Bioinformatics Exercise

Scientific Skills Bioinformatics Exercise (30%)

Module Title: Scientific Skills

Program: MSc Biomedical Science

Assessment Weighting: 30% of Module Marks

Exercise Overview

This exercise introduces students to bioinformatics data handling, focusing on data cleaning, sequence analysis, and reproducibility. Students will work in a pre-configured Google Colab notebook with guided sections to facilitate learning without requiring prior programming experience.

Learning Objectives

1. Perform basic bioinformatics data cleaning and analysis on sequence data.
2. Calculate GC content and codon usage within specific regions of SARS-CoV-2 sequences.
3. Apply reproducibility and good organization practices in bioinformatics workflows.

Instructions for Accessing the Colab Notebook

1. **Access the Notebook**: Open the link provided on Moodle to access the pre-configured Google Colab notebook.
2. **Notebook Overview**: The notebook is divided into sections with instructions and explanations for each step.
3. **Running Cells**: Click on each code cell and press the “Run” button to execute the code and view results.
4. **Follow Along**: Each code block is accompanied by an explanation, so follow along and read the comments carefully.

Exercise Outline and Assessment Tasks

1. **Introduction to Bioinformatics Data Skills**
   Background: Brief overview of bioinformatics data formats, such as FASTA, and the importance of reproducibility. Students will work with SARS-CoV-2 sequences, focusing on calculating coverage and identifying specific genomic regions.
2. **Data Cleaning and Quality Control**
   - **Tasks**:
     1. Load the provided SARS-CoV-2 FASTA file (20 sequences, each 29,903 bp) into the Colab notebook.
     2. Filter out sequences with coverage below 85% (counting only A, C, T, and G bases).
     3. Summarize the cleaning process and explain how data quality impacts analysis.
   - **Code Block**: The Colab notebook includes a code block that loads and filters the sequences based on coverage. Students need to run the cell and observe the output.

```
# 1.Install and load the required packages (It will run slower because of the ne
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("Biostrings")
library(Biostrings)

# 2.Load the SARS-CoV-2 FASTA file
fasta_file <- "https://github.com/huobei1997/Bioinformatics_Exercise/raw/refs/he
sequences <- readDNAStringSet(fasta_file)
sequences

# 3. Calculate the total length of each sequence
total_length <- width(sequences)

# 4. Calculate the frequency of A, C, T, and G in each sequence
atcg_counts <- letterFrequency(sequences, letters = c("A", "C", "T", "G"))
coverage <- rowSums(atcg_counts) / total_length

# 5. Filter out sequences with coverage below 85%
filtered_sequences <- sequences[coverage >= 0.85]
filtered_sequences

# 6. Save the filtered sequences to a FASTA file
```

```
dir.create("Bioinformatics Exercise")
writeXStringSet(filtered_sequences, filepath = "./Bioinformatics Exercise/filter
```

```
DNAStringSet object of length 20:
     width seq                                                names
 [1] 29903 ------------------------...---------------------- hCoV-
19/HongKong/...
 [2] 29903 ------------------------...---------------------- hCoV-
19/HongKong/...
 [3] 29903 ------------------------...---------------------- hCoV-
19/HongKong/...
 [4] 29903 ------------------------...---------------------- hCoV-
19/HongKong/...
 [5] 29903 ------------------------...---------------------- hCoV-
19/HongKong/...
 ...   ... ...
[16] 29903 ------------------------...---------------------- hCoV-
19/HongKong/...
[17] 29903 ------------------------...---------------------- hCoV-
19/HongKong/...
[18] 29903 ------------------------...---------------------- hCoV-
19/HongKong/...
[19] 29903 ------------------------...---------------------- hCoV-
19/HongKong/...
[20] 29903 ------------------------...---------------------- hCoV-
19/HongKong/...
DNAStringSet object of length 19:
     width seq                                                names
 [1] 29903 ------------------------...---------------------- hCoV-
19/HongKong/...
 [2] 29903 ------------------------...---------------------- hCoV-
19/HongKong/
```

3. **Sequence Analysis**
   - **Tasks**:
     1. Calculate GC content for each of two randomly selected sequences.

```
# 1.Load libraries
library(Biostrings)

# 2.Set seed for reproducibility and Load the filtered sequences
set.seed(1)
filtered_sequences <- readDNAStringSet("./Bioinformatics Exercise/filtered_sequ

# 3.Randomly select two sequences
random_indices <- sample(length(filtered_sequences), 2)
selected_sequences <- filtered_sequences[random_indices]

# 4.Calculate GC content for each selected sequence
gc_counts <- letterFrequency(selected_sequences, letters = c("G", "C"))
total_lengths <- width(selected_sequences)
gc_content <- rowSums(gc_counts) / total_lengths

# 5.Output in the format "FASTA name: GC content" and Save the filtered sequenc
output <- paste0(names(selected_sequences), ": ", round(gc_content, 5))
output
writeXStringSet(selected_sequences, filepath = "./Bioinformatics Exercise/filte
```

```
'hCoV-19/HongKong/VB24316620/2024: 0.37595' · 'hCoV-19/HongKong/VB24316982/2024: 0.37592'
```

3. **Sequence Analysis**
   - **Tasks**:
     2. Extract the spike gene region (positions 21,563 to 25,384) for both sequences.

```
# 1.Load libraries
library(Biostrings)

# 2.Load the filtered sequences
filtered_sequences <- readDNAStringSet("./Bioinformatics Exercise/filtered_sequ

# 3.Define the positions for the spike gene region
start_position <- 21563
end_position <- 25384

# 4.Extract the spike gene region for both sequences
spike_gene_regions <- lapply(filtered_sequences, function(seq) {
  subseq(seq, start = start_position, end = end_position)
})

# 5.Output in the format "FASTA name: spike region" and Save the filtered seque
names(spike_gene_regions) <- names(filtered_sequences)
output <- sapply(spike_gene_regions, function(region) {
  paste0(names(region), ": ", as.character(region))
})
output

spike_dna_strings <- DNAStringSet(spike_gene_regions)  # Convert list to DNAStr
writeXStringSet(spike_dna_strings, "./Bioinformatics Exercise/filtered_sequence
```

```
hCoV-19/HongKong/VB24316620/2024:
': ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTATAACTACAA
TAAAGGTCCTAATTGTTACTTTCCTTTACAATCATATGGTTTCCGACCCACTTATGGTGTTGGT
hCoV-19/HongKong/VB24316982/2024:
': ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTATAACTACAA
TAAAGGTCCTAATTGTTACTTTCCTTTAGAATCATATGGTTTCCGACCCACTTATGGTGTTGGT
```

3. **Sequence Analysis**
   - **Tasks**:
     3. Calculate the codon usage for one of the extracted sequences.

```
# 1.Load necessary package
library(Biostrings)

# 2.Define the codon table
codon_table <- list(
  'ATA' = 'I', 'ATC' = 'I', 'ATT' = 'I', 'ATG' = 'M',
```

```
  'ACA' = 'T', 'ACC' = 'T', 'ACG' = 'T', 'ACT' = 'T',
  'AAC' = 'N', 'AAT' = 'N', 'AAA' = 'K', 'AAG' = 'K',
  'AGC' = 'S', 'AGT' = 'S', 'AGA' = 'R', 'AGG' = 'R',
  'CTA' = 'L', 'CTC' = 'L', 'CTG' = 'L', 'CTS' = 'L',
  'CCA' = 'P', 'CCC' = 'P', 'CCG' = 'P', 'CCT' = 'P',
  'CAC' = 'H', 'CAT' = 'H', 'CAA' = 'Q', 'CAG' = 'Q',
  'CGA' = 'R', 'CGC' = 'R', 'CGG' = 'R', 'CGT' = 'R',
  'GTA' = 'V', 'GTC' = 'V', 'GTG' = 'V', 'GTT' = 'V',
  'GCA' = 'A', 'GCC' = 'A', 'GCG' = 'A', 'GCT' = 'A',
  'GAC' = 'D', 'GAT' = 'D', 'GAA' = 'E', 'GAG' = 'E',
  'GGA' = 'G', 'GGC' = 'G', 'GGG' = 'G', 'GGT' = 'G',
  'TCA' = 'S', 'TCC' = 'S', 'TCG' = 'S', 'TCT' = 'S',
  'TTC' = 'F', 'TTT' = 'F', 'TTA' = 'L', 'TTG' = 'L',
  'TAC' = 'Y', 'TAT' = 'Y', 'TAA' = 'End', 'TAG' = 'End',
  'TGC' = 'C', 'TGT' = 'C', 'TGA' = 'End', 'TGG' = 'W'
)

# 3.Read the sequences
dna_sequences <- readDNAStringSet("./Bioinformatics Exercise/filtered_sequences

# 4.Extract the first sequence and convert it to a string
first_sequence <- dna_sequences[[1]]
dna_sequence <- as.character(first_sequence)

# 5.Split the sequence into codons
codons <- sapply(seq(1, nchar(dna_sequence) - 2, by = 3),
                 function(i) substring(dna_sequence, i, i + 2))

# 6.Create a data frame to store results
results <- data.frame(
  Codon = codons,
  AminoAcid = sapply(codons, function(codon) {
    if (codon %in% names(codon_table)) {
      return(codon_table[[codon]])  # Return the corresponding amino acid
    } else {
      return(NA)  # Return NA for codons not in the table
    }
  })
)

# 7.Filter out NA values
results <- results[!is.na(results$AminoAcid), ]

# 8.Calculate the frequency of each codon
codon_usage <- as.data.frame(table(results$Codon, results$AminoAcid))
colnames(codon_usage) <- c("Codon", "AminoAcid", "Count")

# 9.Calculate the total number of codons
```

```
total_codons <- sum(codon_usage$Count)

# 10.Calculate the usage probability of each codon
codon_usage$Probability <- codon_usage$Count / total_codons

# 11.Define full names for amino acids
amino_acid_names <- data.frame(
  Abbreviation = c('A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L',
                   'M', 'N', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y', 'End'),
  FullName = c('Alanine', 'Cysteine', 'Aspartic Acid', 'Glutamic Acid',
               'Phenylalanine', 'Glycine', 'Histidine', 'Isoleucine',
               'Lysine', 'Leucine', 'Methionine', 'Asparagine',
               'Proline', 'Glutamine', 'Arginine', 'Serine',
               'Threonine', 'Valine', 'Tryptophan', 'Tyrosine', 'Termination')
)

# 12.Merge full names into the codon usage data frame
codon_usage <- merge(codon_usage, amino_acid_names,
                     by.x = "AminoAcid", by.y = "Abbreviation", all.x = TRUE)

# 13.Output
output <- codon_usage[codon_usage$Count > 0, ]
colnames(output) <- c("Amino_acids", "Codon", "Count", "Probability", "Full Nam
print(output)
write.csv(output,"./Bioinformatics Exercise/codon_usage.csv",row.names=F)
```

```
    36            A   GCC     8 0.0065520066       Alanine
    37            A   GCG     2 0.0016380016       Alanine
    38            A   GCT    41 0.0335790336       Alanine
    114           C   TGC    12 0.0098280098      Cysteine
    116           C   TGT    28 0.0229320229      Cysteine
    152           D   GAC    18 0.0147420147 Aspartic Acid
    154           D   GAT    44 0.0360360360 Aspartic Acid
    211           E   GAA    33 0.0270270270 Glutamic Acid
    213           E   GAG    13 0.0106470106 Glutamic Acid
    287         End   TAA     1 0.0008190008   Termination
    358           F   TTC    20 0.0163800164 Phenylalanine
    360           F   TTT    59 0.0483210483 Phenylalanine
    399           G   GGA    18 0.0147420147       Glycine
    400           G   GGC    15 0.0122850123       Glycine
    401           G   GGG     4 0.0032760033       Glycine
    402           G   GGT    44 0.0360360360       Glycine
    438           H   CAC     5 0.0040950041     Histidine
    440           H   CAT    14 0.0114660115     Histidine
    493           I   ATA    18 0.0147420147    Isoleucine
    494           I   ATC    14 0.0114660115    Isoleucine
    496           I   ATT    43 0.0352170352    Isoleucine
    541           K   AAA    45 0.0368550369        Lysine
    543           K   AAG    23 0.0188370188        Lysine
    628           L   CTA     9 0.0073710074       Leucine
    629           L   CTC    11 0.0090090090       Leucine
```

630	L	CTG	2	0.0016380016	Leucine
657	L	TTA	27	0.0221130221	Leucine
659	L	TTG	19	0.0155610156	Leucine
675	M	ATG	14	0.0114660115	Methionine
722	N	AAC	34	0.0278460278	Asparagine
724	N	AAT	48	0.0393120393	Asparagine
801	P	CCA	26	0.0212940213	Proline
802	P	CCC	3	0.0024570025	Proline
803	P	CCT	26	0.0212940213	Proline
857	Q	CAA	44	0.0360360360	Glutamine
859	Q	CAG	15	0.0122850123	Glutamine
909	R	AGA	15	0.0122850123	Arginine
911	R	AGG	11	0.0090090090	Arginine
924	R	CGA	1	0.0008190008	Arginine
925	R	CGC	1	0.0008190008	Arginine
926	R	CGG	2	0.0016380016	Arginine
927	R	CGT	10	0.0081900082	Arginine
970	S	AGC	5	0.0040950041	Serine
972	S	AGT	18	0.0147420147	Serine
1010	S	TCA	25	0.0204750205	Serine
1011	S	TCC	9	0.0073710074	Serine
1012	S	TCG	3	0.0024570025	Serine
1013	S	TCT	37	0.0303030303	Serine
1025	T	ACA	40	0.0327600328	Threonine
1026	T	ACC	9	0.0073710074	Threonine
1027	T	ACG	4	0.0032760033	Threonine
1028	T	ACT	44	0.0360360360	Threonine
1123	V	GTA	15	0.0122850123	Valine
1124	V	GTC	20	0.0163800164	Valine
1125	V	GTG	12	0.0098280098	Valine
1126	V	GTT	46	0.0376740377	Valine
1195	W	TGG	13	0.0106470106	Tryptophan
1248	Y	TAC	12	0.0098280098	Tyrosine
1249	Y	TAT	43	0.0352170352	Tyrosine

3. **Sequence Analysis**
    - **Guided Code Blocks**: Each analysis task has its own code block with comments explaining what each line does. Students simply run the cells and observe the results.

4. **Reproducibility and Data Organization**
    - **Tasks**:
        1. Follow best practices for naming and organizing files within the notebook.
        2. Ensure reproducibility by setting a random seed for the code block that selects sequences.
        3. Observe comments within the notebook, explaining how reproducibility is maintained.
    - **Guided Example**: Students will be guided to set up a structured folder in Colab and see an example of setting a random seed to make results reproducible.
5. **Reflection on Bioinformatics Data Skills**
    - Reflect on the importance of reproducibility and good data organization practices in bioinformatics workflows.

Submission Guidelines

The report should include:

1. **Introduction (10 marks)**
    - Overview of bioinformatics data skills and significance.
2. **Data Cleaning and Quality Control Summary (20 marks)**
    - Explanation of the data cleaning process based on coverage and its importance.
3. **Sequence Analysis (35 marks)**
    - Results and interpretation of GC content and codon usage analysis (Are the results the same for the two sequences? Interpret your observation).
4. **Notebook Organization and Reproducibility (20 marks)**
    - A print of the organized notebook which you used for generating the results, with readable code and ensuring the results are reproduceable.
5. **Reflection (15 marks)**
    - Reflection on the importance of reproducibility and data organization.