COMP90016 – Assignment 1

Haonan Li

April 9, 2018

1 Introduction

This assignment consists of three tasks.

For the first task, we first build a k-mer index for a reference file, then aligned all reads from a FASTQ reads file using the index.

The second task is to investigate the runtime of the aligner build in task 1. In this task, we compare the speed of the new aligner with the naive aligner proposed in group assignment. In addition, we also evaluate the relation between size of k-mer and the alignment speed.

For the last task, we extend our algorithm to further investigate mismatches to the reference.

2 Results & Discussion

2.1 Task 2

k	0	4	8	13	16	32	50
ref1	24.43	18.98	1.76	1.48	1.40	1.04	0.51
ref2	50.87	35.34	1.83	1.49	1.49	1.14	0.66
ref3	80.54	52.65	1.91	1.56	1.47	1.22	0.78

Table 1: Time consuming for k-mer indexed alignment. (k=0, refers to the naive aligner.)

Table 1 and Figure 1 show the time consuming for k-mer aligner work on the given datasets. $K = 0$ refers to the naive aligner proposed in group assignment. From these results, we find that new aligner is much more efficiently than the naive alignments. As the increase of k, the speed of alignment increase. For a small k ($k < 6$), the change is obvious. But if k is large ($k > 8$), the acceleration become less obviously.

The result is not surprised. Because the complexity of naive Hamming distance aligner is $O(ngl)$, where n is the number of reads, g the length of the genome, and l the length of the reads. But for indexed aligner, the complexity is $\mathrm{O}(nl^2 + g)$, here are theoretically analysis.

First assuming the length of k-mer is k, where $k \ll l$, then for each read, we have $l - k$ k-mers and suppose for each k-mer, the number of indexed positions is p. So we need to compre $O((l - k) * l * p)$ times for each read. As $k \ll l$,

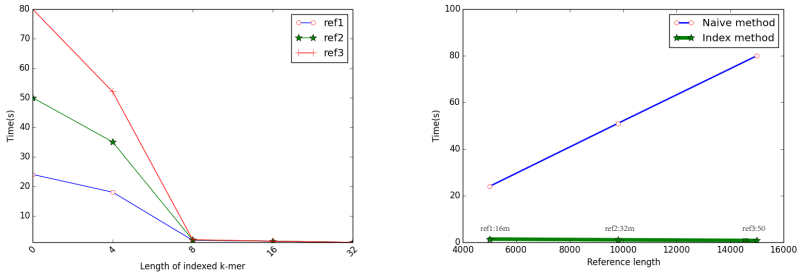


Figure 1: Time consuming for k-mer indexed alignment. (k=0, refers to the naive aligner.

$O(l-k) = O(l)$. Here, we have a complexity of $O(nl^2p)$. We normally assume that we can build a linear spaced index with constant number of positions for each k-mer,[1] and length of genome maybe very large, so we need to traverse it to build the index. Finally, the complexity is $O(nl^2 + g)$.

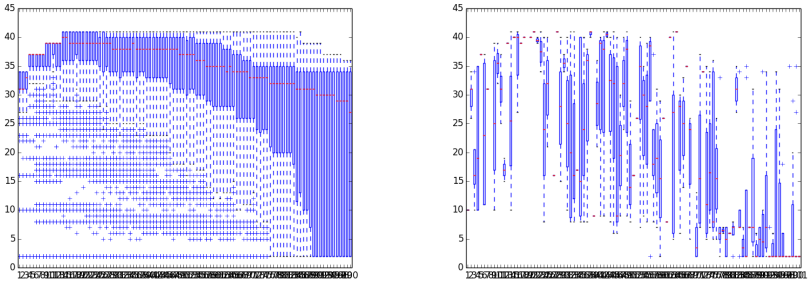
2.2 Task 3

Figure 2(a) shows the distribution of base quality scores of all the reads for each position, and Figure 2(b) the distributions of quality scores of all mismatches for best alignment to the given reference ref1.fa (also by position).

Compare these two images, we find that the range of each position's quality score of two pictures are similar, which indicates even a position has a high average quality score or most quality scores of this position is high, mismatches occurred. In addition, both of them shows that the middle position's quality score commonly higher than the start and end of the reads, this might because of the sequencing method.

However, no matter the upper quartile, lower quartile and median, two pictures show some differences. For most positions, the box of upper quartile to lower quartile of mismatched picture always lower than that for all reads. This might due to more low quality scores appear in the mismatched set. Algorithm there are exceptions that median for mismatches upper than it for all reads, which might because the dataset is small. Besides, as the position varies form 1 to 100, the average quality score becomes lower, mismatches increase. This was reflected by the median lines that get closer to the bottom and even overlapped with it as the positions near to 100.

[1] Actually, if k is large enough, p will equals 1



(a) All the reads for each position

(b) All mismatches

Figure 2: Distribution of base quality scores.