Data Analytics Assignment - 6

G Ramesh Babu
SR No. 20950

Colour Blindness

Abstract

This report presents the analysis conducted on color blindness as a component of Data Analytics Assignment-6.

1 Implementation Summary

A class `GeneticAnalysis` is built and used for the analysis. Here are the methods used in it

1.1 Importing Data

The method `load_data()` imports the given data from the specified directory and does the following.

- Each file that we use it on is converted into a list of lines and stores in `self.indices, self.ref, self.bwt`
- Occurrences of the nucleotides `A, C, G, T` in every line are noted and based on that their Ranks are calculated and stored in `self.Ranks`, a dictionary which has the values as the list of Ranks with each of those characters as keys

1.2 Pattern Matching and Mismatch Tracking

The `Match()` method is responsible for comparing two DNA sequences (`str1` and `str2`) to identify if they match. It counts the number of mismatches, and if the count does not exceed a threshold of 3 and the lengths of the sequences are the same, it returns True. The function checks for pattern matching and mismatch counts within the sequences.

1.3 Pattern Search and Band Updating

The `search()` method is designed to search for patterns in a given DNA sequence, specifically in reverse order. It maintains a search band, which defines the range in which the pattern is sought. It iterates through the sequence, calculating the first and last ranks of a character within the search band. The function updates the search band based on the character's properties and returns the final search band and the number of mismatches found within the pattern.

1.4 Pattern Extraction

The `extractfromref()` function extracts a substring of a specified length from a reference DNA sequence based on an index. This function likely plays a role in comparing and matching patterns within the DNA sequences.

1.5 Exon Matching and Updating

The `findRedGreenMatches()` function searches for specific patterns in the DNA sequences and updates two sets of matches: "RedMatches" and "GreenMatches." It utilizes the search results to identify matches within predefined exon ranges, and it appears to be an essential component for genetic analysis, potentially used in identifying genetic patterns or variations within the sequences

1.6 Read Processing and Exon Count Updating

The `process_reads()` function processes DNA sequences from a file, calling the `findRedGreenMatches()` function for each read. It then updates exon matches based on the matches found in the sequences, maintaining counts for "RExons" and "GExons."

1.7 Determining Best Configuration

The `possible_config()` function serves the purpose of determining the best configuration based on the division of values from RExons and GExons. It calculates the cosine similarity between the division result and a set of 4 configurations provided as input. This comparison helps identify which configuration is most similar to the division result. The function returns a softmax of the similarity scores, providing a probabilistic representation of how well each configuration matches the observed division result.

2 Results

2.1 No. of Matches

:

Exon Number	Red Exons	Green Exons
Exon 1	90.5	90.5
Exon 2	78.0	228.0
Exon 3	71.0	125.0
Exon 4	146.0	127.0
Exon 5	261.5	327.5
Exon 6	222.0	222.0

Table 1: Red Exons and Green Exons

2.2 Most suitable Configuration

Configuration	Description	Score	Best?
C1	[0.5, 0.5, 0.5, 0.5]	0.2804	
C2	[1, 1, 0, 0]	0.1689	
C3	[0.33, 0.33, 1, 1]	0.2954	Yes
C4	[0.33, 0.33, 0.33, 1]	0.2553	

Table 2: Scores of Configurations

In conclusion, after a comprehensive analysis of DNA sequencing data, it has been established that the most probable configuration associated with color blindness is Configuration 3 (C3) characterized by the values [0.33, 0.33, 1, 1]. This specific configuration exhibits the highest probability among the provided options and holds substantial significance in the broader context of comprehending the genetic basis of color blindness.