BScBI-CG

Practicals Report

Name SURNAME

Exercise 04

— November 7, 2024—

Computational Genomics 2024–25

BSc in Bioinformatics

UPF/ESCI – UPC – UB – UAB

Contents

List of Tables

List of Figures

1 Introduction

From the previous exercise, we are going to check completeness of the *de novo* assemblies we have generated for *Saccharomyces cerevisiae* and to compare them against the available reference genome. To speed up those comparisons we can focus on specific chromosomes, so that we will need to filter out sequences or alignments to continue with the analyses. After that we can take a look to the repetitive elements that may be present in contigs from our best assembly set.

1.1 Objectives

- To compare our assembly against a reference genome, so we can assess the performance of the assembler and the protocol.
- To check completeness of our assemblies, so that we can choose the best assembly for the downstream analyses, such as masking, gene-prediction, and so on.
- To complete the assembly protocol by masking the repeats detected on the chosen contigs/scaffolds.
- We introduce some LaTeX examples for citing paper references as footnotes.

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:

```
#################################
# We are assuming that you have already installed from previous exercises
# the following packages:
#   samtools bamtools picard-tools bwa bowtie2 igv

# ncbi-blast+ - next generation suite of BLAST sequence search tools
sudo apt-get install ncbi-blast+

# gnuplot-qt - a portable command-line driven interactive data and function plotting utility
#              It is required for mummerplot later on...
sudo apt-get install gnuplot-qt
sudo ln -vfs /usr/bin/gnuplot-qt /usr/bin/gnuplot
# just in case mummerplot returns error: Inappropriate ioctl for device
sudo ln -vfs /usr/bin/gnuplot-qt /etc/alternatives/gnuplot
##notDone ^^

# graphicsmagick - collection of image processing tools (replacement for imagemagick)
sudo apt-get install graphicsmagick
##notDone ^^

# mummer - Efficient sequence alignment of full genomes
sudo apt-get install mummer --reinstall
##notDone VV
#
# IMPORTANT: Some fixes are needed prior to run gnuplot within mummerplot
#            (install mummer package first)
#
#   + just in case mummerplot returns error: Can't use 'defined(%hash)'
sudo sed -i 's/defined (%/(%/
            ' /usr/bin/mummerplot
#
#   + gnuplot fails because some instruction was implemented in later versions
sudo sed -i 's/^\(.*set mouse.*\)$/#\1/
```

```
            ' /usr/bin/mummerplot
#
#   + mummerplot cannot find gnuplot:
sudo sed -i 's/system (\"gnuplot --version\")/system (\"\/usr\/bin\/gnuplot --version\")/
            ' /usr/bin/mummerplot
sudo sed -i 's/my \$cmd = \"gnuplot\";/my \$cmd = \"\/usr\/bin\/gnuplot\";/
            ' /usr/bin/mummerplot
#
#   + just in case mummerplot returns error: Inappropriate ioctl for device
sudo ln -vfs /usr/bin/gnuplot-qt /etc/alternatives/gnuplot;
#
# Further NOTES:
#   You probably need to run all those fixes if your system has already gnuplot version > 4.
#   MacOS users lacking sed can try with "perl -i -pe" instead of "sed -i".

# hmmer2 - profile hidden Markov models for protein sequence analysis
sudo apt-get install hmmer

#
# Note: Before installing BUSCO from source it is advisable to install bbmap and metaeuk
#
# bbmap - BBTools genomic aligner and other tools for short sequences
cd $BIN
wget https://sourceforge.net/projects/bbmap/files/BBMap_39.01.tar.gz/download \
     -O BBMap_39.01.tar.gz
tar -zxvf BBMap_39.01.tar.gz bbmap/
#
# metaeuk - toolkit for large-scale gene discovery
#           and annotation in eukaryotic metagenomic contigs.
cd $BIN
git clone https://github.com/soedinglab/metaeuk.git
mv metaeuk MetaEuk
cd MetaEuk/
mkdir build
cd build/
cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=. ..
make -j
##ERROR: Does not work, needs too much RAM, use $make
make
make install
cd $BIN
ln -vs ./MetaEuk/build/bin/metaeuk ./metaeuk
#
# BUSCO - estimating the completeness and redundancy of processed genomic data
#         based on universal single-copy orthologs.
#
# Note: you can also download and install BUSCO using conda,
#       see further instructions at:  https://anaconda.org/bioconda/busco
#
#       If using conda, you can try:
#             conda create -n busco5 -c conda-forge -c bioconda busco=5.4.3
#       Then, run it after activating the environment:
#             conda activate busco5
#       And remember to set up all project vars again as you enter into a new shell.
#
## cd $BIN/
## git clone https://gitlab.com/ezlab/busco.git
## cd busco/
## sudo python3 setup.py install
## cd $WDR

#%
wget https://gitlab.com/ezlab/busco/-/archive/5.4.3/busco-5.4.3.tar.gz \-O busco-5.4.3.tar.gz
```

```
tar -zxvf busco-5.4.3.tar.gz
mv busco-5.4.3 busco
cd busco
python3 setup.py install --user
cd $BIN
ln -vs ../busco/ .
#-%#

#
# repeatmasker - finds repeat families from biological sequences
#
### IMPORTANT NOTE ### -----------------------------------------------------------------------
#
# As RepeatMasker may have some installation issues due to system and/or conda libraries,
# it is recommended to use the docker containers as explained on the short tutorials
# from the virtual campus. Anyway, it should be easy to install with the following instructions:
#
# Repeatmasker may require some extra dependencies
# and you only need those provided by the commands below.
# For further info you can visit: http://www.repeatmasker.org/RepeatMasker/
#
sudo perl -MCPAN -e'install Text::Soundex'
sudo apt install python3-h5py
#
#   we need to download the binaries for tandem repeats finder from:
#              http://tandem.bu.edu/trf/trf409.linux64.download.html
cd $BIN
git clone https://github.com/Benson-Genomics-Lab/TRF.git
cd TRF
mkdir build
cd build
../configure
make
cd $BIN
ln -vs ./TRF/build/src/trf ./
#
# Now it's time to install RepeatMasker (take care, it's a large file, >800Mbp).
wget http://www.repeatmasker.org/RepeatMasker/RepeatMasker-4.1.3-p1.tar.gz \
     -O $BIN/RepeatMasker-4.1.3-p1.tar.gz
cd $BIN
tar -zxvf RepeatMasker-4.1.3-p1.tar.gz
mv RepeatMasker RepeatMasker-4.1.3
cd RepeatMasker-4.1.3
sudo perl ./configure

#/home/jj/Desktop/Bioinformatics/3rd_year/1term/Computational_genomics/Seminars/exercise_04/bin/trf
#/usr/bin/ is the correct folder > Enter

# Important: provide here paths only to trf and to hmmer,
#            and set hmmer as default search engine.
cd $BIN
ln -vs ./RepeatMasker-4.1.3/RepeatMasker .
cd $WDR

### NOTE ###
## You can install docker server and download the repeatmasker container
## following the instructions from the Virtual Campus if you experience problems
## when you try to compile or install this tool.
```

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 5.2.1 on page 19). Finally, you may need to perform some fixes on the mummerplot script that was installed in your conda environment:

```
#
# IMPORTANT: Some fixes are needed prior to run gnuplot within mummerplot
#            (install mummer package first)
#
# Remind that this folder can change form one system to another:
MUMMER=/home/jj/anaconda3/envs/CG_exercises/bin/mummerplot;
#MUMMER=/usr/bin/mummerplot
#
#   + just in case mummerplot returns error: Can't use 'defined(%hash)'
sed -i 's/defined (%/(%/' $MUMMER;
#
#   + gnuplot fails because some instruction was implemented in later versions
sed -i 's/^\(.*set mouse.*\)$/#\1/' $MUMMER;
#
#   + mummerplot cannot find gnuplot:
sed -i 's/system (\"gnuplot --version\")/system (\"\/usr\/bin\/gnuplot --version\")/
       ' $MUMMER;
sed -i 's/my \$cmd = \"gnuplot\";/my \$cmd = \"\/usr\/bin\/gnuplot\";/
       ' $MUMMER;
```

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_04.tgz
cd exercise_04

# Rename report file including your "NAME" and "SURNAME"
mv -v README_BScBICG2425_exercise04_SURNAME_NAME.md \
      README_BScBICG2425_exercise04_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_exercise04_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...

1.3 Datasets

On the previous exercise we have started to work on genomic datasets for the budding yeast *Saccharomyces cerevisiae*, one of the major model organisms for understanding cellular and molecular processes in eukaryotes. We also told on that exercise to keep the files for re-using them on the current practical, so that we assume you already have some of the initial files. The *S. cerevisiae* (strain S288C)
assembly R64 reference genome version[1] has 16 chromosome sequences, plus the mitochondrion genome, totaling 12,157,105bp (see Table 1).

Instead of copying all the folders and duplicate all of its files, we will re-use from previous exercise by creating symbolic links (something similar to a MS-Windows direct access) with the commands below:

```
#
# We are about to link previous exercise folders,
# assuming you have already set your current working directory
# on the exercise_04 folder...
#
for FD in seqs bowtie soapdenovo;
```

[1] Engel et al. "The Reference Genome Sequence of *Saccharomyces cerevisiae*: Then and Now". *G3 (Bethesda)*, g3.113.008995v1, 2013 (PMID:24374639)

```
  do {
   ln -vs ../exercise_03/$FD ./;
  }; done

tail -6 $WDR/soapdenovo/SRR6130428/SRR6130428_k63_contig.log
# There are 3156 contig(s) longer than 100, sum up 11779878 bp, with average length 3732.
# The longest length is 69946 bp, contig N50 is 14294 bp,contig N90 is 3279 bp.
# 4867 contig(s) longer than 64 output.
# Time spent on constructing contig: 0m.

# IMPORTANT: We have this data already from previous exercise,
#            we include here for completeness sake.
URL="https://downloads.yeastgenome.org/sequence/S288C_reference/genome_releases"
wget ${URL}/S288C_reference_genome_Current_Release.tgz \
     -O $WDR/seqs/S288C_reference_genome_Current_Release.tgz

pushd $WDR/seqs/
tar -zxvf S288C_reference_genome_Current_Release.tgz
popd

export GENOMEDIR=$WDR/seqs/S288C_reference_genome_R64-5-1_20240529;
export GENOMEFSA=S288C_reference_sequence_R64-5-1_20240529;
# you may consider copying those vars to your projectvars.sh file

zcat $GENOMEDIR/$GENOMEFSA.fsa.gz | \
  infoseq -only -length -noheading -sequence fasta::stdin  2> /dev/null | \
  gawk '{ s+=$1 } END{ print "# Yeast genome size (v.R64): "s }'
# Yeast genome size (v.R64): 12157105


#
######-------------------------------------------------------
#
# You can include a LaTeX table with the chromosome sizes and GC content,
# which can be produced with a command like this:
#
zcat $GENOMEDIR/$GENOMEFSA.fsa.gz | \
  infoseq -noheading -sequence fasta::stdin  2> /dev/null | \
    gawk 'BEGIN {
            printf "%15s & %10s & %12s & %10s \\\\n",
                   "Chromosome", "GenBank ID", "Length (bp)", "GC content";
          }
          $0 != /^[ \t]*$/ {
            L=$0;
            sub(/^.*\[(chromosome|location)=/,"",L);
            sub(/\].*$/,"",L);
            sub(/_/,"\\_",$3);
            printf "%15s & %10s & %12d & %8.2f\\% \\\\n",
                   L, $3, $6, $7;
          }' > $WDR/docs/chromosomes_info.tex;
#
######-------------------------------------------------------
```

1.3.1 Table of chromosome sizes and GC content

Chromosome	GenBank ID	Length (bp)	GC content
I	NC_001133	230218	39.27%
II	NC_001134	813184	38.34%
III	NC_001135	316620	38.53%
IV	NC_001136	1531933	37.91%
V	NC_001137	576874	38.51%
VI	NC_001138	270161	38.73%
VII	NC_001139	1090940	38.06%
VIII	NC_001140	562643	38.50%
IX	NC_001141	439888	38.90%
X	NC_001142	745751	38.37%
XI	NC_001143	666816	38.07%
XII	NC_001144	1078177	38.48%
XIII	NC_001145	924431	38.20%
XIV	NC_001146	784333	38.64%
XV	NC_001147	1091291	38.16%
XVI	NC_001148	948066	38.06%
mitochondrion	NC_001224	85779	17.11%

Table 1: **Reference *Saccharomyces cerevisiae* chromosome summary.** This table will show information about length and GC content for the chromosomes of the *S. cerevisiae* reference genome.

```
# We need to compute some extra stats from the assemblies,
# we can download from github one of the assemblathon scripts for this purpose
# (to facilitate the task, it is already available on the bin folder)
#
## GITASM=https://github.com/KorfLab/Assemblathon/raw/master
## wget $GITASM/assemblathon_stats.pl  -O bin/assemblathon_stats_unfixed.pl
## wget http://korflab.ucdavis.edu/Unix_and_Perl/FAlite.pm -O bin/FAlite.pm
## chmod a+x bin/assemblathon_stats.pl
#
# however the main script fails due to a syntax error due to updates on Perl interpreter,
# here you can see the fixes introduced to the original script:
#
# 301c283
# <     foreach my $size qw(1000 10000 100000 1000000 10000000){
# ---
# >     foreach my $size (qw(1000 10000 100000 1000000 10000000)) {
# 428c409
# <     foreach my $base qw (A C G T N){
# ---
# >     foreach my $base (qw(A C G T N)) {

#
# IMPORTANT: run this first, otherwise you will get an error
#            about perl not finding FAlite.pm module
#
export PERL5LIB=$BIN;

#
zcat $GENOMEDIR/$GENOMEFSA.fsa.gz | \
  $BIN/assemblathon_stats.pl \
            -csv -genome_size 12160000 - \
          > $WDR/stats/assembly_stats_$GENOMEFSA.txt

$BIN/assemblathon_stats.pl \
            -csv -genome_size 12160000 \
            $WDR/soapdenovo/SRR6130428/SRR6130428_k63_graph.contig.fa \
          > $WDR/stats/assembly_stats_SRR6130428_soapdenovo_k63_graph.contig.txt
```

We will continue using the smaller dataset from the previous exercise, SRR6130428, for most of the examples on this exercise. However, if you had already chosen another SRA reads set to generate your assembly; you will need it to complete the table below with the outcomes from the initial assembly steps (see the example from Table 2). If you ran assembly over a second set or on a different reads sampling, you are also welcome to include such information as another summary row on this table.

Check out the analysis done above to fill the table:

```
#%
cat $WDR/stats/assembly_stats_SRR6130428_soapdenovo_k63_graph.contig.txt
#-%#
```

For my sequence (SRR7548448)

```
#%
$BIN/assemblathon_stats.pl \
            -csv -genome_size 12160000 \
            $WDR/soapdenovo/SRR7548448/SRR7548448_k63_graph.contig.fa\
          > $WDR/stats/assembly_stats_SRR7548448_soapdenovo_k63_graph.contig.txt

cat $WDR/stats/assembly_stats_SRR7548448_soapdenovo_k63_graph.contig.txt
#-%#
```

1.3.2 Assembly results

Assembly	Status	#Seqs	Total Length	Longest Seq	Shortest Seq	Avg Length	N_{50}	N_{50}
			bp	*bp*	*bp*	*bp*	#Seqs.	*bp*
Scer__ref__vR64	Chromosomes	17	12157105	1531933	85779	715124	6	924431
SRR6130428	Contigs	4867	11905871	69946	64	2446	256	14236
SRR7548448	Contigs	72962	11531630	3721	64	158	212	14167

Table 2: **Summary of assembly results on each set.** We have computed same parameters over *S. cerevisiae* reference genome as for the assembled versions.

For some of the following analyses, we will focus on a couple of reference genome chromosomes, chrI and chrM (mitochondrion genome). Here we are going to filter them out:

```
mkdir $WDR/seqs/chrs;

for SQ in chrI:NC_001133 chrM:NC_001224;
  do {
    SQN=${SQ%%:*}; # get the chr from SQ string
    SQI=${SQ##*:}; # get the refseq id from SQ string
    echo "# Filtering $SQN [$SQI] from whole genome fasta file..." 1>&2;
    echo 'ref|'$SQI'|' | \
      seqtk subseq $GENOMEDIR/$GENOMEFSA.fsa.gz - | \
        sed 's/^>.*$/>Scer_'$SQN'/;' \
                 > $WDR/seqs/chrs/$SQN.fa;
  }; done
```

2 Exploring the assemblies

2.1 Filter out contigs mapping to reference chromosomes

Later on we are going to run dnadiff from the MUMmer package[2] to compare assembled contigs against the reference or between them. In order to speed up the example, we will first use NCBI-BLAST[3] to project all the assembled contigs into the chosen reference chromosomes, so we will reduce all the downstream calculations. We need to create a database for each assembly sequence sets and we will query by the reference selected chromosomes.

[2] S. Kurtz, A. Phillippy, A.L. Delcher, M. Smoot, M. Shumway, C. Antonescu, and S.L. Salzberg. "Versatile and open software for comparing large genomes." *Genome Biology*, 5:R12, 2004.

[3] C. Camacho, G. Coulouris, V. Avagyan, N. Ma, J. Papadopoulos, K. Bealer, and T.L. Madden. "BLAST+: architecture and applications." *BMC Bioinformatics*, 10:421, 2008.

```
export SQSET=SRR6130428

mkdir -vp $WDR/blast/dbs

makeblastdb -in $WDR/soapdenovo/$SQSET/${SQSET}_k63_graph.contig.fa \
            -dbtype nucl \
        -title "${SQSET}_SOAPdenovo_k63_contigs" \
        -out $WDR/blast/dbs/${SQSET}_SOAPdenovo_k63_contigs \
          2> $WDR/blast/dbs/${SQSET}_SOAPdenovo_k63_contigs.log 1>&2;
```

Now we can BLAST reference sequences against the newly created database; we will use megablast option as we are comparing sequences for the same species, and that BLAST program has the parameters optimized for this kind of genomic searches.

```
# we define here a custom BLAST tabular output format
BLASTOUTFORMAT='6 qseqid qlen sseqid slen qstart qend sstart send length';
BLASTOUTFORMAT=$BLASTOUTFORMAT' score evalue bitscore pident nident ppos positive';
BLASTOUTFORMAT=$BLASTOUTFORMAT' mismatch gapopen gaps qframe sframe';
export BLASTOUTFORMAT;

for SQ in chrI chrM;
  do {
    echo "# Running MEGABLAST: $SQSET x $SQ ..." 1>&2;
    mkdir -vp $WDR/blast/${SQ}-x-${SQSET};
    blastn -task megablast -num_threads 8                       \
           -db    $WDR/blast/dbs/${SQSET}_SOAPdenovo_k63_contigs \
           -outfmt "$BLASTOUTFORMAT"                            \
           -query $WDR/seqs/chrs/$SQ.fa                         \
           -out   $WDR/blast/${SQ}-x-${SQSET}/${SQ}-x-${SQSET}_megablast.out \
             2>   $WDR/blast/${SQ}-x-${SQSET}/${SQ}-x-${SQSET}_megablast.log;
  }; done
```

Once we have found the matching contigs, let's filter them out from the whole assembly fasta file.

```
for SQ in chrI chrM;
  do {
    echo "# Getting $SQSET contigs matching $SQ ..." 1>&2;
    OFBN="$WDR/blast/${SQ}-x-${SQSET}/${SQ}-x-${SQSET}_megablast";
    # this is a check to ensure we start with an empty contigs fasta file
    if [ -e "$OFBN.fa" ];
      then
        printf '' > "$OFBN.fa"; # rm can be also used here, but dangerous for novice
      fi;
    # get the contig IDs from third column and filter out sequences
    gawk '{ print $3 }' "$OFBN.out" | sed 's/^>//' | sort | uniq | \
      while read SQID;
        do {
           samtools faidx \
                    $WDR/soapdenovo/$SQSET/${SQSET}_k63_graph.contig.fa \
                    "$SQID";
        }; done >> "$OFBN.fa" \
                2> "$OFBN.fa.log";
  }; done
```

Now that we have the filtered out the required contigs, we can start the following sequence comparison procedure based on dnadiff.

```
for SQ in chrI chrM;
  do {
    #
    printf "# Running DNADIFF protocol: $SQSET x $SQ ..." 1>&2;
    IFBN="$WDR/blast/${SQ}-x-${SQSET}/${SQ}-x-${SQSET}_megablast";
    OFBD="$WDR/blast/${SQ}-x-${SQSET}/${SQ}-x-${SQSET}_megablast.dnadiff";
    #
    dnadiff -p $OFBD              \
            $WDR/seqs/chrs/$SQ.fa \
            $IFBN.fa              \
         2> $OFBD.log;
    printf " DNAdiff..." 1>&2;
    #
    mummerplot --large --layout --fat --postscript \
               -t "Alignment Plot: ${SQ}-x-${SQSET}" \
               -p $OFBD           \
                  $OFBD.1delta    \
               2> $OFBD.alnplot.log;
    # add this to convert PostScript image to PNG
    convert-im6 -verbose $OFBD.ps $OFBD.png;
    printf " ALNplot..." 1>&2;
    mummerplot --large --layout --fat --coverage --postscript \
               -t "Coverage Plot: ${SQ}-x-${SQSET}" \
               -p $OFBD.covg      \
                  $OFBD.1delta    \
               2> $OFBD.cvgplot.log;
    # add this to convert PostScript image to PNG
    convert-im6 -verbose $OFBD.covg.ps $OFBD.covg.png;
    printf " CVGplot..." 1>&2;
    #
    ( grep "TotalBases"   $OFBD.report;
      grep "AlignedBases" $OFBD.report;
      grep "AvgIdentity"  $OFBD.report
      ) > $OFBD.shortsummary;
    printf " DONE\n" 1>&2;
    #
  }; done
```

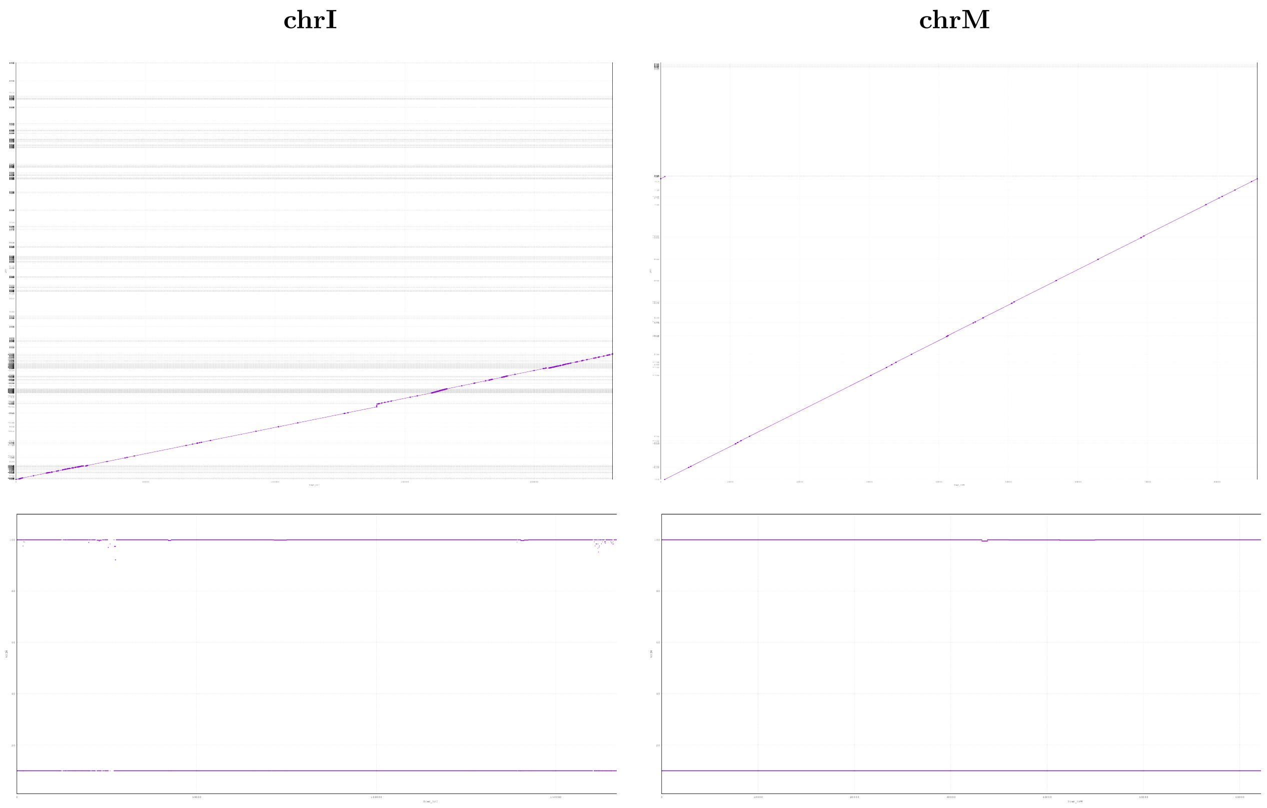

Figure 1: **Comparison between two reference chromosomes and SRR6130428 contigs.** Top panels show the alignment plots of contigs from SRR6130428 assembly mapped over two reference *Saccharomyces cerevisiae* chromosomes, chrI and chrM on left and right panels respectively. Bottom panels show the alignment coverage for the same sequence relations. It is evident from the comparison that contigs aligning to chrM have better contiguity despite overall coverages are quite similar on both reference chromosomes.

Repeat the process for my sequence (SRR7548448):

```
#%

export SQSET=SRR7548448

#Create the database folders

mkdir -vp $WDR/blast/dbs

makeblastdb -in $WDR/soapdenovo/$SQSET/${SQSET}_k63_graph.contig.fa \
            -dbtype nucl \
        -title "${SQSET}_SOAPdenovo_k63_contigs" \
        -out $WDR/blast/dbs/${SQSET}_SOAPdenovo_k63_contigs \
          2> $WDR/blast/dbs/${SQSET}_SOAPdenovo_k63_contigs.log 1>&2;

##################################################
#Blast the reference sequences against my sequence

BLASTOUTFORMAT='6 qseqid qlen sseqid slen qstart qend sstart send length';
BLASTOUTFORMAT=$BLASTOUTFORMAT' score evalue bitscore pident nident ppos positive';
BLASTOUTFORMAT=$BLASTOUTFORMAT' mismatch gapopen gaps qframe sframe';
export BLASTOUTFORMAT;

for SQ in chrI chrM;
  do {
    echo "# Running MEGABLAST: $SQSET x $SQ ..." 1>&2;
    mkdir -vp $WDR/blast/${SQ}-x-${SQSET};
    blastn -task megablast -num_threads 8                          \
           -db    $WDR/blast/dbs/${SQSET}_SOAPdenovo_k63_contigs \
           -outfmt "$BLASTOUTFORMAT"                             \
           -query $WDR/seqs/chrs/$SQ.fa                          \
           -out   $WDR/blast/${SQ}-x-${SQSET}/${SQ}-x-${SQSET}_megablast.out \
             2>   $WDR/blast/${SQ}-x-${SQSET}/${SQ}-x-${SQSET}_megablast.log;
  }; done

##################################################
#Filter contigs for the whole assembly fasta file

for SQ in chrI chrM;
  do {
    echo "# Getting $SQSET contigs matching $SQ ..." 1>&2;
    OFBN="$WDR/blast/${SQ}-x-${SQSET}/${SQ}-x-${SQSET}_megablast";
    # this is a check to ensure we start with an empty contigs fasta file
    if [ -e "$OFBN.fa" ];
      then
        printf '' > "$OFBN.fa"; # rm can be also used here, but dangerous for novice
      fi;
    # get the contig IDs from third column and filter out sequences
    gawk '{ print $3 }' "$OFBN.out" | sed 's/^>//' | sort | uniq | \
      while read SQID;
        do {
           samtools faidx \
                    $WDR/soapdenovo/$SQSET/${SQSET}_k63_graph.contig.fa \
                    "$SQID";
        }; done >> "$OFBN.fa" \
                2> "$OFBN.fa.log";
  }; done

##################################################
#Follow the sequence comparison procedure based on dnadiff

for SQ in chrI chrM;
  do {
    #
    printf "# Running DNADIFF protocol: $SQSET x $SQ ..." 1>&2;
    IFBN="$WDR/blast/${SQ}-x-${SQSET}/${SQ}-x-${SQSET}_megablast";
    OFBD="$WDR/blast/${SQ}-x-${SQSET}/${SQ}-x-${SQSET}_megablast.dnadiff";
    #
    dnadiff -p $OFBD              \
            $WDR/seqs/chrs/$SQ.fa \
            $IFBN.fa              \
         2> $OFBD.log;
    printf " DNAdiff..." 1>&2;
    #
    mummerplot --large --layout --fat --postscript \
               -t "Alignment Plot: ${SQ}-x-${SQSET}" \
               -p $OFBD           \
                  $OFBD.1delta    \
               2> $OFBD.alnplot.log;
    # add this to convert PostScript image to PNG
    convert-im6 -verbose $OFBD.ps $OFBD.png;
    printf " ALNplot..." 1>&2;
    mummerplot --large --layout --fat --coverage --postscript \
               -t "Coverage Plot: ${SQ}-x-${SQSET}" \
               -p $OFBD.covg      \
                  $OFBD.1delta    \
               2> $OFBD.cvgplot.log;
    # add this to convert PostScript image to PNG
    convert-im6 -verbose $OFBD.covg.ps $OFBD.covg.png;
    printf " CVGplot..." 1>&2;
    #
    ( grep "TotalBases"   $OFBD.report;
      grep "AlignedBases" $OFBD.report;
      grep "AvgIdentity"  $OFBD.report
      ) > $OFBD.shortsummary;
    printf " DONE\n" 1>&2;
    #
  }; done

#-%#
```

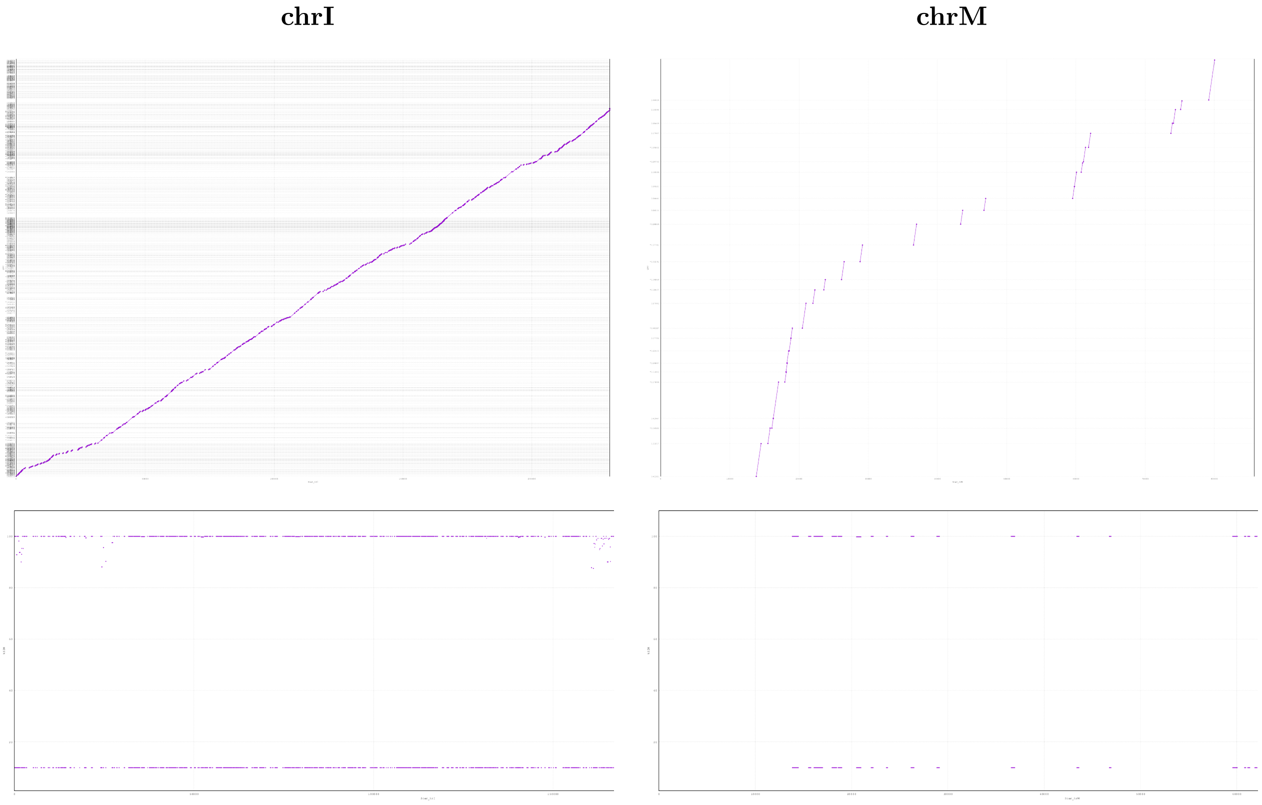

Figure 2: **dnadiff comparison between two reference chromosomes and SRR7548448 contigs.** Top panels show the alignment plots of contigs from SRR7548448 assembly mapped over two reference *Saccharomyces cerevisiae* chromosomes, chrI and chrM on left and right panels respectively. Bottom panels show the alignment coverage for the same sequence relations. It is evident from the comparison that contigs aligning to chrM have better contiguity despite overall coverages are quite similar on both reference chromosomes.

##TODISCUSS Discuss later on which of the two chromosome assemblies do you think had a better outcome and try to guess why. We may need to analyze whether the contigs fully align to the reference chromosomes, **can you calculate the average coverage of the aligned reads?**

2.2 Assessment of genome completeness with BUSCO

BUSCO[4] estimates the completeness and redundancy of processed genomic data based on universal single-copy orthologs. First of all, we need to check if there is a clade-specific parameters set that fits with our organism, *Saccharomyces cerevisiae* belong to the *Saccharomycetes* class, within the *Ascomycota* phylum in the Fungi kingdom (see NCBI Taxonomy browser species card).

```
busco --list-datasets
# 2022-10-24 19:06:49 INFO: Downloading information on latest versions of BUSCO data...
```

[4] M. Manni, M.R. Berkeley, M. Seppey, F.A. Simão, and E.M. Zdobnov. "BUSCO Update: Novel and Streamlined Workflows along with Broader and Deeper Phylogenetic Coverage for Scoring of Eukaryotic, Prokaryotic, and Viral Genomes." *Molecular Biology and Evolution*, 38(10)4647-–4654, 2021.

```
# 2022-10-24 19:06:52 INFO: Downloading file 'https://busco-data.ezlab.org/v5/data/information/lineages_li
# 2022-10-24 19:06:53 INFO: Decompressing file '/home/lopep/SANDBOX/BScCG2425/exercise_04/busco_downloads/
#
# ################################################
#
# Datasets available to be used with BUSCO v4 and v5:
#
#  bacteria_odb10
#      - ...
#  archaea_odb10
#      - ...
#  eukaryota_odb10
#      - ...
#      - fungi_odb10
#          - ascomycota_odb10
#              - ...
#              - saccharomycetes_odb10
#              - ...
#          - ...
#      - ...
#  viruses (no root dataset)
#      - ...
#
```

Luckily for us, there is one class specific set to evaluate the completeness of our genome assembly `saccharomycetes_odb10`. However, it is worth that you consider also to run the `BUSCO` commands below using a more general parameters set, such as `fungi_odb10` or even `eukaryota_odb10` (or both), and compare the corresponding results. This can be useful to extrapolate the assessment to other species genomes for which we will not have as much information as for this yeast.

```
export SQSET=SRR6130428

mkdir -vp $WDR/busco/$SQSET

# fix PATH to point bin folders where you installed bbmap and busco programs
export PATH=$BIN:$BIN/bbmap:$BIN/busco/bin:$PATH

# cd $WDR
busco -m genome \
      -i $WDR/soapdenovo/$SQSET/${SQSET}_k63_graph.contig.fa \
      -o ./busco/$SQSET/${SQSET}_k63_contigs \
      -l saccharomycetes_odb10 -f

## if you get an error "A run with the name xxx already exists"
## you should add command-line option "-f" to force overwriting those files
#    ---------------------------------------------------
#    |Results from dataset saccharomycetes_odb10       |
#    ---------------------------------------------------
#    |  C:98.8%[S:96.7%,D:2.1%],F:0.5%,M:0.7%,n:2137   |
#    |2111  Complete BUSCOs (C)                        |
#    |2067  Complete and single-copy BUSCOs (S)        |
#    |44    Complete and duplicated BUSCOs (D)         |
#    |11    Fragmented BUSCOs (F)                      |
#    |15    Missing BUSCOs (M)                         |
#    |2137  Total BUSCO groups searched                |
#    ---------------------------------------------------
# 2024-11-05 18:44:06 INFO: BUSCO analysis done. Total running time: 577 seconds
```

You can take a step further to plot the resulting completeness values, using the `generate_plot.py` script that is provided under the scripts folder of your `busco` installation:

```
# If you installed BUSCO from sources, this script should be at:
#    $BIN/busco/scripts/generate_plot.py
# however if you got from the conda/mamba environment you must
# search on the conda env bin folder; check your installation
# for the equivalent file, i.e.:
#    $HOME/.conda/envs/BScBI-CG2425_exercises/bin/generate_plot.py
python3 $BIN/busco/scripts/generate_plot.py \
      -wd ./busco/$SQSET/${SQSET}_k63_contigs;
```

For the second sequence(SRR7548448):

```
#%
export SQSET=SRR7548448

mkdir -vp $WDR/busco/$SQSET

export PATH=$BIN:$BIN/bbmap:$BIN/busco/bin:$PATH

busco -m genome \
      -i $WDR/soapdenovo/$SQSET/${SQSET}_k63_graph.contig.fa \
      -o ./busco/$SQSET/${SQSET}_k63_contigs \
      -l saccharomycetes_odb10 -f

#    --------------------------------------------------
#    |Results from dataset saccharomycetes_odb10      |
#    --------------------------------------------------
#    |C:12.2%[S:12.1%,D:0.1%],F:13.2%,M:74.6%,n:2137  |
#    |260   Complete BUSCOs (C)                         |
#    |258   Complete and single-copy BUSCOs (S)         |
#    |2 Complete and duplicated BUSCOs (D)            |
#    |282   Fragmented BUSCOs (F)                       |
#    |1595  Missing BUSCOs (M)                        |
#    |2137  Total BUSCO groups searched               |
#    --------------------------------------------------
# 2024-11-05 18:54:54 INFO: BUSCO analysis done. Total running time: 542 seconds


#Plot the resulting completeness values

python3 $BIN/busco/scripts/generate_plot.py \
      -wd ./busco/$SQSET/${SQSET}_k63_contigs;

#-%#
```

Embed here the corresponding figure; if you have ran more than one assembly, you can combine the busco results to compare completeness among them.

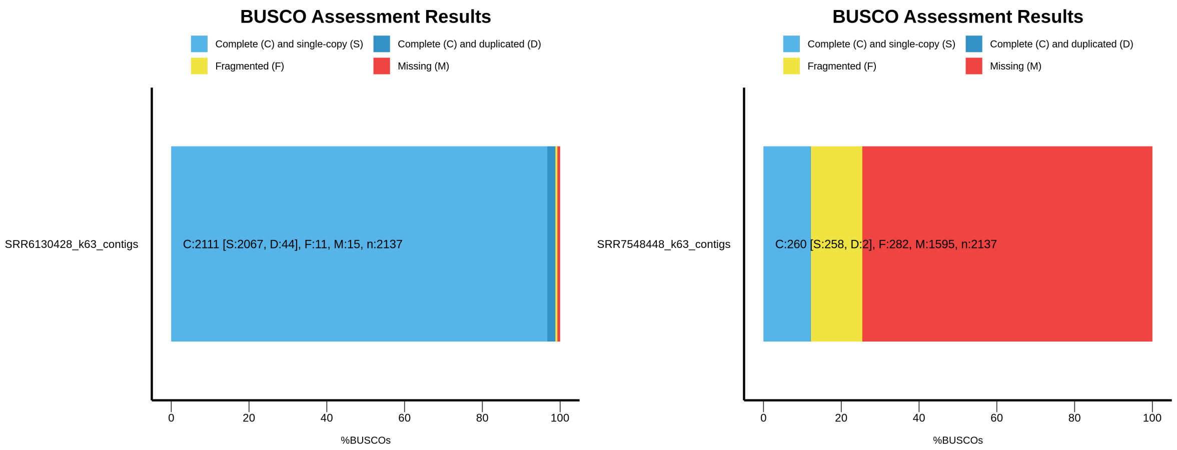

Figure 3: **Completeness `busco` analysis for `SRR6130428` and `SRR7548448`.** Analysis for the completeness of our genome assembly, comparing the completeness values of `SRR6130428` and `SRR7548448` that were generated by `busco` and plotted using `generate_plot.py`.

3 Masking assembly repetitive sequences

Although installing `RepeatMasker`[5] could be a complex task, specially if we want to use the `RepBase` database of curated repeats, we can just install one of the search engines (`hmmer` in our case for this exercise). We are going to run `RepeatMasker` on the set of contigs that map to the reference chromosome chrM for testing purposes (looking that everything is working); we are going to use some extra parameters to speed up the searches (`-qq`, much faster but 10% less sensitive), to use lowercase nucleotides to mask repeats on the output masked sequences (`-small`), and to get extra output files (`-gff`, for the repeats location coords in GFF).

```
SQ=chrM;
SQSET=SRR6130428;
##CONDABIN=$HOME/.conda/envs/CG_exercises/bin; #Doesen't exist
CONDABIN=$BIN;  #due to symbolic links, its the same as CONDABIN=$BIN/RepeatMasker-4.1.3

conda activate CG_exercises; #CG_exercises is the same as BScBI-CG2425_exercises
mkdir -vp $WDR/repeatmasker;


# Run first on the chrM contigs set for a quick test
#
mkdir -vp $WDR/repeatmasker/${SQ}-x-${SQSET}\_megablast
$CONDABIN/RepeatMasker \
           $WDR/blast/${SQ}-x-${SQSET}/${SQ}-x-${SQSET}\_megablast.fa \
           -qq -small -gff -species 'Saccharomyces' -e hmmer \
           -dir $WDR/repeatmasker/${SQ}-x-${SQSET}_megablast    \
             2> $WDR/repeatmasker/${SQ}-x-${SQSET}_megablast.rptmskr.log 1>&2

# You can run now over the whole genome assemblies # CAUTION! it may take too long
#
mkdir -vp $WDR/repeatmasker/${SQSET}_k63_graph.contig
$CONDABIN/RepeatMasker \
           $WDR/soapdenovo/$SQSET/${SQSET}_k63_graph.contig.fa \
           -qq -small -gff -species 'Saccharomyces' -e hmmer \
           -dir $WDR/repeatmasker/${SQSET}_k63_graph.contig \
             2> $WDR/repeatmasker/${SQSET}_k63_graph.contig.rptmskr.log 1>&2

# You can also mask the second assembly...
```

Masking for the second assembly (SRR7548448):

[5] A.F.A. Smit, R. Hubley, and P. Green, "RepeatMasker at http://repeatmasker.org" (*unpublished*).

```
#%

SQ=chrM;
SQSET=SRR7548448;
##CONDABIN=$HOME/.conda/envs/CG_exercises/bin; #Doesen't exist
CONDABIN=$BIN;  #due to symbolic links, its the same as CONDABIN=$BIN/RepeatMasker-4.1.3

conda activate CG_exercises; #Same as BScBI-CG2425_exercises
mkdir -vp $WDR/repeatmasker;

#chrM contigs set for a test
mkdir -vp $WDR/repeatmasker/${SQ}-x-${SQSET}_megablast
$CONDABIN/RepeatMasker \
           $WDR/blast/${SQ}-x-${SQSET}/${SQ}-x-${SQSET}\_megablast.fa \
           -qq -small -gff -species 'Saccharomyces' -e hmmer \
           -dir $WDR/repeatmasker/${SQ}-x-${SQSET}_megablast    \
             2> $WDR/repeatmasker/${SQ}-x-${SQSET}_megablast.rptmskr.log 1>&2

##TODO
#Whole genome
mkdir -vp $WDR/repeatmasker/${SQSET}_k63_graph.contig
$CONDABIN/RepeatMasker \
           $WDR/soapdenovo/$SQSET/${SQSET}_k63_graph.contig.fa \
           -qq -small -gff -species 'Saccharomyces' -e hmmer \
           -dir $WDR/repeatmasker/${SQSET}_k63_graph.contig \
             2> $WDR/repeatmasker/${SQSET}_k63_graph.contig.rptmskr.log 1>&2

#-%#
```

Include a summary table here listing the repeats families found with the total counts. Describe the number of masked nucleotides too.

Repeat Type	Number of Elements	Total Length	Percentage of Sequence
Retroelements	0	0 bp	0.00%
DNA Transposons	0	0 bp	0.00%
Rolling-circles	0	0 bp	0.00%
Unclassified	0	0 bp	0.00%
Small RNA	0	0 bp	0.00%
Satellites	0	0 bp	0.00%
Simple Repeats	3248	135605 bp	1.14%
Low Complexity	505	23684 bp	0.20%

Table 3: **Repeats and masked nucleotides of `SRR6130428` summary.**

Repeats the second sequence(SRR7548448):

Repeat Type	Number of Elements	Total Length	Percentage of Sequence
Retroelements	0	0 bp	0.00%
DNA Transposons	0	0 bp	0.00%
Rolling-circles	0	0 bp	0.00%
Unclassified	0	0 bp	0.00%
Small RNA	0	0 bp	0.00%
Satellites	0	0 bp	0.00%
Simple Repeats	2204	118277 bp	1.03%
Low Complexity	160	8134 bp	0.07%

Table 4: **Repeats and masked nucleotides of `SRR7548448` summary.**

4 Discussion

We focused on metrics such as sequence contiguity (N50 values), alignment coverage, and completeness. SRR6130428 showed superior assembly quality with higher BUSCO completeness scores (98.8%) and higher N50 value, which indicates longer sequences. SRR7548448 shows a smaller completeness score (12.2%), probably because of missing or fragmented sequences.

The alignment to the reference chromosomes(M and I) shows a better continuity (better than average) hinting at a better performance in the mitochondrial assembly. Also coverage was more consistent across these mitochondrial chromosomes, which implies more reliability in the alignment.

RepeatMasker identifies repetitive sequences within the assemblies which can aid in future analysis. Masking these sequences minimizes biases from redundant sequences, which can help in later analysis.

5 Appendices

5.1 Software

We have used the following versions:

```
uname -a
# Linux aleph 5.15.0-48-generic #54-Ubuntu SMP
# Fri Aug 26 13:26:29 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux

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

infoseq -version
# EMBOSS:6.6.0.0

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

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

mamba --version
# mamba 1.4.2
# conda 23.3.1

bunzip2 --version
# bzip2, a block-sorting file compressor.  Version 1.0.8, 13-Jul-2019.

mummerplot -v
# mummerplot 3.5

busco -v
# BUSCO 5.4.3

RepeatMasker -v
# RepeatMasker version 4.1.5
```

5.2 Supplementary files

5.2.1 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
  - texlive-core
  - pandoc
  - pandocfilters
  - emboss
  - jellyfish
  - sra-tools
  - seqtk
  - fastqc
  - trimmomatic
  - samtools
  - bamtools
  - picard
  - bwa
  - bowtie2
  - soapdenovo2
  - igv
  - blast
  - gnuplot
  - graphicsmagick
  - mummer
  - hmmer
  - bbmap
  - busco
  - repeatmasker
  - trf
  # R-packages
  - r-ggplot2
  - r-reshape2
```

5.2.2 Project specific scripts

5.2.3 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_exercise04"; #->            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;
    /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...
export GENOMEDIR=$WDR/seqs/S288C_reference_genome_R64-5-1_20240529;
export GENOMEFSA=S288C_reference_sequence_R64-5-1_20240529;
```

5.3 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`).