國立成功大學
資訊工程研究所
碩士論文

中文論文標題

English Thesis Title

研究生：蘇文弘　　Student : Wen Hong Su
指導教授：賀保羅　　Advisor : Paul Horton

Institute of Computer Science and Information Engineering,
National Cheng Kung University,
Tainan, Taiwan, R.O.C.
Thesis for Master of Science Degree
July, 2021

中華民國一一十年七月

中文論文標題

蘇文弘* 賀保羅†

國立成功大學資訊工程學系

摘要

近年來由於次世代定序技術(NGS)的出現，使得我們可以同時對大量DNA進行定序，大幅降低定序成本，許多相關的研究也跟著大幅擴展，NGS數據中變異調用幾乎是所有下游分析和解釋過程都依賴的關鍵步驟，但是對於準確基因變異體偵測仍然有許多問題及挑戰。因此我們先前提出過一個工具Explicit Alternative Genome Likelihood Evaluator(EAGLE)，透過使用顯示概率模型，處理變異調用過程中固有的不確定性。明確評估測序數據與推定變異所隱含的替代基因組序列的匹配程度。但是不論是EAGLE的模型抑或是一般的變異調用方法，或多或少都仰賴將讀序映射到參考基因組序列上的結果。而映射的方法是透過將讀序與參考基因組序列比對，根據比對的相似程度決定他們可能來自參考基因組序列上的位置。

由於我們比對的目標參考基因組序列是線性的，僅包含單個序列，因此無法捕獲基因組的多樣性。所以比對會出現可能一定程度的誤差，造成我們讀序可能被映射到錯誤的位置或是無法映射，這種現象我們稱之參考偏差。錯誤的讀序映射會導致假陰性或假陽性變異調用，進一步影響到我們對於辨認基因體變異的準確性。而先前已經有許多研究討論消除參考偏差所帶來的影響，其中一種方法就是透過改進參考基因組序列，使其能夠包含基因組的多樣性，從而達到更好的映射結果。本文中我們提出一個類似概念的方法，並應用於我們先前提出的工具-EAGLE上，探討我們提出的方法對於降低參考偏差帶來的影響是否可行以及其潛力。

*學生

†指導教授

我們的方法可以分成五個部分，第一個部分受透過基因突變資料(VCF)中取得變異點資訊，結合參考基因組序列建立假設序列。第二個部分是爲讀序檔案(FASTQ)建立一個能夠快速搜尋的索引結構。第三個部分是透過Burrows-Wheeler Aligner(BWA)以及我們上一部份建立的讀序索引結構在讀序檔案中找出所有能夠與我們建立的假設序列匹配的讀序。第四個部分則是檢查原本的映射結果檔案(BAM)中的堆積序列，堆積序列是在映射結果中被映射到相同區域的所有讀序，檢查是否存在原有的堆積序列中，對那些不存在堆積序列中的讀序建立相關的比對資訊。第五部分則是將我們找到的讀序以及比對資訊加入堆積序列中，改進我們EAGLE的計算結果。

在實驗中，我們模擬參考偏差發生的情景，透過我們的方法驗證是否能找到那些受到參考偏差的影響而遺失的讀序，結果顯示我們能夠找到當中大部分被遺失的讀序，但是在一些重複序列過多的區域可能有所限制。我們同時也在 dbSNP的變異資料及上進行測試，基本上也能夠取得相似的結果，惟在單核酸多態性的變異上效果沒有達到理想的成效。但大體而言，綜合我們所增加的時間以及成效，我們認爲我們的方在在解決參考偏差的影響是相當有潛力的，但是應用到實際情況還是需要更進一步的驗證以及謹慎的解釋。

關鍵字： 次世代定序、變異點偵測、參考偏差

English Thesis Title

Wen Hong Su* Paul Horton†

Institute of Computer Science and Information Engineering
National Cheng Kung University

Abstract

In recent years, due to the emergence of next-generation sequencing technology (NGS), we can sequence a large amount of DNA at the same time, greatly reducing the cost of sequencing, and expanding its applications. Genome variant calling from NGS data is a key step for many applications in biology and medicine, but there are still many problems and challenges for accurate variant calling. Most variant calling procedures start with the so called "pile-up", an alignment of reads to the reference genome around the position of a putative variant. However this alignment cannot be expected to be certain; because the human genome is repetitive (many distinct regions are be similar to each other) and the reference genome sequence differs from any individual. Moreover alignment only to the reference genome naturally tends to favor the reference sequence over others (so called reference bias).

Reference bias is a widely recognized problem in variant calling. One research direction is to represent the reference genome with a graph instead of a simple string to include some common variants. In a different approach, we previously addressed reference bias by proposing a method (EAGLE: Explicit Alternative Gene Likelihood Evaluator) which locally aligns reads not only to the reference sequence, but also to putative variant sequences; using an explicit probability model to combine evidence

*Student
†Advisor

from all possible local alignments. The results were encouraging but fell short of completely eliminating reference bias, because that version of EAGLE still relies on read mapping to identify the reads which should be considered when evaluating a variant.

To completely eliminate reference bias in putative variant evaluation, any read similar to the variant sequence should be considered, even it does not match the reference sequence. Naïvely matching all reads to all variants would be far too slow, but in principle the search indexes used to accelerate standard read mapping could be adapted to search reads against a variant sequence query. But one might ask 1) if this would be feasible in practice (in terms of computer memory and time) and 2) if it would really make a difference in the variant calling results. Last year we conducted a feasibility study which suggested that constructing a search index on reads is feasible for whole exome sequencing and that searching such a read index can indeed find relevant reads not present in the "pile-up". That study however was only a feasibility study and did not integrate the read index with the EAGLE variant likelihood computation.

Here we explore the read index idea more thoroughly and integrate it into the EAGLE software. Our method can be divided into two preprocessing steps and then four steps for each putative variant. In the first pre-processing step we align the reads to the reference genome. In the second pre-processing step, we create an Burrows-Wheeler Aligner (BWA) index structure for the read sequence data (FASTQ format file) that can be quickly queried. Then for each variant in a list of putative variants (VCF format file), we first edit the reference genome to include the putative variant, then use that to query the read index structure, merge the matching reads with any reads mapping to the reference genome near the putative variant genome position, and

finally apply the EAGLE probabilistic model computation to the merged set of reads to obtain an unbiased likelihood ratio between variant and reference sequence for that position.

We perform some experiments to evaluate the method. In one experiment we use real sequence read data but doctor the reference genome to produce a false reference genome sequence, thus simulating a situation in which we know the position of differences between individual and reference. Then we map reads to this false genome sequence to obtain a corresponding false pile-up. We tested if our read index can find reads supporting the actual genome sequence, but missing in the false pile-up. The experimental results show that we can find most of the missing reads, but have some difficulty in region with too many repeat sequences. We also tested real putative variants from the dbSNP dataset, observing qualitatively similar results. We show that for putative indel variants, the extra reads found by the read index have a large effect on the EAGLE likelihood ratio (and in the correct direction in cases where we know what the answer should be). The likelihood ratio of Single Nucleotide Variants (SNV)s, on the other hand is not greatly affected, a reasonable outcome since standard read mapping can tolerate single mismatchs, largely obviating the need to use a special read index. More comprehensive benchmarking is left for future work.

Keywords: NGS、variant calling、reference bias

Acknowledgements

Add your acknowledgements here.

Wen Hong Su

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background

In recent years, High-throughput sequencing has become an essential part of Genomics, with its high-throughput, scalability and fast technology for a large number of short-reads, many related research follows.

Most of these NGS-based studies almost inevitably depend on the accurate variants detection and genotype calling of variants from sequence data [**?**], and due to the high similarities across genomes of the same species, accurate variant calling is a crucial analytic step that almost all downstream analysis and interpretation process rely on.

Just as NGS technologies, variant calling software and approaches have rapid development over the past 10 years. There have many variant caller and algorithm developed for variant calling, such as GATK [**?**], Strelka2 [**?**], Freebayes [**?**], DeepVariant [**?**] and etc.

Generally, typical variant calling process include sequencing, quality control, mapping, mark duplicates & sort & merge, followed by variant calling, filtering and annotation [**?**].

Although the variant calling pipelines are straightforward in theory, errors are prone to occur when detecting variants from NGS data due to multiple factors such as sequencing error, inaccurate alignment, and low read coverage. Some researches

have been published discussing using different variation calls pipelines [?] [?] and with different alignment tools [?] [?]. Research indicates that the results from different variation of the caller to agree only on a small part of the variation [?], and indels(insertion/deletion) are still difficult for any pipeline to handle with [AC15]. However, indels are supremely important as they are considered to have strong impact on cancer [?] and phenotype [?]. that is why we need to do accurate assessment of variants.

1.2 Motivation

We recently developed a variant evaluator - EAGLE [?], a method that uses generative probability models to evaluate the confidence of input VCF file containing a list of candidate variant (For brevity, hereafter we will simply call these "variants") in sequencing data supporting a given candidate genome, which can help us evaluate the accuracy of mutation points

However, whether it is these variant callers or EAGLE, they all rely on the pileup of alignment tools in the workflow. In variant calling, an intuitive method is to scan the pileup, check the alignment of the reads and look for the difference between the reference genome and the individual being sequenced. This method is indeed the essence of most variant calling methods and can achieve good results in many cases. But the premise is that this method completely relies on correctly mapping the variants containing the sequencing reads to their corresponding positions in the reference genome sequence.

Since the alignment tools conceptually maps reads to similar reference sequence regions, reads containing variants may not be mapped to the correct region. This effect

is called reference bias [**?**]. Reference bias is expected to be a source of false negatives (failure to call the true variant), especially for longer indel variants.

Although compared with tools that require a strong dependence on the pileup in the process of variant calling, EAGLE reduces the dependence on the pileup in the computation process through considering a large number of possible read across the genome. But inevitably, there is still a reference bias in theory, this will affect our accuracy in evaluating variant.

1.3 Research Objectives

Therefore, in this study, we hope to find out the reads that may be affected by reference bias and lost in the alignment process, and use these reads to improve the accuracy of EAGLE's evaluation of variants.

1.4 Research framework

The organization structure of this paper is as follows: Chapter One introduces the research background, motivation and research goals. In Chapter 2, we introduce the tool BWA and review our previous research on EAGLE and reference bias. In Chapter 3, we describe the workflow of combining our EAGLE and the method of our system. In Chapter 4, we conducted experiments on simulated data and real data respectively. In Chapter 5, we discuss the results, provide conclusions and future work.

Chapter 2

Related Works

In this chapter, we review the related work in recent years. In section 2.1, we introduce the Burrows-Wheeler Aligner (BWA), a well-known alignment tool. In section 2.2, we introduce our previous research EAGLE, a method for evaluating the degree to which sequencing data supports a given candidate genome variant. In section 2.3, we explain what is reference bias and related solutions.

2.1 Burrows-Wheeler Aligner

Burrows-Wheeler Aligner (BWA) [**?**] is a tool to align the sequences on selected large reference genome such as human. It consists of three algorithms: BWA-backtrack, BWA-SW [**?**] and BWA-MEM [**?**]. Among which BWA-MEM is the latest and generally recommended for high-quality queries as it is faster and more accuracy, basically replace the first two algorithm.

Based on our read data and some previous studies, we believe that BWA-MEM will show better performance [HKL15], and its algorithm is robust to sequencing errors and applicable to a wide range of sequence lengths, so we decided to use BWA-MEM as our main algorithm for finding reads

BWA-MEM first needs to build the FM-index [**?**] for the reference genome with the 'index' command, FM-index compressed substring index based on the Burrows-Wheeler Transform (BWT) [**?**]and suffix array, enables the backward search, generate a fast and efficient index for querying.

Then in the core alignment algorithm, BWA-MEM use seed-and-extend paradigm, and the process can be roughly divided into four steps. First step is seed, seed is found through super maximal exact matches (SMEM). Seconded step is re-seeding, in order to reduce the misalignment caused by missing seeds. Third step is chaining and filtering, chaining the seeds those are collinear and closed each other as a chain, and filter overlapped or short chain. Forth step seed extension, extending the seed in the chain to achieve the final exact match region.

2.2 EAGLE：Explicit Alternative Genome Likelihood Evaluator

EAGLE [**?**] is our previous research, based on generative probabilistic model, variant calling evaluator, EAGLE main function is giving individual sequencing data (exome or whole genome sequencing) and a list of candidate variants, will returns a likelihood score of each candidate variants, the concept of EAGLE is in the process of executing variant calling, there will be many uncertain factors, such as the error rate of alignment, low coverage sequencing and error rate of base-calling, etc., and these will affect downstream analysis. And EAGLE handles these possible errors and uncertainties through generative probabilistic model (Figure 2.1).

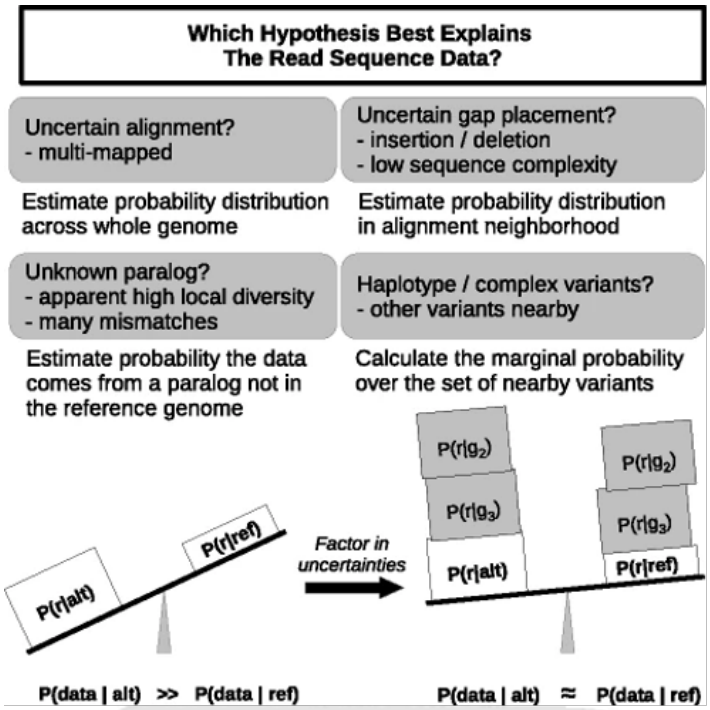

Figure 2.1: EAGLE use generative probabilistic model handle probabilistically.

Overall, the more uncertain the data the more uncertain the explanation, Thus, in principle the likelihood computed by EAGLE can be used rank the reliability of putative variants.

2.3 Reference Bias

Sequencing data analysis often begins with aligning reads to a reference genome, but this sequence only contains reference allele at polymorphic sites. [**?**], it will leads to reference bias (Figure 2.2): for reads that contain alternate alleles, tend to misalignment or incorrect alignments [**?**], and due to reference bias, at heterozygous site, the reads containing the reference allele constitute the majority of the reads, these situations can cause false negative or false positive variant calls. [**?**]

Figure 2.2: Reference bias. Only a small portion of the read containing inserts (dark blue) are mapped to the correct positions during the alignment to the reference genome.

There have some studies discuss how to mitigated reference bias, such as genome graph aligners [**?**] [**?**], or using high-coverage NGS data, and improve the reference genome so that it includes the genomic diversity of the entire population. In this paper, we modify the reference sequence to include alternate alleles, and try to find some matching reads to support this hypothetical sequence, then add these reads to the calculation of EAGLE to discuss the impact of this method on reducing reference bias and improve the accuracy of assessing variation.

Chapter 3

Method

In this chapter, we will introduce the method we implemented on EAGLE to reduce the reference bias.Explain in detail how we create our hypothetical sequence based on the variant and how to quickly search through FM-index to obtain those missing reads, and finally add these reads to the pileup that we eagle considers.

3.1 Overview

The red area in Figure 3.1 shows our implementation of the core workflow of reducing reference bias on EAGLE, and the whole Figure 3.1 is our complete workflow after combining EAGLE. We can roughly divide our workflow into the following five parts:

1. According variant in VCF files and reference genome to create our hypothetical sequence, this sequence is important because including individual differences.

2. Then we need to convert our FASTQ format read to FASTA format to build the FM-index of read, (For brevity, hereafter we will simply call these "read-index").

3. Use BWA and read-index to find the read that include variant, and then check if it in already exists in the pileup. (Multiple alignment of reads to the genome)

4. For read that does not exist in the pileup, create read's alignment information mapped against reference sequences

5. Add read and alignment information in the pileup, improve the eagle calculate.

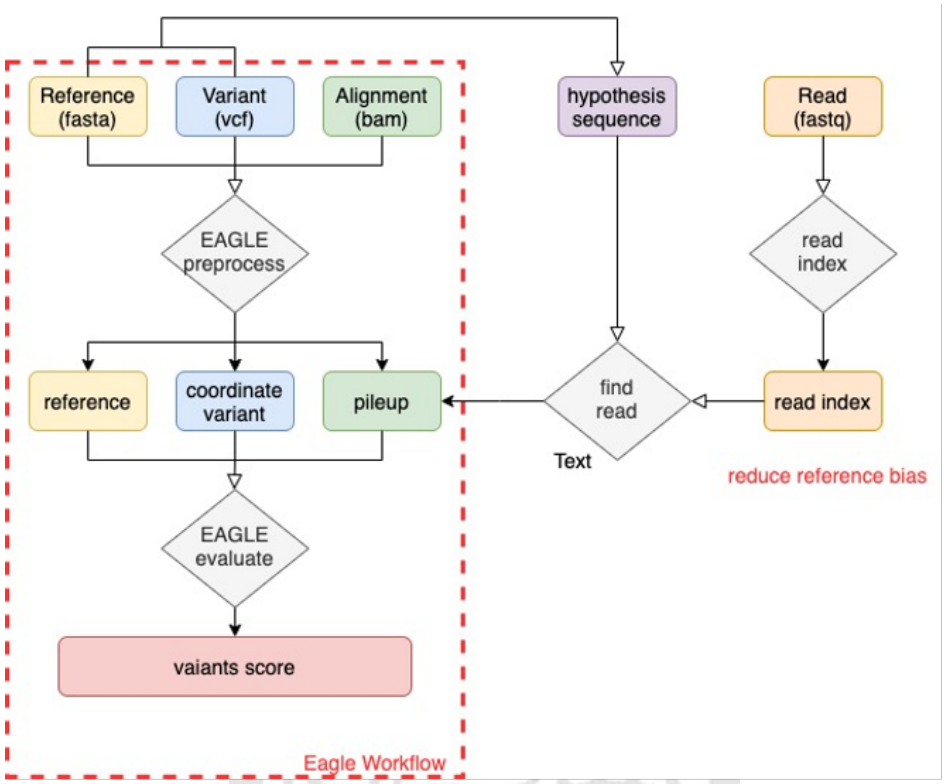

Figure 3.1: The core workflow after adding reduce reference bias method on EAGLE.

3.2 Hypothetical Sequence

To reduce the impact of reference bias, the first step we need to construct hypothetical sequence, before construct hypothetical sequence we need VCF file and FASTA file, VCF is the standard file format for storing variation data, used by most variant callers (Figure 3.2), and FASTA is reference genome format (Figure 3.3), it is a text-based format for representing either nucleotide sequences or peptide sequences.

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS      ID         REF   ALT    QUAL  FILTER   INFO                              FORMAT      NA00001        NA00002          NA00003
20     14370    rs6054257  G     A      29    PASS     NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ  0|0:48:1:51,51  1|0:48:8:51,51   1/1:43:5:.,.
20     17330    .          T     A      3     q10      NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ  0|0:49:3:58,50  0|1:3:5:65,3     0/0:41:3
20     1110696  rs6040355  A     G,T    67    PASS     NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ  1|2:21:6:23,27  2|1:2:0:18,2     2/2:35:4
20     1230237  .          T     .      47    PASS     NS=3;DP=13;AA=T                   GT:GQ:DP:HQ  0|0:54:7:56,60  0|0:48:4:51,51   0/0:61:2
20     1234567  microsat1  GTC   G,GTCT 50    PASS     NS=3;DP=9;AA=G                    GT:GQ:DP     0/1:35:4        0/2:17:2         1/1:40:3
```

Figure 3.2: Example for VCF format.

```
>MCHU - Calmodulin - Human, rabbit, bovine, rat, and chicken
MADQLTEEQIAEFKEAFSLFDKDGDGTITTKELGTVMRSLGQNPTEAELQDMINEVDADGNGTID
FPEFLTMMARKMKDTDSEEEIREAFRVFDKDGNGYISAAELRHVMTNLGEKLTDEEVDEMIREA
DIDGDGQVNYEEFVQMMTAK*
```

Figure 3.3: Example for FASTA format.

One of the reasons for reference bias is that the reference genome cannot contain individual differences. And our hypothetical sequence is a sequence that combines reference genome and variant, simulate the occur of mutation of the reference sequence by combining variant, make it includes individual differences. So, we will base on the information of each variant in the VCF file, including variant position (POS), chromosome name (CHROM), sequence before mutation (REF), and sequence after mutation (ALT) to generate a hypothetical sequence.

For each hypothetical sequence, we will according to the variant position, replace sequence before mutation (REF) to sequence after mutation (ALT), at the same time we will extract the reference genome part front and back in this region, and the length will be the length of a read, The length of this area is long enough to include all the reads we are interested in and also include individual differences. Figure 3.4 illustrates in detail how we generate a hypothetical sequence based on different types of variants.

We hope to find a read that matches our hypothetical sequence in the FASTQ file to achieve the effect of reducing the reference bias

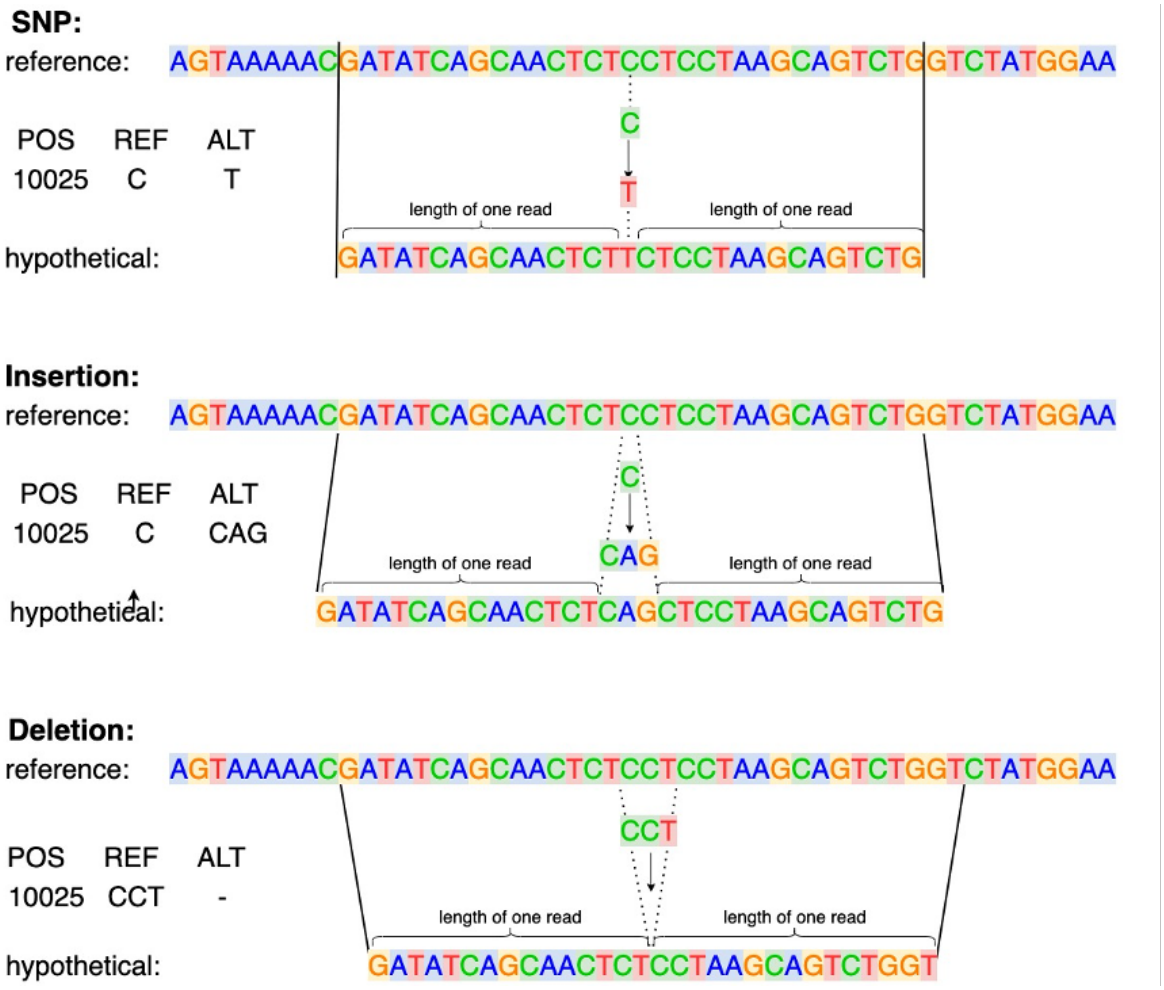

Figure 3.4: Three different cases to construct hypothetical sequence.

3.3 Read-index

Next is an important step for us to find the read that matches our hypothetical sequence. But if we follow the practice of BWA-MEM, looking for read in FASTQ file will cost us a lot of time.

First for general practice, hypothetical sequence is the reference genome, and we

need to map reads on this, just like alignment. so we need to construct index for each hypothetical sequence and then query all the read in FASTQ file. As we knew that build BWT index need $O(nlogn)$ (n is reference length)to constructed, and query will be $O(m)$ (m is read length)for each read.

In general situation, Time spent each query as follows:

$$O(m) + O(nlogn)$$

It is indeed a very efficient method in normal times. It only needs to query once for each read, because all reads only need to be aligned to one reference genome sequence. But in our situation, our hypothetical sequence (reference) will not only have one. For each variant we will establish a hypothetical sequence, so if we follow the traditional method, we need to search all the reads every time, our speed will be very slow.

But what if we construct index for read and used our hypothetical sequence to query?

1. We just need to build index once a time.

2. For each hypothetical sequence, our query will be very fast.

Let's take an example and some assumptions :

In general situations, we need to create an index for each hypothetical and do a complete read search, it will probably take time:

$$hypothetical\ amount\ *\ (\ index\ hypothetical\ sequence\ +\ query\ all\ reads)$$

Because the hypothetical sequence is very short compared to all reads, the time of its index can be ignored, then its time will probably be:

$$hypothetical\ amount\ *\ query\ all\ reads$$

And through our way of creating a read index, the time will probably be:

$$index\ all\ reads\ +\ (\ hypothetical\ amount\ *\ query\ hypothetical\ sequence)$$

Also because the hypothetical sequence is very short, we can ignore its query time, then our time will probably be:

$$index\ all\ reads\ +\ (\ hypothetical\ amount\ quickly\ queries)$$

According to the above results, we decided to index our read to generate the read -index, which helps us quickly find the matching read. Before indexing, we need to remove some information in read file to convert FASTQ format to FASTA format. This is very simple like (Table 3.1) then we can directly call BWA index function to construct read-index.

Table 3.1: The difference between FASTQ and FASTA format

		Fasta	Fastq
format	Line 1	>description of sequence	@sequence id
	Line 2	sequence	sequence
	Line 3		+
	Line 4		Quality value
example		>NT_113878.1 chromosome 1 GATAGCATCTGGCCTTATGGCCTTA	@SEQ_ID ACCACCTGCTCCTGGCCTTATGGCCTTA + ???DB+=2C>??E;EC>??E;EC>??E;EC>?

3.4 Find Similar Reads

In previous section, we have introduced how BWA-MEM work, and in this section, we will introduce how do we use BWA and read-index to find the read that matched hypothetical sequence and filter the read we find.

BWA is an open-source software. Fortunately, we can directly use the functions of BWA-MEM and modify part of the code according to EAGLE needs to make it meet our read-index search requirements. Basically, we will use seed alignment based on super maximal exact matching to search for our hypothetical sequence.

The following will describe the changes we made to our needs. First of all, for the original BWA-MEM, a read will have only one best matching position, and the others will be marked as secondary (or no primary) (Figure 3.5), because for BWA only need an optimal alignment, but in our opinion, we should consider all matching reads as primary if it includes the variants (Figure 3.6).

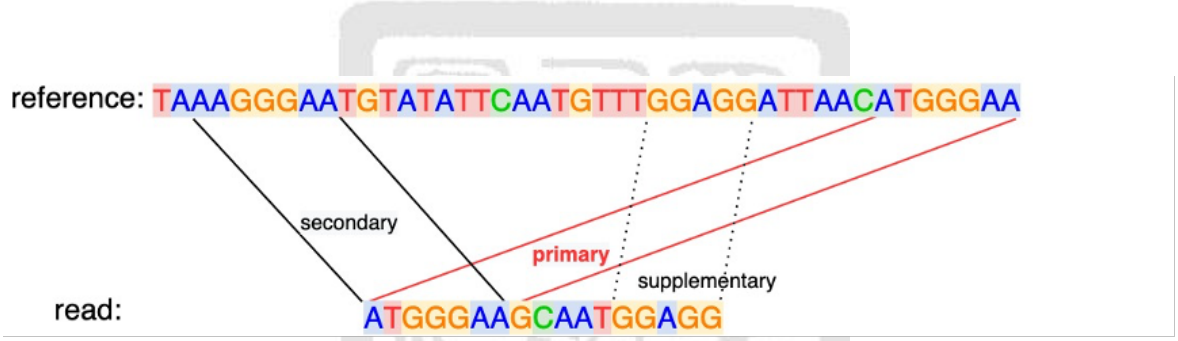

Figure 3.5: BWA secondary alignment, supplementary alignment.

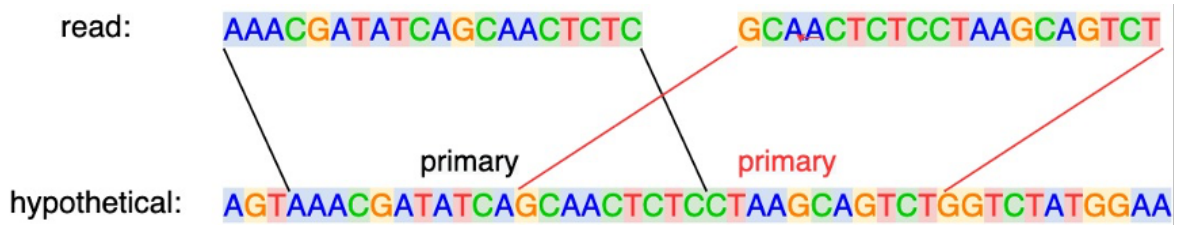

Figure 3.6: all alignment should consider as primary alignment.

Then we use BWA-MEM to search for our hypothetical sequence using seed alignment based on super maximal exact match, after this, we will get two sets of index as Figure 3.7 shows, one set represents the start and end positions of the region on the hypothetical sequence, and the other set represents the index of the corresponding

read region on the read-index.

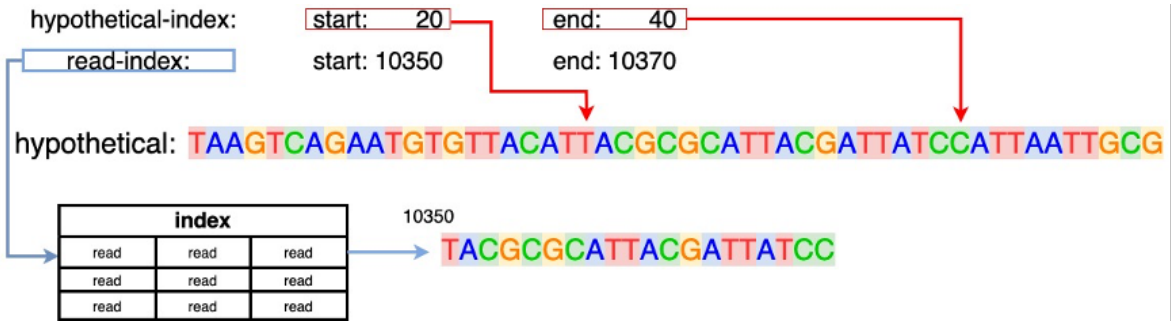

Figure 3.7: how to find matching read through read-index.

Through the index of hypothetical sequence, we can check whether it contains variant, if not, we will filter it ,like Figure 3.8 shows. End of this step, we will get a set of meet the requirements read, then we can add to the pileup

Figure 3.8: Filter reads that didn't overlap variant.

3.5 Add reads to the pileup and generate information

In the last step, we need to compare all the read sets obtained in the previous step with the read in the pileup. What is the pileup? First, in the BAM file, we can observe that the read is mapped to a position on reference genome. Many reads may alignment at the same region, which is called the pileup. Our final goal is to find reads that overlap

with the same variant position but do not exist in the pileup, (Figure 3.9 shows the pileup reads which overlap on a variant position.)

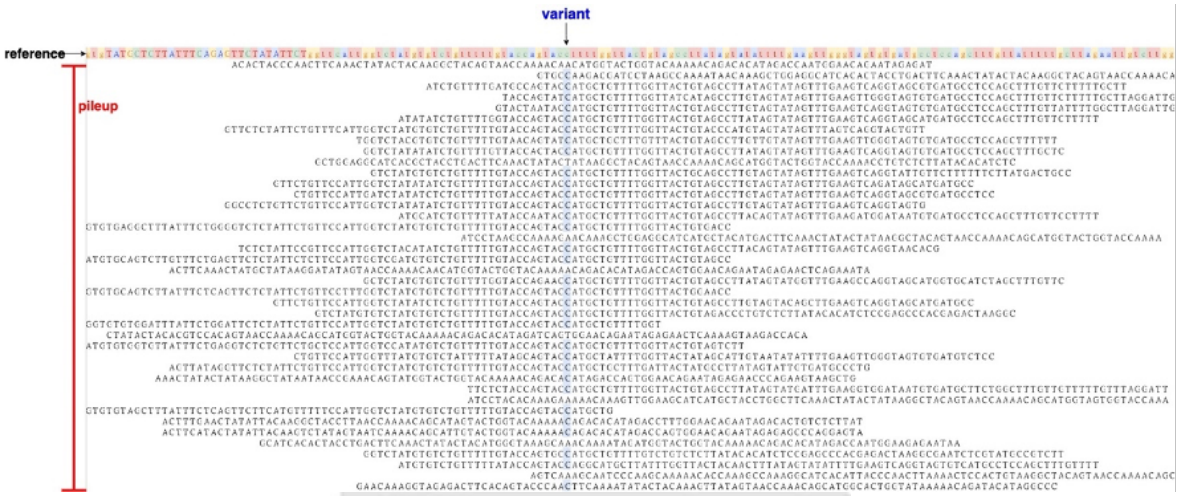

Figure 3.9: pileup on variant position.

We can get the pileup with BAM file quickly through utility library HTSlib, and after we compare the read set with these pileup reads, if the read does not exist in the pileup, it means that we have got the results we want. So, next step for our work is to convert this read, and use the two sets of index obtained in the previous step to generate some essential information that will be needed in our eagle calculation, like SAM flag and CIGAR string, SAM flag is represents the mapping status of this reads and CIGAR string is alignment console of this read. We will explain how we get CIGAR string through two sets of indexes.

And if we get the SAM flag and CIGAR directly through these two sets of indexes, what we get is the alignment information of hypothetical sequence mapping to read, like previous Figure 3.8 shows, but what EAGLE needs is the alignment information of read mapping to reference sequence. So we need do some processing on index.

First of all, one of the sets is read index, which represents the position of read

in read-index, we can directly obtain our read through it. Another set is the start and end on our hypothetical sequence, this is what we said before, the corresponding mapping region of read. We need to project this region onto reference genome to help us produce the alignment of read to reference genome.

This problem can be divided into 4 cases: (Figure 3-10)

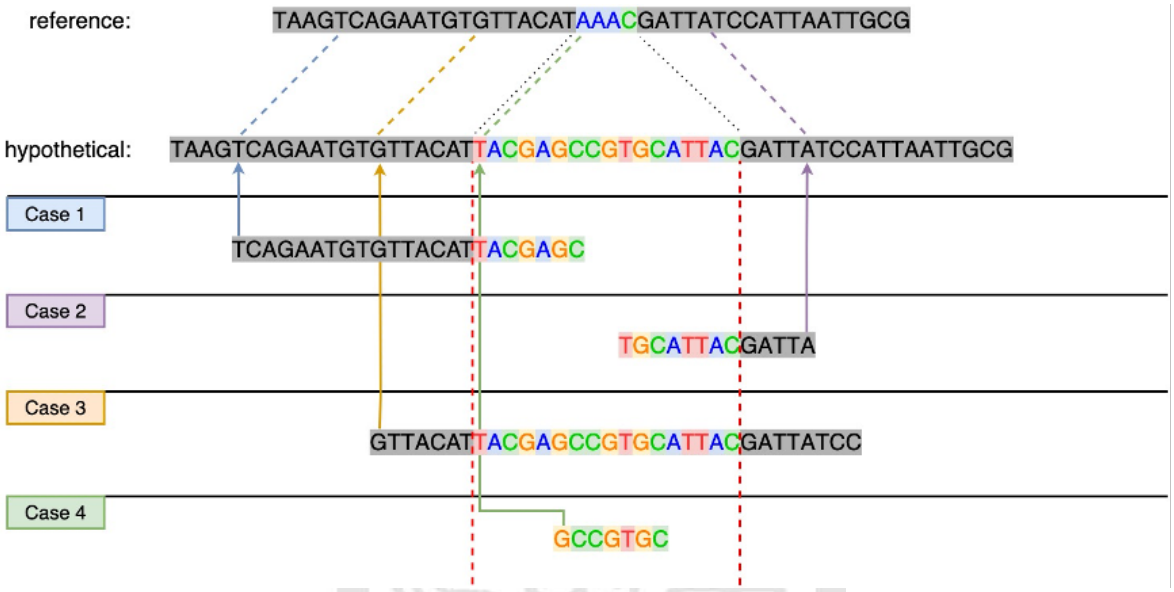

Figure 3.10: Four cases deal read with reference position.

1. the tail of read sequence overlapped with variant.

2. the front of read sequence overlapped with variant.

3. the center of read sequence overlapped with variant

4. all of read sequence overlapped with variant

For first case and third case we can just directly get the position in reference, and forth case we use the position of variant. A different example is second case, in this case, we need to use the end of the region to push-back the position.

In the last step, we already have read mapping position on reference genome, so that we can create read alignment information. Most of them are nothing special and will not change as we change the alignment position, except for CIGAR string, we need to convert it from hypothetical sequence's alignment information mapped against read to read's alignment information mapped against reference sequences (Figure 3.11). we use small function to constructed CIGAR string for our read.

Figure 3.11: how to convert CIGAR strings.

Go through all the steps we can add read to the pileup for eagle to calculate likelihood.

Chapter 4

Experiment and Results

In this section, we first introduce our dataset and some tools, then we will explain the steps of our experiment, and then analyze the experimental results in three different simulation scenarios, and finally use the VCF file from dbSNP and divide it into SNP and INDEL according to the type of mutation Discuss the experimental results separately.

4.1 Dataset

Eagle need three main input files, they are reference genome (FASTA), read data (FASTQ), and variant file (VCF).

First, the reference genome sequencing data is Genome Reference Consortium Human Build 37 patch release 13 (GRCh37.p13) version from NCBI website. There are 1 to 22, X and Y chromosome of primary assembly and some extra sequences.

Second, the read sequencing data is real sequencing data from NA12878 which is a cell line of an individual female from a CEPH pedigree that is Utah residents with Northern and Western European ancestry, using an exome sequencing dataset (Garvan HG001) by sequencer HiSeq2500 from Genome-In-A-Bottle (GIAB).

Third, we will use VCF files from the Single Nucleotide Polymorphism Database (dbSNP). dbSNP is a public domain archive for a wide collection of simple genetic polymorphisms. The polymorphism set includes single bases. Base nucleotide substitutions (also called single nucleotide polymorphisms or SNPs) , small-scale multi-base

deletions or insertions (also called deletion insertion polymorphisms or INDELs), We will discuss according to these two different types.

Last, Finally, we will also use the simulated VCF file, the variant file is generated by our simulation, we will introduce it in detail in the next chapter.

4.2 Simulate workflow

In order to carefully evaluate our effect and verify whether our method can reduce the influence of reference bias, we use simulation methods to conduct experiments. We modify the GRCh37 reference genome sequence to simulate the actual occurrence of mutations in our reference genome (Figure4.1), record the result of our modification to generate simulate VCF file.

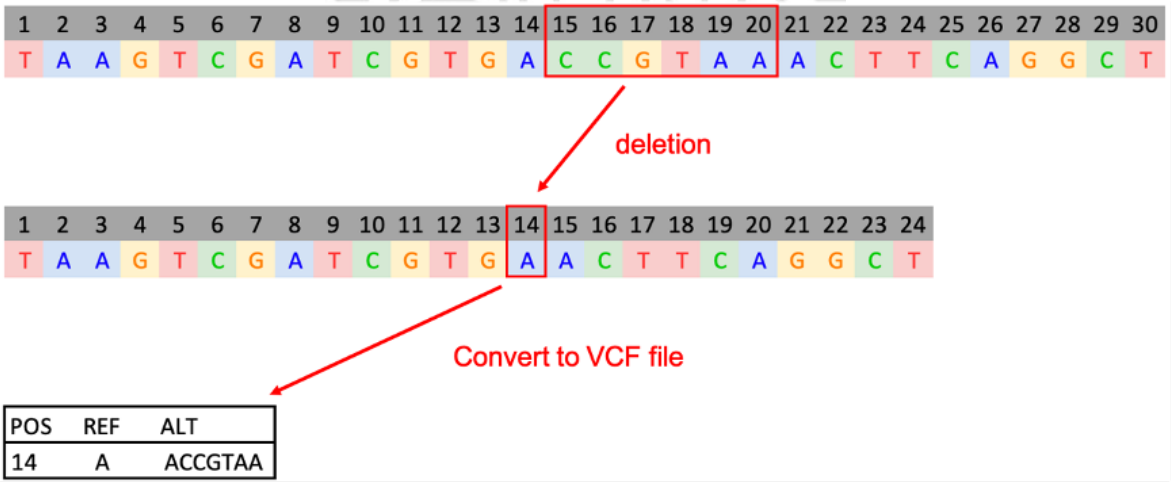

Figure 4.1: Simulate variants.

Then we will use BWA, follow the traditional approach, and do the alignment again. Use the re-alignment step to simulate the occurrence of reference bias, because we modified the reference, when re-alignment, the pileup at the original position will have some reads affected by the reference bias, and these affected reads will be mis-

alignment or incorrect alignments. After generating the files needed by eagle, we can start our experiment, and the whole experiment process will look like Figure 4.2.

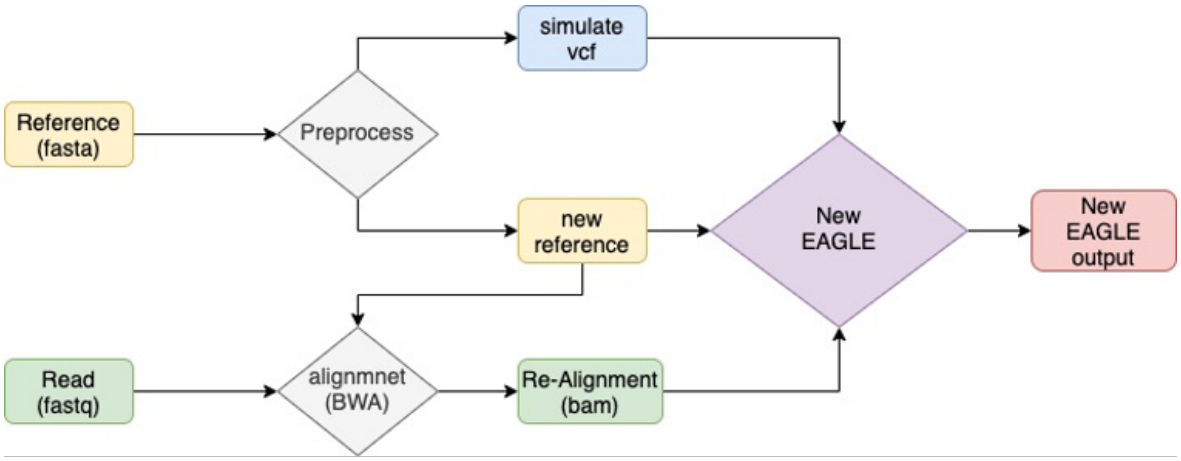

Figure 4.2: Simulate workflow.

In addition, we use SAMtools to check the read coverage, and test it in 3 cases according to different read coverage(Table 4.1), discussing different read coverage, we reduce the influence of reference bias

Table 4.1: total simulate indels amount

Read Depth	5X-20X	20X-40X	40Xup
Indel length	1~50	1~50	1~50
amount	20 for each	20 for each	20 for each
Total variant	1000	1000	1000

For each case, we will simulate 20 variants for indels of different lengths from 1 to 50. At the same time, we will discuss the results after re-alignment into two situations, one is that after re-alignment, BWA can still find read and generate pileup at that position, and the other is that after re-alignment, no read can be aligned to that position.

For the first case, EAGLE is still able to evaluate the variants, so the focus of our discussion will be the changes in EAGLE evaluation after the introduction of our method. For the second case, we will focus on the results calculated by EAGLE

And our evaluation standard is to observe the output of eagle for each variant odds:

$$\log(P[\text{Alt}—v]/P[\text{Ref}—v])$$

4.3 Case 1: Low coverage

First of all, we can see that for the original eagle found (Table 4.2 the first two rows), We can see that in the case of low read coverage, since we only have a small number of reads, once there is a reference bias, we will easily be severely affected. And therefore most of our reads cannot be aligned correctly to the right position, at the same time, you can also see that the longer the length of indels, the more serious the impact, but relatively, after we find the read, it also has more impact on the longer indels.

Table 4.2: low coverage variants

Indel length	1~10		11~20		21~30		31~40		41~50		Total
Not changed	47	42%	38	41.8%	30	29.1%	10	19.2%	0	0%	125
changed	65	58%	53	58.2%	73	70.9%	42	80.8%	5	100%	238
New	88		109		97		148		195		637

Note: Explain that in the case of low coverage, after we add read, eagle evaluates the number of variants affected.

Next, let's take a look at the changes after adding our method. We can see that eagle prefers that the position does not contain variants in the original judgment, but

in most cases (82%), when we find some reads that contain variants, we can help eagle judge whether the position contains variants. (Table 4.3)

Table 4.3: The number of variant changes on lower coverage

Indel length / P[Alt]/P[Ref] < 1	1~10	11~20	21~30	31~40	41~50	Total
before	7	2	3	4	3	19
after	2	1	0	1	0	4

We checked the cases that changed after we found the read, and we found that most of the examples were as we predicted, and we were able to find reads in FASTQ that were misalignment due to reference bias. Figure 4.3 is one of the cases.

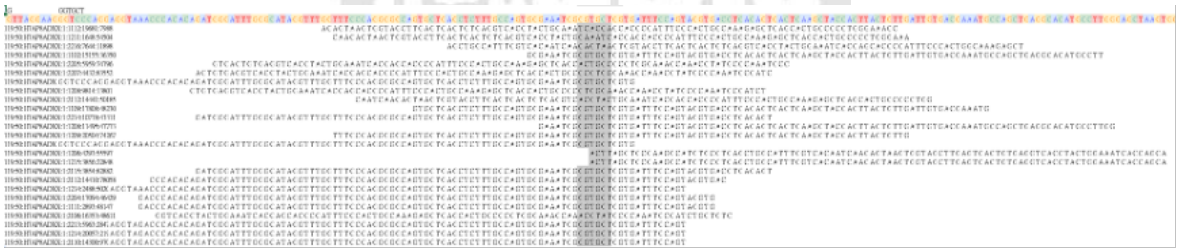

Figure 4.3: Find reads containing variant in low coverage.

Similarly, there are some reads that we find more ref biased. This is because when we change reference sequence, we don't require the base of each position to be the same as the reference genome, so this situation is reasonable. (Figure 4.4)

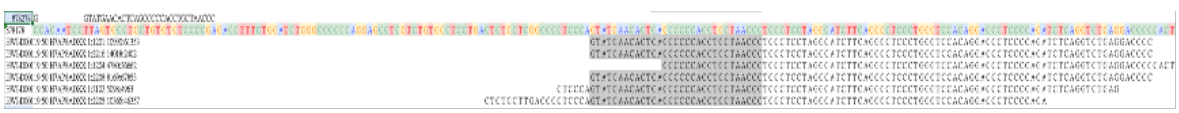

Figure 4.4: Find read but similar with reference sequence in low coverage.

Next, we can look at the comparison of eagle odds. The numerical calculation method is: odd(after)/odd(before) , can represent our help to EAGLE

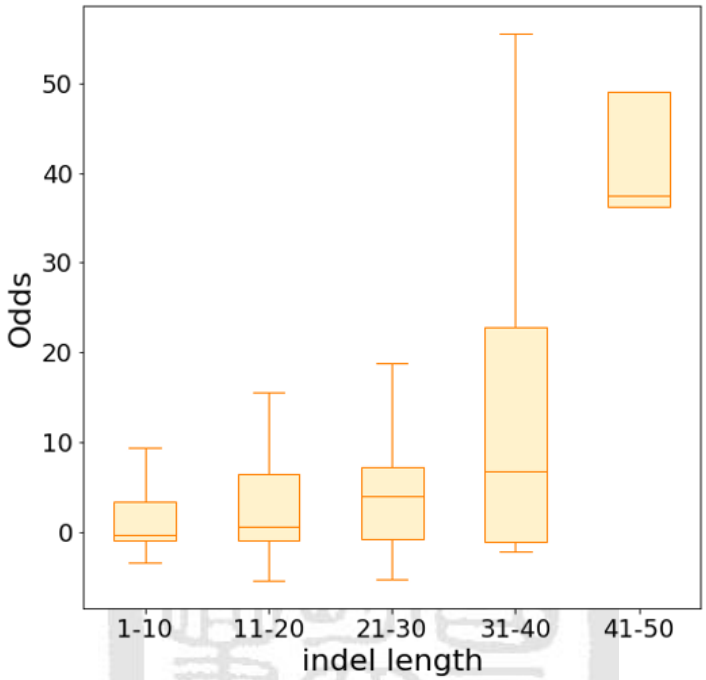

Figure 4.5: low coverage odds change ratio.

In the case of low coverage, first of all, we can observe that after we add read, the confidence value of most of the variants increases because we find more reads that contain mutations (Figure 4.5), and regardless of the length of the indel, their distribution of odds increase percentage is mostly above zero. This may mean that even in the case of low coverage, our method is very helpful for most mutation points, especially for indels with a length of 40 or more, where the increase is particularly obvious.

Finally, we look at the last row of Table 4.2, we can see these positions are affected by the reference bias, so that no read can be mapped to these positions to form a pileup. Our method is obviously very helpful. For the remaining variants in the 1,000 variants, we can find those reads that are affected by reference bias and are lost, thereby helping

eagle evaluate the possibility of these mutations.

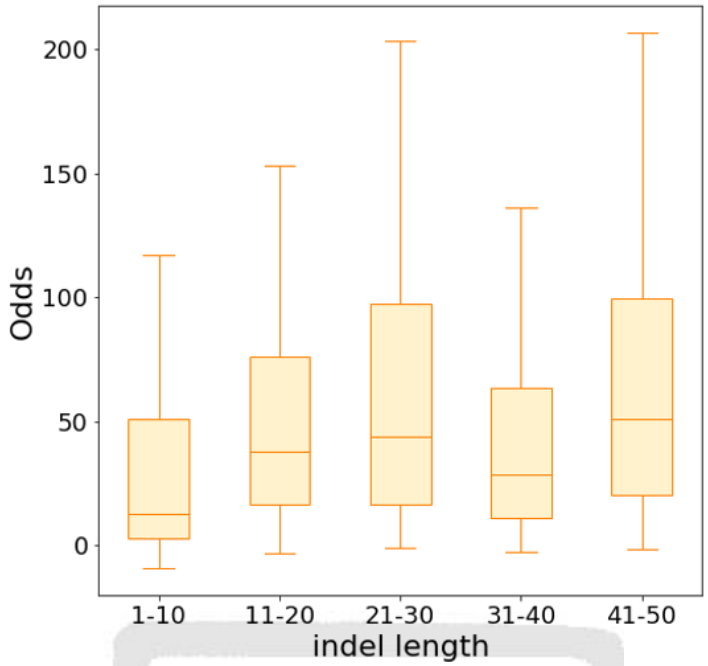

Figure 4.6: low coverage odds ratio on no pileup variants.

Then see the odds calculated by eagle after we find the read (Figure 4.6), we can see the odds calculated by eagle. For most of the variants, eagle gives the conclusion that contained the variant at this position, which is also in line with our expected results.

4.4 Case 2: Medium coverage

Table 4.4: medium coverage variants

Indel length	1~10		11~20		21~30		31~40		41~50		Total
Not changed	36	28.3%	21	13.4%	25	15.8%	33	20.4%	22	17.1%	137
changed	91	71.7%	136	86.6%	133	84.2%	129	79.6%	107	82.9%	596
New	72		41		42		38		71		264

Note: Explain that in the case of medium coverage, after we add read, eagle evaluates the number of variants affected.

First of all, we can see that in this case,Table 4.4, because of the increase in read coverage, the variants affected by reference bias are significantly reduced, and the number of reads that we can overlap with the variant position has also increased. For the same reason, more misaligned reads are affected by reference bias.

Table 4.5: The number of variant changes on medium coverage

Indel length / P[Alt]/P[Ref] < 1	1~10	11~20	21~30	31~40	41~50	Total
before	10	11	9	6	5	41
after	1	1	0	1	0	3

With the same increase in read coverage, In EAGLE's initial judgment, the number of tendencies that position does not contain variant has also increased, because more misalignments have occurred. And just for this reason, the more reads that we can find to help EAGLE. Therefore, we can see that after we find the read, the affected variant has also increased, nearly 91% of vaiants in Table 4.5 have been changed to the correct situation. (Figure 4.7 ,4.8 are some examples)

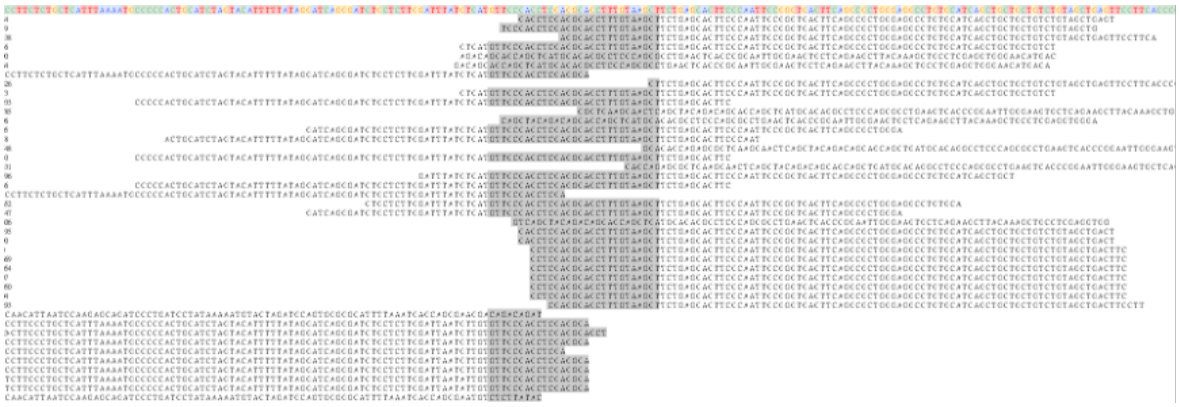

Figure 4.7: Find reads containing variant in medium coverage.

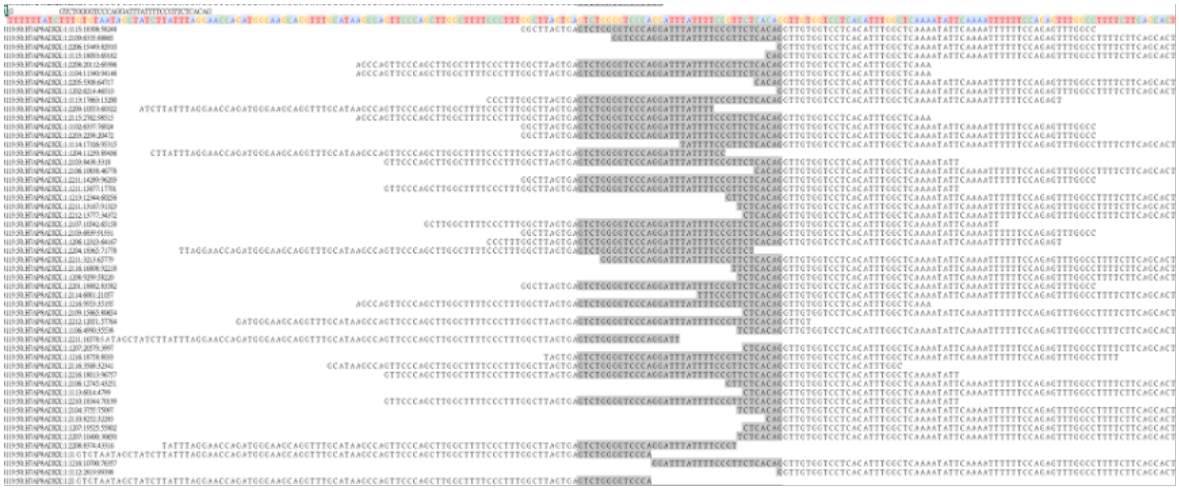

Figure 4.8: Find read but similar with reference sequence in medium coverage.

We can find that in bad cases, although we found reads similar to hypothetical sequences, we did not improve the score of EAGLE on variants. In these few examples, we found that their mutated sequences happened to be repeated sequences. Therefore, the similarity between it and the reference sequence is also very high(Figure 4.9). We think facing this reason squarely affects EAGLE's judgment on him

Figure 4.9: pileup with the find read similar with reference sequence.

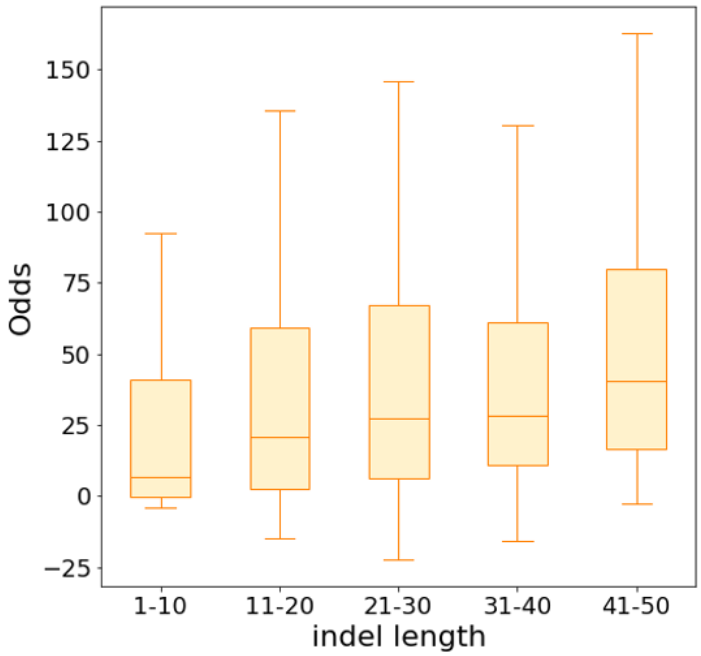

Figure 4.10: medium coverage odds change ratio.

Next, we can look at the comparison of eagle odds in medium read coverage (Figure 4.10). Basically, the change is similar to the situation of low read coverage. We can also see larger growth in the longer indel. The difference is that we also have a good performance in the shorter indel. And the average growth rate is better than that of low coverage. This result is quite reasonable, because we find more reads that match the variant, which can help eagle give these mutations greater confidence. We can also see that indels with a length of 10 or more on our boxplot have Q3 greater than 0, which means that most of our reads can help us eliminate the influence of reference bias.

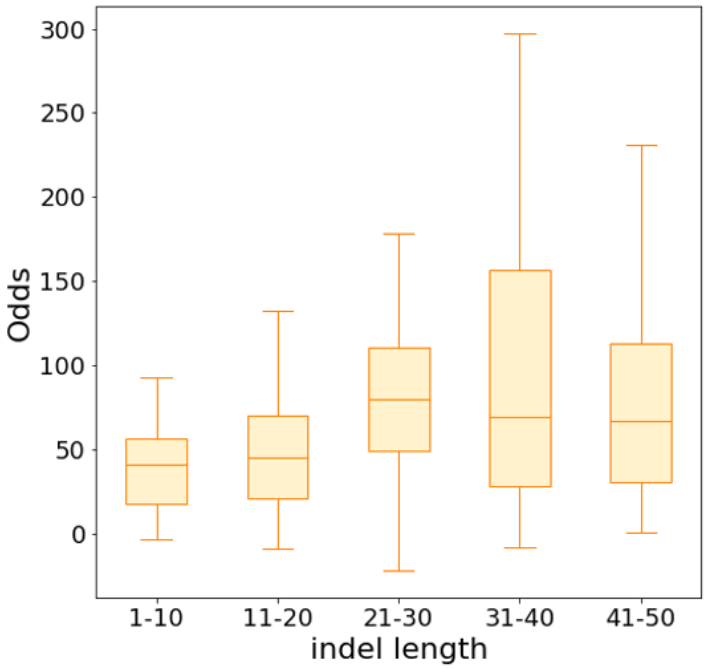

Figure 4.11: medium coverage odds ratio on no pileup variants.

Finally, we look at the last row of Table 4-4, we can see that our method still can help some variants that EAGLE cannot calculate because there is no pileup at the position. Figure 4.11 shows these variant odds calculated by eagle.

In this case, EAGLE can get more correct conclusions, that is, the position does contain variants, because we can find more read support variants. It can be seen that compared to the previous case, regardless of the length of the indel, their Q3 is 0 some distance higher, this result is also as we predicted ,and Figure 4.12 is some example.

Figure 4.12: find reads on no pileup position.

4.5 Case 3: High coverage

Table 4.6: high coverage variants

Indel length	1~10		11~20		21~30		31~40		41~50		Total
Not changed	44	25.9%	31	16.9%	36	19.6%	4	4%	0	0%	116
changed	129	74.1%	152	83.1%	148	80.4%	96	96%	80	100%	605
New	26		17		16		100		120		279

Note: Explain that in the case of high coverage, after we add read, eagle evaluates the number of variants affected.

Basically, in this case, we will observe results that are very similar to medium coverage, that's because there are quite a lot of reads, most positions can still produce pileup (Table 4.6).

Table 4.7: high coverage variants

Indel length / P[Alt]/P[Ref] < 1	1~10	11~20	21~30	31~40	41~50	Total
before	6	6	2	35	36	85
after	1	0	0	0	1	2

Note: Explain that in the case of high coverage, after we add read, eagle evaluates the number of variants affected.

Interestingly, here we observe that the coverage rate is twice as high as that of the medium coverage, EAGLE initially judged that the position does not contain variation, but also through our method, we can complete more corrections than the medium coverage case. There are nearly 97% of variants in Table 4.7 have been changed to the correct situation. (Fig 4.13,4.14 are some examples)

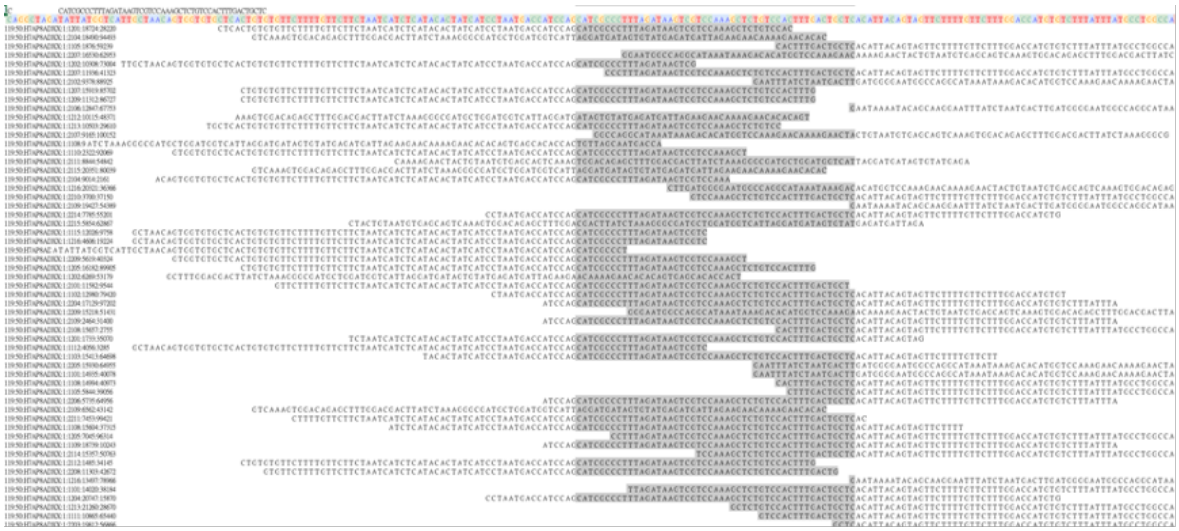

Figure 4.13: Find a large number of matching reads.

Figure 4.14: Pileup after adding reads.

Then, we look at the comparison of eagle odds in high read coverage (Figure 4.15). It can be seen that compared with the previous example, it is very obvious that the longer the indel, the better the growth. However, we found many cases where the length of the mutation is greater than 50. This also shows that the more reads that we can find that are lost, the better we can reduce the impact of reference bias.

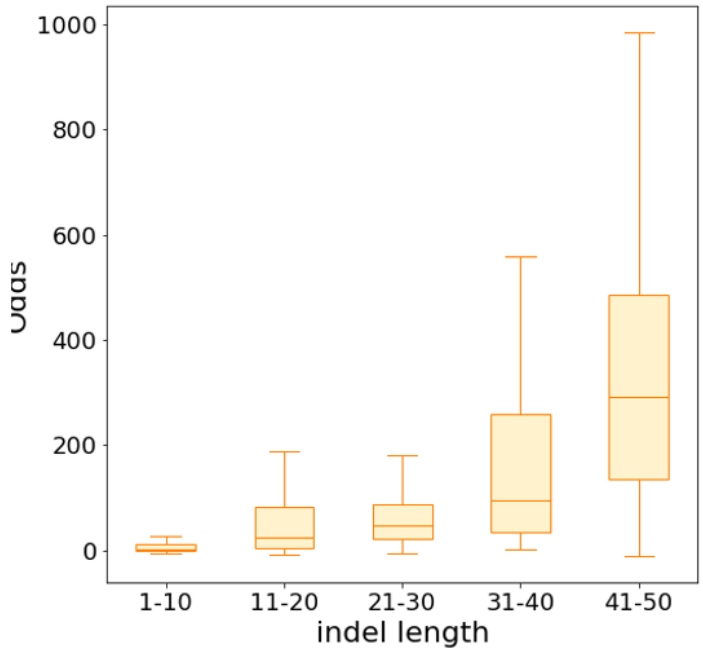

Figure 4.15: high coverage odds change ratio.

Finally, we look at the last row of Table 4-6, we can see the same result as the previous case medium coverage. Figure 4-16 shows these variant odds calculated by eagle.

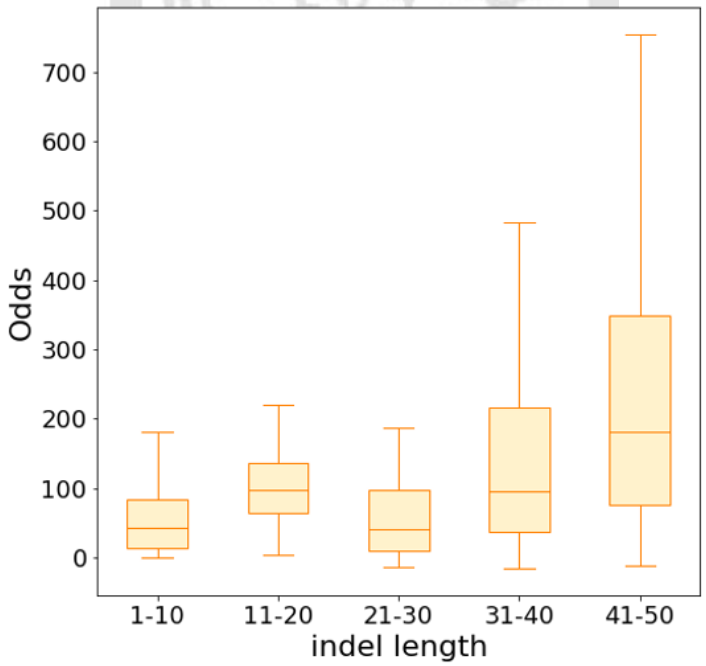

Figure 4.16: high coverage odds ratio on no pileup variants.

4.6 SNPs in dbSNP dataset

There are 15,936,832 SNP variants form the dpSNP dataset, we random select 100,000 variants from the dataset, and the amount of matching pileup of the querying SNP variants are 50488. After joining our method, there are 22995 variants that have been changed as a result, we have looked at a few cases (Figure 4.17,4.18,4.19,4.20), we find many sequences that include the variants.

Figure 4.17: found match reads with hypothetical sequence in SNP.

Figure 4.18: The pileup corresponding to the Figure 4-17 variants.

Figure 4.19: found match reads but more similar to reference sequence in SNP.

Figure 4.20: The pileup corresponding to the Figure 4-19 variants.

And we can find that the read we found gives eagle more basis for judgment, but the magnitude of the change is not very obvious in the SNPs (Figure 4-21), and most of them are less than 0. We speculate that the single-base nucleotide substitutions will not have a great impact on the alignment results.

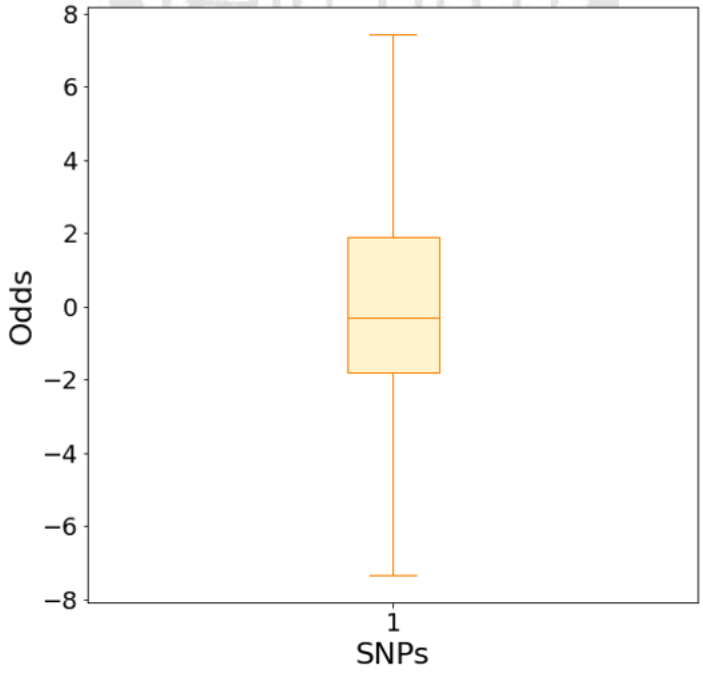

Figure 4.21: SNPs odds change ratio.

4.7 INDELs in dbSNP dataset

There are 1,759,193 INDEL variants form the dpSNP dataset, we random select 10,000 variants from the dataset, and the amount of matching pileup of the querying INDEL variants are 5093. After joining our method, there are 2482 variants that have been changed as a result, we have looked at a few cases (Figure 4.22,4.23,4.24,4.25).

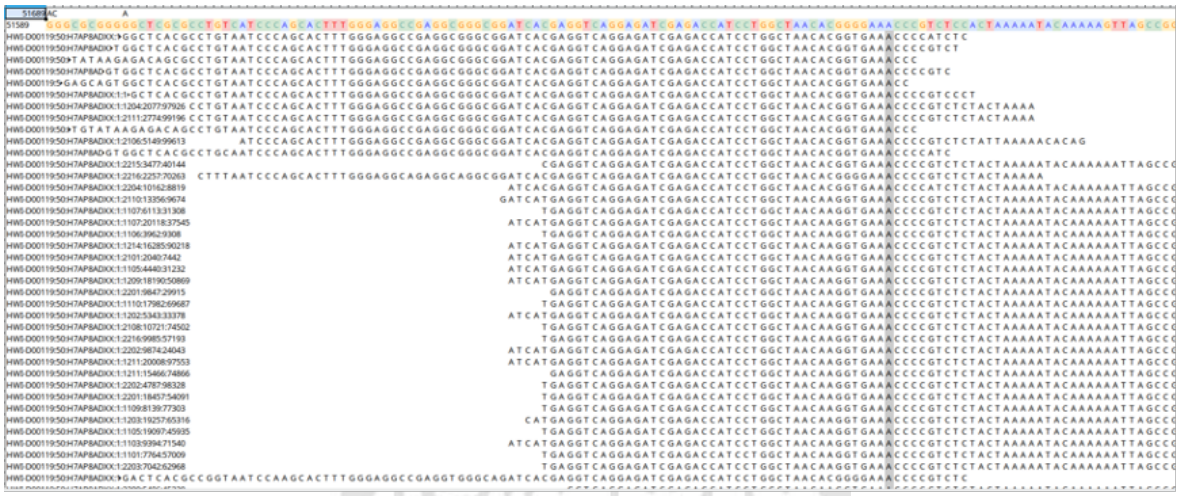

Figure 4.22: found match reads with hypothetical sequence in INDEL.

Figure 4.23: The pileup corresponding to the Figure 4-22 variants.

Figure 4.24: found match reads but more similar to reference sequence in INDEL.

Figure 4.25: The pileup corresponding to the Figure 4-24 variants.

It can also be found that the read we found gave eagle more evidence for judgment, but the magnitude of change in INDEL was significantly greater than SNP, just like the experiment we simulated before.

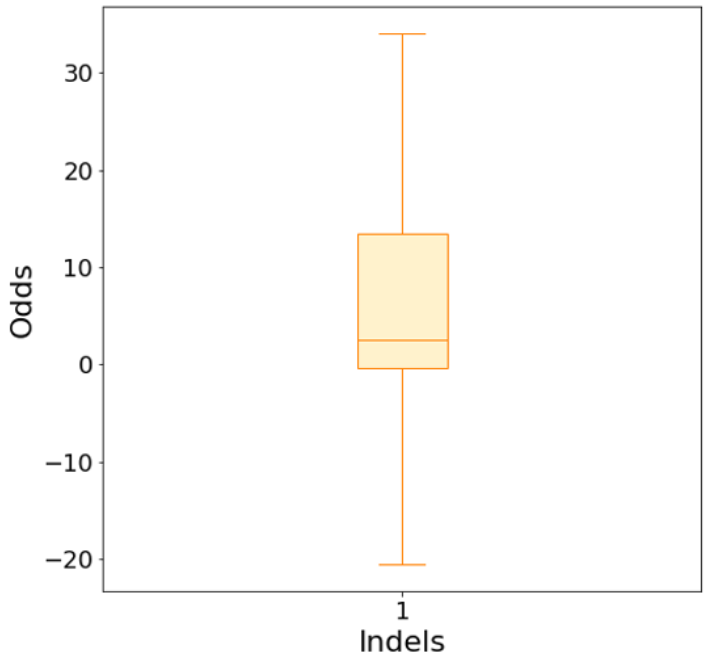

Figure 4.26: INDEL odds change ratio.

4.8 Execution time and memory consumption

Finally, let's discuss our execution time and memory consumption, our test data are as follows Table4.8, test environment as Figure 4.27.

Table 4.8: Test Dataset

	Reference genome	Read	VCF
Dataset	Genome Reference Consortium Human Build 37 patch release 13	Garvan_NA12878_HG001_Hiseq_Exome	dbSNP of single nucleotide polymorphisms
File Size	3.1G	5G	669.7MB

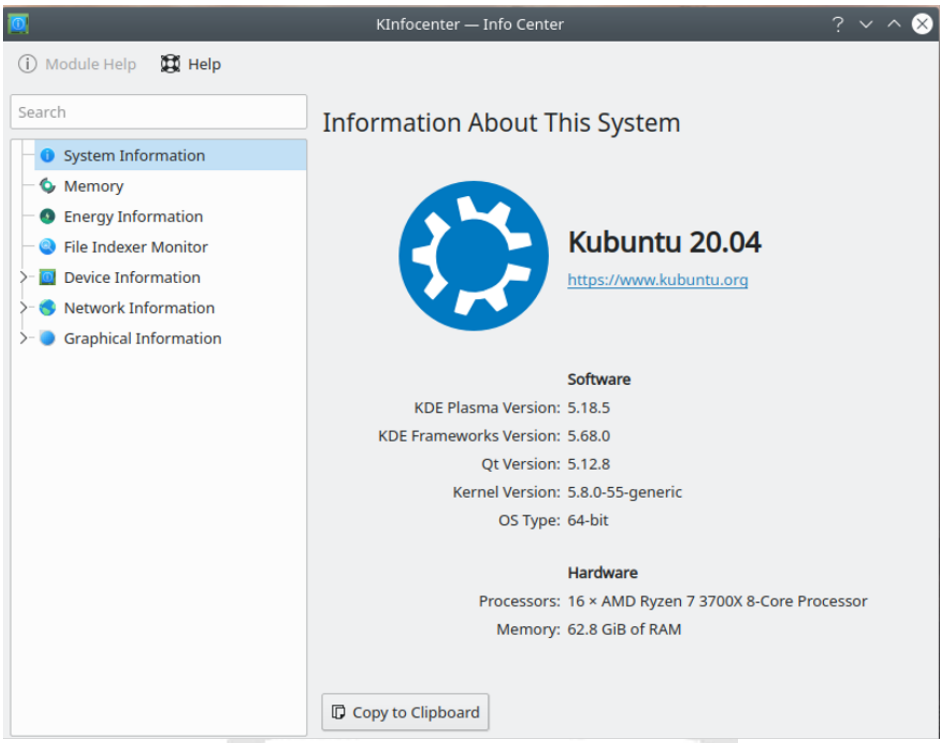

Figure 4.27: Test Environment.

Experimenting with the above conditions, first we can see that the memory we consume is 11.64G, and the memory consumed by EAGLE is 0.36G, we can find that the memory space we need is indeed much larger than the original EAGLE. This is because we need to access read-index, reference-index and read files to complete quick searches, so the memory size we need is closely related to these three. But what we care most about is our execution speed. The time taken by EAGLE is 3618.18 seconds and our method is 6426.28 seconds. Although it seems to take a lot of time, we can find out that we have increased by carefully analyzing our execution time. A large part of the time is to index the read. The time it takes is 1649.33 seconds, and the time we search is very quick, as shown in Figure 4.28, and a part of the time is because we find more reads, the calculation added by EAGLE Time. Considering this situation, we can say that the added time is relatively small.

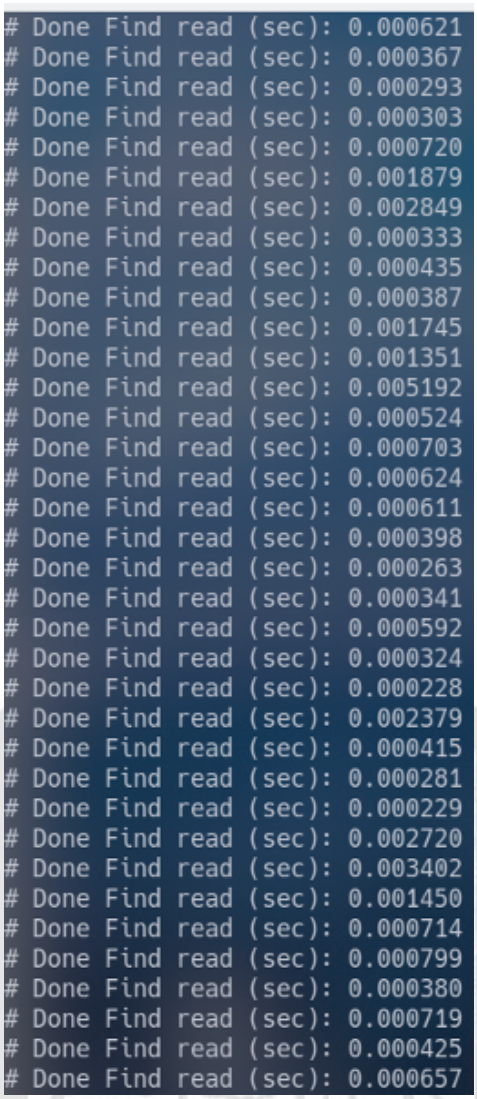

Figure 4.28: searching per hypothetical sequence time.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

In order to try to reduce the influence of reference bias and improve the accuracy of EAGLE for evaluating variants, we have expanded the function of EAGLE. Before EAGLE calculation, by generating hypothetical sequences that binding variants and reference sequences, and create a read-index to achieve quick search, find out those misalignment or incorrect reads, and finally add it to pileup.

After this series of steps, we conducted an experiment that simulates the real situation to verify our method. The experimental results of our simulation show that our method can effectively reduce the influence of reference bias, and in a longer indel The effect is particularly significant. At the same time, we can work normally in any read coverage situation.

We are also testing in the real data set dbSNP, we can get the same good results as before, but the effect on SNPs is relatively limited.The reason why the method we implemented can be successfully applied to EAGLE is mainly because the EAGLE probability model is mainly used read to estimate the likelihood of each variant, and we can find many reads that are affected by reference bias and are discarded. We only need to increase the establishment of read-index and a little search time to eliminate the influence of reference bias.

5.2 Future Work

In future research, I think we can think about how to establish a more perfect hypothetical sequence. At present, our method is to establish our hypothetical sequence for a single mutation, but we have found that adjacent mutations are prone to occur. Maybe we can consider neighboring variants. Different permutations and combinations generate hypothesis sequences to improve the effect of our method, but it may consume a lot of time.

The possibility of combining good alignment tools by creating better hypotheses is worthy of further investigation.