Assignment 4: Colour Blindness

Author : Aksheit Saxena

Introduction

In this report, I will be using the data provided to us to determine the most likely configuration of the red and green genes that leads to colour-blindness. I will do this by aligning the reads to the reference sequence with up to two mismatches, and then counting reads mapping to exons of the red and green genes. I will then determine the probability of generating these counts given each of the possible red-green gene configurations.

Data

The data that I have been provided with includes:

- Reads from a genome – 3 million reads of the 150m I actually generated
- The reference sequence of chromosome X – 150m instead of 3b for the whole genome
- The BWT last column and the pointers back to the reference for chromosome X
- The locations of the exons of the red and green genes in chromosome X

1. chrX.fa: This file contains the reference sequence for Chr X in FASTA format. The first line is a header line, and the subsequent lines contain the sequence without any newlines. The reference sequence represents the original DNA sequence of the Chr X chromosome.

2. chrX_last_col.txt: This file contains the last column of the Burrows-Wheeler Transform (BWT) for Chr X. The BWT is a reversible transformation used for data compression and indexing in DNA sequencing. The sequence in this file is one character longer than the reference sequence as it includes a special character, "$," which is appended to the end.

3. chrX_map.txt: This file provides the mapping betIen the indexes in the BWT and the corresponding indexes in the reference sequence. Each line in the file represents the starting index in the reference sequence for the ith sorted circular shift in the BWT. For example, the first line contains "3435," indicating that the string starting at index 3435 (0-based) is the first in the sorted order of the BWT rotated strings.

4. reads: This folder contains approximately 3 million reads, with one read per line. The reads are of variable length, roughly around 100 characters. It's important to

note that each read can originate from either the reference sequence or its reverse complement. Therefore, the analysis should consider the reverse complements of each read as Ill.

5. Red and Green gene locations: The locations of the Red and Green genes within the Chr X chromosome are provided. Both genes start with the sequence "ATGGCCCAGCAGTGGAGCCTC." The starting positions (0-based) for the Red exons are as follows:
- 149249757 - 149249868
- 149256127 - 149256423
- 149258412 - 149258580
- 149260048 - 149260213
- 149261768 - 149262007
- 149264290 - 149264400

The starting positions (0-based) for the Green exons are as follows:
- 149288166 - 149288277
- 149293258 - 149293554
- 149295542 - 149295710
- 149297178 - 149297343
- 149298898 - 149299137
- 149301420 - 149301530

To demonstrate the functionality of the code, I will use the following read for testing:

GAGGACAGCACCCAGTCCAGCATCTTCACCTACACCAACAGCAACTCCACCAG
AGGTGAGCCAGCAGGCCCGTGGAGGCTGGGTGGCTGCACTGGGGGCCA

Methodology

I will use the following methodology to determine the most likely configuration of the red and green genes that leads to color-blindness:

1. I will align the reads to the reference sequence with up to two mismatches.
2. I will count reads mapping to exons of the red and green genes.
3. I will determine the probability of generating these counts given each of the possible red-green gene configurations.
4. I will then select the configuration with the highest probability as the most likely configuration that leads to color-blindness.

Implementation

1. The code starts by importing the necessary libraries and modules, such as `pandas`, `torch`, `numpy`, `time`, and `math`. It also imports some specific functions from the `google.colab` module.

2. Several functions are defined:
    - `last_col_(path)`: Reads the last column data from a file and returns it as a string.
    - `mapping_(path)`: Reads the map data from a file and returns it as an array.
    - `refer_(path)`: Reads the reference sequence from a file and returns it as a string.
    - `reads_(path)`: Reads the reads from a file and returns them as an array.
    - `calcrank(ch, i)`: Calculates the rank query based on a delta value.
    - `band_(idx, rank)`: Calculates the band value based on an index and rank.
    - `get_first_column_data(last_column_data)`: Extracts the first column data from the last column data and returns the counts of each character.
    - `count_(lst_col)`: Counts the occurrences of each character in the last column data and returns arrays for each character and their cumulative counts.
    - `match_idx_(txt)`: Finds the band indices for a given string.
    - `exon_match_counts(start, end, read_length)`: Calculates the counts of matches to red and green genes based on start and end indices and read length.
    - `is_green_(idx, length)`: Checks if an index belongs to a green gene and returns the corresponding exon number.
    - `is_red_(index, length)`: Checks if an index belongs to a red gene and returns the corresponding exon number.
    - `match_ref_idx(start_index, end_index)`: Finds the reference sequence indices that match a given range.
    - `mismatch_count(index, read)`: Counts the number of mismatches betlen a reference sequence and a read.
    - `permut(n, r)`: Calculates the permutation of n and r.

3. Data is read from various files, including the last column data, map data, reference sequence, and reads.

4. The code defines the ranges for red and green genes based on their start and end indices.

5. The first column data is extracted from the last column data, and the counts of each character are obtained.

6. The `count_` function is called to count the occurrences of each character in the last column data.

7. Empty lists `red_exon` and `green_exon` are initialized.

8. The code selects a subset of reads and preprocesses them by replacing 'N' with 'A' and finding the reverse complement of each read.

9. The `match_idx_` function is called to find the band indices for a test string.

10. Exact matching is performed for each read, and the counts of matches to red and green genes are updated accordingly.

11. The code calculates the length of the reference sequence.

12. For each read, the read is split into three parts, and band indices are calculated for each part.

13. Matches to the reference sequence are found for each part, and the counts of matches to red and green genes are updated.

14. The code outputs the counts of exact matches for the red and green genes.

15. The elapsed time of the code execution is printed.

Results

The results of our analysis are shown in the table below.

```
Length of the map data: 151100561
{'A': 45648952, 'C': 29813353, 'G': 29865831, 'T':
45772424, '$': 1}
Found $ at position: 65402173
Band first match index: 81213340
Band last match index: 81213343
```

```
Map value for test string: 149249814
Counts for exact match of red gene
=  [67.5, 56.5, 50.0, 105.5, 216.5, 179.0]
Counts for exact match of green gene
G_c =  [67.5, 180.5, 58.0, 100.5, 269.5, 179.0]
Count of red exons considering up to two mismatches
G_c =  [88.5, 192.0, 100.0, 138.0, 284.0, 216.5]
Count of green exons considering up to two mismatches
G_c =  [88.5, 253.0, 150.0, 129.0, 342.0, 216.5]
Time taken :: 1084.594288110733
Total matches :: 2198
Log probability for config-1 :: -78.79276315204297
Log probability for config-3 :: -63.76141210622356
Log probability for config-4 :: -107.11072407754281

Probability for config-1 :: 6.035840421394026e-35
Probability for config-3 :: 2.035966200484613e-28

Probability for config-4 :: 3.03671182397752e-47
```