

LIBS457 BIOINFORMATICS

# Python Assignment Final Report

BHAWARNAWA GEDE PRASIDHA - 2023952691

“This assignment is for writing Python scripts to calculate G content in a given window size and identify G-enriched regions in SARS-CoV-2 RNA genome over background (G content from shuffled sequences). Please submit Python scripts and result files for each question.”

**Q1:** This is the first step, which we try to determine the level of significant G-content from shuffled sequences that result in  $P < 0.05$ . For the precise calculation, we should do it with many times of iteration (e.g.  $n=1000$ ).

**Script 1:** Please write a python script, which take an input sequence from SARS-CoV-2 RNA genome (NC\_045512.2), and make a given number ( $n=1000$ ) of shuffled sequence in FASTA format. Please submit this python script and an output FASTA file, consist of 1000 shuffled sequences.

**Answer:**

For the solution to this part, there are two python files involved: sequence\_class.py and script\_1.py.

**sequence\_class.py**

```
sequence_class.py
1  class Seq:
2      def __init__(self, seq_num, seq_str):
3          self.seq_num = seq_num
4          self.seq_str = seq_str
5
6      def get_str(self):
7          return self.seq_str
8
9      def get_len(self):
10         return len(self.seq_str)
11
12     def get_seq_num(self):
13         return self.seq_num
14
15     def print_fasta_format(self):
16         print(">Seq{}".format(self.seq_num))
17         print(self.seq_str)
18         print()
19
20     def return_fasta_format(self):
21         return ">Seq{}\n{}\n".format(self.seq_num, self.seq_str)
```

This python file contains an object class to initiate instances of “Seq” that encapsulates two different data types: “seq\_num” which is the sequence number and “seq\_str” which contains the sequence string. The class also holds multiple public functions, such as: `__init__(self, seq_num, seq_str)` that is used to initiate a class instance or an object of “Seq” with the corresponding sequence number and sequence string, `get_str()` that returns the sequence, `get_len()` that returns the sequence string’s

length, `get_seq_num()` that returns the sequence number, `print_fasta_format()` that prints a text of sequence number and sequence string in the FASTA format into the user's interface, and `return_fasta_format()` that returns the FASTA format text as an output.

### script\_1.py

```
script_1.py
1  import os
2  import time
3  import random
4  from sequence_class import Seq
5
6  def readSARS_CoV2_FASTA():
7      # Assumes that the FASTA file is in the same directory as the script
8      # Also assumes that the file is named "sequence.fasta"
9      with open("sequence.fasta", "r") as f:
10         # Read the file and store the contents in a variable
11         file_contents = f.readlines()
12         lines = []
13         for line in file_contents:
14             # Skip the lines that start with ">" or lines that doesn't start with "A", "G", "T", or "C"
15             if line.startswith(">") or not line.startswith(("A", "G", "T", "C")):
16                 continue
17             # Print the lines of the file
18             lines.append(line.strip())
19         # Create initial Seq object
20         seq = Seq(0, "".join(lines))
21         return seq
22
23  def main():
24      start_time = time.time()
25      seq_0 = readSARS_CoV2_FASTA()
26      # Create 1000 shuffled sequences in FASTA format
27      for i in range(0, 1000):
28          # Creates a new Seq object with the shuffled sequence
29          seq = Seq(i+1, ''.join(random.sample(seq_0.get_str(), len(seq_0.get_str()))))
30          # Write the FASTA format to a file
31          with open("output_script_1.txt", "a") as f:
32              f.write(seq.return_fasta_format())
33      end_time = time.time()
34      print("Done!")
35      print("Time elapsed: " + str(end_time - start_time) + " seconds")
36
37  if __name__ == "__main__":
38      main()
```

This python file reads the previously downloaded FASTA format of the SARS-CoV-2 RNA Genome, with the reference sequence (RefSeq) accession number of NC\_045512.2. Note that the FASTA format must be downloaded into a file name of "sequence.fasta" or the script will not be able to read and process the file. It will ignore the first line of sequence description and process the sequence data alone, obtaining only the "A", "G", "T", "C" nucleotide bases. After obtaining the raw data of nucleotide bases, it will store it as the object "Seq" with identifier number 0, named variable "seq\_0". This will be the baseline for the 1000 randomized sequences. After that, 1000 randomized sequences will be created and written as multiple FASTA formats accumulated in the "output\_script\_1.txt" file. There's also a timer to count the time the script took to process the randomized sequences and print it to file.

### output\_script\_1.txt (Snippet: Seq 19/1000)

```
output_script_1.txt
1  >Seq1
2  AAAGTAACCAAGCTCTTGCCAGGGAAGCCAGAATTCGTTCCGCTGATTTGGATTTGAATGACGGGCAATAACAATTGACTAGCAGAGCTCGGTT
3  >Seq2
4  GTTAAATCGAATTTAAAGTATCCGTATTAACATCTGACCGTTTAAGAGCTTATTGCGACATCTACGAGTACTTCCCAAGACCAATCCTATTCAA
5  >Seq3
6  GCCTATACGATACAGTATGAAGCTCTATGGCTTATTCGATAATGATAAACTCGGTAGGACTTGATCTGCACATCGTTACTCAATCATTGAGACGG
7  >Seq4
8  TATTAGGGAATCGAAGTATCACTTTTAAAGGTCTAGAACATTGCTCACATATTATGTATATACGGGATTAAGGGTCTGATATTACGATCTCCAAC
9  >Seq5
10 ATGACTTGAAATCTTCGCAAAAGTAAGTATATGGCTGGTTCCTTCATAATCAGAACATCTATAGATGCAGCTAATTGTGTATGTTAATTTGCCAT
11 >Seq6
12 ACCTACACACGTCCTCAAAACGTCAGTTTATAAAAACTAATGGTAGCTTGCTGCTCTATCGCGTACAGTGAGGTATTCTCGATAAATTTTCGT
13 >Seq7
14 TAAATTATTCGGAGCCATCTCAACACTAGTCGGACCAATCCGGGATCATTGTTTCATTGACATTGCTAGGGGCAATAACCCAGATATAT
15 >Seq8
16 ATCTCAAATGCAACTTTTAGAAAATCGCTCTCAAGTAGTGTTCTTTTTTGAGGAGAGGACATTGATCGCTCTCTTGTATGCAGAAAGGGCAAC
17 >Seq9
18 AAAAATAGCAAGAATTTCCAACCTCCAGCGCAAGATTAGATATAACACTTTCCAGGATAACCTTTTATTAGGTTCTAATCTGAGAGAACTACGT
19 >Seq10
20 AGGTTTTGATTTGTTATAGTACGTCCATCATCTATGCAACGTCATTCACAGACAGATGTTGGGCGAACAGACCGAAATCGAAGTCTAGTCCGCTGA
21 >Seq11
22 TACCGCTGGCCTTTCAATGCGCGCTGTACCAAAATCCCCGAGCAGGGAATAATTATGTCTTCTTCCGAAAGGCTAGCACATTAAATTAACATG
23 >Seq12
24 GCGATTCTGGGATCGCTCCAGATTGTTCTAACGTGGTTTATGATTTCATCCGGTTAACATACTACTAGATATACAACCTTTAGGACTATAATTAA
25 >Seq13
26 AGCATAACCTAGTCTCATATAAAACAGGTGCCTCGAAGGGATTCTATAGTGTCCCTAATAACCTATTACGTCAATATGGCATGAGCAGCCCTAA
27 >Seq14
28 AAATAAATACTTGTAAACATTGAAGTTTGCCCGGACTTGTCGGTAGCTTATCGAGTCGCTGTCTAATTTGATGCTGTTCTTGATACGCTTATAC
29 >Seq15
30 CGTTATCCCATTTTCGAGATAGCATTGATGATATTGACGCGCTCATTCTGTAGACTTGGGCAAAGCTGACTAGGTGATTCAATATCGACTCAAG
31 >Seq16
32 TTAGTATTGCCATTAACATATGTACATCTTGACGGTTAGAGAATTGTCATGCCTATCTTGAAAGTACAGTGACTGCTTAGAATTATGCACGACCG
33 >Seq17
34 CGCTCAGCATAAATGGTAGAGAATTTAACTCAATAATACAAATCCGCAGTCGCAACATCAACCAATCTCGATGTGTGAGTCATAAAATCA
35 >Seq18
36 CACTGTTGAATTTTAGGCGAATACGTTCAAGGATATTACTGTCGCGAGATCATTGCTTCTATTTTAGTAACTGCTTAGTCGACATGTCTGTGA
37 >Seq19
38 ATGATACTATGAATATGCGACTGGTCGAACATTTTATTTAAGCAATGTTTATTAACCGAATCATGGAGAACTTCAAACAAAACTGGTCT
```

### Output of running script\_1.py

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS
PS C:\Users\Gede Prasadha\Documents\Kuliah Semester 6\Amin IISMA\LIBS457 Bioinformatics\Projects & Assignm
ents\Python Assignment\LIBS457-Bioinformatics-SARS-CoV-Python-Analyzer> py script_1.py
Done!
Time elapsed: 11.301375389099121 seconds
```

**Script 2:** Please write a python script, which take input sequences from FASTA file (shuffled 1000 sequences) and counts number of G in 50nt window there (G-content) by sliding it down with 1nt movement, stores the results (maybe dictionary data structure) and finds the values reaching more than top 5% of the G-content population. You have to provide both a python script and value of significant G-content ( $P < 0.05$ ), that will be used as cutoff for following question.

**Answer:**

For the solution to this part, only one script is necessary: script\_2.py

## script\_2.py

```
script_2.py
1  import time
2
3  def read_output_script_1():
4      # Read the file and return an array of the contents
5      filename = "output_script_1.txt"
6      with open(filename, "r") as f:
7          file_contents = f.readlines()
8          lines = []
9          for line in file_contents:
10             if line.startswith(">"):
11                 continue
12             lines.append(line.strip())
13     return lines
14
15 def main():
16     # Steps on how this algorithm works:
17     # 1. Read the file and store the contents in an array
18     # 2. There are 1000 sequences now stored in the array, now we need to split
19     #    each sequence into 50nt windows
20     # 3. Count the number of Gs in each 50nt window
21     # 4. Save the number of Gs in each 50nt window in a 2D array named temp_results
22     #    Structure of said array is [start index, percentage of Gs in 50nt window]
23     # Sort the results, return the threshold of the top 5% of the G-content population
24     # Repeat for all 1000 sequences, store in a 2D array named final_results with the
25     #    structure of [sequence number, threshold of the top 5% of the G-content population]
26     # Return the average of the top 5% of the G-content population from all 1000 sequences
27     start_time = time.time()
28     sequences = read_output_script_1()
29     final_results = []
30     for i in range(len(sequences)):
31         temp_results = []
32         for j in range(len(sequences[i])-49):
33             temp_results.append([j, sequences[i][j:j+50].count("G")])
34         temp_results = sorted(temp_results, key=lambda x: x[1])
35         threshold = temp_results[int(len(temp_results)*0.95)][1]
36         final_results.append([i+1, threshold])
37     threshold = sum([i[1] for i in final_results])/len(final_results)
38     end_time = time.time()
39     print("Threshold for G-enrich regions in a 50 nucleotide window is {}".format(threshold))
40     print("Calculation is finished!")
41     print("Time elapsed: {} seconds".format(end_time - start_time))
42     # The threshold is 14.665 with current random seed of 1000 sequences
43
44
45 if __name__ == "__main__":
46     main()
```

This python script runs by starting to look for the output of the previous script, which is "output\_script\_1.txt", and note every genomic sequence in FASTA format inside using the read\_output\_script\_1() function. Note that if the file is missing or is located outside the folder the script is run, the program will not work. Every genomic sequence is then stored in the variable "sequences" as a list data type. Reason for choosing a list data type with indexes from 0 to 99 instead of a dictionary data type with key-value pair of 1 to 100 and their respective sequence data is lists are simpler to generate and have the same read/write by index (for lists) or key (for dictionaries) time complexity, which is  $O(1)$ . It will also help with the second step of the computation process, which is to order the amount of guanine (G) bases inside the range.

Now that 1000 sequences has been stored inside “sequences”, the script will start to iterate those sequences one by one to find each 95% threshold of the G-content population for a 50 nucleotide window range. The script runs in a loop, where it feeds a sequence inside and then slice it per 50 nucleotide range and slides the slicer by 1 nucleotide. For each slice, it counts the number of G-content inside and stores it inside the “temp\_results” list with the format of [start\_index, G-content count]. After slicing the sequence to the end, the contents of the “temp\_results” will be sorted by the amount of G-content count for each 50 nucleotide window. Then, it will retrieve the index of the array that holds the 95% top G-content value, namely “threshold”, and append the value into the “final\_results” list in the format of [sequence\_number in base 1, threshold]. This will loop for each of the 1000 sequences until every sequence has been accounted for in the “final\_results” list. After that, the script will return the average threshold for all 1000 sequences. It will also return the total calculation time of the whole script.

### Output of running script\_2.py

```
PS C:\Users\Gede Prasadha\Documents\Kuliah Semester 6\Amin IISMA\LIBS457 Bioinformatics\Projects & Assignments\Python Assignment\LIBS457-Bioinformatics-SARS-CoV-2\Python-Analyzer> py script_2.py
Threshold for G-enrich regions in a 50 nucleotide window is 14.665
Calculation is finished!
Time elapsed: 27.983439207077026 seconds
```

For this assignment, the threshold returned from over background analysis of 1000 randomized sequences is 14.665 G-content per 50 nucleotide window.

**Q2: Script 3:** Please write another Python script, which also takes input sequences from FASTA format (this time input is SARS-CoV-2 RNA genome; NC\_045512.2), calculates G-content as we did in the previous question (number of G in 50nt window with 1nt sliding), stores the result only if the G-content is more than threshold (the cutoff value decided as  $P < 0.05$  in the previous step) and returns the final results as a following tab-delimited text file (containing tabs that separate information with one record per line), comprising name, start position of the window (start), end position of the window (end) and G-content. You have to provide both a python script and an output tab-delimited text files.

### Answer:

For the solution to this part, only one script is required: script\_3.py

### script\_3.py

```
script_3.py
1  import csv
2  import time
3
4  def readSARS_CoV2_FASTA():
5      # Assumes that the FASTA file is in the same directory as the script
6      # Also assumes that the file is named "SARS-CoV-2 RNA Genome FASTA.txt"
7      with open("sequence.fasta", "r") as f:
8          # Read the file and store the contents in a variable
9          file_contents = f.readlines()
10         lines = []
11         for line in file_contents:
12             # Skip the lines that start with ">" or lines that doesn't start with "A", "G", "T", or "C"
13             if line.startswith(">") or not line.startswith(("A", "G", "T", "C")):
14                 continue
15             # Print the lines of the file
16             lines.append(line.strip())
17         # Create initial Seq object
18         return "".join(lines)
19
20 def main():
21     threshold_limit = float(input("Input the threshold value from the previous script: "))
22     start_time = time.time()
23     sequence = readSARS_CoV2_FASTA()
24     final_results = []
25     for idx in range(len(sequence) - 49):
26         g_count = sequence[idx:idx+50].count("G")
27         if g_count > threshold_limit:
28             final_results.append([idx, g_count])
29     final_results = sorted(final_results, key=lambda x: x[1], reverse=True)
30     with open("top_5_g_enriched_regions.txt", "w", newline="") as f:
31         writer = csv.writer(f, delimiter="\t")
32         writer.writerow(["Name", "Start", "End", "G-content"])
33         for i in range(5):
34             writer.writerow(["NC_045512.2", final_results[i][0], final_results[i][0]+49, final_results[i][1]])
35     with open("all_g_enriched_regions.txt", "w", newline="") as f:
36         writer = csv.writer(f, delimiter="\t")
37         writer.writerow(["Name", "Start", "End", "G-content"])
38         for i in range(len(final_results)):
39             writer.writerow(["NC_045512.2", final_results[i][0], final_results[i][0]+49, final_results[i][1]])
40     end_time = time.time()
41     print("Calculation is finished!")
42     print("Time elapsed: {} seconds".format(end_time - start_time))
43
44 if __name__ == "__main__":
45     main()
```

This script firstly read the same SARS-CoV-2 RNA Genome FASTA format file (named "sequence.fasta"), but instead of saving it in a "Seq" class instance, it returns it as a single-line string ready to be processed and further analysis. The script then asks the user for manual input of the threshold that was outputted from the previous step. Then, it will slice the original SARS-CoV-2 RNA sequence data into multiple 50 nucleotides windows and retrieved the number of G-content inside said windows. It will then store the starting index of the nucleotide window and the amount of G-content inside the window as an appendage to "final\_results" variable IF the amount of G-content is more than the previously inputted threshold, which is 14.665 guanine bases in this assignment. After every window is accounted for, the contents of "final\_results" variable is sorted again from the window with the most enriched G-content to the window with the least enriched G-content. The top 5 G-content enriched regions are saved inside "top\_5\_g\_enriched\_regions.txt" as a tab-delimited text file and all G-content enriched regions are saved inside "all\_g\_enriched\_regions.txt" as a similar text output. Note that all indexes (start and end) are of base 1.

## Output of running script\_3.py

```
PS C:\Users\Gede Prasadha\Documents\Kuliah Semester 6\Amin IISMA\LIBS457 Bioinformatics\Projects & Assignments\Python Assignment\LIBS457-Bioinformatics-SARS-CoV
-Python-Analyzer> py script_3.py
Input the threshold value from the previous script: 14.665
Calculation is finished!
Time elapsed: 0.06330251693725586 seconds
```

## top\_5\_g\_enriched\_regions.txt

```
top_5_g_enriched_regions.txt
1  Name      Start   End G-content
2  NC_045512.2 328 377 20
3  NC_045512.2 329 378 20
4  NC_045512.2 330 379 20
5  NC_045512.2 331 380 20
6  NC_045512.2 236 285 19
7
```

## all\_g\_enriched\_regions.txt (Snippet: 23/1296 lines)

```
all_g_enriched_regions.txt
1  Name      Start   End G-content
2  NC_045512.2 328 377 20
3  NC_045512.2 329 378 20
4  NC_045512.2 330 379 20
5  NC_045512.2 331 380 20
6  NC_045512.2 236 285 19
7  NC_045512.2 327 376 19
8  NC_045512.2 332 381 19
9  NC_045512.2 340 389 19
10 NC_045512.2 341 390 19
11 NC_045512.2 520 569 19
12 NC_045512.2 2154 2203 19
13 NC_045512.2 2155 2204 19
14 NC_045512.2 18295 18344 19
15 NC_045512.2 18296 18345 19
16 NC_045512.2 28870 28919 19
17 NC_045512.2 28871 28920 19
18 NC_045512.2 29734 29783 19
19 NC_045512.2 235 284 18
20 NC_045512.2 237 286 18
21 NC_045512.2 246 295 18
22 NC_045512.2 325 374 18
23 NC_045512.2 326 375 18
24 NC_045512.2 333 382 18
```



**Q3.** Please analyze top 5 G-enriched regions from your results from question 2. What are the names of genes encoded in SARS-CoV-2 genome and their functions. Where are those top 5 Genriched regions located (5'UTR, CDS, or 3'UTR)? Please submit all answers for these questions.

**Answer:**

Using the UCSC SARS-CoV-2 Genome Browser and the NIH's NCBI Virus' SARS-CoV-2 Data Hub as the reference and literature source and the answers gained on the previous question (top\_5\_g\_enriched\_regions.txt), the following table is created:

RefSeq Accession Number	Start Index (Base 1)	End Index (Base 1)	G-Content	Gene Name	Location of Nucleotide Window in Transcript Annotation
NC_045512.2	328	377	20	ORF1ab	CDS
	329	378	20		
	330	379	20		
	331	380	20		
	327	376	19		

#### ORF1ab Gene Identifier (Taken from NCBI GenBank)

```
>lcl|NC_045512.2_gene_1 [gene=ORF1ab] [locus_tag=GU280_gp01]
[db_xref=GeneID:43740578] [location=266..21555] [gbkey=Gene]
```

The top 5 G-enriched regions are all parts of the ORF1ab gene. It's main (local) identifier on the National Institute of Health's (NIH) National Center for Biotechnology Information (NCBI) Database is NC\_045512.2\_gene\_1 and its locus tag, or the shorthand label for tracking and referencing in specific studies, is GU280\_gp01. The location of this gene on the SARS-CoV-2 genome sequence is from the base-1 index 266 to 21555. This gene is likely to be a housekeeping gene. The reason for this is that it consists of only the CDS (coding sequence) without any 5' untranslated region (5' UTR) or 3' untranslated region (3' UTR). Other possible reason for this is that ORF1ab gene is the largest gene within the SARS-CoV-2 virus and it's responsible for many main functionalities which will be described later on and housekeeping genes are mostly essential for the basic cellular processes such as metabolism, protein synthesis, and DNA maintenance.

SARS-CoV-2 is an enveloped (protected by a protective lipid bilayer membrane), positive-sense single-stranded RNA (which means that the RNA sequence directly matches the viral mRNA needed for protein production; this means that the virus can immediately hijack the host cell's ribosomes and start translating its RNA into viral proteins) that causes the coronavirus disease 2019 (COVID-19). It infects the host through invasion of host cells by the virus particles that contains the RNA genetic material and structural proteins. Once inside the cell, the infecting RNA then encodes structural proteins that make up virus particles. Within SARS-CoV-2 genomic structure, exists ORF1ab, its largest gene. It contains the overlapping open reading frames that encode polyproteins PP1ab and PP1a. Overlapping open reading frames are instances where the same stretch of a DNA sequence can encode for two or more different proteins, but in different reading frames. When separated, the polyproteins can be cleaved to yield 16 nonstructural proteins (NSPs), NSP1-16, with the production

of the PP1ab or PP1a depends on the ribosomal frameshifting event. The following are some explanations on the NSPs and their impact on the spread of COVID-19 pandemic:

1. NSP3 and NSP5 are proteases or enzymes that cut proteins into smaller pieces, specifically IRF-3, NLRP12, and TAB1. They break down those proteins to provide amino acids for viral replication and specifically for NLRP12 and TAB1, when they are broken down, it causes enhanced production of cytokines and inflammatory response observed in COVID-19 patients.
2. NSP12 is the RNA-dependent RNA polymerase (RdRp) or the enzyme that makes copies of the viral RNA genome.
3. NSP13 is a helicase or the enzyme that unwinds double-stranded RNA for RdRp to access the RNA template and make more copies of itself post-infection.

Therefore, it can be summarized that the top 5 regions with the highest G-content within the 50 nucleotide window are all components within the part of SARS-CoV-2 that causes the most damage and infection rate.

**Q4. Script 4:** Please go to the website and download all reported mutations in SARS-CoV-2 “mutations in SARS-CoV-2 SRA Data ([https://www.ncbi.nlm.nih.gov/labs/virus/vssi/#/scov2\\_snp](https://www.ncbi.nlm.nih.gov/labs/virus/vssi/#/scov2_snp)) It will be downloaded as “mutations.csv” and extract its coordination in NC\_045512.2, located in the third column (You can open csv file by using Excel). Please write a python script that calculate G-content in the 50nt flanking regions from the mutation sites (-25/+25) and select top 10 enriched regions with G-content. Do these selected regions contain G-content more than threshold (the cutoff value decided as  $P < 0.05$  in the previous step)? Why this region in SARS-CoV-2 genome was selected? What is the function of these region (what kind of protein in SARS-CoV-2 is expressed there)? Can you explain pathology of SARS-CoV-2 with this mutation region? Please submit this python script and results with answer.

**Answer:**

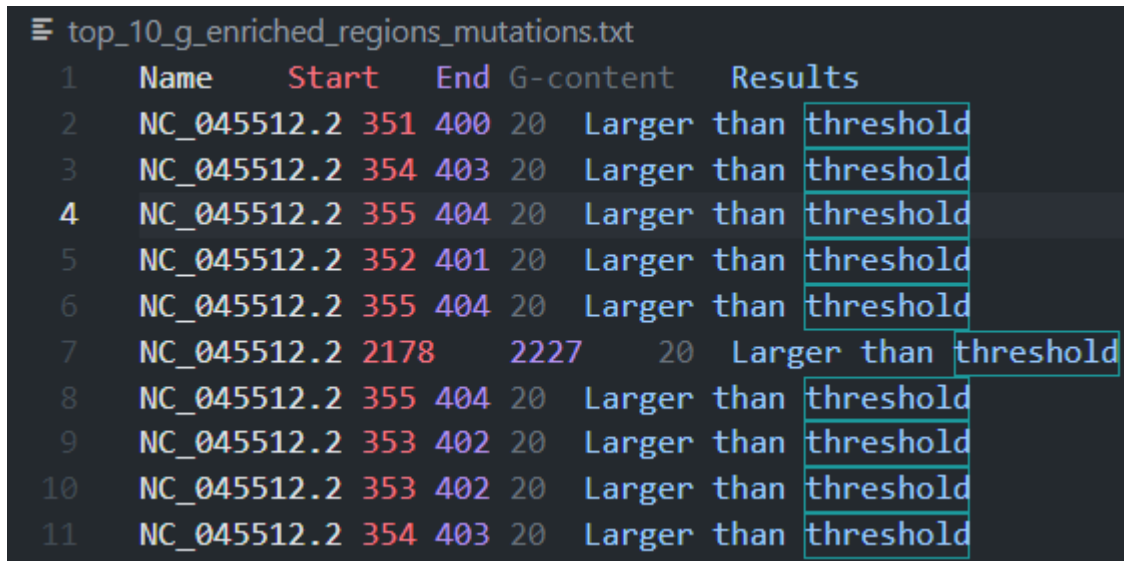
For the solution to this part, only one script is required: script\_4.py

## script\_4.py

```
script_4.py
1  import csv
2
3  def read_SARS_CoV_2_FASTA():
4      with open("sequence.fasta", "r") as f:
5          # Read the file and store the contents in a variable
6          file_contents = f.readlines()
7          lines = []
8          for line in file_contents:
9              # Skip the lines that start with ">" or lines that doesn't start with "A", "G", "T", or "C"
10             if line.startswith(">") or not line.startswith(("A", "G", "T", "C")):
11                 continue
12             # Print the lines of the file
13             lines.append(line.strip())
14         # Create initial Seq object
15         return "".join(lines)
16
17 def main():
18     # Read into "mutations.csv" file and retrieve only the fourth column
19     # Reason being we only need the genomic locations of the mutations
20     with open("mutations.csv", "r") as f:
21         file_contents = f.readlines()
22         locations = []
23         for line in file_contents:
24             locations.append(line.strip().split(",")[3])
25     # Then, we read the FASTA format of the original SARS-CoV-2 sequence
26     sequence = read_SARS_CoV_2_FASTA()
27     # Setting the min and max value of the range so that we don't go out of bounds
28     min_value = 0
29     max_value = len(sequence)
30     # For each genomic location in the "mutations.csv" file, named X
31     # We create a range of [X - 25, X + 25] and count the G-population inside the range
32     # Then, we append the count to a list named "final_results"
33     # in the format of [genomic_location, G-population]
34     threshold = 14.665 # Hard-coded, change if needed
```

```
script_4.py
34     threshold = 14.665 # Hard-coded, change if needed
35     final_results = []
36     for elem in locations:
37         elem = int(elem)
38         count = 0
39         min_range_val = elem - 25 if elem - 25 >= min_value else min_value
40         max_range_val = elem + 25 if elem + 25 <= max_value else max_value
41         g_count = sequence[min_range_val:max_range_val+1].count("G")
42         final_results.append([elem, g_count, "Larger than threshold" if g_count > threshold else "Smaller than threshold"])
43     final_results = sorted(final_results, key=lambda x: x[1], reverse=True)
44     with open("top_10_g_enriched_regions_mutations.txt", "w", newline="") as f:
45         writer = csv.writer(f, delimiter="\t")
46         writer.writerow(["Name", "Start", "End", "G-content", "Results"])
47         for i in range(10):
48             writer.writerow(["NC_045512.2", final_results[i][0], final_results[i][0]+49, final_results[i][1], final_results[i][2]])
49     less_than_threshold = 0
50     more_than_threshold = 0
51     for result in final_results:
52         if result[2] == "Smaller than threshold":
53             less_than_threshold += 1
54         else:
55             more_than_threshold += 1
56     print("Number of regions with G-content less than threshold: {} ({}% of total)".format(less_than_threshold, (less_than_threshold/(less_than_threshold+more_than_threshold))*100))
57     print("Number of regions with G-content more than threshold: {} ({}% of total)".format(more_than_threshold, (more_than_threshold/(less_than_threshold+more_than_threshold))*100))
58
59
60
61 if __name__ == "__main__":
62     main()
```

#### top\_10\_g\_enriched\_regions\_mutations.txt



	Name	Start	End	G-content	Results
1	NC_045512.2	351	400	20	Larger than threshold
2	NC_045512.2	354	403	20	Larger than threshold
4	NC_045512.2	355	404	20	Larger than threshold
5	NC_045512.2	352	401	20	Larger than threshold
6	NC_045512.2	355	404	20	Larger than threshold
7	NC_045512.2	2178	2227	20	Larger than threshold
8	NC_045512.2	355	404	20	Larger than threshold
9	NC_045512.2	353	402	20	Larger than threshold
10	NC_045512.2	353	402	20	Larger than threshold
11	NC_045512.2	354	403	20	Larger than threshold

#### Results of running script\_4.py

```
PS C:\Users\Gede Prasadha\Documents\Kuliah Semester 6\Amin IISMA\LIBS457 Bioinformatics\Projects & Assignments\Python Assignment\LIBS457-  
Bioinformatics-SARS-CoV-Python-Analyzer> py script_4.py  
Number of regions with G-content less than threshold: 24524 (94.30494135743126% of total)  
Number of regions with G-content more than threshold: 1481 (5.695058642568736% of total)
```

As viewed on the results, the top 10 50 nucleotide window that contains the mutation location at the center of the window have larger amounts of guanine bases than the previously determined threshold. Coincidentally enough, all of them are located inside the ORF1ab gene, the largest gene that contains most parts of the virus' regular cellular activities. In molecular biology and genetics, there is a term called GC-content or guanine-cytosine content or the percentage of nitrogenous bases in a DNA/RNA molecule that are either guanine (G) or cytosine (C). GC-rich, included G-regions, typically include many protein-coding genes within them and while further research is required to pinpoint which of the polyprotein (PP1ab or PP1a) are contained within each window and to pinpoint the change caused by a certain mutation towards the virus' functionality, it's evident enough that the mutations that are in an area with high ratio of G-content are more likely to change the way the SARS-CoV-2 behave and function. It needs to be noted that not all mutations can cause huge changes towards the behaviour of the virus, considering that 94.3% exists in regions with lower G-content than the over background analysis' threshold.

## REFERENCES

- Zhang, X., Yang, Z., Pan, T., Sun, Q., Chen, Q., Wang, P.-H., Li, X., & Kuang, E. (2023). SARS-COV-2 NSP8 suppresses MDA5 antiviral immune responses by impairing TRIM4-mediated K63-linked polyubiquitination. *PLOS Pathogens*, 19(11). <https://doi.org/10.1371/journal.ppat.1011792>
- Gan, H., Zhou, X., Lei, Q., Wu, L., Niu, J., & Zheng, Q. (2024). RNA-dependent RNA polymerase of SARS-COV-2 regulate host mrna translation efficiency by hijacking EEF1A factors. *Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease*, 1870(1), 166871. <https://doi.org/10.1016/j.bbadis.2023.166871>
- Zhang, Y., Xin, B., Liu, Y., Jiang, W., Han, W., Deng, J., Wang, P., Hong, X., & Yan, D. (2023). SARS-COV-2 protein NSP9 promotes cytokine production by targeting TBK1. *Frontiers in Immunology*, 14. <https://doi.org/10.3389/fimmu.2023.1211816>
- Marx, S. K., Mickolajczyk, K. J., Craig, J. M., Thomas, C. A., Pfeffer, A. M., Abell, S. J., Carrasco, J. D., Franzi, M. C., Huang, J. R., Kim, H. C., Brinkerhoff, H., Kapoor, T. M., Gundlach, J. H., & Laszlo, A. H. (2023). Observing inhibition of the SARS-COV-2 helicase at single-nucleotide resolution. *Nucleic Acids Research*, 51(17), 9266–9278. <https://doi.org/10.1093/nar/gkad660>