Exploration d'une table d'annotations génomiques (GTF)

Ecole de Bioinformatique Aviesan-IFB 2018

Jacques van Helden

2018-11-27

Contents

But de ce TP

Durant ce TP, vous serez amenés à effectuer les tâches suivantes:

1. Manipuler une table de données génomique (les annotations du génome de la levure).
2. Sélectionner un sous-ensemble des données en filtrant les lignes sur base d'un critère déterminé (type d'annotation, chromosome).
3. Générer des graphiques pour représenter différents aspects liés à ces données.
4. Calculer des statistiques qui résument les différents types d'annotations.

Le format GTF

Le format **GTF** (**General Transfer Format**) est très largement utilisé pour fournir des annotations génomiques dans un format facilement lisible, tout en étant facilement manipulable au moyen de l'ordinateur.

Fichiers textuels,

- une ligne par "objet" génomique (gène, transcrit, exon, intron, CDS, ...)
- une colonne par attribut (nom, source, type d'objet, coordonnées génomique, description).

Le format est décrit sur les sites suivants.

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

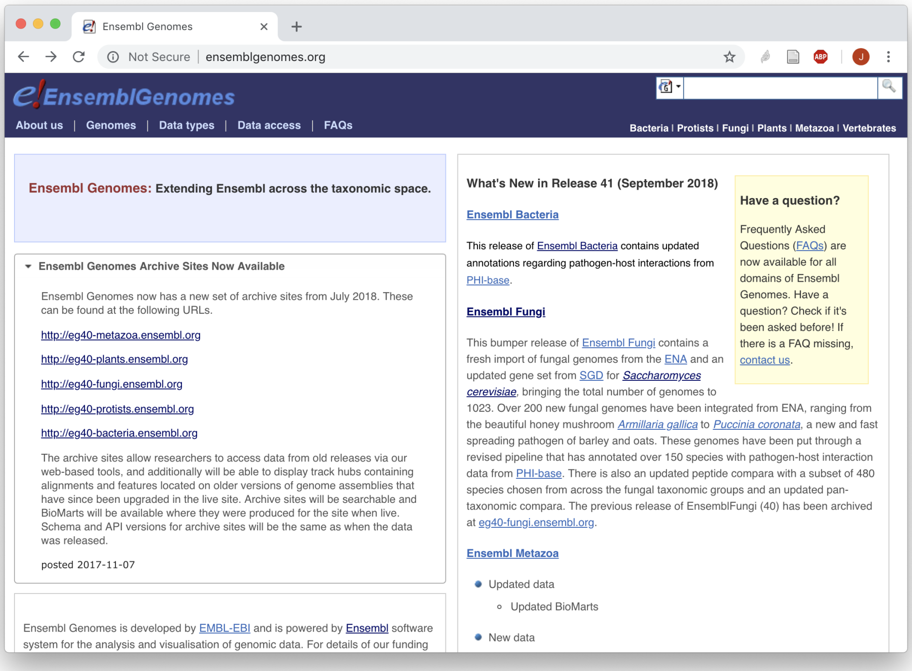
Localiser l'URL d'un fichier GTF

N'hésitez pas à adapter le protocole ci-dessous pour travailler avec votre propre génome.

1. Connectez-vous à http://ensemblgenomes.org/.
2. Cliquez sur le lien Fungi.
3. Cliquez Download
4. Dans la boîte **Filter**, tapez *Saccharomyces cerevisiae*. Pendant que vous écrivez, la liste des organismes proposés s'affine.
5. Copiez le lien du fichier gtf (Saccharomyces_cerevisiae.R64-1-1.41.gtf.gz).

Page d'accueil d'EnsemblGenomes

```
include_graphics(path = "images/ensemblgenomes_home.png")
```


Ensembl Genomes
Not Secure | ensemblgenomes.org
EnsemblGenomes
About us | Genomes | Data types | Data access | FAQs
Bacteria | Protists | Fungi | Plants | Metazoa | Vertebrates

Ensembl Genomes: Extending Ensembl across the taxonomic space.

Ensembl Genomes Archive Sites Now Available

Ensembl Genomes now has a new set of archive sites from July 2018. These can be found at the following URLs.

http://eg40-metazoa.ensembl.org

http://eg40-plants.ensembl.org

http://eg40-fungi.ensembl.org

http://eg40-protists.ensembl.org

http://eg40-bacteria.ensembl.org

The archive sites allow researchers to access data from old releases via our web-based tools, and additionally will be able to display track hubs containing alignments and features located on older versions of genome assemblies that have since been upgraded in the live site. Archive sites will be searchable and BioMarts will be available where they were produced for the site when live. Schema and API versions for archive sites will be the same as when the data was released.

posted 2017-11-07

Ensembl Genomes is developed by EMBL-EBI and is powered by Ensembl software system for the analysis and visualisation of genomic data. For details of our funding

What's New in Release 41 (September 2018)

Ensembl Bacteria

This release of Ensembl Bacteria contains updated annotations regarding pathogen-host interactions from PHI-base.

Ensembl Fungi

This bumper release of Ensembl Fungi contains a fresh import of fungal genomes from the ENA and an updated gene set from SGD for *Saccharomyces cerevisiae*, bringing the total number of genomes to 1023. Over 200 new fungal genomes have been integrated from ENA, ranging from the beautiful honey mushroom *Armillaria gallica* to *Puccinia coronata*, a new and fast spreading pathogen of barley and oats. These genomes have been put through a revised pipeline that has annotated over 150 species with pathogen-host interaction data from PHI-base. There is also an updated peptide compara with a subset of 480 species chosen from across the fungal taxonomic groups and an updated pan-taxonomic compara. The previous release of EnsemblFungi (40) has been archived at eg40-fungi.ensembl.org.

Ensembl Metazoa

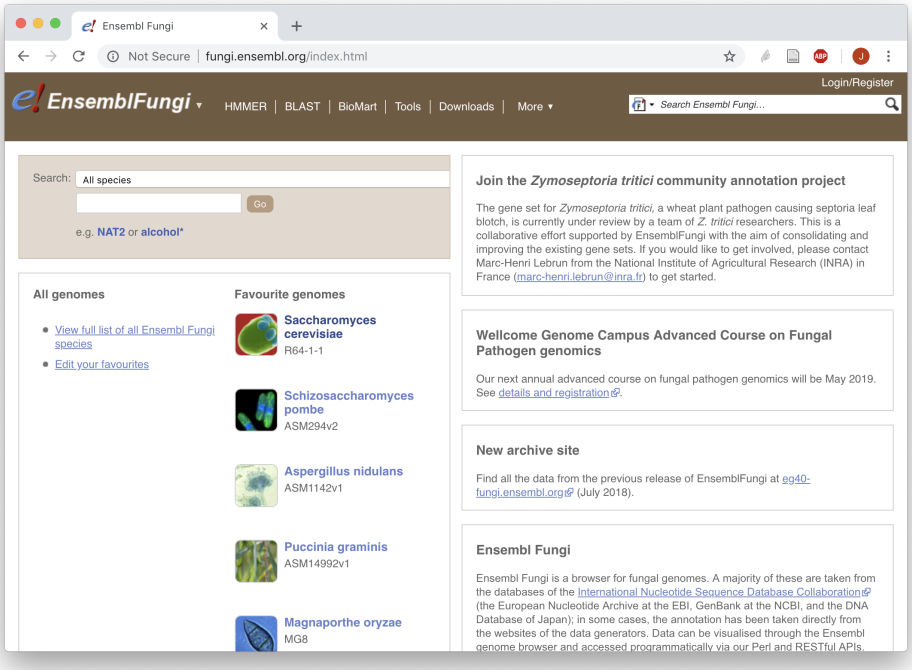
- Updated data
  - Updated BioMarts
- New data

Have a question?

Frequently Asked Questions (FAQs) are now available for all domains of Ensembl Genomes. Have a question? Check if it's been asked before! If there is a FAQ missing, contact us.

EnsemblGenomes Fungi

```
include_graphics(path = "images/ensemblgenomes_fungi.png")
```



All genomes

- View full list of all Ensembl Fungi species
- Edit your favourites

Favourite genomes

Saccharomyces cerevisiae
R64-1-1

Schizosaccharomyces pombe
ASM294v2

Aspergillus nidulans
ASM1142v1

Puccinia graminis
ASM14992v1

Magnaporthe oryzae
MG8

Join the *Zymoseptoria tritici* community annotation project

The gene set for *Zymoseptoria tritici,* a wheat plant pathogen causing septoria leaf blotch, is currently under review by a team of *Z. tritici* researchers. This is a collaborative effort supported by EnsemblFungi with the aim of consolidating and improving the existing gene sets. If you would like to get involved, please contact Marc-Henri Lebrun from the National Institute of Agricultural Research (INRA) in France (marc-henri.lebrun@inra.fr) to get started.

Wellcome Genome Campus Advanced Course on Fungal Pathogen genomics

Our next annual advanced course on fungal pathogen genomics will be May 2019. See details and registration.

New archive site

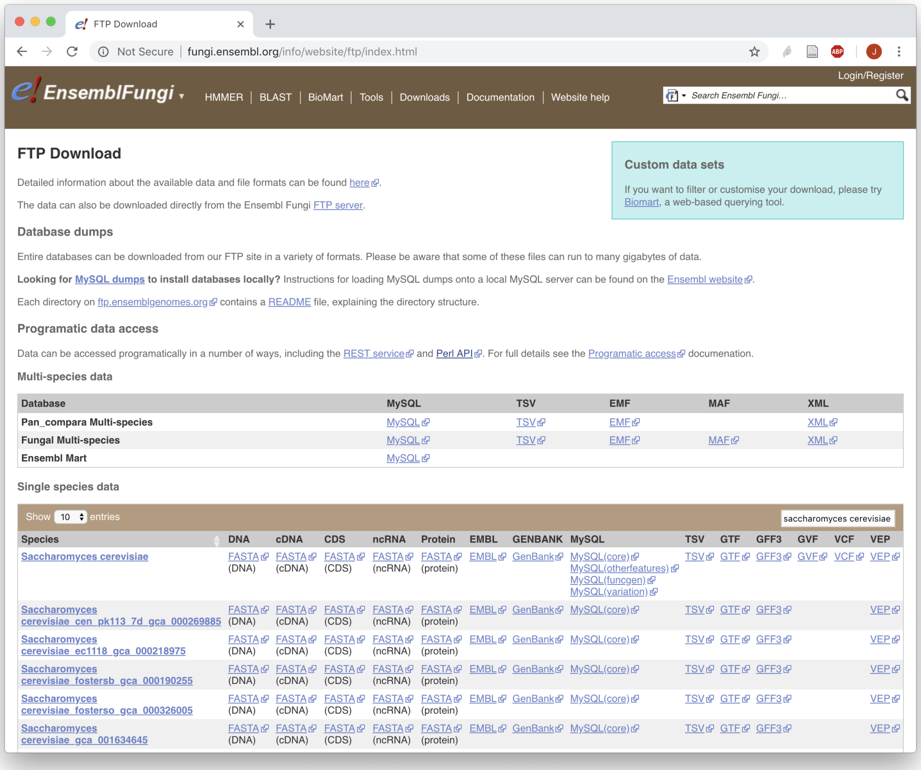
Find all the data from the previous release of EnsemblFungi at eg40-fungi.ensembl.org (July 2018).

Ensembl Fungi

Ensembl Fungi is a browser for fungal genomes. A majority of these are taken from the databases of the International Nucleotide Sequence Database Collaboration (the European Nucleotide Archive at the EBI, GenBank at the NCBI, and the DNA Database of Japan); in some cases, the annotation has been taken directly from the websites of the data generators. Data can be visualised through the Ensembl genome browser and accessed programmatically via our Perl and RESTful APIs.

EnsemblGenomes Download page

```
include_graphics(path = "images/ensemblgenomes_download_yeast.png")
```


FTP Download

fungi.ensembl.org/info/website/ftp/index.html

EnsemblFungi | HMMER | BLAST | BioMart | Tools | Downloads | Documentation | Website help | Login/Register | Search Ensembl Fungi…

FTP Download

Detailed information about the available data and file formats can be found here.

The data can also be downloaded directly from the Ensembl Fungi FTP server.

Custom data sets

If you want to filter or customise your download, please try Biomart, a web-based querying tool.

Database dumps

Entire databases can be downloaded from our FTP site in a variety of formats. Please be aware that some of these files can run to many gigabytes of data.

Looking for MySQL dumps to install databases locally? Instructions for loading MySQL dumps onto a local MySQL server can be found on the Ensembl website.

Each directory on ftp.ensemblgenomes.org contains a README file, explaining the directory structure.

Programatic data access

Data can be accessed programatically in a number of ways, including the REST service and Perl API. For full details see the Programatic access documenation.

Multi-species data

Database	MySQL	TSV	EMF	MAF	XML
Pan_compara Multi-species	MySQL	TSV	EMF		XML
Fungal Multi-species	MySQL	TSV	EMF	MAF	XML
Ensembl Mart	MySQL				

Single species data

Show 10 entries — saccharomyces cerevisiae

Species	DNA	cDNA	CDS	ncRNA	Protein	EMBL	GENBANK	MySQL	TSV	GTF	GFF3	GVF	VCF	VEP
Saccharomyces cerevisiae	FASTA (DNA)	FASTA (cDNA)	FASTA (CDS)	FASTA (ncRNA)	FASTA (protein)	EMBL	GenBank	MySQL(core) MySQL(otherfeatures) MySQL(funcgen) MySQL(variation)	TSV	GTF	GFF3	GVF	VCF	VEP
Saccharomyces cerevisiae_cen_pk113_7d_gca_000269885	FASTA (DNA)	FASTA (cDNA)	FASTA (CDS)	FASTA (ncRNA)	FASTA (protein)	EMBL	GenBank	MySQL(core)	TSV	GTF	GFF3			VEP
Saccharomyces cerevisiae_ec1118_gca_000218975	FASTA (DNA)	FASTA (cDNA)	FASTA (CDS)	FASTA (ncRNA)	FASTA (protein)	EMBL	GenBank	MySQL(core)	TSV	GTF	GFF3			VEP
Saccharomyces cerevisiae_fostersb_gca_000190255	FASTA (DNA)	FASTA (cDNA)	FASTA (CDS)	FASTA (ncRNA)	FASTA (protein)	EMBL	GenBank	MySQL(core)	TSV	GTF	GFF3			VEP
Saccharomyces cerevisiae_fosterso_gca_000326005	FASTA (DNA)	FASTA (cDNA)	FASTA (CDS)	FASTA (ncRNA)	FASTA (protein)	EMBL	GenBank	MySQL(core)	TSV	GTF	GFF3			VEP
Saccharomyces cerevisiae_gca_001634645	FASTA (DNA)	FASTA (cDNA)	FASTA (CDS)	FASTA (ncRNA)	FASTA (protein)	EMBL	GenBank	MySQL(core)	TSV	GTF	GFF3			VEP

Le chemin de la maison (automatique)

Sous Linux et Mac OS X, on peut identifier la racine de son compte avec la commande **R** `Sys.getenv()`.

- Invoquée sans paramètre, cette commande liste toutes les variables d'environnement (votre configuration système).
- On peut restreindre l'output à une variable d'environnement donnée, par exemple `Sys.getenv("HOME")` retourne le chemin de la racine de votre compte.
- Une écriture équivalente : le symbole tilde ~ indique également le chemin de la rachine de votre compte.
- La notation '~' fonctionne également sous Windows, nous l'utiliserons donc ci-dessous.

Créer un espace de travail

Exercice: créer un dossier de travail nommé `workDir` à la racine de votre compte, et déplacez-vous dans ce dossier.

Solution ci-dessous.

```r
## 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
```

```
[1] "/Users/jvanheld/intro_R/explorer_un_GTF"
```

```
list.files()  ## List files (should be empty if just created)
```

```
[1] "chrom_sizes.tsv"
[2] "Saccharomyces_cerevisiae.R64-1-1.41.gtf.gz"
```

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 rowking directory. If yes, we skip the download.

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

Downloading the GTF file: solution

```
## Define the file name (without path) in a separate variable, we will need it later
gtf.file <- 'Saccharomyces_cerevisiae.R64-1-1.41.gtf.gz'

## Define the URL by concatenating the URL of the directory and the file name
gtf.url <- file.path('ftp://ftp.ensemblgenomes.org/pub/release-41/fungi/gtf/saccharomyces_cerevisiae/',

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

## Check the files in the work directory
list.files()
```

```
[1] "data"
[2] "figures"
[3] "gtf_exploration_files"
[4] "gtf_exploration.html"
[5] "gtf_exploration.md"
[6] "gtf_exploration.pdf"
[7] "gtf_exploration.Rmd"
[8] "images"
[9] "Saccharomyces_cerevisiae.R64-1-1.41.gtf.gz"
```

Loading a data table in R

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

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” 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(
  gtf.file, comment.char = "#", sep="\t",
  header=FALSE, row.names = NULL)

## The GTF format has no header, but we can define it based on the specification
names(featureTable) <- c("seqname", "source", "feature", "start", "end", "score", "strand", "frame", "a
```

Exploring the content of a data table

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

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

```
[1] 41606     9
```

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

```
[1] 41606
```

```
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 frame
1      IV    sgd        gene  1802 2953     .      +     .
2      IV    sgd  transcript  1802 2953     .      +     .
3      IV    sgd        exon  1802 2953     .      +     .
4      IV    sgd         CDS  1802 2950     .      +     0
5      IV    sgd start_codon  1802 1804     .      +     0
```

```
## Equivalent: selecting subsets of rows and columns
featureTable[1:5, 1:8]
```

```
  seqname source     feature start  end score strand frame
1      IV    sgd        gene  1802 2953     .      +     .
2      IV    sgd  transcript  1802 2953     .      +     .
3      IV    sgd        exon  1802 2953     .      +     .
4      IV    sgd         CDS  1802 2950     .      +     0
5      IV    sgd start_codon  1802 1804     .      +     0
```

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
[5] start_codon    stop_codon     five_prime_utr
7 Levels: CDS exon five_prime_utr gene start_codon ... transcript
```

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

```

          exon           gene     transcript            CDS    start_codon
          7416           7036           7036           6913           6601
    stop_codon five_prime_utr
          6600              4
```

Décompte par valeur

La fonction `table()` permet de compter le nombre d'occurrences de chaque valeur dans un vecteur ou un tableau. Quelques exemples d'utilisation ci-dessous.

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

```
   I   II  III   IV   IX Mito    V   VI  VII VIII    X   XI  XII XIII  XIV 
 731 2841 1170 5185 1520  312 2055  898 3688 2012 2511 2180 3690 3196 2712 
  XV  XVI 
3706 3199 
```

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

```

           CDS            exon five_prime_utr            gene     start_codon 
          6913            7416               4            7036            6601 
    stop_codon      transcript 
          6600            7036 
```

On peut calculer des tables de contingence en comptant le nombre de combinaisons entre 2 vecteurs (ou 2 colonnes d'un tableau).

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

```

                   I  II III  IV  IX Mito   V  VI VII VIII   X  XI XII
  CDS            120 483 192 870 252   59 338 146 605  340 412 361 604
  exon           129 500 210 907 269   87 367 166 659  358 446 385 668
  five_prime_utr   0   0   0   0   1    0   0   0   0    0   1   0   0
  gene           124 472 200 868 258   55 352 154 629  336 428 369 631
  start_codon    117 458 184 836 241   28 323 139 583  321 398 348 578
  stop_codon     117 456 184 836 241   28 323 139 583  321 398 348 578
  transcript     124 472 200 868 258   55 352 154 629  336 428 369 631
                
                 XIII XIV  XV XVI
  CDS             531 454 609 537
  exon            573 478 647 567
  five_prime_utr    1   0   1   0
  gene            541 455 628 536
  start_codon     504 435 596 512
  stop_codon      505 435 597 511
  transcript      541 455 628 536
```

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

```
                seqname
feature            I  II III  IV  IX Mito   V  VI VII VIII   X  XI XII
  CDS            120 483 192 870 252   59 338 146 605  340 412 361 604
  exon           129 500 210 907 269   87 367 166 659  358 446 385 668
  five_prime_utr   0   0   0   0   1    0   0   0   0    0   1   0   0
  gene           124 472 200 868 258   55 352 154 629  336 428 369 631
  start_codon    117 458 184 836 241   28 323 139 583  321 398 348 578
  stop_codon     117 456 184 836 241   28 323 139 583  321 398 348 578
  transcript     124 472 200 868 258   55 352 154 629  336 428 369 631
                seqname
feature          XIII XIV  XV XVI
  CDS             531 454 609 537
  exon            573 478 647 567
  five_prime_utr    1   0   1   0
  gene            541 455 628 536
```

```
start_codon     504 435 596 512
stop_codon      505 435 597 511
transcript      541 455 628 536
```

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$start + 1
```

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" attribute
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. 
     51     468    1005    1275    1717   14733 
```

Downloading chromosome sizes

- Download chromosome sizes (chrom_sizes.tsv)

```
## Download tab-delimited file with chromosome sizes (unless already there)
chromsizes.url <- "https://github.com/jvanheld/stats_avec_RStudio_EBA/blob/gh-pages/practicals/gtf_expl
chrom.size.file <- file.path(workDir, "chrom_sizes.tsv")

if (file.exists(chrom.size.file)) {
} else {
    download.file(chromsizes.url, destfile = chrom.size.file)
}
```

```
NULL
```

Loading chromosome sizes

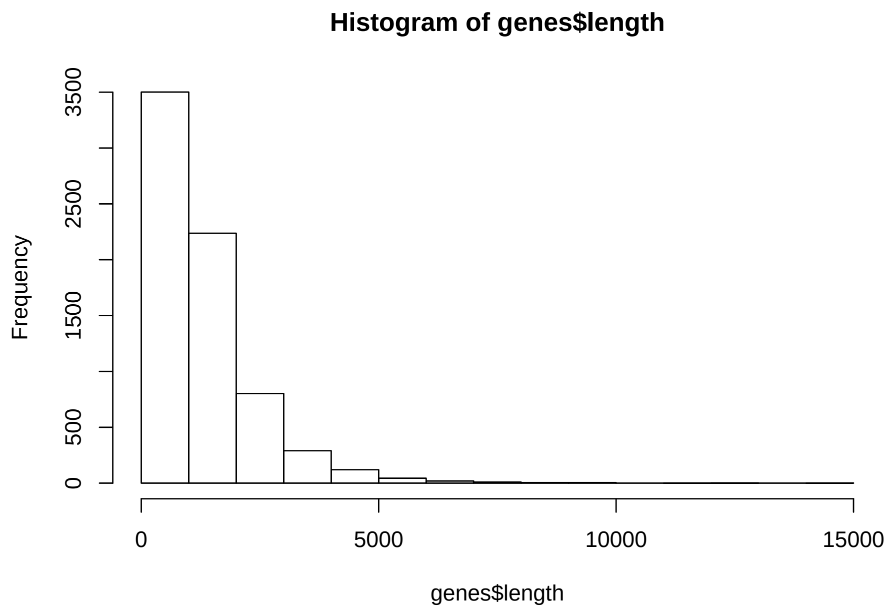
```
## Read chromosome sizes
chrom.size <- read.delim(
  file = chrom.size.file,
  header = FALSE, row.names = 1)

## Assign a name to the columns
names(chrom.size) <- c("chromID", "size")
# View(chrom.size)

## print the size of hte third chromosome
message("Length of chromosome III = ", chrom.size["III", "size"], " bp.")
```

Exercices

1. Draw an histogram with gene length distribution. Choose a relevan number of breaks to display an informative histogram.
2. Draw a barplot showing gene density per chromosome (number of genes per Mb).
3. Draw a boxplot of gene lengths per chromosome.

Gene length histogram

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



Setting a relevant number of breaks

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

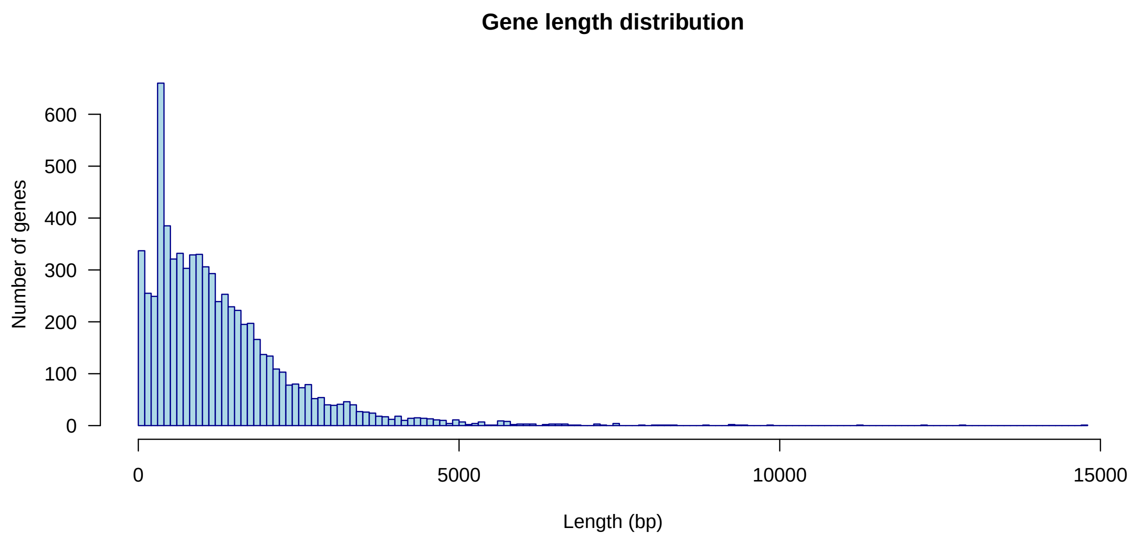
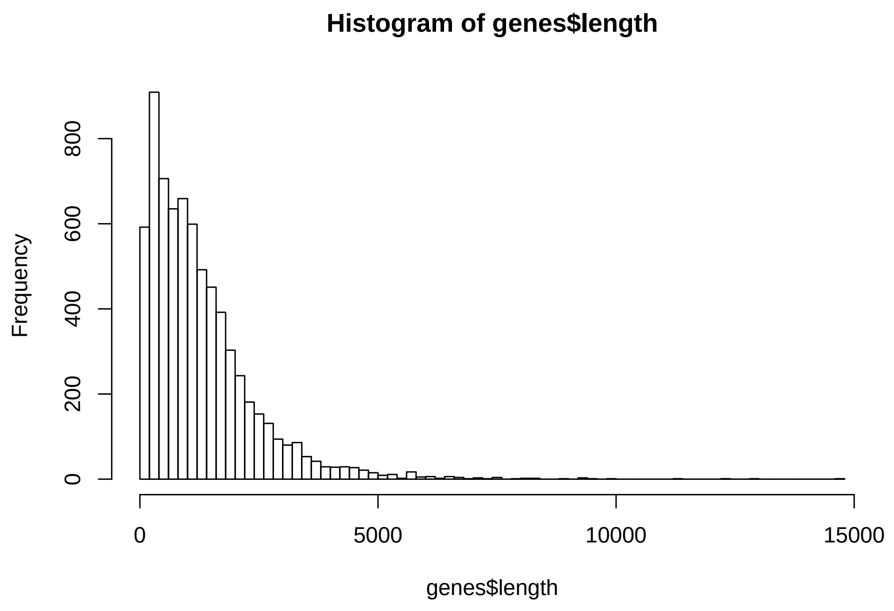


Figure 1: Distribution of cds lengths for Saccharomyces cerevisiae.

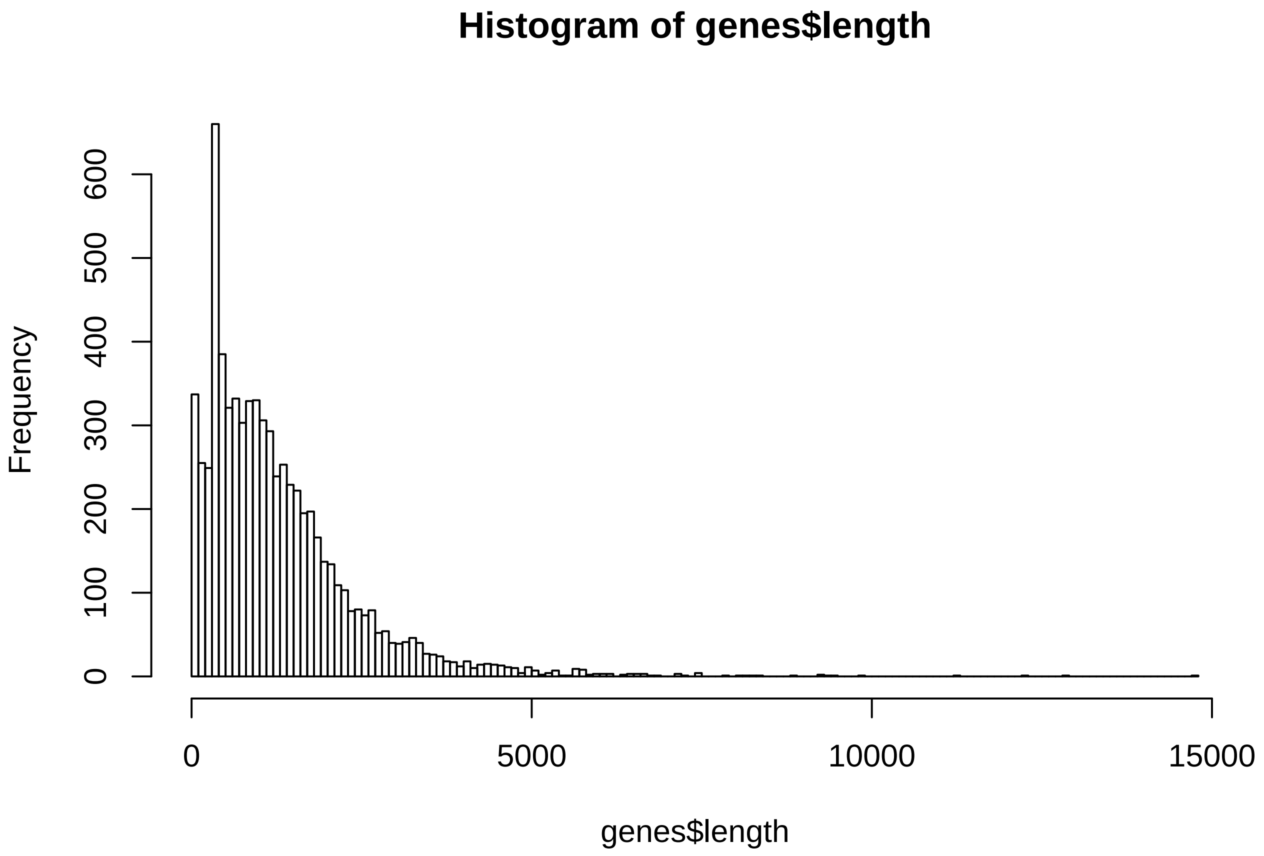


Gene length distribution – improving the output

Getting the hist() data

Récupérez le résultat de `hist()` dans une variable nommée `{histData}`.

```
## Define breaks exactly in the way you wish
histData <- hist(genes$length, breaks=seq(from=0, to=max(genes$length)+100, by=100))
```



Imprimez le résultat à l'écran (`print()`) et analysez la structure de la variable `histData` (il s'agit d'une variable de type liste).

Fonctions utiles:

- `class(histData)`
- `attributes(histData)`
- `print(histData)`

```
## Display the values used to draw the histogram
print(histData )
```

Gene length box plot

D'autres types de graphiques permettent d'explorer la distribution d'un ensemble des données. En particulier, les boîtes à moustaches (box plots) affichent, pour une série de données, la médiane, l'écart interquartile, un intervalle de confiance et les valeurs aberrantes.

```
boxplot(length ~ seqname, data = genes, col="palegreen", horizontal=TRUE, las=1, xlab="Gene length", yl
```

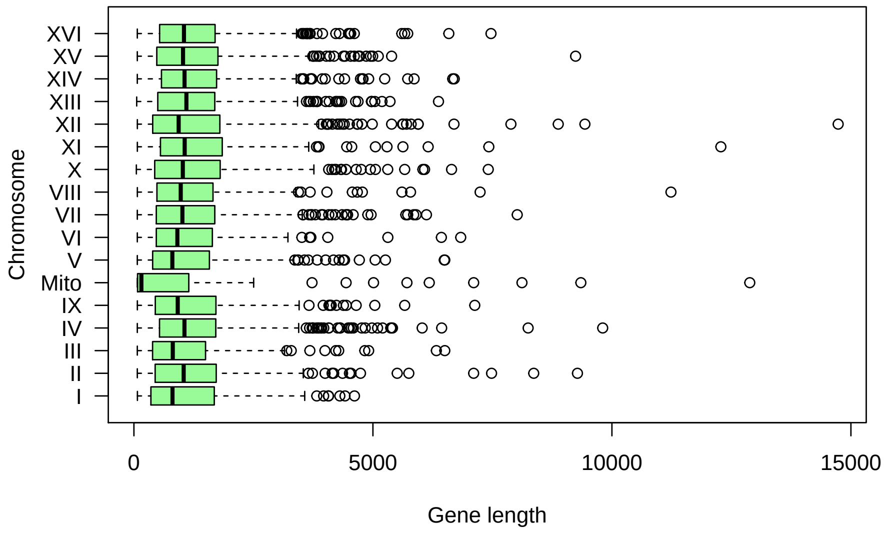


Figure 2: Boxplot of gene lengths per chromosome