coli_prots <- unlist(e_coli_prot)

head(e_coli_prots)

## AAC7311121 AAC7311122 AAC7311123 AAC7311124 AAC7311125 AAC7311126
##          "M"          "K"          "R"          "I"          "S"          "T"

e_coli_counts_3 <- count(e_coli_prots, wordsize=3, alphabet=aa)
head(e_coli_counts_3)
```

```
##
##   AAA  AAD  AAE  AAF  AAG  AAI
## 1338  568  749  481 1075  863

df_e_coli_counts_3 <- as.data.frame(e_coli_counts_3)
head(df_e_coli_counts_3)
```

```
##   Var1 Freq
## 1   AAA 1338
## 2   AAD  568
## 3   AAE  749
## 4   AAF  481
## 5   AAG 1075
## 6   AAI  863
```

A data frame of 3 length K-mers was created for *E.coli* to be able to sort over represented and underrepresented 3 length k-mers. The data frame was created with the `as.data.frame()` command.

Top 10 most overrepresented 3 length K-mers in Salmonella

```
over_salmonella_counts_3 <- arrange(df_salmonella_counts_3, -Freq)
head(over_salmonella_counts_3, 10)
```

```
##   Var1 Freq
## 1   LLA 1973
```

```
## 2   ALL 1950
## 3   ALA 1949
## 4   LAA 1752
## 5   AAL 1749
## 6   LLL 1723
## 7   LAL 1700
## 8   AAA 1575
## 9   LAG 1273
## 10  TLL 1205
```

Top 10 most overrepresented 3 length K-mers in E.coli

```
over_e_coli_counts_3 <-arrange(df_e_coli_counts_3,-Freq)
head(over_e_coli_counts_3,10)
```

```
##      Var1 Freq
## 1     LLA 1817
## 2     ALL 1744
## 3     ALA 1740
## 4     LAA 1601
## 5     AAL 1594
## 6     LLL 1594
## 7     LAL 1536
## 8     AAA 1338
## 9     LAG 1178
## 10    TLL 1102
```

The 3 length K-mers overrepresented in *Salmonella* and *E.coli*, are all the same, and in the same order of overrepresentation, however they do vary on the amount of representation In the gene. The identification of the overrepresented 3 length K-mers were found by using the `arrange()` command, with a - negative frequency, to order the data frame in decreasing order.

Top 10 most underrepresented 3 length K-mers in Salmonella

```
under_salmonella_counts_3 <-arrange(df_salmonella_counts_3,Freq)
head(under_salmonella_counts_3,10)
```

```
##      Var1 Freq
## 1     WWW    5
## 2     WMW    6
## 3     YMW    6
## 4     NWW    7
## 5     MWP    8
## 6     WWT    8
## 7     MWN    9
## 8     MWW    9
## 9     WYW    9
## 10    KWW   10
```

Top 10 most underrepresented 3 length K-mers in E.coli

```
under_e_coli_counts_3 <-arrange(df_e_coli_counts_3,Freq)
head(under_e_coli_counts_3,10)
```

```
##      Var1 Freq
```

```
## 1   WMW   5
## 2   WWW   6
## 3   YMW   6
## 4   MWW   7
## 5   NWW   7
## 6   WWT   7
## 7   FKW   9
## 8   WWM   9
## 9   WYW   9
## 10  KWW  10
```

The 3 length K-mers underrepresented in *Salmonella* and *E.coli*, share similar K-mers, with majority of the K-mers shown being in both organism, with a few exceptions of 'FKW' in *Salmonella* and 'MWN' in *E.coli*, the shared K-mers do vary on the amount of representation In the gene. The identification of the underrepresented 3 length K-mers were found by using the `arrange()` command, with a + positive frequency, to order the data frame in increasing order.

Creating data frame of 4 length K-mers in Salmonella

```
salmonella_counts_4 <- count(salmonella_prots,wordsize=4,alphabet=aa)
head(salmonella_counts_4)
```

```
##
## AAAA AAAD AAAE AAAF AAAG AAAI
## 196  58  81  58 141 104
```

```
df_salmonella_counts_4 <- as.data.frame(salmonella_counts_4)
head(df_salmonella_counts_4)
```

```
##   Var1 Freq
## 1 AAAA  196
## 2 AAAD   58
## 3 AAAE   81
## 4 AAAF   58
## 5 AAAG  141
## 6 AAAI  104
```

A data frame of 4 length K-mers was created for *Salmonella* to be able to sort over represented and underrepresented 4 length k-mers. The data frame was created with the `as.data.frame()` command.

Creating data frame of 4 length K-mers in E.coli

```
e_coli_counts_4 <- count(e_coli_prots,wordsize=4,alphabet=aa)
head(e_coli_counts_4)
```

```
##
## AAAA AAAD AAAE AAAF AAAG AAAI
## 148  56  76  38 100  88
```

```
df_e_coli_counts_4 <- as.data.frame(e_coli_counts_4)
head(df_e_coli_counts_4)
```

```
##   Var1 Freq
## 1 AAAA  148
## 2 AAAD   56
## 3 AAAE   76
## 4 AAAF   38
```

```
## 5 AAAG 100
## 6 AAAI 88
```

A data frame of 4 length K-mers was created for *E.coli* to be able to be able to sort over represented and underrepresented 4 length k-mers. The data frame was created with the `as.data.frame()` command.

Top 10 most overrepresented 4 length K-mers in Salmonella

```
over_salmonella_counts_4 <-arrange(df_salmonella_counts_4,-Freq)
head(over_salmonella_counts_4,10)
```

```
##      Var1 Freq
## 1  ALAA  255
## 2  ALLA  252
## 3  LLAL  239
## 4  AALA  238
## 5  LAAL  233
## 6  LLLL  224
## 7  LLAA  218
## 8  LALA  214
## 9  ALAL  213
## 10 LALL  212
```

Top 10 most overrepresented 4 length K-mers in E.coli

```
over_e_coli_counts_4 <-arrange(df_e_coli_counts_4,-Freq)
head(over_e_coli_counts_4,10)
```

```
##      Var1 Freq
## 1  LAAL  234
## 2  ALAA  233
## 3  LLAL  215
## 4  LLLL  209
## 5  AALA  207
## 6  LLAA  206
## 7  LLLA  197
## 8  LALA  196
## 9  LALL  195
## 10 ALLA  193
```

The 4 length K-mers overrepresented in *Salmonella* and *E.coli*, share similar K-mers, with majority of the K-mers shown being in both organisms, with a few exceptions of 'ALAL' in *Salmonella* and 'LLLA' in *E.coli*, the shared K-mers do vary on the amount of representation In the gene. The identification of the over represented 4 length K-mers were found by using the `arrange()` command, with a - negative frequency, to order the data frame in decreasing order.

Top 10 most underrepresented 4 length K-mers in Salmonella

```
under_salmonella_counts_4 <-arrange(df_salmonella_counts_4,Freq)
head(under_salmonella_counts_4,10)
```

```
##      Var1 Freq
## 1  ADPW    0
## 2  AFKM    0
## 3  AFMW    0
```



```
## 4 AGWW 0
## 5 AKQW 0
## 6 AKWN 0
## 7 AKWP 0
## 8 AKWW 0
## 9 AMMF 0
## 10 AMNW 0
```

Top 10 most underrepresented 4 length K-mers in E.coli

```
under_e_coli_counts_4 <- arrange(df_e_coli_counts_4, Freq)
head(under_e_coli_counts_4, 10)
```

```
##   Var1 Freq
## 1 AEMW  0
## 2 AEWW  0
## 3 AEWY  0
## 4 AFMK  0
## 5 AFWI  0
## 6 AKWF  0
## 7 AKWT  0
## 8 AMMW  0
## 9 AMNW  0
## 10 AMWE  0
```

The 4 length K-mers underrepresented in *Salmonella* and *E.coli*, all show 0 representation, as there is just 10 K-mers shown, there are other K-mers also at 0 representation, so while there are limited shared K-mers between both organisms shown in the results, more shared K-mers of underrepresentation cannot be ruled out. The identification of the underrepresented 4 length K-mers were found by using the `arrange()` command, with a `+` positive frequency, to order the data frame in increasing order.

Creating data frame of 5 length K-mers in Salmonella

```
salmonella_counts_5 <- count(salmonella_prots, wordsize=5, alphabet=aa)
head(salmonella_counts_5)
```

```
##
## AAAAA AAAAD AAAAE AAAAF AAAAG AAAAI
##   45   13    9    4   17    8
```

```
df_salmonella_counts_5 <- as.data.frame(salmonella_counts_5)
head(df_salmonella_counts_5)
```

```
##   Var1 Freq
## 1 AAAAA  45
## 2 AAAAD  13
## 3 AAAAE   9
## 4 AAAAF   4
## 5 AAAAG  17
## 6 AAAAI   8
```

A data frame of 5 length K-mers was created for *Salmonella* to be able to sort over represented and underrepresented 5 length k-mers. The data frame was created with the `as.data.frame()` command.

Creating data frame of 5 length K-mers in E.coli

```
e_coli_counts_5 <- count(e_coli_prots,wordsize=5,alphabet=aa)
head(e_coli_counts_5)
```

```
##
## AAAAA AAAAD AAAAE AAAAF AAAAG AAAAI
##    27    5    10    4    13    10
df_e_coli_counts_5 <- as.data.frame(e_coli_counts_5)
head(df_e_coli_counts_5)
```

```
##      Var1 Freq
## 1 AAAAA    27
## 2 AAAAD     5
## 3 AAAAE    10
## 4 AAAAF     4
## 5 AAAAG    13
## 6 AAAAI    10
```

A data frame of 5 length K-mers was created for *E.coli* to be able to sort over represented and underrepresented 5 length k-mers. The data frame was created with the `as.data.frame()` command.

Top 10 most overrepresented 5 length K-mers in Salmonella

```
over_salmonella_counts_5 <- arrange(df_salmonella_counts_5, -Freq)
head(over_salmonella_counts_5, 10)
```

```
##      Var1 Freq
## 1 GKSTL    55
## 2 AAAAA    45
## 3 LLLAL    43
## 4 AALAA    42
## 5 GSGKS    40
## 6 AALLA    39
## 7 ALAAL    37
## 8 LAAAL    36
## 9 ALALA    33
## 10 LLALA    33
```

Top 10 most overrepresented 5 length K-mers in E.coli

```
over_e_coli_counts_5 <- arrange(df_e_coli_counts_5, -Freq)
head(over_e_coli_counts_5, 10)
```

```
##      Var1 Freq
## 1 GKSTL    58
## 2 GSGKS    42
## 3 AALAA    37
## 4 LLAAL    37
## 5 SGSGK    37
## 6 LLLDE    35
## 7 LAAAL    34
## 8 LLLAL    34
## 9 LLLLL    34
## 10 LDEPT    33
```

The 5 length K-mers overrepresented in *Salmonella* and *E.coli*, some similar K-mers, with approximately half being shown in both organisms, with the exceptions of “AAAAA”, “AALLA”, “ALAAL”, “ALALA” in *Salmonella* and “SGSGK”, “LLLDE”, “LLLLL”, “LDEPT” in *E.coli*, the shared K-mers do vary on the amount of representation In the gene. The identification of the overrepresented 5 length K-mers were found by using the `arrange()` command, with a - negative frequency, to order the data frame in decreasing order.

Top 10 most underrepresented 5 length K-mers in Salmonella

```
under_salmonella_counts_5 <-arrange(df_salmonella_counts_5,Freq)
head(under_salmonella_counts_5,10)
```

```
##      Var1 Freq
## 1  AAAAW    0
## 2  AAADP    0
## 3  AAADQ    0
## 4  AAADS    0
## 5  AAAEW    0
## 6  AAAEY    0
## 7  AAAFK    0
## 8  AAAFM    0
## 9  AAAIF    0
## 10 AAAIW    0
```

Top 10 most underrepresented 5 length K-mers in E.coli

```
under_e_coli_counts_5 <-arrange(df_e_coli_counts_5,Freq)
head(under_e_coli_counts_5,10)
```

```
##      Var1 Freq
## 1  AAAAW    0
## 2  AAADF    0
## 3  AAADM    0
## 4  AAADN    0
## 5  AAADS    0
## 6  AAADW    0
## 7  AAAEE    0
## 8  AAAEW    0
## 9  AAAFD    0
## 10 AAAFK    0
```

The 5 length K-mers underrepresented in *Salmonella* and *E.coli*, all show 0 representation, as there is just 10 K-mers shown, there are other K-mers also at 0 representation, so while there are limited shared K-mers between both organisms shown in the results, more shared K-mers of underrepresentation cannot be ruled out. The identification of the underrepresented 5 length K-mers were found by using the `arrange()` command, with a + positive frequency, to order the data frame in increasing order.

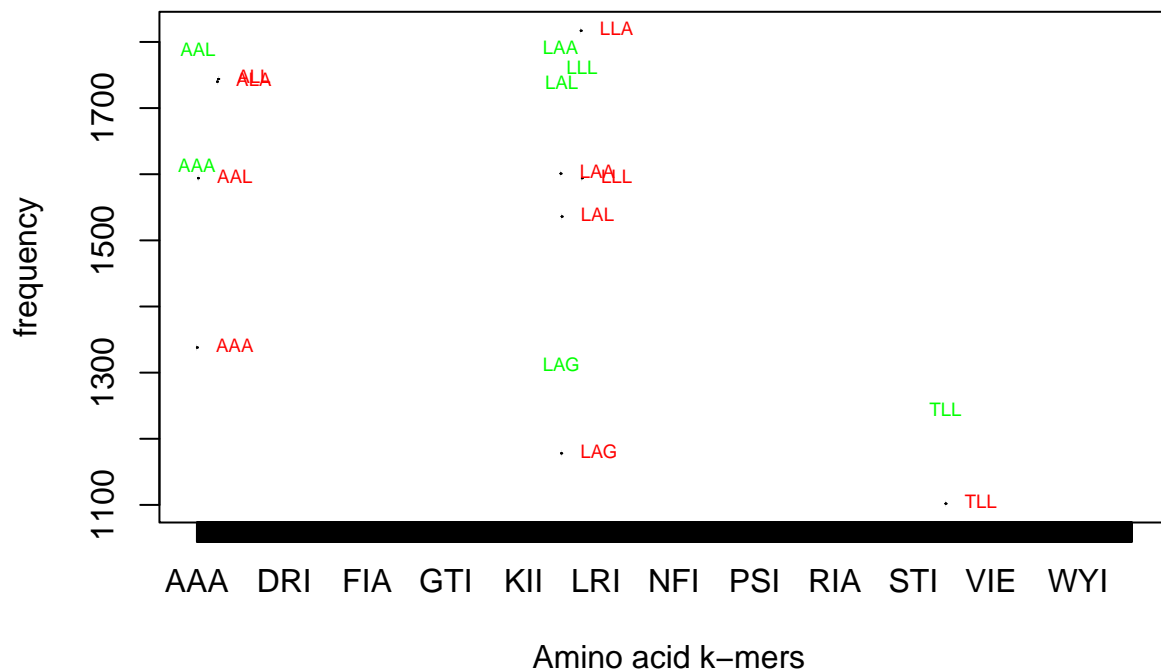
Plot of K-mers of 3 amino acids overrepresented of *Salmonella* compared to *E.coli*

```
ecoli_3_over <- data.frame(over_e_coli_counts_3[,-1], row.names=over_e_coli_counts_3[,1])
salmonella_3_over <- data.frame(over_salmonella_counts_3[,-1], row.names=over_salmonella_counts_3[,1])

plot(head(over_e_coli_counts_3,10),head(over_salmonella_counts_3,10),
     main = "K-mers of 3 amino acids over represented of *Salmonella* compared to E.coli",
     xlab = " Amino acid k-mers", ylab = "frequency")
```

```
text(head(over_e_coli_counts_3,10),
     labels = row.names(head(ecoli_3_over,10)),
     cex = 0.6, pos = 4, col = "red")
text(head(over_salmonella_counts_3,10),
     labels = row.names(head(salmonella_3_over,10)),
     cex = 0.6, pos = 3, col = "green")
```

-mers of 3 amino acids over represented of *Salmonella* compared to



The plot above shows the overrepresented 3 Amino acid length K-mers in both *Salmonella* (Green) and *E. coli* (red). The plot shows the labelled K-mers and their corresponding frequency, and the closeness in shared K-mers between both organisms, but also shows the variation in frequency between the K-mers when expressed in *Salmonella* compared to *E. coli*.

Plot of K-mers of 3 amino acids underrepresented in *Salmonella* compared to *E. coli*

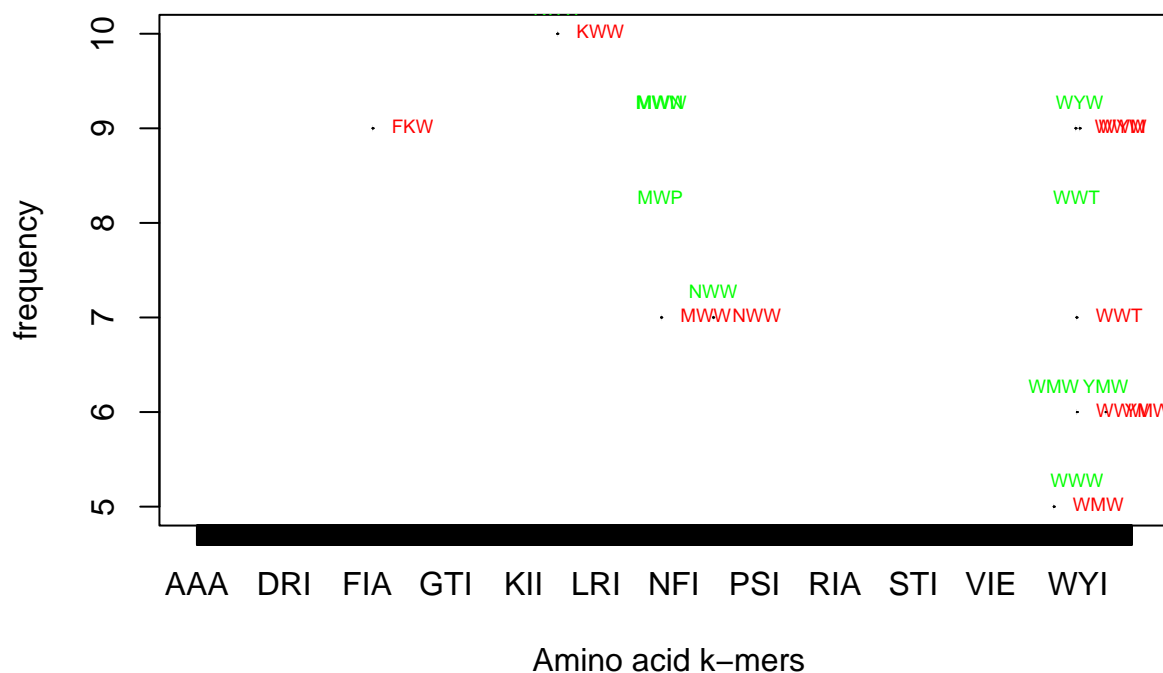
```
ecoli_3_under <- data.frame(under_e_coli_counts_3[,-1], row.names=under_e_coli_counts_3[,1])
salmonella_3_under <- data.frame(under_salmonella_counts_3[,-1], row.names=under_salmonella_counts_3[,1])

plot(head(under_e_coli_counts_3,10),head(under_salmonella_counts_3,10),
     main = "K-mers of 3 amino acids under represented of *Salmonella* compared to E.coli",
     xlab = " Amino acid k-mers", ylab = "frequency")

text(head(under_e_coli_counts_3,10),
     labels = row.names(head(ecoli_3_under,10)),
     cex = 0.6, pos = 4, col = "red")
text(head(under_salmonella_counts_3,10),
```

```
labels = row.names(head(salmonella_3_under,10)),
cex = 0.6, pos = 3, col = "green")
```

-mers of 3 amino acids under represented of *Salmonella* compared to



The plot above shows the underrepresented 3 Amino acid length K-mers in both *Salmonella* (green) and *E.coli* (red). The plot shows the labelled K-mers and their corresponding frequency, and the closeness in shared K-mers between both organisms, but also shows the variation in frequency between the K-mers when expressed in *Salmonella* compared to *E.coli*.

The variation in K-mer expression in *Salmonella* and *E.coli* is due to the differences in nucleotide sequences which create the proteins, which then create the 3,4,5 length K-mers.

Session Info

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 22.04.5 LTS
##
## Matrix products: default
## BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0
##
## locale:
## [1] LC_CTYPE=C.UTF-8 LC_NUMERIC=C LC_TIME=C.UTF-8
## [4] LC_COLLATE=C.UTF-8 LC_MONETARY=C.UTF-8 LC_MESSAGES=C.UTF-8
## [7] LC_PAPER=C.UTF-8 LC_NAME=C LC_ADDRESS=C
```

```

## [10] LC_TELEPHONE=C          LC_MEASUREMENT=C.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats      graphics  grDevices  utils      datasets  methods   base
##
## other attached packages:
## [1] coRdon_1.12.0      BiocManager_1.30.25 readr_2.1.5
## [4] R.utils_2.12.3     R.oo_1.26.0        R.methodsS3_1.8.2
## [7] seqinr_4.2-36      dplyr_1.1.4
##
## loaded via a namespace (and not attached):
## [1] tidyselect_1.2.1      xfun_0.48            purrr_1.0.2
## [4] colorspace_2.1-1      vctrs_0.6.5          generics_0.1.3
## [7] htmltools_0.5.8.1     stats4_4.1.2         yaml_2.3.10
## [10] utf8_1.2.4            rlang_1.1.4          pillar_1.9.0
## [13] glue_1.8.0            BiocGenerics_0.40.0  GenomeInfoDbData_1.2.7
## [16] lifecycle_1.0.4       stringr_1.5.1        zlibbioc_1.40.0
## [19] Biostrings_2.62.0     munsell_0.5.1        gtable_0.3.5
## [22] evaluate_1.0.1        Biobase_2.54.0       knitr_1.48
## [25] tzdb_0.4.0            IRanges_2.28.0       fastmap_1.2.0
## [28] GenomeInfoDb_1.30.1   fansi_1.0.6          highr_0.11
## [31] Rcpp_1.0.13           scales_1.3.0         S4Vectors_0.32.4
## [34] XVector_0.34.0        ggplot2_3.5.1        hms_1.1.3
## [37] digest_0.6.37         stringi_1.8.4        grid_4.1.2
## [40] ade4_1.7-22           cli_3.6.3            tools_4.1.2
## [43] bitops_1.0-9          magrittr_2.0.3       RCurl_1.98-1.16
## [46] tibble_3.2.1          crayon_1.5.3         pkgconfig_2.0.3
## [49] MASS_7.3-55           data.table_1.16.2    rmarkdown_2.28
## [52] rstudioapi_0.16.0     R6_2.5.1             compiler_4.1.2

```