Project 1 (report)

Part A

Q1- Download SRR8185316 (short-read WGS of E.coli) from SRA using the SRA Toolkit in Linux or Windows.

Downloading sratoolkit:

```
[14] !wget --output-document sratoolkit.tar.gz https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-ubuntu64.tar.gz

--2023-12-24 10:25:18--  https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-ubuntu64.tar.gz
Resolving ftp-trace.ncbi.nlm.nih.gov (ftp-trace.ncbi.nlm.nih.gov)... 130.14.250.11, 130.14.250.12, 2607:f220:41e:250::11, ...
Connecting to ftp-trace.ncbi.nlm.nih.gov (ftp-trace.ncbi.nlm.nih.gov)|130.14.250.11|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 93498661 (89M) [application/x-gzip]
Saving to: 'sratoolkit.tar.gz'

sratoolkit.tar.gz   100%[===================>]  89.17M   166MB/s    in 0.5s

2023-12-24 10:25:19 (166 MB/s) - 'sratoolkit.tar.gz' saved [93498661/93498661]
```

Unzip sratoolkit:

```
!tar -vxzf sratoolkit.tar.gz

sratoolkit.3.0.10-ubuntu64/
sratoolkit.3.0.10-ubuntu64/CHANGES
sratoolkit.3.0.10-ubuntu64/schema/
sratoolkit.3.0.10-ubuntu64/schema/align/
sratoolkit.3.0.10-ubuntu64/schema/align/align.vschema
sratoolkit.3.0.10-ubuntu64/schema/align/seq.vschema
sratoolkit.3.0.10-ubuntu64/schema/align/refseq.vschema
sratoolkit.3.0.10-ubuntu64/schema/align/pileup-stats.vschema
sratoolkit.3.0.10-ubuntu64/schema/align/qstat.vschema
sratoolkit.3.0.10-ubuntu64/schema/align/mate-cache.vschema
sratoolkit.3.0.10-ubuntu64/schema/ncbi/
sratoolkit.3.0.10-ubuntu64/schema/ncbi/sra.vschema
sratoolkit.3.0.10-ubuntu64/schema/ncbi/stats.vschema
sratoolkit.3.0.10-ubuntu64/schema/ncbi/varloc.vschema
sratoolkit.3.0.10-ubuntu64/schema/ncbi/spotname.vschema
sratoolkit.3.0.10-ubuntu64/schema/ncbi/seq.vschema
sratoolkit.3.0.10-ubuntu64/schema/ncbi/seq-graph.vschema
sratoolkit.3.0.10-ubuntu64/schema/ncbi/wgs-contig.vschema
sratoolkit.3.0.10-ubuntu64/schema/ncbi/trace.vschema
sratoolkit.3.0.10-ubuntu64/schema/ncbi/clip.vschema
sratoolkit.3.0.10-ubuntu64/schema/ncbi/ncbi.vschema
sratoolkit.3.0.10-ubuntu64/schema/ncbi/pnbrdb.vschema
sratoolkit.3.0.10-ubuntu64/schema/csra2/
sratoolkit.3.0.10-ubuntu64/schema/csra2/stats.vschema
sratoolkit.3.0.10-ubuntu64/schema/csra2/reference.vschema
sratoolkit.3.0.10-ubuntu64/schema/csra2/read.vschema
sratoolkit.3.0.10-ubuntu64/schema/csra2/csra2.vschema
sratoolkit.3.0.10-ubuntu64/schema/sra/
sratoolkit.3.0.10-ubuntu64/schema/sra/helicos.vschema
sratoolkit.3.0.10-ubuntu64/schema/sra/ion-torrent.vschema
sratoolkit.3.0.10-ubuntu64/schema/sra/pacbio.vschema
sratoolkit.3.0.10-ubuntu64/schema/sra/abi.vschema
sratoolkit.3.0.10-ubuntu64/schema/sra/illumina.vschema
sratoolkit.3.0.10-ubuntu64/schema/sra/generic-fastq.vschema
sratoolkit.3.0.10-ubuntu64/schema/sra/pevents.vschema
```

Prefetching SRR8185316:

```
!/content/sratoolkit.3.0.10-ubuntu64/bin/prefetch -O output_dir SRR8185316

2023-12-24T10:34:32 prefetch.3.0.10: Current preference is set to retrieve SRA Normalized Format files with full base quality scores.
2023-12-24T10:34:33 prefetch.3.0.10: 1) Downloading 'SRR8185316'...
2023-12-24T10:34:33 prefetch.3.0.10: SRA Normalized Format file is being retrieved, if this is different from your preference, it may be due to current file availability.
2023-12-24T10:34:33 prefetch.3.0.10:  Downloading via HTTPS...
2023-12-24T10:34:36 prefetch.3.0.10:  HTTPS download succeed
2023-12-24T10:34:37 prefetch.3.0.10:  'SRR8185316' is valid
2023-12-24T10:34:37 prefetch.3.0.10: 1) 'SRR8185316' was downloaded successfully
```

Find the fastq file of SRR8185316 with sratoolkit:

```
[25] !/content/sratoolkit.3.0.10-ubuntu64/bin/fastq-dump --split-files output_dir/SRR8185316/SRR8185316.sra

Read 2297280 spots for output_dir/SRR8185316/SRR8185316.sra
Written 2297280 spots for output_dir/SRR8185316/SRR8185316.sra
```

Q2

Answer the following questions about the short reads fastq file:

1. How many reads are in the fastq file?

Loop through each line in the fastq file and count the number of lines. Then divide number of lines by 4. Because each read is 4 lines. The result is number of reads in the file which is for SRR8185316 2297280 reads.

```
[28] # Part A, Q2 I

     with open("/content/SRR8185316_1.fastq", "r") as f:
       line_count = 0
     # Loop through each line in the file
       for line in f:
         line_count += 1
     # Divide the line count by four to get the read count
       read_count = line_count // 4
       print(f"The number of reads in the fastq file is {read_count}")
```

```
The number of reads in the fastq file is 2297280
```

II. Print the identifier, quality, and sequence of the first read of the fastq file.

Open the fastq file and read the first 4 lines which are belong to first read. Then print identifier(line 1), quality(line4), and sequence(line 2)

```
[29] # Part A Q2 II

     with open("/content/SRR8185316_1.fastq", "r") as f:
       # Read the first four lines of the file
       identifier = f.readline().rstrip()
       sequence = f.readline().rstrip()
       separator = f.readline().rstrip()
       quality = f.readline().rstrip()

       print(f"Identifier: {identifier}")
       print(f"Quality: {quality}")
       print(f"Sequence: {sequence}")
```

```
Identifier: @SRR8185316.1 ERR022075.10741970 length=100
Quality: IIGIIIIIIIIIIIIHIIIIIIIIIIIIIIIII@IHHEHIIIIIIIIIIIHIIHIIIIIIIIIGIHIIHFIIEIIGIHHGH@BE3BB>@>>2?@8?>?A@1
Sequence: AGCGGTACACATTATGGGTCTGCTCTCCGCAGGCGGCGTACACAGCCACGAAGATCACATCATGGCGATGGTAGAACTGGCAGCTGAACGCGGCGCAGAA
```

III. How many times does the TTAAATGGAA subsequence appear in the file?

Define given subsequence. Then read all reads and put them in a list. Then loop through each read and find the subsequence in the sequence part of it and count them.

```
# Part A Q2 III
import re

# Define the subsequence to search for
subseq = "TTAAATGGAA"

count = 0
all_reads = []
with open("/content/SRR8185316_1.fastq", "r") as f:
    for i in range(read_count):          # read all reads
      identifier = f.readline().rstrip()
      sequence = f.readline().rstrip()
      separator = f.readline().rstrip()
      quality = f.readline().rstrip()
      all_reads.append([identifier, sequence, quality])       # save all reads in a list


# Find all the matches of the subsequence in the content
for seq in all_reads:
  matches = re.findall(subseq, seq[1])
  count += len(matches)

print(f"The subsequence {subseq} appears {count} times in the file.")
```

```
The subsequence TTAAATGGAA appears 179 times in the file.
```

IV. Extract the first 1000 sequences of the fastq files (4000 lines).

Reading the first 4000 lines(1000 reads) and save their sequences in a file, then write the list on a new file that is attached to project.

```
# Part A Q2 IV
first1000 = []
 # extract first 1000 reads of all reads in a separate list
with open("/content/SRR8185316_1.fastq", "r") as f:
  for i in range(1000):          # read 1000 reads
    f.readline()
    line = f.readline()
    f.readline()
    f.readline()
    first1000.append(line)
with open("/content/first1000lines.txt", "w") as f:
  for line in first1000:
    f.write(str(line))
    f.write('\n')
```

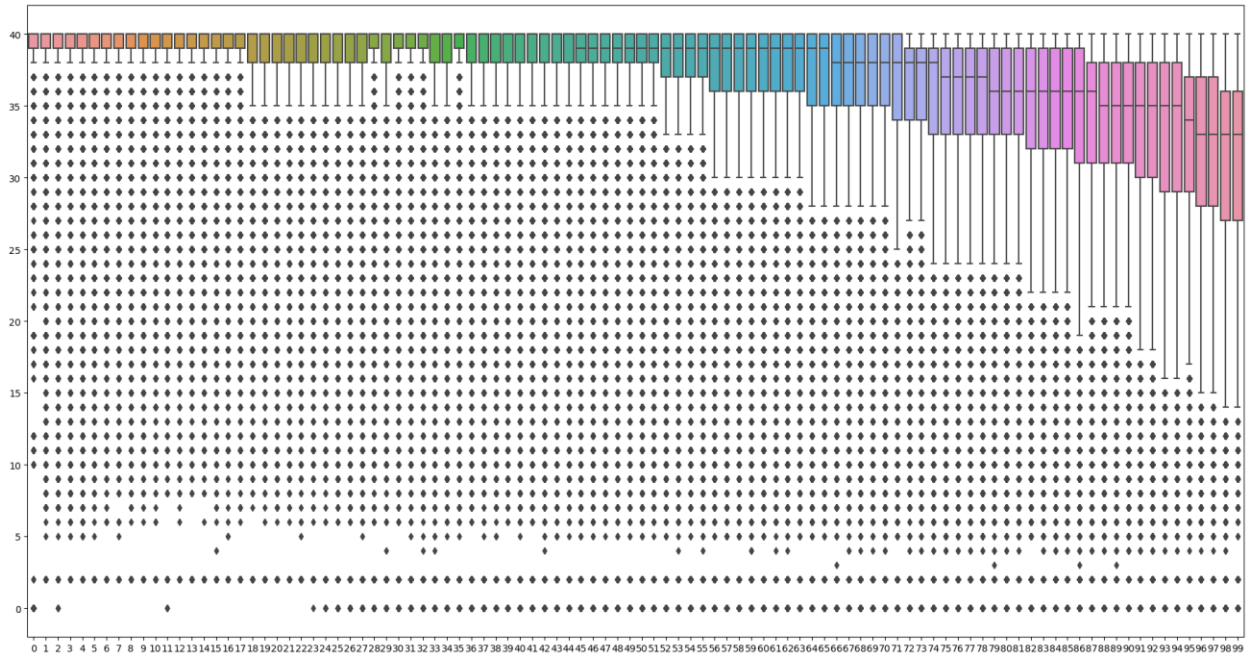
V. Plot the quality of the reads in the fastq file using a box plot.

Define a function to gives a list of ascii scores and compute the phred score of each entry in it. Extract first 100000 reads and compute phred score of each position in it. The 2D array has 100000 rows and 100 columns that each entry i, j represents quality of j'th position in i'th read. Then plot the box plot of this matrix which shows error of each position in all 100000 reads.

```
# Part A Q2 V

import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
def compute_phread_score(quality_string):
    phred_scores = []
    for char in quality_string:
        phred_score = ord(char) - 33
        phred_scores.append(phred_score)
    return phred_scores

scores = np.zeros((100000, 100))
for i, read in enumerate(all_reads[:100000]):
    for j,q in enumerate(read[2]):
        score = compute_phread_score(q)
        scores[i][j]= score[0]

scores = np.transpose(scores)
data = scores.reshape(-1)
# Create a 1D array of column labels
labels = np.repeat(np.arange(100), 100000)
plt.figure(figsize = (23, 12))
# Create a boxplot of the data with labels
sns.boxplot(x=labels, y=data)
plt.show()
```

The result is as below:

It shows that how much we get closer to the end of reads their error becomes larger.



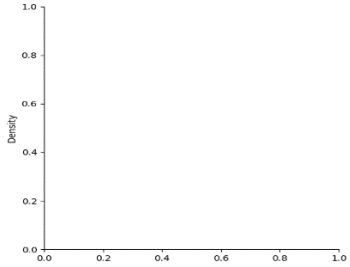
VI. Show the distribution of read lengths using a density plot.

Append all read's lengths to a list and then plot the density plot of the read's lengths.

```
# Part A Q2 VI
import seaborn as sns
from scipy.stats import gaussian_kde
import numpy as np
lengths = [len(read[1]) for read in all_reads]
density = gaussian_kde(lengths)
density._compute_covariance()
xs = np.linspace(0, read_count)
sns.displot(x=lengths, kind="kde")
```

The result is as follow:

Because data values are all the same, then there is no variation or spread in the data. This means that there is only one bin that contains all the data points, and the height of that bin is equal to the total probability of 1. Therefore, the density plot will be a flat line at the bottom of the plot, with a single spike at the value of 100. This spike will be very narrow and hard to see, because the width of the bin is determined by the range of the data values.



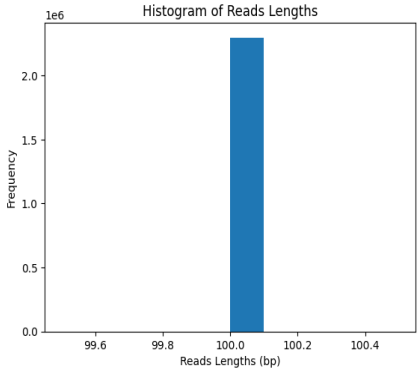
Using histogram to show it:

Plot the length of reads again with histogram.

```
reads_lengths = [len(read[1]) for read in all_reads]
# import the matplotlib library
import matplotlib.pyplot as plt

# plot a histogram with 10 bins
plt.hist(reads_lengths, bins=10)

# add labels and title
plt.xlabel("Reads Lengths (bp)")
plt.ylabel("Frequency")
plt.title("Histogram of Reads Lengths")

# show the plot
plt.show()
```

The result is:

Because there is no variation or spread in the data, there is only one bin that contains all the data points, and the height of bin is equal to the total number of data points. Therefore, the histogram is a single bar at the value of 100, with a width equal to the bin size.



Q3

Perform quality control of the reads using FastQC and interpret the results.

Installing the fastqc

```
narges@narges-VirtualBox:~$ sudo apt-get -y install fastqc
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java default-jre default-jre-headless fonts-dejavu-extra
  java-common libaec0 libapache-pom-java libargs4j-java libatk-wrapper-java
  libatk-wrapper-java-jni libcommons-compress-java libcommons-io-java
  libcommons-jexl2-java libcommons-lang3-java libcommons-logging-java
  libcommons-math3-java libcommons-parent-java libfindbin-libs-perl
```

Performing fastqc on file.

```
narges@narges-VirtualBox:~/SRR8185316/fastq$ fastqc -o /home/narges/SRR8185316/ SRR8185316_pass.fastq
Started analysis of SRR8185316_pass.fastq
Approx 5% complete for SRR8185316_pass.fastq
Approx 10% complete for SRR8185316_pass.fastq
Approx 15% complete for SRR8185316_pass.fastq
Approx 20% complete for SRR8185316_pass.fastq
Approx 25% complete for SRR8185316_pass.fastq
Approx 30% complete for SRR8185316_pass.fastq
Approx 35% complete for SRR8185316_pass.fastq
Approx 40% complete for SRR8185316_pass.fastq
Approx 45% complete for SRR8185316_pass.fastq
Approx 50% complete for SRR8185316_pass.fastq
Approx 55% complete for SRR8185316_pass.fastq
Approx 60% complete for SRR8185316_pass.fastq
Approx 65% complete for SRR8185316_pass.fastq
Approx 70% complete for SRR8185316_pass.fastq
Approx 75% complete for SRR8185316_pass.fastq
Approx 80% complete for SRR8185316_pass.fastq
Approx 85% complete for SRR8185316_pass.fastq
Approx 90% complete for SRR8185316_pass.fastq
Approx 95% complete for SRR8185316_pass.fastq
Analysis complete for SRR8185316_pass.fastq
```

The result file (SRR8185316_pass_fastqc) is attached.

Summary of each part of the result file is as follows:

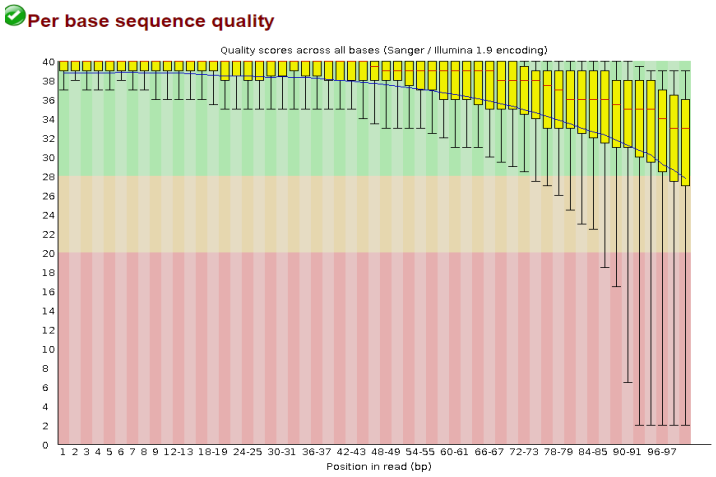
Summary table: all od the files are green(pass) and one file has warning

basic statistics table shows some general information about the FASTQ file, such as the file name, the file type, the total number of sequences, the sequence length, the GC content, etc.

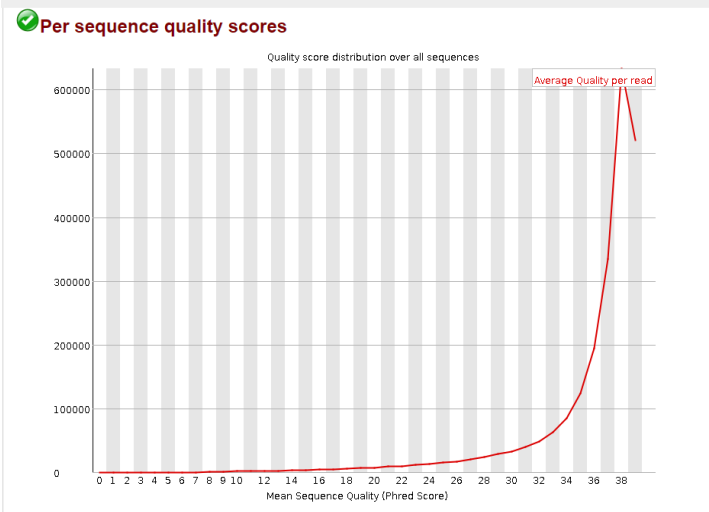
Basic Statistics

Measure	Value
Filename	SRR8185316_pass.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	2297280
Sequences flagged as poor quality	0
Sequence length	100
%GC	49

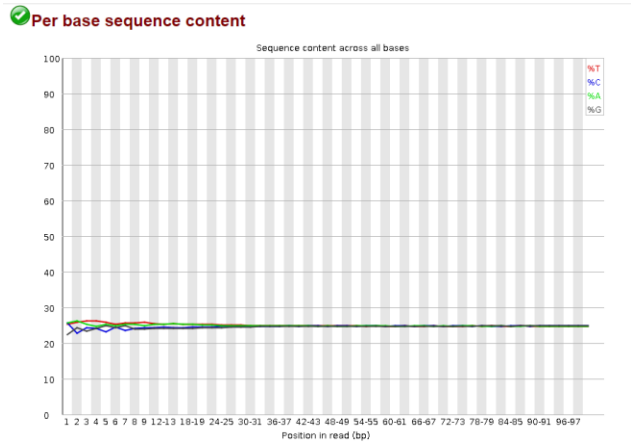
Per base sequence quality: This plot shows the quality scores across all bases at each position in the FASTQ file. As we mentioned before, it has a good result(green) on most of the bases.



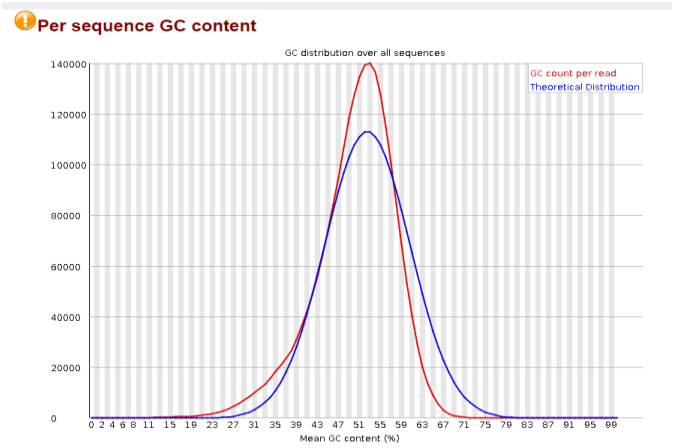
Per sequence quality scores: This plot shows the distribution of the average quality score over full length reads.



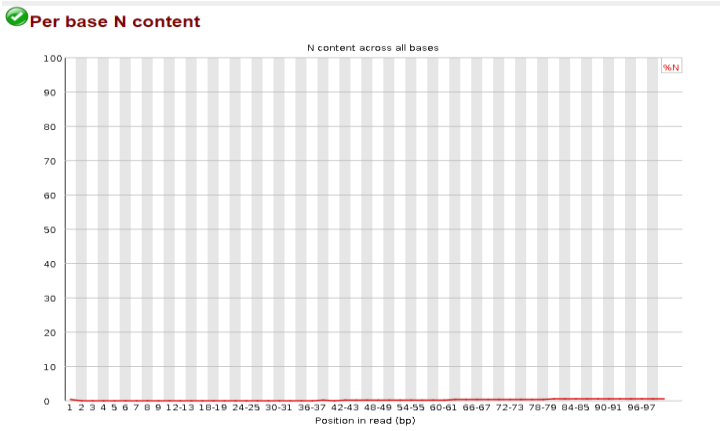
Per base sequence content: This plot shows the proportion of each base (A, T, G, C) at each position in the FASTQ file. It is almost the same for all bases.



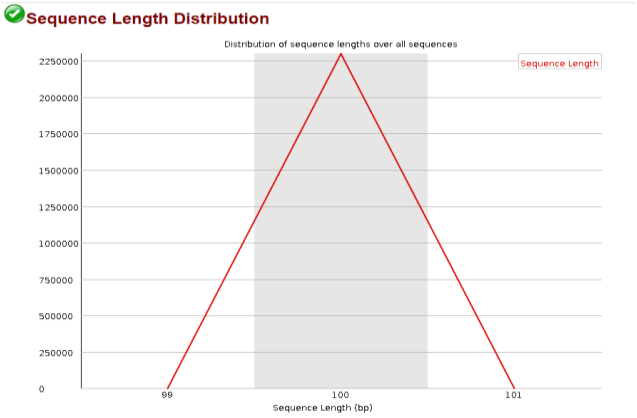
Per sequence GC content: This plot shows the distribution of the GC content over full length reads. The y-axis shows the number of reads, and the x-axis shows the GC content. The plot shows a blue curve that represents the actual distribution, and a red curve that represents the theoretical distribution expected for a random sequence with the same GC content.



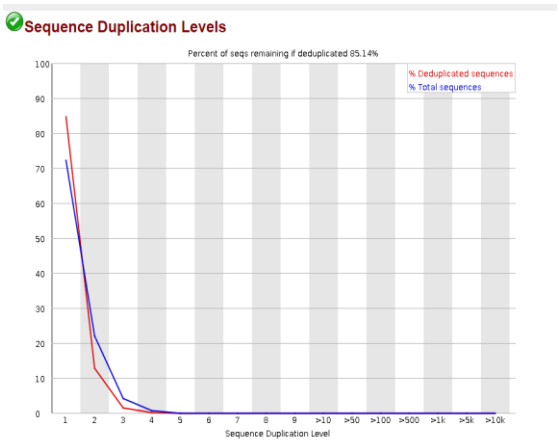
Per base N content: This plot shows the percentage of bases at each position in the FASTQ file that have an N (unknown) base call. The y-axis shows the percentage of N bases, and the x-axis shows the position in the read.



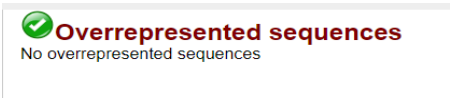
Sequence length distribution: This plot shows the distribution of the lengths of the reads in the FASTQ file. The y-axis shows the number of reads, and the x-axis shows the length of the reads. All reads has the same size(100).



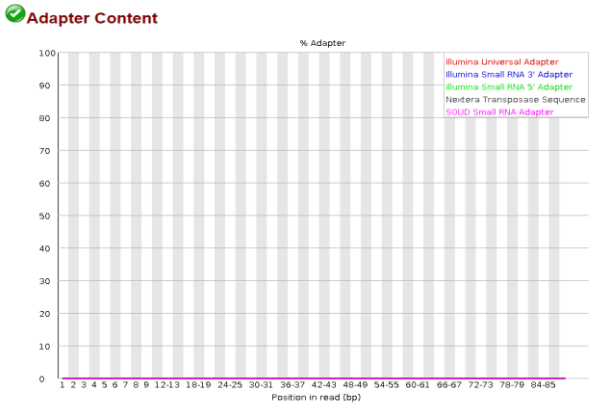
Sequence duplication levels: This plot shows the relative level of duplication for each sequence in the FASTQ file. The y-axis shows the percentage of reads, and the x-axis shows the duplication level. The plot shows a blue curve that represents the actual distribution, and a red curve that represents the theoretical distribution expected for a random sequence with no duplication. This value gets lower and lower.



Overrepresented sequences: This table shows the sequences that are present in the FASTQ file with a greater frequency than expected by chance. It has no overrepresented sequence.



Adapter content: This plot shows the cumulative percentage of reads that have an adapter sequence at each position in the FASTQ file. The y-axis shows the percentage of reads, and the x-axis shows the position in the read. The plot shows multiple colored lines that represent different adapter sequences. For this data it is pink(solid small RNA adapter).



Part B

De novo genome assembly

1. *Run SPAdes to generate draft genome assemblies from short reads.*
   Downloadinf SPAdes:

```
[1] !wget https://github.com/steventango/colab-spades/releases/download/v3.15.5/SPAdes-3.15.5-Colab.tar.gz
    !tar -xzf SPAdes-3.15.5-Colab.tar.gz

--2023-12-24 09:50:25--  https://github.com/steventango/colab-spades/releases/download/v3.15.5/SPAdes-3.15.5-Colab.tar.gz
Resolving github.com (github.com)... 140.82.113.3
Connecting to github.com (github.com)|140.82.113.3|:443... connected.
```

Run SPAdes on fastq file:

```
!python ./bin/spades.py -s /content/SRR8185316_1.fastq -o spades_output3

  0:01:30.542    60M / 1134M INFO    PEResolver               (pe_resolver.cpp          :  63)  Sorting paths
  0:01:30.543    60M / 1134M INFO    PEResolver               (pe_resolver.cpp          :  70)  Marking overlaps
  0:01:30.543    60M / 1134M INFO    OverlapRemover           (overlap_remover.hpp      : 116)  Marking start/end overlaps
  0:01:30.544    60M / 1134M INFO    OverlapRemover           (overlap_remover.hpp      : 119)  Marking remaining overlaps
  0:01:30.548    60M / 1134M INFO    PEResolver               (pe_resolver.cpp          :  73)  Splitting paths
  0:01:30.549    61M / 1134M INFO    PEResolver               (pe_resolver.cpp          :  78)  Deduplicating paths
  0:01:30.550    60M / 1134M INFO    PEResolver               (pe_resolver.cpp          :  80)  Overlaps removed
  0:01:30.551    60M / 1134M INFO    General                  (launcher.cpp             : 273)  Paths finalized
  0:01:30.551    60M / 1134M INFO    General                  (launcher.cpp             : 454)  Closing gaps in paths
  0:01:30.553    60M / 1134M INFO    General                  (launcher.cpp             : 484)  Gap closing completed
  0:01:30.553    60M / 1134M INFO    General                  (launcher.cpp             : 302)  Traversing tandem repeats
  0:01:30.555    60M / 1134M INFO    General                  (launcher.cpp             : 312)  Traversed 4 loops
  0:01:30.556    60M / 1134M INFO    General                  (launcher.cpp             : 250)  Finalizing paths
  0:01:30.556    60M / 1134M INFO    General                  (launcher.cpp             : 252)  Deduplicating paths
  0:01:30.557    60M / 1134M INFO    General                  (launcher.cpp             : 256)  Paths deduplicated
  0:01:30.557    60M / 1134M INFO    PEResolver               (pe_resolver.cpp          :  60)  Removing overlaps
  0:01:30.557    60M / 1134M INFO    PEResolver               (pe_resolver.cpp          :  63)  Sorting paths
  0:01:30.557    60M / 1134M INFO    PEResolver               (pe_resolver.cpp          :  70)  Marking overlaps
  0:01:30.557    60M / 1134M INFO    OverlapRemover           (overlap_remover.hpp      : 116)  Marking start/end overlaps
  0:01:30.557    60M / 1134M INFO    OverlapRemover           (overlap_remover.hpp      : 119)  Marking remaining overlaps
  0:01:30.558    60M / 1134M INFO    PEResolver               (pe_resolver.cpp          :  73)  Splitting paths
  0:01:30.558    60M / 1134M INFO    PEResolver               (pe_resolver.cpp          :  78)  Deduplicating paths
  0:01:30.558    60M / 1134M INFO    PEResolver               (pe_resolver.cpp          :  80)  Overlaps removed
  0:01:30.558    60M / 1134M INFO    General                  (launcher.cpp             : 273)  Paths finalized
  0:01:30.558    60M / 1134M INFO    General                  (launcher.cpp             : 664)  ExSPAnder repeat resolving tool finished
  0:01:30.562    51M / 1134M INFO    StageManager             (stage.cpp                : 185)  STAGE == Contig Output (id: contig_output)
  0:01:30.562    51M / 1134M INFO    General                  (contig_output.hpp        :  16)  Outputting contigs to /content/spades_output3/K55/before_rr.fasta
  0:01:30.625    51M / 1134M INFO    General                  (contig_output_stage.cpp  : 151)  Writing GFA graph to /content/spades_output3/K55/assembly_graph_with_scaffolds.gfa
  0:01:30.653    51M / 1134M INFO    General                  (contig_output_stage.cpp  : 165)  Outputting FastG graph to /content/spades_output3/K55/assembly_graph.fastg
  0:01:30.747    51M / 1134M INFO    General                  (contig_output_stage.cpp  : 196)  Breaking scaffolds
  0:01:30.766    56M / 1134M INFO    General                  (contig_output_stage.cpp  :  98)  Outputting contigs to /content/spades_output3/K55/final_contigs.fasta
  0:01:30.794    56M / 1134M INFO    General                  (contig_output_stage.cpp  : 104)  Outputting FastG paths to /content/spades_output3/K55/final_contigs.paths
  0:01:30.814    55M / 1134M INFO    General                  (contig_output_stage.cpp  :  98)  Outputting contigs to /content/spades_output3/K55/scaffolds.fasta
  0:01:30.841    55M / 1134M INFO    General                  (contig_output_stage.cpp  : 104)  Outputting FastG paths to /content/spades_output3/K55/scaffolds.paths
  0:01:30.843    55M / 1134M INFO    General                  (contig_output_stage.cpp  : 111)  Populating GFA with scaffold paths
  0:01:30.843    51M / 1134M INFO    General                  (pipeline.cpp             : 287)  SPAdes finished
  0:01:30.847     1M / 1134M INFO    General                  (main.cpp                 : 136)  Assembling time: 0 hours 1 minutes 30 seconds
```

The output is a folder that most files of it with low size are attached.

output folder contains the following files and subfolders:

spades.log: records the SPAdes command line, the version, the parameters, the progress, and the results of the assembly.

Params.txt: shows the SPAdes parameters and the input files used for the assembly.

Dataset.info: shows the information about the input files, such as the library type, the orientation, the insert size, etc.

Corrected: subfolder that contains the FASTQ files corrected by BayesHammer or IonHammer, the error correction tools for Illumina or IonTorrent reads.

Assembly_graph.fastg: a FASTG file that contains the assembly graph constructed by SPAdes, where each node represents a contig and each edge represents an overlap between contigs.

Contigs.fasta: contains the resulting contigs generated by SPAdes, where each sequence header shows the contig ID, the length, and the coverage.

Scaffolds.fasta: contains the resulting scaffolds generated by SPAdes, where each sequence header shows the scaffold ID, the length, the coverage, and the number of contigs. The gaps between contigs are represented by Ns.

before_rr.fasta: contains the intermediate scaffolds before the repeat resolution step, which removes the redundant contigs and resolves the repeats in the assembly graph.

Scaffolds.path: shows the paths of contigs that form each scaffold, where each line shows the scaffold ID and the contig IDs separated by commas.

Contigs.path: shows the paths of kmers that form each contig, where each line shows the contig ID and the kmer IDs separated by commas.

---

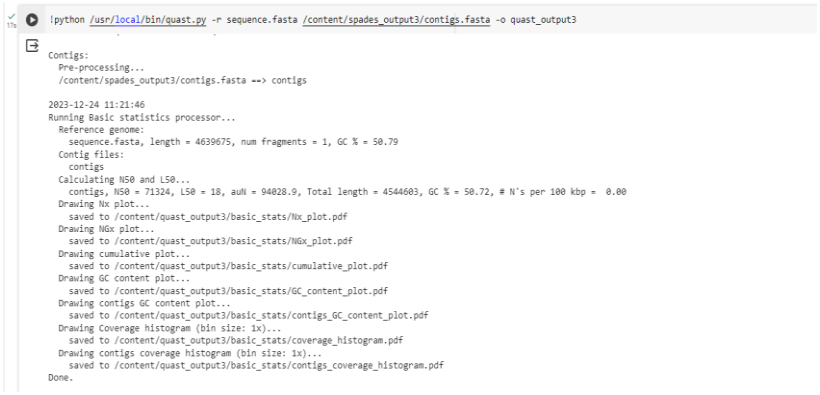
2. Assess the quality of the draft genome assembly using Quast and compare it to the reference genome.

Installing quast:

```
[8] !pip install quast

Collecting quast
  Downloading quast-5.2.0.tar.gz (34.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 34.2/34.2 MB 18.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from quast) (1.3.2)
Collecting simplejson (from quast)
  Downloading simplejson-3.19.2-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (137 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 137.9/137.9 kB 15.7 MB/s eta 0:00:00
Building wheels for collected packages: quast
  Building wheel for quast (setup.py) ... done
  Created wheel for quast: filename=quast-5.2.0-py3-none-any.whl size=31206445 sha256=e2a9153f22aee4381981247274 3a3d6c021a52a6a0f705c144f370920db36a6d
  Stored in directory: /root/.cache/pip/wheels/89/09/d5/511e4a9adc7cf195a54f39b139a2c61e348f32b3836ca77e2c
Successfully built quast
Installing collected packages: simplejson, quast
Successfully installed quast-5.2.0 simplejson-3.19.2
```

Run quast on contigs.fasta which is the assembly result from the previous question.

```
[38] !python /usr/local/bin/quast.py  /content/spades_output3/contigs.fasta -o quast_output3

Started: 2023-12-24 11:21:25

Logging to /content/quast_output3/quast.log
NOTICE: Maximum number of threads is set to 1 (use --threads option to set it manually)

CWD: /content
Main parameters:
  MODE: default, threads: 1, min contig length: 500, min alignment length: 65, min alignment IDY: 95.0, \
  ambiguity: one, min local misassembly length: 200, min extensive misassembly length: 1000

Contigs:
  Pre-processing...
  /content/spades_output3/contigs.fasta ==> contigs

2023-12-24 11:21:26
Running Basic statistics processor...
  Contig files:
    contigs
  Calculating N50 and L50...
    contigs, N50 = 71324, L50 = 18, auN = 94028.9, Total length = 4544603, GC % = 50.72, # N's per 100 kbp =  0.00
  Drawing Nx plot...
    saved to /content/quast_output3/basic_stats/Nx_plot.pdf
  Drawing cumulative plot...
    saved to /content/quast_output3/basic_stats/cumulative_plot.pdf
  Drawing GC content plot...
    saved to /content/quast_output3/basic_stats/GC_content_plot.pdf
  Drawing contigs GC content plot...
    saved to /content/quast_output3/basic_stats/contigs_GC_content_plot.pdf
  Drawing Coverage histogram (bin size: 1x)...
    saved to /content/quast_output3/basic_stats/coverage_histogram.pdf
  Drawing contigs coverage histogram (bin size: 1x)...
    saved to /content/quast_output3/basic_stats/contigs_coverage_histogram.pdf
Done.
```



The result is a quast folder that contains many files and attached to the project.
The result in report.html is as follows:

- contigs: The number of contigs in the assembly.
- Largest contig: The length of the longest contig in the assembly.
- Total length: The sum of the lengths of all contigs in the assembly.
- GC (%): The percentage of GC bases in the assembly.
- N50: A measure of the contig size that is defined as the length of the shortest contig in the set that contains the fewest (largest) contigs whose combined length represents at least 50% of the assembly length.
- L50: The number of contigs in the N50 set.
- N's per 100 kbp: The number of N (unknown) bases per 100,000 bases in the assembly.
- misassemblies: The number of contigs that contain a misassembly, which is a structural error in the assembly, such as a relocations, translocations, inversions, or interspersed repeats.
- misassembled contigs: The number of contigs that contain at least one misassembly.
- Misassembled contigs length: The sum of the lengths of the misassembled contigs.
- local misassemblies: The number of local misassemblies, which are misassemblies that do not affect the global structure of the assembly, but only the order and orientation of the contigs within a scaffold. They are usually caused by incorrect gap size estimation or repetitive regions.
- unaligned contigs: The number of contigs that are not aligned to the reference genome.
- Unaligned length: The sum of the lengths of the unaligned contigs.
- mismatches per 100 kbp: The number of mismatches (substitutions) per 100,000 aligned bases in the assembly.
- indels per 100 kbp: The number of indels (insertions or deletions) per 100,000 aligned bases in the assembly.
- genes: The number of genes (complete or partial) in the assembly that are annotated in the reference genome.
- operons: The number of operons (complete or partial) in the assembly that are annotated in the reference genome.
- **Genome fraction:** describe the proportion of a genome that is covered by a set of aligned sequences which is 97.937 and shows a good match of assembled sequences to the reference genome.

24 December 2023, Sunday, 11:21:57

View in Icarus contig browser

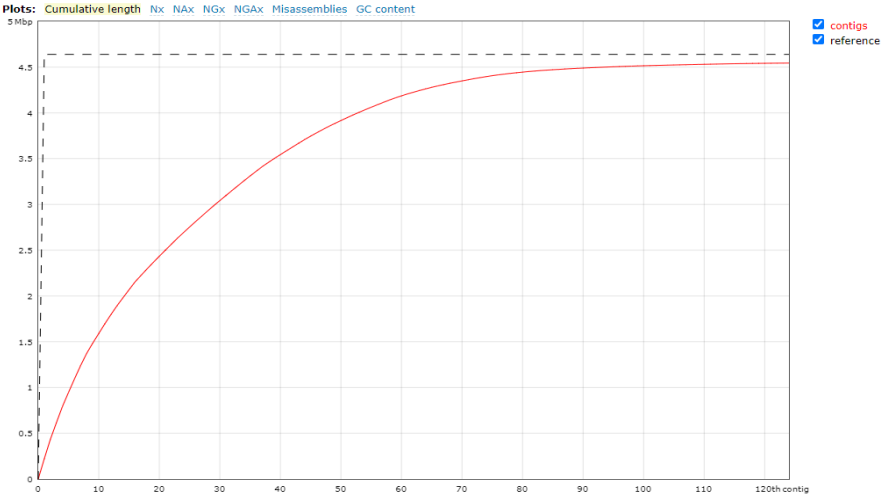
All statistics are based on contigs of size >= 500 bp, unless otherwise noted (e.g., "# contigs (>= 0 bp)" and "Total length (>= 0 bp)" include all contigs).

Aligned to "sequence" | 4 639 675 bp | 1 fragment | 50.79 % G+C

Genome statistics	contigs
Genome fraction (%)	97.937
Duplication ratio	1
Largest alignment	221 546
Total aligned length	4 543 859
NGA50	69 051
LGA50	19
Misassemblies	
# misassemblies	1
Misassembled contigs length	182 275
Mismatches	
# mismatches per 100 kbp	1.34
# indels per 100 kbp	0.22
# N's per 100 kbp	0
Statistics without reference	
# contigs	124
Largest contig	221 601
Total length	4 544 603
Total length (>= 1000 bp)	4 540 236
Total length (>= 10000 bp)	4 406 717
Total length (>= 50000 bp)	3 413 284

Extended report

Plots: Cumulative length Nx NAx NGx NGAx Misassemblies GC content



Contigs are ordered from largest (contig #1) to smallest.

Part C

1. *Map the Illumina short-read data to the reference genome using BWA.*

   Installing bwa:

```
!sudo apt install bwa
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  samtools
The following NEW packages will be installed:
  bwa
```

Indexing reference genome:

```
!bwa index ./sequence.fasta
```

```
[bwa_index] Pack FASTA... 0.04 sec
[bwa_index] Construct BWT for the packed sequence...
[bwa_index] 2.59 seconds elapse.
[bwa_index] Update BWT... 0.04 sec
[bwa_index] Pack forward-only FASTA... 0.04 sec
[bwa_index] Construct SA from BWT and Occ... 1.30 sec
[main] Version: 0.7.17-r1188
[main] CMD: bwa index ./sequence.fasta
[main] Real time: 7.880 sec; CPU: 4.011 sec
```

Mapping short reads to reference genome and put the result in align_result.sam:

```
!bwa mem -t 4 ./sequence.fasta /content/SRR8185316_1.fastq > align_result.sam

[M::bwa_idx_load_from_disk] read 0 ALT contigs
[M::process] read 400000 sequences (40000000 bp)...
[M::process] read 400000 sequences (40000000 bp)...
[M::mem_process_seqs] Processed 400000 reads in 26.528 CPU sec, 16.643 real sec
[M::process] read 400000 sequences (40000000 bp)...
[M::mem_process_seqs] Processed 400000 reads in 25.630 CPU sec, 16.146 real sec
[M::process] read 400000 sequences (40000000 bp)...
[M::mem_process_seqs] Processed 400000 reads in 25.485 CPU sec, 16.052 real sec
[M::process] read 400000 sequences (40000000 bp)...
[M::mem_process_seqs] Processed 400000 reads in 25.700 CPU sec, 16.114 real sec
[M::process] read 297280 sequences (29728000 bp)...
[M::mem_process_seqs] Processed 400000 reads in 25.542 CPU sec, 16.034 real sec
[M::mem_process_seqs] Processed 297280 reads in 19.649 CPU sec, 16.563 real sec
[main] Version: 0.7.17-r1188
[main] CMD: bwa mem -t 4 ./sequence.fasta /content/SRR8185316_1.fastq
[main] Real time: 99.127 sec; CPU: 149.479 sec
```

2. *Print the head of the obtained SAM file from previous question.*
   Installing samtools:

```
sudo apt install samtools

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libhts3 libhtscodecs2
Suggested packages:
  cwltool
The following NEW packages will be installed:
  libhts3 libhtscodecs2 samtools
0 upgraded, 3 newly installed, 0 to remove and 24 not upgraded.
Need to get 963 kB of archives.
After this operation, 2,270 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libhtscodecs2 amd64 1.1.1-3 [53.2 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libhts3 amd64 1.13+ds-2build1 [390 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/universe amd64 samtools amd64 1.13-4 [520 kB]
Fetched 963 kB in 0s (2,535 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78, <> line 3.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
```

We showed the first 3 line of sam file. 3 first lines are header:

```
[17] !samtools view -h -S /content/align_result.sam | head -n 3

@SQ     SN:U00096.2     LN:4639675
@PG     ID:bwa  PN:bwa  VN:0.7.17-r1188 CL:bwa mem -t 4 ./sequence.fasta /content/SRR8185316_1.fastq
@PG     ID:samtools     PN:samtools     PP:bwa  VN:1.13 CL:samtools view -h -S /content/align_result.sam
```

The header section contains lines that start with the '@' symbol, which indicate the metadata of the alignment, such as the version, the reference sequences, the read groups, the program information, ... Each header line has a two-letter record type code followed by one or more tab-separated fields.

@SQ SN:U00096.2 LN:4639675: first line indicates a reference sequence (@SQ) with the name (SN) of U00096.2 and the length (LN) of 4639675 bp. This line provides information about the reference genome that the reads are aligned to.

@PG ID:bwa PN:bwa VN:0.7.17-r1188 CL:bwa mem -t 4 ./sequence.fasta /content/SRR8185316_1.fastq: second line indicates a program (@PG) with the ID (ID) of bwa, the name (PN) of bwa, the version (VN) of 0.7.17-r1188, and the command line (CL) of bwa mem -t 4 ./sequence.fasta /content/SRR8185316_1.fastq. This line provides information about the program that was used to align the reads to the reference genome, and the parameters and input files that were used.

@PG ID:samtools PN:samtools PP:bwa VN:1.13 CL:samtools view -h -S /content/part3/align_result.sam: 3'rd line indicates another program (@PG) with the ID (ID) of samtools, the name (PN) of samtools, the previous program ID (PP) of bwa, the version (VN) of 1.13, and the command line (CL) of samtools view -h -S /content/part3/align_result.sam. This line provides information about the program that was used to view or manipulate the SAM file, and the parameters and input files that were used.

3. Convert the SAM file to an indexed BAM file. Hint: use samtools view, samtools sort, samtools index.

```
[50] !samtools view -S -b align_result.sam > align_result.bam
```

```
[51] !samtools sort align_result.bam -o sorted_align.bam
```

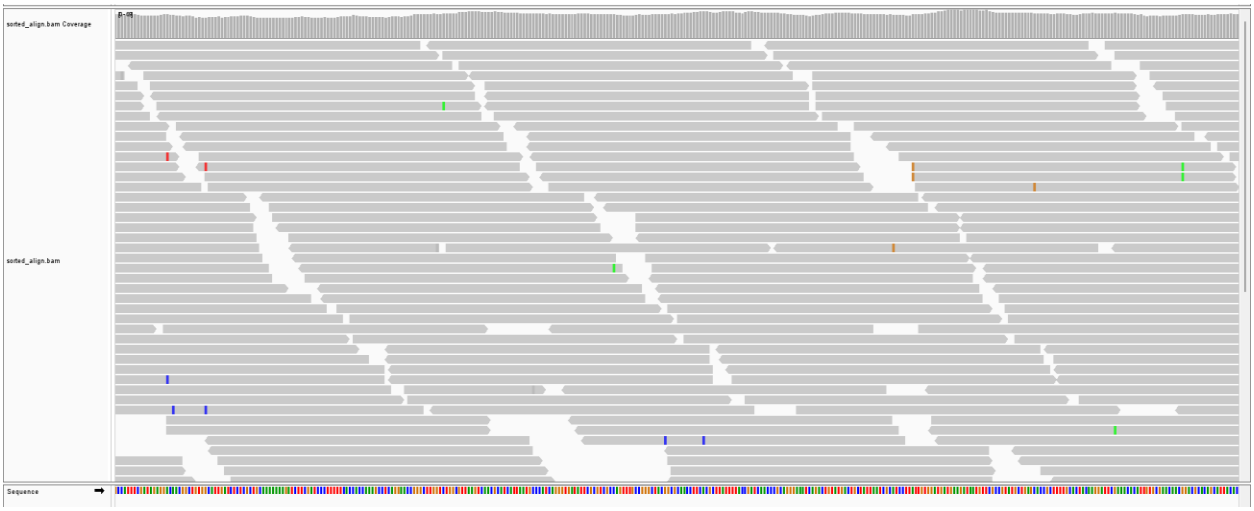
```
[52] !samtools index sorted_align.bam
```

```
[53] !zip -r /content/part3_outputs.zip /content/part3
```

```
  adding: content/part3/ (stored 0%)
  adding: content/part3/sorted_align.bam (deflated 0%)
  adding: content/part3/align_result.sam (deflated 70%)
  adding: content/part3/align_result.bam (deflated 0%)
```

Files are attached.

4-Use the Integrative Genomics Viewer (IGV) to visualize the mapped reads in a 200-b genomic region of your choice.



The gray and white parts in IGV view represent the read alignments to the reference genome. The gray parts indicate that the read matches the reference base, while the white parts indicate that the read has a mismatch or a gap. The color of the mismatched base corresponds to the nucleotide. The matching phase has done properly because mismathches are small.

5. Determine the percentage of short reads that are mapped to the reference genome.

The output result shows the total number of reads and the number of mapped reads in the first and fifth rows, respectively.

```
!samtools flagstat sorted_align.bam

2297765 + 0 in total (QC-passed reads + QC-failed reads)
2297280 + 0 primary
0 + 0 secondary
485 + 0 supplementary
0 + 0 duplicates
0 + 0 primary duplicates
2294728 + 0 mapped (99.87% : N/A)
2294243 + 0 primary mapped (99.87% : N/A)
0 + 0 paired in sequencing
0 + 0 read1
0 + 0 read2
0 + 0 properly paired (N/A : N/A)
0 + 0 with itself and mate mapped
0 + 0 singletons (N/A : N/A)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

6. Get the read depth for the sorted BAM file at all positions of the reference genome and report the mean of all reads.

I've used these 2 commands. The first one finds the depth of each read and the second one finds the mean of all reads. Output files are attached.

```
[26] !samtools depth -a sorted_align.bam > reads_depth.bam

!samtools depth -a sorted_align.bam | awk '{sum+=$3} END {print sum/NR}'

49.1858
```

7. Make yourself familiar with the CIGAR format.

Short summary of CIGAR: The CIGAR format is a way of representing the alignment of a query sequence to a reference sequence. It has a series of numbers and letters. Numbers represent the length of operation and the letters represent the operation. The operations are:

- M: match or mismatch
- I: insertion to the query
- D: deletion from the query
- N: skipped region from the reference
- S: soft clipping (clipped sequences present in SEQ)
- H: hard clipping (clipped sequences NOT present in SEQ)
- P: padding (silent deletion from padded reference)
- =: sequence match
- X: sequence mismatch

How do you interpret the following expressions?

29S21=1X25= : This means that the first 29 bases of the query sequence are soft clipped (not aligned), then there are 21 matches, 1 mismatch, and 25 matches.

20M2I1M1D10M : This means that there are 20 matches, 2 insertions to the query, 1 match, 1 deletion from the query, and 10 matches.

5M10N25M : This means that there are 5 matches, 10 skipped regions from the reference, and 25 matches. This is typical of a spliced alignment, where the query sequence spans an intron.