Exploration of a genomic annotations table (GTF)

DUBii 2019

Jacques van Helden, Hugo Varet and Frédéric Guyon

2019-01-23

Exploration of a genomic annotations table (GTF)

Goal of the practical session

During this practical session, we will cover the following items:

1. Manipulate a table containing genomics data (*E. Coli* genome annotations).
2. Select a subset of the data/rows according to a given criterion.
3. Generate different graphical representations of these data.
4. Compute statistics describing the different types of annotations.

The GTF file format

The **GTF** (**General Transfer Format**) file format is extensively used to provide easily readable genomics annotations while being very handy with a computer.

Text files,

- one row per genomic "object" (gene, transcript, exon, intron, CDS, ...)
- one column per attribute (name, source, object type, genomic coordinates, description).

The GTF format is described on the following websites:

- http://www.ensembl.org/info/website/upload/gff.html
- https://genome.ucsc.edu/FAQ/FAQformat.html#format4

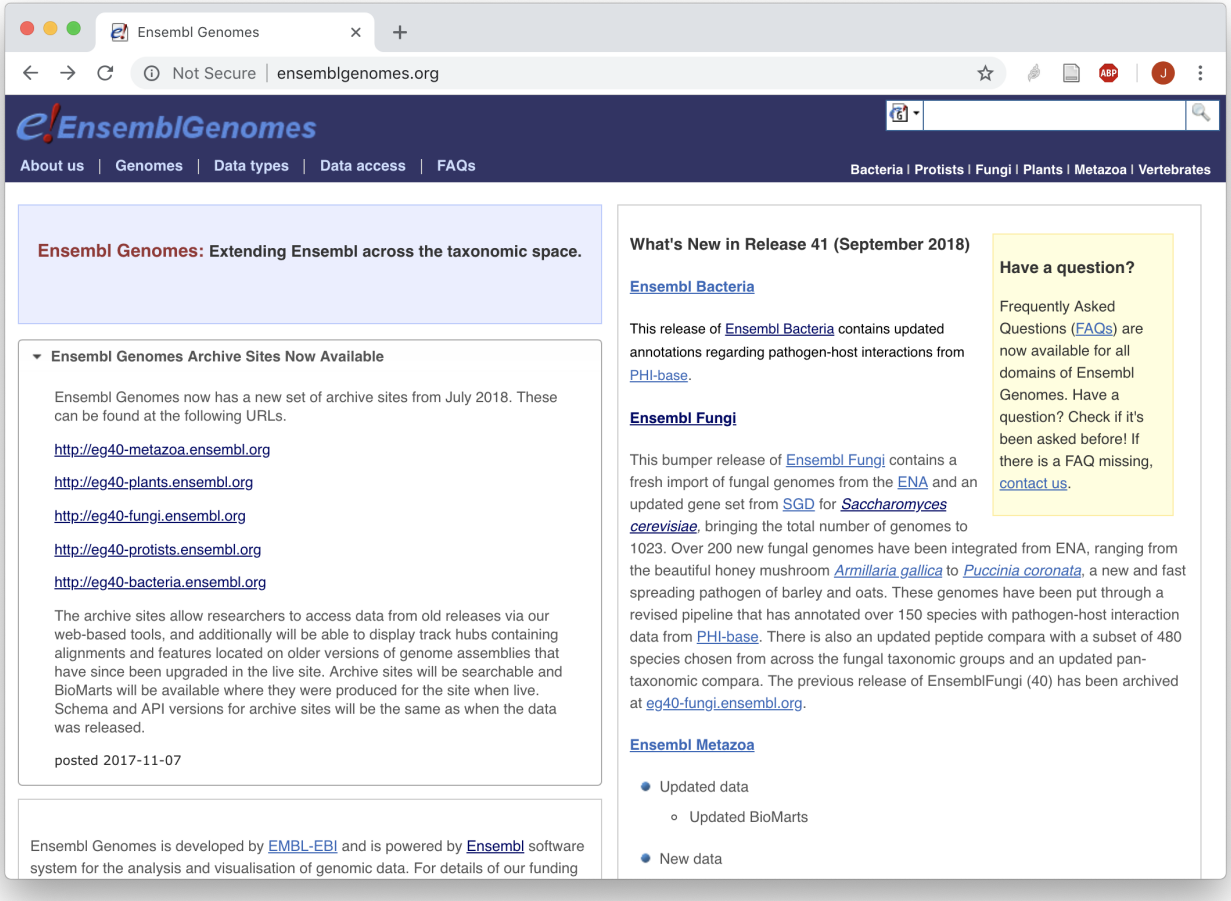
Find the GTF file of your favorite organism (on Ensembl)

1. Visit http://ensemblgenomes.org/.
2. Click on the link Bacteria.
3. Click on Download
4. In the **Filter** box, write *Escherichia coli*. The list of the proposed organisms is changing while you are writing.

For this session we will use the *E. Coli* GTF annotation file available here.

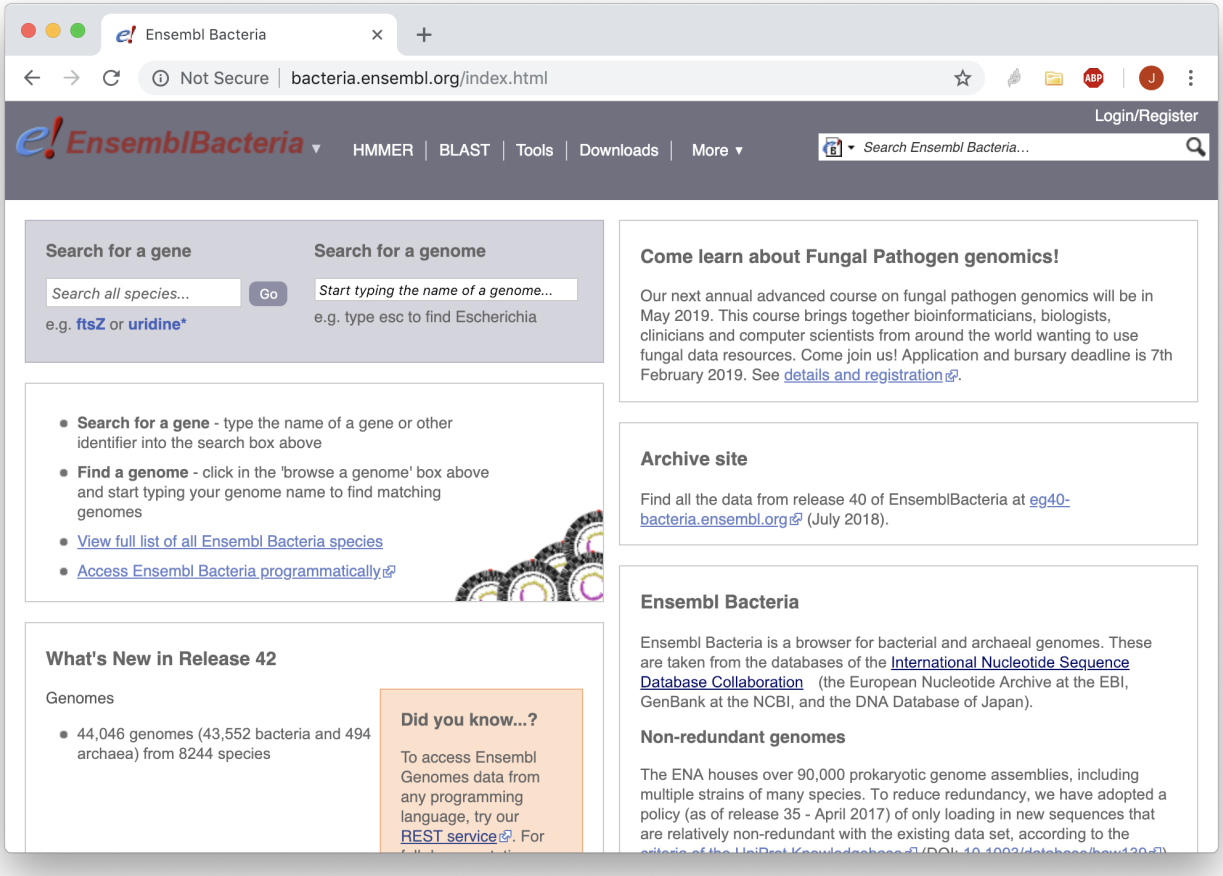
Page d'accueil d'EnsemblGenomes

http://ensemblgenomes.org/



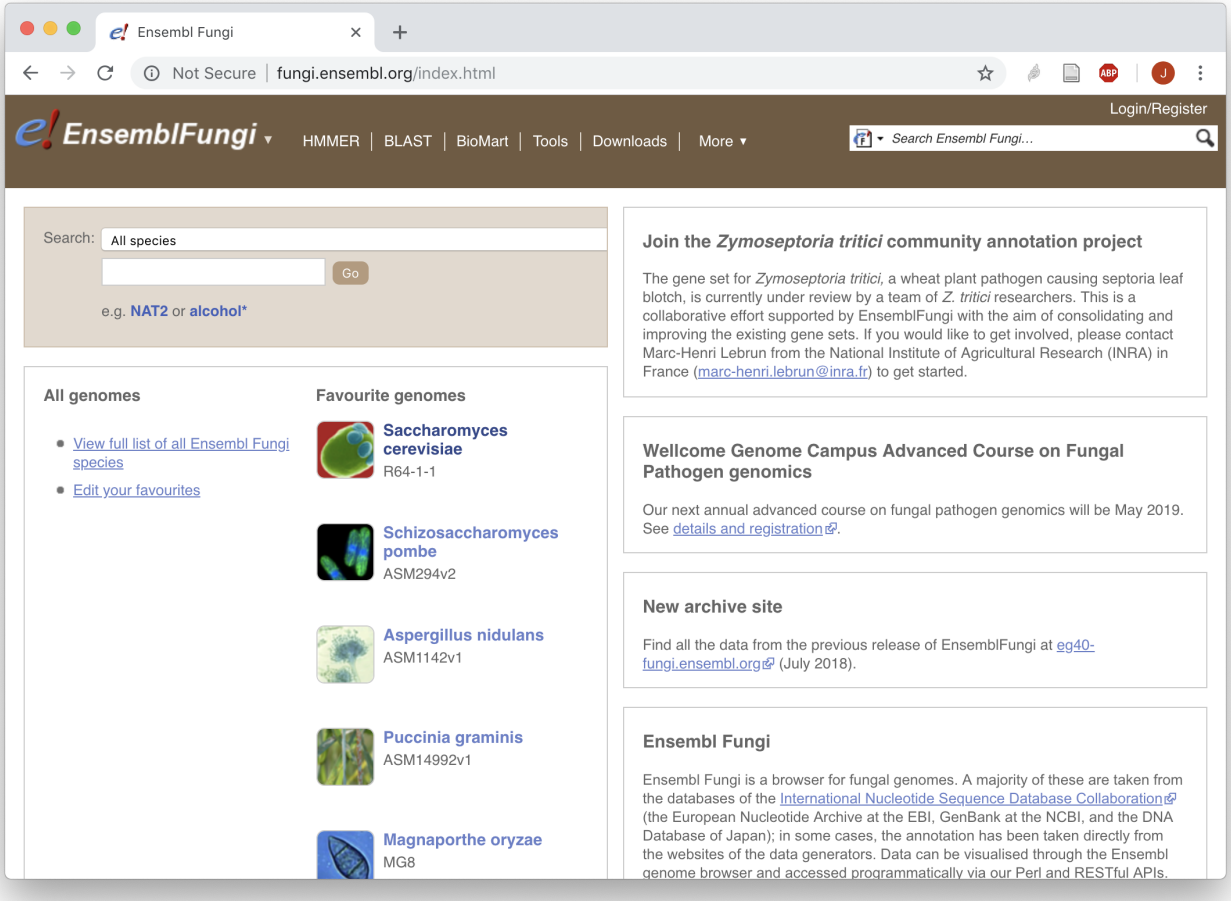
EnsemblGenomes Bacteria

http://bacteria.ensembl.org/



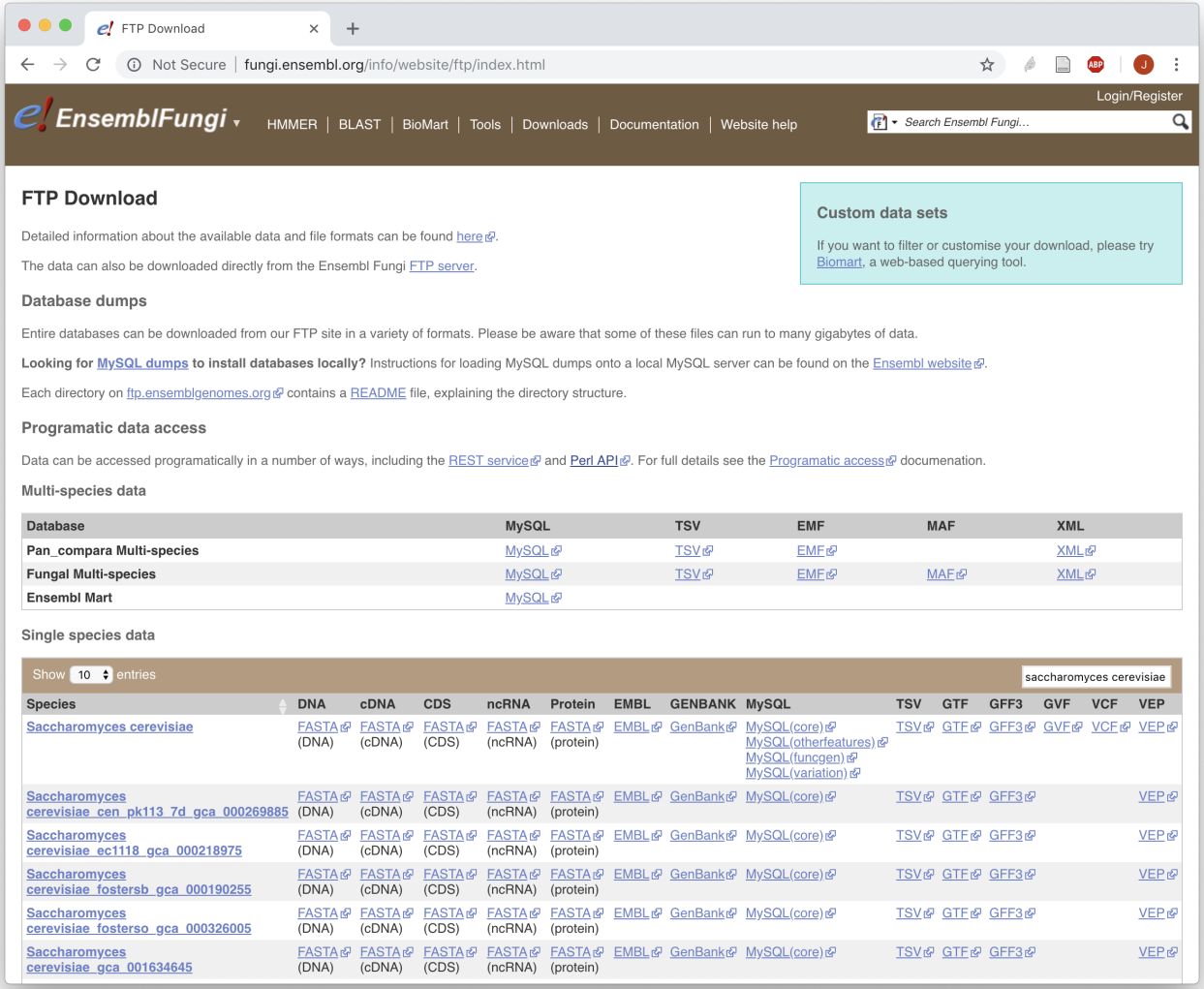
EnsemblGenomes Fungi

http://fungi.ensembl.org/



EnsemblGenomes Fungi Download page

http://fungi.ensembl.org/info/website/ftp/



Define and create your working directory

Exercise: create a working directory named `workDir` in your home folder and go inside.

```
## Define the working directory
workDir <- "~/intro_R/explorer_un_GTF"

## Create the working directory
dir.create(workDir, recursive = TRUE, showWarnings = FALSE)

## Go to the working directory
setwd(workDir)
getwd()        ## Check your current location
list.files()   ## List files (should be empty if just create
```

Downloading the GTF file

Exercise: download the GTF file in the working directory (optionally, adapt the command to load a GTF of your interest). Before downloading the file we check if it is already present in the working directory. If yes, we skip the download.

Tip: use the commands `file.exists`, `download.file`.

Downloading the GTF file: solution

```
## Define the URL of the file to download
gtf.url <- "ftp://ftp.ensemblgenomes.org/pub/bacteria/relea
## create a directory to store the file
dir.create("data", showWarnings = FALSE)
## create a local filename
destfile <- paste0("data/", basename(gtf.url))
print(destfile)
```

```
[1] "data/Escherichia_coli_str_k_12_substr_mg1655.ASM584v2
```

```
## Download the file, but only if not yet there
if (file.exists(destfile)) {
  message("GTF annotation file already there: ", destfile)
} else {
  message("Downloading GTF annotation file")
  download.file(url = gtf.url, destfile = destfile)
```

Loading a data table in R

Commands: `read.table`, `read.delim`, `read.csv`.

R includes several types of tabular structures (matrix, data.frame, table). The most widely used is `data.frame()`, which consists in a table of values with a type (strings, integer, ..) attached to each column, and names associated to rows and columns.

The function `read.table()` enables to read a text file containing tabular data, and to store its content in a variable.

Several finctions derived from `read.table()` facilitate the loading of different formats.

- `read.delim()` for files where a particular charcater is used as column separator (by default the tab character "̃").
- `read.csv()` for "comma-searated values".

Loading the GTF file

Load the GTF file in a variable named `featureTable`.

Tip: command `read.delim`.

```
## Load GTF file in a data.frame
featureTable <- read.delim(destfile, comment.char = "#", se
                           header=FALSE, row.names = NULL)

## The GTF format has no header, but we can define it based
names(featureTable) <- c("seqname", "source", "feature", "s
                         "score", "strand", "frame", "attri
```

Exploring the content of a data table

Immediately after having loaded a data table, check its dimensions.

```
dim(featureTable) ## Dimensions of the tbale
```

```
[1] 25979     9
```

```
nrow(featureTable) ## Number of rows
```

```
[1] 25979
```

```
ncol(featureTable) ## Number of columns
```

```
[1] 9
```

Checking heads and tails

Displaying the full annotation table would not be very convenient, since it contains tens of thousands of rows.

We can display the first rows of the file with the function `head()`, and the last rows with `tail()`.

```
## Display the 5 first rows of the feature table
head(featureTable, n = 5)

## Display the 5 last rows of the feature table
tail(featureTable, n = 5)
```

Viewing a table

If you are using the **RStudio** environment, you can display the table in a dynamic viewer pane with the function `View()`.

```
## In RStudio, display the table in a separate tab
View(featureTable)
```

The `View()` function is interactive, so it should not be used in a script because it would perturbate its execution.

Selecting columns

The last column of GTF files is particularly heavy, it contains a lof of semi-structured information.

We can select the 8 first columns and display the 5 first rows of this sub-table.

```
## Column selection + head
head(featureTable[,1:8], n=5)
```

```
     seqname source     feature start end score strand fra
1 Chromosome    ena        gene   190 255     .      +
2 Chromosome    ena  transcript   190 255     .      +
3 Chromosome    ena        exon   190 255     .      +
4 Chromosome    ena         CDS   190 252     .      +
5 Chromosome    ena start_codon   190 192     .      +
```

```
## Equivalent: selecting subsets of rows and columns
```

Feature types

Exercise: the column *feature* of the GTF indicates the feature table.

- List the feature types found in the GTF
- Count the number of features per type, and sort them by decreasing values.

Tip: commands `unique`, `table` and `sort`.

```
## List the types of features
unique(featureTable$feature)
```

```
[1] gene         transcript   exon         CDS          start_c
Levels: CDS exon gene start_codon stop_codon transcript
```

```
## Count the number of features per type
sort(table(featureTable$feature), decreasing = TRUE)
```

Décompte par valeur

The `table()` function allows to count the frequency of each value in a qualitative variable:

```
## Count the number of features per chromosome
table(featureTable$seqname)
```

```
Chromosome
     25979
```

```
## Count the number of features per strand
table(featureTable$strand)
```

```
    -     +
13246 12733
```

Contingency table

We can compute the number of combinations between two qualitatives variables:

```
##  Table with two vectors
table(featureTable$strand, featureTable$feature)
```

```
   CDS exon gene start_codon stop_codon transcript
- 2129 2307 2277        2128       2128       2277
+ 2012 2257 2220        2012       2012       2220
```

```
## Same result with a 2-column data frame
table(featureTable[, c("strand", "feature")])
```

```
      feature
strand  CDS exon gene start_codon stop_codon transcript
     - 2129 2307 2277        2128       2128       2277
     + 2012 2257 2220        2012       2012       2220
```

Computing feature lengths

- Add a column with feature lengths.

Note about feature length computation (explain why) :

$$L = \text{end} - \text{start} + 1$$

```
## Add a column to the table with genes lengths
featureTable$length <- featureTable$end - featureTable$star
```

Filtering rows based on a column content

The function `subset()` enables to select a subset of rows based on a filter applied to the content of one or several columns.

We can use it to select the subset of features corresponding to genes.

Selecting genes from the GTF table

- Select of genes from the GTF table and store them in a separate variable named genes.
- Compute summary statistics about gene lengthhs

Tip: commands `subset`, `summary`.

```
## Select subset of features having "CDS" as "feature" attr
genes <- subset(featureTable, feature == "gene")

## Print a message with the number of genes
message("Number of genes: ", nrow(genes))

## Compute basic statistics on genes lengths
summary(genes$length)
```

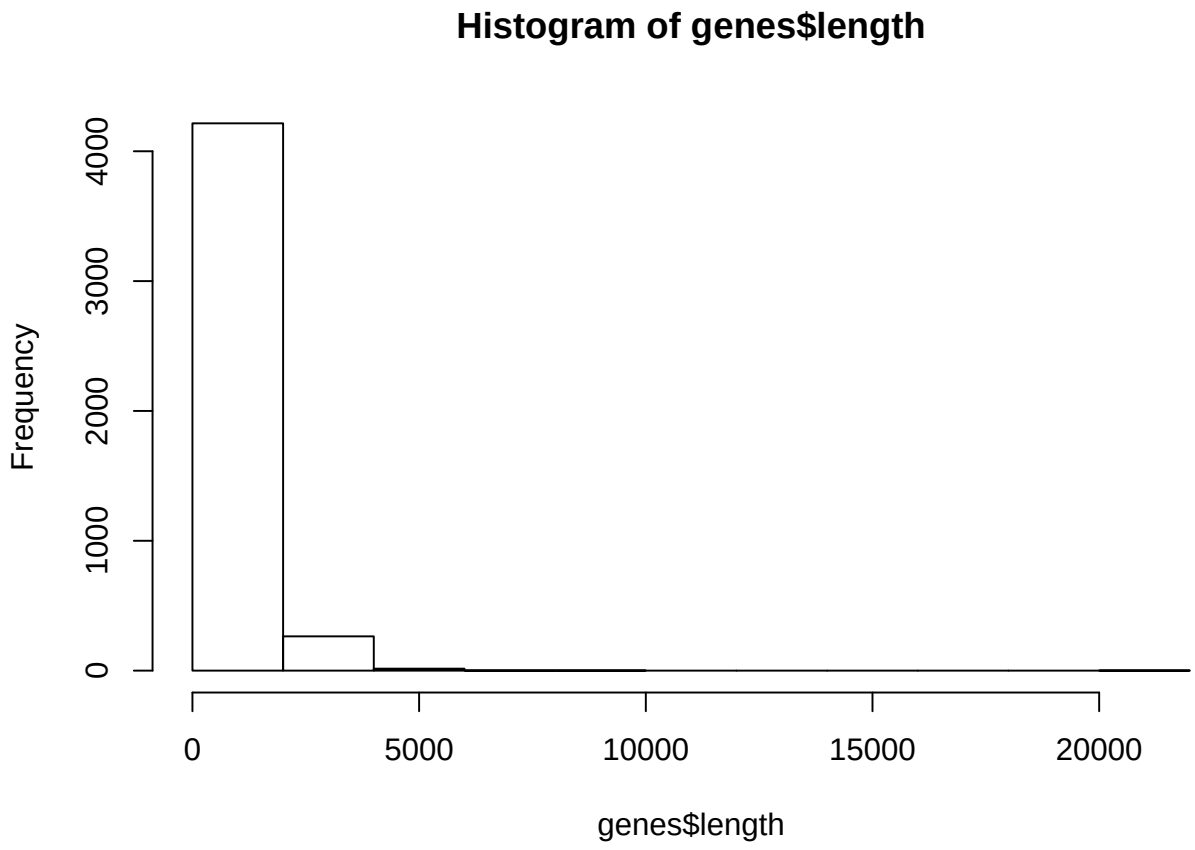
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   14.0   462.0   813.0   929.4  1221.0 21837.0
```

Exercices

1. Draw an histogram with gene length distribution. Choose a relevant number of breaks to display an informative histogram.
2. Draw a boxplot of gene lengths per strand. Are gene longer on the minus or plus strand?

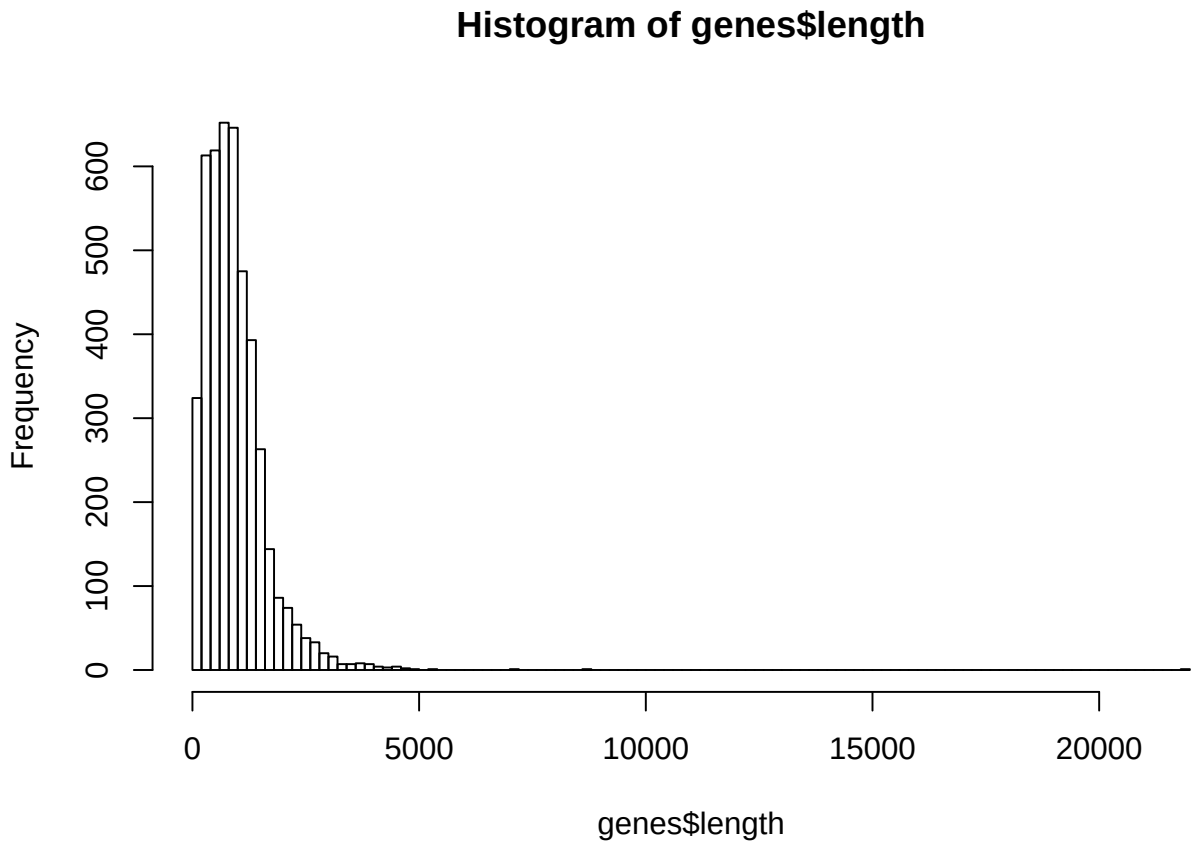
Gene length histogram

```
hist(genes$length)
```



Setting a relevant number of breaks

```
## Take more or less 100 bins
hist(genes$length, breaks = 100)
```



Gene length distribution – improving the output

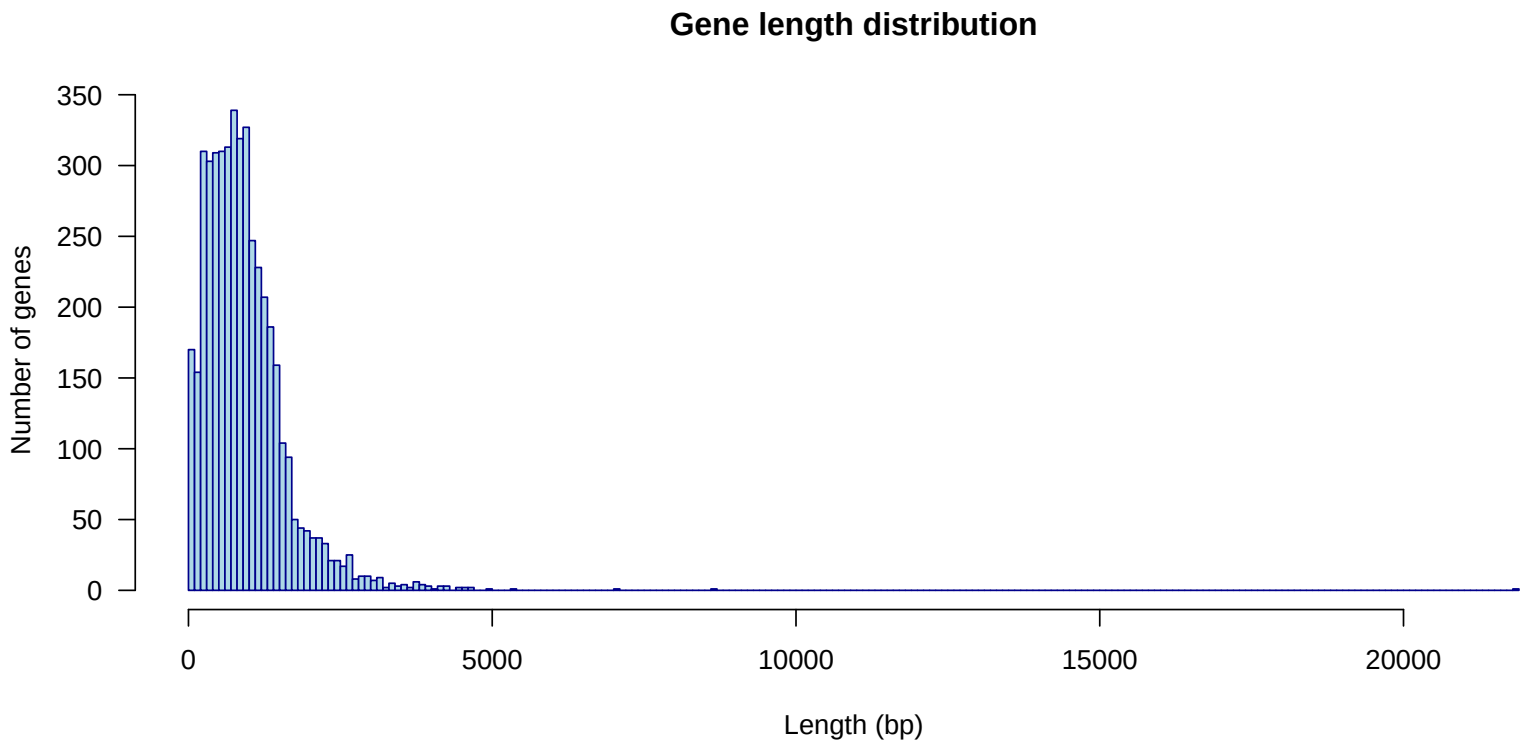


Figure 1: Distribution of gene lengths for E. Coli.

Gene length box plot

Other types of plots allow to explore the distribution of some data. In particular, boxplots display the median, the first and third quartiles and outlier values.

```
boxplot(length ~ strand, data = genes, col="palegreen", hor
        las=1, xlab="Gene length", ylab="Strand")
```