BScBI-CG

Practicals Report

Jan Izquierdo

Exercise 02

— **October 17, 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 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 16),

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


wget https://ftp.ncbi.nlm.nih.gov/genomes/all/$dr -O data/$file


# 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
```

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

```
#  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.
# 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;
```

2.1.0.1 **Exercise 1** File download

```
#%
#Obtaining the files
```

```
dr=GCF/000/063/585/GCF_000063585.1_ASM6358v1/GCF_000063585.1_ASM6358v1_genomic.gbff.gz
file=Cbot_referencegenome.gb.gz
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/$dr -O data/$file
dr=GCF/000/005/845/GCF_000005845.2_ASM584v2/GCF_000005845.2_ASM584v2_genomic.gbff.gz
file=Ecol_referencegenome.gb.gz
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/$dr -O data/$file
dr=GCF/000/027/325/GCF_000027325.1_ASM2732v1/GCF_000027325.1_ASM2732v1_genomic.gbff.gz
file=Mgen_referencegenome.gb.gz
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/$dr -O data/$file
dr=GCF/000/027/345/GCF_000027345.1_ASM2734v1/GCF_000027345.1_ASM2734v1_genomic.gbff.gz
file=Mpne_referencegenome.gb.gz
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/$dr -O data/$file
#-%#
```

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 data/${SPC}_referencegenome.gb.gz|seqret -sequence genbank::stdin -outseq fasta::stdout | gzip -9c -

# 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.
```

2.2.0.1 **Exercise 2** Sequence extraction

```
#%
#Extracting the sequences
SPC=Cbot
zcat data/${SPC}_referencegenome.gb.gz|seqret -sequence genbank::stdin -outseq fasta::stdout | gzip -9c -
SPC=Ecol
zcat data/${SPC}_referencegenome.gb.gz|seqret -sequence genbank::stdin -outseq fasta::stdout | gzip -9c -
SPC=Mgen
zcat data/${SPC}_referencegenome.gb.gz|seqret -sequence genbank::stdin -outseq fasta::stdout | gzip -9c -
SPC=Mpne
zcat data/${SPC}_referencegenome.gb.gz|seqret -sequence genbank::stdin -outseq fasta::stdout | gzip -9c -
#-%#
```

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

2.3.1.1 Exercise 3 `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=Cbot
zcat data/${SPC}_referencegenome.fa.gz | chaos -sequence fasta::stdin -verbose -graph png -gtitle "$SPC ch
SPC=Ecol
zcat data/${SPC}_referencegenome.fa.gz | chaos -sequence fasta::stdin -verbose -graph png -gtitle "$SPC ch
SPC=Mgen
zcat data/${SPC}_referencegenome.fa.gz | chaos -sequence fasta::stdin -verbose -graph png -gtitle "$SPC ch
SPC=Mpne
zcat data/${SPC}_referencegenome.fa.gz | chaos -sequence fasta::stdin -verbose -graph png -gtitle "$SPC ch
#-%#
```

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`").

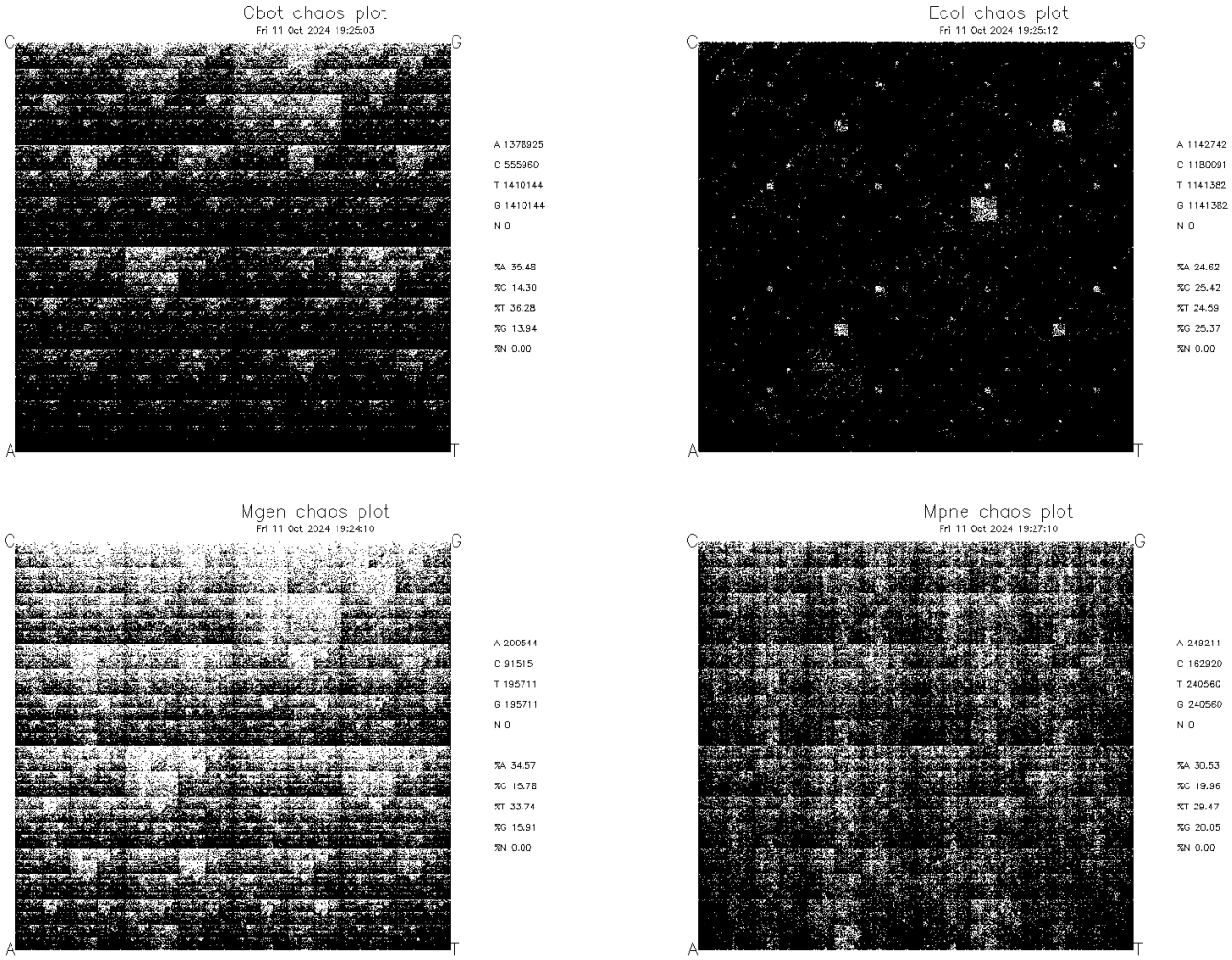

Figure 1: **Chaos plots: graphical representation of sequence composition.**

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
```

2.3.2.1 **Exercise 4** Creation of files for the various window lengths

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

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

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 ./data/${SPC}_referencegenome.fa.gz | \
      ./bin/genomicgcwindows.pl $WNDW - | \
        gzip -c9 - > ./stats/${SPC}_genomegcanalysis_wlen$WNDW.tbl.gz;
  }; done;

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

File check

```
#%
#Checking file conversion
### repeat the commands for the other three genomes

SPC=Cbot
ls -1 ./stats/${SPC}_genomegcanalysis_wlen*.tbl.gz | \
  while read FL;
    do {
      echo $FL;
      zcat $FL | head -2;
    }; done;

SPC=Ecol
ls -1 ./stats/${SPC}_genomegcanalysis_wlen*.tbl.gz | \
  while read FL;
    do {
      echo $FL;
      zcat $FL | head -2;
    }; done;

SPC=Mgen
ls -1 ./stats/${SPC}_genomegcanalysis_wlen*.tbl.gz | \
  while read FL;
    do {
      echo $FL;
      zcat $FL | head -2;
    }; done;

SPC=Mpne
ls -1 ./stats/${SPC}_genomegcanalysis_wlen*.tbl.gz | \
  while read FL;
    do {
      echo $FL;
      zcat $FL | head -2;
    }; done;
#-%#
```

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.

```
#In R
# then assuming you use R command-line shell from the terminal...

# 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);
```

2.3.2.2 **Exercise 5** Plots for all window sizes of Ecol

```
#%
plotting for all window sizes of Ecol
GC_avg <- 50.79
for (size in c(100, 200, 500, 1000, 2000, 5000, 10000)){
  ZZ <- gzfile(paste0('stats/Ecol_genomegcanalysis_wlen',size ,'.tbl.gz'));
  GC_name<-paste0("GC_w", size)
  GC_name <- read.table(ZZ, header=FALSE);
  colnames(GC_name) <- c("CHRid","NUCpos","GCpct");

  summary(GC_name)

  library(ggplot2);

  G <- ggplot(GC_name, aes(x=NUCpos, y=GCpct)) +
          geom_line(colour = "blue") +
          theme_bw() +
          geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", size=1.5) +
          ggtitle(paste0("E.coli GC content over the genome (window length =", size, "bp )")) +
          labs(x="Genomic Coords (bp)", y="%GC Content");

  ggsave(paste0("images/Ecol_genomegcanalysis_wlen", size, ".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.

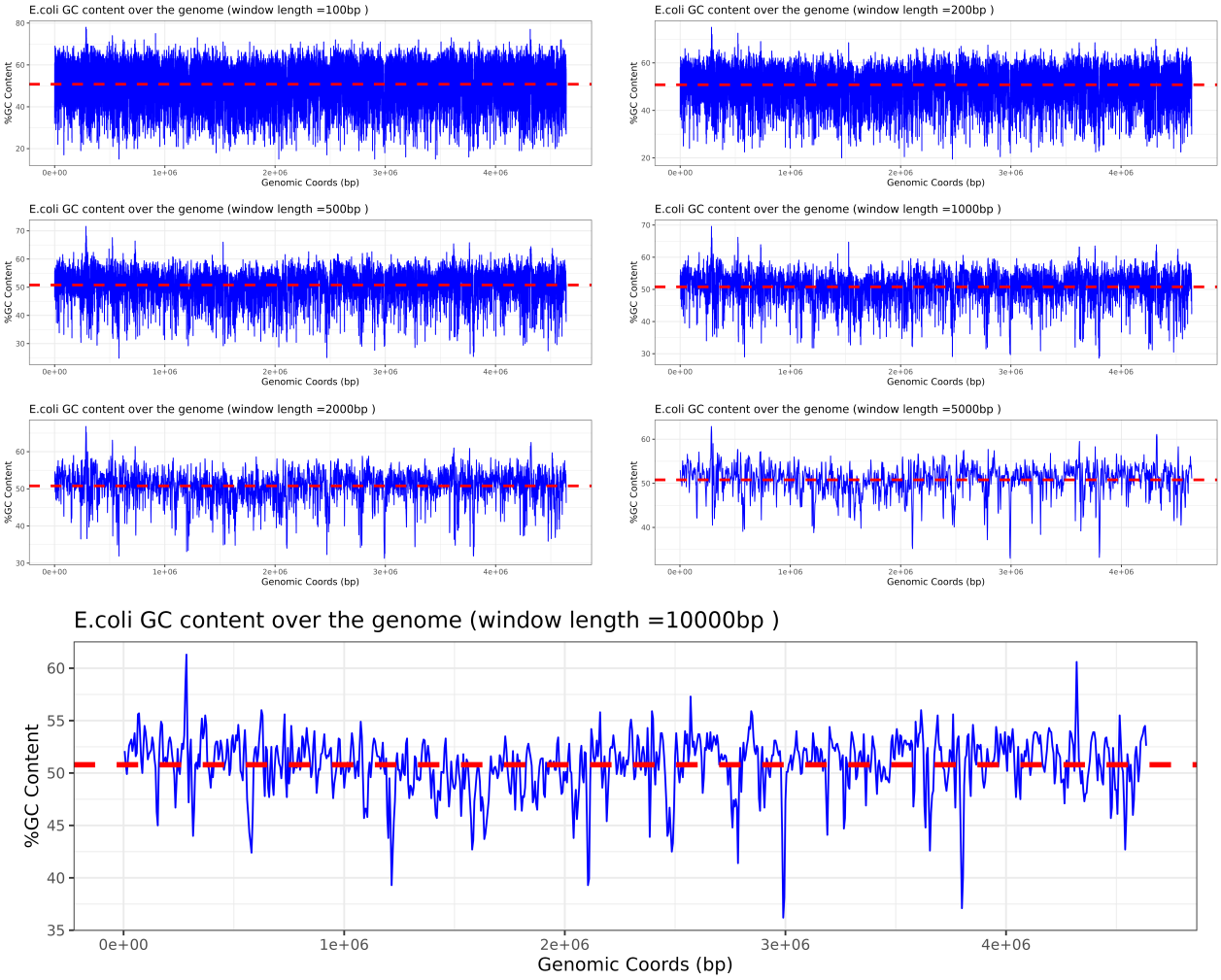

Figure 2: **Window length plots: Plot Ecol for window lengths 100, 200, 500, 1000, 2000, 5000, and 10000.**

2.3.2.3 **Exercise 6** Analysis for 5000 window length

```
#%
#Analysis for 5000 window size
#Its a large enough window size to be able to interpret the graphic while
  #still allowing for detail (unlike 10000)
### repeat the commands for the other three genomes

for (org in c("Cbot", "Ecol", "Mgen", "Mpne")){
  ZZ <- gzfile(paste0("stats/", org, "_genomegcanalysis_wlen5000.tbl.gz"));
  GC_w5000 <- read.table(ZZ, header=FALSE);
  colnames(GC_w5000) <- c("CHRid","NUCpos","GCpct");

  s<-data.frame(summary(GC_w5000))
  GC_meanList<-s$Freq[16]
  GC_avg<-as.numeric(strsplit(GC_meanList, ":")[[1]][2])

  library(ggplot2);

  G <- ggplot(GC_w5000, aes(x=NUCpos, y=GCpct)) +
          geom_line(colour = "blue") +
          theme_bw() +
          geom_hline(yintercept=GC_avg, colour="red", linetype="dashed", size=1.5) +
          ggtitle(paste0(org, " GC content over the genome (window length = 5000bp)")) +
          labs(x="Genomic Coords (bp)", y="%GC Content");
```

```
  ggsave(paste0("images/", org, "_genomegcanalysis_wlen5000.png"),
         plot=G, width=25, height=8, units="cm", dpi=600);
}
#-%#
```

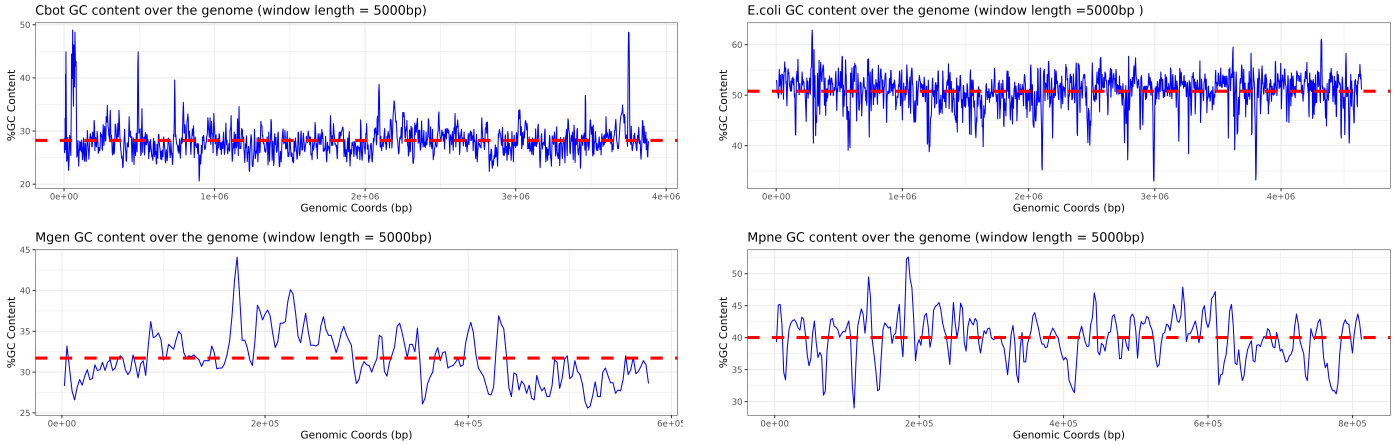

Figure 3: **Multiple species window length: Plot various species for window length 5000.**

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 $WDR/stats/${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.
```

2.4.0.1 Exercise 7 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.

Kmer composition analysis

```
#%
#kmer composition analysis
### repeat the commands for the other three genomes

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

sizeList="10 15 20 25 30 35 40"
specieList="Cbot Ecol Mgen Mpne"

for specie in $specieList; do
    for size in $sizeList; do
      echo "$specie $size"> ./stats/"$specie"_"$size.txt"
        jellyfish_on_kmer $specie $size | cat >> ./stats/"$specie"_"$size.txt"
    done
done

#Get the results and copy them to a table
cat ./stats/Cbot_*.txt
cat ./stats/Ecol_*.txt
cat ./stats/Mgen_*.txt
cat ./stats/Mpne_*.txt
#-%#
```

Copy the results from the cat commands manually to example table (change column names and unrotate them to match) and fix the caption

Species	K-mer Size	Unique K-mers	Distinct K-mers	Total K-mers	Max count
Clostridium botulinum	10	81674	356387	3903242	867
	15	3216404	3489254	3903232	62
	20	3798425	3828717	3903222	30
	25	3815550	3839201	3903212	28
	30	3823200	3844280	3903202	22
	35	3828308	3847684	3903192	20
	40	3832046	3850192	3903182	20
Escherichia coli	10	40625	490389	4641643	284
	15	4357730	4462229	4641638	137
	20	4507760	4542190	4641633	82
	25	4516906	4548910	4641628	75
	30	4522959	4553477	4641623	48
	35	4527791	4557129	4641618	42
	40	4531656	4559993	4641613	12
Mycoplasma genitalium	10	87099	192529	580067	122
	15	550161	562149	580062	39
	20	564076	569990	580057	31
	25	566174	571601	580052	20
	30	567957	572877	580047	11
	35	569477	573915	580042	8
	40	570718	574743	580037	7
Mycoplasma pneumoniae	10	123499	292158	816385	58
	15	740058	766612	816380	36
	20	757793	778224	816375	15
	25	764947	783662	816370	15
	30	770702	787967	816365	14
	35	775615	791523	816360	14
	40	779921	794536	816355	14

Table 2: **K-mer table for multiple species.** Kmer count at multiple sizes for each species, displaying their unique, distinct, total and max count of K-mers.

3 Discussion

The results that can be interpreted from the the chaos plots are that the GC content is highest in E. coli and lowest in C. botilium, with M. genitalium and M. pneumoniae have an intermediate density, we can observe that in the C and G percentages.

In the window size plots in different species we can corroborate this observation, as GC content in Cbot is indeed the lowest (<30%) and in Ecol its the highest (>50%), we can also see that the graphs of Mpne and Mgen have much less noise and better defined peaks.

The k-mer table shows us that as size increases unique and distinct k-mers increase, longer k-mers repeat less often and max count also experience a reduction. Unique and distinct k-mers show a clear increase when k-mer sizes are small, and plateau when k-mer sizes are big. Ecol presents the most total k-mers and Mgen the least, meaning the greatest and simplest genetic structure variation respectively.

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: CG_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="Izquierdo_Jan";                  #-> 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`).