BScBI-CG Practicals Report

Maria Cobo

Exercise 1

— October 10, 2024—

ESCI upf. School of International Studies

Computational Genomics 2024–25

BSc in Bioinformatics

UPF/ESCI – UPC – UB – UAB

Contents

List of Tables

List of Figures

1 Introduction

We are going to reuse an existing `MarkDown` report template, which for this practical exercise describes a basic sequence analysis protocol to explore a fly transcriptome dataset. Some parameters will be estimated from the set of sequences provided. Following the same protocol, you can provide a similar analysis for another species trancriptome, and then compare the results on the discussion section (see page 17).

1.1 Objectives

- We will practice again how to report the commands and the results required to solve the practical exercises, using a text file, this report of course, and the `MarkDown` syntax. Some enhancements using LaTeX will be introduced at every practical session.
- The main goal though is to reproduce a simple sequence analysis protocol, on the provided species transcriptome, and then extend to a second species transcriptome. This will make possible to discuss results separately but also to compare between each set.
- We will install and use basic programs from the Bioinformatics software suite `EMBOSS`, to calculate for instace GC content from the command-line interface.

1.2 Prerequisites

1.2.1 Installing required software

As for the previous practical, we must ensure that at least `pandoc` and `pdflatex` commands are running smoothly over our report files. You probably may need to install the following packages before working on anything else, unless you got them running from the previous exercise of course. We will need to install the `EMBOSS` suite, but those instructions are described on the data retrieval section (see section 5 on page 5). Just consider the different package manager tools that are available on each system distribution:

```
##################################
# Examples of package managers installing "emboss"
# on different operative systems, use only the ones suited for your operative system...

## For Linux distributions:

# on a debian/ubuntu/mint linux system (DEBs)
apt-cache search emboss     # to check if there is such a package
sudo apt-get install emboss # to install such a package

# on a redhat/fedora/centos linux system (RPMs)
yum search emboss           # to check if there is such a package
su -c 'yum install emboss'

# on a SUSE/openSuse  linux system
zypper search "emboss"
sudo zypper install emboss

## For MacOS distributions:

# on a Mac system using homebrew packages (**recommended option on a Mac**,
# see tutorial on the course introduction section materials at virtual campus)
brew search emboss
# check the above command output, i.e. "brewsci/bio/emboss", to use on install:
sudo brew install brewsci/bio/emboss

# on a Mac system using anaconda packages (https://conda.io/docs/index.html)
conda search emboss
# check the above command output to use on install:
sudo conda install -c bioconda emboss

# on a Mac system using mac ports (https://guide.macports.org/)
port search emboss
```

```
# check the above command output to use on install:
sudo port install emboss

## IMPORTANT ## Do not mess your Mac system using all
#               of the previous three install options, use the one
#               already available on your system or install "homebrew".

## Other:

# you can also install the package if available for the CygWin environment
# running on a Windows box (hhtp://www.cygwin.com/)

# add your packaging system here if you have not used any of the above commands...

# ...

#
###################################
```

1.2.1.1 Linux users: You must submit an exercise report for each practical as two single files, a `MarkDown` text file and a `PDF` compiled from that `MarkDown` file. In order to run such procedure, we must ensure that we have the software tools and the corresponding dependencies. This has to be done for the first exercise only, as you will have those tools already available for future compilations of the other exercises. The command below will install those tools:

```
sudo apt-get install pandoc \
                     texlive-latex-recommended \
                     texlive-latex-extra \
                     texlive-fonts-recommended

# getting the latest version from repository
#     https://github.com/jgm/pandoc/releases/latest
# for a Debian-based linux distribution you can run those two commands:
wget https://github.com/jgm/pandoc/releases/download/3.1.8/pandoc-3.4-1-amd64.deb
sudo dpkg -i pandoc-3.4-1-amd64.deb
```

When using `pandoc` version greater than `2.x` we will be able to apply further macros and tags on our report file (like embed `LaTeX` blocks). In case you want to play with LaTeX, I will recommend you to install the complete set with this command:

```
sudo apt-get install texlive-full \
                     texlive-fonts-recommended \
                     texlive-fonts-extra
```

You can also install optional packages, such a text editor with programming facilities and extensions, like `emacs` or `geany` (you can also use `sublime`, `atom`, `gedit`, ...):

```
sudo apt-get install emacs geany vim vim-gtk
```

Further instructions will be given on the templates in case a practical requires that you install further software...

1.2.1.2 MacOS users: MacOS users can try to install ports for the software tools from `homebrew` or `conda` repositories. Here we have a brief summary of such commands. `Homebrew` is a package manager for **MacOS**; it also has a port for **Linux**, known as `Linuxbrew`. To install this package manager, just copy the following command and paste it to a Terminal window:

```
# setting up brew tool and its repositories
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

A list of all the available packages (known as *formulae*) is found at formulae.brew.sh. Examples of `brew` command to install the `EMBOSS` or the `pandoc` are shown here:

```
# Getting EMBOSS installed with brew
brew search emboss
# homebrew/science/emboss  #<-- check the output of the previous command to use in your system
brew install homebrew/science/emboss

# Getting pandoc installed with brew
brew install pandoc
# this is optional
brew install pandoc-citeproc
# you need this to typeset PDFs with LaTeX
brew install librsvg python homebrew/cask/basictex

# You can also download the pandoc MacOS instaler from the following URL:
# https://github.com/jgm/pandoc/releases/download/3.4/pandoc-3.4-x86_64-macOS.pkg
```

1.2.1.3 Using `conda/mamba` environments: As we saw in the previous exercise, another way to install the software required to complete the exercises is to use `conda` environments. You can install `conda` following the instructions from this link; you can also use `mamba` instead, which is a compact and faster implementation of `conda`, from the instructions at this link. Once you have one of those environment managers installed, you can follow the commands in the next code block to create the `BScBI-CG2425_exercises` environment and activate it. **You probably have the conda environment created from the previous exercise, then you can jump to the next block of code.**

```
#
# ***Important***: ensure that you run the create command
#                  outside any other environment (even the `base` one),
#                  for a fresh install of the proper dependencies.
#
# If you have conda instead of mamba already installed on your system
# you can just replace 'mamba' by 'conda' on the commands below:
mamba env create --file environment.yml

# Now you can run the tools installed on that environment by activating it:
mamba activate BScBI-CG2425_exercises

# Remember that each time you deactivate a conda environment
# all shell variables defined inside will be lost
# (unless they were exported before activating the conda environment).
# Anyway, you can reload project vars with:
source projectvars.sh

# To return to the initial terminal state, you must deactivate the environment:
mamba deactivate
```

IMPORTANT: For this exercise we only need to update our environment, in order to include the tools introduced to complete current the protocol (basically adding `emboss` suite to the current environment). The `environment.yml` file included in the exercise tarball is the same as that of `exercise_00`, including an extra dependency line.

```
#
# ***Important***: ensure that you run the update command
#                  outside any mamba/conda environment too.
#
# Again, if you have conda instead of mamba already installed on your system
# you can just replace 'mamba' by 'conda' on the commands below:
mamba env update --file environment.yml

# Now you can run the tools installed on that environment by activating it:
mamba activate BScBI-CG2425_exercises

# Remember that each time you deactivate a conda environment
# all shell variables defined inside will be lost
# (unless they were exported before activating the conda environment).
# Anyway, you can reload project vars with:
```

```
source projectvars.sh

# To return to the initial terminal state, you must deactivate the environment:
mamba deactivate
```

You can review the contents of the environment YAML file at the Appendices (see section 4.2.1 on page 18),

1.2.2 Initializing the main report files

In a `bash` command-line terminal, create a folder for all the practicals on this subject and change working dir to that folder:

```
mkdir practicals
cd practicals
```

Then, we need to download the exercise tarball (the `*.tgz` file) from the Computational Genomics Virtual Campus at ESCI into that folder, unpack such file, modify the files accordingly to the user within the exercise folder, and set it as the current working directory for the rest of the practical session...

```
# Uncompress and unpack the exercise files from tarball
#
# NOTE: If you are reading this, you probably have already done this step.
#
tar -zxvf BScBI_CG2425_exercise_01.tgz

# Move into the new extracted folder
cd exercise_01

# Rename the MarkDown README_*.md file by replacing NAME and SURNAME strings
# with your "NAME" and "SURNAME" with the following command
mv -v README_BScBICG2425_exercise01_SURNAME_NAME.md \
      README_BScBICG2425_exercise01_YourSurname_YourName.md

# Open exercise files using your text editor of choice
# (for instance vim, emacs, gedit, sublime, atom, ...);
emacs  projectvars.sh \
       README_BScBICG2425_exercise01_YourSurname_YourName.md

# Fix "NAME" and "SURNAME" placeholders on those files
# and save the changes before continuing.

# Load the bash definitions from projectvars.sh
source projectvars.sh
# for instance the variable WDR was set to the absolute path
# to current exercise working directory
echo $WDR

# Now you are ready to start the practical by looking at
# the MarkDown file for further instructions to run the corresponding code blocks.

# Each time you include your answers/code/results on the README file, you can compile it into PDF.
# So that, let's tests if we can compile the modified MarkDown document.
# You probably must install some dependencies yet...
runpandoc
```

Again, remember to submit both files, the MD and the PDF, to the Computational Genomics Virtual Campus at ESCI, once errors/warnings have been fixed, all the requested task have been completed, and you have discussed your results on the corresponding sections.

If you have succeeded on the software installation step, then you can start with the analyses provided on the next section... May the shell be with you...

2 Genomic Analyses on Command-line

2.1 Datasets

We have to analyze sequence length distribution and GC content for the current mRNA sequences annotated on *Drosophila melanogaster* genome (BDGP Release `6 + ISO1 MT/dm6`, Aug. 2014). We have connected to the UCSC genome browser download web site, and followed the "Genome sequence files and select annotations (2bit, GTF, GC-content, etc)" link. We can see that there is a file named `mrna.fa.gz`, we can just copy the corresponding link to the following command:

```
export DT=$WDR/data
mkdir -v $DT
# You can also add the previous var definition to your 'projectvars.sh' file
# so it will be saved and can be easily reused when sourcing the file again.

wget https://hgdownload.soe.ucsc.edu/goldenPath/dm6/bigZips/mrna.fa.gz \
     -O $DT/dmel_mrna.fa.gz
```

We obtained a compressed file containing all the mRNA sequences annotated for this version of the fly genome in `fasta` format. We can check it's content:

```
zcat $DT/dmel_mrna.fa.gz | head -10
# >DQ327735 1
# ttgttgcgcacacgcaccagaagagaggaggatcgaccaggcagctgttc
# tctggctctctggaaagtggtcaaaggagaaggaggaggtcgtaagagcg
# gtagaatcgacaatataatcggagtcatatcgggcatcaacgtcggccac
# atcaacatcaacaacggcagcagacgtcgctaattgcaaccaacacggga
# gctgcagcctggaccctacatatccaatgttcagaatttaaatgcaccac
# agcagcaacatcgaggtcctcagcagcagcatcagacggccggttgcctc
# aaagttccttagcctgcgtggcacgtggcatccatcagcgcctcaggctg
# aataaaaggaaatcccagccataaaaagtaaatcacttcggcgccaacca
# gttccagtcatcatggctcactccaaggtgatccttaagatcctgttttc
```

Let's see how many mRNA sequences do we have. We can uncompress the file and keep going with the flat text file, which can take a lot of disk resources, or we can keep using the compressed file as in the previous command.

```
zcat $DT/dmel_mrna.fa.gz | egrep -c '^>'
# 75218
```

In case that a program can deal with compressed files, we can take adantage of that feature. There is an `egrep` version that can read such files, if you were guessing is `zegrep` of course (as it happens with `cat` and `zcat`).

```
zegrep -c '^>' $DT/dmel_mrna.fa.gz
# 75218
```

2.2 Simple analysis of mRNA sequences

To calculate the nucleotide length and the GC content, we can use one of the programs that is provided within the EMBOSS suite: `infoseq`. You can get information about this tool from this link.

```
# You get info about this tool command-line options
infoseq -help
# If 'emboss-doc' has been installed youcan also use for further details
man infoseq

# Let's run it on our dataset
zcat $DT/dmel_mrna.fa.gz | \
  infoseq -sequence fasta::stdin \
          -noheading -only -name -length -pgc | \
    head -5
```

```
# Display basic information about sequences
# DQ327735       1883   53.48
# DQ327736       272    55.15
# DQ327737       338    60.36
# DQ327738       1294   50.85
# DQ327739       758    50.92
```

In the above command we are just looking to the expected output, the following is doing the job and saving the output into `dmel_mrna.lengc.tbl` file.

```
zcat $DT/dmel_mrna.fa.gz | \
  infoseq -sequence fasta::stdin \
          -outfile $DT/dmel_mrna.lengc.tbl \
          -noheading -only -name -length -pgc
```

However, it is advisable to work as much as possible with compressed files, so the following commands could be also useful.

```
zcat $DT/dmel_mrna.fa.gz | \
  infoseq -sequence fasta::stdin \
          -noheading -only -name -length -pgc | \
    gzip -vc9 - > $DT/dmel_mrna.lengc.tbl.gz
```

2.3 Visualizing the analysis

By running `R` command, we enter in the `R` shell interpreter, which understands `R` commands of course.

```
R
#
# R version 4.1.2 (2021-11-01) -- "Bird Hippie"
# Copyright (C) 2021 The R Foundation for Statistical Computing
# Platform: x86_64-pc-linux-gnu (64-bit)
#
# R is free software and comes with ABSOLUTELY NO WARRANTY.
# You are welcome to redistribute it under certain conditions.
# Type 'license()' or 'licence()' for distribution details.
#
# R is a collaborative project with many contributors.
# Type 'contributors()' for more information and
# 'citation()' on how to cite R or R packages in publications.
#
# Type 'demo()' for some demos, 'help()' for on-line help, or
# 'help.start()' for an HTML browser interface to help.
# Type 'q()' to quit R.
#
```

Now, we must load the tabular data into a variable.

```
# if we have an uncompresed tabular file
DATA <- read.table("data/dmel_mrna.lengc.tbl", header=FALSE);
# otherwise you can run this command
ZZ <- gzfile('data/dmel_mrna.lengc.tbl.gz');
DATA <- read.table(ZZ, header=FALSE);

# just checking the data structure
head(DATA,4);
#         V1   V2    V3
# 1 DQ327735 1883 53.48
# 2 DQ327736  272 55.15
# 3 DQ327737  338 60.36
# 4 DQ327738 1294 50.85
```

We can rename the table columns, so they are more meaningful:

```
colnames(DATA) <- c("ID","NUClen","GCpct");
head(DATA,4);
#         ID NUClen GCpct
# 1 DQ327735   1883 53.48
# 2 DQ327736    272 55.15
# 3 DQ327737    338 60.36
# 4 DQ327738   1294 50.85
```

Let's calculate some stats on the dataset.

```
summary(DATA)
#      ID                NUClen            GCpct
# Length:75218       Min.   :   16.0   Min.   :11.43
# Class :character   1st Qu.:  713.2   1st Qu.:47.81
# Mode  :character   Median : 1262.0   Median :51.62
#                    Mean   : 1587.5   Mean   :50.91
#                    3rd Qu.: 2073.0   3rd Qu.:54.82
#                    Max.   :19444.0   Max.   :72.73
```

Now, let's make an histogram:

```
png(file="images/dmel_hist_length.png");
hist(DATA$NUClen);
dev.off();
png(file="images/dmel_hist_pgc.png");
hist(DATA$GCpct);
dev.off();
```

Or to compare both measures and save into a PNG image...

```
png(file="images/dmel_plot.png");
plot(DATA$NUClen ~ DATA$GCpct);
dev.off();
```

Just take care that this is not a single-variable distribution but a two axis scatter-plot, each measure is a continuous random variable that may fit (or not) a normal distribution. The real data distribution is shown on the histograms, we can merge then those three plots... Before doing this, what about getting an improved integration of images in the report by using a LaTeX floating environment. Compare result with previous image after compilation, you can choose to fix that report line.

```
png(file="images/dmel_plot2.png");
def.par <- par();
# preparing a layout grid where to combine different plots
nf <- layout(matrix(c(2,0,1,3), # matrix contents is plot order
                    2, 2, byrow=TRUE),
             c(3,1), c(1,3), # cell relative sizes
             TRUE);
# computing data distribution
xhist <- hist(DATA$GCpct, plot=FALSE);
yhist <- hist(DATA$NUClen, plot=FALSE);
datamax <- max(xhist$counts,
               xhist$counts);
#
# drawing the main plot
par(mar=c(5,5,1,1));
plot(DATA$NUClen ~ DATA$GCpct,
      main="",
      xlab="GC%",
      ylab="Sequence length (bp)",
```

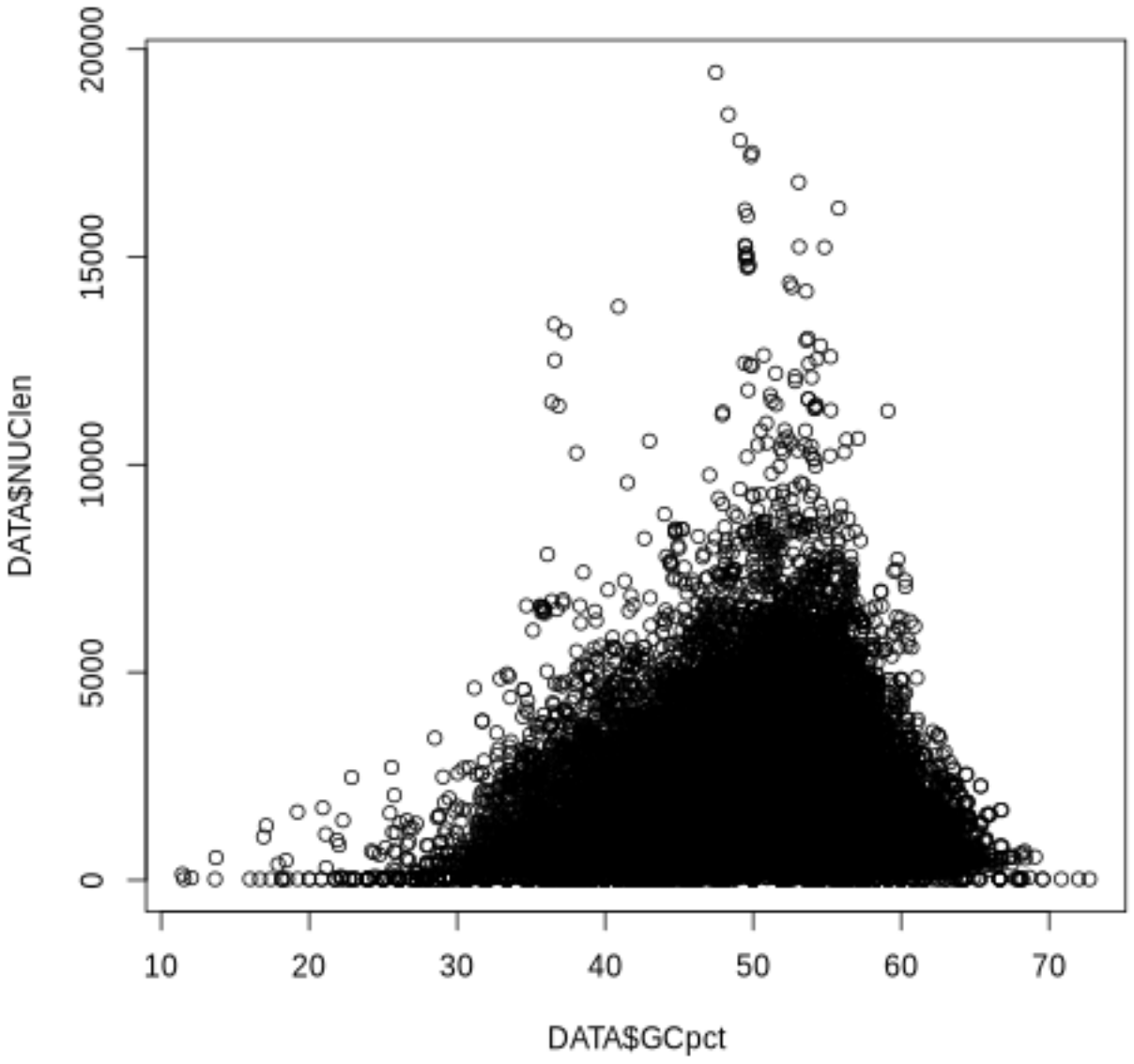


Figure 1: Showing GC content versus sequence length

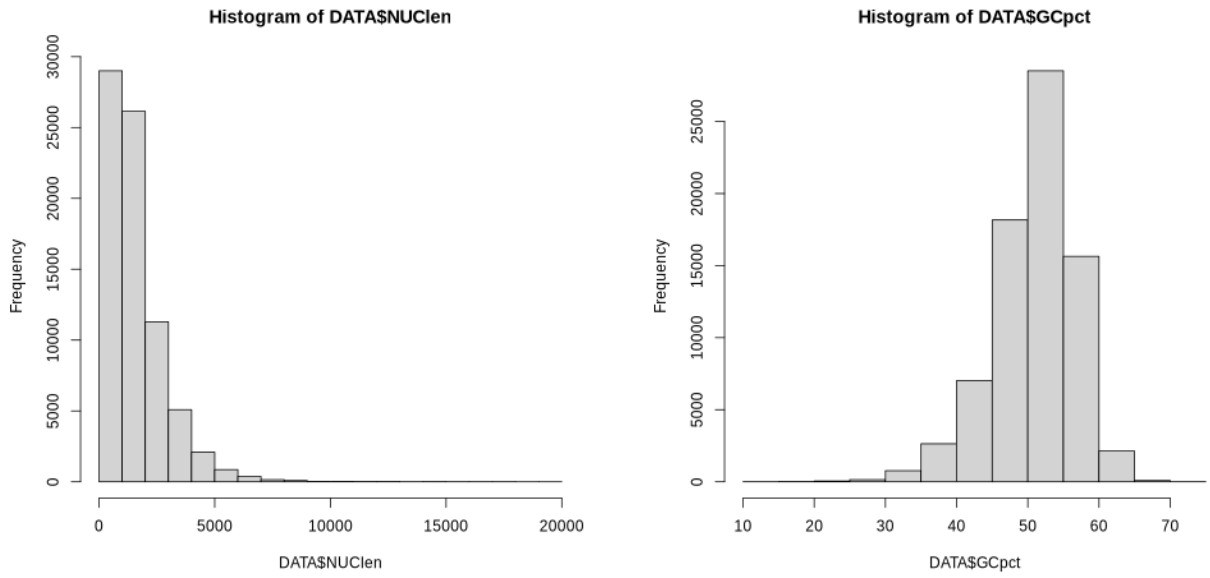


Figure 2: ***D. melanogaster* histograms on mRNA sequences.** Sequence length (bp) distribution on top and %GC content on bottom for the downloaded 75,218 fly sequences.

```
      col="green");
lines(lowess(DATA$NUClen ~ DATA$GCpct),
      col="red", lty="dotted");
mtext(paste("n=",nrow(DATA),sep=""),
      side=3, line=-1);
# drawing x-axis histogram
par(mar=c(0,5,1,1));
barplot(xhist$counts, ylim=c(0, datamax),
        axes=FALSE, space=0,
        col="steelblue", main="D.melanogaster mRNAs")
# drawing y-axis histogram
par(mar=c(5,0,1,1));
barplot(yhist$counts, xlim=c(0, datamax),
        axes=FALSE, horiz=TRUE, space=0,
        col="steelblue", main="")
#
par(def.par); # reseting graphical parameters
dev.off();
```

You can try to recode the `dmel_plot2.png` using `ggplot2` `R` library if you like, but then you must include the corresponding `MarkDown` code block.

2.4 Analyzing differences among chromosomes

Let's check if there are changes in the global distributions of mRNAs length and GC content by chromosome, by strand, or by both; we will need to merge the chromosome sequence identifiers and strands to define the corresponding factor variables in our `R` dataframe. We need a `bed` file relating each mRNA identifier to the chromosome location, again on the UCSC ftp site from "Full database link".

```
wget https://hgdownload.soe.ucsc.edu/goldenPath/dm6/database/all_mrna.txt.gz \
     -O $DT/dmel_all_mrna.txt.gz

zcat data/dmel_all_mrna.txt.gz| wc
# 114843 2526546 15008388

# It seems that there are more annotated mRNAs on the genomic sequences
```

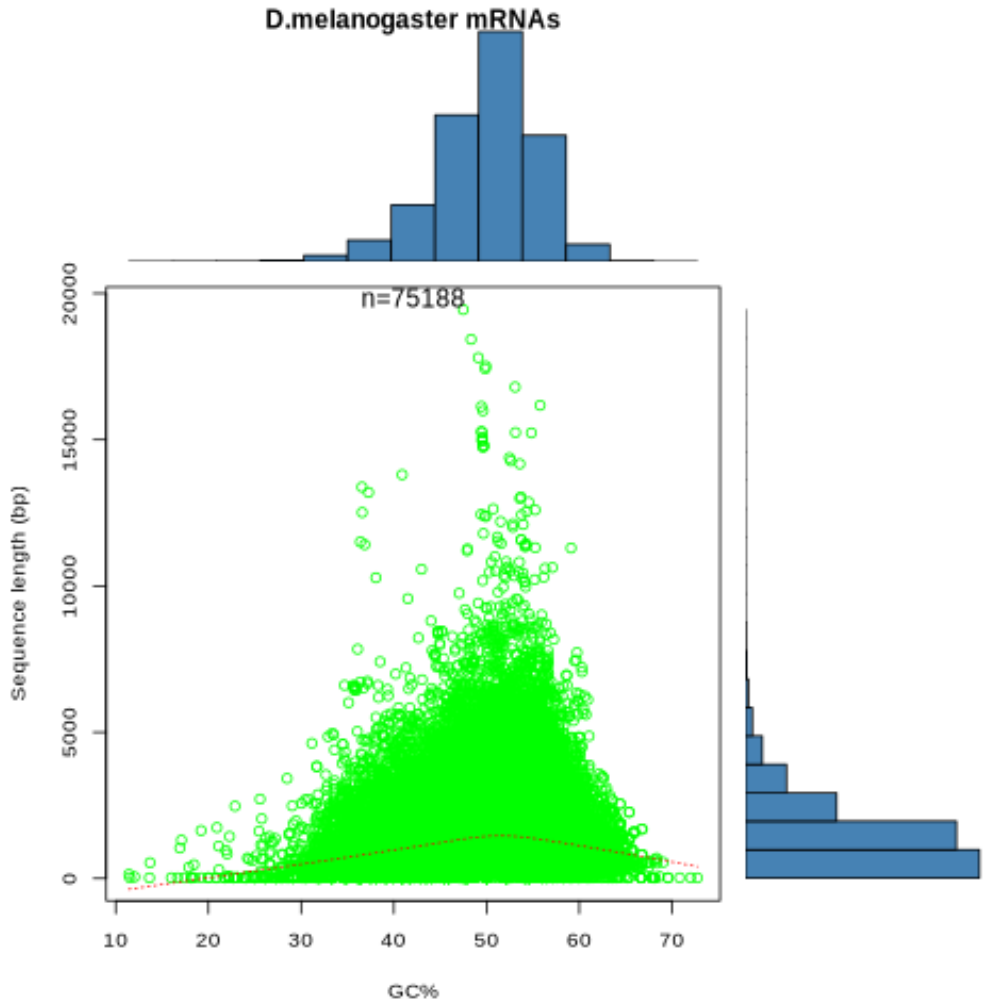


Figure 3: **GC content versus sequence length scatter-plot.** Each point maps a fly mRNA sequence length and GC content values, while the right and top panels display the frequency distributions for those variables on all the data.

```
# than mRNA sequences (75218). However, a simple command shows us that
# probably some of those sequences were mapped at multiple genomic locations.
zcat data/dmel_all_mrna.txt.gz | gawk '{print $11}' | sort | uniq -c | wc
#  75130  150260  1274138
```

We need a command-line or script (`Perl`, `Python`, `awk`, ..., as you like), to merge the strand and chromosome columns from the new donwloaded file. Just remember to include such script on the appendices (see 18).

```
# a command-line?
# selecting chromosome and strand columns
zcat dmel_all_mrna.txt.gz | awk '{print $15, $10}' > merged_chr_strand.txt

# sort both file so we can join them
zcat dmel_mrna.lengc.tbl.gz | sort -k1,1 > sorted_dmel_mrna.lengc.tbl
zcat dmel_all_mrna.txt.gz | sort -k11,11 > sorted_dmel_all_mrna.txt

# join by sequences ID
join -1 1 -2 11 -o 1.1,1.2,1.3,2.10,2.15 sorted_dmel_mrna.lengc.tbl sorted_dmel_all_mrna.txt > merged_mrna
```

Now, the easiest way to compare distributions of lengths and/or GC content by chromosome, strand or both is to use boxplots. Provide the `R` commands and include the figure on this report.

```
library(ggplot2)
merged <- read.table("merged_mrna.txt")
```

```r
colnames(merged) <- c("ID", "Length", "GC", "Strand", "Chr")

library(dplyr)
# The`Chr` column has chromosome names with additional identifiers, we need to fix this
merged <- merged %>%
  mutate(Chr = gsub("_.*", "", Chr)) %>%  # Clean chromosome labels
  mutate(Chr = as.factor(Chr))            # Convert to factor

# Boxplot of lenghts in all chromosome
png(file="images/boxplot_leng_chr.png")
ggplot(data = merged, aes(x = Chr, y = log10(Length), fill = Chr)) +
  geom_boxplot(alpha = 0.5) +
  geom_jitter(aes(col = Chr), alpha = 0.03, width = 0.25) +
  labs(x = "All chromosomes", y = "Logarithm of the gene size (bp)") +
  theme_minimal()
dev.off()

# Boxplot of the GC content in all chromosomes
png(file="images/boxplot_GC_chr.png")
ggplot(data=merged, aes(x=Chr, y=GC, fill=Chr)) +
  geom_boxplot(alpha = 0.5) +
  geom_jitter(aes(col = Chr), alpha = 0.03, width = 0.25) +
  labs(x = "All chromosomes", y = "GC content (%)") +
  theme_minimal()
dev.off()

# Lenghts in both strands
png(file="images/boxplot_leng_strand.png")
ggplot(data=merged, aes(x=Strand, y=log(Length), fill=Strand)) +
  geom_boxplot(alpha = 0.5) +
  geom_jitter(aes(col = Strand), alpha = 0.03, width = 0.25) +
  labs(x = "DNA Strand", y = "Logarithm of the gene size (bp)")
dev.off()

# GC content in both strands
png(file="images/boxplot_GC_strand.png")
ggplot(data=merged, aes(x=Strand, y=GC, fill=Strand)) +
  geom_boxplot(alpha = 0.5) +
  geom_jitter(aes(col = Strand), alpha = 0.03, width = 0.25) +
  labs(x = "DNA Strand", y = "GC content (%)")
dev.off()

# Lengths of both strands in all chromosomes
png(file="images/boxplot_leng_chr_strand.png")
ggplot(data=merged, aes(x=Chr, y=log(Length), fill=Strand)) +
  geom_boxplot(alpha = 0.5) +
  labs(x = "All chromosomes", y = "Logarithm of the gene size (bp)")
dev.off()

# GC content of both strands in all chromosomes
png(file="images/boxplot_GC_chr_strand.png")
ggplot(data=merged, aes(x=Chr, y=GC, fill=Strand)) +
  geom_boxplot(alpha = 0.5) +
  labs(x = "All chromosomes", y = "GC content (%)")
dev.off()
```

Table 1: Comparison of all chromosomes based on length and GC content

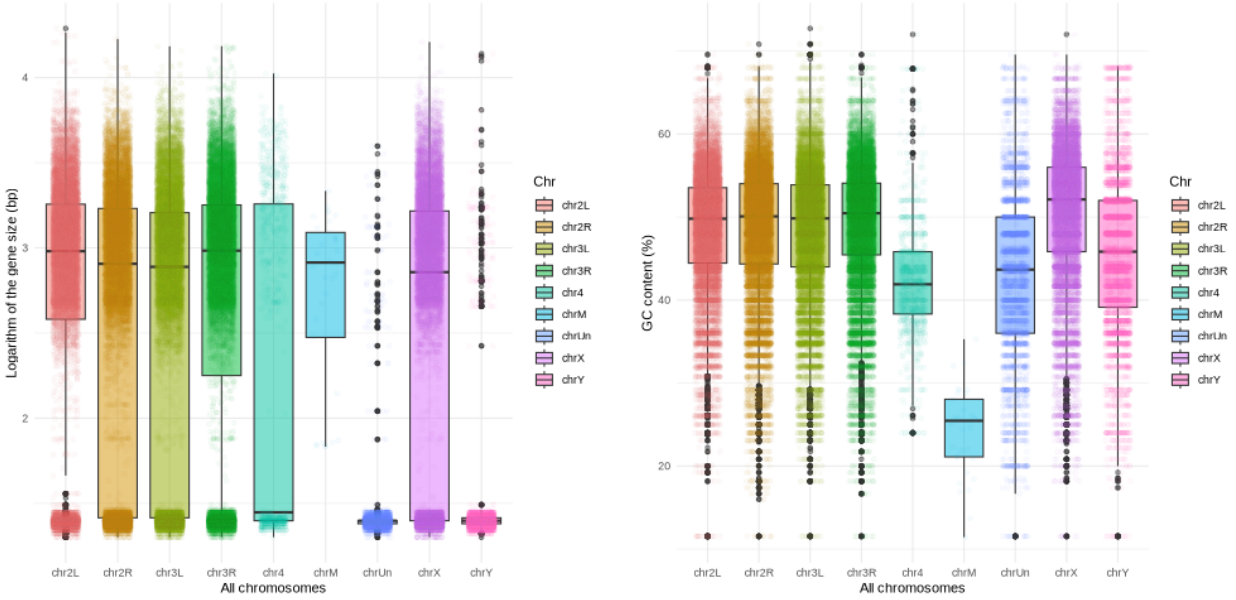


Table 2: Comparison of strands based on lengths and GC content

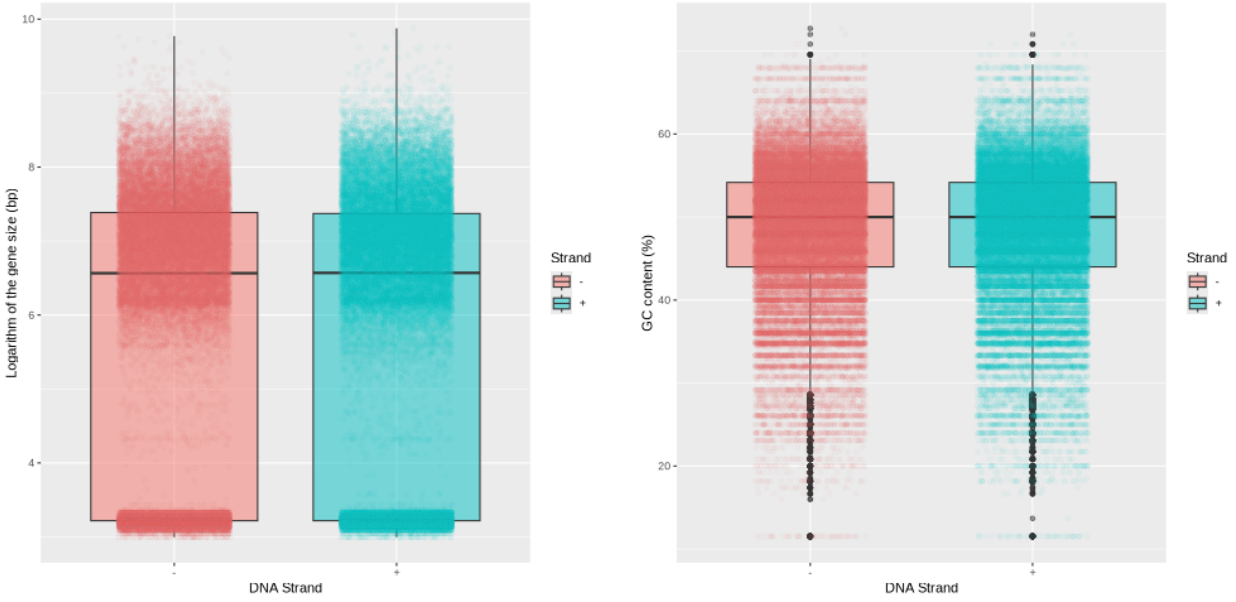
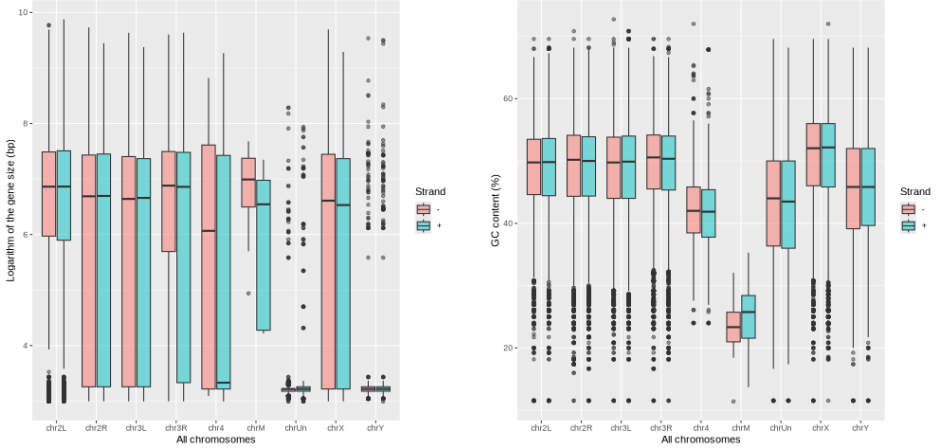


Table 3: Comparison of all chromosomes and strands based on lengths and GC content



2.5 Further analyses

You can provide here the `bash` and `R` commands to reuse the already described protocol to analyze another species transcriptome, for instance the honey bee *Apis mellifera* ("Genome sequence files and select annotations (2bit, GTF, GC-content, etc)" and "Full database" links). Then, you can compare the results with respect the fly transcriptome on the discussion.

```
# Your shell commands here

# honey bee files at UCSC genomes repository
wget https://hgdownload.soe.ucsc.edu/goldenPath/apiMel2/bigZips/mrna.fa.gz \
     -O $DT/apimel_mrna.fa.gz

zcat $DT/apimel_mrna.fa.gz | head -10
# >GU358185 1
# agtagtgcaggaactgctgtactcataaaatggcgaggatattaaaggta
# ggccaatttggccatacttccacgcgaggtatggaagaatatgtaaaaac
# agtagaacaacgtactcatcatatttcagaaggttctataaaattatgga
# aatctatcacttttttgtcgcttttcctataattggacttgcaatggct
# aattgttatttgaaacatcaagaagaacattcaaaaccgccccagaatt
# tgttcattatccttatttaaaaatcatgaataagccttttccatggggag
# atggtaaacataca
# >GU358186 1
# gcttttagtttcgcccgggacattagaaccgacgttctaacgtgcaacaa

zcat $DT/apimel_mrna.fa.gz | egrep -c '^>'
# 120637

zcat $DT/apimel_mrna.fa.gz | \
  infoseq -sequence fasta::stdin \
          -noheading -only -name -length -pgc | \
    head -5
# Display basic information about sequences
# GU358185       314    35.35
# GU358186       719    57.30
# GU358187       598    52.17
# GU358188       1117   48.70
# GU358189       1018   45.58

zcat $DT/apimel_mrna.fa.gz | \
  infoseq -sequence fasta::stdin \
          -outfile $DT/apimel_mrna.lengc.tbl \
          -noheading -only -name -length -pgc

zcat $DT/apimel_mrna.fa.gz | \
  infoseq -sequence fasta::stdin \
          -noheading -only -name -length -pgc | \
    gzip -vc9 - > $DT/apimel_mrna.lengc.tbl.gz

wget https://hgdownload.soe.ucsc.edu/goldenPath/apiMel2/database/all_mrna.txt.gz \
     -O $DT/apimel_all_mrna.txt.gz

zcat apimel_all_mrna.txt.gz| wc
# 127159 2797498 24018653

zcat apimel_all_mrna.txt.gz| gawk '{print $11}' | sort | uniq -c | wc
# 118911  237822 2021443

# Your R commands here
DATA <- read.table("apimel_mrna.lengc.tbl", header=FALSE);
head(DATA, 4);
#        V1   V2    V3
```

```
# 1 GU358185  314 35.35
# 2 GU358186  719 57.30
# 3 GU358187  598 52.17
# 4 GU358188 1117 48.70

colnames(DATA) <- c("ID","NUClen","GCpct");
head(DATA,4);
#         ID NUClen GCpct
# 1 GU358185    314 35.35
# 2 GU358186    719 57.30
# 3 GU358187    598 52.17
# 4 GU358188   1117 48.70

summary(DATA)
#      ID                NUClen          GCpct
# Length:120637      Min.   :   21   Min.   : 0.00
# Class :character   1st Qu.:  513   1st Qu.:29.80
# Mode  :character   Median :  937   Median :35.07
#                    Mean   : 1455   Mean   :35.94
#                    3rd Qu.: 1938   3rd Qu.:41.87
#                    Max.   :17525   Max.   :76.36
```

Histogram:

```
png(file="images/apimel_hist_length.png");
hist(DATA$NUClen);
dev.off();
png(file="images/apimel_hist_pgc.png");
hist(DATA$GCpct);
dev.off();

png(file="images/apimel_plot.png");
plot(DATA$NUClen ~ DATA$GCpct);
dev.off();

png(file="images/apimel_plot2.png");
def.par <- par();
# preparing a layout grid where to combine different plots
nf <- layout(matrix(c(2,0,1,3), # matrix contents is plot order
                    2, 2, byrow=TRUE),
             c(3,1), c(1,3), # cell relative sizes
             TRUE);
# computing data distribution
xhist <- hist(DATA$GCpct, plot=FALSE);
yhist <- hist(DATA$NUClen, plot=FALSE);
datamax <- max(xhist$counts,
               xhist$counts);
#
# drawing the main plot
par(mar=c(5,5,1,1));
plot(DATA$NUClen ~ DATA$GCpct,
      main="",
      xlab="GC%",
      ylab="Sequence length (bp)",
      col="green");
lines(lowess(DATA$NUClen ~ DATA$GCpct),
      col="red", lty="dotted");
mtext(paste("n=",nrow(DATA),sep=""),
      side=3, line=-1);
# drawing x-axis histogram
par(mar=c(0,5,1,1));
barplot(xhist$counts, ylim=c(0, datamax),
```

```
        axes=FALSE, space=0,
        col="steelblue", main="Apis mellifera mRNAs")
# drawing y-axis histogram
par(mar=c(5,0,1,1));
barplot(yhist$counts, xlim=c(0, datamax),
        axes=FALSE, horiz=TRUE, space=0,
        col="steelblue", main="")
#
par(def.par); # reseting graphical parameters
dev.off();

# selecting chromosome and strand columns
zcat apimel_all_mrna.txt.gz | awk '{print $15, $10}' > merged_apimel_chr_strand.txt

# sort both file so we can join them
zcat apimel_mrna.lengc.tbl.gz | sort -k1,1 > sorted_apimel_mrna.lengc.tbl
zcat apimel_all_mrna.txt.gz | sort -k11,11 > sorted_apimel_all_mrna.txt

# join by sequences ID
join -1 1 -2 11 -o 1.1,1.2,1.3,2.10,2.15 sorted_apimel_mrna.lengc.tbl sorted_apimel_all_mrna.txt > merged_
```

Now, the easiest way to compare distributions of lengths and/or GC content by chromosome, strand or both is to use boxplots. Provide the `R` commands and include the figure on this report.

```
library(ggplot2)
merged <- read.table("merged_apimel_mrna.txt")
colnames(merged) <- c("ID", "Length", "GC", "Strand", "Chr")

library(dplyr)
# The`Chr` column has chromosome names with additional identifiers, we need to fix this
merged <- merged %>%
  mutate(Chr = gsub("_.*", "", Chr)) %>%  # Clean chromosome labels
  mutate(Chr = as.factor(Chr))            # Convert to factor

# Boxplot of lenghts in all chromosome
png(file="images/apimel_boxplot_leng_chr.png")
ggplot(data = merged, aes(x = Chr, y = log10(Length), fill = Chr)) +
  geom_boxplot(alpha = 0.5) +
  geom_jitter(aes(col = Chr), alpha = 0.03, width = 0.25) +
  labs(x = "All chromosomes", y = "Logarithm of the gene size (bp)") +
  theme_minimal()
dev.off()

# Boxplot of the GC content in all chromosomes
png(file="images/apimel_boxplot_GC_chr.png")
ggplot(data=merged, aes(x=Chr, y=GC, fill=Chr)) +
  geom_boxplot(alpha = 0.5) +
  geom_jitter(aes(col = Chr), alpha = 0.03, width = 0.25) +
  labs(x = "All chromosomes", y = "GC content (%)") +
  theme_minimal()
dev.off()

# Lenghts in both strands
png(file="images/apimel_boxplot_leng_strand.png")
ggplot(data=merged, aes(x=Strand, y=log(Length), fill=Strand)) +
  geom_boxplot(alpha = 0.5) +
  geom_jitter(aes(col = Strand), alpha = 0.03, width = 0.25) +
  labs(x = "DNA Strand", y = "Logarithm of the gene size (bp)")
dev.off()

# GC content in both strands
png(file="images/apimel_boxplot_GC_strand.png")
ggplot(data=merged, aes(x=Strand, y=GC, fill=Strand)) +
```

```
  geom_boxplot(alpha = 0.5) +
  geom_jitter(aes(col = Strand), alpha = 0.03, width = 0.25) +
  labs(x = "DNA Strand", y = "GC content (%)")
dev.off()

# Lengths of both strands in all chromosomes
png(file="images/apimel_boxplot_leng_chr_strand.png")
ggplot(data=merged, aes(x=Chr, y=log(Length), fill=Strand)) +
  geom_boxplot(alpha = 0.5) +
  labs(x = "All chromosomes", y = "Logarithm of the gene size (bp)")
dev.off()

# GC content of both strands in all chromosomes
png(file="images/apimel_boxplot_GC_chr_strand.png")
ggplot(data=merged, aes(x=Chr, y=GC, fill=Strand)) +
  geom_boxplot(alpha = 0.5) +
  labs(x = "All chromosomes", y = "GC content (%)")
dev.off()
```

Table 4: Comparison of all chromosomes based on length and GC content

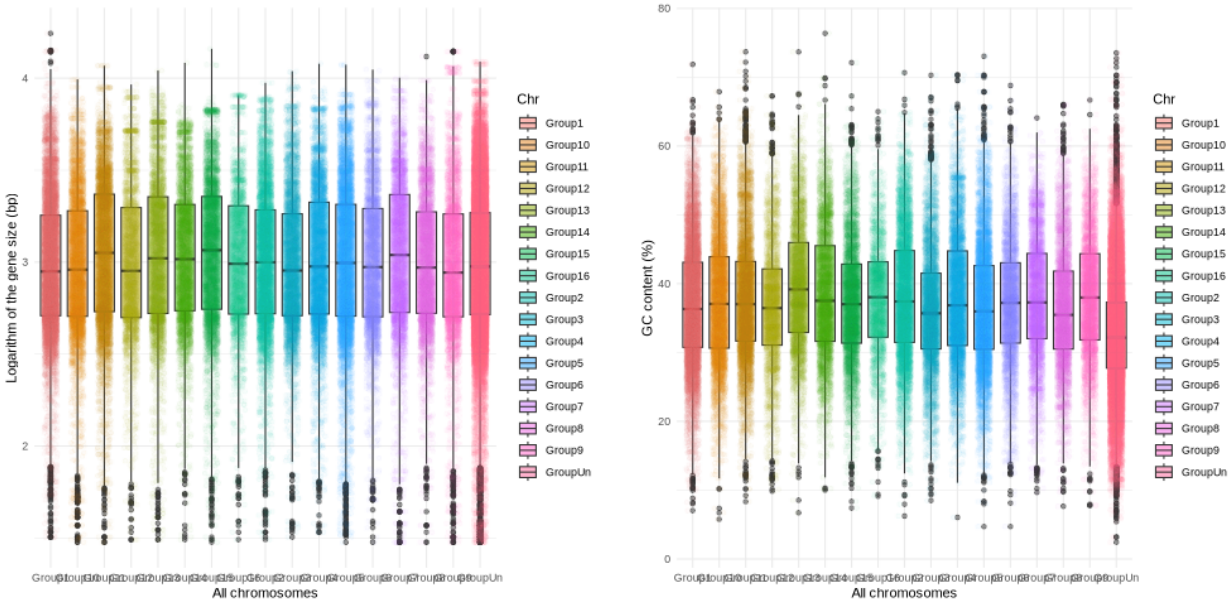


Table 5: Comparison of strands based on lengths and GC content

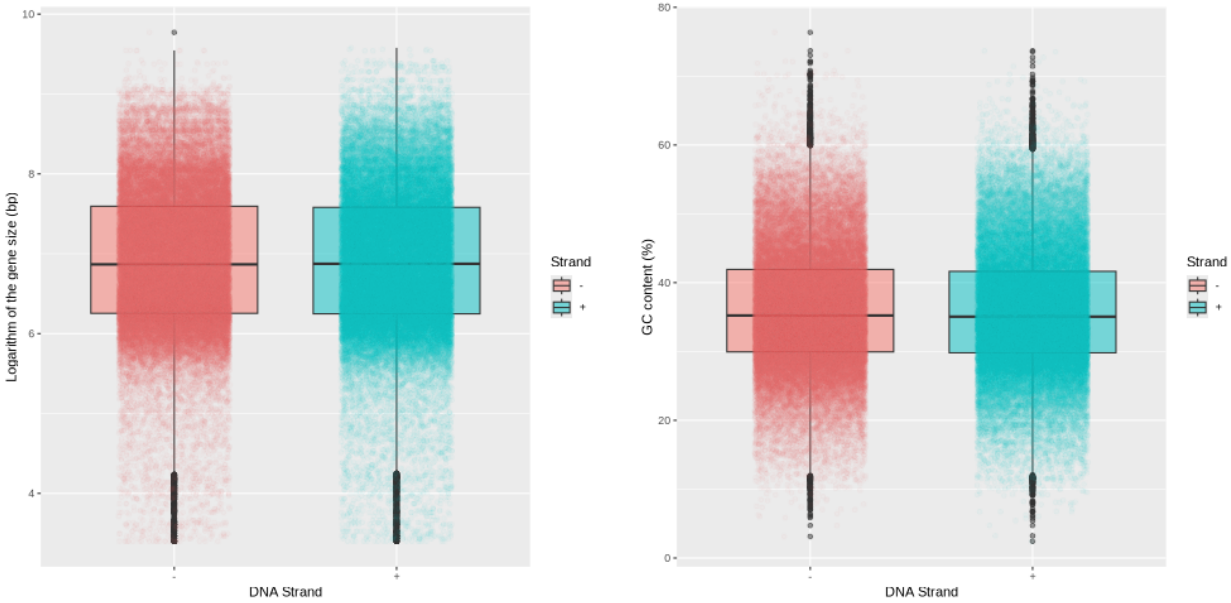
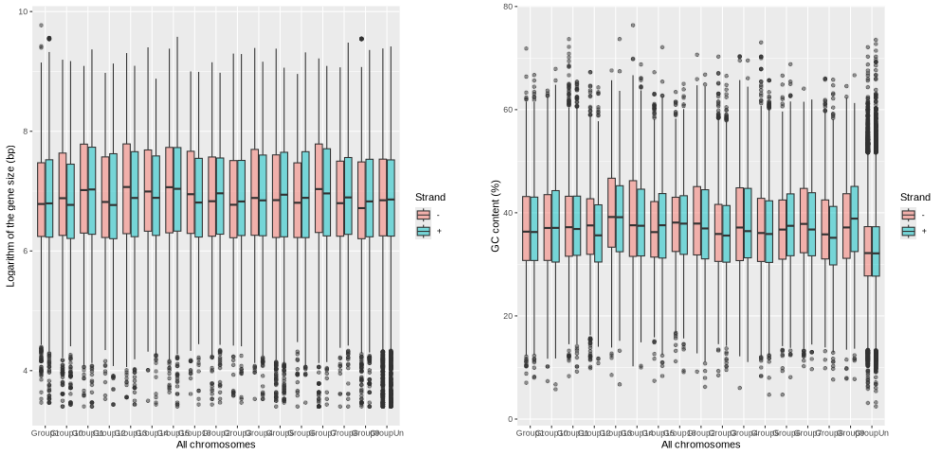


Table 6: Comparison of all chromosomes and strands based on lengths and GC content



3 Discussion

IMPORTANT Discuss your results here (around 300 words). And remember to include in the Appendices section (see page 18), any extra script you wrote from this exercise `bin` folder using the `loadfile` macro. We can take advantage of the LaTeX referencing capabilities, for instance: Figure 2 on page 9 shows the GC and sequence lengths histograms distribution for the set of *D. melanogaster*
mRNA sequences.

The comparison of **Drosophila melanogaster** and **Apis mellifera** reveals distinct patterns in both gene length and GC content across their genomes.

For **D. melanogaster**, the GC content shows substantial variability along different chromosomes. This indicates that certain regions of the genome may have distinct functional or evolutionary pressures that cause these fluctuations in GC composition. GC content is often linked to gene expression, recombination rates, and DNA stability, and the variation we observe in D. melanogaster may reflect adaptation to diverse cellular and environmental conditions. In contrast, for **A. mellifera**, the GC content is more uniform across all chromosomes. This uniformity suggests more stable or consistent selective pressures across the genome, where regions of the chromosomes may experience fewer localized mutations or recombination events that could otherwise lead to fluctuations in GC content.

In terms of gene lengths, **D. melanogaster** also demonstrates more variability. Some genes are similar in length, but others are quite different. This variation could be related to the complexity of gene regulation, alternative splicing, or differing functional requirements of genes in various regions of the genome. The differences in gene length may also hint at varying evolutionary histories of specific chromosomal regions. Conversely, in **A. mellifera**, gene lengths are more consistent. This consistency suggests that the length of coding sequences across different chromosomes may be under similar functional constraints or evolutionary pressures, leading to less variation.

When the data is separated by strands, the patterns in **D. melanogaster** reveal that while GC content is relatively consistent between the strands, gene length still varies. This indicates that while the nucleotide composition is stable across strands, the functional coding regions (genes) are more divergent. In **A. mellifera**, both GC content and gene lengths are relatively uniform across strands. The lack of variability in both parameters suggests a more conserved genomic architecture.

Finally, comparing all chromosomes by strands reinforces these trends. For **D. melanogaster**, some chromosomes show similar values for GC content and gene length, but others exhibit stark differences. This further highlights the diverse evolutionary pressures shaping the D. melanogaster genome. In contrast, **A. mellifera** maintains consistency across its chromosomes, supporting the notion of more uniform selective forces acting across its genome.

4 Appendices

4.1 Software

We have used the following versions:

```
uname -a
# Linux aleph 5.15.0-117-generic #127-Ubuntu SMP
# Fri Jul 5 20:13:28 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux

R --version
# R version 4.3.1 (2023-06-16) -- "Beagle Scouts"
# Copyright (C) 2023 The R Foundation for Statistical Computing
# Platform: x86_64-conda-linux-gnu (64-bit)

infoseq -version
# EMBOSS:6.6.0.0

wget --version
# GNU Wget 1.21.2 built on linux-gnu.

pandoc --version
# pandoc 3.1.3
# Features: +server +lua
# Scripting engine: Lua 5.4

mamba --version
# mamba 1.4.2
# conda 23.3.1
```

4.2 Supplementary files

4.2.1 conda environment dependencies for the exercise

environment.yml

```
#
## ##############################################################################################
##
##   environment.yml
##
##   Defining conda/mamba software dependencies to run BScBI-CG practical exercises.
##
## ##############################################################################################
##
##                CopyLeft 2024 (CC:BY-NC-SA) --- Josep F Abril
##
##   This file should be considered under the Creative Commons BY-NC-SA License
##   (Attribution-Noncommercial-ShareAlike). The material is provided "AS IS",
##   mainly for teaching purposes, and is distributed in the hope that it will
##   be useful, but WITHOUT ANY WARRANTY; without even the implied warranty
##   of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
##
## ##############################################################################################
#
# To install software for the exercise use the following command:
#
#    conda env create --file environment.yml
#
# then run the command below to activate the conda environment:
#
#    conda activate BScBI-CG2425_exercises
#
name: BScBI-CG2425_exercises
channels:
  - bioconda
  - conda-forge
  - defaults
dependencies:
```

```
  - htop
  - vim
  - emacs
  - gawk
  - perl
  - python
  - biopython
  - wget
  - curl
  - gzip
  - r-ggplot2
  - texlive-core
  - pandoc
  - pandocfilters
  - emboss
```

4.2.2 Project specific scripts

an_script_example.pl

```
#!/usr/bin/perl
#
# an_script_example.pl - just a silly example for the MarkDown template
#
use strict;
use warnings;
#

print STDOUT "\n";

for (my $i = 0; $i < 15; $i++) {

    printf STDOUT "\r\tHi, this loop example has iterated %02d times already...", $i + 1;

    sleep(1);

} # for $i

print STDOUT "\n... Bye!!!\n\n";

exit(0);
```

4.2.3 Shell global vars and settings for this project

projectvars.sh

```
## ############################################################################################
##
##   projectvars.sh
##
##   A BASH initialization file for BScBI-CG practical exercise folders
##
## ############################################################################################
##
##                 CopyLeft 2024 (CC:BY-NC-SA) --- Josep F Abril
##
##   This file should be considered under the Creative Commons BY-NC-SA License
##   (Attribution-Noncommercial-ShareAlike). The material is provided "AS IS",
##   mainly for teaching purposes, and is distributed in the hope that it will
##   be useful, but WITHOUT ANY WARRANTY; without even the implied warranty
##   of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
##
## ############################################################################################

#
# Base dir
export WDR=$PWD; # IMPORTANT: If you provide the absolute path, make sure
                 #            that your path DOES NOT contains white-spaces,
                 #            otherwise, you will get weird execution errors.
                 #            If you cannot fix the dir names containing such white-space
                 #            chars, you MUST set this var using the current folder '.'
                 #            instead of '$PWD', i.e:    export WDR=.;
export BIN=$WDR/bin;
```

```
export DOC=$WDR/docs;

#
# Formating chars
export TAB=$'\t';
export RET=$'\n';
export LC_ALL="en_US.UTF-8";

#
# pandoc's vars
NM="SURNAME_NAME";                  #-> IMPORTANT: SET YOUR SURNAME and NAME ON THIS VAR,
RB="README_BScBICG2425_exercise01"; #->            MUST FIX ON MARKDOWN README FILE
                                    #->            FROM TARBALL (AND INSIDE TOO)
RD="${RB}_${NM}";
PDOCFLGS='gfm+pipe_tables+header_attributes';
PDOCFLGS=$PDOCFLGS'+raw_tex+latex_macros+tex_math_dollars';
PDOCFLGS=$PDOCFLGS'+citations+yaml_metadata_block';
PDOCTPL=$DOC/BScBI_CompGenomics_template.tex;
export RD PDOCFLGS PDOCTPL;

#
### IMPORTANT ###
#
#   MacOSX users may need to remove /usr/bin/ from below shell functions,
#   just try first if that path works anyway...
#
function ltx2pdf () {
    RF=$1;
    /usr/bin/pdflatex $RF.tex;
    /usr/bin/bibtex $RF;
    /usr/bin/pdflatex $RF.tex;
    /usr/bin/pdflatex $RF.tex;
}

function runpandoc () {
  /usr/bin/pandoc -f $PDOCFLGS      \
          --template=$PDOCTPL       \
          -t latex --natbib         \
          --number-sections         \
          --highlight-style pygments \
          -o $RD.tex $RD.md;
  ltx2pdf $RD;
}

#
# add your bash defs/aliases/functions below...
```

4.3 About this document

This document was be compiled into a PDF using `pandoc` (see `projectvars.sh` from previous subsection) and some `LaTeX` packages installed in this linux system. `synaptic`, `apt-get` or `aptitude` can be used to retrieve and install those tools from linux repositories. As the `raw_tex` extension has been provided to the `markdown_github` and `tex_math_dollars` formats, now this document supports inline LaTeX and inline formulas!

You can get further information from the following links about the Mark Down syntax, as well as from the manual pages (just type `man pandoc` and/or `man pandoc_markdown`).