BSc BI-CG

Practicals Report

Jan Izquierdo

Exercise 03

— October 31, 2024—

Computational Genomics 2024–25

BSc in Bioinformatics

UPF/ESCI – UPC – UB – UAB

Contents

List of Tables

List of Figures

1 Introduction

Different high-throughput sequencing experiments have been performed over genomic DNA samples of *Saccharomyces cerevisiae* and paired-end raw reads were provided. We were asked to assemble the reads obtained from at least two of the suggested datasets. Those sequence sets may have differences in sequencing methodology, but mainly they differ in whole-genome coverage. We can evaluate if differences in coverage can have an impact on the final assembled genome as we already have a reference genome.

1.1 Objectives

- To check qualities and other properties of sequencing reads.
- To analyze the *k*-mers counts distribution for those reads.
- To start an assembly protocol: cleaning reads and assembling the contigs.
- To map the reads back to the assembly so we can visualize the coverage across the sequences.

1.2 Prerequisites

1.2.1 Installing required software

As for the previous practicals, 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:

```
#################################
# NCBI SRA Toolkit https://github.com/ncbi/sra-tools/wiki/01.-Downloading-SRA-Toolkit
wget https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/3.0.0/sratoolkit.3.0.0-ubuntu64.tar.gz \
     -O ./bin/sratoolkit.3.0.0-ubuntu64.tar.gz
pushd ./bin
tar -zxvf sratoolkit.3.0.0-ubuntu64.tar.gz
cd sratoolkit.3.0.0-ubuntu64
popd

export PATH=$WDR/bin/sratoolkit.3.0.0-ubuntu64/bin:$PATH
# NOTE: you can add the above setting to the projectvars.sh file

# seqtk - sampling, trimming, fastq2fasta, subsequence, reverse complement
sudo apt-get install seqtk

# jellyfish - count k-mers in DNA sequences (installed on previous exercise)
sudo apt-get install jellyfish

# fastqc - quality control for NGS sequence data
sudo apt-get install fastqc

# trimmomatic - flexible read trimming tool for Illumina NGS data
sudo apt-get install trimmomatic

# samtools - processing sequence alignments in SAM and BAM formats
# bamtools - toolkit for manipulating BAM (genome alignment) files
# picard-tools - Command line tools to manipulate SAM and BAM files
sudo apt-get install samtools bamtools picard-tools

# Downloading the latest version of picard
# see https://github.com/broadinstitute/picard/releases/latest
wget https://github.com/broadinstitute/picard/releases/download/2.27.5/picard.jar \
     -O $BIN/picard.jar

# bwa - Burrows-Wheeler Aligner
# bowtie2 - ultrafast memory-efficient short read aligner
sudo apt-get install bwa bowtie2

# soapdenovo2 - short-read assembly method to build de novo draft assembly
sudo apt-get install soapdenovo2

# igv - Integrative Genomics Viewer
sudo apt-get install igv
```

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 27),

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_03.tgz
cd exercise_03

# Rename report file including your "NAME" and "SURNAME"
mv -v README_BScBICG2425_exercise03_SURNAME_NAME.md \
      README_BScBICG2425_exercise03_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_exercise03_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 Datasets: Yeast Sequences

The budding yeast *Saccharomyces cerevisiae* is one of the major model organisms for understanding cellular and molecular processes in eukaryotes. This single-celled organism is also important in industry, where it is used to make bread, beer, wine, enzymes, and pharmaceuticals. The *S. cerevisiae* genome is approximately 12Mbp, distributed in 16 chromosomes. Here, we are going to work on raw reads from four different sequencing experiments that have been already submitted to the Short Reads Archive (SRA) and we will compare the resulting assemblies against the reference *S. cerevisiae* (strain S288C). Table 1 summarizes the information about those sets.

Sequence set	SRA accession	Sequencer	Run info	Run ID
Original WT strain	SRX3242873	Illumina HiSeq 4000	2.8M spots, 559.2M bases, 221.2Mb sra	SRR6130428
S288c	SRX1746300	Illumina HiSeq 2000	5.3M spots, 1.1G bases, 733.6Mb sra	SRR3481383
MiSeq PE-sequencing of SAMD00065885	DRX070537	Illumina MiSeq	5.6M spots, 3.3G bases, 1.6Gb sra	DRR076693
S288c genomic DNA library	SRX4414623	Illumina HiSeq 2500	43.1M spots, 8.6G bases, 3.2Gb sra	SRR7548448

Table 1: **Summary of raw reads datasets that we can use from SRA database.** Smaller set will be used for the examples, and you can choose another one to complete the exercise.

2.1 Downloading Reference Genome

We will get first the reference genome[1], *S. cerevisiae* (strain S288C) (assembly R64). This genome version has 16 chromosome sequences, including the mitochondrion genome, totaling 12,157,105bp (see Table 2).

```
URLBASE="https://downloads.yeastgenome.org/sequence/S288C_reference/genome_releases";

wget ${URLBASE}/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.fsa.gz;
# you can export those variables from your projectvars.sh user's section

zcat $GENOMEDIR/$GENOMEFSA | \
  infoseq -only -length -noheading -sequence fasta::stdin \
    2> /dev/null | \
  gawk '{ s+=$1 }
     END{ printf "# Yeast genome size (v.R64.4.1): %dbp\n", s }';
# Yeast genome size (v.R64.4.1): 12157105bp

#
### NOTE ###
#
# 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 | \
  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);
```

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

```
      sub(/\].*$/,"",L);
      sub(/_/,"\\_",$3);
          printf "%15s & %10s & %12d & %8.2f\\%% \\\\n",
                 L, $3, $6, $7;
        }' > $WDR/stats/chromosomeinfo.tex;
```

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

2.2 Downloading Sequencing Runs

In order to run the exercise, we will start with the smaller dataset, SRR6130428, which has a 221.2Mb sra file to be downloaded.

```
## The example reads dataset: SRR6130428

# 221.2Mb downloads
mkdir -vp $WDR/seqs/SRR6130428
prefetch -v SRR6130428  -O $WDR/seqs
#
# Ensure that the SRA file is stored as $WDR/seqs/SRR6130428/SRR6130428.sra,
#   not as $WDR/seqs/SRR6130428/SRR6130428/SRR6130428.sra (you can move it to parent folder if needed).
#
# NOTE: Old versions of prefetch do not implement option "-O",
#       so that they can return an error. Just remove that option
#       and its argument, and look for the downloaded files
#       at the following folder: $HOME/ncbi/public/sra
#       You probably will need to move the file with these commands:
#         mkdir -vp $WDR/seqs/SRR6130428/;
#         mv -v $HOME/ncbi/public/sra/SRR6130428.sra $WDR/seqs/SRR6130428/;

fastq-dump -X 2 -Z $WDR/seqs/SRR6130428/SRR6130428.sra;
# Read 2 spots for seqs/SRR6130428/SRR6130428.sra
# Written 2 spots for seqs/SRR6130428/SRR6130428.sra
# @SRR6130428.1 1 length=202
# NGACCTTGGCGTTGGTGTAGGTCACCACTCCGATTTTGAGCTAGAATATGAAATTCATCACTGGAATAAGTTTCATAAGGACAAAGATC --->
# +SRR6130428.1 1 length=202
# #AAFFJJJJJJJJJJJJJJJJJJJJJJJJJJJJFJJJJJJJJJJJJJJJJJJJJJJJJJFJJJJJFJJJJJJJJJJFFJJJJJJJJJJ --->
# @SRR6130428.2 2 length=202
# NTGCTACTCTCATGGTCTCAATACTGCCGCCGACATTCTGTCCCACATACTAAATCTCTTCCCGTCATTATCGCCCGCATCCGGTGCCG --->
# +SRR6130428.2 2 length=202
# #AAAAFFAFFJJJJJJJJJJJJJJJJJJFJJJFJJJJJJJFJJJJAJJJFJ<JJFA<FFAJJFFJJ<JFJJJJJJJFFJ7JJJJFJJJJ --->

SQSET=SRR6130428;
fastq-dump -I --split-files $WDR/seqs/${SQSET}/${SQSET}.sra \
           --gzip --outdir $WDR/seqs/${SQSET}/;
#    Read 2768518 spots for seqs/SRR6130428/SRR6130428.sra
# Written 2768518 spots for seqs/SRR6130428/SRR6130428.sra
```

2.2.1 Exercise 1

Here we have the commands to download **any** of the other three reads datasets.

```
## Choose another dataset for an alternative assembly

# 733.6Mb downloads
mkdir -vp $WDR/seqs/SRR3481383;
prefetch -v SRR3481383 -O $WDR/seqs/SRR3481383;

# 1.6Gb downloads
mkdir -vp $WDR/seqs/DRR076693;
prefetch -v DRR076693  -O $WDR/seqs/DRR076693;

# 3.2Gb downloads ##done
mkdir -vp $WDR/seqs/SRR7548448;
prefetch -v SRR7548448 -O $WDR/seqs/SRR7548448;
#%
mv ./seqs/SRR7548448/SRR7548448 ./seqs/SRR7548448
#-%#

for SQSET in SRR3481383 DRR076693 SRR7548448;
  do {
    echo "# Working on $n" 1>&2;
    if [ -e $WDR/seqs/${SQSET}/${SQSET}.sra ];
      then
        fastq-dump -I --split-files $WDR/seqs/${SQSET}/${SQSET}.sra \
                      --gzip --outdir $WDR/seqs/${SQSET}/;
      else
        echo "# SRA FILE NOT FOUND: $WDR/seqs/${SQSET}/${SQSET}.sra" 1>&2;
      fi;
  }; done;
# # Working on SRR3481383
# Read 5346334 spots for seqs/SRR3481383/SRR3481383.sra
# Written 5346334 spots for seqs/SRR3481383/SRR3481383.sra
# # Working on DRR076693
# Read 5625017 spots for seqs/DRR076693/DRR076693.sra
# Written 5625017 spots for seqs/DRR076693/DRR076693.sra
# # Working on DRR076693
# Read 43091394 spots for seqs/SRR7548448/SRR7548448.sra
# Written 43091394 spots for seqs/SRR7548448/SRR7548448.sra
```

Sequence of my choosing:

```
#%
SQSET=SRR7548448
echo "# Working on $n" 1>&2;
if [ -e $WDR/seqs/${SQSET}/${SQSET}.sra ];
then
fastq-dump -I --split-files $WDR/seqs/${SQSET}/${SQSET}.sra \
              --gzip --outdir $WDR/seqs/${SQSET}/;
else
echo "# SRA FILE NOT FOUND: $WDR/seqs/${SQSET}/${SQSET}.sra" 1>&2;
```

```
fi;
# Working on
Read 43091394 spots for /home/jj/Desktop/Bioinformatics/3rd_year/1term/Computational_genomics/Seminars/exe
Written 43091394 spots for /home/jj/Desktop/Bioinformatics/3rd_year/1term/Computational_genomics/Seminars/
#-%#
```

3 The Assembly Protocol

3.1 Exploratory data analysis of the raw reads

On this initial step, we will run fastQC over FASTQ files for the forward (R1) and reverse (R2) raw reads sets. You will need to compare those two sets from three of the resulting plots: the quality distribution per base position (boxplots), the base content per position (lineplots), and the read sequences GC content distribution (lineplot). The program will generate an HTML summary page, as well as a zip file containing all the figures in a folder and some results in tabular format.

```
export SQSET=SRR6130428;

mkdir -vp $WDR/seqs/${SQSET}/${SQSET}.QC;

for READSET in 1 2;
  do {
    echo "# Running fastQC on $SQSET R${READSET}..." 1>&2;
    fastqc -t 8 --format fastq                                  \
           --contaminants $BIN/fastqc_conf/contaminant_list.txt \
           --adapters     $BIN/fastqc_conf/adapter_list.txt     \
           --limits       $BIN/fastqc_conf/limits.txt           \
           -o $WDR/seqs/${SQSET}/${SQSET}.QC                    \
              $WDR/seqs/${SQSET}/${SQSET}_${READSET}.fastq.gz   \
           2> $WDR/seqs/${SQSET}/${SQSET}.QC/fastQC_${SQSET}_${READSET}.log 1>&2;
  }; done;
mkdir: created directory '/home/jj/Desktop/Bioinformatics/3rd_year/1term/Computational_genomics/Seminars/e
# Running fastQC on SRR6130428 R1...
# Running fastQC on SRR6130428 R2...
```

For my sequence

```
#%
export SQSET=SRR7548448;

mkdir -vp $WDR/seqs/${SQSET}/${SQSET}.QC;

for READSET in 1 2;
  do {
    echo "# Running fastQC on $SQSET R${READSET}..." 1>&2;
    fastqc -t 8 --format fastq                                  \
           --contaminants $BIN/fastqc_conf/contaminant_list.txt \
           --adapters     $BIN/fastqc_conf/adapter_list.txt     \
           --limits       $BIN/fastqc_conf/limits.txt           \
           -o $WDR/seqs/${SQSET}/${SQSET}.QC                    \
              $WDR/seqs/${SQSET}/${SQSET}_${READSET}.fastq.gz   \
           2> $WDR/seqs/${SQSET}/${SQSET}.QC/fastQC_${SQSET}_${READSET}.log 1>&2;
  }; done;
mkdir: created directory '/home/jj/Desktop/Bioinformatics/3rd_year/1term/Computational_genomics/Seminars/e
# Running fastQC on SRR7548448 R1...
# Running fastQC on SRR7548448 R2...

#-%#
```

You can now open the HTML page or the zip file that fastQC has produced for each reads set, and save the requested figures into the exercise images folder. Rename those images in png format so they fit in the table cells that make Figure 2.

Unzipping and moving the images for R1 and R2 in sequence SRR6130428

```
#%

#For R1

export SQSET=SRR6130428

#For R1
READSET=1

pathInZip=${SQSET}_${READSET}_fastqc/Images
OutZip=$WDR/images

unzip -j $WDR/seqs/${SQSET}/${SQSET}.QC/${SQSET}_${READSET}_fastqc.zip $pathInZip/per_base_quality.png $pa

mv $OutZip/per_base_quality.png $OutZip/fastqc_${SQSET}_quality_R${READSET}.png
mv $OutZip/per_base_sequence_content.png $OutZip/fastqc_${SQSET}_basecontent_R${READSET}.png
mv $OutZip/per_sequence_gc_content.png $OutZip/fastqc_${SQSET}_gccontent_R${READSET}.png

#For R2
READSET=2

pathInZip=${SQSET}_${READSET}_fastqc/Images
OutZip=$WDR/images

unzip -j $WDR/seqs/${SQSET}/${SQSET}.QC/${SQSET}_${READSET}_fastqc.zip $pathInZip/per_base_quality.png $pa

mv $OutZip/per_base_quality.png $OutZip/fastqc_${SQSET}_quality_R${READSET}.png
mv $OutZip/per_base_sequence_content.png $OutZip/fastqc_${SQSET}_basecontent_R${READSET}.png
mv $OutZip/per_sequence_gc_content.png $OutZip/fastqc_${SQSET}_gccontent_R${READSET}.png

#-%#
```

Unzipping and moving the images for R1 and R2 in sequence SRR7548448

```
#%

#For R1

export SQSET=SRR7548448

#For R1
READSET=1

pathInZip=${SQSET}_${READSET}_fastqc/Images
OutZip=$WDR/images

unzip -j $WDR/seqs/${SQSET}/${SQSET}.QC/${SQSET}_${READSET}_fastqc.zip $pathInZip/per_base_quality.png $pa

mv $OutZip/per_base_quality.png $OutZip/fastqc_${SQSET}_quality_R${READSET}.png
mv $OutZip/per_base_sequence_content.png $OutZip/fastqc_${SQSET}_basecontent_R${READSET}.png
mv $OutZip/per_sequence_gc_content.png $OutZip/fastqc_${SQSET}_gccontent_R${READSET}.png

#For R2
READSET=2

pathInZip=${SQSET}_${READSET}_fastqc/Images
OutZip=$WDR/images

unzip -j $WDR/seqs/${SQSET}/${SQSET}.QC/${SQSET}_${READSET}_fastqc.zip $pathInZip/per_base_quality.png $pa

mv $OutZip/per_base_quality.png $OutZip/fastqc_${SQSET}_quality_R${READSET}.png
mv $OutZip/per_base_sequence_content.png $OutZip/fastqc_${SQSET}_basecontent_R${READSET}.png
mv $OutZip/per_sequence_gc_content.png $OutZip/fastqc_${SQSET}_gccontent_R${READSET}.png

#-%#
```

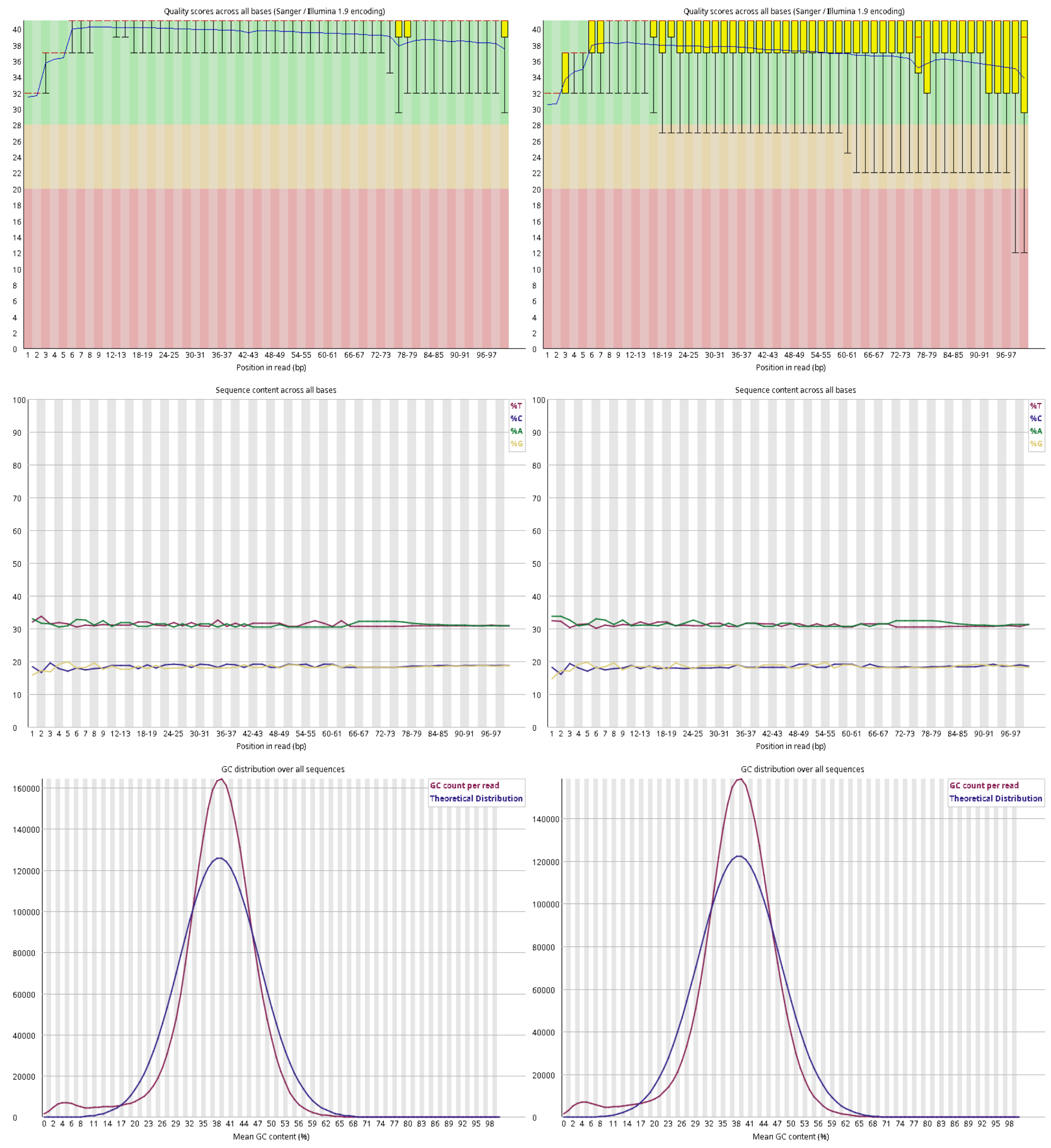


Figure 1: **Raw reads basic sequence analyses for `SRR6130428`.** Left column shows results for the forward reads (R1), right column for the reverse reads (R2). On top panels we can observe phred scores distribution per base position, mid panels correspond to the base composition per position, while the bottom ones show the GC content distribution across reads.

3.2 k-mer content analysis with `jellyfish`

In order to analyze the distribution of the k-mers found in the reads sequences, we will run again `jellyfish`. Now we will change the k-mer size parameter so we can spot differences in counts.

NOTE: Take care that this step requires lots of computer memory. If you are experiencing problems due to this fact you can consider to get a subsample of the reads, for instance a 50%, 25%, or 10%, taken from the begining of the R1 and R2 files or just randomly chosen. Remember then to include the commands you have used to filter out the reads for the downstream analyses.

```
export JLD=$WDR/jellyfish;
export TMPDIR=$WDR/tmpsort;

mkdir -vp $JLD $TMPDIR;

# Using a shell variable can facilitate the analysis
# of another dataset later on...
SQSET=SRR6130428;

##
## IMPORTANT ##
#   If you run out ouf memory or it takes more than 6 hours
#   for the jellyfish or the assembly steps,
#   you may consider subsampling the fasq sequences.
#
##  You can take the first 100000 sequences with the "head" command:
#
# gunzip -c $WDR/seqs/${SQSET}/${SQSET}_1.fastq.gz | head -n 400000 | \
#        gzip -c9 - > $WDR/seqs/${SQSET}/${SQSET}_1.subset100k.fastq.gz;
# gunzip -c $WDR/seqs/${SQSET}/${SQSET}_2.fastq.gz | head -n 400000 | \
#        gzip -c9 - > $WDR/seqs/${SQSET}/${SQSET}_2.subset100k.fastq.gz;
#
##  You can also use "seqtk" program (if installed), as follows:
#
# seqtk sample -s11 $WDR/seqs/${SQSET}/${SQSET}_1.fastq.gz 0.1 | \
#         gzip -c9 - > $WDR/seqs/${SQSET}/${SQSET}_1.subset10pct.fastq.gz;
# seqtk sample -s11 $WDR/seqs/${SQSET}/${SQSET}_2.fastq.gz 0.1 | \
#         gzip -c9 - > $WDR/seqs/${SQSET}/${SQSET}_2.subset10pct.fastq.gz;
#
##  The -s option specifies the seed value for the random number generator,
#   it needs to be the same value for file1 and file2 to keep the paired reads
#   on the sampled output. 0.1 argument sets the percent of seqs to sample (10%).
#
##  Just fix the following commands to work with the proper subset files instead.

##

( echo "gunzip -c $WDR/seqs/${SQSET}/${SQSET}_1.fastq.gz";
  echo "gunzip -c $WDR/seqs/${SQSET}/${SQSET}_2.fastq.gz"
  ) > $JLD/${SQSET}.generators;

#
# IMPORTANT:
#     If you have few disk space, it is advisable no to dump
#     the compressed kmers into the *.kmer_seqs.tbl.gz files.
#     In that case, you will need to replace those files with a system call
#     to jellyfish "dump" later on the "FREQS" block on shell loop below,
#     in that case you can comment the "DUMP" block too.
#     Then, "FREQS" block may look like:
#
#       printf " ... FREQS" 1>&2;
#       jellyfish dump -c $JLD/${SQSET}.$Ksize.counts.jf | \
#           gawk '{ print $2 }' | sort -nr | uniq -c \
#                       > $JLD/${SQSET}.$Ksize.kmer_seqs.tbl.counts
#       echo " ... DONE" 1>&2;
#
for Ksize in 16 24 32 48;
  do {
    printf "# K-mer: $Ksize ... COUNTS" 1>&2;
    jellyfish count --mer-len=$Ksize --threads=8 --size=1000000 \
                    --generator=$JLD/${SQSET}.generators -G 12  \
                    --counter-len=16 --min-qual-char='/'        \
                    --timing=$JLD/${SQSET}.$Ksize.time          \
                    --output=$JLD/${SQSET}.$Ksize.counts.jf     \
                          2> $JLD/${SQSET}.$Ksize.jf.log 1>&2;
    printf " ... STATS" 1>&2;
    jellyfish stats $JLD/${SQSET}.$Ksize.counts.jf \
                  > $JLD/${SQSET}.$Ksize.stats.jf \
                2>> $JLD/${SQSET}.$Ksize.jf.log;
    printf " ... DUMPS" 1>&2;
    ( jellyfish dump -c $JLD/${SQSET}.$Ksize.counts.jf | \
        sort --temporary-directory=$TMPDIR \
             -k2,2nr -k1,1 | \
          gzip -vc9 - > $JLD/${SQSET}.$Ksize.kmer_seqs.tbl.gz
      ) 2>> $JLD/${SQSET}.$Ksize.jf.log 1>&2;
    printf " ... FREQS" 1>&2;
    zcat   $JLD/${SQSET}.$Ksize.kmer_seqs.tbl.gz | \
      gawk '{ print $2 }' | uniq -c \
         > $JLD/${SQSET}.$Ksize.kmer_seqs.tbl.counts
    echo " ... DONE" 1>&2;
  }; done;

# as we have generated a sorted list of k-mers (in decreasing order of counts),
# we can take a glimpse to the most abundant sequences, for instance for k=16
zcat $JLD/${SQSET}.16.kmer_seqs.tbl.gz | head -10;
# TTTTTTTTTTTTTTTT 89857
# ATCGGAAGAGCACACG 47075
# ATCATTAAAAAAAAAA 46994
# TCGGAAGAGCACACGT 46939
# CGGAAGAGCACACGTC 46889
# GGAAGAGCACACGTCT 46750
# GAAGAGCACACGTCTG 46713
# AGAGCACACGTCTGAA 46706
# GAGCACACGTCTGAAC 46549
# AAGAGCACACGTCTGA 46420
```

Redo the process but for the chosen sequence:

```
#%
export JLD=$WDR/jellyfish;
export TMPDIR=$WDR/tmpsort;

mkdir -vp $JLD $TMPDIR;

SQSET=SRR7548448

( echo "gunzip -c $WDR/seqs/${SQSET}/${SQSET}_1.fastq.gz";
  echo "gunzip -c $WDR/seqs/${SQSET}/${SQSET}_2.fastq.gz"
  ) > $JLD/${SQSET}.generators;

for Ksize in 16 24 32 48;
  do {
    printf "# K-mer: $Ksize ... COUNTS" 1>&2;
    jellyfish count --mer-len=$Ksize --threads=8 --size=1000000 \
                    --generator=$JLD/${SQSET}.generators -G 12  \
                    --counter-len=16 --min-qual-char='/'        \
                    --timing=$JLD/${SQSET}.$Ksize.time          \
                    --output=$JLD/${SQSET}.$Ksize.counts.jf     \
                          2> $JLD/${SQSET}.$Ksize.jf.log 1>&2;
    printf " ... STATS" 1>&2;
    jellyfish stats $JLD/${SQSET}.$Ksize.counts.jf \
                  > $JLD/${SQSET}.$Ksize.stats.jf \
                2>> $JLD/${SQSET}.$Ksize.jf.log;
    printf " ... DUMPS" 1>&2;
    ( jellyfish dump -c $JLD/${SQSET}.$Ksize.counts.jf | \
        sort --temporary-directory=$TMPDIR \
             -k2,2nr -k1,1 | \
          gzip -vc9 - > $JLD/${SQSET}.$Ksize.kmer_seqs.tbl.gz
      ) 2>> $JLD/${SQSET}.$Ksize.jf.log 1>&2;
    printf " ... FREQS" 1>&2;
    zcat  $JLD/${SQSET}.$Ksize.kmer_seqs.tbl.gz | \
      gawk '{ print $2 }' | uniq -c \
         > $JLD/${SQSET}.$Ksize.kmer_seqs.tbl.counts
    echo " ... DONE" 1>&2;
  }; done;
# K-mer: 16 ... COUNTS ... STATS ... DUMPS ... FREQS ... DONE
# K-mer: 24 ... COUNTS ... STATS ... DUMPS ... FREQS ... DONE
# K-mer: 32 ... COUNTS ... STATS ... DUMPS ... FREQS ... DONE
# K-mer: 48 ... COUNTS ... STATS ... DUMPS ... FREQS ... DONE

#-%#
```

Once we have computed the frequencies of the k-mer counts, (the `*.kmer_seqs.tbl.counts` files), with the last command of the previous shell loop, we can produce a plot to compare the distributions of the counts for different k sizes, $k = (16, 24, 32, 48)$, considered for this practical. From those results, we can decide to filter out those reads, but it is also a task that can be undertaken by the assembly software.

```
library(ggplot2);
library(reshape2);

# Loading k-mer counts
DT.K16<-read.table(file="jellyfish/SRR6130428.16.kmer_seqs.tbl.counts",
                   col.names=c("FREQ","COUNT"),header=FALSE);
DT.K24<-read.table(file="jellyfish/SRR6130428.24.kmer_seqs.tbl.counts",
                   col.names=c("FREQ","COUNT"),header=FALSE);
DT.K32<-read.table(file="jellyfish/SRR6130428.32.kmer_seqs.tbl.counts",
                   col.names=c("FREQ","COUNT"),header=FALSE);
DT.K48<-read.table(file="jellyfish/SRR6130428.48.kmer_seqs.tbl.counts",
                   col.names=c("FREQ","COUNT"),header=FALSE);

summary(DT.K16);
#      FREQ              COUNT
# Min.   :      1   Min.   :    1.0
# 1st Qu.:      1   1st Qu.:  507.8
# Median :     18   Median : 1014.5
# Mean   :  11668   Mean   : 3818.7
# 3rd Qu.:     71   3rd Qu.: 1700.2
# Max.   :1894065   Max.   :89857.0
summary(DT.K24);
#      FREQ              COUNT
# Min.   :      1   Min.   :    1.0
# 1st Qu.:      3   1st Qu.:  343.8
# Median :     34   Median :  686.5
# Mean   :  17778   Mean   : 3696.8
# 3rd Qu.:     87   3rd Qu.: 1029.2
# Max.   :2094575   Max.   :45223.0
summary(DT.K32);
#      FREQ                COUNT
# Min.   :      1.0   Min.   :    1.0
# 1st Qu.:      6.0   1st Qu.:  285.5
# Median :     44.0   Median :  570.0
# Mean   :  21538.2   Mean   : 3299.7
# 3rd Qu.:     95.5   3rd Qu.:  854.5
# Max.   :2271353.0   Max.   :43764.0
summary(DT.K48);
#      FREQ                COUNT
# Min.   :      1.0   Min.   :    1.0
# 1st Qu.:      8.2   1st Qu.:  208.2
# Median :     59.5   Median :  415.5
# Mean   :  29490.8   Mean   : 2573.5
# 3rd Qu.:     118.0   3rd Qu.:  622.8
# Max.   :2715811.0   Max.   :36732.0

# Initializing some plot parameters
x.pow10breaks <- c(1,10,100,1000,10000,100000,1000000);
y.pow10breaks <- c(1,10,100,1000,10000,100000,1000000,10000000,100000000);
x.pow10mnbrks <- c();
for (i in 1:length(x.pow10breaks)) {
    x.pow10mnbrks <- c(x.pow10mnbrks,
                       x.pow10breaks[[i]] * c(2,3,4,5,6,7,8,9));
};
y.pow10mnbrks <- c();
for (i in 1:length(y.pow10breaks)) {
    y.pow10mnbrks <- c(y.pow10mnbrks,
                       y.pow10breaks[[i]] * 5);
};

ALL.xlim <- c(1, max(DT.K16$COUNT, DT.K24$COUNT, DT.K32$COUNT, DT.K48$COUNT));
ALL.ylim <- c(1, max(DT.K16$FREQ,  DT.K24$FREQ,  DT.K32$FREQ,  DT.K48$FREQ));

p.brk <- c("K16", "K24", "K32", "K48");
p.lbl <- c("K16" = "k=16", "K24" = "k=24",
           "K32" = "k=32", "K48" = "k=48");
p.lty <- c("K16" = "solid",  "K24" = "solid",
           "K32" = "solid", "K48" = "solid");
p.col <- c("K16" = "orange", "K24" = "red",
           "K32" = "blue",  "K48" = "darkgreen");

p.tit <- "SRR6130428\nk-mers";

# merging the four k-mer counts datasets
KALL <- melt(list(K16 = DT.K16,
                  K24 = DT.K24,
                  K32 = DT.K32,
                  K48 = DT.K48),
             id.vars = c("COUNT", "FREQ"));

# Making the k-mers distribution plot
png(file="images/kmer_count_distribution.SRR6130428.png",
    res=300, width=4000, height=3000);
ggplot(KALL, aes(x=COUNT, y=FREQ,
                 group=as.factor(L1),
                 color=as.factor(L1),
                 linetype=as.factor(L1)
                 )) +
      geom_line() +
      ggtitle("k-mer Counts Distribution for the SRR6130428 Sequence Sets") +
      scale_x_log10(name="k-mer Counts",    breaks=x.pow10breaks) +
      scale_y_log10(name="Count Frequency", breaks=y.pow10breaks) +
      scale_linetype_manual(name   = p.tit,
                            breaks = p.brk,
                            labels = p.lbl,
                            values = p.lty) +
      scale_color_manual(name   = p.tit,
                         breaks = p.brk,
                         labels = p.lbl,
                         values = p.col);
dev.off();
```

Executed without any parameter change

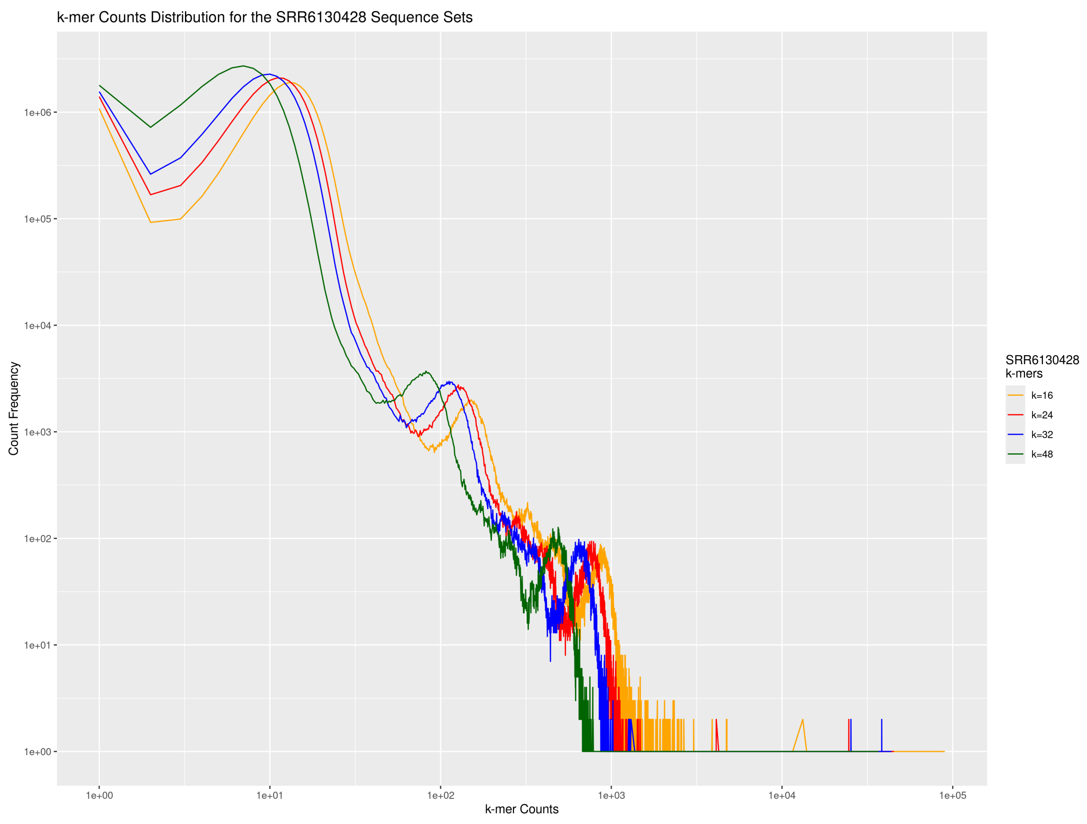


Figure 2: **Analysis of k-mer frequencies for `SRR6130428`.** k-mer counts distribution was calculated on considering all k-mers from reads (R1+R2) for this dataset at different sizes, $k = (16, 24, 32, 48)$.

3.3 Cleaning and trimming reads with `trimmomatic`

A crucial step before starting the assembly is to remove any contaminant sequences, like the sequencing adapters, and low quality segments. For this opurpose, we are going to use `trimmomatic`, but you can take another raw reads cleaner if you like, such as `cutadapt`, to perform this task and compare the results.

```
SQSET=SRR6130428
# to find where trimmomatic was installed in your linux system
find /usr -name trimmomatic.jar;
# /usr/share/java/trimmomatic.jar
#
# on conda environments it may be located at the following path
#   TMBCONDA=$HOME/.conda/envs/BScBI-CG2425_exercises/share/trimmomatic-0.39-2/

# Take the result of the previous linux command as the folder to set on TMC:
export TMC=/usr/share/java/trimmomatic.jar;
#
#  or (conda envs):  export TMC=$TMBCONDA/trimmomatic.jar;
#
# Set this variable to suit your system installation:
export TMA=/usr/share/trimmomatic
#
#  or (conda envs):  export TMA=$TMBCONDA/adapters;
#
# You can find trimmomatic parameters documentation at:
# http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf
export TRMPAR="LEADING:15 TRAILING:15 SLIDINGWINDOW:4:15 MINLEN:30 TOPHRED33";
export TRMPECLP="ILLUMINACLIP:$TMA/TruSeq2-PE.fa:2:30:10";

java -jar $TMC  PE                          \
     $WDR/seqs/${SQSET}/${SQSET}_1.fastq.gz \
     $WDR/seqs/${SQSET}/${SQSET}_2.fastq.gz \
     $WDR/seqs/${SQSET}/${SQSET}_1.trimmo_pe.fastq.gz \
     $WDR/seqs/${SQSET}/${SQSET}_1.trimmo_sg.fastq.gz \
     $WDR/seqs/${SQSET}/${SQSET}_2.trimmo_pe.fastq.gz \
     $WDR/seqs/${SQSET}/${SQSET}_2.trimmo_sg.fastq.gz \
     $TRMPECLP $TRMPAR                             \
  2> $WDR/seqs/${SQSET}/${SQSET}.trimmo.log 1>&2;

tail -n 2 $WDR/seqs/${SQSET}/${SQSET}.trimmo.log;
#  Input Read Pairs:          2768518
#            Both Surviving: 2536084 (91.60%)
#    Forward Only Surviving:  176365 ( 6.37%)
#    Reverse Only Surviving:    4082 ( 0.15%)
#                   Dropped:   51987 ( 1.88%)
# TrimmomaticPE: Completed successfully
```

For the sequence of my choosing

```
#%
SQSET=SRR7548448

export TMC=/usr/share/java/trimmomatic.jar;
export TMA=/usr/share/trimmomatic
export TRMPAR="LEADING:15 TRAILING:15 SLIDINGWINDOW:4:15 MINLEN:30 TOPHRED33";
export TRMPECLP="ILLUMINACLIP:$TMA/TruSeq2-PE.fa:2:30:10";

java -jar $TMC  PE                               \
     $WDR/seqs/${SQSET}/${SQSET}_1.fastq.gz \
     $WDR/seqs/${SQSET}/${SQSET}_2.fastq.gz \
     $WDR/seqs/${SQSET}/${SQSET}_1.trimmo_pe.fastq.gz \
     $WDR/seqs/${SQSET}/${SQSET}_1.trimmo_sg.fastq.gz \
     $WDR/seqs/${SQSET}/${SQSET}_2.trimmo_pe.fastq.gz \
     $WDR/seqs/${SQSET}/${SQSET}_2.trimmo_sg.fastq.gz \
     $TRMPECLP $TRMPAR                             \
  2> $WDR/seqs/${SQSET}/${SQSET}.trimmo.log 1>&2;

tail -n 2 $WDR/seqs/${SQSET}/${SQSET}.trimmo.log;
Input Read Pairs: 43091394
Both Surviving: 35373375 (82.09%)
Forward Only Surviving: 3341514 (7.75%)
Reverse Only Surviving: 631866 (1.47%) Dropped: 3744639 (8.69%)
TrimmomaticPE: Completed successfully
#-%#
```

For the example sequence:

```
#%
#  Input Read Pairs:          2768518
#            Both Surviving: 2536084 (91.60%)
#    Forward Only Surviving:  176365 ( 6.37%)
#    Reverse Only Surviving:    4082 ( 0.15%)
#                   Dropped:   51987 ( 1.88%)
# TrimmomaticPE: Completed successfully
#-%#
```

For my sequence (SRR7548448):

```
#%
#  Input Read Pairs:          43091394
#            Both Surviving: 35373375 (82.09%)
#    Forward Only Surviving:  3341514 (7.75%)
#    Reverse Only Surviving:   631866 (1.47%)
#                   Dropped:  3744639 (8.69%)
# TrimmomaticPE: Completed successfully
#-%#
```

You must compare the results from that `trimmomatic` run with those produced for the second set of *S. cerevisiae* reads you have choosen to analyze from Table 1. Next we should check the new reads filtered, i.e. by running `fastqc` over the pair-end reads from `trimmomatic`.

```
SQSET=SRR6130428;

mkdir -vp $WDR/seqs/${SQSET}/${SQSET}.trimmo.QC;

for READSET in 1 2;
  do {
    echo "# Running fastQC on trimmed PE reads $SQSET R${READSET}..." 1>&2;
    fastqc -t 8 --format fastq                                    \
           --contaminants $BIN/fastqc_conf/contaminant_list.txt \
           --adapters     $BIN/fastqc_conf/adapter_list.txt     \
           --limits       $BIN/fastqc_conf/limits.txt           \
           -o $WDR/seqs/${SQSET}/${SQSET}.trimmo.QC             \
              $WDR/seqs/${SQSET}/${SQSET}_${READSET}.trimmo_pe.fastq.gz \
           2> $WDR/seqs/${SQSET}/${SQSET}.trimmo.QC/fastQC_${SQSET}_${READSET}.log 1>&2;
  }; done;
# Running fastQC on trimmed PE reads SRR6130428 R1...
# Running fastQC on trimmed PE reads SRR6130428 R2...
```

Check the new filtered read for my chosen sequence as well:

```
#%
SQSET=SRR7548448

mkdir -vp $WDR/seqs/${SQSET}/${SQSET}.trimmo.QC;

for READSET in 1 2;
  do {
    echo "# Running fastQC on trimmed PE reads $SQSET R${READSET}..." 1>&2;
    fastqc -t 8 --format fastq                                    \
           --contaminants $BIN/fastqc_conf/contaminant_list.txt \
           --adapters     $BIN/fastqc_conf/adapter_list.txt     \
           --limits       $BIN/fastqc_conf/limits.txt           \
           -o $WDR/seqs/${SQSET}/${SQSET}.trimmo.QC             \
              $WDR/seqs/${SQSET}/${SQSET}_${READSET}.trimmo_pe.fastq.gz \
           2> $WDR/seqs/${SQSET}/${SQSET}.trimmo.QC/fastQC_${SQSET}_${READSET}.log 1>&2;
  }; done;
# Running fastQC on trimmed PE reads SRR7548448 R1...
# Running fastQC on trimmed PE reads SRR7548448 R2...
#-%#
```

IMPORTANT: Now, you can embed here a figure with the same panels as in Figure~2, summarizing nucleotide quality, nucleotide composition, and GC content distributions for `R1` and `R2` cleaned reads.

Unizpping the files for SRR6130428:

```
#%

#For R1

export SQSET=SRR6130428

#For R1
READSET=1
```

```
pathInZip=${SQSET}_${READSET}.trimmo_pe_fastqc/Images
OutZip=$WDR/images

unzip -j $WDR/seqs/${SQSET}/${SQSET}.trimmo.QC/${SQSET}_${READSET}.trimmo_pe_fastqc.zip $pathInZip/per_bas

mv $OutZip/per_base_quality.png $OutZip/Tfastqc_${SQSET}_quality_R${READSET}.png
mv $OutZip/per_base_sequence_content.png $OutZip/Tfastqc_${SQSET}_basecontent_R${READSET}.png
mv $OutZip/per_sequence_gc_content.png $OutZip/Tfastqc_${SQSET}_gccontent_R${READSET}.png

#For R2
READSET=2

pathInZip=${SQSET}_${READSET}.trimmo_pe_fastqc/Images
OutZip=$WDR/images

unzip -j $WDR/seqs/${SQSET}/${SQSET}.trimmo.QC/${SQSET}_${READSET}.trimmo_pe_fastqc.zip $pathInZip/per_bas

mv $OutZip/per_base_quality.png $OutZip/Tfastqc_${SQSET}_quality_R${READSET}.png
mv $OutZip/per_base_sequence_content.png $OutZip/Tfastqc_${SQSET}_basecontent_R${READSET}.png
mv $OutZip/per_sequence_gc_content.png $OutZip/Tfastqc_${SQSET}_gccontent_R${READSET}.png

#-%#
```

Figure is named fastqcsetTrim

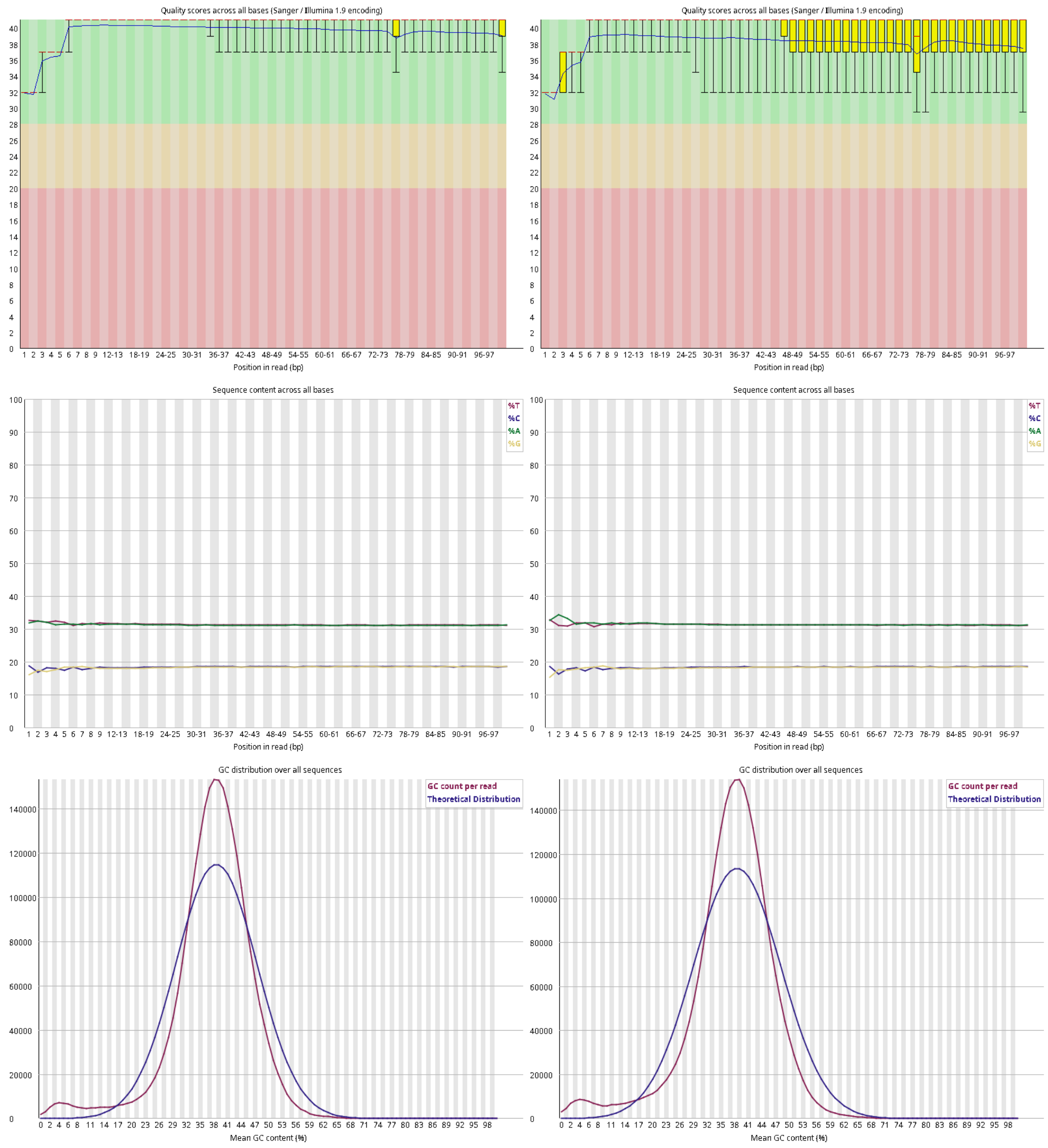


Figure 3: **Raw reads basic sequence analyses for trimmed `SRR6130428`.** Left column shows results for the forward reads (R1), right column for the reverse reads (R2). On top panels we can observe phred scores distribution per base position, mid panels correspond to the base composition per position, while the bottom ones show the GC content distribution across reads.

At this point, we can estimate the sequencing coverage, as we already have the size of the reference genome, and the total amount of nucleotides generated by the sequencing projects.

```
SQSET=SRR6130428
# raw PE-reads
gunzip -c $WDR/seqs/${SQSET}/${SQSET}_[12].fastq.gz | \
  infoseq -sequence fastq::stdin -only -length -noheading | \
    gawk '{ s+=$1; n++ }
          END{ printf "# Total %d sequences and %d nucleotides\n", n, s }';
# Total 5,537,036 sequences and 559,240,636 nucleotides

# cleaned PE-reads
gunzip -c $WDR/seqs/${SQSET}/${SQSET}_[12].trimmo_pe.fastq.gz | \
  infoseq -sequence fastq::stdin -only -length -noheading | \
    gawk '{ s+=$1; n++ }
          END{ printf "# Total %d sequences and %d nucleotides\n", n, s }';
# Total 5,072,168 sequences and 499,530,830 nucleotides
```

NOTE: Update the following line according to your results...

For raw and cleaned reads from the first sample we can estimate a sequencing coverage of $559,240,636/12,157,105 = 46.00$X and $499,530,830/12,157,105 = 41.09$X respectively. **Can you provide those numbers for a second SRA set?**

For my chosen SRA set (SRR7548448):

```
#%
SQSET=SRR7548448

# raw PE-reads
gunzip -c $WDR/seqs/${SQSET}/${SQSET}_[12].fastq.gz | \
  infoseq -sequence fastq::stdin -only -length -noheading | \
    gawk '{ s+=$1; n++ }
          END{ printf "# Total %d sequences and %d nucleotides\n", n, s }';
#Display basic information about sequences
# Total 86182788 sequences and 8618278800 nucleotides

# cleaned PE-reads
gunzip -c $WDR/seqs/${SQSET}/${SQSET}_[12].trimmo_pe.fastq.gz | \
  infoseq -sequence fastq::stdin -only -length -noheading | \
    gawk '{ s+=$1; n++ }
          END{ printf "# Total %d sequences and %d nucleotides\n", n, s }';
#Display basic information about sequences
# Total 70746750 sequences and 6938798641 nucleotides
#-%#
```

For raw and cleaned reads from the first sample we can estimate a sequencing coverage of $8,618,278,800/12,157,105 = 708.91$X and $6,938,798,641/12,157,105 = 570.76$X respectively.

3.4 Assembling reads with SOAPdenovo

SOAPdenovo[2] is easy to install and to configure. It has been reported to perform like other more complex tools, generating in some tests less chimeric contigs and missasemblies.

```
# SQSET=SRR6130428;
export SPD=$WDR/soapdenovo/$SQSET;
mkdir -vp $SPD;

# # If your fastq files were compressed using bzip2, not with gzip,
# # this can make soapdenovo to crash; in this case you will need
# # to uncompress the files and provide the proper reference in config file.
```

[2]Luo et al. "SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler". *GigaScience*, 1:18, 2012.

```
# for R in 1 2;
#   do {
#     bunzip2 -c $WDR/seqs/${SQSET}/${SQSET}_$R.trimmo_pe.fastq.bz2 \
#               > $WDR/seqs/${SQSET}/${SQSET}_$R.trimmo_pe.fastq
#   }; done;

GENOMESIZE=12157105;   # estimated or real... ;^D

# Generating the configuration file on the fly

cat > $SPD/${SQSET}_soap.conf <<EOF
# Raw reads from SRA experiment ${SQSET}
# Description: Original WT strain, Illumina HiSeq 4000 (2x100bp)
# Insert size was not described on the SRA project, using 450bp
# ---
# maximal read length
max_rd_len=100
#
[LIB]
# average insert size
avg_ins=450
# if sequence needs to be reversed (0 for short insert PE libs)
reverse_seq=0
# in which part(s) the reads are used (3 for both contig and scaffold assembly)
asm_flags=3
# use only first 100 bps of each read
rd_len_cutoff=100
# in which order the reads are used while scaffolding
rank=1
# cutoff of pair number for a reliable connection (at least 3 for short insert size)
pair_num_cutoff=3
# minimum aligned length to contigs for a reliable read location (at least 32 for short insert size)
map_len=32
# a pair of fastq file, read 1 file should always be followed by read 2 file
# we are going to use pair-end reads after filtering with trimmomatic
q1=$WDR/seqs/${SQSET}/${SQSET}_1.trimmo_pe.fastq.gz
q2=$WDR/seqs/${SQSET}/${SQSET}_2.trimmo_pe.fastq.gz
#
EOF

### IMPORTANT ###
# Just to note that linux program name is soapdenovo2-63mer
# whilst the conda version program name would be SOAPdenovo-63mer

# Building the k-mers graph
soapdenovo2-63mer pregraph -s $SPD/${SQSET}_soap.conf -K 63 -R -p 8 \
                          -o $SPD/${SQSET}_k63_graph               \
                          2> $SPD/${SQSET}_k63_pregraph.log 1>&2;

# Contiging stage
soapdenovo2-63mer contig   -g $SPD/${SQSET}_k63_graph -R -p 8 \
                          2> $SPD/${SQSET}_k63_contig.log 1>&2;

# Mapping reads back over the contigs
soapdenovo2-63mer map      -s $SPD/${SQSET}_soap.conf -p 8 \
                          -g $SPD/${SQSET}_k63_graph      \
                          2> $SPD/${SQSET}_k63_map.log 1>&2;

# Scaffolding contigs if we have long range reads (i.e. a 2kbp mate-pairs run)
# in our case, as we only have a single pair-ends library, we will get a result
# that will be similar or the same as what we have obtained in the contiging stage
soapdenovo2-63mer scaff    -F -p 8 -N $GENOMESIZE            \
                          -g $SPD/${SQSET}_k63_graph_prefix \
                          2> $SPD/${SQSET}_k63_scaff.log 1>&2;

# just by looking at the $SPD/${SQSET}_k63_contig.log file,
# we can retrieve info about contigs assembly, such as N50
tail $SPD/${SQSET}_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.
```

For my chosen sequence:

```
#%
SQSET=SRR7548448;
export SPD=$WDR/soapdenovo/$SQSET;
mkdir -vp $SPD;

GENOMESIZE=12157105;   # estimated or real... ;^D

cat > $SPD/${SQSET}_soap.conf <<EOF
# Raw reads from SRA experiment ${SQSET}
# Description: Original WT strain, Illumina HiSeq 4000 (2x100bp)
# Insert size was not described on the SRA project, using 450bp
# ---
# maximal read length
max_rd_len=100
#
[LIB]
# average insert size
avg_ins=450
# if sequence needs to be reversed (0 for short insert PE libs)
reverse_seq=0
# in which part(s) the reads are used (3 for both contig and scaffold assembly)
asm_flags=3
# use only first 100 bps of each read
rd_len_cutoff=100
# in which order the reads are used while scaffolding
rank=1
# cutoff of pair number for a reliable connection (at least 3 for short insert size)
pair_num_cutoff=3
# minimum aligned length to contigs for a reliable read location (at least 32 for short insert size)
map_len=32
# a pair of fastq file, read 1 file should always be followed by read 2 file
# we are going to use pair-end reads after filtering with trimmomatic
q1=$WDR/seqs/${SQSET}/${SQSET}_1.trimmo_pe.fastq.gz
q2=$WDR/seqs/${SQSET}/${SQSET}_2.trimmo_pe.fastq.gz
#
EOF
##########

soapdenovo2-63mer pregraph -s $SPD/${SQSET}_soap.conf -K 63 -R -p 8 \
                          -o $SPD/${SQSET}_k63_graph               \
                          2> $SPD/${SQSET}_k63_pregraph.log 1>&2;

soapdenovo2-63mer contig   -g $SPD/${SQSET}_k63_graph -R -p 8 \
                          2> $SPD/${SQSET}_k63_contig.log 1>&2;

soapdenovo2-63mer map      -s $SPD/${SQSET}_soap.conf -p 8 \
                          -g $SPD/${SQSET}_k63_graph      \
                          2> $SPD/${SQSET}_k63_map.log 1>&2;

soapdenovo2-63mer scaff    -F -p 8 -N $GENOMESIZE            \
                          -g $SPD/${SQSET}_k63_graph_prefix \
                          2> $SPD/${SQSET}_k63_scaff.log 1>&2;
```

```
# retrieve info about contigs assembly, such as N50
tail $SPD/${SQSET}_k63_contig.log;
# There are 40228 contig(s) longer than 100, sum up 9243800 bp, with average length 229.
# The longest length is 3721 bp, contig N50 is 276 bp,contig N90 is 127 bp.
# 72962 contig(s) longer than 64 output.

#-%#
```

You can check files `*.contig` and `*.scafSeq` in case they were created, for the fasta sequences assembled contigs and scaffolds respectively.

Open questions arise:

- Can you calculate the contigs lengths and plot that distribution?
- What do you think it will happen when using a larger coverage reads set?
- Do you think knowing the real insert size will improve the assembly?

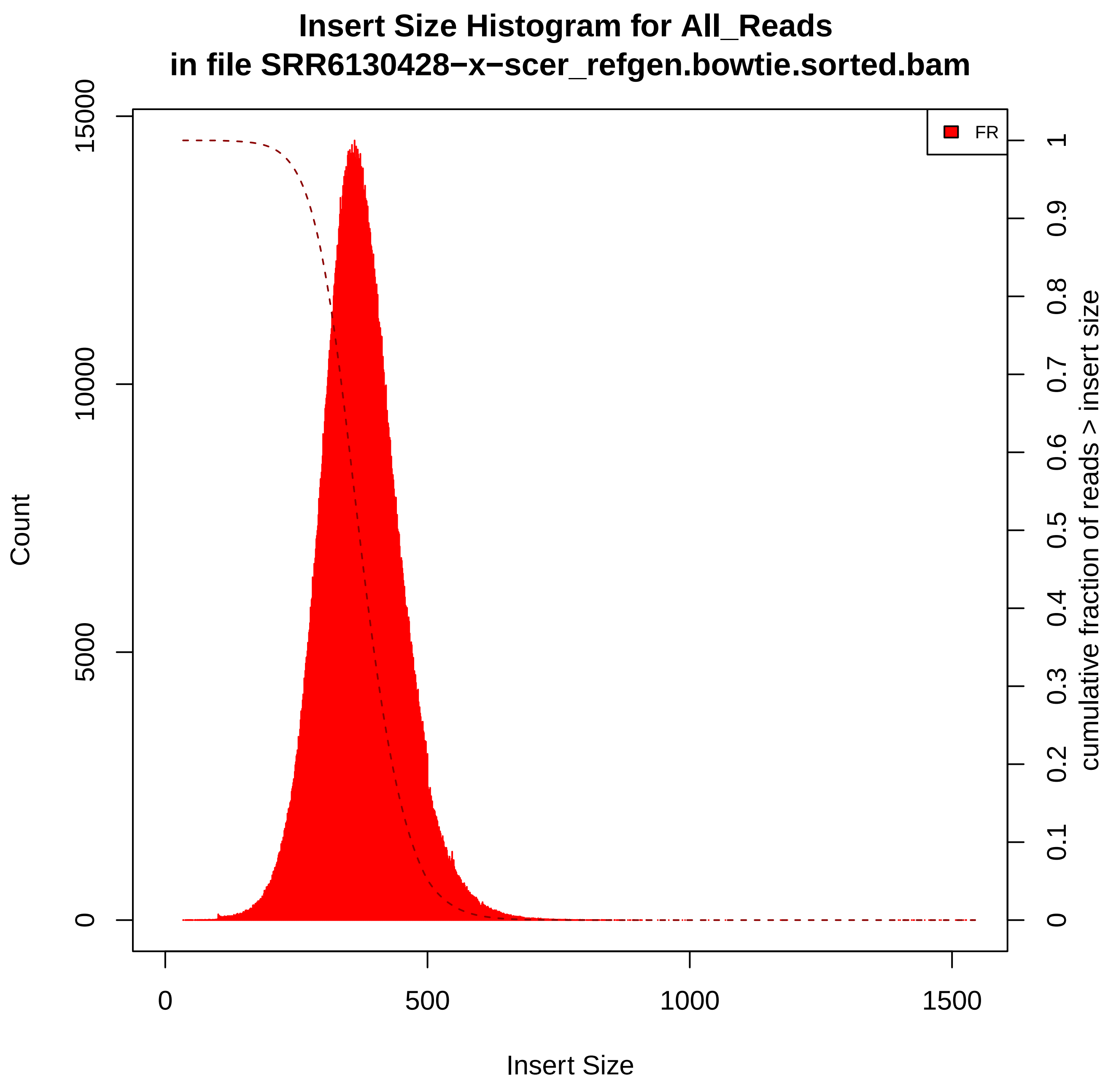
Some can be answered here by comparing with the assembly produced over another of the suggested raw-reads sets.

3.5 Estimating insert size with picard

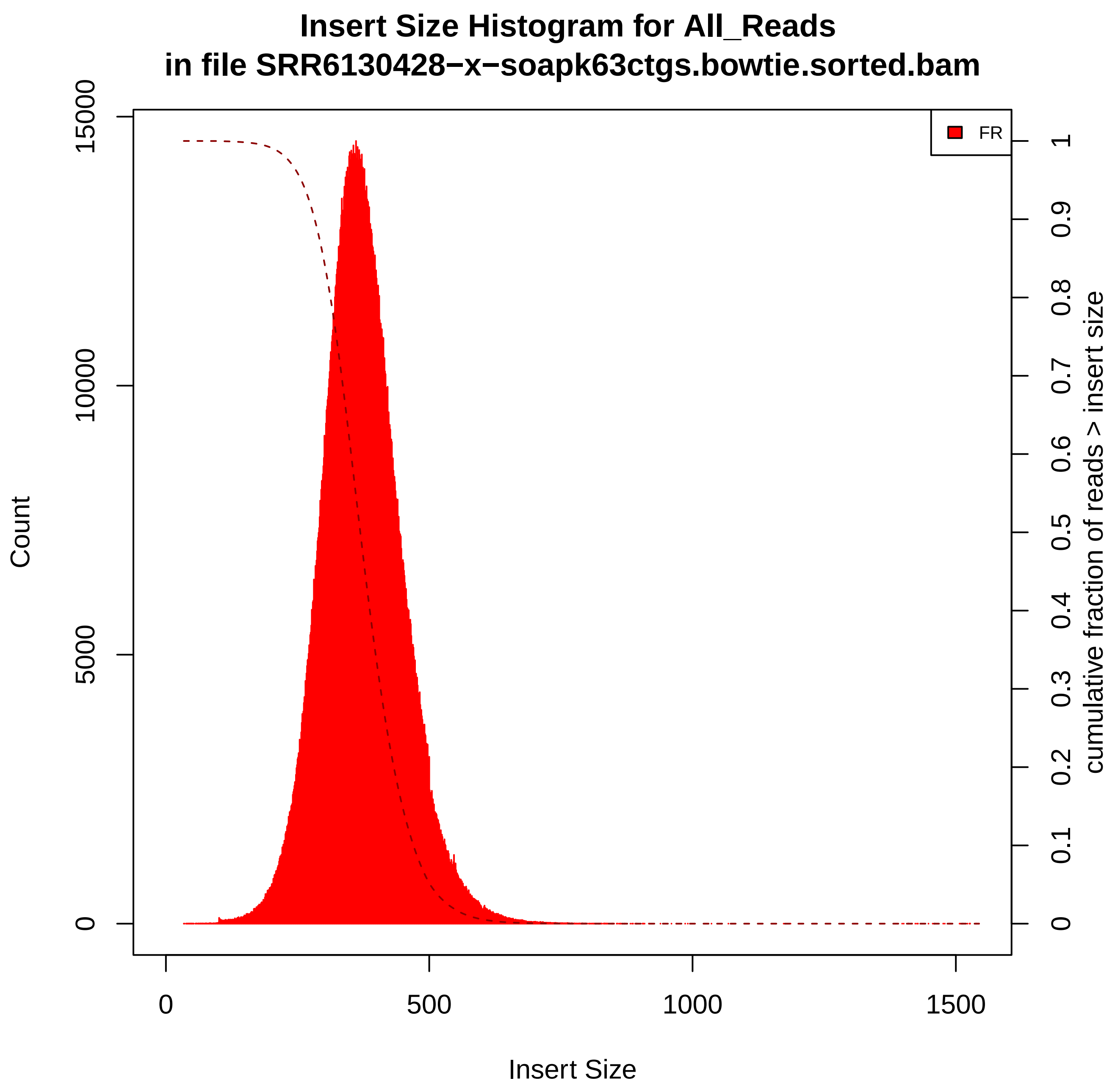
Let's check whether estimated insert size was an educated guess. We must first align the PE reads against the reference genome or the assembly, then we can take the alignments in `bam` format to estimate insert size with `picard` tool.

```
SQSET=SRR6130428;

export GENOMEDIR=$WDR/seqs/S288C_reference_genome_R64-5-1_20240529;
export GENOMEFSA=S288C_reference_sequence_R64-5-1_20240529.fsa.gz;

export BWT=$WDR/bowtie;

mkdir -vp $BWT/$SQSET;

# by linking contigs file to one file with a fasta suffix,
# we can facilitate using those sequences on IGV later on...
ln -vs ./${SQSET}_k63_graph.contig \
       $WDR/soapdenovo/${SQSET}/${SQSET}_k63_graph.contig.fa;

# preparing reference sequence databases for bowtie
bowtie2-build --large-index -o 2 \
              $GENOMEDIR/$GENOMEFSA \
              $BWT/scer_refgenome.bowtiedb \
           2> $BWT/scer_refgenome.bowtiedb.log 1>&2;

bowtie2-build --large-index -o 2 \
              $WDR/soapdenovo/${SQSET}/${SQSET}_k63_graph.contig \
              $BWT/${SQSET}/${SQSET}_k63_graph.contig.bowtiedb \
           2> $BWT/${SQSET}/${SQSET}_k63_graph.contig.bowtiedb.log 1>&2;

# mapping pe reads over reference sequences
TMP=$WDR/tmpsort;
PEfileR1=$WDR/seqs/${SQSET}/${SQSET}_1.trimmo_pe.fastq.gz;
PEfileR2=$WDR/seqs/${SQSET}/${SQSET}_2.trimmo_pe.fastq.gz;

BWTBF=$BWT/${SQSET}/${SQSET}-x-scer_refgen.bowtie;
TMPBF=$TMP/${SQSET}-x-scer_refgen.bowtie;

bowtie2 -q --threads 8 -k 5 -L 12                   \
        --local --sensitive-local --no-unal --met 60 \
        --met-file $BWTBF.metrics                    \
        -x         $BWT/scer_refgenome.bowtiedb      \
        -1          $PEfileR1  \
        -2          $PEfileR2  \
        -S          $TMPBF.sam \
        2>          $BWTBF.log 1>&2;

#Run by lines
( samtools view -Sb -o $TMPBF.bam $TMPBF.sam;
  samtools sort $TMPBF.bam $TMPBF.sorted;
  mv -v $TMPBF.sorted.bam $BWTBF.sorted.bam;
  rm -v $TMPBF.sam $TMPBF.bam
  ) 2> $BWTBF.bowtie2sortedbam.log 1>&2;

samtools index $BWTBF.sorted.bam;

# to find where picard was installed in your linux system
find / -name picard.jar;
# PCRD=/usr/share/java/picard.jar

# Take the result of the previous linux command as the folder to set on TMC:
#
# on conda environments it may be located at the following path
#   PCRD=$HOME/.conda/envs/BScBI-CG2425_exercises/share/picard-3.1.0-0/picard.jar

java -jar $PCRD CollectInsertSizeMetrics             \
          HISTOGRAM_FILE=$BWTBF.insertsize.hist.pdf  \
                   INPUT=$BWTBF.sorted.bam           \
                  OUTPUT=$BWTBF.insertsize_stats.txt \
           ASSUME_SORTED=true \
              DEVIATIONS=25   \
                      2> $BWTBF.insertsize_stats.log;


# now let's check the assembled contigs

BWTBF=$BWT/${SQSET}/${SQSET}-x-soapk63ctgs.bowtie;
TMPBF=$TMP/${SQSET}-x-soapk63ctgs.bowtie;

bowtie2 -q --threads 8 -k 5 -L 12                    \
        --local --sensitive-local --no-unal --met 60 \
        --met-file $BWTBF.metrics                    \
        -x          $BWT/scer_refgenome.bowtiedb     \
        -1          $PEfileR1  \
        -2          $PEfileR2  \
        -S          $TMPBF.sam \
        2>          $BWTBF.log 1>&2;

( samtools view -Sb -o $TMPBF.bam $TMPBF.sam;
  samtools sort $TMPBF.bam $TMPBF.sorted;
  mv -v $TMPBF.sorted.bam $BWTBF.sorted.bam;
  rm -v $TMPBF.sam $TMPBF.bam
  ) 2> $BWTBF.bowtie2sortedbam.log 1>&2;

samtools index $BWTBF.sorted.bam;

java -jar $PCRD CollectInsertSizeMetrics             \
          HISTOGRAM_FILE=$BWTBF.insertsize.hist.pdf  \
                   INPUT=$BWTBF.sorted.bam           \
                  OUTPUT=$BWTBF.insertsize_stats.txt \
           ASSUME_SORTED=true \
              DEVIATIONS=25   \
                      2> $BWTBF.insertsize_stats.log;
```

Regular PDF


Insert Size Histogram for All_Reads
in file SRR6130428−x−scer_refgen.bowtie.sorted.bam
FR
Count
0
5000
10000
15000
Insert Size
0
500
1000
1500
cumulative fraction of reads > insert size
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

K63 in PDF



You can include here the two PDF histograms generated by `picard` from the reads alignment over the reference and `soapdenovo` assemblies. You can open the genome set and the aligned reads using a browser like `igv` or `tablet`.

That's all for the moment, do not remove the data generated on this exercise, we will retake the analyses over assemblies and the reference genome on the forthcoming exercise.

4 Discussion

The untrimmed sequences coverage was 46.00 and 41.09X for R1 and R2 of the example sequence (SRR6130428) and for my chosen sequence (SRR7548448) the coverage was 708.91 for R1 and 570.76 for R2. After trimming, 91.6% of reads survived in the first dataset, while 82.09% survived in the second dataset.

Trimming generally improves sequence alignments by reducing sequencing errors. The lower survival rate of reads in the second dataset could be due to its higher number of contigs over 100bp or a much lower N50, suggesting this assembly may be less compact or a result of sequencing technique differences.

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

jellyfish -V
# jellyfish 2.2.10

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.

apt-cache policy jellyfish fastqc trimmomatic \
                 samtools bamtools picard-tools \
                 bwa bowtie2 soapdenovo2 | \
  gawk '$0 !~ /^ / { printf "#%15s ", $1 }
        $1 == "Installed:" { print $2 }';
#     jellyfish: 2.3.0-12ubuntu2
#        fastqc: 0.11.9+dfsg-5
#   trimmomatic: 0.39+dfsg-2
#      samtools: 1.13-4
#      bamtools: 2.5.1+dfsg-10build1
#  picard-tools: 2.26.10+dfsg-1
#           bwa: 0.7.17-6
#       bowtie2: 2.4.4-1
#   soapdenovo2: 242+dfsg-2

java -jar $TMC -version
# 0.39
#
# picard.jar from https://github.com/broadinstitute/picard/ : 2.27.5
#       https://github.com/broadinstitute/picard/releases/tag/2.27.5

prefetch --version
# prefetch : 3.0.8
#
# sratoolkit from https://github.com/ncbi/sra-tools/wiki/01.-Downloading-SRA-Toolkit : 3.0.7
#       https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/3.0.7/sratoolkit.3.0.7-ubuntu64.tar.gz
```

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
  # 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;
export PATH=$WDR/bin/sratoolkit.3.0.0-ubuntu64/bin:$PATH

#
# 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_exercise03"; #->            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...
```

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 Downsyntax, as well as from the manual pages (just type `man pandoc` and/or `man pandoc_markdown`).