Final_Project
Script Programming

Zain Nawaz

Part 1: Linux

1) Counting the words in both files

```
zain@zain-HP-EliteBook-840-G1:~/Desktop/ScriptProgramming
/MainProject$ wc genome*.dat
1      5 100034 genome_01.dat
1      5 100034 genome_02.dat
2     10 200068 total
```

2) Comparing the files

```
zain@zain-HP-EliteBook-840-G1:~/Desktop/ScriptProgramming
/MainProject$ cmp genome*.dat
genome_01.dat genome_02.dat differ: byte 8, line 1
```

```
zain@zain-HP-EliteBook-840-G1:~/Desktop/ScriptProgramming
/MainProject$ cmp -l genome*.dat
     8  61  62
   950 101 107
   1510 101 103
  .................
  .................
 93224 124 107
 93537 107 103
```

3) Piping the result

```
zain@zain-HP-EliteBook-840-G1:~/Desktop/ScriptProgramming
/MainProject$ cmp -l genome*.dat | wc -l
273
```

4) Finding all occurrence in the String

```
zain@zain-HP-EliteBook-840-G1:~/Desktop/ScriptProgramming
/MainProject$ grep "AAAAAAAAAATTTTTTTTTT" genome_01.dat
TGATCGGACCCCAATCTTGTACGGAGTGGTGTACCAAGTAATAATTAGGGCGATTAG
CACCTCATCGCGAACCCCCGATTAGTCACTGGATGGGTTGCGATTGCGTCTAATATA
GATTTAGCACATTAGATCGCTTCGCATCGACAGGCAATTAGATATAAATAGTTCGCG
AATGAGGCGGCCTTTCCACATAACGAACTGTGTTTCAACTAAAGCACATCTTAGGGT
ACGATACGTGAGGAAAGTGAGAGGGTTAAACATCCTAGCGATAACTTACCTACTGAG
ACGTAAGCACACTCGCACCATAAGCAAATCGGGAAATCAGCGTACTCTCCTCATTCC
ATGAGACACCCATATCGCCAATGTCGCGTCTCTCTTCCCATACCATAGACCAGCGGC
TGGCAGCTCCGAGTCCACTCACCTCGGGTTGTTTACCCGAGCCTCCAGGGATAACTG
CGTATTCCAACAATCCTGGCCCGTACCTAGGATAGTAAGTCAGGAACGCCGATCCTG
AGTAAATACCAATCTGATCAACCTGCCCCGGAGTCTCGGTCGGGTTCCGAAAAGGCC
GAGACTTCGAATTTCAGTGTATGTGGAGATCCCGGAGATAGAGTGTCGATCAGTAGT
TGCCAGCCTGCCAAAAAAAAAATTTTTTTTTTGGGGCCTATGGCCACTTCGAACCCT
CGGATCGGACACGGCCGAGATACGGACAGTAGA
```

```
zain@zain-HP-EliteBook-840-G1:~/Desktop/ScriptProgramming
/MainProject$ grep -o "AAAAAAAAAATTTTTTTTTT"
genome_01.dat
AAAAAAAAAATTTTTTTTTT
AAAAAAAAAATTTTTTTTTT
AAAAAAAAAATTTTTTTTTT
AAAAAAAAAATTTTTTTTTT
.........................
AAAAAAAAAATTTTTTTTTT
AAAAAAAAAATTTTTTTTTT
```

```
zain@zain-HP-EliteBook-840-G1:~/Desktop/ScriptProgramming
/MainProject$ grep -o "AAAAAAAAAATTTTTTTTTT"
genome_01.dat | wc -l
99
```

5) Searching the string with the grep

```
zain@zain-HP-EliteBook-840-G1:~/Desktop/ScriptProgramming
/MainProject$ grep -Eo "A{10}T{10}" genome_01.dat | wc -l
99
```

6) Saved in the bash file (part1.sh)

Part 2 :

Python

The results of the first 10 Task in python

```
zain@zain-HP-EliteBook-840-G1:~/Desktop/ScriptProgramming/MainProj
ect$ python3 project.py
The first base is: A
Task 1: The first base by: A
Task 2: Number of basses of sequence in Task1: 8
Task 3: DNA sequence of Task1: True
Task 4: Compliment of Task1 sequence AAGTAATA is: TTcATTAT
Task 5: Pair of non-matching bases: 1
Task 6: Total number of characters in "genome_01" is: 100000
Task 7: Length of first genes is: 639
Task 7: Numer of DNA sequence "genome_01.dat" file: 100
Task 9: Total Number of DNA sequence "genome_02.dat" file: 272
Task 10: test_size=0.5 Intercept of predicted line : [0.64475409]
Task 10: test_size=0.5 The slope of line is : [[0.0034713]]
Task 10: test_size=0.7 Intercept of predicted line : [[0.0034713]]
Task 10: test_size=0.7 The slope of line is : [[0.0034713]]
```

Task 1 :

For this task created another instance “first_base” and returning the first index via self.

Task 2 :

In this task using the “ bases_in_DNA” instance for finding the bases in different sequences by getting each index of the sequence and getting a count of bases.

Task 3:

In this task using the instance “bool_DNA” simply putting an if-else condition on (A, T, C, G) to find if the sequence is DNA or not.

Task 4:

In this task using the instance “compliment_of_sequence” again simply putting an if-else condition on (A, T, C, G) and replace it with them accordingly and returning the updated sequence.

Task 5:

In this task using the instance “matching_sequence” getting the one sequence one class is initialized and the second in the argument of this instance. First comparing the sequence length and then comparing them index-wise to if they are equal or not.

Task 6:

In this task writing a method “read_genome_file” in the project.py to read the file and then using the task 2 instance “bases_in_DNA” to find the number of bases in genome files.

Task 7:

In this task using a splitter sequence “AAAAAAAAAATTTTTTTTTT” which most common in genome files according to task 4 of part 1. Finding the length of genes with this sequence and the number of sequences that are different from splitters in genome files.

Task 8:

In this task reading a sequence of genome_01.dat containing a single gene sequence.

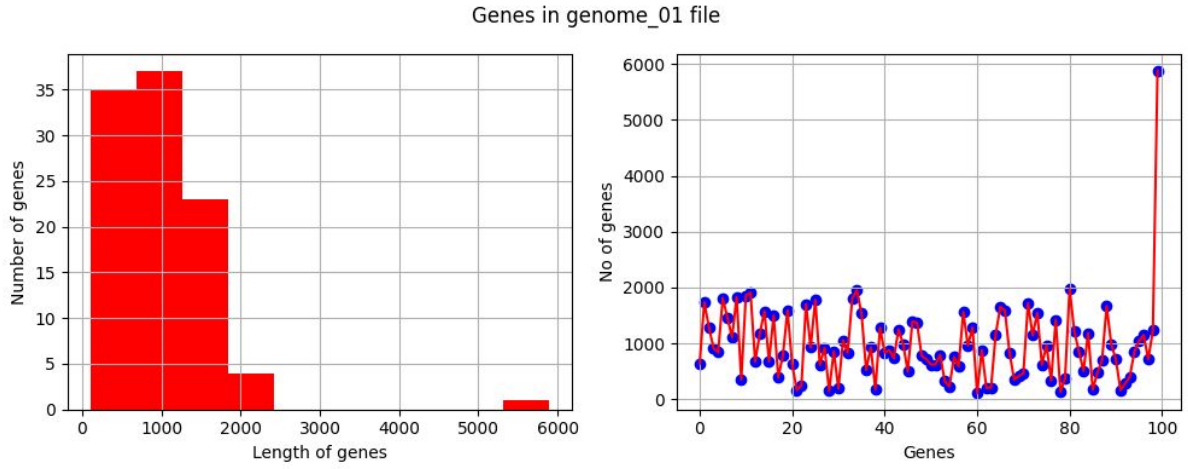
In this task reading the file then splitting it with the sequence and with a for loop reading the length of the different bases in each split genome.

Task 2: “bases_in_DNA ” to read bases
Task 6: “read_genome_file” to read the file
Task 7: “split_sequence” to split

The block in each histogram on the right shows the number of genes of different sequences in the y-axis number of genes and on the x-axis length of genes.

The left plot shows the scatter plot to represent data in a different way having a number in a total of 100 genes.

The histogram is a much better representation it also shows the data which is not having any genes.



Task 9:

In this task, we have to find a number of bases of swapped mutation.

Used the scatter plot which compares both files **“genome_01.dat”** and **“genome_02.dat”** compares them sequence by sequence and shows mutation data in blue dots.

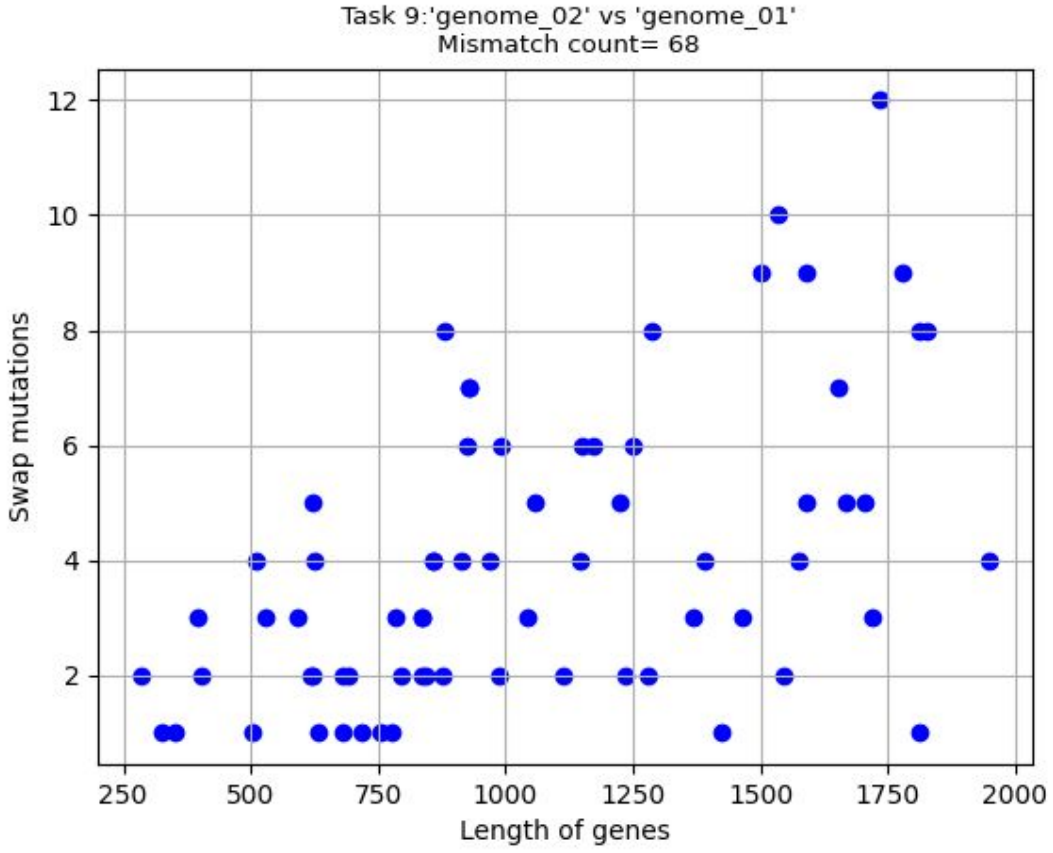
In this task, it reads the files then split the sequence for comparing both files sequence in nested for loop and return the difference data (swap mutation), genes length, count of swap mutation bases, number of different basses.

Task 5: “matching_sequence” to compare

Task 6: “read_genome_file” to read the file

Task 7: “split_sequence” to split

In x.axis showing genes length and the y-axis showing the mutation in a scatter plot.



Task 10:

In this task using linear regression to find the line best fit from data in task 9.

For this task import sklearn for training the data, implementing the linear regression model, and use matrices. For training the data used train_test_split() to find the train and test data of genes length and swap mutation.
Then on train and test data using .fit, .predict, .intercept, .coef method of LinearRegression() to find the best fit line.

The graphs below are showing different results based on different test_size and their difference by RMSD(root mean squared difference).