BScBI-CG
1.25ex Practicals
1.25ex Report

Maria Cobo

Exercise 2

— October 17, 2024—

ESCI upf. School of International Studies

Computational Genomics 2024–25

BSc in Bioinformatics

UPF/ESCI – UPC – UB – UAB

Contents

1 Introduction **1**

1.1 Objectives . . . 1

1.2 Prerequisites . . . 1

1.2.1 Installing required software . . . 1

1.2.2 Initializing the main report files . . . 3

2 Calculating Genome Sequence Properties **4**

2.1 Datasets . . . 4

2.2 Retrieving the sequences . . . 5

2.3 Summary of sequence content . . . 7

2.3.1 Chaos-plot . . . 7

2.3.2 Computing GC content variation across the genome . . . 8

2.4 Analysis of *k*-mer composition . . . 16

3 Discussion **18**

4 Appendices **19**

4.1 Software . . . 19

4.2 Supplementary files . . . 19

4.2.1 `conda` environment dependencies for the exercise . . . 19

4.2.2 Project specific scripts . . . 20

4.2.3 Shell global vars and settings for this project . . . 20

4.3 About this document . . . 21

List of Tables

1 Genome sequence information for four bacteria species downloaded from GenBank . . . 4

2 Summary of k-mer statistics for four species genomes using different k-mer sizes . . . 18

List of Figures

1 Chaos plots for the four species . . . 8

2 Comparison of window sizes of GC content in E.coli . . . 14

3 Comparison of GC content for the four species with window size 2000bp . . . 16

1 Introduction

We want to analyze genome sequences of four bacteria species: *Escherichia coli*, *Clostridium botulinum*, *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. Remind that we assume that you are using a Debian-based linux distribution, so we will show only the corresponding set of commands for that distribution.

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** (refer to exercises, `00` and `01`, as well as the Aula Global tutorials).

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

1.2.1.1 Using `conda/mamba` environments: As we saw in the previous exercises, 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 19),

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_CG2425_exercise_02.tgz
cd exercise_02

# Rename report file including your "NAME" and "SURNAME"
mv -v README_BScBICG2425_exercise02_SURNAME_NAME.md \
      README_BScBICG2425_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_BScBICG2425_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	Level	RefSeq ID	INSDC	Size (Mb)	GC%	Proteins	rRNAs	tRNAs	Other RNAs	Pseudo genes	Total Genes
Escherichia coli	Chr	NC_000913.3	U00096.3	4.64	50.8	4,288	22	86	120	145	4,661
Clostridium botulinum	Chr	NC_009495.1	AM412317.1	3.89	28.0	3,545	27	80	31	41	3,724
Mycoplasma genitalium	Chr	NC_000908.2	L43967.2	0.58	31.7	504	3	36	3	17	563
Mycoplasma pneumoniae	Chr	NC_000912.1	U00089.2	0.82	40.0	687	3	37	7	41	775

Table 1: **Genome sequence information for four bacteria species downloaded from GenBank.** Whole-genome summary table showing number of annotated gene features (protein-coding genes, rRNAs, tRNAs, other RNA genes, and pseudogenes), along with sequence characteristics, such as genome size, average GC content, or assembly level (all assemblies at finished chromosome level).

Table 1 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 (assembly 'ASM584v2').
- *C. botulinum* is the bacteria that produces one of the most potent toxin known to mankind, natural or synthetic, with a lethal dose of 1.3–2.1 ng/kg in humans. Representative genome for this species is *C. botulinum* A strain ATCC 3502 (assembly 'ASM6358v1').
- 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 (assembly 'ASM2732v1').
- *M. pneumoniae* causes respiratory tract infections. We are going to use *M. pneumoniae* M129 as representative genome (assembly 'ASM2734v1').

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=https://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:
#
#  Cbot GCF/000/063/585/GCF_000063585.1_ASM6358v1/GCF_000063585.1_ASM6358v1_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 Cbot_referencegenome.gb.gz, Mgen_referencegenome.gb.gz,
# and Mpne_referencegenome.gb.gz respectively.
```

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

```
# 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'
Cbot GCF/000/063/585/GCF_000063585.1_ASM6358v1/GCF_000063585.1_ASM6358v1
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
EOF

### IMPORTANT NOTE ###
#
# If the firewall does not allow you to connect to the NCBI https 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-CG2425/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
Cbot GCF_000063585.1_ASM6358v1
Mgen GCF_000027325.1_ASM2732v1
Mpne GCF_000027345.1_ASM2734v1
EOF

### YET ANOTHER WAY TO GET THE SEQUENCE FILES ###
#
# Using curl command we can download same datasets as zip files.
# This zip file will contain a folder "ncbi_dataset/data/GCF_0000xxxxx.x/"
# where you can find a "*.fna" file with the genome nucleotide sequence
# in FASTA format instead of GenBank (thus, it lacks the annotations).
#
pushd $DT;
PRE='https://api.ncbi.nlm.nih.gov/datasets/v1/genome/accession/';
PST='/download?include_annotation_type=GENOME_GFF,RNA_FASTA,CDS_FASTA,PROT_FASTA&filename=';
for GFL in GCF_000005845.2 GCF_000063585.1 GCF_000027325.1 GCF_000027345.1;
  do {
    curl -OJX GET "$PRE$GFL$PST$GFL.zip" -H "Accept: application/zip";
  }; done;
popd;
```

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 09-MAR-2022

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 ###
#
### --- IMPORTANT ---
###
### Take care that the Cbot genbank file provides the cromosome
###   and a plasmid sequence, you should discard the later.

SPC="Cbot"

zcat $DT/${SPC}_referencegenome.gb.gz | \
  awk '
  BEGIN {in_sequence=0}
  /^LOCUS/ {
    if ($2 == "NC_009495") {in_sequence=1}
    else {in_sequence=0}
  }
  {if (in_sequence) print}
  ' > $DT/${SPC}_referencegenome.gb

#Convert the extracted chromosome from GenBank format to FASTA format using seqret
seqret -sequence $DT/${SPC}_referencegenome.gb -outseq fasta::stdout | \
  gzip -9c - > $DT/${SPC}_referencegenome.fa.gz

# Verify if FASTA sequence has the same length as reported in GenBank
zegrep '^LOCUS'  data/Cbot_referencegenome.gb
# LOCUS       NC_009495            3886916 bp    DNA     circular CON 11-APR-2024

zcat $DT/${SPC}_referencegenome.fa.gz | \
  infoseq -sequence fasta::stdin \
          -noheading -only -name -length -pgc
# Display basic information about sequences
# NC_009495      3886916 28.24

SPC="Mgen"

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


zegrep '^LOCUS'  $DT/${SPC}_referencegenome.gb.gz
# LOCUS       NC_000908             580076 bp    DNA     circular CON 28-FEB-2024

zcat $DT/${SPC}_referencegenome.fa.gz | \
  infoseq -sequence fasta::stdin \
          -noheading -only -name -length -pgc
# Display basic information about sequences
# NC_000908      580076 31.69
```

```
SPC="Mpne"

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


zegrep '^LOCUS'  $DT/${SPC}_referencegenome.gb.gz
# LOCUS       NC_000912             816394 bp    DNA     circular CON 26-FEB-2024


zcat $DT/${SPC}_referencegenome.fa.gz | \
  infoseq -sequence fasta::stdin \
          -noheading -only -name -length -pgc
# Display basic information about sequences
# NC_000912      816394 40.01
```

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 1, 50.79% versus 50.8% respectively (so the difference is due to rounding to one decimal position).

2.3 Summary of sequence content

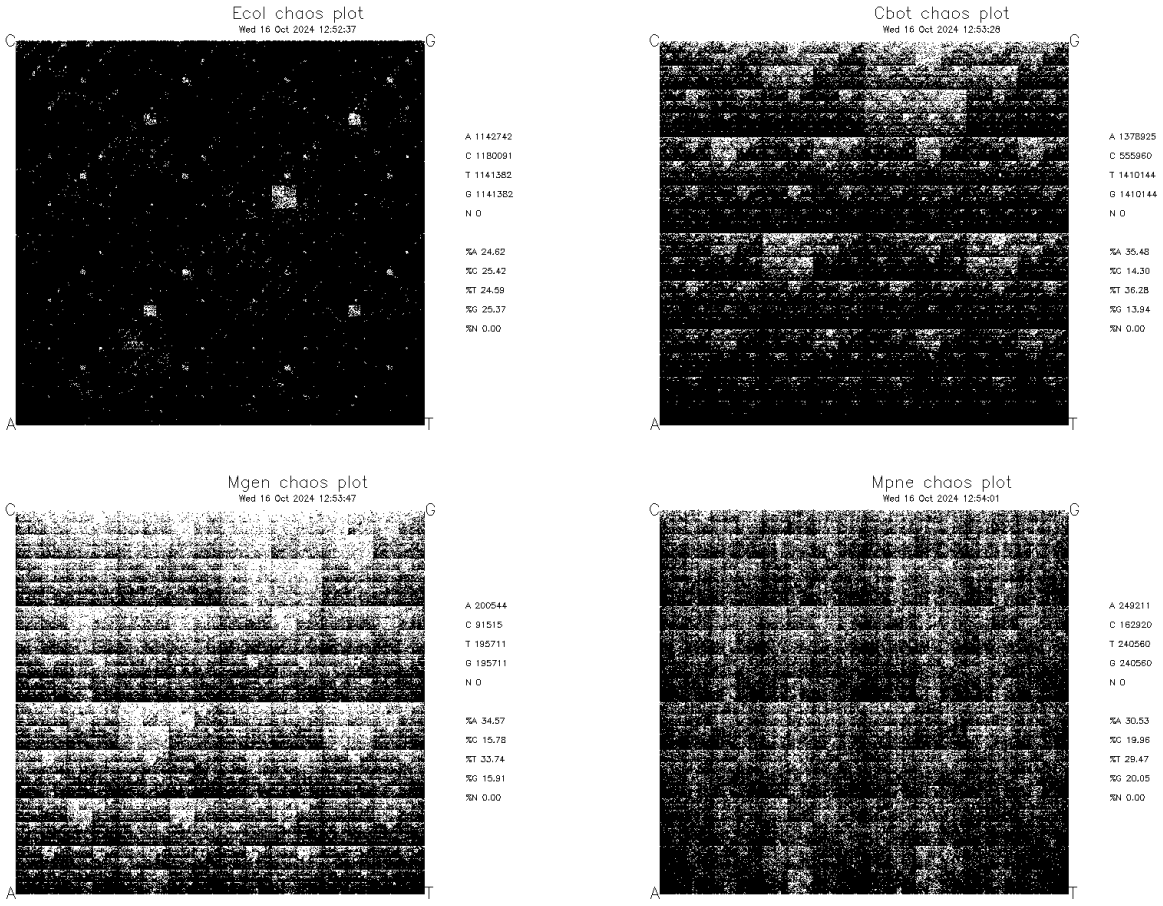
2.3.1 Chaos-plot

`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.

```
SPC="Ecol"

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 ###
SPC="Cbot"

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

SPC="Mgen"

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

SPC="Mpne"

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

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`").

Figure 1: Chaos plots for the four species



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
SPC="Ecol"

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 ###

SPC="Cbot"

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/Cbot_genomegcanalysis_wlen10000.tbl.gz
#> >NC_009495 5000 29.3
#> >NC_009495 10000 38.7
#> stats/Cbot_genomegcanalysis_wlen1000.tbl.gz
#> >NC_009495 500 26.9
#> >NC_009495 1000 26.2
#> stats/Cbot_genomegcanalysis_wlen100.tbl.gz
#> >NC_009495 50 30.0
#> >NC_009495 100 27.0
#> stats/Cbot_genomegcanalysis_wlen2000.tbl.gz
#> >NC_009495 1000 26.5
#> >NC_009495 2000 25.0
#> stats/Cbot_genomegcanalysis_wlen200.tbl.gz
#> >NC_009495 100 29.5
#> >NC_009495 200 23.0
#> stats/Cbot_genomegcanalysis_wlen5000.tbl.gz
#> >NC_009495 2500 26.3
#> >NC_009495 5000 28.7
#> stats/Cbot_genomegcanalysis_wlen500.tbl.gz
#> >NC_009495 250 28.0
#> >NC_009495 500 26.2


SPC="Mgen"

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/Mgen_genomegcanalysis_wlen10000.tbl.gz
#> >NC_000908 5000 29.4
#> >NC_000908 10000 28.6
#> stats/Mgen_genomegcanalysis_wlen1000.tbl.gz
#> >NC_000908 500 17.7
#> >NC_000908 1000 21.6
#> stats/Mgen_genomegcanalysis_wlen100.tbl.gz
#> >NC_000908 50 16.0
#> >NC_000908 100 16.0
#> stats/Mgen_genomegcanalysis_wlen2000.tbl.gz
#> >NC_000908 1000 21.5
#> >NC_000908 2000 26.9
#> stats/Mgen_genomegcanalysis_wlen200.tbl.gz
#> >NC_000908 100 14.5
#> >NC_000908 200 15.0
#> stats/Mgen_genomegcanalysis_wlen5000.tbl.gz
#> >NC_000908 2500 28.3
#> >NC_000908 5000 33.2
#> stats/Mgen_genomegcanalysis_wlen500.tbl.gz
#> >NC_000908 250 16.2
#> >NC_000908 500 16.8


SPC="Mpne"

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/Mpne_genomegcanalysis_wlen10000.tbl.gz
#> >NC_000912 5000 42.2
#> >NC_000912 10000 39.9
#> stats/Mpne_genomegcanalysis_wlen1000.tbl.gz
#> >NC_000912 500 26.7
#> >NC_000912 1000 30.5
#> stats/Mpne_genomegcanalysis_wlen100.tbl.gz
#> >NC_000912 50 36.0
#> >NC_000912 100 30.0
#> stats/Mpne_genomegcanalysis_wlen2000.tbl.gz
#> >NC_000912 1000 31.5
#> >NC_000912 2000 39.4
```

```
#> stats/Mpne_genomegcanalysis_wlen200.tbl.gz
#> >NC_000912 100 28.5
#> >NC_000912 200 19.5
#> stats/Mpne_genomegcanalysis_wlen5000.tbl.gz
#> >NC_000912 2500 39.3
#> >NC_000912 5000 45.1
#> stats/Mpne_genomegcanalysis_wlen500.tbl.gz
#> >NC_000912 250 26.4
#> >NC_000912 500 25.0
```

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('/home/maria/practicals/exercise_02/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", linewidth=1.5) +
        ggtitle("E.coli GC content over the genome (window length = 100bp)") +
        labs(x="Genomic Coords (bp)", y="%GC Content");

ggsave("/home/maria/practicals/exercise_02/images/Ecol_genomegcanalysis_wlen100.png",
       plot=G, width=25, height=8, units="cm", dpi=600);
```

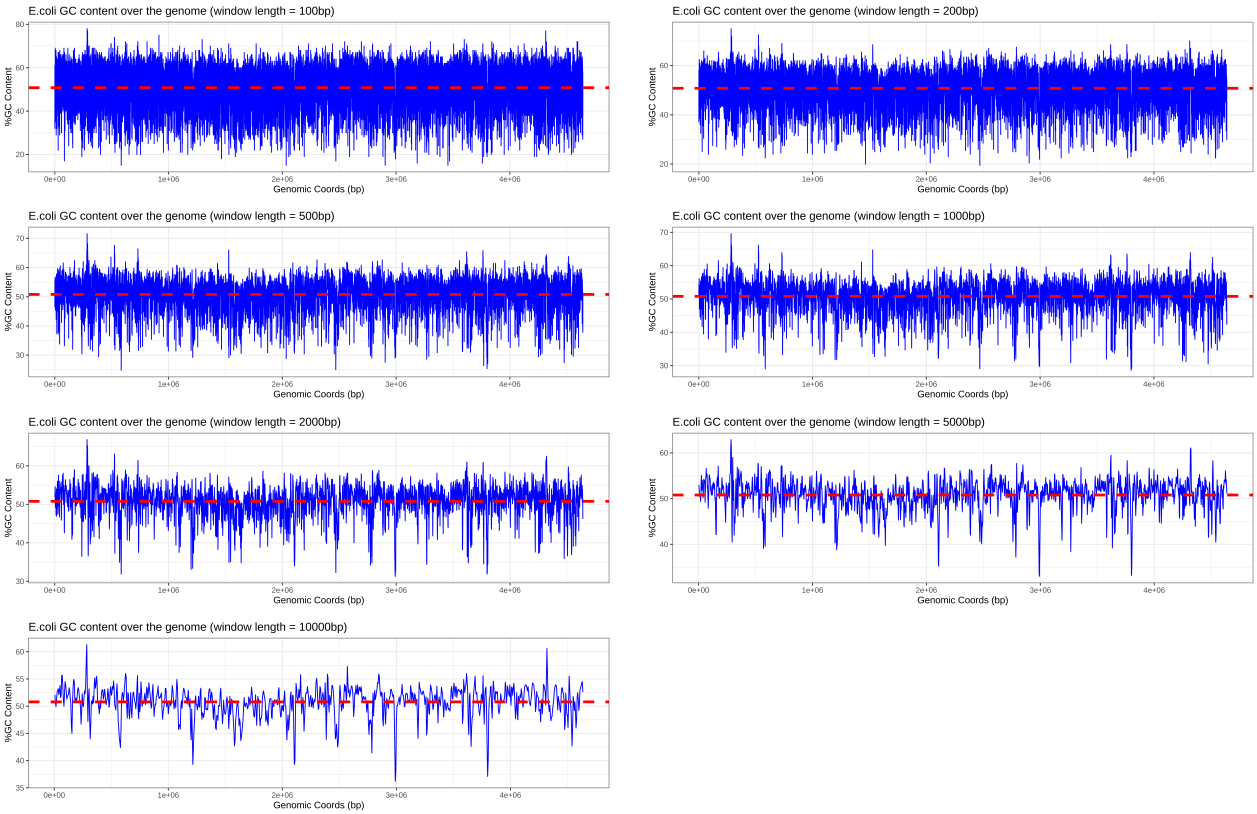
Include here a figure combining the plots for the set of window lengths (100, 200, 500, 1000, 2000, 5000, and 10000). Then, **choose one of those windows lengths** and provide the commands to analyze the other three genomic sequences. After that, you can include another figure stacking the results for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

```
### DIFFERENT WINDOW SIZES ###
GC_avg <- 50.79; # the whole genome average GC content

### 200 ###
ZZ_200 <- gzfile('/home/maria/practicals/exercise_02/stats/Ecol_genomegcanalysis_wlen200.tbl.gz');
GC_w200 <- read.table(ZZ_200, header=FALSE);
colnames(GC_w200) <- c("CHRid","NUCpos","GCpct");

G_200 <- ggplot(GC_w200, aes(x=NUCpos, y=GCpct)) +
        geom_line(colour = "blue") +
        theme_bw() +
```

```
        geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", linewidth=1.5) +
        ggtitle("E.coli GC content over the genome (window length = 200bp)") +
        labs(x="Genomic Coords (bp)", y="%GC Content");

ggsave("/home/maria/practicals/exercise_02/images/Ecol_genomegcanalysis_wlen200.png",
       plot=G_200, width=25, height=8, units="cm", dpi=600);


### 500 ###
ZZ_500 <- gzfile('/home/maria/practicals/exercise_02/stats/Ecol_genomegcanalysis_wlen500.tbl.gz');
GC_w500 <- read.table(ZZ_500, header=FALSE);
colnames(GC_w500) <- c("CHRid","NUCpos","GCpct");

G_500 <- ggplot(GC_w500, aes(x=NUCpos, y=GCpct)) +
        geom_line(colour = "blue") +
        theme_bw() +
        geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", linewidth=1.5) +
        ggtitle("E.coli GC content over the genome (window length = 500bp)") +
        labs(x="Genomic Coords (bp)", y="%GC Content");

ggsave("/home/maria/practicals/exercise_02/images/Ecol_genomegcanalysis_wlen500.png",
       plot=G_500, width=25, height=8, units="cm", dpi=600);


### 1000 ###
ZZ_1000 <- gzfile('/home/maria/practicals/exercise_02/stats/Ecol_genomegcanalysis_wlen1000.tbl.gz');
GC_w1000 <- read.table(ZZ_1000, header=FALSE);
colnames(GC_w1000) <- c("CHRid","NUCpos","GCpct");

G_1000 <- ggplot(GC_w1000, aes(x=NUCpos, y=GCpct)) +
        geom_line(colour = "blue") +
        theme_bw() +
        geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", linewidth=1.5) +
        ggtitle("E.coli GC content over the genome (window length = 1000bp)") +
        labs(x="Genomic Coords (bp)", y="%GC Content");

ggsave("/home/maria/practicals/exercise_02/images/Ecol_genomegcanalysis_wlen1000.png",
       plot=G_1000, width=25, height=8, units="cm", dpi=600);


### 2000 ###
ZZ_2000 <- gzfile('/home/maria/practicals/exercise_02/stats/Ecol_genomegcanalysis_wlen2000.tbl.gz');
GC_w2000 <- read.table(ZZ_2000, header=FALSE);
colnames(GC_w2000) <- c("CHRid","NUCpos","GCpct");

G_2000 <- ggplot(GC_w2000, aes(x=NUCpos, y=GCpct)) +
        geom_line(colour = "blue") +
        theme_bw() +
        geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", linewidth=1.5) +
        ggtitle("E.coli GC content over the genome (window length = 2000bp)") +
        labs(x="Genomic Coords (bp)", y="%GC Content");

ggsave("/home/maria/practicals/exercise_02/images/Ecol_genomegcanalysis_wlen2000.png",
       plot=G_2000, width=25, height=8, units="cm", dpi=600);


### 5000 ###
ZZ_5000 <- gzfile('/home/maria/practicals/exercise_02/stats/Ecol_genomegcanalysis_wlen5000.tbl.gz');
GC_w5000 <- read.table(ZZ_5000, header=FALSE);
colnames(GC_w5000) <- c("CHRid","NUCpos","GCpct");

G_5000 <- ggplot(GC_w5000, aes(x=NUCpos, y=GCpct)) +
        geom_line(colour = "blue") +
```

```
        theme_bw() +
        geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", linewidth=1.5) +
        ggtitle("E.coli GC content over the genome (window length = 5000bp)") +
        labs(x="Genomic Coords (bp)", y="%GC Content");

ggsave("/home/maria/practicals/exercise_02/images/Ecol_genomegcanalysis_wlen5000.png",
       plot=G_5000, width=25, height=8, units="cm", dpi=600);


### 10000 ###
ZZ_10000 <- gzfile('/home/maria/practicals/exercise_02/stats/Ecol_genomegcanalysis_wlen10000.tbl.gz');
GC_w10000 <- read.table(ZZ_10000, header=FALSE);
colnames(GC_w10000) <- c("CHRid","NUCpos","GCpct");

G_10000 <- ggplot(GC_w10000, aes(x=NUCpos, y=GCpct)) +
        geom_line(colour = "blue") +
        theme_bw() +
        geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", linewidth=1.5) +
        ggtitle("E.coli GC content over the genome (window length = 10000bp)") +
        labs(x="Genomic Coords (bp)", y="%GC Content");

ggsave("/home/maria/practicals/exercise_02/images/Ecol_genomegcanalysis_wlen10000.png",
       plot=G_10000, width=25, height=8, units="cm", dpi=600);
```

Figure 2: Comparison of window sizes of GC content in E.coli

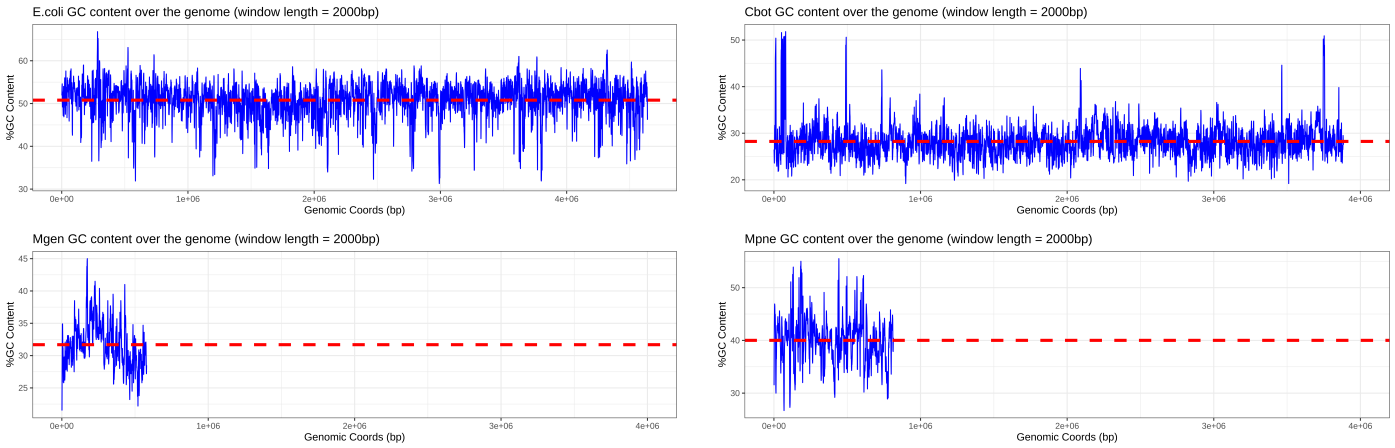


I believe that the best window sizes are between the 1000bp and 2000bp as these are not either too noisy or too smooth like the other window sizes. For the next plots I will chose window size of 2000bp as I believe it does not lose variation as it is a balance between smoothness and noise.

```
### CBOT ###
GC_avg <- 28.24;
ZZ_Cbot <- gzfile('/home/maria/practicals/exercise_02/stats/Cbot_genomegcanalysis_wlen2000.tbl.gz');
GC_w2000_Cbot <- read.table(ZZ_Cbot, header=FALSE);
```

```
colnames(GC_w2000_Cbot) <- c("CHRid","NUCpos","GCpct");

G_Cbot <- ggplot(GC_w2000_Cbot, aes(x=NUCpos, y=GCpct)) +
        geom_line(colour = "blue") +
        theme_bw() +
        geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", linewidth=1.5) +
        ggtitle("Cbot GC content over the genome (window length = 2000bp)") +
        labs(x="Genomic Coords (bp)", y="%GC Content") + xlim(0, 4e+06);

ggsave("/home/maria/practicals/exercise_02/images/Cbot_genomegcanalysis_wlen2000.png",
       plot=G_Cbot, width=25, height=8, units="cm", dpi=600);

### MGEN ###
GC_avg <- 31.69;
ZZ_Mgen <- gzfile('/home/maria/practicals/exercise_02/stats/Mgen_genomegcanalysis_wlen2000.tbl.gz');
GC_w2000_Mgen <- read.table(ZZ_Mgen, header=FALSE);
colnames(GC_w2000_Mgen) <- c("CHRid","NUCpos","GCpct");

G_Mgen <- ggplot(GC_w2000_Mgen, aes(x=NUCpos, y=GCpct)) +
        geom_line(colour = "blue") +
        theme_bw() +
        geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", linewidth=1.5) +
        ggtitle("Mgen GC content over the genome (window length = 2000bp)") +
        labs(x="Genomic Coords (bp)", y="%GC Content") + xlim(0, 4e+06);

ggsave("/home/maria/practicals/exercise_02/images/Mgen_genomegcanalysis_wlen2000.png",
       plot=G_Mgen, width=25, height=8, units="cm", dpi=600);

### MPNE ###
GC_avg <- 40.01;
ZZ_Mpne <- gzfile('/home/maria/practicals/exercise_02/stats/Mpne_genomegcanalysis_wlen2000.tbl.gz');
GC_w2000_Mpne <- read.table(ZZ_Mpne, header=FALSE);
colnames(GC_w2000_Mpne) <- c("CHRid","NUCpos","GCpct");

G_Mpne <- ggplot(GC_w2000_Mpne, aes(x=NUCpos, y=GCpct)) +
        geom_line(colour = "blue") +
        theme_bw() +
        geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", linewidth=1.5) +
        ggtitle("Mpne GC content over the genome (window length = 2000bp)") +
        labs(x="Genomic Coords (bp)", y="%GC Content") + xlim(0, 4e+06);

ggsave("/home/maria/practicals/exercise_02/images/Mpne_genomegcanalysis_wlen2000.png",
       plot=G_Mpne, width=25, height=8, units="cm", dpi=600);
```

Figure 3: Comparison of GC content for the four species with window size 2000bp



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.

```
SPC="Ecol"

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 $WDR/stats/${THYSPC}_jellyfish_k${KMERSZ}.counts;
  jellyfish stats $WDR/stats/${THYSPC}_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 Cbot 35;
# # Cbot - 35
# Unique:    3811998
# Distinct:  3831374
# Total:     3886882
# Max_count: 20
#
#   or within loops...
#
for SPC in Ecol Cbot;
  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 (**IMPORTANT:** take caution with large k-mer sizes as they may require large amount of disk space and CPU time). You can take "`docs/tbl_genbank_summary_info_genomes.tex`" as example to create this table.

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

for SPC in Ecol Cbot Mgen Mpne;
  do {
    for KSZ in 10 15 20 25 30 35 40;
      do {
       jellyfish_on_kmer $SPC $KSZ;
      }; done;
  }; done;
```

Species	K-mer Size	Unique	Distinct	Total	Max Count
Escherichia coli	10	40,625	490,389	4,641,643	284
Escherichia coli	15	4,357,730	4,462,229	4,641,638	137
Escherichia coli	20	4,507,760	4,542,190	4,641,633	82
Escherichia coli	25	4,516,906	4,548,910	4,641,628	75
Escherichia coli	30	4,522,959	4,553,477	4,641,623	48
Escherichia coli	35	4,527,791	4,557,129	4,641,618	42
Escherichia coli	40	4,531,656	4,559,993	4,641,613	12
Clostridium botulinum	10	81,662	355,884	3,886,907	865
Clostridium botulinum	15	3,203,958	3,475,271	3,886,902	62
Clostridium botulinum	20	3,782,158	3,812,421	3,886,897	30
Clostridium botulinum	25	3,799,230	3,822,881	3,886,892	28
Clostridium botulinum	30	3,806,885	3,827,965	3,886,887	22
Clostridium botulinum	35	3,811,998	3,831,374	3,886,882	20
Clostridium botulinum	40	3,815,741	3,833,887	3,886,877	20
Mycoplasma genitalium	10	87,099	192,529	580,067	122
Mycoplasma genitalium	15	550,161	562,149	580,062	39
Mycoplasma genitalium	20	564,076	569,990	580,057	31
Mycoplasma genitalium	25	566,174	571,601	580,052	20
Mycoplasma genitalium	30	567,957	572,877	580,047	11
Mycoplasma genitalium	35	569,477	573,915	580,042	8
Mycoplasma genitalium	40	570,718	574,743	580,037	7
Mycoplasma pneumoniae	10	123,499	292,158	816,385	58
Mycoplasma pneumoniae	15	740,058	766,612	816,380	36
Mycoplasma pneumoniae	20	757,793	778,224	816,375	15
Mycoplasma pneumoniae	25	764,947	783,662	816,370	15
Mycoplasma pneumoniae	30	770,702	787,967	816,365	14
Mycoplasma pneumoniae	35	775,615	791,523	816,360	14
Mycoplasma pneumoniae	40	779,921	794,536	816,355	14

Table 2: **Summary of k-mer statistics for four species genomes using different k-mer sizes.** Table with metrics like the number of unique and distinct k-mers, total k-mers, and the maximum count of any specific k-mer across varying k-mer sizes (10 to 40)

3 Discussion

IMPORTANT Discuss your results here (around 300 words). And remember to include in the Appendices section (see page 19), 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.

To start, in our GC content plots, we are able to compare the genomic characteristics of our four organisms: *Escherichia coli*, *Clostridium botulinum*, *Mycoplasma genitalium*, and *Mycoplasma pneumoniae*. E.coli has the largest genome (~4.6 million bp) and the highest average GC content (~50%), with significant fluctuations across the genome. Next, we have C. botulinum which has a lower average of GC content (~30%) and a moderate variation over its genome size (similar to E.coli). Then, we have M.genitalium, with a smaller genome size (~1 million bp), this plot shows initial fluctuations in GC content but stabilizes whereas M. pneumoniae, with also a smaller genome size, has a relatively constant GC content with minial variation. To resume, out of these plots we can say that E.coli is the organism that shows the most dynamic GC content, while M. pneumoniae shows the most stable pattern, and both Mycoplasma species have significantly smaller genomes and lower GC contents compared to E. coli.

In our table results where we are seeing a k-mer analysis for our four organisms comparing metrics like the number of unique and distinct k-mers, total k-mers, and the maximum count of any specific k-mer across varying k-mer sizes (10 to 40). E. coli, with the largest genome, shows a consistent number of total k-mers (~4.64 million), and its unique k-mers slightly increase with k-mer size. The repetition in shorter k-mers is evident from the maximum count decreasing from 284 at k=10 to 28 at k=40. C. botulinum follows a similar pattern but with fewer total k-mers (~3.88 million) and a higher degree of short k-mer repetition (maximum count of 865 at k=10). In contrast, the smaller genomes of M. genitalium and M. pneumoniae result in significantly fewer total k-mers (~580,000 and ~816,000, respectively), with unique k-mers rising sharply as k-mer size increases. In particular, the maximum count of repetitive k-mers in these smaller genomes is much lower, reflecting less sequence complexity. Overall, larger genomes (E. coli and C. botulinum) show more k-mer diversity and repetition compared to the smaller Mycoplasma genomes.

In conclusion, the k-mer analysis supports the patterns observed in the GC content plots: species with larger genomes and more complex GC content patterns (like E. coli and C. botulinum) have more diverse and repetitive k-mers, while species with smaller, more stable genomes (like M. genitalium and M. pneumoniae) show less k-mer diversity and lower GC content variation.

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

jellyfish -V
# jellyfish 2.2.10

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
  - jellyfish
```

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_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;

#
### 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`).