Project 2: Global Alignment Scoring for Population Identification

Dan Blanchette, Taylor Martin, and Jordan Reed

CS415: Computational Biology, University of Idaho

April 1, 2023

Abstract

In the modern world, DNA sequencing has led to many important discoveries in stem cell research and forensic science. This type of research is difficult and time-consuming without a means to measure how related two DNA sequences are to one another. In this project, we use a genetic algorithm to generate individuals based on a high, low, and randomized method(high and low) for mutation. Each codon A, T, G, and C have individual scores comprising each individual's average fitness. The individuals are separated into a corresponding group from which they are generated. Afterward, they are sorted into one of three populations. Using a global alignment scoring algorithm, we strive to identify which individual belongs to a specific population. Our results indicate difficulties in correctly identifying an individual's group above 80% certainty. This is due to the population generated using the random multiple mutation rate method. Due to this issue, the global alignment scores can be repeated even though an individual has a unique codon composition.

1 Introduction

For this project, we generated three populations based on genetic crossover and at varying mutation rates. Population one was generated using a high mutation rate set to 80%, Population two was set to 20% mutation rate, and population three oscillates between the two. We chose to do this to add a degree of complexity to our global alignment scoring algorithm to identify which population selected individuals belonged. Our goal was to create a global alignment algorithm that would correctly identify which population an individual belongs to based on its alignment score and codon composition.

2 Genetic Algorithm

2.1 Individuals

Each individual comprises a genome sequence containing 50 nucleic acids, i.e. 'TCGA'. The genome sequence was chosen at random from the available nucleic acids.

2.2 Fitness Function

The fitness function is simple: it looks at a single character of the genome sequence at a time. If the character is a 'T,' one point is awarded. If the character is an 'A,' 2 points are awarded. If the character is a 'G,' 3 points are awarded, and 4 points are awarded to a 'C.' This means that the maximum fitness score for an individual, based on our genome size of 50, is 200.

2.3 Single Mutation Rate Algorithm

An individual is given a certain rate at which it can mutate. This rate is defined by the user and is described in more detail below. This mutation rate represents the chance an individual character from the genome sequence has to change into a different nucleic acid. The mutation function for an individual will walk down the genome sequence and pick a random number. If the random number is lower than the mutation rate, that position gets a new character (also chosen randomly). It is important to note that when mutating, a character has a slight chance of mutating into the same character.

2.4 Multiple Mutation Rate Algorithm

An individual can also mutate using multiple mutation rates. This algorithm is very similar to the Single Mutation Rate Algorithm above, with one key difference: There is a 25% chance that the mutation rate will switch between the low and high rates (defined by the user).

2.5 Populations and the Generational Model

A population is comprised of 50 individuals, created at random.

A single generation of a population is loosely modeled after biological generations. Two of the best individuals are selected (selection process defined below) and undergo crossover (also described below). The individuals are then mutated and placed into the new generation. This is repeated until we again reach a population size of 50. The goal is to allow the most desirable characteristics to be selected and passed on to the offspring.

2.6 Selection and Crossover

The selection process we used in our algorithm is known as tournament selection. Essentially, 5 individuals are picked at random and the individual with the best fitness score is kept.

The crossover process is the intermixing of two individuals in the population. The one-point crossover process we used walked down the length of the genome sequence and had a 10% chance to swap the characters between the two individuals.

3 Data Collected

We started with the same population of 50 individuals, copied 3 times and applied different mutation rates. The first population had a low mutation rate of 2%. The second population had a very high mutation rate of 80%. The third population had a mix of mutation rates. It would randomly oscillate between 5% and 85% with a 25% chance of oscillation.

We ran the generational model for 20 generations, then randomly chose between each population to fill a fourth population. We stored these individuals in a file and which people they came from in a different file.

The population with the low mutation rate comprised individuals mostly made up of C's, as expected. We found in the last assignment that a low mutation rate is optimal for obtaining the

best fitness. As our fitness function prioritizes the character C over the rest, seeing individuals with mostly C's was expected. The average fitness for this population was around 176.

The second population was the high mutation rate population. This population had a much lower average fitness and comprised somewhat random characters in the genome sequence. The average fitness of this population was about 129.

As expected, the third population was a mix of the previous two populations. Its average fitness was slightly better (sitting at about 133). This is because there were some points during the mutation process when it was allowed to mutate at a lower rate rather than at a higher rate for the entire time.

4 Alignment Algorithm

Our experiment used the Needleman-Wunsch global alignment algorithm and the BLOSUM50 scoring matrix, which we applied to the generated individuals. Each comparison evaluated two individuals and generated a table and a score for the optimal alignment. This approach provided a necessary framework to analyze the scores and predictions based on which individual belonged to which population.

4.1 Analyzing the Alignment Scores

We broke up the analyzing process into three key parts corresponding to our population groups. The individuals from the population with the lowest mutation rate would have more scores aligned with it than any others. It seemed reasonable to assume that the higher mutation rate population would have the lowest scores and the multi-rate population would have scored between the two. The following process is how we set up the algorithm:

First, we looked for the lowest mutation rate individuals. For each individual, if the alignment score for another individual is within 60% to 100% of the maximum, we placed it in group 1, our guess for the lowest mutation rate individuals.

Next, we looked at the other extreme. After discounting the individuals from group 1, the rest of the alignment scores were all fairly similar to each other. So we looked at the average of the scores, and if the average was less than 35% of the maximum, we placed it in group 2, our guess for the highest mutation rate individuals.

Every other individual was placed into group 3, our guess for the multi-rate individuals.

4.2 calculate_accuracy()

This function reads in our answer key output file and the guess output file for population comparison. If the inputs match, the tally for a correct answer is incremented. Conversely, the wrong tally is incremented if the guess does not match the answer key file's output. Based on the data analysis, we calculate the mean value for the correct guesses and divide that by the total guess outcomes(both right and wrong) to determine the accuracy.

```
def calculate_accuracy(group1, group2, group3):

    ansDF = pd.read_csv('answer_key.csv', usecols=[3])

    right = 0
    wrong = 0
    for i in group1: #pop0
        if ansDF['Population'][i] == 'pop0':
            right += 1
        else:
            wrong += 1
```

```
    print(f'right {right} wrong {wrong} total: {right+wrong}')

    for i in group2: #pop1
        if ansDF['Population'][i] == 'pop1':
            right += 1
        else:
            wrong += 1
    print(f'right {right} wrong {wrong} total: {right+wrong}')

    for i in group3: #pop2
        if ansDF['Population'][i] == 'pop2':
            right += 1
        else:
            wrong += 1
    print(f'right {right} wrong {wrong} total: {right+wrong}')
    print(f'accuracy: {right/(right+wrong)}')
```

5 Results

In doing the alignment algorithm, we found that our random population sample fell into some distinct groups (almost as expected). The first set of individuals that was the easiest to identify were those from the lower mutation rate population. This is because these individuals had the optimal mutation rate to 'evolve' to the best genetic sequence according to the fitness function. This means they had a higher occurrence of 'C' characters. How the global alignment algorithm works with the BLOSUM50 matrix assigned high scores to individuals with more 'C's. These individuals also had scores that were closer together.

You can see in the following image (Figure 1) how some individuals score above 500. These also had more individuals aligned more closely, which you can see in the figure. These are the individuals from the low mutation rate population.

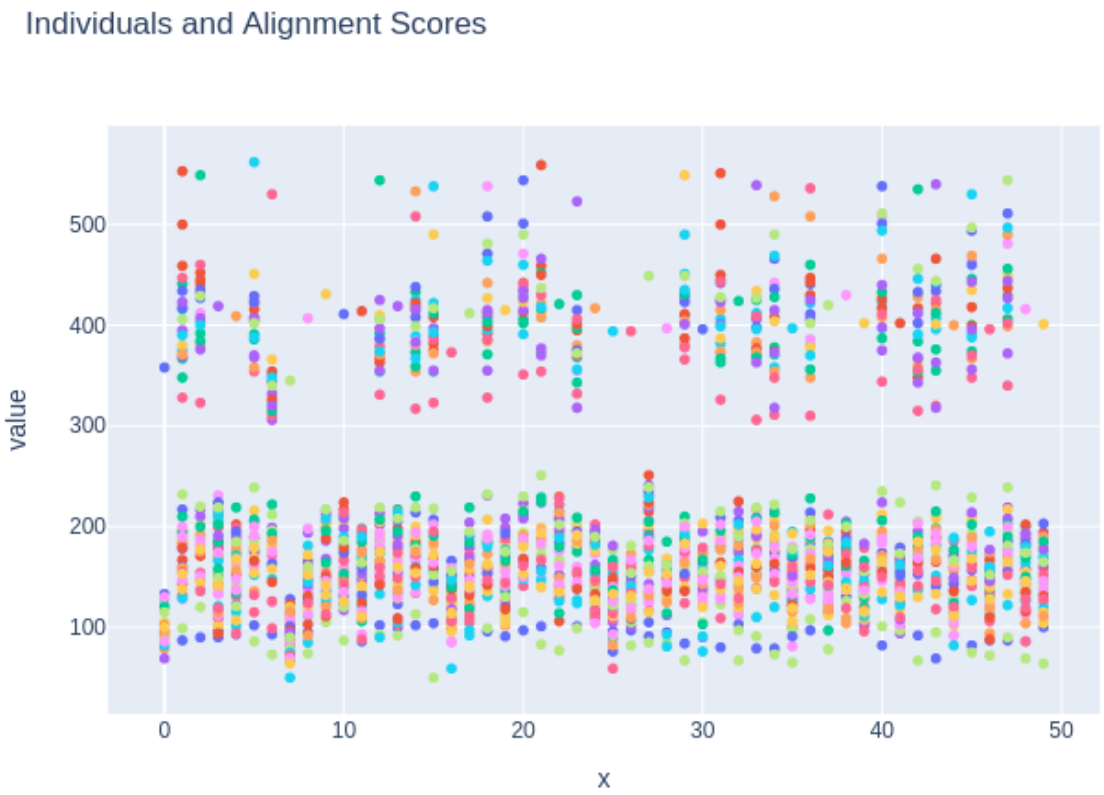

Figure 1: All Alignment scores

The individuals from the other two populations were harder to tell apart. The higher mutation rate means that the individuals' genome sequence could not 'evolve' to their best fitness score. They were mutating into random characters much more often. The population with a higher mutation rate had some pretty low fitness scores.

The population with a mixed mutation rate had points where they could try and achieve the optimal genome sequence, but it was interrupted with times of high mutation. This translates to a higher average fitness than the high mutation rate population, but not by much. For this reason, these two groups were harder to tell apart. The following figure (Figure 2) shows these two populations and their alignment scores.

What we did was we took the average of the alignment scores, and if the average was below 35% of the maximum for that individual, we placed that individual in the high mutation rate

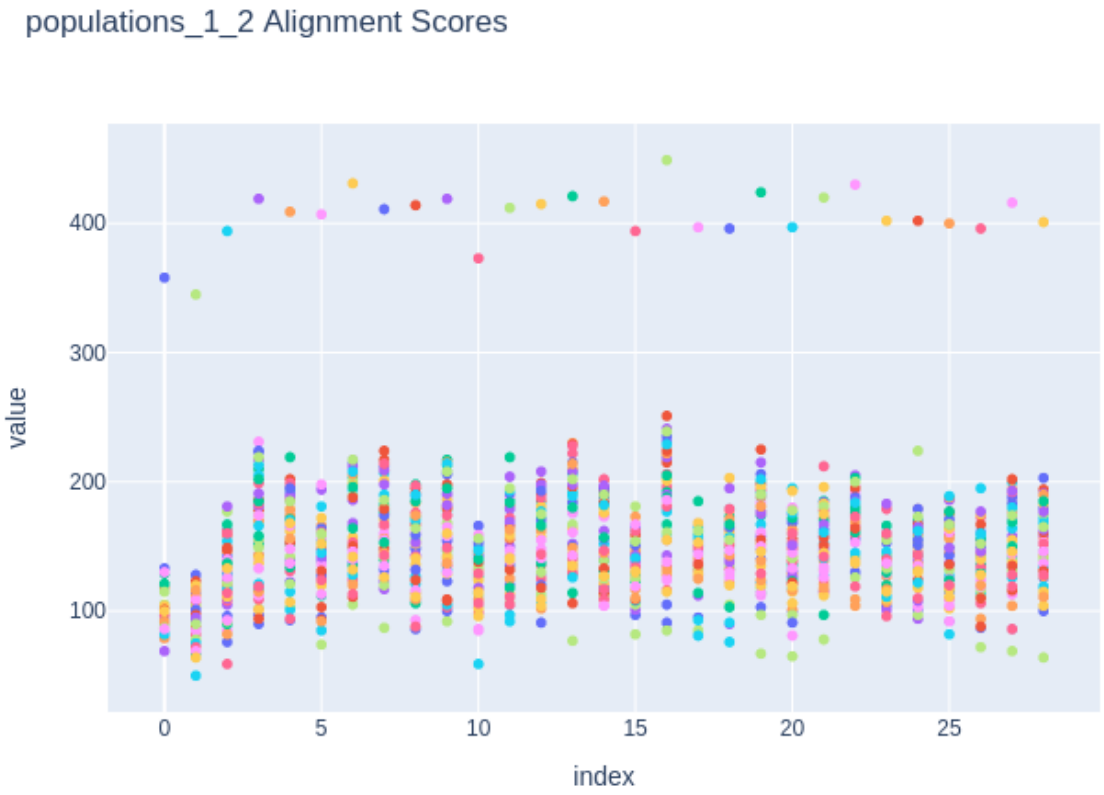

Figure 2: All Alignment scores

group. Every other individual who wasn't accounted for went into the mixed rate group. It is important to note that this method only found 3 individuals for the high mutation rate group. This suggests that our current method could be improved because it's not catching all the individuals.

We were able to identify the individuals with an 80% accuracy rate. All of our population 0 and 1 guesses were correct, but our population 2 guesses were right about 60% of the time.

6 Discussion/Conclusion

From our results, we can see the effectiveness and helpfulness of the global alignment algorithm. With further refining and testing, it is possible the alignment algorithm could correctly identify which population our generated individuals originated from. The difficulty with this algorithm comes from individuals with seemingly random characters in their genome sequence (originating from high mutation rates). A local alignment could have performed slightly better because we could have compared portions of the individuals together.

It was interesting to see the average fitness of a population with a mutation rate that oscillates between high and low. However, it may not be biologically accurate. It might be more interesting (and easier to experiment with) sections of the code that might oscillate between high and low rates.

Another way to improve the accuracy of this experiment could be with machine learning. Machine learning could be used to train a model to accurately identify an individual. This could be done using the same alignment algorithm, or a different alignment algorithm.