Assessment 4_SLE777_R Project

Ameeta

2025-10-05

Gene expression Step 1: Downloading and reading the geneexpression file

The link for raw file were copied from the github and then downloaded in R-markdown using te download.file function The downloaded file was read using the read.table function and the head function was used display the first 6 rows, showing 6 gene identifiers.

```
# download the gene expression file

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

# 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

A new column called meanexpression, which contain the mean value of other columns was created in the table using rowMeans function and the first six rows were displayed.

```
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

The meanexpression in gene_data were ordered in the descending order using order (-)function. 10 genes with highest meanexpression were displayed using a drop argument.

```
# 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, the logical condition was created to line checks for every gene in the datasetiwith meanexpression value less than 10 and the sum() function was used to sum the logical vectors.

```
# 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
sum(gene_data_mean_10)
```

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

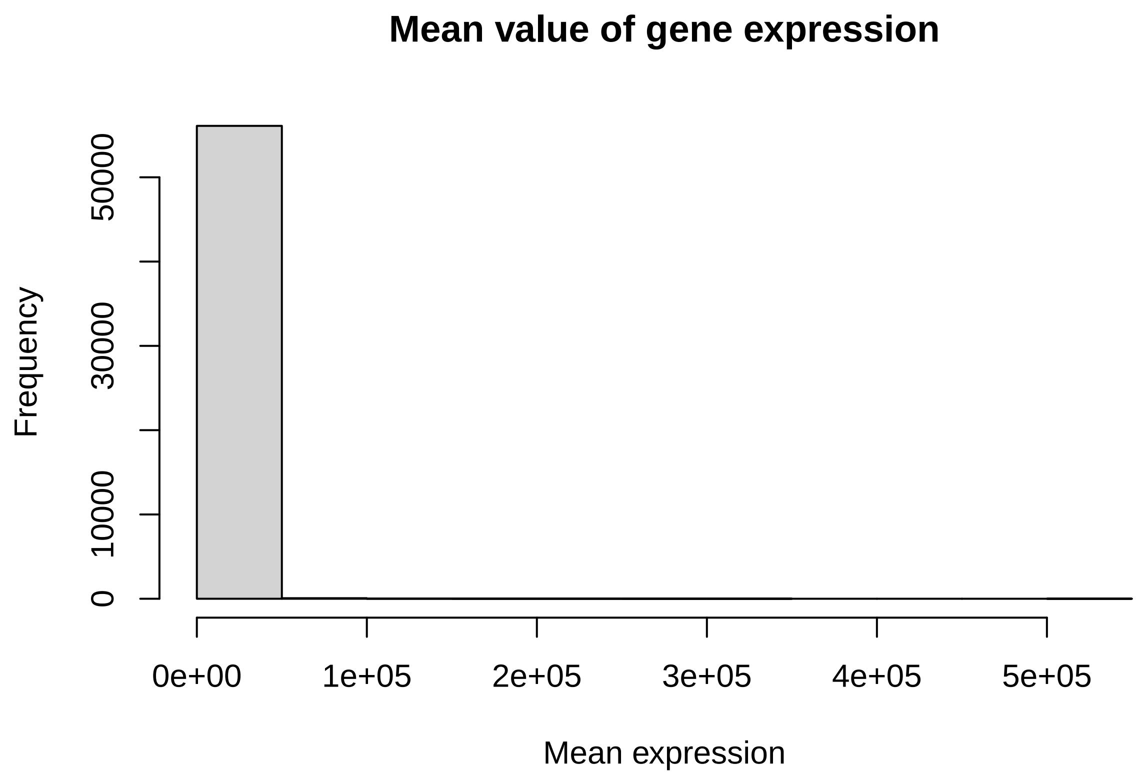
Step 5: Making a histogram plot of the mean values

A histogram was generated using hist() function to represent the distribution of the mean expression value of the genes.
The 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.

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

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

```
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 the file was imported into R as a data frame using the read.csv() function. The colnames() function was used to display all the column names of the data set.

```
# download the growth data file

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

# 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.

The mean and standard deviation was created using the mean() and sd() function. The data.frame was created to display the values

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

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

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

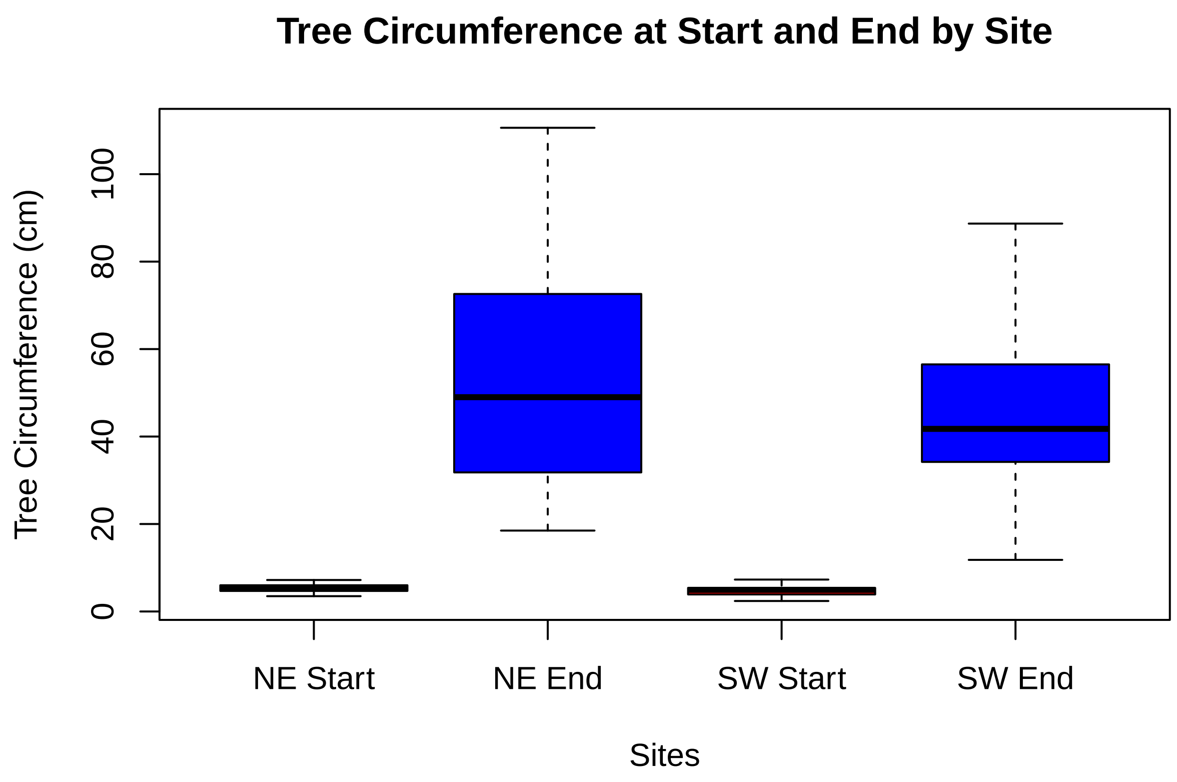
```r
# 2. Mean and sd for Northeast site at the start (Circumf_2020_cm)

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

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

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

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

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


# 4. Mean and sd for Southwest site at the end (Circumf_2020_cm)

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

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

# create summary table

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)
)
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.

To create Boxplots, 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.The boxplots were placed side by side by entering data sets for boths sites in the same boxplot() function. For both northeast and southwest, the tree circumference increased substantially from 2005 to 2020, indicating significant tree growth over period of time.

```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)",
        xlab = "Sites",
        main = "Tree Circumference at Start and End by Site")
```



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

The 10 year growth values were extracted by finding difference in the circumference between 2020 and 2010 and saved in the new column. The data were separated by site to calculate the mean growth for each location and the mean() function was used to find the mean value.

```
# calculate growth data from 2010 to 2020

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

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

# calculate 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
```

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

# 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

The p-value for the two sites were determined using t-test function to compare the 10 years growth between two sites.The study observed higher mean 10-year growth at the northeast site compared to soutwest site but the growth difference was not statistically significant at 5% significance level

```
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 and counting CDS in the Saprospirale and ecoli sequences

The FASTA files were downloaded from the ENSEMBL website using the download.file() function and gunzip() function was used to uncompress the file. The read.fasta() function loaded the sequence into R as lists and the length() function was applied to determine the total number of CDS for each organism. The table was created using data.frame() function The Saprospirales contain slightly higher number of coding sequences (4527) than E.coli (4239).

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

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 and saprospirales

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

gunzip("saprospirales_cds.fa.gz", overwrite  = TRUE)

# Read the FASTA sequences
library("seqinr")
```

```
e.coli_cds <- seqinr::read.fasta("ecoli_cds.fa") # read.fasta is used to read the FASTA file

saprospirales_cds <- seqinr::read.fasta("saprospirales_cds.fa")

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

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.

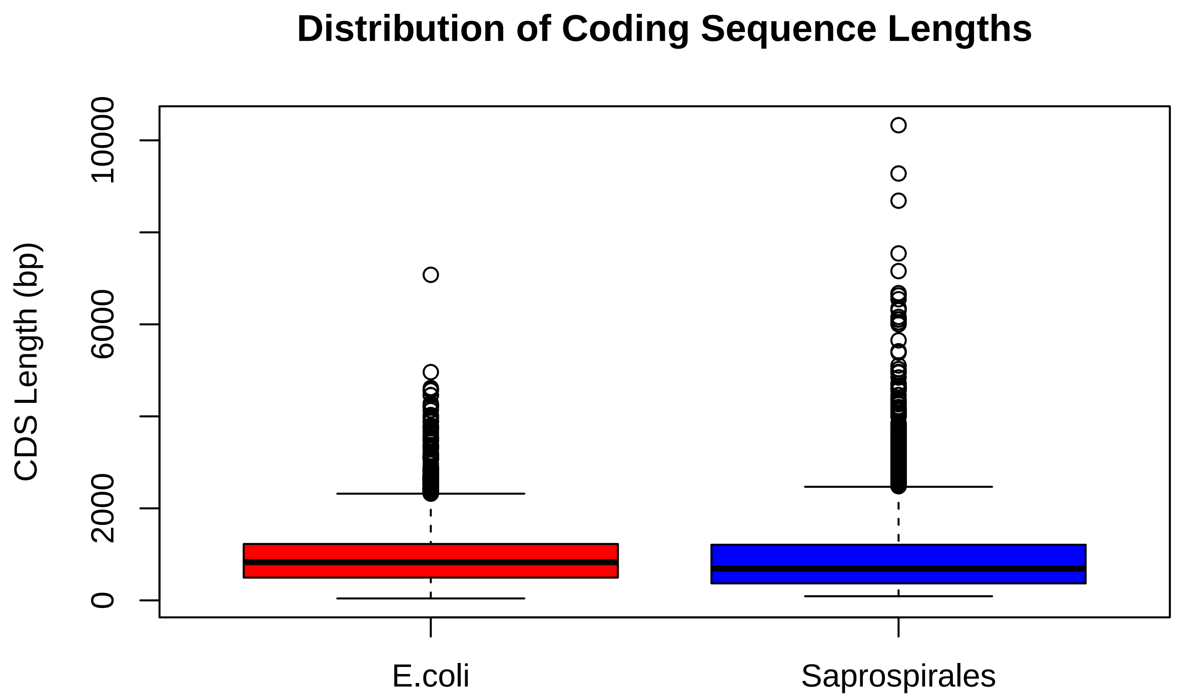
```
# Calculate the CDS length for e.coli
e.coli_cds_length<- as.numeric(summary(e.coli_cds)[,1]) # 1 select the first column of the matrix

# 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: Calculating and making boxplot of CDS length

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 cds lengths. The mean() and median() function was used to summarize the central tendency of CDS lengths and table was created to display the result. The mean coding sequence length of E.coli is 938.5534 and median is 831, indicating it contains higher cds length than with moderately long E.coli genes compared to Saprospirales with mean value of 927.8376 and the median is 690.

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

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

The unlist() function was used to turn the list of CDS into a single long vectors of single characters and count() function was used to calculate the frequency. The table was created for bar plotting followed by visualisation using the barplot using barplot().

The lapply was used to translate the dna to protein and unlist() was used to 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.

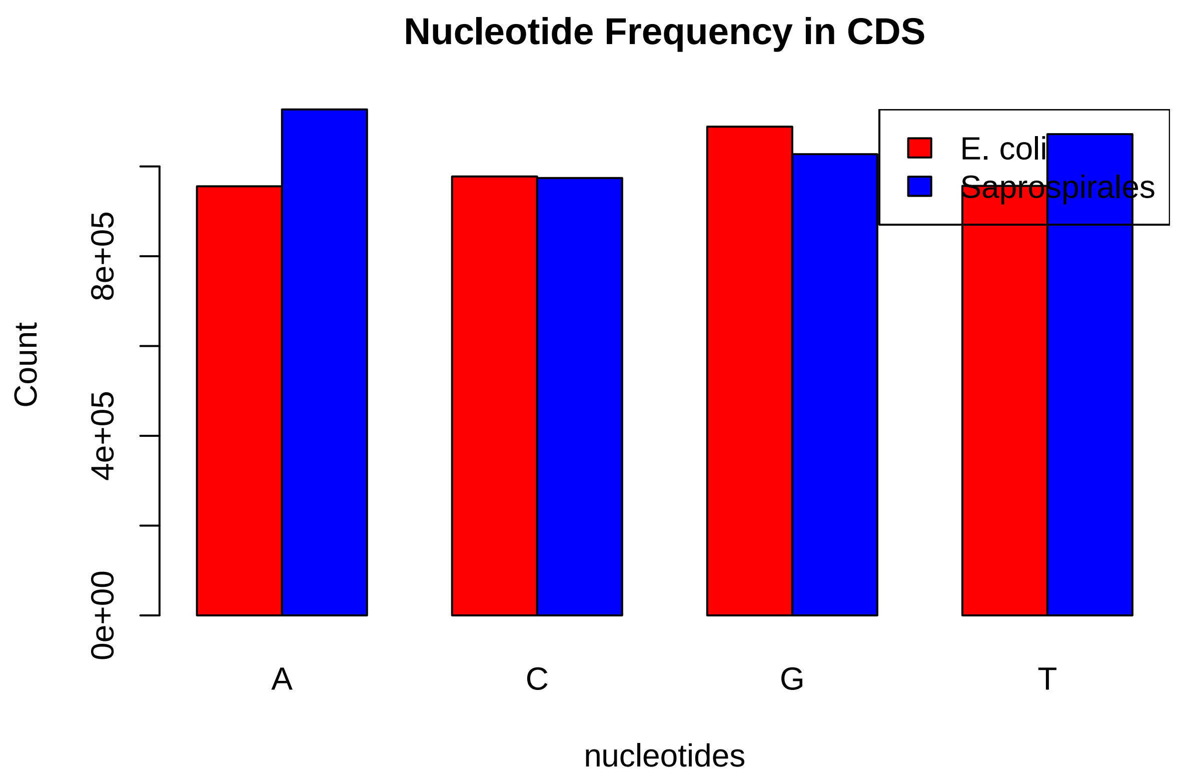
The higher frequency of adenine (A) was recorded higher in Saprospirales while E.coli recorded slightly higher frequency of guanine(G) content. Cytosine (C) and Thymine (T) levels were comparable between the two species.

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

```r
# 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
```

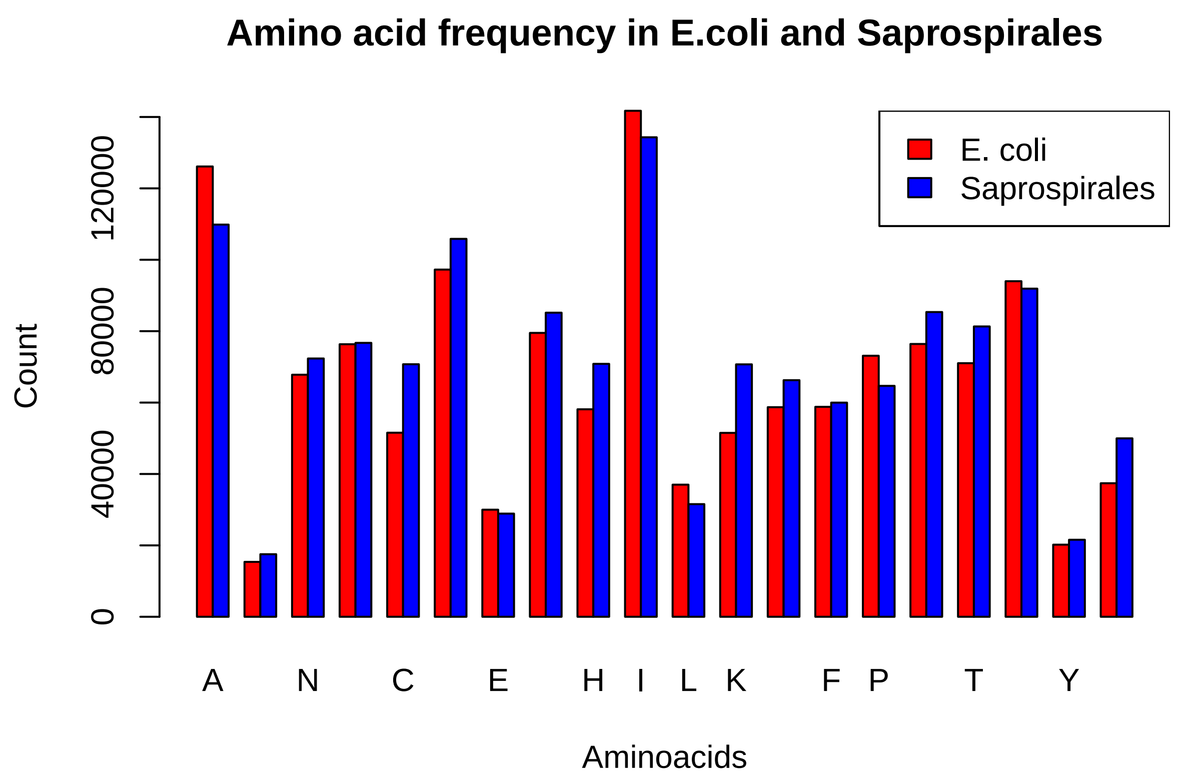
```r
# Translate the sequences for Saprospirales
saprospirales_prot <- lapply(saprospirales_cds, translate)

# 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 amino acid 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 = "Amino acid frequency in E.coli and Saprospirales",
  ylab = "Count",
  xlab = "Aminoacids"
)
legend("topright", legend = c("E. coli", "Saprospirales"), fill = c("red", "blue"))
```



Step 5: Qunaitifying the codon usage bias among all coding sequences.

The uco() function 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 and table was created. 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.

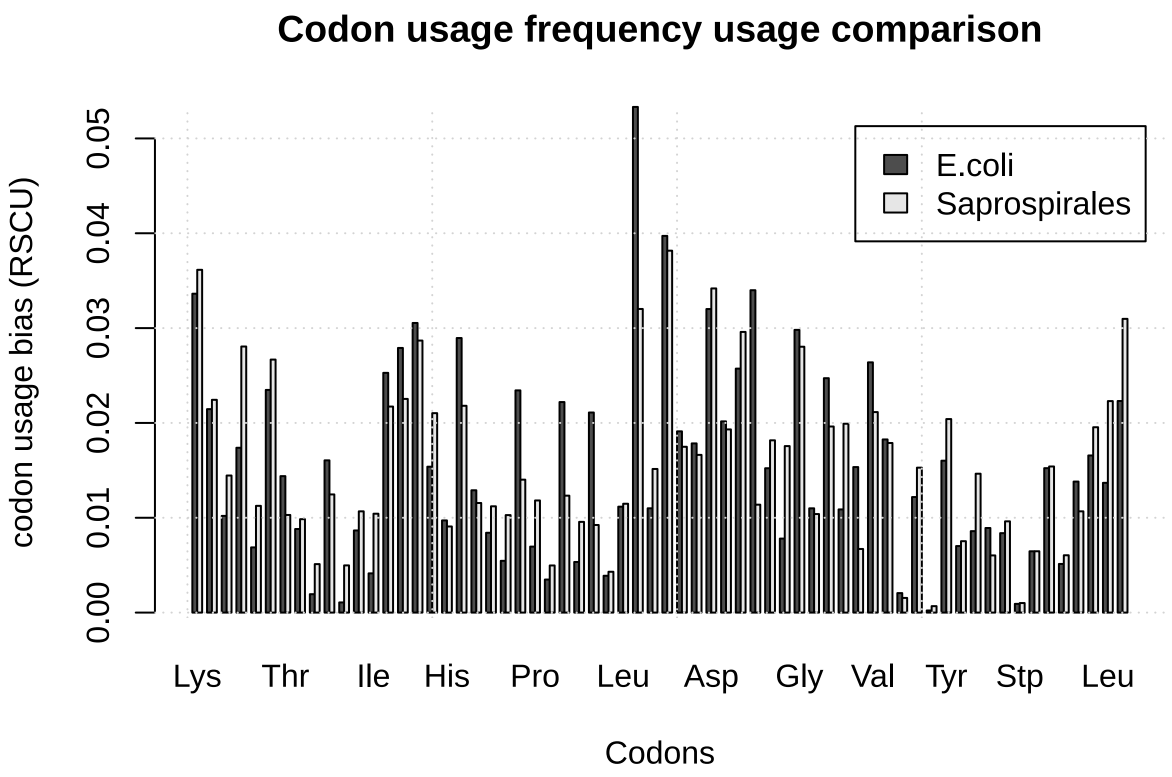
The barchart was generated for codon usage bias using rbind() function.

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.

```r
# 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

# 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

# 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)
  )

# 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: Identifying over- and under- expressed k-mers of length 3-5

The k-mer frequencies of 3- 4- and 5- were calculated using count() function and the custom top k-markers function which sort the k-mer by frequency was used to extract the top 10 most over- and under- expressed k-mers in both organisms. Then, the common over- and under-expressed k-merrss were check using the intersect function and the unique k-mers were checked using setdiff() function. To check if the the top over- and under-pressed k-mers in saprospirales are also expressed to similar extent in E.coli, the raw k-mer counts from E.coli was taken, using 0 if the k-mer was absent. This included the frequency even if the k-mer was not over- or under-represented in E.coli. For each set of k-mers, the matrix were created and saprospirales count were taken from top-kmers() results. The barplot() function was used to create side-by-side bars for each k-mer. Using this function, separate bar plots were created for 3-,4-, and 5-ners, and for over- and under-expressed k-mers for visual idnetification and comparison with that of E.coli.

When comparing the top over- and under-expressed k-mers in Saprospirales with E.coli, most k-mers were not represented to the same extent in E.coli. Some K-mers that are highly frequent in Saprspirales are rare in E.coli, an dvice versa. This can be visually confirmed from the side-by-side barplots, where the heights of the bars of E.coli are often lower or higher than those for saprospirales. This differences could be attributed to difference in codon usage as different organisms prefer certain codons, which affect aminoacid triplet frequencies in proteins. The variation in GC content or overall amino acid composition can bias the presence of specific k-mers and the certain k-mers may be high in one organism due to other selective pressures such proteing function and envuronmental adaptations.

```
# Calculate K-mer 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)
```

```r
# Calculate K-mer 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)

# create Function to get top N over- and under-represented k-mers
top_kmers <- function(kmer_counts, N = 10) {
  # Sort k-mers by frequency
  kmer_sorted <- sort(kmer_counts)
    # Get N least frequent (under-represented) k-mers
  under <- head(kmer_sorted[kmer_sorted > 0], N)

  # Get N most frequent (over-represented) k-mers
  over  <- tail(kmer_sorted, N)

  list(over = over, under = under)
}

# CalculatE the top over- and under-expressed k-mers
Saprospirales_3_mer<- top_kmers(Saprospirales_prot_3_count, N = 10)
Saprospirales_3_mer
```

```
## $over
##
##  AAL  LLS  GLL  LGL  ALA  LAA  LAL  ALL  LLA  LLL
##  925  951  963  964  976  993 1065 1127 1180 1659
##
## $under
##
## CMC CCM CWC WCQ CWM MCW MWC WCC WCY CCW
##   1   2   2   2   3   3   3   3   3   4
```

```r
Saprospirales_4_mer <- top_kmers(Saprospirales_prot_4_count, N = 10)
Saprospirales_4_mer
```

```
## $over
##
## LLAL FLLL SLLL GLLL ALLL LLLS GGGG PNPA LLLA LLLL
##  121  123  124  128  131  133  136  161  176  241
##
## $under
##
## AAYM ACAH ACAM ACCE ACCG ACCK ACCV ACCW ACDC ACEF
##    1    1    1    1    1    1    1    1    1    1
```

```r
Saprospirales_5_mer <- top_kmers(Saprospirales_prot_5_count, N = 10)
Saprospirales_5_mer
```

```
## $over
##
## FPNPA GGGGG PNPAS TVTVT TYTVT YPNPA LLLLL VTVTD YTVTV GTYTV
##    38    38    38    40    40    43    44    47    47    50
##
## $under
##
```

```
## AAAAC AAAAD AAACC AAACD AAACI AAACK AAACL AAACM AAACQ AAACY
##     1     1     1     1     1     1     1     1     1     1
```

```r
e.coli_3_mer<- top_kmers(e.coli_prot_3_count, N = 10)
e.coli_3_mer
```

```
## $over
##
##  TLL  LAG  AAA  LAL  AAL  LLL  LAA  ALA  ALL  LLA
## 1102 1178 1338 1536 1594 1594 1601 1740 1744 1817
##
## $under
##
## CMY MWC WMC CMC CMW WCM WWC CHW CWW MCM
##   1   1   1   2   2   2   2   3   3   3
```

```r
e.coli_4_mer <- top_kmers(e.coli_prot_4_count, N = 10)
e.coli_4_mer
```

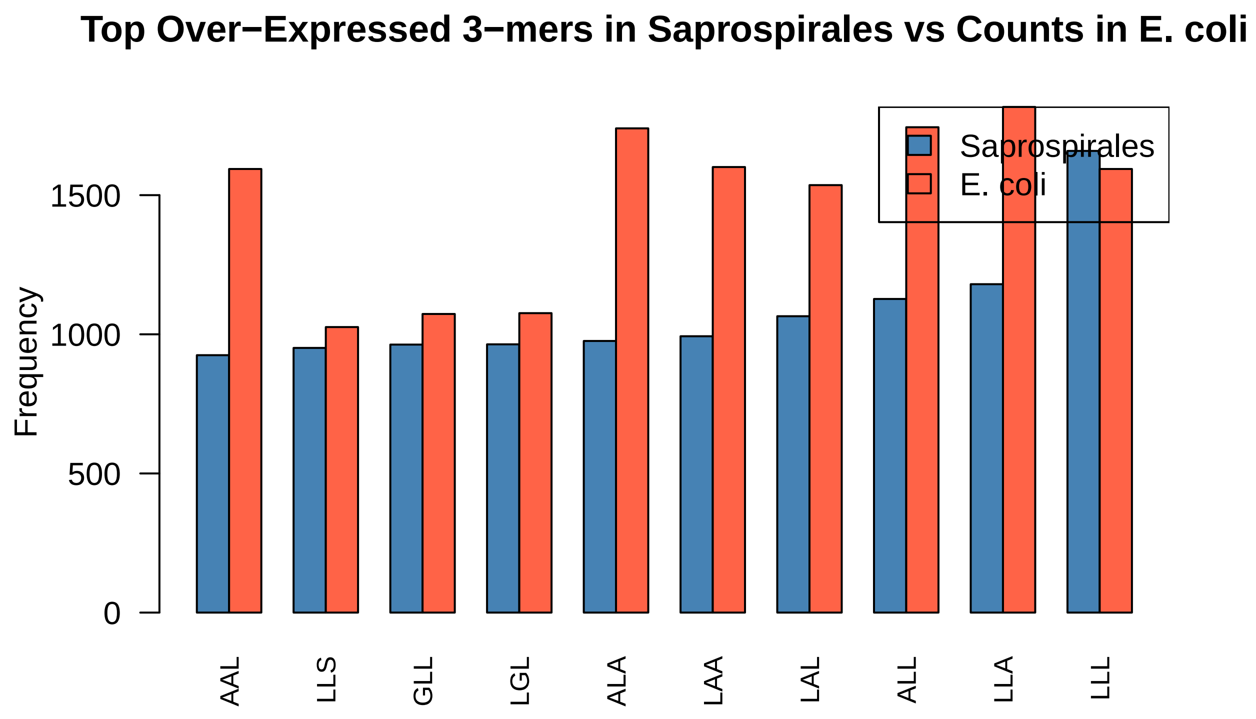
```
## $over
##
## ALLA LALL LALA LLLA LLAA AALA LLLL LLAL ALAA LAAL
##  193  195  196  197  206  207  209  215  233  234
##
## $under
##
## AACW AAMW AAYW ACAW ACCC ACCD ACCN ACCP ACCR ACCY
##    1    1    1    1    1    1    1    1    1    1
```

```r
e.coli_5_mer <- top_kmers(e.coli_prot_5_count, N = 10)
e.coli_5_mer
```

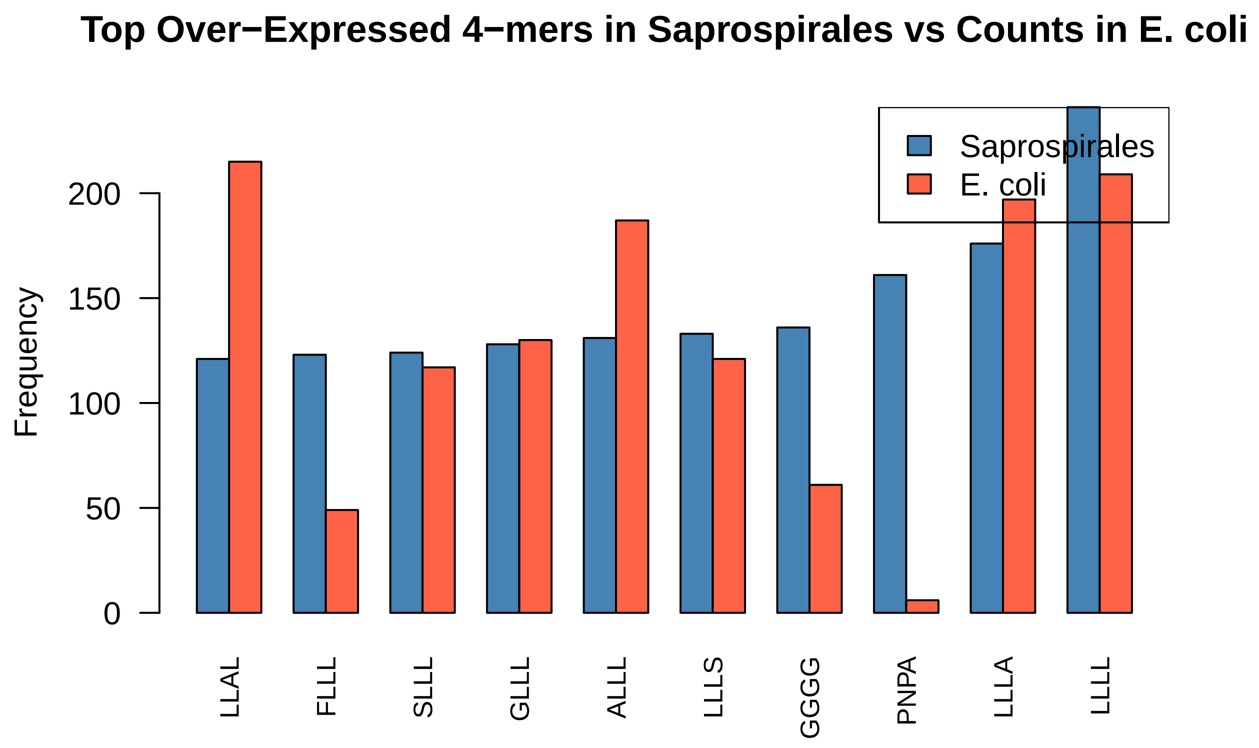
```
## $over
##
## LDEPT LAAAL LLLAL LLLLL LLLDE AALAA LLAAL SGSGK GSGKS GKSTL
##    33    34    34    34    35    37    37    37    42    58
##
## $under
##
## AAAAH AAAAY AAACD AAACF AAACQ AAACR AAACV AAACY AAADQ AAAEC
##     1     1     1     1     1     1     1     1     1     1
```

```r
# Common over-expressed k-mers in both organisms

common_over_3 <- intersect(names(Saprospirales_3_mer$over), names(e.coli_3_mer$over))

common_over_4 <- intersect(names(Saprospirales_4_mer$over), names(e.coli_4_mer$over))

common_over_5 <- intersect(names(Saprospirales_5_mer$over), names(e.coli_5_mer$over))

# Common under-expressed k-mers in both organisms
common_under_3 <- intersect(names(Saprospirales_3_mer$under), names(e.coli_3_mer$under))
common_under_4 <- intersect(names(Saprospirales_4_mer$under), names(e.coli_4_mer$under))
common_under_5 <- intersect(names(Saprospirales_5_mer$under), names(e.coli_5_mer$under))

# over-expressed k-mers in Saprospirales but not over expressed in E.coli
unique_over_3 <- setdiff(names(Saprospirales_3_mer$over), names(e.coli_3_mer$over))
```

```r
unique_over_4 <- setdiff(names(Saprospirales_4_mer$over), names(e.coli_4_mer$over))

unique_over_5 <- setdiff(names(Saprospirales_5_mer$over), names(e.coli_5_mer$over))

# for overexpressed 3-mers

# Names of top over-expressed 3-mers in Saprospirales
sapro_3mers <- names(Saprospirales_3_mer$over)

# Initialize matrix: rows = organisms, columns = k-mers
compare_matrix <- matrix(0, nrow = 2, ncol = length(sapro_3mers),
                         dimnames = list(c("Saprospirales", "E.coli"), sapro_3mers))

# Fill counts for Saprospirales
compare_matrix["Saprospirales", ] <- Saprospirales_3_mer$over[sapro_3mers]

# Fill counts for E. coli: take counts if present, otherwise 0
compare_matrix["E.coli", ] <- sapply(sapro_3mers, function(k) {
  if(k %in% names(e.coli_prot_3_count)){
  # k %in% names
    e.coli_prot_3_count[k]   # use the raw count from E. coli
  } else {
    0
  }
})

barplot(compare_matrix,
        beside = TRUE,                  # side-by-side bars
        col = c("steelblue", "tomato"),  # Sapro = blue, E. coli = red
        las = 2,                         # rotate x-axis labels
        main = "Top Over-Expressed 3-mers in Saprospirales vs Counts in E. coli",
        ylab = "Frequency",
        cex.names = 0.8)
legend("topright", legend = c("Saprospirales", "E. coli"), fill = c("steelblue", "tomato"))
```

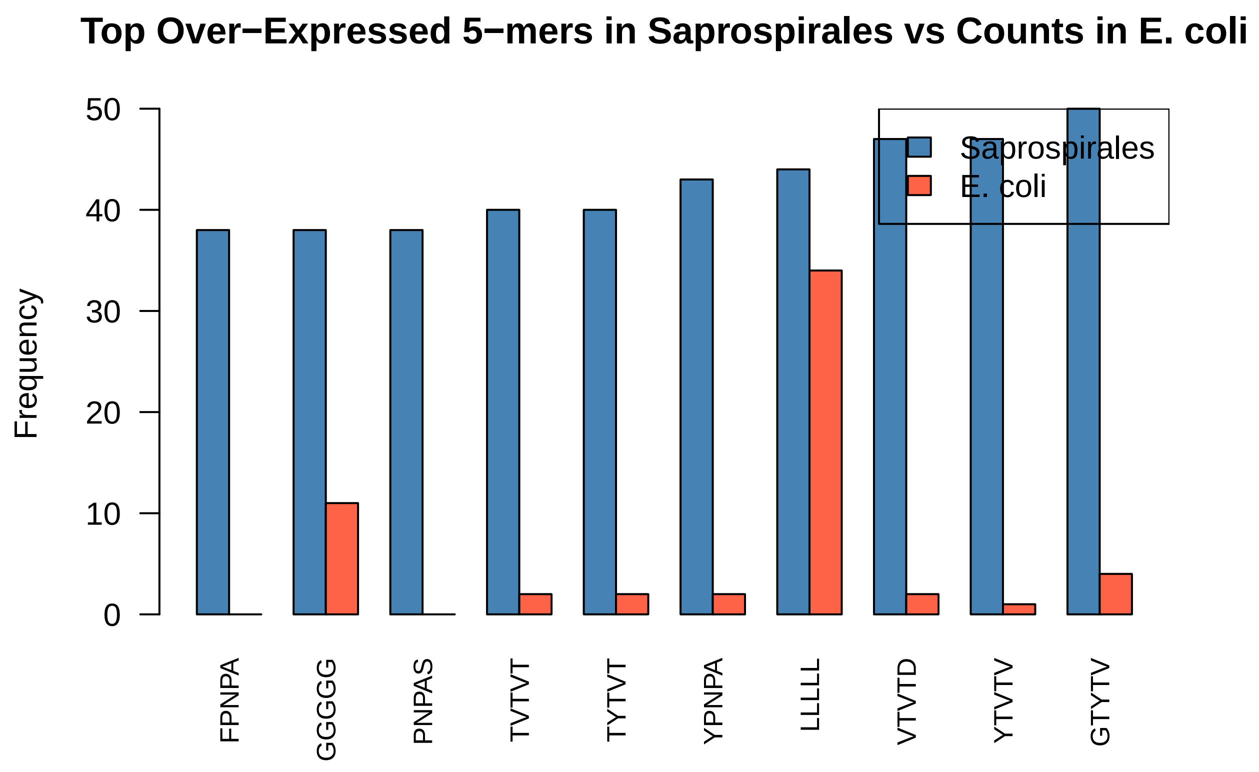


```r
# for overexpressed 4-mers

# Names of top over-expressed 3-mers in Saprospirales
sapro_4mers <- names(Saprospirales_4_mer$over)

# Initialize matrix: rows = organisms, columns = k-mers
compare_matrix <- matrix(0, nrow = 2, ncol = length(sapro_4mers),
                         dimnames = list(c("Saprospirales", "E.coli"), sapro_4mers))

# Fill counts for Saprospirales
compare_matrix["Saprospirales", ] <- Saprospirales_4_mer$over[sapro_4mers]

# Fill counts for E. coli: take counts if present, otherwise 0
compare_matrix["E.coli", ] <- sapply(sapro_4mers, function(k) {
  if(k %in% names(e.coli_prot_4_count)){
    e.coli_prot_4_count[k]    # use the raw count from E. coli
  } else {
    0
  }
})

barplot(compare_matrix,
        beside = TRUE,                   # side-by-side bars
        col = c("steelblue", "tomato"),  # Sapro = blue, E. coli = red
        las = 2,                         # rotate x-axis labels
        main = "Top Over-Expressed 4-mers in Saprospirales vs Counts in E. coli",
        ylab = "Frequency",
        cex.names = 0.8)
legend("topright", legend = c("Saprospirales", "E. coli"), fill = c("steelblue", "tomato"))
```

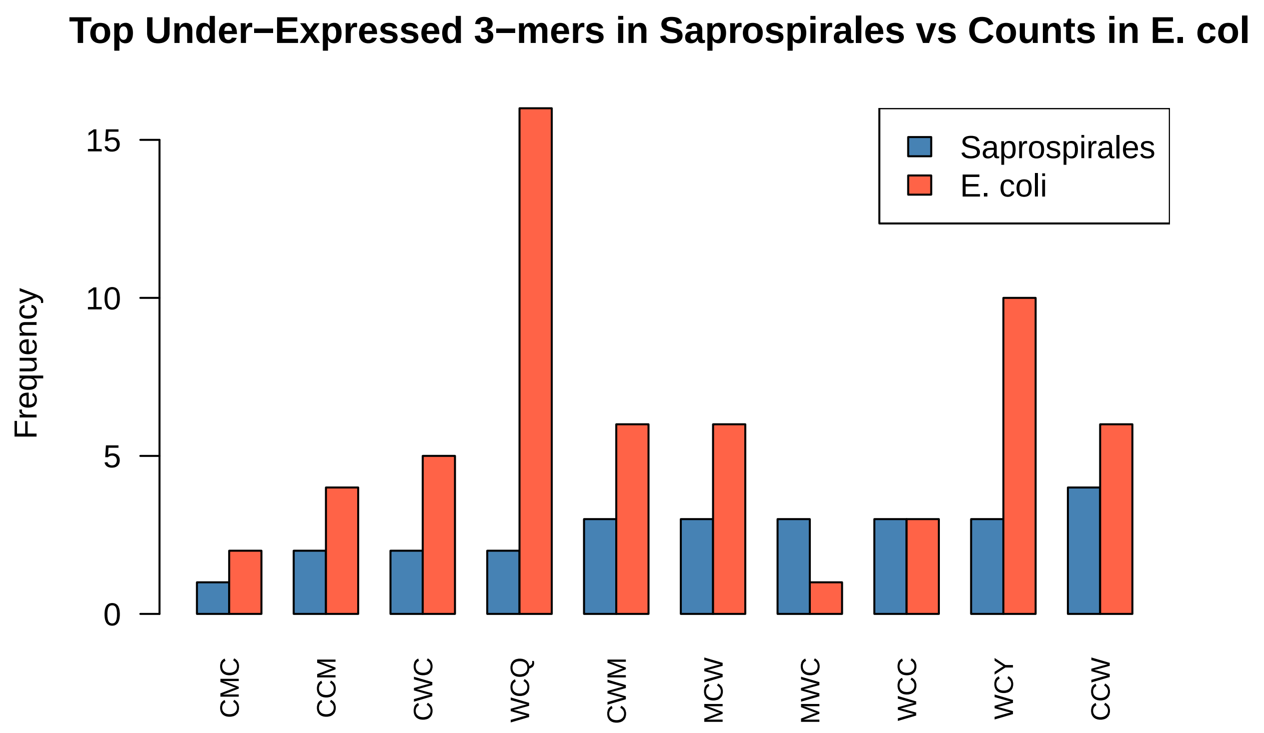


```
# for overexpressed 5-mers in Saprospirales

# Names of top over-expressed 5-mers in Saprospirales
sapro_5mers <- names(Saprospirales_5_mer$over)

# Initialize matrix: rows = organisms, columns = k-mers
compare_matrix <- matrix(0, nrow = 2, ncol = length(sapro_5mers),
                         dimnames = list(c("Saprospirales", "E.coli"), sapro_5mers))

# Fill counts for Saprospirales
compare_matrix["Saprospirales", ] <- Saprospirales_5_mer$over[sapro_5mers]

# Fill counts for E. coli: take counts if present, otherwise 0
compare_matrix["E.coli", ] <- sapply(sapro_5mers, function(k) {
  if(k %in% names(e.coli_prot_5_count)){
    e.coli_prot_5_count[k]    # use the raw count from E. coli
  } else {
    0
  }
})

barplot(compare_matrix,
```

```r
        beside = TRUE,                  # side-by-side bars
        col = c("steelblue", "tomato"),  # Sapro = blue, E. coli = red
        las = 2,                         # rotate x-axis labels
        main = "Top Over-Expressed 5-mers in Saprospirales vs Counts in E. coli",
        ylab = "Frequency",
        cex.names = 0.8)
legend("topright", legend = c("Saprospirales", "E. coli"), fill = c("steelblue", "tomato"))
```

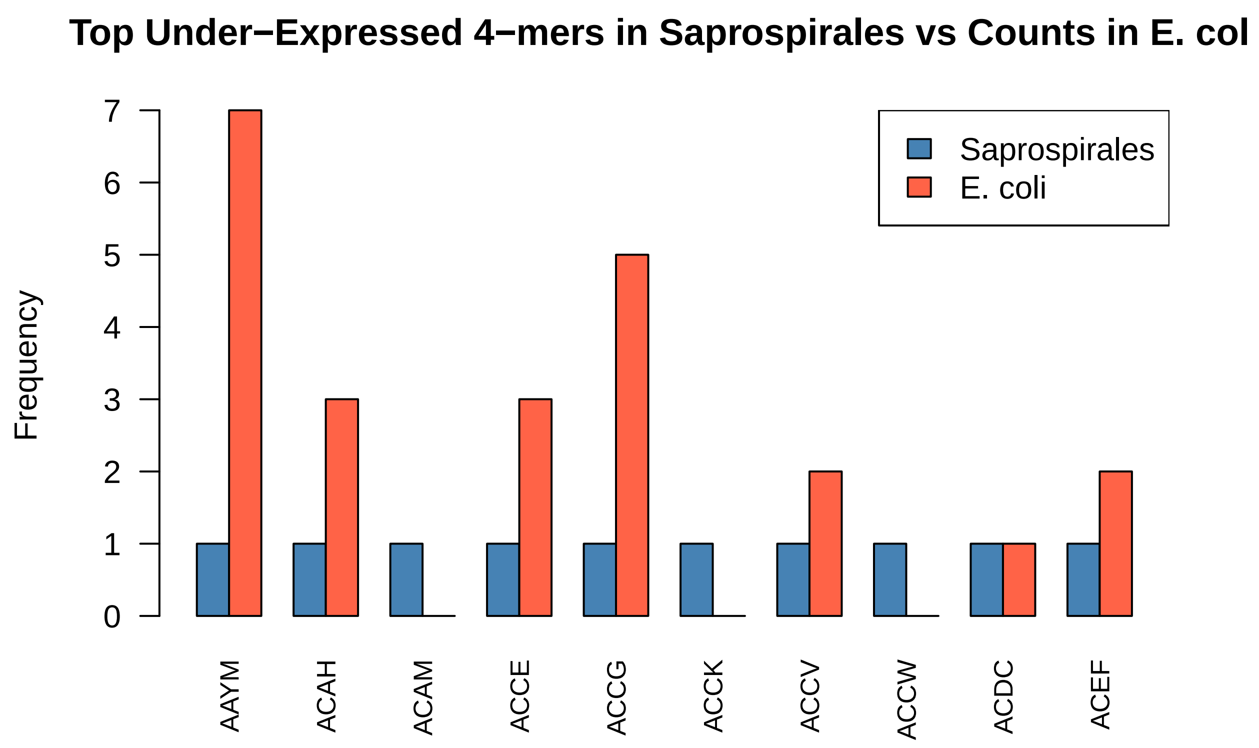


```r
# For under-expressed 3-mer genes of Saprospirales

# Names of top under-expressed 3-mers in Saprospirales
sapro_3mers_under <- names(Saprospirales_3_mer$under)

# Initialize matrix: rows = organisms, columns = k-mers
compare_matrix <- matrix(0, nrow = 2, ncol = length(sapro_3mers_under),
                         dimnames = list(c("Saprospirales", "E.coli"), sapro_3mers_under))

# Fill counts for Saprospirales
compare_matrix["Saprospirales", ] <- Saprospirales_3_mer$under[sapro_3mers_under]

# Fill counts for E. coli: take counts if present, otherwise 0
compare_matrix["E.coli", ] <- sapply(sapro_3mers_under, function(k) {
  if(k %in% names(e.coli_prot_3_count)){
    e.coli_prot_3_count[k]    # use the raw count from E. coli
  } else {
    0
  }
})

barplot(compare_matrix,
        beside = TRUE,                    # side-by-side bars
        col = c("steelblue", "tomato"),   # Sapro = blue, E. coli = red
        las = 2,                          # rotate x-axis labels
        main = "Top Under-Expressed 3-mers in Saprospirales vs Counts in E. coli",
        ylab = "Frequency",
        cex.names = 0.8)
legend("topright", legend = c("Saprospirales", "E. coli"), fill = c("steelblue", "tomato"))
```



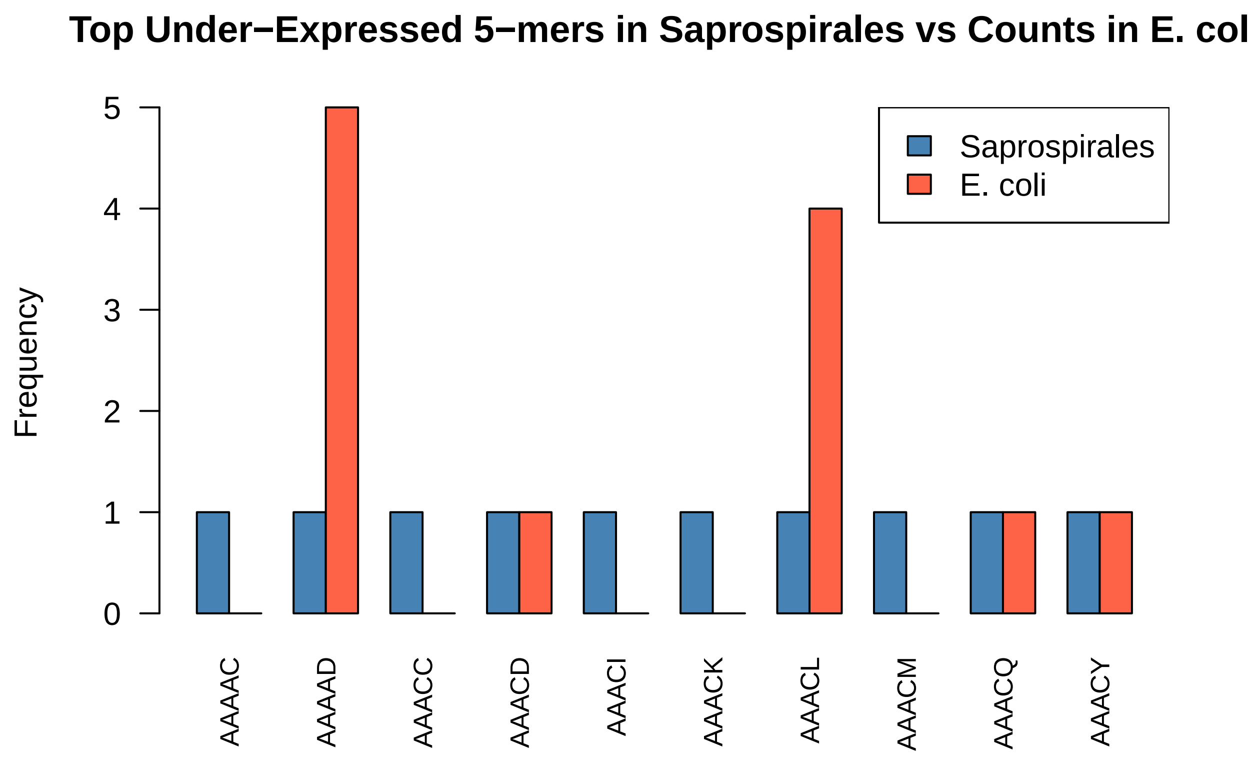
```
# For under-expressed 4-mer genes of Saprospirales

# Names of top under-expressed 4-mers in Saprospirales
sapro_4mers_under <- names(Saprospirales_4_mer$under)

# Initialize matrix: rows = organisms, columns = k-mers
compare_matrix <- matrix(0, nrow = 2, ncol = length(sapro_4mers_under),
                         dimnames = list(c("Saprospirales", "E.coli"), sapro_4mers_under))

# Fill counts for Saprospirales
compare_matrix["Saprospirales", ] <- Saprospirales_4_mer$under[sapro_4mers_under]

# Fill counts for E. coli: take counts if present, otherwise 0
compare_matrix["E.coli", ] <- sapply(sapro_4mers_under, function(k) {
```

```r
  if(k %in% names(e.coli_prot_4_count)){
    e.coli_prot_4_count[k]    # use the raw count from E. coli
  } else {
    0
  }
})

barplot(compare_matrix,
        beside = TRUE,                        # side-by-side bars
        col = c("steelblue", "tomato"),   # Sapro = blue, E. coli = red
        las = 2,                               # rotate x-axis labels
        main = "Top Under-Expressed 4-mers in Saprospirales vs Counts in E. coli",
        ylab = "Frequency",
        cex.names = 0.8)
legend("topright", legend = c("Saprospirales", "E. coli"), fill = c("steelblue", "tomato"))
```



```r
# For under-expressed 5-mer genes of Saprospirales

# Names of top under-expressed 5-mers in Saprospirales
sapro_5mers_under <- names(Saprospirales_5_mer$under)

# Initialize matrix: rows = organisms, columns = k-mers
compare_matrix <- matrix(0, nrow = 2, ncol = length(sapro_5mers_under),
                         dimnames = list(c("Saprospirales", "E.coli"), sapro_5mers_under))

# Fill counts for Saprospirales
```

```r
compare_matrix["Saprospirales", ] <- Saprospirales_5_mer$under[sapro_5mers_under]

# Fill counts for E. coli: take counts if present, otherwise 0
compare_matrix["E.coli", ] <- sapply(sapro_5mers_under, function(k) {
  if(k %in% names(e.coli_prot_5_count)){
    e.coli_prot_5_count[k]    # use the raw count from E. coli
  } else {
    0
  }
})

barplot(compare_matrix,
        beside = TRUE,                  # side-by-side bars
        col = c("steelblue", "tomato"),  # Sapro = blue, E. coli = red
        las = 2,                         # rotate x-axis labels
        main = "Top Under-Expressed 5-mers in Saprospirales vs Counts in E. coli",
        ylab = "Frequency",
        cex.names = 0.8)
legend("topright", legend = c("Saprospirales", "E. coli"), fill = c("steelblue", "tomato"))
```



The chatgpt was used to find the error in the code and to find the suitable codes when required.