BScBI-CG

Practicals Report

Adria Cabello

Exercise 02

— October 18, 2020—

ESCI upf. School of International Studies

Computational Genomics 2020–21

BSc in Bioinformatics

UPF – UPC – UB

Contents

List of Tables

List of Figures

1 Introduction

We want to analyze genome sequences of four bacteria species: *Escherichia coli*, *Bacillus subtilis*, *Mycoplasma genitalium*, and *Mycoplasma pneumoniae*. All the downstream commands from the initial template will focus on the first of them, *E. coli*; you will need to perform similar analyses for the other three species, then discuss differences among those genomes from the data you will obtain.

1.1 Objectives

- To practice sequence retrieval commands and how to reformat records, for instance extracting `FASTA` records from a `GenBank` formatted file.
- To implement and apply a running-windows approach to calculate sequence properties across a small set of genomic sequences.
- To visualize those properties in order to compare the results obtained for the provided sequences.
- To introduce LaTeX variables, item lists, and improved tabular environments.

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. If you still need to install the base software, please refer to `exercise_00` and `exercise_01`, as well as the short tutorials from the Computational Genomics Virtual Campus at ESCI.

For this practical you probably may need to install the following packages:

```
###################################
# emboss - European molecular biology open software suite

# 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

# 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".
```

```
# 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...
```

From now on, we assume that you are using a Debian-based linux distribution, so we will show only the corresponding set of commands for that distribution.

```
##################################
# jellyfish - count k-mers in DNA sequences
sudo apt-get install jellyfish
```

1.2.2 Initializing the main report files

As in the previous exercises, remember to download first the exercise tarball from the Computational Genomics Virtual Campus at ESCI, unpack this 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 exercise...

```
# You probably have already done this step.
tar -zxvf BScBI_CG2021_exercise_02.tgz
cd exercise_02

# Rename report file including your "NAME" and "SURNAME"
mv -v README_BScBICG2021_exercise02_SURNAME_NAME.md \
      README_BScBICG2021_exercise02_yourSurname_yourName.md

# Open exercise files using your text editor of choice
# (for instance vim, emacs, gedit, sublime, atom, ...);
# fix "NAME" and "SURNAME" placeholders on them
# and save those changes before continuing.
emacs  projectvars.sh \
       README_BScBICG2021_exercise02_yourSurname_yourName.md &

# Let's start with some initialization.
source projectvars.sh
echo $WDR

# Once you have run the commands that are already in the initial
# MarkDown document, you are probably ready to run this:
runpandoc
```

Let's start with the analyses, and may the shell be with you...

2 Calculating Genome Sequence Properties

2.1 Datasets

Species	Type	RefSeq ID	INSDC	Size (Mb)	GC%	Protein	rRNA	tRNA	Other RNA
Escherichia coli	Chr	NC_000913.3	U00096.3	4.64	50.8	4,140	22	89	6
Bacillus subtilis	Chr	NC_000964.3	AL009126.3	4.22	43.5	4,174	30	86	6
Mycoplasma genitalium	Chr	NC_000908.2	L43967.2	0.58	31.7	515	3	36	
Mycoplasma pneumoniae	Chr	NC_000912.1	U00089.2	0.82	40.0	691	3	37	31

Table 1: **Genome sequence information for four bacteria species downloaded from GenBank.** Whole-genome summary table showing number of annotated features, such genes and proteins, along with sequence characteristics, such as size and average GC content.

Table 2 provides an overview of the four bacterial genomes we have to analyze on this exercise, for which we provide a short description here:.

E. coli is typically present in the lower intestine of humans; is easily grown in a laboratory setting and also readily amenable to genetic manipulation, making it one of the most studied prokaryotic model organisms. We will work with this species representative genome, which is *E. coli* strain K-12 substr. MG1655. *B. subtilis* is a model organism for prokaryotic cell differentiation and development, and was one of the first bacteria studied. Representative genome for this species is *B. subtilis* subsp. *subtilis* strain 168. Mycoplasmas carry the smallest genomes of self-replicating cells together with the smallest set of functional coding regions; *Mycoplasma genitalium* genome was the second to be reported in 1995[1]. The representative genome is *M. genitalium* G37. *M. pneumoniae* causes respiratory tract infections. We are going to use *M. pneumoniae* M129 as representative genome.

It's time to get the sequences from a set of links we have retrieved from GenBank genome division. We are not going to take just the sequences in `fasta` format, we will download them in GenBank format this time.

```
# IMPORTANT: ensure that your WDR variable definition in projectvars.sh
#            does not contain a path having white-spaces on the folder names.

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.

# Downloading the Ecol genome in GenBank format
GBFTP=ftp://ftp.ncbi.nlm.nih.gov/genomes/all

wget $GBFTP/GCF/000/005/845/GCF_000005845.2_ASM584v2/GCF_000005845.2_ASM584v2_genomic.gbff.gz \
     -O $DT/Ecol_referencegenome.gb.gz

# the other three genomes are available through the following paths:
#
#  Bsub GCF/000/009/045/GCF_000009045.1_ASM904v1/GCF_000009045.1_ASM904v1_genomic.gbff.gz
#  Mgen GCF/000/027/325/GCF_000027325.1_ASM2732v1/GCF_000027325.1_ASM2732v1_genomic.gbff.gz
#  Mpne GCF/000/027/345/GCF_000027345.1_ASM2734v1/GCF_000027345.1_ASM2734v1_genomic.gbff.gz
#
# save them as Bsub_referencegenome.gb.gz, Mgen_referencegenome.gb.gz,
# and Mpne_referencegenome.gb.gz respectively.
# For such a task, you can use a shell loop for instance:

while read Ospc Gftp;
  do {
    echo "# Downloading genome sequence for $Ospc" 1>&2;
    wget $GBFTP/${Gftp}_genomic.gbff.gz \
         -O $DT/${Ospc}_referencegenome.gb.gz
  }; done <<'EOF'
Bsub GCF/000/009/045/GCF_000009045.1_ASM904v1/GCF_000009045.1_ASM904v1
Mgen GCF/000/027/325/GCF_000027325.1_ASM2732v1/GCF_000027325.1_ASM2732v1
Mpne GCF/000/027/345/GCF_000027345.1_ASM2734v1/GCF_000027345.1_ASM2734v1
```

[1] "The minimal gene complement of *Mycoplasma genitalium*". Fraser CM, et al. *Science*, 1995.

```
EOF

### IMPORTANT NOTE ###
#
# If the firewall does not allow you to connect to the original NCBI ftp site
# then you can run the following commands to download files from
# the https alternate repository at compgen.bio.ub.edu server.
# Just remind to replace the user and password strings with those
# from the slides for the introduction to the practicals.
#
GBFTP=https://compgen.bio.ub.edu/~jabril/teaching/BScBI-CG2021/repo_ex2

while read Ospc Gftp;
  do {
    echo "# Downloading genome sequence for $Ospc" 1>&2;
    wget --user="UUUUUUUUUUUU" \
         --password="XXXXXXXX" \
         $GBFTP/${Ospc}_referencegenome.gb.gz \
         -O $DT/${Ospc}_referencegenome.gb.gz
  }; done <<'EOF'
Ecol GCF_000005845.2_ASM584v2
Bsub GCF_000009045.1_ASM904v1
Mgen GCF_000027325.1_ASM2732v1
Mpne GCF_000027345.1_ASM2734v1
EOF
```

2.2 Retrieving the sequences

Let's extract the raw genomic sequences from the GenBank formated files:

```
# for manual pages on this emboss tool run:  tfm seqret
#
SPC="Ecol"

zcat $DT/${SPC}_referencegenome.gb.gz | \
  seqret -sequence genbank::stdin -outseq fasta::stdout | \
    gzip -9c - > $DT/${SPC}_referencegenome.fa.gz

# let's verify if fasta sequence has same length as reported in the GenBank file

zegrep '^LOCUS'  $DT/${SPC}_referencegenome.gb.gz
#> LOCUS       NC_000913            4641652 bp    DNA     circular CON 08-AUG-2016

zcat $DT/${SPC}_referencegenome.fa.gz | \
  infoseq -sequence fasta::stdin \
          -noheading -only -name -length -pgc
#> Display basic information about sequences
#> NC_000913      4641652 50.79

### repeat the commands for the other three genomes ###

# This part of the code can be found in runpipeline.sh
# and calls the script raw_seq.sh

echo "RAW FORMAT (FASTA)"
for SPC in $args;
do
  . $WDR/bin/raw_seq.sh $SPC
done
echo ""
```

From the output of the two commands, we can conclude that fasta sequence for the downloaded *E. coli* genome has the

correct length, 4641652bp, and that the GC content is almost the same as the one reported on Table 2, 50.79% versus 50.8% respectively (so the difference is due to rounding to one decimal position).

2.3 Summary of sequence content

2.3.1 Chaos-plot

`EMBOSS` suite has a command to calculate chaos plots, a simple graphical representation of sequence composition that we can use to visually compare the four genomes analyzed on this exercise.

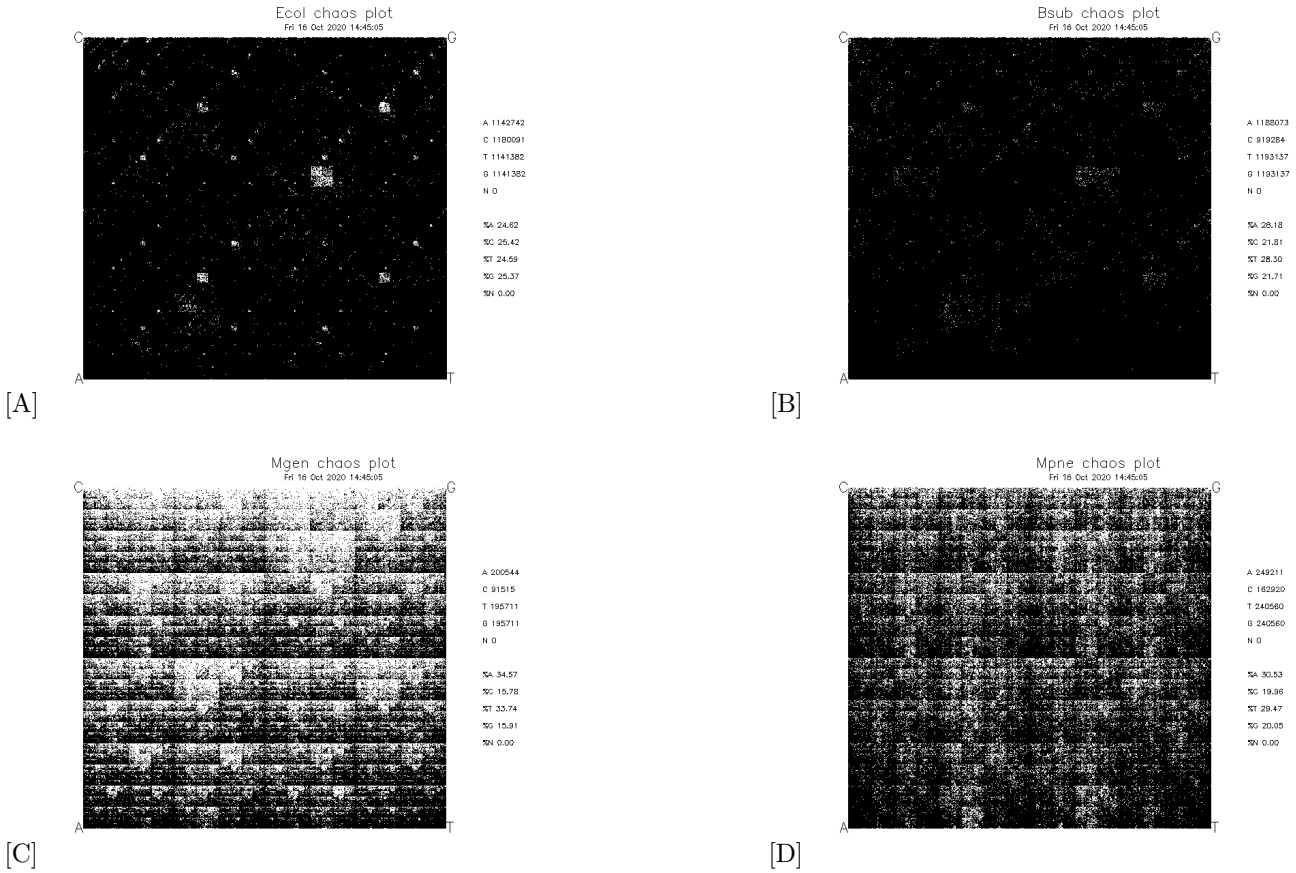
```
zcat $DT/${SPC}_referencegenome.fa.gz | \
  chaos -sequence fasta::stdin -verbose \
        -graph png -gtitle "$SPC chaos plot" \
        -goutfile $WDR/images/${SPC}_chaosplot

### repeat the commands for the other three genomes ###

# Can be found in runpipeline.sh

echo "CHAOS-PLOT"
for SPC in $args;
do
  . $WDR/bin/chaos-plot.sh $SPC # this loop calls the function of above
done
echo ""
```

You **must** include here a LaTeX figure, defined as a table of two rows and two columns containing the four `png` plots, using `input` to load an external `tex` file stored in the `docs` directory (we had already examples on the previous exercise, see for instance "`exercise_01/docs/fig_histograms.tex`").



[A] [B]

[C] [D]

Figure 1: **Chaos plot.**

2.3.2 Computing GC content variation across the genome

We will use a short script in `Perl` to calculate a parameter over the genome, GC content for instance, using a running window. You can try to rewrite in `Python` or any other programming language, but we should focus here on the effect of the window size on the final results. We define a short script in the following code chunk that you must copy into a file within the bin folder. It takes two parameters, the window length and the input sequence, so that you can play with several lengths to evaluate which one can provide the best comparison across genomes. Once you choose a window length on the *Escherichia coli* genome, you can run the same command fixing that parameter and changing the input file for the other three genomes.

```
#!/usr/bin/perl
#
# you can save this script as "bin/genomicgcwindows.pl"
#
use strict;
use warnings;

# variables initialization
my $window = shift @ARGV;
$window < 10 && die("# ERROR: window length must be a positive integer equal or greater than 10\n");
my $step = int($window / 2); # we have chosen to fix this parameter
my %SEQS = (); # just in case there is more than one sequence on the input

# read sequences
my $sid = undef;
while (<>) {
    next if /^\s*$/o;
    chomp;
    $_ =~ /^>/ && do {  # finding the sequence header with its name
        ($sid, undef) = split /\s+/, $_;
        exists($SEQS{$sid}) || ($SEQS{$sid} = '');
        next;
    };
    defined($sid) || next;
    $_ =~ s/\s+//og; # just in case there are white spaces on the sequence
    $SEQS{$sid} .= uc($_);
}; # while $_

# analyze sequences
foreach my $sid (keys %SEQS) {
    my $seq = $SEQS{$sid};
    for (my $n = 0; $n < length($seq) - $window + 1; $n += $step) {
        my $winseq = substr($seq, $n, $window);
        printf "%s %d %.1f\n", $sid, $n + $step, &getGC(\$winseq,$window);
    };
}; # foreach $sid

exit(0);

# available functions
sub getGC() {
    my ($sq, $wn) = @_;
    my $gc = 0;
    for (my $c = 0; $c < $wn; $c++) {
        $gc++ if substr($$sq, $c, 1) =~ /[GC]/o;
    }; # for $c
    return $gc / $wn * 100;
} # getGC
```

Let's run the `Perl` scrip on a set of increasing windows lengths.

```
# provide execution permissions to the perl script
chmod a+x $WDR/bin/genomicgcwindows.pl
```

```
# running on Ecoli genome sequence

for WNDW in 100 200 500 1000 2000 5000 10000;
  do {
    echo "# Running windowed GC analysis on $SPC for window length = $WNDW" 1>&2;
    zcat $DT/${SPC}_referencegenome.fa.gz | \
      $WDR/bin/genomicgcwindows.pl $WNDW - | \
         gzip -c9 - > $WDR/stats/${SPC}_genomegcanalysis_wlen$WNDW.tbl.gz;
  }; done;

# just check the output
ls -1 $WDR/stats/${SPC}_genomegcanalysis_wlen*.tbl.gz | \
  while read FL;
    do {
      echo $FL;
      zcat $FL | head -2;
    }; done;
#> stats/Ecol_genomegcanalysis_wlen100.tbl
#> >NC_000913 50 42.0
#> >NC_000913 100 34.0
#> stats/Ecol_genomegcanalysis_wlen200.tbl
#> >NC_000913 100 37.0
#> >NC_000913 200 44.0
#> stats/Ecol_genomegcanalysis_wlen500.tbl
#> >NC_000913 250 46.4
#> >NC_000913 500 52.8
#> stats/Ecol_genomegcanalysis_wlen1000.tbl
#> >NC_000913 500 50.7
#> >NC_000913 1000 54.4
#> stats/Ecol_genomegcanalysis_wlen2000.tbl
#> >NC_000913 1000 51.9
#> >NC_000913 2000 52.5
#> stats/Ecol_genomegcanalysis_wlen5000.tbl
#> >NC_000913 2500 53.0
#> >NC_000913 5000 52.2
#> stats/Ecol_genomegcanalysis_wlen10000.tbl
#> >NC_000913 5000 52.1
#> >NC_000913 10000 50.8

### repeat the commands for the other three genomes ###
# provide execution permissions to the perl script
chmod a+x $WDR/bin/genomicgcwindows.pl

echo "Genomic GC Windows"
for SPC in $args;
do
  for w in 100 200 500 1000 2000 5000 10000;
  do
    echo "Windows analysis of "$SPC" with windows length "$w;
    zcat $DT/${SPC}_referencegenome.fa.gz | \
    $WDR/bin/genomicgcwindows.pl $w - | \
    gzip -c9 -> $WDR/stats/${SPC}_genomegcanalysis_wlen$w.tbl.gz;
  done;
done;
echo ""

echo "Checking output of Genomic GC Windows"
for SPC in $args;
do
  ls -1 $WDR/stats/${SPC}_genomegcanalysis_wlen*.tbl.gz | \
    while read FL;
      do {
```

```
        echo $FL;
        zcat $FL | head -2;
      }; done;
done;
echo ""
```

We can plot each of those tables using the nucleotide positions on the X-axis and the computed GC content as Y-axes, those figures should be five times wider than taller that will allow us to stack them for comparing the results of the different window lengths.

```
R
# then assuming you use R command-line shell from the terminal... ;^D

# example here for Ecol and window length equal to 100bp

GC_avg <- 50.79; # the whole genome average GC content

ZZ <- gzfile('stats/Ecol_genomegcanalysis_wlen100.tbl.gz');
GC_w100 <- read.table(ZZ, header=FALSE);
colnames(GC_w100) <- c("CHRid","NUCpos","GCpct");

summary(GC_w100)
#>         CHRid           NUCpos            GCpct
#>  NC_000913:92832   Min.   :     50   Min.   :15.00
#>                    1st Qu.:1160438   1st Qu.:47.00
#>                    Median :2320825   Median :52.00
#>                    Mean   :2320825   Mean   :50.79 (*)
#>                    3rd Qu.:3481212   3rd Qu.:56.00
#>                    Max.   :4641600   Max.   :78.00
# mean of all GCpct (*) should be closer to the whole genome average GC, should it?

library(ggplot2);

G <- ggplot(GC_w100, aes(x=NUCpos, y=GCpct)) +
        geom_line(colour = "blue") +
        theme_bw() +
        geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", size=1.5) +
        ggtitle("E.coli GC content over the genome (window length = 100bp)") +
        labs(x="Genomic Coords (bp)", y="%GC Content");

ggsave("images/Ecol_genomegcanalysis_wlen100.png",
       plot=G, width=25, height=8, units="cm", dpi=600);
```

Include here a figure combining the plots for the set of window lengths (100, 200, 500, 1000, 2000, 5000, and 10000). Then choose one of those windows lengths and provide the commands to analyze the other three genomic sequences. After that, you can include another figure stacking the results for that window length on all the genomes.

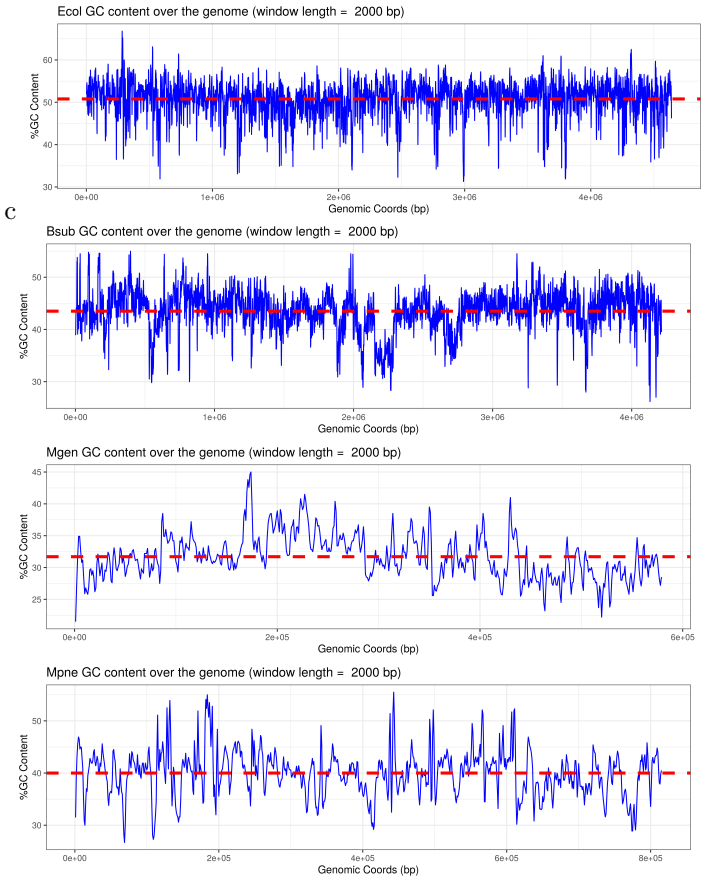


Figure 2: **Windows size analysis.**

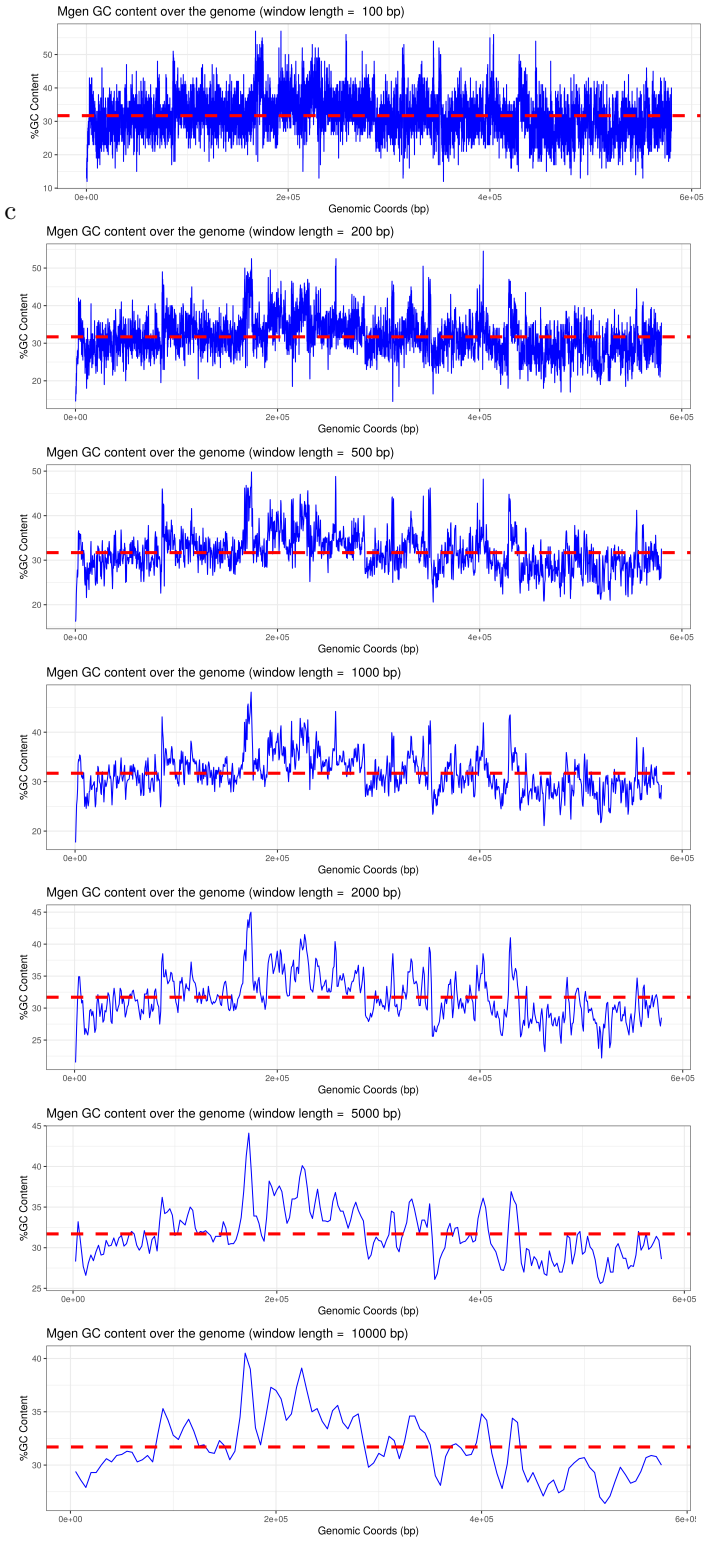


Figure 3: **Windows size analysis.**

```
### repeat the commands for the other three genomes ###
#!/usr/bin/Rscript
#give permisions:
#chmod +x R_plot_.r
#execute:
#./R_plot_.r Ecol Bsub Mgen Mpne

args = commandArgs(trailingOnly=TRUE)
for (SPC in args){
  for (w in c(100, 200, 500, 1000, 2000, 5000, 10000)){
    GC_avg <- 50.79; # the whole genome average GC content
    d <- '/home/adria/Escritorio/Computational_genomics/exercise_02/stats/'
    middle = '_genomegcanalysis_wlen'
    extension = '.tbl.gz'
    path<-(paste(d,as.character(SPC),middle,as.character(w),extension, sep=""))
    print(path)
    ZZ <- gzfile(path);
    GC_w <- read.table(ZZ, header=FALSE);
    colnames(GC_w) <- c("CHRid","NUCpos","GCpct");

    summary(GC_w)

    library(ggplot2);

    G <- ggplot(GC_w, aes(x=NUCpos, y=GCpct)) +
      geom_line(colour = "blue") +
      theme_bw() +
      geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", size=1.5) +
      ggtitle(paste("E.coli GC content over the genome (window length = ",w,"bp)")) +
      labs(x="Genomic Coords (bp)", y="%GC Content");
    dirs <-'/home/adria/Escritorio/Computational_genomics/exercise_02/images/'
    mid2 <- '_genomegcanalysis_wlen'
    ex <-'.png'
    p <- (paste(dirs,SPC,mid2,w,ex,sep=''))
    ggsave(p,plot=G, width=25, height=8, units="cm", dpi=600);

  }
}
#!/usr/bin/Rscript
#give permisions:
#chmod +x R_plot_.r
#execute:
#./R_plot_.r Ecol Bsub Mgen Mpne

args = commandArgs(trailingOnly=TRUE)
#getwd()
dir <- setwd('/home/adria/Escritorio/Computational_genomics/exercise_02/stats/')
for (SPC in args){
  if (SPC == "Ecol"){
    GC_avg <- 50.79;}
  else if (SPC=="Bsub"){
    GC_avg <- 43.5;}
  else if (SPC == "Mgen"){
    GC_avg <- 31.7;}
  else if (SPC == "Mpne"){
    GC_avg <- 40.0;
  }

  for (w in c(100, 200, 500, 1000, 2000, 5000, 10000)){
    #GC_avg <- 50.79; # the whole genome average GC content
    d <- '/home/adria/Escritorio/Computational_genomics/exercise_02/stats/'
    middle = '_genomegcanalysis_wlen'
    extension = '.tbl.gz'
```

```
    path<-(paste(d,as.character(SPC),middle,as.character(w),extension, sep=""))
    print(path)
    ZZ <- gzfile(path);
    GC_w <- read.table(ZZ, header=FALSE);
    colnames(GC_w) <- c("CHRid","NUCpos","GCpct");

    summary(GC_w)

    library(ggplot2);

    G <- ggplot(GC_w, aes(x=NUCpos, y=GCpct)) +
      geom_line(colour = "blue") +
      theme_bw() +
      geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", size=1.5) +
      ggtitle(paste(SPC,"GC content over the genome (window length = ",w,"bp)")) +
      labs(x="Genomic Coords (bp)", y="%GC Content");
    dirs <-'/home/adria/Escritorio/Computational_genomics/exercise_02/images/'
    mid2 <- '_genomegcanalysis_wlen'
    ex <-'.png'
    p <- (paste(dirs,SPC,mid2,w,ex,sep=''))
    ggsave(p,plot=G, width=25, height=8, units="cm", dpi=600);

  }
}
```

2.4 Analysis of *k*-mer composition

There are many software tools to account for the k-mers appearing in a genomic sequence, we will use `jellyfish` for this purpose. It produces a summary file where we can easily get the number of total, distinct and unique k-mers. We can even compare which k-mers appear in more than one species genome, but we will focus this analysis on those numbers from the summary file.

```
zcat $DT/${SPC}_referencegenome.fa.gz | \
    jellyfish count -m 20 -C -t 4 -c 8 -s 10000000 /dev/fd/0 \
              -o $WDR/stats/${SPC}_jellyfish_k20.counts;

jellyfish stats $WDR/stats/${SPC}_jellyfish_k20.counts;
#> Unique:    4507760
#> Distinct:  4542190
#> Total:     4641633
#> Max_count:      82
# also consider that the total theoretical sequences of k=20 is 4^20 = 1,099511628e+12
```

We can also combine commands into a shell function, the code below runs the same two `jellyfish` commands of the previous code block:

```
# we define here the shell function
function jellyfish_on_kmer () {
  THYSPC=$1;
  KMERSZ=$2;
  echo "# ${THYSPC} - ${KMERSZ}" 1>&2;
  zcat $DT/${THYSPC}_referencegenome.fa.gz | \
    jellyfish count -m $KMERSZ -C -t 4 -c 8 -s 10000000 /dev/fd/0 \
                -o $WDR/stats/${SPC}_jellyfish_k${KMERSZ}.counts;
  jellyfish stats $WDR/stats/${SPC}_jellyfish_k${KMERSZ}.counts;
}

# here we use the previous function on a shell command-line,
#   with different parameters
#
jellyfish_on_kmer Ecol 20;
# # Ecol - 20
```

```
# Unique:    4507760
# Distinct:  4542190
# Total:     4641633
# Max_count: 82
#
jellyfish_on_kmer Bsub 35;
# # Bsub - 35
# Unique:    4152426
# Distinct:  4164735
# Total:     4215572
# Max_count: 10
#
#   or within loops...
#
for SPC in Ecol Bsub;
  do {
    for KSZ in 10 15 20;
      do {
       jellyfish_on_kmer $SPC $KSZ;
      }; done;
  }; done;
# ...

# yet another option is to move the shell commands
# that perform single tasks into shell scripts instead of functions.
```

Try different k-mer sizes (i.e. 10, 15, 20, 25, 30, 35, and 40), on the genomic sequences of the four species and summarize them into another LaTeX table to include below. You can take "`docs/tbl_genbank_summary_info_genomes.tex`" as example to create this table.

Species/K-mers 10	Unique	Distinct	Total	Max count
Ecol	40625	490389	4641643	284
Bsub	47840	494580	4215597	333
Mgen	87099	192529	580067	122
Mpne	123499	292158	816385	58
Species/K-mers 15	Unique	Distinct	Total	Max count
Ecol	4357730	4462229	4462229	10
Bsub	4015017	4092956	4215592	20
Mgen	550161	562149	580062	39
Mpne	740058	766612	816380	36
Species/K-mers 20	Unique	Distinct	Total	Max count
Ecol	4507760	4542190	4641633	82
Bsub	4143882	4159409	4215587	10
Mgen	564076	569990	580057	31
Mpne	757793	778224	816375	15

Table 2: **Information about K-mers in 4 species from GenBank.** Summary table showing number of k-mers found with jellyfish software. Unique k-mers, distinct k-mers, total k-mers and max count.

I used the following script for this part, we can find more information in the file runpipeline.sh:

```
#IT USES THE ARGUMENTS INPUTS FROM runpipeline.sh SCRIPT.

function jellyfish_on_kmer () {
  THYSPC=$1;
  KMERSZ=$2;
  echo "# ${THYSPC} - ${KMERSZ}" 1>&2;
  zcat $DT/${THYSPC}_referencegenome.fa.gz | \
    jellyfish count -m $KMERSZ -C -t 4 -c 8 -s 10000000 /dev/fd/0 \
                -o $WDR/stats/${SPC}_jellyfish_k${KMERSZ}.counts;
  jellyfish stats $WDR/stats/${SPC}_jellyfish_k${KMERSZ}.counts;
  jellyfish stats $WDR/stats/${SPC}_jellyfish_k${KMERSZ}.counts >> $WDR/stats/jellyfish_total.count;
}


echo "Analysis of k-mer composition"
for SPC in $args;
do
  for KSZ in 10 15 20;
  do
    echo "Analysis of "$SPC" K-mers with KSZ =" $KSZ;
    jellyfish_on_kmer $SPC $KSZ;
  done;
done;
```

I tried to plot the results from Jellyfish:

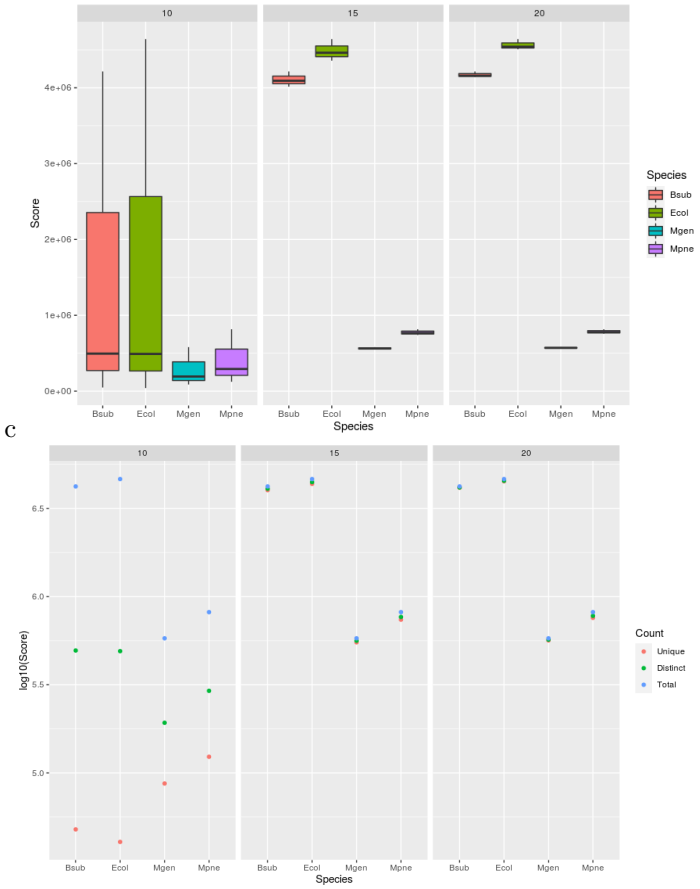


Figure 4: **Total from Jellyfish by species and by K-mers size, and Distribution of scores from Jellyfish by species and by K-mers size.**

3 Discussion

IMPORTANT Discuss your results here (around 300 words). And remember to include in the Appendices section (see page 16), any extra script you wrote from this exercise `bin` folder using the `loadfile` macro. We can take advantage of the LaTeX referencing capabilities, as described in the first exercise template.

The chaos plot or CGR is based on recognizing patterns in the nucleotides sequences by considering DNA sequences as strings {A,C,T,G}. Oligomers and frequencies are written into the boxes and are visualized as white (none) or different intensities of black color (depending on its frequency).1

After performing the chaos plot 5 for each one of the species, we can see that E.col and B.sub have a lot of possible combinations of nucleotides, we can see very few white boxes, while Mgen and Mpne have a lot of white boxes in their plots. This means that the total amount of nucleotide combinations is expected to be smaller than Ecol and Bsub.

I have performed the windows size analysis of the GC content for different species with different windows sizes 10. The size of the windows is an important factor to avoid some noise in the plot. It is better to work with the length of a window of 1000bp, as we can see in 9, the fact that we choose 2000bp introduces some noise in Bsub and Ecol plots.

For instance, we can see that Ecol does not have high repetitive peaks of GC content in the genome, it is more or less homogeneous. Bsub also has a few high peaks of GC content. Moreover, the other species (Mgen and Mpne) have a lot of variability in its GC content.

On the other hand, we can see the analysis of k-mers for each specie 14. Since we know from the lectures, that 'Unique < Distinct < Total', we can say that this rule is applied in the results that we have found in the plot of the distribution of scores for each species.

As we can see in the Figure 4.1 14, Ecol and Bsub have more K-mers in comparison to Mgen and Mpne species, which may be because Ecol and bsub genomes are larger. This can be also seen in the chaos plot 5, since they have more dark boxes they should have more variation of nucleotides in its genome.

The size of the k-mers seems to be visually significant since Ecol and Bsub have more k-mers of size 15 and 20 compared to all other species.

In conclusion, we have seen that Ecol and Bsub are more complex organisms than Mgen and Mpne, Ecol and Bsub have more homogeneous GC content and more variability in its genome sequence. Moreover, Mgen and Mpne are more simple species, less homogeneous, and with less variability in its genome sequence.

4 Appendices

4.1 Software

We have used the following versions:

```
uname -a
# Linux aleph 4.15.0-118-generic #119-Ubuntu SMP
# Tue Sep 8 12:30:01 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux

wget --version
# GNU Wget 1.20.1 built on linux-gnu.

infoseq -version
# EMBOSS:6.6.0.0

jellyfish -V
# jellyfish 2.2.8

R --version
# R version 3.5.2 (2018-12-20) -- "Eggshell Igloo"
# Copyright (C) 2018 The R Foundation for Statistical Computing
# Platform: x86_64-pc-linux-gnu (64-bit)
```

4.2 Supplementary files

4.2.1 Project specific scripts

bin/runpipeline.sh

```
#!/bin/bash
#run:
#. runpipeline.sh Ecol Bsub Mgen Mpne

args="$@"

export BIN=$WDR/bin
export DT=$WDR/data
echo $args
# Be sure to define correctly the genomes in ret_refgenomes.sh
#
echo "RETREIVE GENOMES"
. $WDR/bin/ret_refgenomes.sh
echo ""


echo "RAW FORMAT (FASTA)"
for SPC in $args;
do
  . $WDR/bin/raw_seq.sh $SPC
done
echo ""


echo "CHAOS-PLOT"
for SPC in $args;
do
  . $WDR/bin/chaos-plot.sh $SPC
done
echo ""

# provide execution permissions to the perl script
chmod a+x $WDR/bin/genomicgcwindows.pl

echo "Genomic GC Windows"
```

```
for SPC in $args;
do
  for w in 100 200 500 1000 2000 5000 10000;
  do
    echo "Windows analysis of "$SPC" with windows length "$w;
    zcat $DT/${SPC}_referencegenome.fa.gz | \
    $WDR/bin/genomicgcwindows.pl $w - | \
    gzip -c9 -> $WDR/stats/${SPC}_genomegcanalysis_wlen$w.tbl.gz;
  done;
done;
echo ""

echo "Checking output of Genomic GC Windows"
for SPC in $args;
do
  ls -1 $WDR/stats/${SPC}_genomegcanalysis_wlen*.tbl.gz | \
    while read FL;
      do {
        echo $FL;
        zcat $FL | head -2;
      }; done;
done;
echo ""

function jellyfish_on_kmer () {
  THYSPC=$1;
  KMERSZ=$2;
  echo "# ${THYSPC} - ${KMERSZ}" 1>&2;
  zcat $DT/${THYSPC}_referencegenome.fa.gz | \
    jellyfish count -m $KMERSZ -C -t 4 -c 8 -s 10000000 /dev/fd/0 \
                -o $WDR/stats/${SPC}_jellyfish_k${KMERSZ}.counts;
  jellyfish stats $WDR/stats/${SPC}_jellyfish_k${KMERSZ}.counts;
  jellyfish stats $WDR/stats/${SPC}_jellyfish_k${KMERSZ}.counts >> $WDR/stats/jellyfish_total.count;
}


echo "Analysis of k-mer composition"
for SPC in $args;
do
  for KSZ in 10 15 20;
  do
    echo "Analysis of "$SPC" K-mers with KSZ =" $KSZ;
    jellyfish_on_kmer $SPC $KSZ;
  done;
done;
```

bin/genomicgcwindows.pl

```
#!/usr/bin/perl
#
# you can save this script as "bin/genomicgcwindows.pl"
#
use strict;
use warnings;

# variables initialization
my $window = shift @ARGV;
$window < 10 && die("# ERROR: window length must be a positive integer equal or greater than 10\n");
my $step = int($window / 2); # we have chosen to fix this parameter
my %SEQS = (); # just in case there is more than one sequence on the input

# read sequences
my $sid = undef;
while (<>) {
    next if /^\s*$/o;
    chomp;
    $_ =~ /^>/ && do {  # finding the sequence header with its name
```

```
        ($sid, undef) = split /\s+/, $_;
        exists($SEQS{$sid}) || ($SEQS{$sid} = '');
        next;
    };
    defined($sid) || next;
    $_ =~ s/\s+//og; # just in case there are white spaces on the sequence
    $SEQS{$sid} .= uc($_);
}; # while $_

# analyze sequences
foreach my $sid (keys %SEQS) {
    my $seq = $SEQS{$sid};
    for (my $n = 0; $n < length($seq) - $window + 1; $n += $step) {
        my $winseq = substr($seq, $n, $window);
        printf "%s %d %.1f\n", $sid, $n + $step, &getGC(\$winseq,$window);
    };
}; # foreach $sid

exit(0);

# available functions
sub getGC() {
    my ($sq, $wn) = @_;
    my $gc = 0;
    for (my $c = 0; $c < $wn; $c++) {
        $gc++ if substr($$sq, $c, 1) =~ /[GC]/o;
    }; # for $c
    return $gc / $wn * 100;
} # getGC
```

bin/chaos-plot.sh

```
#!/bin/bash

SPC=("$@")

echo "Chaos Plot for "$SPC
echo "Length: " ${SPC}_referencegenome.gb.gz
zcat $DT/${SPC}_referencegenome.fa.gz | \
  chaos -sequence fasta::stdin -verbose \
        -graph png -gtitle "$SPC chaos plot" \
        -goutfile $WDR/images/${SPC}_chaosplot
```

bin/ret_refgenomes.sh

```
#!/bin/bash

GBFTP=https://compgen.bio.ub.edu/~jabril/teaching/BScBI-CG2021/repo_ex2


while read Ospc Gftp;
  do {
    echo "# Downloading genome sequence for $Ospc" 1>&2;
    wget --user="bioinformatics" \
         --password="atg01tga" \
         $GBFTP/${Ospc}_referencegenome.gb.gz \
         -O $DT/${Ospc}_referencegenome.gb.gz
  }; done <<'EOF'
Ecol GCF_000005845.2_ASM584v2
Bsub GCF_000009045.1_ASM904v1
Mgen GCF_000027325.1_ASM2732v1
Mpne GCF_000027345.1_ASM2734v1
EOF
```

bin/raw_seq.sh

```
#!/bin/bash

SPC=("$@")


echo "Converting: " $SPC
zcat $DT/${SPC}_referencegenome.gb.gz | \
  seqret -sequence genbank::stdin -outseq fasta::stdout | \
    gzip -9c - > $DT/${SPC}_referencegenome.fa.gz

echo ""
echo "CHECKING LENGTH SEQUENCES: "

echo "Length: " ${SPC}_referencegenome.gb.gz
zegrep '^LOCUS'  $DT/${SPC}_referencegenome.gb.gz
echo "Length GenBank file: " ${SPC}_referencegenome.gb.gz
zcat $DT/${SPC}_referencegenome.fa.gz | \
  infoseq -sequence fasta::stdin \
          -noheading -only -name -length -pgc
echo ""
```

bin/R_plot_.r

```
#!/usr/bin/Rscript
#give permisions:
#chmod +x R_plot_.r
#execute:
#./R_plot_.r Ecol Bsub Mgen Mpne

args = commandArgs(trailingOnly=TRUE)
#getwd()
dir <- setwd('/home/adria/Escritorio/Computational_genomics/exercise_02/stats/')
for (SPC in args){
  if (SPC == "Ecol"){
    GC_avg <- 50.79;}
  else if (SPC=="Bsub"){
    GC_avg <- 43.5;}
  else if (SPC == "Mgen"){
    GC_avg <- 31.7;}
  else if (SPC == "Mpne"){
    GC_avg <- 40.0;
  }

  for (w in c(100, 200, 500, 1000, 2000, 5000, 10000)){
    #GC_avg <- 50.79; # the whole genome average GC content
    d <- '/home/adria/Escritorio/Computational_genomics/exercise_02/stats/'
    middle = '_genomegcanalysis_wlen'
    extension = '.tbl.gz'
    path<-(paste(d,as.character(SPC),middle,as.character(w),extension, sep=""))
    print(path)
    ZZ <- gzfile(path);
    GC_w <- read.table(ZZ, header=FALSE);
    colnames(GC_w) <- c("CHRid","NUCpos","GCpct");

    summary(GC_w)

    library(ggplot2);

    G <- ggplot(GC_w, aes(x=NUCpos, y=GCpct)) +
      geom_line(colour = "blue") +
      theme_bw() +
      geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", size=1.5) +
      ggtitle(paste(SPC,"GC content over the genome (window length = ",w,"bp)")) +
      labs(x="Genomic Coords (bp)", y="%GC Content");
    dirs <-'/home/adria/Escritorio/Computational_genomics/exercise_02/images/'
```

```
    mid2 <- '_genomegcanalysis_wlen'
    ex <-'.png'
    p <- (paste(dirs,SPC,mid2,w,ex,sep=''))
    ggsave(p,plot=G, width=25, height=8, units="cm", dpi=600);

  }
}
```

4.2.2 Shell global vars and settings for this project

projectvars.sh

```
## ##############################################################################################
##
##   projectvars.sh
##
##   A BASH initialization file for BScBI-CG practical exercise folders
##
## ##############################################################################################
##
##                 CopyLeft 2020 (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="CABELLO_ADRIA";                   #-> IMPORTANT: SET YOUR SURNAME and NAME ON THIS VAR,
RB="README_BScBICG2021_exercise02"; #->            MUST FIX ON MARKDOWN README FILE
                                      #->            FROM TARBALL (AND INSIDE TOO)
RD="${RB}_${NM}";
PDOCFLGS='markdown+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;

function ltx2pdf () {
    RF=$1;
    pdflatex $RF.tex;
    bibtex $RF;
    pdflatex $RF.tex;
    pdflatex $RF.tex;
}
```

```
alias runpandoc='\
  pandoc -f $PDOCFLGS               \
         --template=$PDOCTPL        \
         -t latex --natbib          \
         --number-sections          \
         --highlight-style pygments \
         -o $RD.tex $RD.md;         \
  ltx2pdf $RD';
#
### IMPORTANT ###
#
#   MacOSX users must remember to set the previous alias
#   in a single line for the command "runpandoc" to work, as follows
#   (just uncomment it, and comment the previous alias definition):
#
# alias runpandoc='pandoc -f $PDOCFLGS --template=$PDOCTPL -t latex --natbib --number-sections --highlight-style py

#
# 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`).