Practical – k-mer distributions

Probabilities and statistics for modelling 1 (STAT1)

Jacques van Helden

2020-02-20

Contents

Introduction

In this practical, we will count k-mer occurrences in DNA sequences of different organisms (one organism per sutdent), fit different theoretical distributions of probabilities onto the empirical distributions of counts, and test the goodness of fit for these alternative distributions.

Organisms

Each student will choose one organism of interest among the following ones.

Taxon	Organism
Fungi	Saccharomyces_cerevisiae
Bacteria	Escherichia coli GCF 000005845.2 ASM584v2
Mammalian	Homo sapiens GRCh38
Mammalian	Mus musculus GRCm38
Bird	Gallus gallus EnsEMBL
Fish	Danio_rerio_EnsEMBL
Insect	Drosophila melanogaster
Worm	
Plant	Arabidopsis thaliana.TAIR10.29
Plant	Zea mays.AGPv3.29
Apicomplexa	

Solutions

Working directory

On your computer, create a directory for this practical. I suggest to use a consistent naming for the different practicals of this course.

We will further create a sub-folder with the name of our organism of interest.

```
## Define your organism of interest
org <- "Homo_sapiens"

## Define a working directory
work.dir <- file.path("~", "CMB-STAT2_practicals", "kmer_distrib", org)
dir.create(work.dir, recursive = TRUE, showWarnings = FALSE)


## Print a message with the result directory
message("Result directory\t", work.dir)
```

Counting k-mer occurrences in each promoter of a model organism

1. Open a connection to the Regulatory Sequence Analysis Tools (RSAT) teaching server : http://teaching.rsat.eu/
2. In the tool search box, type "retrieve sequence" and click on the corresponding tool.
3. In the *retrieve-sequence* form,
   - click *Mandatory inputs*, enter the name your organism of interest, and check the option *all genes of this organism*;
   - in *Mandatory options*, select *upstream*, and set the sequence limits from -1 to -500
   - in *Advanced options*, *make sure that this option is* ***unchecked****:* Prevent overlap with neighbour genes (noorf)* [1]
   - click *Run analysis* and *GO*.

After a few seconds (or minutes) the result is displayed. Right-click on the sequence file (extension fasta) and open it in a separate tab to check its content.

4. Come back to the result page of retrieve-sequence. In the *Next Step* box below the result, click on the link to *oligo-analysis*. This will transfer your sequences to the oligo-analysis form.
   - In the *Sequence* section, inactivate the option *purge sequence*.
   - In the *Oligmer counting mode*, **uncheck** the option *prevent overlapping matches*.
   - Select *Count on single strand*.
   - For *oligomer lengths*, select 2 and **uncheck the other lengths**.
   - In *Results*, check the option *Occurrence table*.
   - Type your email address and select the mail output.
   - Click *GO*.

After a few seconds (minutes), the RSAT server should display the result page, with links to the k-mer count table. Copy the URL of the result file.

Download the count table from RSAT

Let us define the name we will give to the local copy of the k-mer count table generated on the RSAT server in the previous steps.

[1] Note: normally it is recommended to check this option, but we intently inactivate it in order to get sequences of the same sizes.

```
## Define the path and the name of the local file
kmer.file <- file.path(work.dir, "2nt-ovlp-1str_Homo_sapiens.tab")
```

One solution is to download manually the k-mer count table generated on the RSAT server, move it to the work directory, and rename it `2nt-ovlp-1str_Homo_sapiens.tab` (to be adapted depending on your organism of interest).

Another possiblity is to use R command `download.file()` download it from the URL of the result file on the RSAT server.

```
## Note: this will only work for a few days, because the temporary files are removed from the server
temp.url <- "http://pedagogix-tagc.univ-mrs.fr/rsat/tmp/www-data/2020/02/17/oligo-analysis_2020-02-17.15

## Provide the arguments in the order of the function definition
download.file(temp.url, kmer.file)

## Equivalent : name the arguments
# download.file(url = temp.url, destfile = kmer.file)

## Note: named arguments can be provided in a different order without problem
# download.file(destfile = kmer.file, url = temp.url)
```

Whichever method was chosen, check that the file is at the right place on your computed.

```
## List the files in the working directory
list.files(work.dir)
```

```
[1] "2nt-ovlp-1str_Homo_sapiens.tab"
```

```
## Send a message with the k-mer file location
message("K-mer count table file:\t", kmer.file)
```

Load the k-mer count table in R

Use the finction `read.table()` to load the k-mer count table in a variable named `kmer.table`.

```
## Call the help for read.table()
# ?read.table

## Load the k-mer count table in a variable
kmer.table <- read.table(
  file = kmer.file,
  comment.char = ";", ## comment lines start with ";" in RSAT
  header = TRUE, # the first row (after the comments) contain the column headers
  row.names = 1, ## the first column contains row names but there might be homonyms
  sep = "\t" ## column separator is the tabulation
  )

## We set uppercases for the dinucleotide sequences, which were in lowercases in the original count tabl
names(kmer.table) <- toupper(names(kmer.table))
```

Check the dimensions of this table.

```
## Check the dimensions of the k-mer count table
dim(kmer.table)
```

```
[1] 60675    16
```

```
## Number of k-mers
m <- ncol(kmer.table)

## Number of genes
n <- nrow(kmer.table)

## Print the result
print(paste0("Number of rows: ", n))
```

```
[1] "Number of rows: 60675"
```

```
print(paste0("Number of columns: ", m))
```

```
[1] "Number of columns: 16"
```

Check the column names

```
## Check the column names
names(kmer.table)
```

```
 [1] "AA" "AC" "AG" "AT" "CA" "CC" "CG" "CT" "GA" "GC" "GG" "GT" "TA" "TC" "TG" "TT"
```

```
colnames(kmer.table) # equivalent
```

```
 [1] "AA" "AC" "AG" "AT" "CA" "CC" "CG" "CT" "GA" "GC" "GG" "GT" "TA" "TC" "TG" "TT"
```

Display the first and last 10 lines of the k-mer count table.

```
## Show the first 10 lines of the k-mer count table
head(kmer.table)
```

```
                                        AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT
ENSG00000210049|Homo_sapiens_GRCh38|MT-TF   48 54 17 36 58 69 14 32 14 24  7 11 35 26 18 36
ENSG00000211459|Homo_sapiens_GRCh38|MT-RNR1 56 58 16 36 62 69  9 31 10 19  7 11 38 25 15 37
ENSG00000210077|Homo_sapiens_GRCh38|MT-TV   57 42 41 22 37 38 18 35 28 26 22 27 39 23 22 22
ENSG00000210082|Homo_sapiens_GRCh38|MT-RNR2 48 42 42 22 35 39 17 36 30 25 22 28 41 21 24 27
ENSG00000209082|Homo_sapiens_GRCh38|MT-TL1  52 41 32 34 40 40 19 29 28 16 23 23 38 32 16 34
ENSG00000198888|Homo_sapiens_GRCh38|MT-ND1  48 38 35 32 39 37 21 29 30 16 24 25 36 37 15 35
```

```
tail(kmer.table)
```

```
                                         AA AC AG AT CA CC CG CT GA GC GG GT TA TC TG TT
ENSG00000128973|Homo_sapiens_GRCh38|CLN6           55 24 34 31 31 31 15 36 30 34 29 19 27 24 34 45
ENSG00000272269|Homo_sapiens_GRCh38|RP11-500C11.3  41 29 33 14 43 90 25 33 21 38 32 15 11 35 16 23
ENSG00000267091|Homo_sapiens_GRCh38|CTBP2P7        21 22 38 25 40 34  9 37 30 35 28 32 15 29 50 54
ENSG00000151655|Homo_sapiens_GRCh38|ITIH2          62 30 35 33 48 23  2 41 27 22 17 19 23 39 30 48
ENSG00000234159|Homo_sapiens_GRCh38|RBPMSLP       112 30 39 36 41 34  4 24 32 20 18 14 33 19 23 20
ENSG00000141338|Homo_sapiens_GRCh38|ABCA8          74 22 39 48 31 13  4 30 43 21 19 16 35 22 37 45
```

Compute marginal statistics

Tips: use the R function `apply()`.

We will compute marginal statistics on the rows and columns of the count table. One possibility is to add columns and rows on this table, but this would not be very convenient for the subsequent computations to be performed on the counts. We will thus create two separate tables: one with the row-wise statistics, and another one with the column-wise statistics.

Rather than looping over each row or column of the count table, we can use the function `apply()`in order to compute a chosen statistics on each row (margin = 1) or column (margin = 2) of the table.

```
## Recall the dimensions of the kmer count table
dim(kmer.table)
```

```
[1] 60675    16
```

```
# View(kmer.table)

## Compute means per column (on the "second" margin)
col.means <- apply(kmer.table, 2, mean)

print(col.means)
```

```
      AA       AC       AG       AT       CA       CC       CG       CT       GA       GC       GG
40.51309 24.69333 36.14415 28.82729 35.02548 36.94124 14.57330 35.30629 30.15806 30.75028 36.17958 24.0
```

We can run the `apply()`function in the same way to compute the other descriptive statistics (median, variance, ...) and store each result in a separate vector. However, it would be more convenient to dispose of a single structure containing all the statistics for each column. For this, we use `data.frame()` to create a data frame with one column per column per computed statistics, and one row per k-mer.

```
## Compute a table summarising the column-wise statistics of the k-mer count table
col.stats <- data.frame(
  min = apply(kmer.table, 2, min),
  mean = apply(kmer.table, 2, mean),
  min = apply(kmer.table, 2, min),
  p05 = apply(kmer.table, 2, quantile, 0.05),
  Q1 = apply(kmer.table, 2, quantile, 0.25),
  median = apply(kmer.table, 2, median),
  Q3 = apply(kmer.table, 2, quantile, 0.75),
  p95 = apply(kmer.table, 2, quantile, 0.95),
  max = apply(kmer.table, 2, max),
  sd = apply(kmer.table, 2, sd),
  var = apply(kmer.table, 2, var)
)


kable(col.stats, caption = "Column-wise means of the k-mer count table. ")
```

Table 2: Column-wise means of the k-mer count table.

	min	mean	min.1	p05	Q1	median	Q3	p95	max	sd	var
AA	0	40.51309	0	10	25	39	54	77	186	20.706267	428.74949
AC	0	24.69333	0	14	20	24	29	36	161	6.860027	47.05997
AG	0	36.14415	0	23	30	36	42	51	139	9.103262	82.86939
AT	0	28.82729	0	7	17	28	39	54	186	14.821645	219.68117
CA	0	35.02548	0	22	30	35	40	48	173	8.344418	69.62931
CC	0	36.94124	0	12	22	33	47	76	208	20.350163	414.12914
CG	0	14.57330	0	1	3	8	20	51	113	16.515726	272.76920
CT	0	35.30629	0	22	29	35	41	50	137	8.887525	78.98810
GA	0	30.15806	0	18	25	30	35	44	139	8.455523	71.49587
GC	0	30.75028	0	12	19	27	38	63	113	15.779050	248.97841
GG	0	36.17958	0	12	22	33	46	73	208	19.283920	371.86955
GT	0	24.01994	0	14	20	24	28	35	130	6.850075	46.92352
TA	0	24.42770	0	5	13	23	34	49	184	13.962177	194.94237
TC	0	29.55186	0	17	24	29	34	43	139	8.177379	66.86953
TG	0	34.20470	0	20	28	34	40	48	144	8.940656	79.93532

	min	mean	min.1	p05	Q1	median	Q3	p95	max	sd	var
TT	0	37.63466	0	10	23	36	50	72	163	19.433749	377.67061

Warning about R sd() and var() functions

Beware: the R functions `sd` and `var` do not compute the standard deviation and variance of the numbers provded! Instead, they consider that the data provided are a sample drawed from a population, and the return an estimate of the standard deviation or variance of this population.

So, instead of the sample variance

$$s^2 = \sum_{i=1}^{n} \frac{1}{n}(x - \bar{x})^2$$

They estimate the variance of the population corrected for the systematic bias of sample variance.

$$\hat{s^2} = \sum_{i=1}^{n} \frac{1}{n-1}(x - \bar{x})^2$$

For this exercise, this is actually what we want, since we can consider that the 60675 genes of our organism of interest are a sampling of all the genes that might be found in organisms of the same taxon. In any case, the impact of the correction is negligible with such a large number of genes ($n = 60675$).

$$\frac{n}{n-1} = 1.0000165$$

Drawing empirical distributions of counts

Draw an histogram with the distribution of counts for a given k-mer.

The simplest way to generate a histogram is to use the `hist()` function. Here is the result with the default parameters.

```
kmer <- "AG" # Select a k-mer
counts <- kmer.table[, kmer] # select the counts for this k-mer
range(counts)
```

```
[1]   0 139
```

```
## Draw an histogram with default parameters (not very beautiful)
hist(x = counts)
```

We can fine-tune the parameters in order to get a more informative representation, by specifying custom breaks.

```
## Draw a nice and informative histogram
hist(x = counts,
     breaks = 0:max(counts + 1),
     col = "#44DDFF", border = "#44DDFF",
     las = 1,
     ylab = "Number of promoters",
     xlab = "K-mer counts",
     main = paste0(kmer, " counts in promoters"))
```

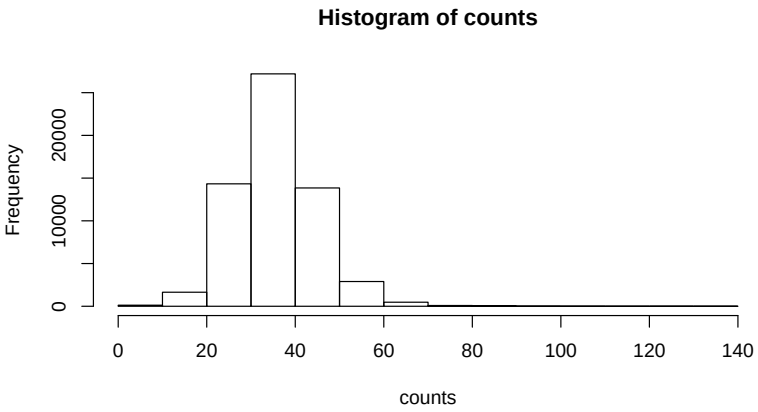

Figure 1: Counts per sequence in all human promoters.

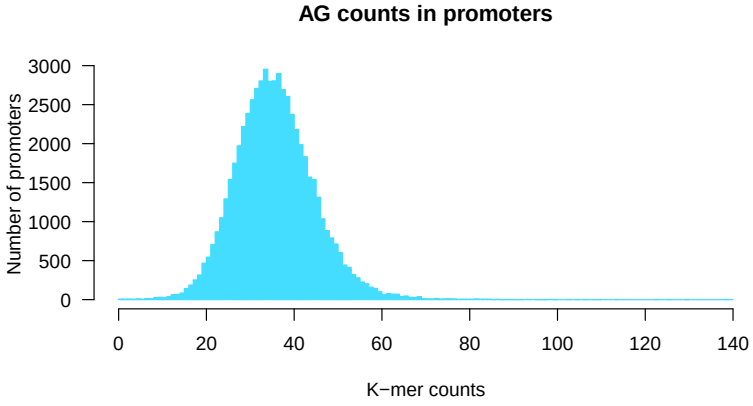

Figure 2: Counts per sequence in all human promoters.

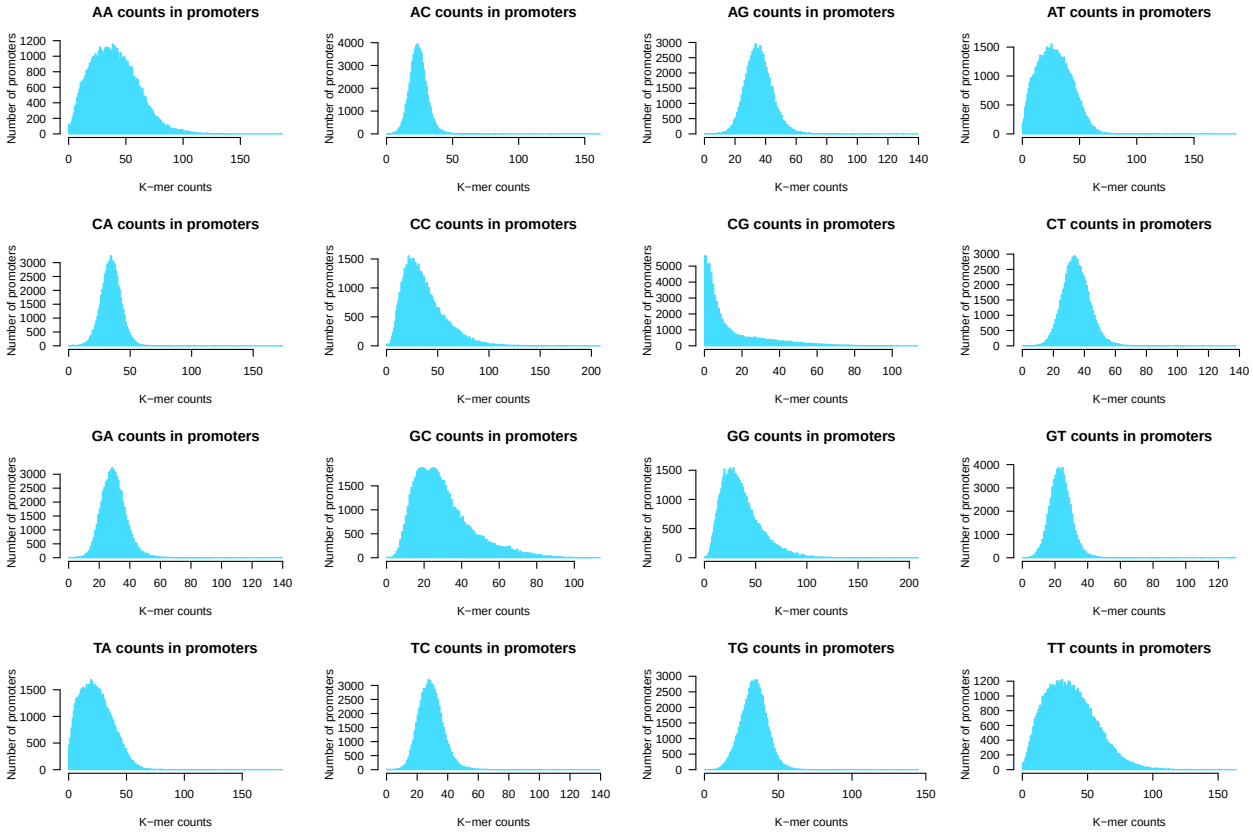

Figure 3: Dinucleotide counts per sequence in all human promoters.

Generate a figure with 4 x 4 panels to depict the histograms of the 16 k-mers

```
par(mfrow = c(4,4))
par(mar = c(4.1, 5.1, 4.1, 1.1))
for (i in 1:16) {
  kmer <- names(kmer.table)[i]
  counts <- kmer.table[, i] # select the ith column of the k-mer table
  ## Draw a nice and informative histogram
  hist(x = counts,
       breaks = 0:max(counts + 1),
       col = "#44DDFF", border = "#44DDFF",
       las = 1,
       ylab = "Number of promoters",
       xlab = "K-mer counts",
       main = paste0(toupper(kmer), " counts in promoters"))
}
```

```
par(mfrow = c(1,1))
par(mar = c(4.1, 5.1, 4.1, 2.1))
```

Use other graphical representations to get an insight of the k-mer count distributions (boxplots, violin plots)

Boxplot

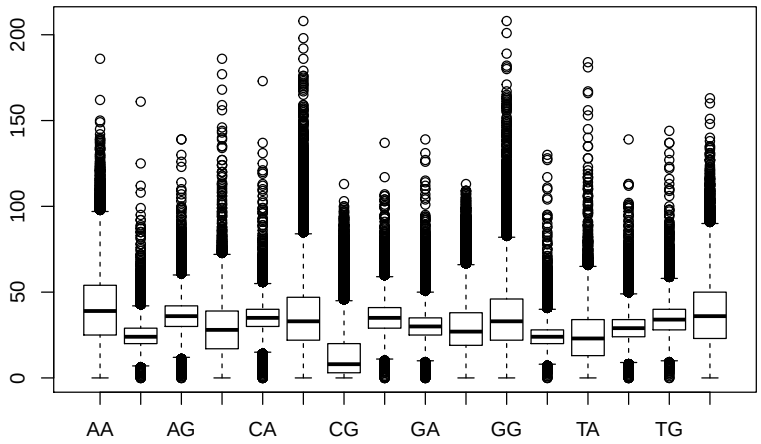

Figure 4: Boxplot of dinucleotide counts in all promoters.

The funciton `boxplot()` provides an informative summary of the data.

```
boxplot(x = kmer.table)
```

We can fine-tue the parameters to enhance the readability

```
boxplot(x = kmer.table, horizontal = TRUE, las = 1, col = "#44DDFF", main = "Dinucleotide distributions
```

Violin plot

We can use the R function `vioplot()` to generate a violin plot. This requires to install the R package `vioplot`, if it is not already done.

```
## Check if the vioplot package is already installed
if (!require(vioplot)) {
  ## Install the vioplot package
  install.packages("vioplot")
}
library(vioplot) ## Load the vioplot package
# help(package = vioplot) # Get help on the package
# help(vioplot) # Get help on the vioplot function

vioplot(kmer.table)
```

Let us fine-tune the parameters to get a pretty violin plot.

```
vioplot(kmer.table, horizontal = TRUE, col = "#44DDFF", las = 1, main = "Violin plot of dinucleotide co
```

Exploring k-mer count distributions in promoter sequences

3. Use other graphical representations to get an insight of the k-mer count distributions (boxplots, violin plots)
4. Compute summary statistics for each column of the count table, including the following estimators
   - min and max
   - mean
   - percentiles 05, 25 (=Q1), 50 (=median), 75 (=Q3), 95
   - variance and standard deviation
   - sum

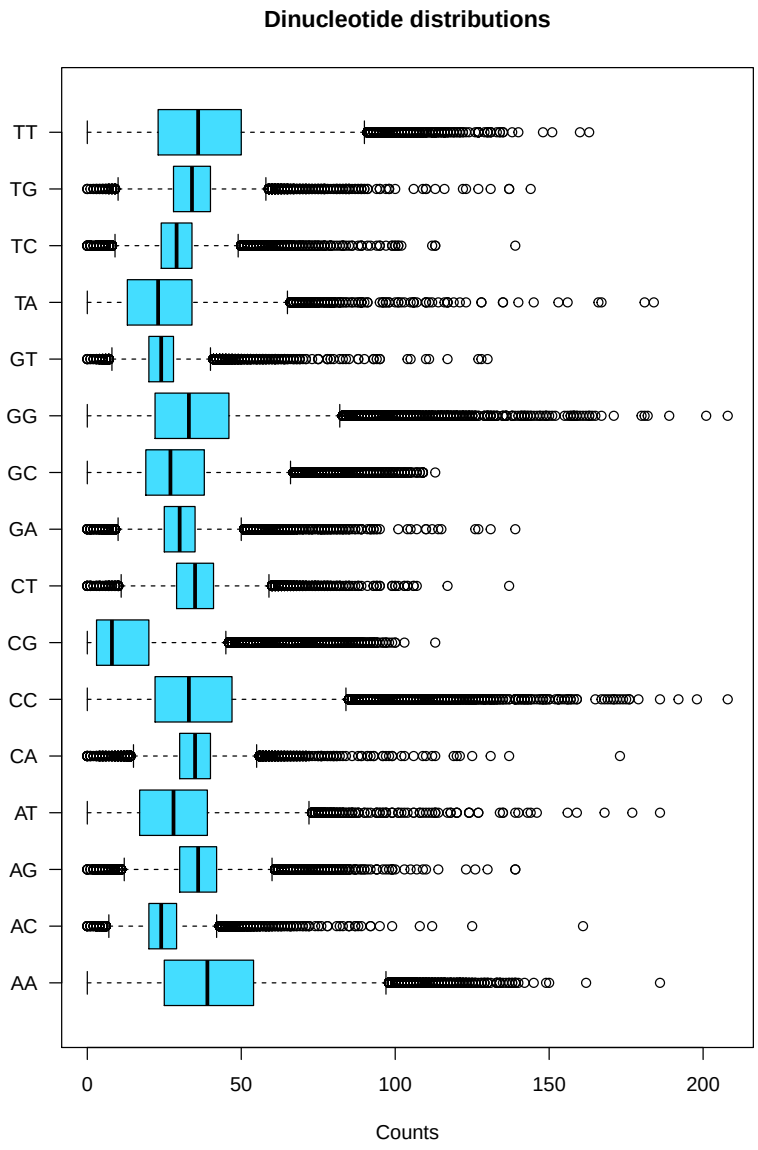

Figure 5: Boxplot of dinucleotide counts in all promoters.

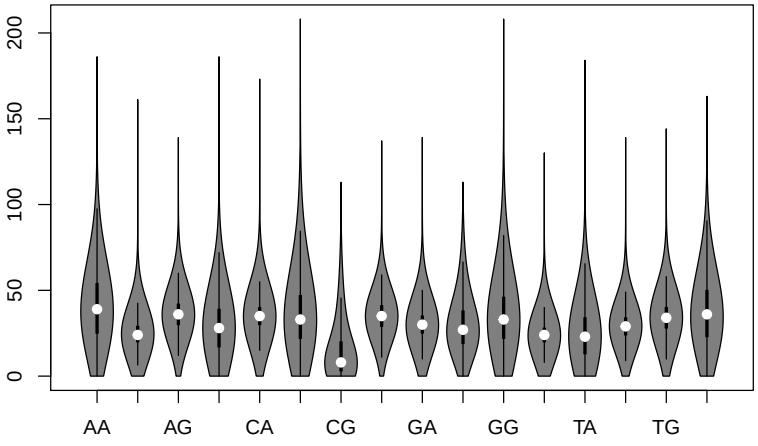

Figure 6: Violin plot of dinucleotide counts in promoter sequences

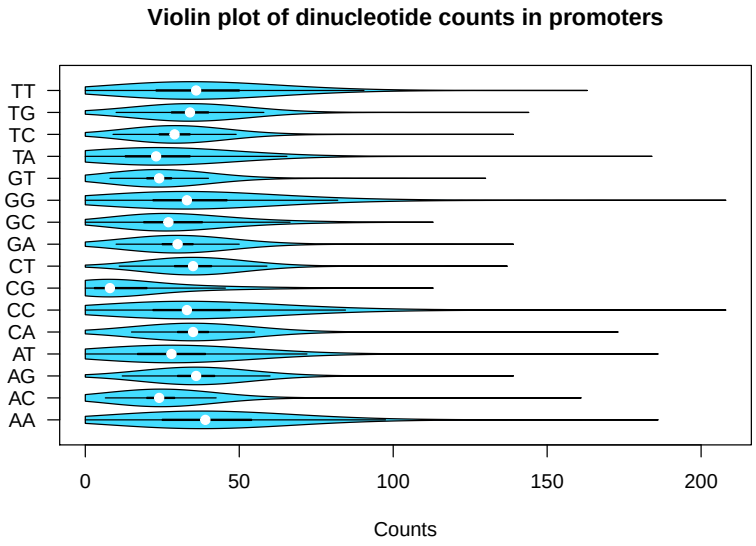

Figure 7: Violin plot of dinucleotide counts in promoter sequences

5. Compute a vector with the relative frequency of each k-mer in all the sequences.
6. Compute a table with the relative frequencies of k-mers per sequence, and compute similar summary statistics per column on this relative frequency table.
7. Write a brief interpretation of the results.

Fitting distributions

1. Fit a Poisson distribution on each empirical distribution of k-mer counts.
   - How do you choose the parameters?
   - Draw the fitted Poisson distribution (as a frequency polygon) over the histogram of k-mer occur
2. Do the same with the following distributions :
   a. binomial
   b. hypergeometric
   c. normal
   d. negative binomial
3. Estimate the goodness of fit for these distributions.