Assessment 4_SLE777_R Project

Ameeta

2025-10-05

Gene expression Step 1: Download file for geneexpression.tsv

The link for raw file were copied from the github and then downloaded in R-markdown using te download.file function and then the file was downloaded locally with the name geneexpression.tsv using destfile argument. The file was sucessfully downloaded.

```
# download the gene expression file

download.file("https://raw.githubusercontent.com/ghazkha/Assessment4/refs/heads/main/gene_expression.ts
```

Step 1: Reading gene expression file with gene identifiers as the row names and dispaying first 6 genes

The downloaded file was read using the read.table function, where the header argument specifies that the first row contain the name of the columns, while sep= "^ indicates that the file is in TSV format. The row.names =1 was used to set the first column as the names of gene identifiers, stringAsFactors=FALSE ensures that the data are represented as plain texts rather than factors. The obtained dataset was saved in the object gene_data. Following that, to inspect the contents, the head function was used display the first 6 rows, showing 6 gene identifiers.The value 6 was used to ensure onky 6 rows are displayed in the table. The first six rows containig 6 gene identifiers were obtained with 3 columns.

```
# Read the downloaded file
gene_data <- read.table("geneexpression.tsv",     # path to the file
                        header=TRUE,       # indicates first row of the file contains column names
                        sep = "\t",        # \t is the standard for tsv files
                        row.names = 1,    #indicates that first colum contains row names
                        stringsAsFactors = FALSE )  # keep as plain character strings

# display the first 6 genes of the file
head(gene_data, 6) # 6 inidcates number of rows to be displayed
```

```
##                                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
```

Step 2: Generating a new column with mean value of other columns and displaying 6 genes

A new column called meanexpression, which contain the mean value of other columns was created in the table using rowMeans function which calculate the mean of other columns for a given row. The output was saved in meanexpression cloumn under gene_data object. To display the output, the rows 1:6 values were selected to display first six genes and column 1 and ncol(last column) was selected to display the first and the last column (meanexpression column). The column for mean value of other columns were successfully generated.

```
gene_data$meanexpression <- rowMeans(gene_data) # rowMeans calculate means across all columns for each
gene_data[1:6, c(1, ncol(gene_data))] # 1:6 selects forst 6 genes of the data
```

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

```
                                    # c(1, ncol(gene_data) selects the first and the last column
```

Step 3: Listing the 10 genes with the highest mean expression

Firstly, the meanexpression in gene_data were ordered in the descending order using order function followed by negative (-) symbol just before the gene_data$meanexpression and then it was saved in gene_data_sorted file. After that, first 10 rows containing 10 genes with highest meanexpression were displayed in gene_data_sorted using a drop argument to ensure the datas are obtained in table format and not in vector format. The 10 genes with the highest mean expression was generated. The gene with the highest mean expression value was ENSG00000198804.2_MT-CO1 with mean value of 529317.3.

```
# order the "meanexpression" of the gene-data in descending order and save in gene_data-sorted file
gene_data_sorted <- gene_data[order(-gene_data$meanexpression), ] # order(-gene_data$meanofcolumns) sor
# show 10 genes with the highest mea expression values
gene_data_sorted[1:10, "meanexpression", drop = FALSE] # drop =FALSE ensures datas are expressed in dat
```

```
##                           meanexpression
## ENSG00000198804.2_MT-CO1       529317.3
## ENSG00000198886.2_MT-ND4       514235.7
## ENSG00000198938.2_MT-CO3       504943.7
## ENSG00000198888.2_MT-ND1       403617.0
## ENSG00000198899.2_MT-ATP6      329751.7
## ENSG00000198727.2_MT-CYB       302254.0
## ENSG00000198763.3_MT-ND2       284217.7
## ENSG00000211445.11_GPX3        270141.7
## ENSG00000198712.1_MT-CO2       265678.0
## ENSG00000156508.17_EEF1A1      232187.3
```

Step 4. Determining the number of genes with a mean <10

to determine the genes with the meanexpression less than 10, firstly the logical vector was created that tests each gene's mean expression value where the line checks for every gene in the dataset whethere its meanexpression value is less than 10. So, if the value is less than 10, the result is true and if it is not, the result is FALSE. The output was then saved in gene_data_mean_10. Then to check how many genes meet the condition of meanexpression < 10, the sum() function was used. As the TRUE is treated as 1 and FALSE as 0 in R, summing the logical vector gives the total number of genes with meanexpression value less than 10. A total of 35,988 genes contain the mean expression value less then 10.

```
# create logical vectors for genes with meanexpression <10
gene_data_mean_10 <- gene_data_sorted$meanexpression <10
# count the total number of genes with mean <10
```

```r
sum(gene_data_mean_10) # sum was used for summing all the logocal vectors
```

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

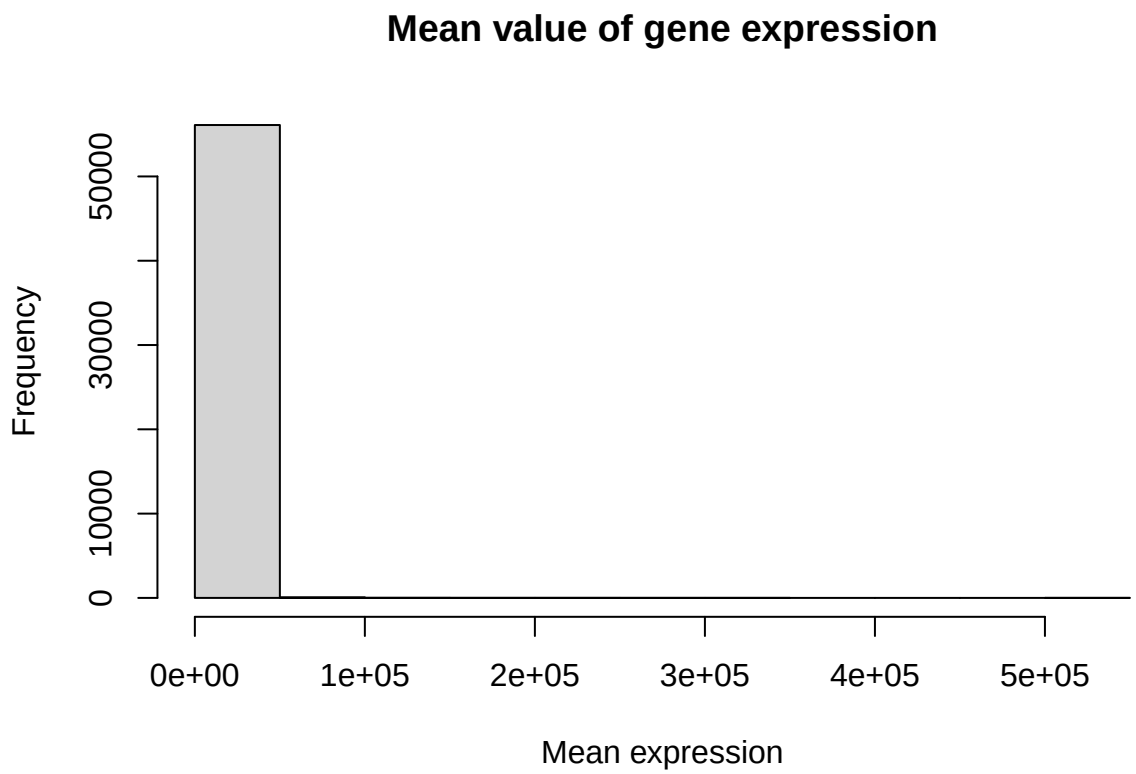
Step 5: Making a histogram plot of the mean values

A histogram was generated using hist() function to represent the distribution of the mean expression value of the genes. The hist() function takes the meanexpression column from the gene_data dataset and plot its frequency distribution, where the gene_data$expression specifies the data to plot, xlab="mean expression", add descriptive label to the x-axis and main="mean value of gene expression" add the title for the plot. As shown in the graph, the majority of the genes have a very low mean expression as indicated by the tallest bin near zero. However, there are a few genes which has extremely higher mean expression (5e+05) but appeared to be empty due to the dominance by the tall bar for low gene expression.

```r
# Step 6: Histogram of mean values of gene expression
data(gene_data)
```

```
## Warning in data(gene_data): data set 'gene_data' not found
```

```r
hist(gene_data$meanexpression, # data to be plotted
     xlab="Mean expression", # label for the x-axis
     main="Mean value of gene expression") # Main title of the histogram
```



GROWTH DATA INTERPRETATION

Step 6: Importing the csv file into an R object

The csv raw file for growth data was downloaded using download.file function and was saved locally as growth_data. then the file was imported into R as a data frame using the read.csv() function. This command reads the CSV file and then the file was stored in the growth_data object. Following that the colnames()

function was used to display all the column names of the data set. This allowed the identification of all the column headers present in the imported dataset. The dataset contain six columns namely site, TreeID, Circumf_2005_cm, Circumf_2015_cm, and Circumf_2020_cm.

```
# download the growth data file

download.file("https://raw.githubusercontent.com/ghazkha/Assessment4/refs/heads/main/growth_data.csv", d

# Read file
growth_data <- read.csv("growthdata.csv")
colnames(growth_data)
```

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

Step 7: Calculating the mean and standard deviation of tree circumference at the start and end of the study at both sites.

Firstly, the mean and standard deviation for each combination of site and year was calculated to summarize the tree circumference at the start (2005) and end (2020) of study for both sites. For this the data was subset by using logocal codition northeast and southwest site followed by application of mean() fuction to calculate the average circumference and the sd() function for calculating the mean deviation. Similarly, the steps were repeated for the northeast end (2020), southwest start(2005), southwest end (2020). After all the calculations were completed, they were combined into one clear summary atble using the data.frame() function and finally the table was displayed.This provided the summary of the average tree size and their variabilitiy at each site over the period of study. As shown in the table, the mean of the tree circumference increased at both the sites from 2005 to 2020. For the northeast site, the mean increased from 5.292 cm to 54.248 cm whereas for the southwest site, the mean increased from 4.862 to 45.596 cm. When compared at both sites, the northeast site recorded slightly higher mean circumfernece than the southwest at both start and end of study periods which could be attributed to slightly better growth conditions or initial size advantage. The standard variation increased dramatically from 2005 to 2020 from 0.91 to 25.22 for northeast site and from 1.14 to 17.87 for southwest site, indicating tree sizes become more variable at the end where some trees grew larger than others at both the sites.

```
# 1. Mean for Northeast site at the start (Circumf_2005_cm)

meannortheast_start<- mean(growth_data$Circumf_2005_cm[growth_data$Site== "northeast"]) # growth_data$S

# 2. standard deviation for the Northeast site at the start (Circumf_2005_cm)

sdnortheast_start <- sd(growth_data$Circumf_2005_cm[growth_data$Site== "northeast"])

# 3. Mean for Northeast site at the start (Circumf_2020_cm)

meannortheast_end <- mean(growth_data$Circumf_2020_cm[growth_data$Site== "northeast"])

# 4. standard deviation for the Northeast site at the start (Circumf_2005_cm)

sdnortheast_end <- sd(growth_data$Circumf_2020_cm[growth_data$Site== "northeast"])

# 5. Mean for Southwest site at the start (Circumf_2005_cm)

meansouthwest_start<- mean(growth_data$Circumf_2005_cm[growth_data$Site== "southwest"]) # growth_data$S

# 6. standard deviation for the Southwest site at the start (Circumf_2005_cm)
```

```r
sdsouthwest_start <- sd(growth_data$Circumf_2005_cm[growth_data$Site== "southwest"])


# 7. Mean for Southwest site at the end (Circumf_2020_cm)

meansouthwest_end<- mean(growth_data$Circumf_2020_cm[growth_data$Site== "southwest"])

# 8. standard deviation for the Southwest site at the end (Circumf_2020_cm)

sdsouthwest_end <- sd(growth_data$Circumf_2020_cm[growth_data$Site== "southwest"])

# creating summary table for mean and standard deviation of two sites at the start and end of study per

summary_table <- data.frame(
  Site = c("Northeast", "Northeast", "Southwest", "Southwest"),
  Year = c("2005 (Start)", "2020 (End)", "2005 (Start)", "2020 (End)"),
  Mean = c(meannortheast_start, meannortheast_end, meansouthwest_start, meansouthwest_end),
  SD = c(sdnortheast_start, sdnortheast_end, sdsouthwest_start, sdsouthwest_end)
)

# Displaying the table
summary_table
```

```
##        Site         Year   Mean         SD
## 1 Northeast 2005 (Start)  5.292  0.9140267
## 2 Northeast   2020 (End) 54.228 25.2279489
## 3 Southwest 2005 (Start)  4.862  1.1474710
## 4 Southwest   2020 (End) 45.596 17.8734549
```

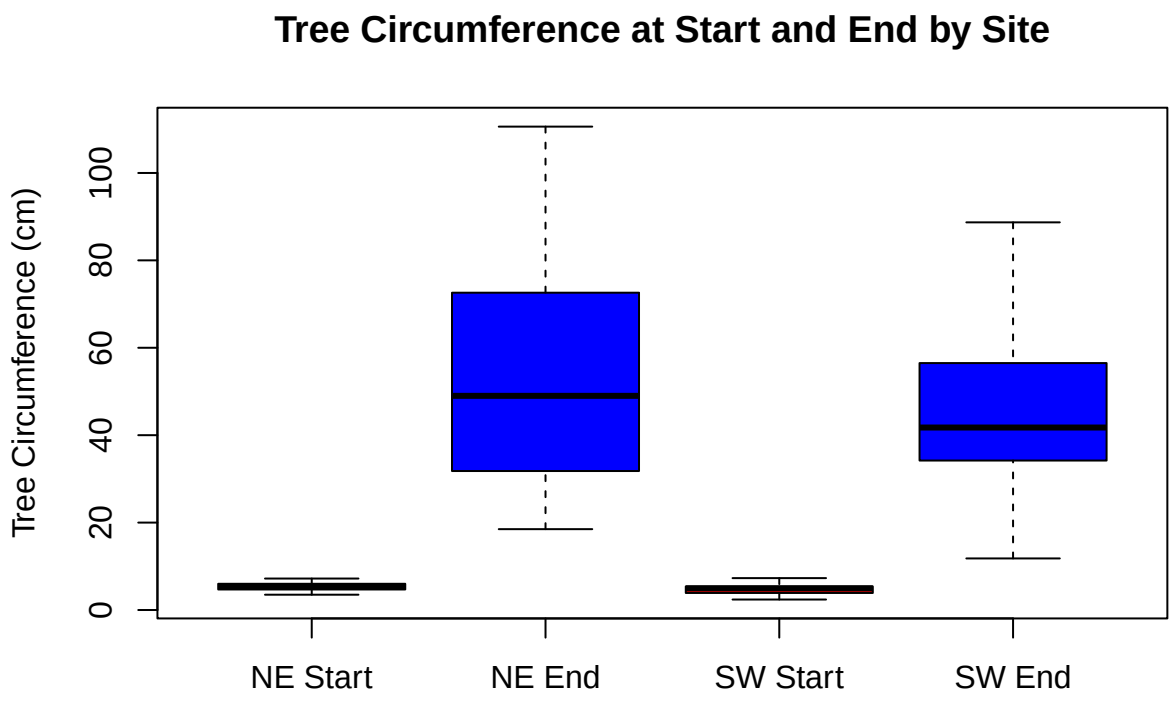
step 8: Making a box plot for the circumference at the start and end of the study at both sites.

Boxplots was created to represent the distribution of the tree circumferences at the start and end of the study for both the sites. Firstly, the datas were splitted into Northeast and Southwest. Then, the boxplot() function was used to display the tree circumferences for both sites at both time points. To plot the box for both the sites, multiple datatsets were placed one after another inside the same boxplot() function, separated by commas. This automatically placed the boxplots side by side in the same figure for easy comparison.

For both northeast and southwest, the tree circumference increased substantially from 2005 to 2020, indicating significant tree growth over period of time. The northeast exhibited slightly higher median and tree circumference than southwest site at both the time points, indicating that trees in the northeast experience slightly greater growth. The data was spreaded largely in 2020 box as shown by interquartile range, reflecting greater variation in tree sizes after 15 years of growth. This indicated that some trees grew significantly larger than others which could be attributed to environmental factors. Overall, the box displayes increased tree size and variability over time at both study sites.

```r
# Splitting of data into Northeast and Southwest

northeast <- growth_data[growth_data$Site == "northeast", ]
southwest <- growth_data[growth_data$Site == "southwest", ]


# Creating boxplots for two different sites at the start and end of the study periods

boxplot(northeast$Circumf_2005_cm, northeast$Circumf_2020_cm,
        southwest$Circumf_2005_cm, southwest$Circumf_2020_cm,
        names = c("NE Start","NE End", "SW Start", "SW End"), # Provides the names to each boxplot
```

```
        col = c("red", "blue", "red", "blue"), # provides red color to the boxplot at the start and blu
        ylab = "Tree Circumference (cm)",
        main = "Tree Circumference at Start and End by Site")
```



Step 9: Calculating the mean growth over the last 10 years at each site.

To determine the mean tree growth over the last 10 years(from 2010 to 2020) at each study site, the difference in circumference between 2020 and 2010 was calculated for each tree and the output was saved in the new column called growth_10_years under object growth_data. Then, data were sepa rated by site to calculate the mean growth for each location. Following that, the mean () function was used separately to the northeast and southwest subsets to obtain the average tree growth over the 10-year period for each site.

```
# calculate growth data from 2010 to 2020

growth_data$ growth_10_years <- (growth_data$Circumf_2020_cm - growth_data$Circumf_2010_cm)

# Extracting 10-year growth values for northsite
North_east_growth_data <- (growth_data$growth_10_years[growth_data$Site == "northeast"])

# calculating mean for 10-year growth values of northwest
North_east_mean_growth_data <- mean(North_east_growth_data)
North_east_mean_growth_data
```

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

```
# Extracting 10-year growth values for southwest
South_west_growth_data <- (growth_data$growth_10_years[growth_data$Site == "southwest"])
```

```r
# calculating mean for 10-year growth values of southwest

Southwest_mean_growth_data <- mean (South_west_growth_data)
Southwest_mean_growth_data
```

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

Step 10: Using the t.test to estimate the p-value that the 10 year growth is different at the two sites

The p-value for the two sites were determined using t-test function. The north_east_growth_data which contains the last 10 years growth data for northeast site and South_west_growth_data which contains last 10 years growt data for southwest site were used to compare the growth between the two sites.The study observed higher mean 10-year growth of 42.94 at the northeast site compared to soutwest site (35.49 cm) but the growth difference was not statistically significant at 5% significance level (t-value = 1.8882, df = 87.978, and p-value = 0.06229).

```r
t.test(North_east_growth_data,South_west_growth_data)
```

```
##
##  Welch Two Sample t-test
##
## data:  North_east_growth_data and South_west_growth_data
## t = 1.8882, df = 87.978, p-value = 0.06229
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.3909251 15.2909251
## sample estimates:
## mean of x mean of y
##     42.94     35.49
```

PART 2: Examining the biological sequence diversity

step 1: Downloading the Saprospirale and ecoli sequences, counting the coding sequences (CDS) and comparing the sequences between the two organisms.

The codes such as sequinr and R.utils were used for smooth downloading, unzipping and reading of CDS data. The compressed FASTA files (.fa.gz) for both the E.coli and Saprospirales were downloaded from the ENSEMBL website using the download.file() function. Since the downloaded file was in gzip format, the gunzip() function was used from the R.utils package to extract the .fa files. The overwrite argument was used to overwrite the unzipped file as R usually refuses to overwrite it. The read.fasta() function from the sequinr package was used to load the sequence into R as lists, where each element represents a coding sequence. Then, the length() function was applied on each list of sequences to determine the total number of CDS for each organism. Finally the table was created using data.frame() function with two columns , one for the organism name and one for the number of coding sequences, and displayed it as a table.

As shown in the table, the Saprospirales contain 4527 coding sequences whereas the E.coli contains 4239 coding sequences. Therefore, the saprospirales has slightly more number of coding sequences than E.coli, which indicates that saprospirales have comparatively more complex genome which could be attributed to their adaptation to diverse environmental conditions.

```r
suppressPackageStartupMessages({
  library("seqinr") # is a package designed to process and analyse sequence data.
  library("R.utils") # general utilities like zip and unzip
})
# loading e.coli and Saprospirales data
library("R.utils")

# Download FASTA file for saprospirales
```

```r
URL="https://ftp.ensemblgenomes.ebi.ac.uk/pub/bacteria/release-62/fasta/bacteria_58_collection/saprospi
download.file(URL,destfile="saprospirales_cds.fa.gz")

# Download FASTA file for E.coli

URL="http://ftp.ensemblgenomes.org/pub/bacteria/release-53/fasta/bacteria_0_collection/escherichia_coli
download.file(URL,destfile="ecoli_cds.fa.gz")

# Uncompress the FASTA files for e.coli

gunzip("ecoli_cds.fa.gz", overwrite  = TRUE)
# overwrite = TRUE tells R to replace the uncompressed .fa file if it already exist in the working dire

# Uncompress the FASTA file for Saprospirales
gunzip("saprospirales_cds.fa.gz", overwrite  = TRUE) # gunzip is used to uncompress the file

# list files

list.files()
```

```
##  [1] "Assessment 4_R Project_SLE777.Rmd"
##  [2] "Assessment-4_R-Project_SLE777_files"
##  [3] "Assessment-4_R-Project_SLE777.pdf"
##  [4] "Assessment-4_R-Project_SLE777.Rmd"
##  [5] "Assessment-4_SLE777_Finalised_Ameeta.Rproj"
##  [6] "ecoli_cds.fa"
##  [7] "geneexpression.tsv"
##  [8] "growthdata.csv"
##  [9] "LICENSE"
## [10] "Part 1_Assessment 4.Rmd"
## [11] "Part-1_Assessment-4.pdf"
## [12] "R-Project.html"
## [13] "R-Project.Rmd"
## [14] "README.md"
## [15] "saprospirales_cds.fa"
```

```r
# Read the FASTA sequences for e.coli
library("seqinr")
e.coli_cds <- seqinr::read.fasta("ecoli_cds.fa") # read.fasta is used to read the FASTA file

# Read the FASTA sequences for Saprospirales
saprospirales_cds <- seqinr::read.fasta("saprospirales_cds.fa")

# Count the number of coding sequences for e.coli
e.coli_number <- length(e.coli_cds)

# Count the number of coding sequences for Saprospirales
saprospirales_num <- length(saprospirales_cds)

# create table
cds_table <- data.frame(
  Bacteria = c("E.coli", "Saprospirales"),
  CDS_count = c(e.coli_number, saprospirales_num)
)
```

```
cds_table
```

```
##        Bacteria CDS_count
## 1        E.coli      4239
## 2 Saprospirales      4527
```

Step 2: Determing and comparing the total coding DNA between the two organisms.

To determine the total length of the the sequences in both organisms, the length of each coding sequences was extracted from the respective organisms' cds using summary() function and the first column of the summary was converted to the numeric value using as.numeric() function. Then the total length of all the genes were calculated by summing these values using the sum() function. Then, to display the compiled result of two organisms, the table was created using data.frame() function with columns for organism, number of CDA and total coding DNA.

It was observed that the Saprospirales has greater total coding DNA (4200321) than that of E.coli (3978528). This indicates that Saprospirales has a larger or more complex genome with potentially more genes or longer coding sequences, which could be attributed to adaptation to a diverse environmental condition.

```
# Calculate the CDS length for e.coli
e.coli_cds_length<- as.numeric(summary(e.coli_cds)[,1]) # 1 select the first column of the matrix

# Calculate the CDS length for Saprospirales
saprospirales_cds_length <- as.numeric(summary(saprospirales_cds)[,1])

# sum total coding DNA of E.coli
e.coli_total_cds_length <- sum(e.coli_cds_length)

#Sum total coding DNA of Saprospirales
saprospirales_total_cds_length <- sum(saprospirales_cds_length)

# create table for the total coding DNA for both organisms
cds_table <- data.frame(
  Bacteria = c("E.coli", "Saprospirales"),
  CDS_count = c(e.coli_number, saprospirales_num),
  total_coding_DNA = c(e.coli_total_cds_length, saprospirales_total_cds_length)
)
cds_table
```

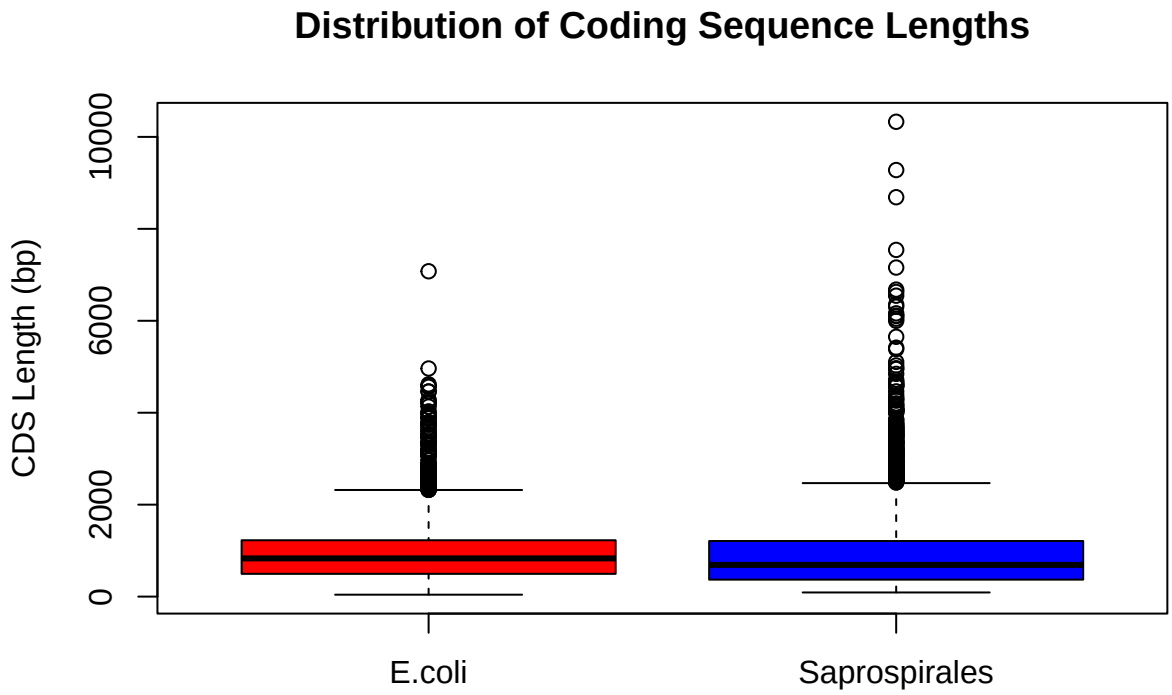
```
##        Bacteria CDS_count total_coding_DNA
## 1        E.coli      4239          3978528
## 2 Saprospirales      4527          4200321
```

Step 3: Calculate the length of all coding sequences in these two organisms. Make a boxplot of coding sequence length in these organisms. What is the mean and median coding sequence length of these two organisms? Describe any differences between the two organisms.

The length of each coding sequences was extracted from the respective organisms' cds using summary() function and the first column of the summary was converted to the numeric value using as.numeric() function. Then the boxplot() was used to display the distribution of coding aequence lengths in both organisms where red color represented the E.coli coding seuqnece length and blue color represented the Saprospirales coding sequence length.Following that the mean() and median() function was used to summarize the central tendency of CDS lengths. To compare the result between two organisms, the table was generated using data.frame() function, representing the value of CDS count, total coding DNA, mean and median CDS length. The mean coding sequence length of E.coli is 938.5534 and the median is 831. Whereas in case of Saprospirales, the mean coding sequence length is 927.8376 and the median is 690. This shows that mean CDS lnegth of E.coli is slightly higher than Saprospirales, indicating that on average E.coli genes are bit longer. Similarly, the

median CDS length is substantially higher than Saprospirales, indicating the most E.coli genes are moderately long, whereas Saprospirales has a larger proportion of shorter genes.

```
# Calculate the CDS length for E.coli
e.coli_cds_length<- as.numeric(summary(e.coli_cds)[,1]) # 1 select the first column of the matrix

# Calculate the CDS length for Saprospirales
saprospirales_cds_length <- as.numeric(summary(saprospirales_cds)[,1])


# Generate box plot for E.coli and Saprospirales
boxplot(list(E.coli = e.coli_cds_length, Saprospirales = saprospirales_cds_length),
        col = c("red", "blue"),
        ylab = "CDS Length (bp)",
        main = "Distribution of Coding Sequence Lengths")
```




```
# Calculate mean and median of E.coli
e.coli_mean_cds_length <- mean(e.coli_cds_length)
e.coli_median_cds_length <- median(e.coli_cds_length)

# Calculate mean and median of Saprospirales
saprospirales_mean_cds_length <- mean(saprospirales_cds_length)
saprospirales_median_cds_length <- median(saprospirales_cds_length)

# Create a table
cds_table <- data.frame(
  Bacteria = c("E.coli", "Saprospirales"),
```

```
  CDS_count = c(e.coli_number, saprospirales_num),
  total_coding_DNA = c(e.coli_total_cds_length, saprospirales_total_cds_length),
  mean_cds_length = c(e.coli_mean_cds_length, saprospirales_mean_cds_length),
  median_cds_length = c(e.coli_median_cds_length, saprospirales_median_cds_length)
)
cds_table
```

```
##        Bacteria CDS_count total_coding_DNA mean_cds_length median_cds_length
## 1        E.coli      4239          3978528        938.5534               831
## 2 Saprospirales      4527          4200321        927.8376               690
```

Step 4: Calculating the frequency of DNA bases and aminoacids in the total coding sequences for both organisms and generating barplots.

The unlist() function was used to turn the list of CDS into a single long vectors of single characters for both the organisms. Then the frequency of dna bases in total coding sequences was calculated using count() function and to obtain single nucleotide count, 1 was used. To obtain the bar plot, the table was plotted using data.frame for nucleotide frequency for both organisms followed by visualisation using the barplot with xlab="nucleotides", ylab="Count", and main="Nucleotide Frequency in CDS". The x-axis represented the nucleotides (A, C, G, T), the y-axis showed their frequencies, and the plot title indicated that it displays the nucleotide composition of organism's coding sequences.

The lapply which applies the translate function to each element of the cds list was used to transalte the dna to protein, resulting in specific organism's prot, a list iif translated protein seuences. Then, unlist() was used ti give a long aminoacid vector. Following that, the aminoacids frequency were counted using the count() function and table was generated for aa frequency of both organisms. Then, the barplot was generated for the aminoacid frequency.

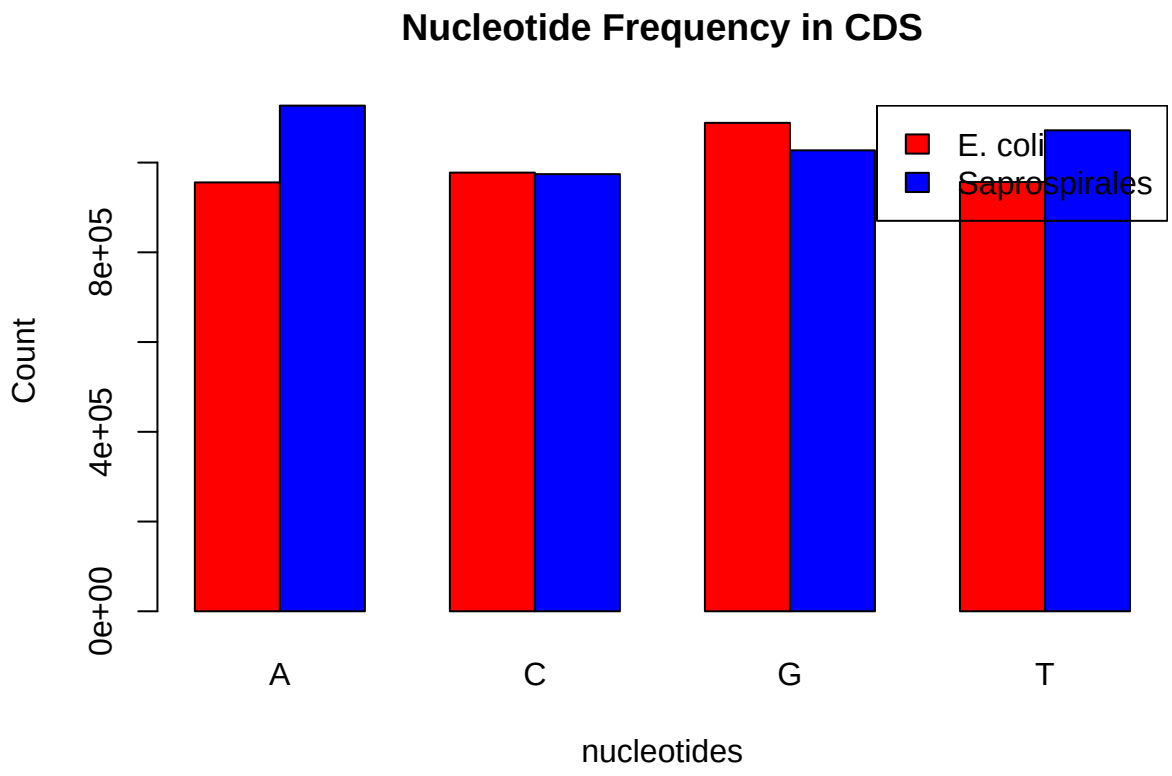
The higher frequency of adenine (A) was recorded higher in Saprospirales while E.coli recorded slightly higher frequency of guanine(G) content. Cytosine (C) and Thymine (T) levels were comparable between the two species.

In both the organisms, Leucine (L) and alanine (A) were the most abundant amino acids , followed by glycine (G), serine (S), and valine (V). While overall profiles were similar, E.coli exhibited higher frequencies of acid residues such as aspartic acid (D) and glutamic acid (E), along with slightly more histidine (H). In contrast, Saprospirlaes showed relatively higher frquencies of valine (V), glutamine (Q), and tryptophan(W)

```
# Unlist the E.coli cds
e.coli_dna <- unlist(e.coli_cds)

# Calculate frequency of dna bases in total coding sequences for e.coli
e.coli_dna_freq <- count(e.coli_dna, 1)   #1 =word size (single nucleotide count)

# Unlist the Saprosipirales cds sequences
saprospirales_dna <- unlist(saprospirales_cds)

# Calculate the frequency of dna bases in total coding sequences for Saprospirales
saprospirales_dna_freq <- count(saprospirales_dna, 1)   # 1 = word size (single nucleotide count)

# Combine into a data frame for plotting
Dna_freq_df <- data.frame(
  Base = c("A","C","G","T"),
  E.coli = e.coli_dna_freq,
  Saprospirales = saprospirales_dna_freq
)
Dna_freq_df
```

```
##   Base E.coli.Var1 E.coli.Freq Saprospirales.Var1 Saprospirales.Freq
## 1    A          a      955768                  a            1126928
## 2    C          c      977594                  c             974191
## 3    G          g     1088501                  g            1027218
## 4    T          t      956665                  t            1071885
```

```r
# Generate bar plot for nucleotide frequency
barplot(
  height = rbind(as.numeric(Dna_freq_df$E.coli.Freq), as.numeric(Dna_freq_df$Saprospirales.Freq)),
  beside = TRUE,      # Place the bar for E.coli and Saprospirales side-by-side
  names.arg = Dna_freq_df$Base, # Tells R to represent nucleotide bases (AGTC) at the x-axis
  col = c("red", "blue"), # Tells R to give red color to E.coli and blue to saprospirales
  main = "Nucleotide Frequency in CDS",
  ylab = "Count",
  xlab="nucleotides"
)
legend("topright", legend = c("E. coli", "Saprospirales"), fill = c("red", "blue"))
```




```r
# topright specify the position of the legend, legend =c("E. coli", "Saprospirales") are the labels of

# Translate the sequences for e.coli
e.coli_prot <-lapply(e.coli_cds, translate) # lapply applies the translate function to each element of

# Unlist aminoacids of E.coli
e.coli_prot_unlist <- unlist(e.coli_prot)

# Count aminoacids
```

```r
aa_alphabet <- c("A","R","N","D","C","Q","E","G","H","I","L","K","M","F","P","S","T","W","Y","V")

# Calculate E.coli aminoacid frequency
e.coli_aa_freq <- count(e.coli_prot_unlist, wordsize=1, alphabet=aa_alphabet)
e.coli_aa_freq
```

```
##
##      A      C      D      E      F      G      H      I      K      L      M
## 126127  15376  67796  76338  51561  97246  29995  79511  58113 141731  37007
##      N      P      Q      R      S      T      V      W      Y
##  51503  58700  58799  73111  76412  71025  93989  20196  37401
```

```r
# Translate the sequences for Saprospirales
saprospirales_prot <- lapply(saprospirales_cds, translate)

# Unlist aminoacids of Saprospirales
saprospirales_prot_unlist <- unlist(saprospirales_prot)

# Calculate the frequency of aa of Saprospirales
Saprospirales_aa_freq <- count(saprospirales_prot_unlist, wordsize=1, alphabet=aa_alphabet)
Saprospirales_aa_freq
```

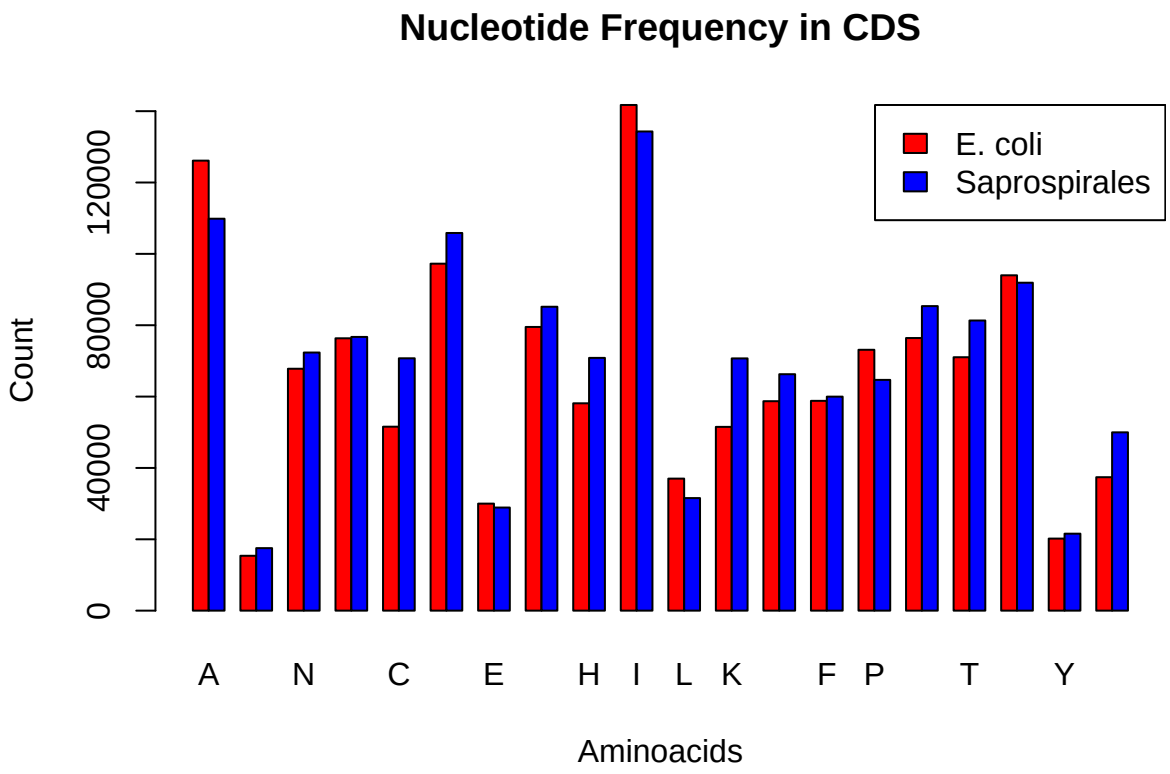
```
##
##      A      C      D      E      F      G      H      I      K      L      M
## 109849  17521  72341  76718  70730 105855  28895  85178  70845 134307  31549
##      N      P      Q      R      S      T      V      W      Y
##  70696  66268  59978  64676  85359  81320  91915  21572  49970
```

```r
# combine the aa frequency into data.frame for plotting
aa_freq_df <- data.frame(
  AA = aa_alphabet,
  E_coli = e.coli_aa_freq,
  Saprospirales = Saprospirales_aa_freq
)
aa_freq_df
```

```
##    AA E_coli.Var1 E_coli.Freq Saprospirales.Var1 Saprospirales.Freq
## 1   A           A      126127                  A             109849
## 2   R           C       15376                  C              17521
## 3   N           D       67796                  D              72341
## 4   D           E       76338                  E              76718
## 5   C           F       51561                  F              70730
## 6   Q           G       97246                  G             105855
## 7   E           H       29995                  H              28895
## 8   G           I       79511                  I              85178
## 9   H           K       58113                  K              70845
## 10  I           L      141731                  L             134307
## 11  L           M       37007                  M              31549
## 12  K           N       51503                  N              70696
## 13  M           P       58700                  P              66268
## 14  F           Q       58799                  Q              59978
## 15  P           R       73111                  R              64676
## 16  S           S       76412                  S              85359
## 17  T           T       71025                  T              81320
## 18  W           V       93989                  V              91915
```

```
## 19  Y           W       20196                   W               21572
## 20  V           Y       37401                   Y               49970
```

```r
# Generate bar plot for nucleotide frequency
barplot(
  height = rbind(as.numeric(aa_freq_df$E_coli.Freq), as.numeric(aa_freq_df$Saprospirales.Freq)),
  beside = TRUE,
  names.arg = aa_freq_df$AA, # Tells R to represent aa texts at the x-axis
  col = c("red", "blue"),
  main = "Nucleotide Frequency in CDS",
  ylab = "Count",
  xlab = "Aminoacids"
)
legend("topright", legend = c("E. coli", "Saprospirales"), fill = c("red", "blue"))
```



Create a codon usage table and quantify the codon usage bias among all coding sequences. Describe any differences between the two organisms with respect to their codon usage bias. Provide charts to support your observations.

The uco() function from the sequinr package was applied to count all codons (3-base sequences) in the DNA sequences. The codons were sorted using order() function to ensure that the codon table is consistent for both organisms. Following that, the table was created to compare the codon counts between the organisms, where each row is a codon with counts in E.coli and Saprospirales. Then, the index="rscu" was employed to compute relative synonymous codon usage which is a measure of codon usage bias. Here, the RSCU > 1 indicates that codon is used more frequently than expected for the amino acid while RSCU <1 indicates that codon is used less frequently than expected. The data.frame was kept TRUE to convert the result into plotting.

The barchart was generated for codon usage bias using rbind() function which makes matrix with rows =

organisms and columns = codons. Then beside=TRUE was used to keep the charts of two organisms side by side for comparison and legend was added to the chart.

As shown in the chart, the E.coli shows stronger peaks for certain codons, indicating higher codon usage bias while Saprospirales exhibit more even distributions across synonymous codons, suggesting weaker codon preference. Moreover, some of the codons which are less used in saprospirales and simlarly, the codons which are are more used in Saprospirales are comparatively less used in E.coli. This shows that the two organisms are adapeted to different tRNA pools and translational pressures.

```
# Determining codon usage for e.coli
e.coli_codon_usage <- uco(e.coli_dna)
e.coli_codon_usage <- e.coli_codon_usage[order(names(e.coli_codon_usage))]  # sort codons
e.coli_codon_usage
```

```
##
##   aaa   aac   aag   aat   aca   acc   acg   act   aga   agc   agg   agt   ata
## 44592 28454 13521 23049  9116 31139 19081 11689  2573 21291  1420 11487  5486
##   atc   atg   att   caa   cac   cag   cat   cca   ccc   ccg   cct   cga   cgc
## 33524 37007 40501 20402 12890 38397 17105 11163  7238 31074  9225  4619 29441
##   cgg   cgt   cta   ctc   ctg   ctt   gaa   gac   gag   gat   gca   gcc   gcg
##  7079 27979  5149 14811 70714 14586 52679 25347 23659 42449 26743 34117 45082
##   gct   gga   ggc   ggg   ggt   gta   gtc   gtg   gtt   taa   tac   tag   tat
## 20185 10350 39536 14581 32779 14430 20350 34996 24213  2726 16160   294 21241
##   tca   tcc   tcg   tct   tga   tgc   tgg   tgt   tta   ttc   ttg   ttt
##  9303 11390 11830 11111  1219  8574 20196  6802 18323 21974 18148 29587
```

```
# Determining codon usage for saprospirales
saprospirales_codon_usage <- uco(saprospirales_dna) # uco() returns counts of each codon.
saprospirales_codon_usage <- saprospirales_codon_usage[order(names(saprospirales_codon_usage))]  # sort
saprospirales_codon_usage
```

```
##
##   aaa   aac   aag   aat   aca   acc   acg   act   aga   agc   agg   agt   ata
## 50608 31412 20237 39284 15761 37355 14412 13792  7157 17447  6970 14950 14597
##   atc   atg   att   caa   cac   cag   cat   cca   ccc   ccg   cct   cga   cgc
## 30418 31549 40163 29446 12712 30532 16183 15692 14394 19625 16557  6955 17264
##   cgg   cgt   cta   ctc   ctg   ctt   gaa   gac   gag   gat   gca   gcc   gcg
## 13395 12935  6028 16073 44826 21204 53433 24482 23285 47859 27044 41438 15933
##   gct   gga   ggc   ggg   ggt   gta   gtc   gtg   gtt   taa   tac   tag   tat
## 25434 24584 39253 14547 27471 27882  9392 29602 25039  2164 21399   952 28571
##   tca   tcc   tcg   tct   tga   tgc   tgg   tgt   tta   ttc   ttg   ttt
## 10542 20505  8442 13473  1411  9055 21572  8466 14950 27365 31226 43365
```

```
# Creating table for codon_usage for E.coli and Saprospirales
codons <- names(e.coli_codon_usage)
bacteria_codon_table <- data.frame(
  Codon = codons,
  E.coli_count = as.numeric(e.coli_codon_usage),
  Saprospirales_count = as.numeric(saprospirales_codon_usage)
  )
bacteria_codon_table
```

```
##    Codon E.coli_count Saprospirales_count
## 1    aaa        44592               50608
## 2    aac        28454               31412
## 3    aag        13521               20237
## 4    aat        23049               39284
```

```
## 5    aca         9116              15761
## 6    acc        31139              37355
## 7    acg        19081              14412
## 8    act        11689              13792
## 9    aga         2573               7157
## 10   agc        21291              17447
## 11   agg         1420               6970
## 12   agt        11487              14950
## 13   ata         5486              14597
## 14   atc        33524              30418
## 15   atg        37007              31549
## 16   att        40501              40163
## 17   caa        20402              29446
## 18   cac        12890              12712
## 19   cag        38397              30532
## 20   cat        17105              16183
## 21   cca        11163              15692
## 22   ccc         7238              14394
## 23   ccg        31074              19625
## 24   cct         9225              16557
## 25   cga         4619               6955
## 26   cgc        29441              17264
## 27   cgg         7079              13395
## 28   cgt        27979              12935
## 29   cta         5149               6028
## 30   ctc        14811              16073
## 31   ctg        70714              44826
## 32   ctt        14586              21204
## 33   gaa        52679              53433
## 34   gac        25347              24482
## 35   gag        23659              23285
## 36   gat        42449              47859
## 37   gca        26743              27044
## 38   gcc        34117              41438
## 39   gcg        45082              15933
## 40   gct        20185              25434
## 41   gga        10350              24584
## 42   ggc        39536              39253
## 43   ggg        14581              14547
## 44   ggt        32779              27471
## 45   gta        14430              27882
## 46   gtc        20350               9392
## 47   gtg        34996              29602
## 48   gtt        24213              25039
## 49   taa         2726               2164
## 50   tac        16160              21399
## 51   tag          294                952
## 52   tat        21241              28571
## 53   tca         9303              10542
## 54   tcc        11390              20505
## 55   tcg        11830               8442
## 56   tct        11111              13473
## 57   tga         1219               1411
## 58   tgc         8574               9055
```

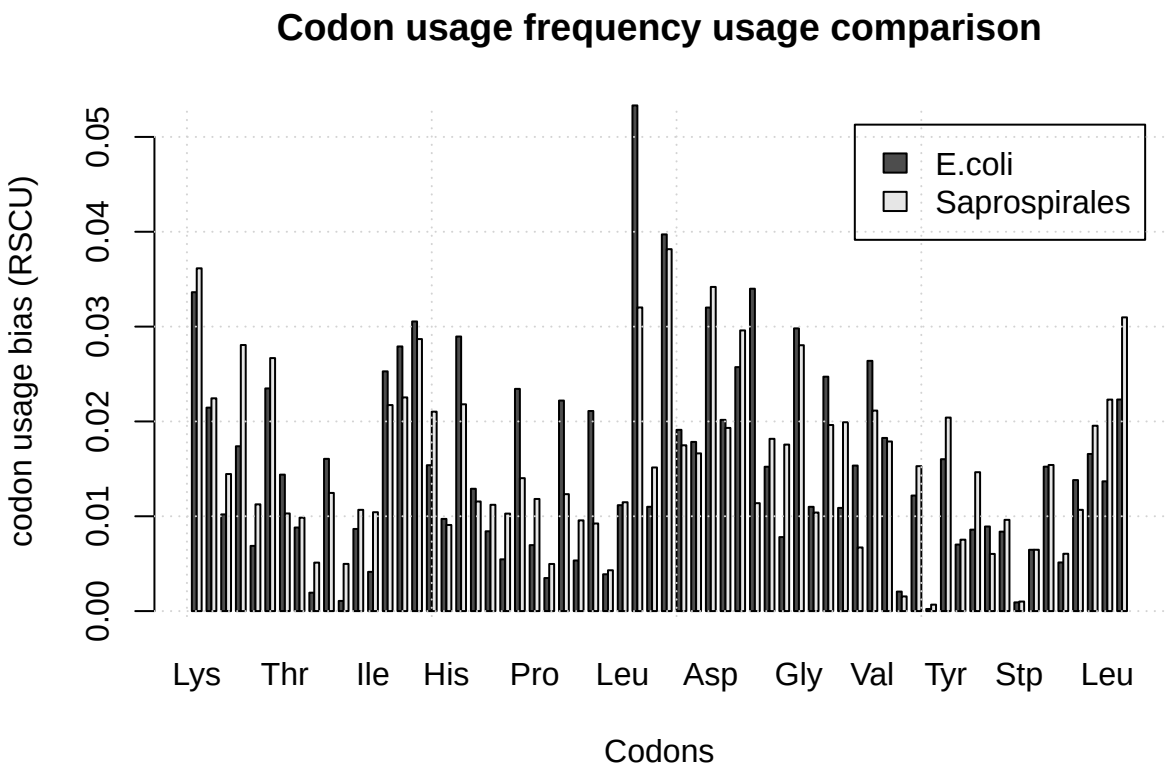
```
## 59   tgg        20196               21572
## 60   tgt         6802                8466
## 61   tta        18323               14950
## 62   ttc        21974               27365
## 63   ttg        18148               31226
## 64   ttt        29587               43365
```

```
# Calculation of RSCU values for E.coli
e.coli_codon_usage_bias <- uco(e.coli_dna, index="rscu", as.data.frame=TRUE)

# Calculation of RSCU values for Saprospirales
saprospirales_codon_usage_bias <- uco(saprospirales_dna, index="rscu", as.data.frame=TRUE)
e.coli_codon_usage_bias
```

```
##      AA codon   eff         freq      RSCU
## aaa Lys   aaa 44592 0.0336244963 1.5346652
## aac Asn   aac 28454 0.0214556741 1.1049453
## aag Lys   aag 13521 0.0101954793 0.4653348
## aat Asn   aat 23049 0.0173800461 0.8950547
## aca Thr   aca  9116 0.0068738991 0.5133967
## acc Thr   acc 31139 0.0234802922 1.7536924
## acg Thr   acg 19081 0.0143879847 1.0746075
## act Thr   act 11689 0.0088140639 0.6583034
## aga Arg   aga  2573 0.0019401648 0.2111584
## agc Ser   agc 21291 0.0160544302 1.6718055
## agg Arg   agg  1420 0.0010707478 0.1165351
## agt Ser   agt 11487 0.0086617463 0.9019787
## ata Ile   ata  5486 0.0041367058 0.2069902
## atc Ile   atc 33524 0.0252786960 1.2648816
## atg Met   atg 37007 0.0279050443 1.0000000
## att Ile   att 40501 0.0305396870 1.5281282
## caa Gln   caa 20402 0.0153840818 0.6939574
## cac His   cac 12890 0.0097196752 0.8594766
## cag Gln   cag 38397 0.0289531706 1.3060426
## cat His   cat 17105 0.0128979864 1.1405234
## cca Pro   cca 11163 0.0084174348 0.7606814
## ccc Pro   ccc  7238 0.0054577975 0.4932198
## ccg Pro   ccg 31074 0.0234312791 2.1174787
## cct Pro   cct  9225 0.0069560903 0.6286201
## cga Arg   cga  4619 0.0034829465 0.3790674
## cgc Arg   cgc 29441 0.0221999192 2.4161344
## cgg Arg   cgg  7079 0.0053379039 0.5809523
## cgt Arg   cgt 27979 0.0210975014 2.2961524
## cta Leu   cta  5149 0.0038825918 0.2179763
## ctc Leu   ctc 14811 0.0111682009 0.6270047
## ctg Leu   ctg 70714 0.0533217311 2.9935864
## ctt Leu   ctt 14586 0.0109985402 0.6174796
## gaa Glu   gaa 52679 0.0397224803 1.3801514
## gac Asp   gac 25347 0.0191128478 0.7477432
## gag Glu   gag 23659 0.0178400152 0.6198486
## gat Asp   gat 42449 0.0320085720 1.2522568
## gca Ala   gca 26743 0.0201654984 0.8481293
## gcc Ala   gcc 34117 0.0257258463 1.0819888
## gcg Ala   gcg 45082 0.0339939797 1.4297335
## gct Ala   gct 20185 0.0152204534 0.6401484
```

```
## gga Gly   gga 10350 0.0078043940 0.4257245
## ggc Gly   ggc 39536 0.0298120310 1.6262263
## ggg Gly   ggg 14581 0.0109947699 0.5997573
## ggt Gly   ggt 32779 0.0247169305 1.3482920
## gta Val   gta 14430 0.0108809087 0.6141144
## gtc Val   gtc 20350 0.0153448713 0.8660588
## gtg Val   gtg 34996 0.0263886543 1.4893658
## gtt Val   gtt 24213 0.0182577576 1.0304610
## taa Stp   taa  2726 0.0020555341 1.9292286
## tac Tyr   tac 16160 0.0121854113 0.8641480
## tag Stp   tag   294 0.0002216900 0.2080679
## tat Tyr   tat 21241 0.0160167278 1.1358520
## tca Ser   tca  9303 0.0070149060 0.7304874
## tcc Ser   tcc 11390 0.0085886036 0.8943621
## tcg Ser   tcg 11830 0.0089203846 0.9289117
## tct Ser   tct 11111 0.0083782243 0.8724546
## tga Stp   tga  1219 0.0009191842 0.8627035
## tgc Cys   tgc  8574 0.0064652052 1.1152445
## tgg Trp   tgg 20196 0.0152287479 1.0000000
## tgt Cys   tgt  6802 0.0051290326 0.8847555
## tta Leu   tta 18323 0.0138164165 0.7756807
## ttc Phe   ttc 21974 0.0165694448 0.8523496
## ttg Leu   ttg 18148 0.0136844582 0.7682723
## ttt Phe   ttt 29587 0.0223100101 1.1476504
```

```r
# generate barchart for codon usage bias for e.coli and saprospirales
RCSU_matrix <-rbind(E.coli = e.coli_codon_usage_bias$freq, Saprospirales = saprospirales_codon_usage_bia
barplot(RCSU_matrix,
        beside = TRUE,
        names.arg = e.coli_codon_usage_bias$AA,
        legend.text = TRUE,
        ylab = "codon usage bias (RSCU)",
        xlab = "Codons",
        main = "Codon usage frequency usage comparison"
        )
grid()
```



Step 6: In the organism of interest, identify 10 protein sequence k-mers of length 3-5 which are the most over- and under-represented k-mers in your organism of interest. Are these k-mers also over- and under-represented in E. coli to a similar extent? Provide plots to support your observations. Why do you think these sequences are present at different levels in the genomes of these organisms?

```
# Translate the sequences for e.coli
e.coli_prot <-lapply(e.coli_cds, translate) # lapply applies the translate function to each element of

# Unlist aminoacids of E.coli
e.coli_prot_unlist <- unlist(e.coli_prot)

# Calculate 3-mer (amino acid triplet) frequency in E. coli protein sequences
e.coli_prot_3_count <- count(e.coli_prot_unlist, wordsize = 3, alphabet = aa_alphabet)

e.coli_prot_4_count <- count(e.coli_prot_unlist, wordsize = 4, alphabet = aa_alphabet)

e.coli_prot_5_count <-  count(e.coli_prot_unlist, wordsize = 5, alphabet = aa_alphabet)


# Translate the sequences for Saprospirales
saprospirales_prot <- lapply(saprospirales_cds, translate)

# Unlist aminoacids of Saprospirales
saprospirales_prot_unlist <- unlist(saprospirales_prot)

# Calculate 3-mer (amino acid triplet) frequency in Saprospirales protein sequences
Saprospirales_prot_3_count <- count(saprospirales_prot_unlist, wordsize = 3, alphabet = aa_alphabet)
```

```
Saprospirales_prot_4_count <- count(saprospirales_prot_unlist, wordsize = 4, alphabet = aa_alphabet)
Saprospirales_prot_5_count <- count(saprospirales_prot_unlist, wordsize = 5, alphabet = aa_alphabet)
```