Bioinformatics Project: Quality Control & Analysis of Sequencing Data

Date: October 26, 2023 **Platform:** Illumina HiSeq 2000 **Library Type:** Paired-end **Ekrem Cemil Kilic**

A. Libraries Instalation

The following Python libraries are required to reproduce the analysis below:

- **Biopython:** For FASTQ parsing.
- **Matplotlib:** For plotting quality graphs.
- **NumPy:** For statistical calculations.

```
from Bio import SeqIO
import gzip
import matplotlib.pyplot as plt
import numpy as np
import os
```

B. Downloading and Inspecting Files

1. Sequencing Platform

Question: What sequencing platform was used to do the sequencing run?

Answer: ILLUMINA

2. Instrument Model

Question: What is the instrument model used?

Answer: Illumina HiSeq 2000

3. Library Type

Question: What is the library type used for the sequencing? (Paired-end, Single-end, or Mate-pair)

Answer: Paired-end

4. Molecule Type

Question: What is the type of molecules that have been sequenced?

Answer: DNA

5. File Identification

Question: At the bottom of the page, you can see two fastq file names.

- `SRR800768_1_sub.fastq`
- `SRR800768_2_sub.fastq`

6. File Extension

Question: What is the extension of these files? What does it indicate?

Answer: `fastq.gz`. This indicates it is a FASTQ file (storing biological sequences and quality scores) compressed using gzip.

7. File Differences

Question: Are these files expected to be different in their contents? Justify your answer.

Answer: Yes. File 1 contains the `Forward` reads and File 2 contains the `Reverse` reads. While they come from the same DNA fragment, the actual nucleotide strings differ because they are sequenced from opposite ends.

9. File Size

Question: What are the sizes of both files?

Answer: Approximately 22.905 KB.

C. Analyzing the files

1. How many reads do we have in each of the two files? Verify with a Python script that they are correctly paired.

Answer:

- `SRR800768_1_sub.fastq`: 100,000 reads
- `SRR800768_2_sub.fastq`: 100,000 reads

Python Script for Verification:

```
print("Counting reads")

def count_reads(filename):
    count = 0

    for record in SeqIO.parse(filename, "fastq"):
        count += 1
    return count

count1 = count_reads(file1)
print(f"{file1} has {count1} reads.")

count2 = count_reads(file2)
print(f"{file2} has {count2} reads.")
```

```
print("Verifying pairs")
records1 = SeqIO.parse(file1, "fastq")
records2 = SeqIO.parse(file2, "fastq")

for r1, r2 in zip(records1, records2):
    if r1.id.split('.')[0] != r2.id.split('.')[0]:
        print("Error: Mismatch!")
        break
else:
    print("match perfectly.")
```

2. What is the read length?

Answer:

- Read length is: 101 bp

Python Script for Verification:

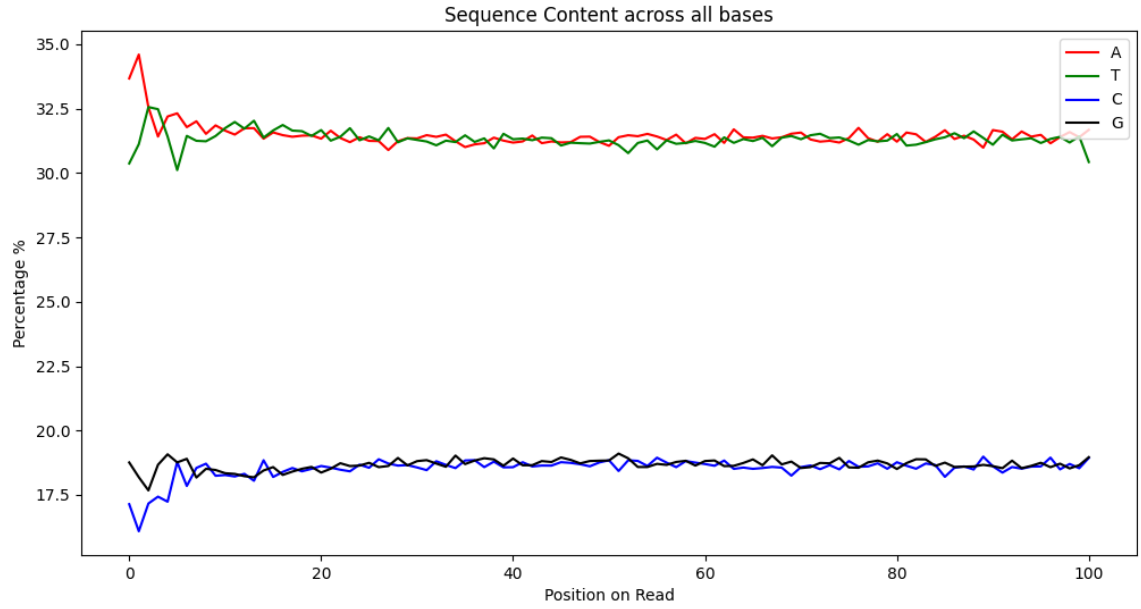
```
filename = "SRR800768_1_sub.fastq"
first_record = next(SeqIO.parse(filename, "fastq"))

length = len(first_record.seq)
print(f"Read length is: {length} bp")
```

3. Plot the reads per base content for each of the fastq files: the x axis is the position on the read (0 to read length - 1) and on the y axis the percentages of each base (A in red, T in green, C in blue and G in black).

Python Script for the plot

```
base_counts = { 'A': np.zeros(read_length),
                'T': np.zeros(read_length),
                'C': np.zeros(read_length),
                'G': np.zeros(read_length) }
for record in records:
    seq = str(record.seq)
    for i, base in enumerate(seq):
        if base in base_counts:
            base_counts[base][i] += 1
plt.figure(figsize=(12, 6))
positions = np.arange(read_length)

colors = {'A': 'red', 'T': 'green', 'C': 'blue', 'G': 'black'}

for base, color in colors.items():
    percentages = (base_counts[base] / num_reads) * 100
    plt.plot(positions, percentages, color=color, label=base)
```

```
plt.xlabel("Position on Read")
plt.ylabel("Percentage %")
plt.title("Sequence Content across all bases")
plt.legend(loc='upper right')
plt.show()
```



4. What do you notice regarding the relative quantity of the different bases? Is it an expected result? Justify your answer.

Answer:

- A and T line basically the same we can also G and C track each other too. A and T bases are significantly more than G and C.This is expected Chargaf's Rule, which is that in double-stranded DNA, the amount of Adenine equals Thymine and Guanine equals Cytosine. Since the sequencing library is double-stranded, we expect these percentages to be mirrored.

5. Estimate the GC content of the Saccharomyces cerevisiae sequenced genome and compare it with the known GC content of the species.

Answer:

Estimated GC Content: ~37.20%

Known GC Content: ~38.0%

- Our estimated value of **37.20%** is very close to the standard reference value of approximately **38%** for *Saccharomyces cerevisiae*. Small discrepancies are expected due to sequencing bias of Illumina.

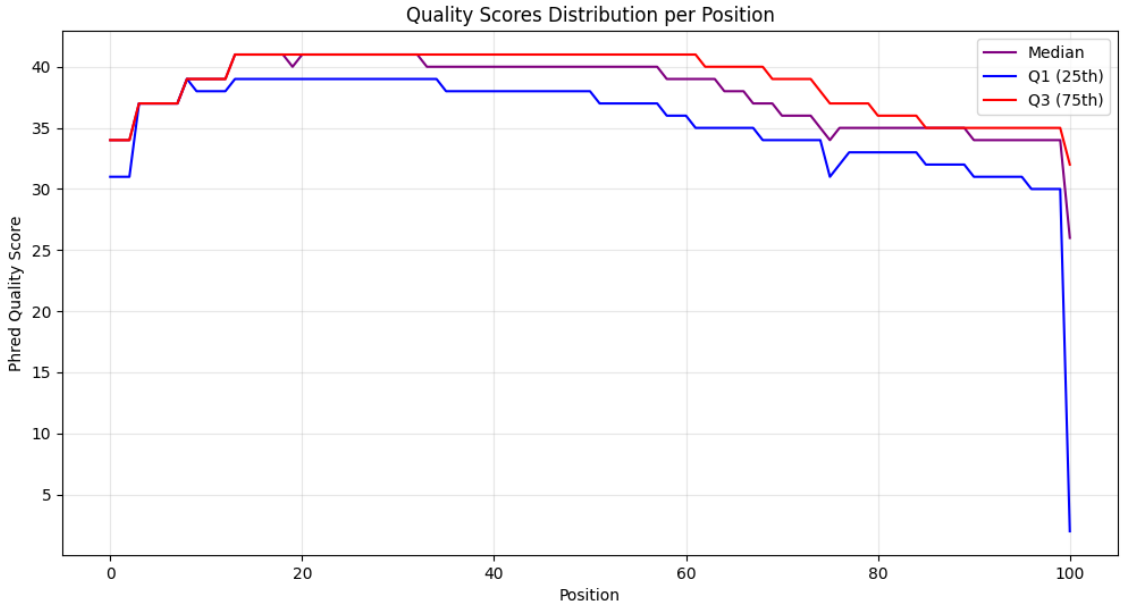
Python Script for Verification:

```
total_bases = 0
gc_bases = 0

for record in records:
    seq = record.seq
    total_bases += len(seq)
    gc_bases += seq.count('G') + seq.count('C')

gc_content = (gc_bases / total_bases) * 100
print(f"Estimated GC Content: {gc_content:.2f}%")
```

6. For each file, generate a plot illustrating the median, first quartile (Q1) and third quartile (Q3) of the Phred scores for each position in the read. The x-axis represents the position on the read, and the y-axis depicts the median (in purple), Q1 (in blue), and Q3 (in red) of the Phred scores.

Python Script for the plot

```
qual_by_pos = [[] for _ in range(read_length)]

for record in records:
    phred_scores = record.letter_annotations["phred_quality"]
    for i, score in enumerate(phred_scores):
        if i < read_length:
            qual_by_pos[i].append(score)

medians = []
q1s = []
q3s = []

for pos_scores in qual_by_pos:
    medians.append(np.median(pos_scores))
    q1s.append(np.percentile(pos_scores, 25))
    q3s.append(np.percentile(pos_scores, 75))

plt.figure(figsize=(12, 6))
plt.plot(positions, medians, color='purple', label='Median')
plt.plot(positions, q1s, color='blue', label='Q1 (25th)')
plt.plot(positions, q3s, color='red', label='Q3 (75th)')

plt.xlabel("Position")
plt.ylabel("Phred Quality Score")
plt.title("Quality Scores Distribution per Position")
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```



7. Is the quality of the bases homogeneous across the read? Propose a hypothesis explaining that.

Answer

As we can see from the graph the quality is not homegeneous across the read.

Hypothesis: When I did research on the internet I found this pattern is a charactheristic of Illumina sequencing chemistry often called signal decay or drop.This can happen because when the machine performs the cycle after cyle, some DNA moleculesbecome out of sync and some molecules lag behind. As the reads gets longer this out of sync molecules become larger. As a result the machine becomes less confident in its base calls towards the end of the read, resulting in lower Phred quality scores at the 3' end.

8. Suggest an algorithm for cleaning reads and implement it in Python. This algorithm should allow trimming low-quality bases at the ends of the reads and then keep reads whose length is above a threshold. As pairing may be disturbed by cleaning, 4 output files are expected:

Python Script for the trimming

```
f1_in = "SRR800768_1_sub.fastq"
f2_in = "SRR800768_2_sub.fastq"

out_p1 = open("paired_1.fastq", "w")
out_p2 = open("paired_2.fastq", "w")
out_u1 = open("unpaired_1.fastq", "w")
out_u2 = open("unpaired_2.fastq", "w")

min_len = 50
qual_thresh = 20

r1_iter = SeqIO.parse(f1_in, "fastq")
```

```
r2_iter = SeqIO.parse(f2_in, "fastq")

for r1, r2 in zip(r1_iter, r2_iter):

    r1_trimmed = trim_read(r1, qual_thresh)
    r2_trimmed = trim_read(r2, qual_thresh)

    r1_pass = len(r1_trimmed) >= min_len
    r2_pass = len(r2_trimmed) >= min_len

    if r1_pass and r2_pass:
        SeqIO.write(r1_trimmed, out_p1, "fastq")
        SeqIO.write(r2_trimmed, out_p2, "fastq")

    elif r1_pass and not r2_pass:
        SeqIO.write(r1_trimmed, out_u1, "fastq")

    elif not r1_pass and r2_pass:
        SeqIO.write(r2_trimmed, out_u2, "fastq")

out_p1.close()
out_p2.close()
out_u1.close()
out_u2.close()

print("Trimming is complete.")
```

8. To check the validity of the cleaning algorithm, plot the qualities of the clean paired files as done in part C.6.

Python Script for the plot

```
print("Generating Plot for Clean Data")

clean_filename = "paired_1.fastq"
clean_records = list(SeqIO.parse(clean_filename, "fastq"))

max_len = max(len(r.seq) for r in clean_records)

clean_qual_by_pos = [[] for _ in range(max_len)]

for record in clean_records:
    scores = record.letter_annotations["phred_quality"]
    for i, score in enumerate(scores):
        clean_qual_by_pos[i].append(score)

c_medians = []
c_q1s = []
c_q3s = []
c_positions = []
```

```
for i, scores in enumerate(clean_qual_by_pos):
    if len(scores) > 0:
        c_positions.append(i)
        c_medians.append(np.median(scores))
        c_q1s.append(np.percentile(scores, 25))
        c_q3s.append(np.percentile(scores, 75))

plt.figure(figsize=(12, 6))
plt.plot(c_positions, c_medians, color='purple', label='Median')
plt.plot(c_positions, c_q1s, color='blue', label='Q1 (25th)')
plt.plot(c_positions, c_q3s, color='red', label='Q3 (75th)')

plt.xlabel("Position")
plt.ylabel("Phred Quality Score")
plt.title("Quality Scores AFTER Cleaning (paired_1.fastq)")
plt.legend(loc='lower left')
plt.ylim(0, 45)
plt.grid(alpha=0.3)
plt.show()
```