國立成功大學
資訊工程研究所
碩士論文

在 EAGLE 變異評估方法中結合 BWA-MEM 讀取索引以消除基因組變異辨認中的參考偏差

Towards Eliminating Reference Bias in Genome Variant Calling — Integrating a BWA-MEM Read Index into the EAGLE Variant Evaluation Method

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

國立成功大學

碩士論文

在 EAGLE 變異評估方法中結合 BWA-MEM 讀取索引以消除基因組變異辨認中的參考偏差

Towards Eliminating Reference Bias in Genome Variant Calling — Integrating a BWA-MEM Read Index into the EAGLE Variant Evaluation Method

研究生：蘇文弘

本論文業經審查及口試合格特此證明

論文考試委員：賀保羅

劉宇秦

許鈞凱

指導教授：賀保羅

單位主管：張茵生

中 華 民 國 110 年 7 月 6 日

Towards Eliminating Reference Bias in Genome Variant Calling — Integrating a BWA-MEM Read Index into the EAGLE Variant Evaluation Method

by

Wen Hong Su

A thesis submitted to the graduate division in partial fulfillment of the requirement for the degree of Master of

Institute of Computer Science and Information Engineering
National Cheng Kung University

6 July 2021

Approved by:

Advisor:

Chairman:

在 EAGLE 變異評估方法中結合 BWA-MEM 讀取索引以消除基因組變異辨認中的參考偏差

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數據中變異辨認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，處理變異辨認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辨認方法，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辨認，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較小。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

Towards Eliminating Reference Bias in Genome Variant Calling — Integrating a BWA-MEM Read Index into the EAGLE Variant Evaluation Method

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

*Student
†Advisor

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 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 a 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 performed 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 regions with repetitive 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

誌謝

本篇論文能順利完成，首先要感謝我的指導老師賀保羅教授，感謝老師這兩年給予我許多幫助及建議，讓我了解對於進行一個研究該有的態度以及方法。在生物知識方面，老師也提供了許多我們想法以及知識，讓我的研究能更順利進行並且更加完整。在實驗室這兩年眞的讓我收穫了很多，不管是專業的能力還是做事的態度，這些都將成爲我之後的工作及人生的養分。感謝口試委員劉宗霖教授及許釗凱教授在生物資訊以及臨床醫學方面給的許多有幫助的建議，令我穫益良多。

在實驗室這兩年，特別感謝的是同屆的同學喚凱、綾娟、士杰以及國勳，不管是課業和生活都陪伴我許多。有你們我的碩士班生活才能過得非常充實且順利。接著要感謝的是實驗室的學長們，感謝東誼分享許多研究上的經驗以及想法，感謝晉昇在給了我許多課業上的幫助，感謝榕辰加入讓我碩一生活充滿樂趣，感謝孟勳不管是在課業抑或是生活上給了我許多幫助及建議，畢業後也不斷關心我們、協助我們。最後謝謝實驗室的學弟妹宥霖、恆霖、力元、威穎、芊瑀在實驗室的幫忙，有你們加入實驗室變得熱鬧許多。這篇論文能完成一切都感謝生醫暨語言資訊實驗室大家的協助。最後也感謝我的家人以及女朋友的支持，謝謝你們無條件的付出與包容，讓我能在求學之路上無後顧之憂，全心全意的精進向上。

最後，將此論文獻給我摯愛的家人，並再次感謝這一路上幫助過我的所有人，誠摯的祝福你們身體健康，一路順風。

蘇文弘 謹誌於

國立成功大學

中華民國 110 年 7 月

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background

In recent years, Next Generation DNA sequencing (NGS) has become an essential part of Genomics; with its high-throughput and scalability used in countless research studies.

Many of these NGS-based studies depend on the accurate variants detection and genotype calling of variants from sequence data [1].

Following the development of NGS technologies, variant calling software and approaches have seen rapid development over the past 10 years. There have many variant caller and algorithm developed for variant calling, such as GATK [2], Strelka2 [3], Freebayes [4], DeepVariant [5] and etc.

Generally, typical variant calling process include sequencing, quality control, mapping, marking duplicates & sort & merge, followed by variant calling, filtering and annotation [6].

Although 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 variant calling pipelines [7,8] and with different alignment tools [9,10]. Research indicates that the results from different vari-

ant calling pipelines to agree only on a small part of the variation [11], and indels(insertion/deletion) are still difficult for any pipeline to handle with [12]. However, indels are supremely important as they are considered to have strong impact on cancer [13] and phenotype [14].

1.2 Motivation

We recently developed a variant evaluator EAGLE [15], a method that uses generative probability models to evaluate the confidence of each entry in a list of candidate genome variants.

However, these variant callers and EAGLE all rely on the alignment of reads to the reference genome (pile-up) in their workflow. In variant calling, an intuitive method is to scan the pile-up, 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 maps reads to similar reference sequence regions, reads containing variants may not be mapped to the correct region. This effect is called reference bias [16]. Reference bias is expected to be a source of false negatives (failure to call the true variant), especially for longer indel variants.

Compared with tools that require a strong dependence on the pile-up in the process of variant calling, EAGLE reduces the dependence on the pile-up in the computation process through considering a large number of possible read alignments across the

genome. But EAGLE still relies on the pile-up to identify reads relevant to a given variant, and therefore potentially suffers from some reference bias.

1.3 Research Objectives

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

Based on our read data and some previous studies, we believe that BWA-MEM will show better performance [9], 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 [17] for the reference genome with the 'index' command. The FM-index compressed substring index in based on the Burrows-Wheeler Transform (BWT) [18] and suffix array, generating a fast and efficient index for querying.

Then in the core alignment algorithm, BWA-MEM uses the 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 [15] is our previous research, based on a generative probabilistic model, variant calling evaluator, EAGLE main function is given individual sequencing data (exome or whole genome sequencing) and a list of candidate variants, and returns a likelihood score of each candidate variants. The concept of EAGLE is that 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).

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

2.2.1 EAGLE computation flow

At a technical level, EAGLE walks through two data files while scanning the genome. The first file is a so-called BAM file, holding the information of reads mapped to one or more positions in the genome. The second file is a so-called VCF file holding candidate genome variants. Conveniently both of these file formats are sorted by genome position. Our modified version of EAGLE maintains this structure and flow,

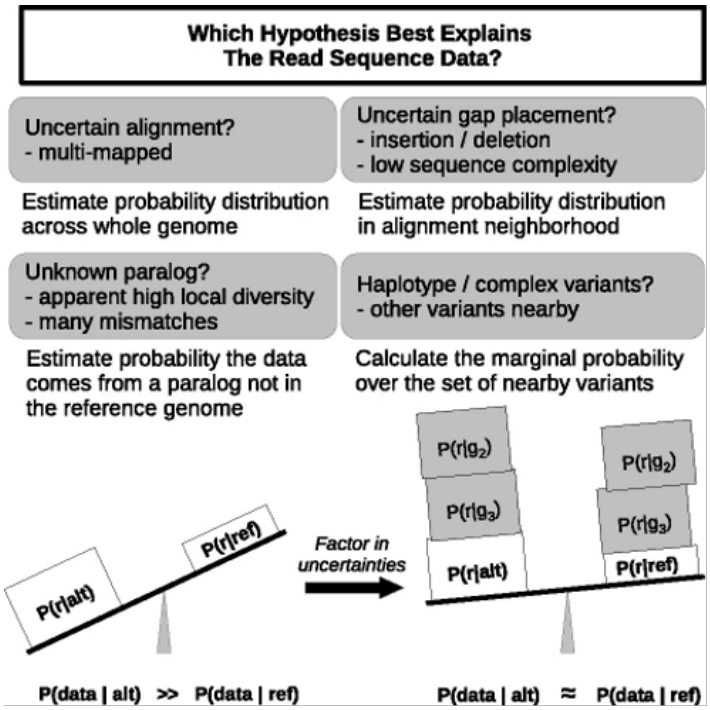

Figure 2.1: EAGLE uses a generative probabilistic model to handle uncertainty. (Figure reproduced from Kuo et al. [15])

but allows consideration of some relevant reads not present in the pileup.

2.3 Reference Bias

Sequencing data analysis often begins with aligning reads to a reference genome, but this sequence only contains the reference allele at polymorphic sites [19], which leads to reference bias (Figure 2.2). Reads that contain alternate alleles tend to misalignment or incorrect alignments [20], so at heterozygous site, the reads containing the reference allele usually constitute the majority of the mapped reads. These problems can cause false negative or false positive variant calls [21]

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. [22]

There have some studies discuss how to mitigate reference bias, such as genome graph aligners [23,24], or using high-coverage NGS data, to 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 added to EAGLE to reduce reference bias; explain in detail how we create our hypothetical sequence based on the variant and how to quickly search through an FM-index to obtain those missing reads, and finally add these reads to the pile-up that 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. Combine variant information in a VCF file and the reference genome to create a hypothetical sequence which includes individual differences.

2. Build a *read-index*, an FM-index of the read sequences.

3. Use BWA and the read-index to find the reads that match the variant, and then check if they already exists in the pile-up.

4. For any reads not found in the pile-up, create that read's alignment information mapped against reference sequences

5. Add read and alignment information to the pile-up, to improve the EAGLE calculation.

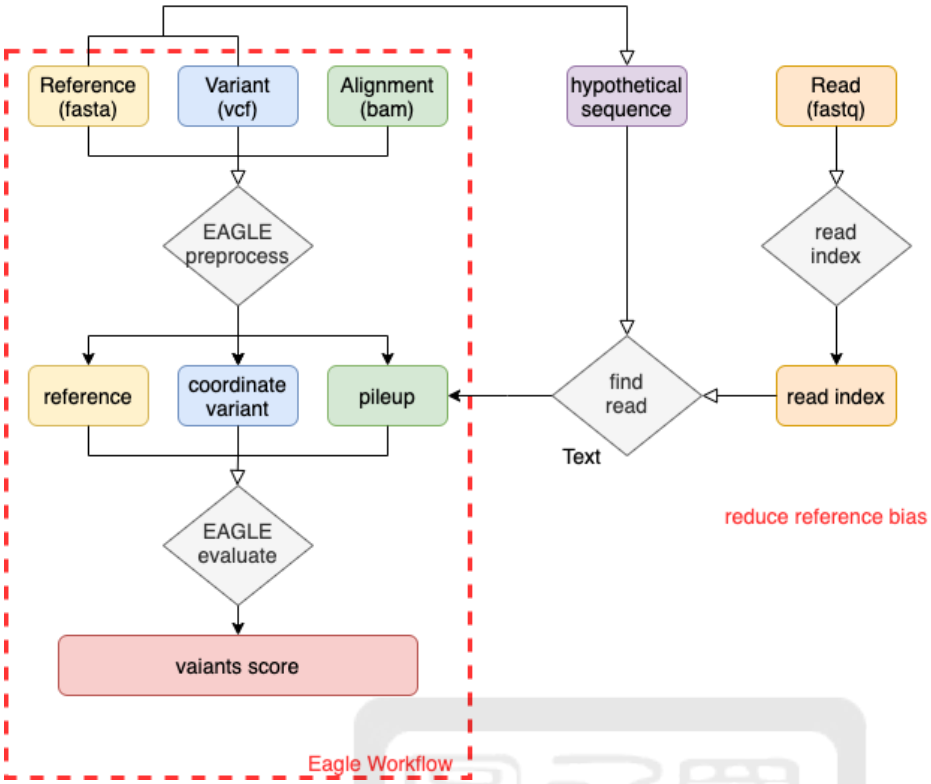

Figure 3.1: The core workflow of EAGLE after adding reference bias reduction.

3.2 Hypothetical Sequence

To reduce the impact of reference bias, the first step we need to construct a hypothetical sequence. Before constructing hypothetical sequence we need variation data and a reference genome. VCF is the standard file format for storing variation data, used by most variant callers (Figure 3.2). FASTA is reference genome format (Figure 3.3), and 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. Because we simulate the occurrence of mutation of the reference sequence by combining variant, it includes individual differences. Thus, bases 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), we generate a hypothetical sequence.

For each hypothetical sequence, according to the variant position, we 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 get all the reads we are interested in and also includes 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.

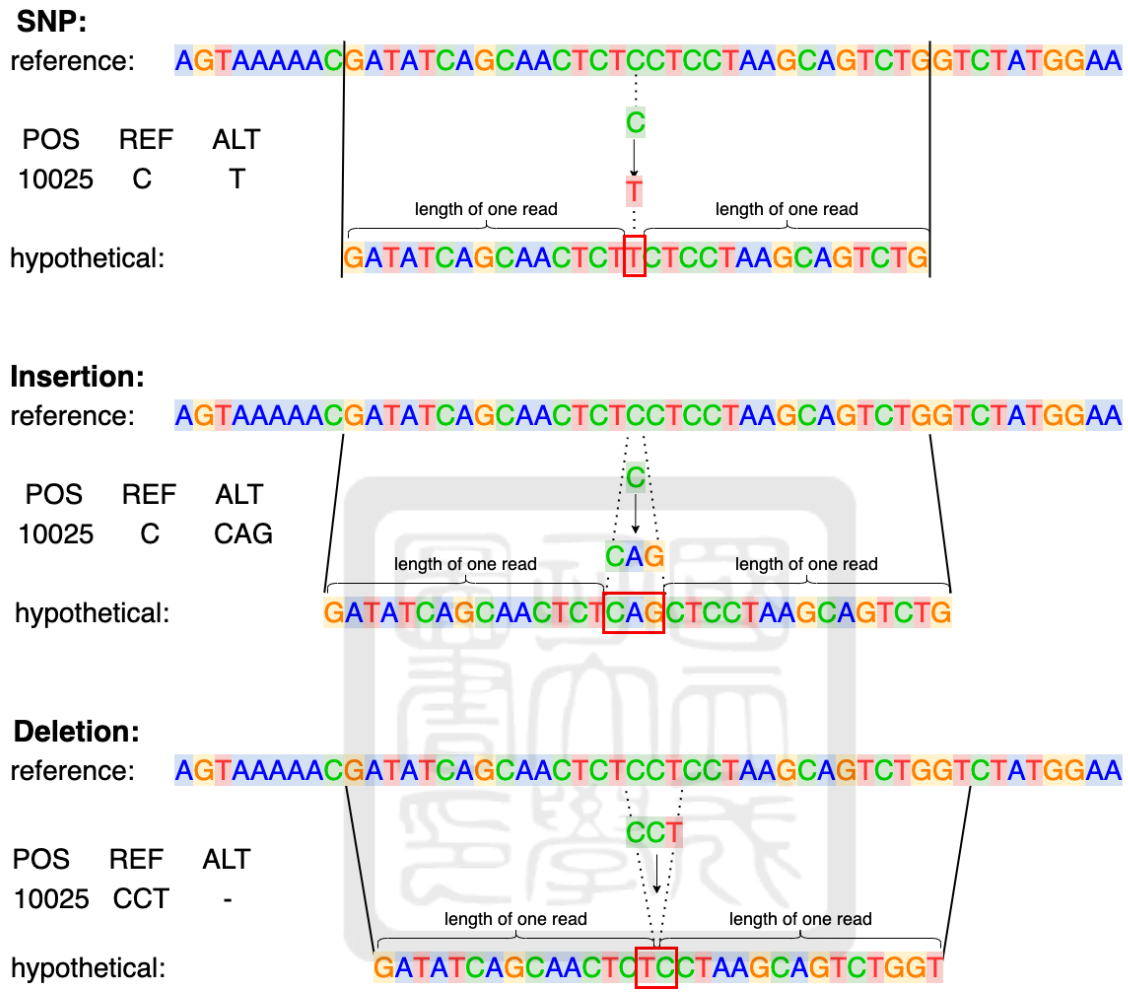

Figure 3.4: Three different cases to construct hypothetical sequence.

3.3 Read-index

Following is an important step for us to find the read that matches our hypothetical sequence. However, if we simply search for matching reads in the FASTQ file, it will cost us a lot of time.

First we describe general practice: the supposed genome sequence is the reference

genome, and we need to map reads on this, just like alignment. Therefore we need to construct index for each hypothetical sequence and then query all the read in FASTQ file. As we mentioned above, we need $O(n \log n)$ (n is reference sequence length) to construct BWT index, and querying will cost $O(\ell)$ (ℓ is read length) for each read.

In general situation, the time spent to build an index and perform queries is:

$$O(n \log n) + O(\ell)$$

It is indeed a very efficient method for standard applications. 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 genome sequence is not constant. 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.

What would the situation will be 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 sequence and do a complete read search. It will take time:

$$\sum_{h \in H} \Big\{ \text{build index on } h + \sum_{r \in R} \text{query } r \text{ on that index} \Big\}$$

where H denotes the set of hypothetical sequences (one for each variant) and R denotes the set of reads.

Since the hypothetical sequences are very short compared to the combined length of all reads, the time spent on index building can be ignored. Query time is linear in the total length of the queries so the time needed is approximately:

$$\sum_{h \in H} \Big\{ \sum_{r \in R} \text{query } r \text{ on index} \Big\} = |H| \times \ell \, |R|$$

Where $|H|$ denotes the number of hypothetical sequences, ℓ the read length, and $|R|$ the number of reads.

On the other hand, when building an index on the reads instead of the hypothetical sequences

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

$$\text{build index on } R + \sum_{h \in H} \text{query } h \;\approx\; \ell \, |R| \;\log(\ell |R|) + 2\ell \, |H|$$

Which is much faster.

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

3.4 Find Similar Reads

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

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 chrom1 GATAGTAGTTGCAGCAGGCTAGCTA	@SEQ_ID GATAGTAGTTGCAGCAGGCTAGCTA + ???DB+=2C>??E;>C>??EEEFF;

Like EAGLE, BWA is an open-source software implemented in C. 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 meet 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), but for our purposes we simply want to collect all reasonable matches (Figure 3.6).

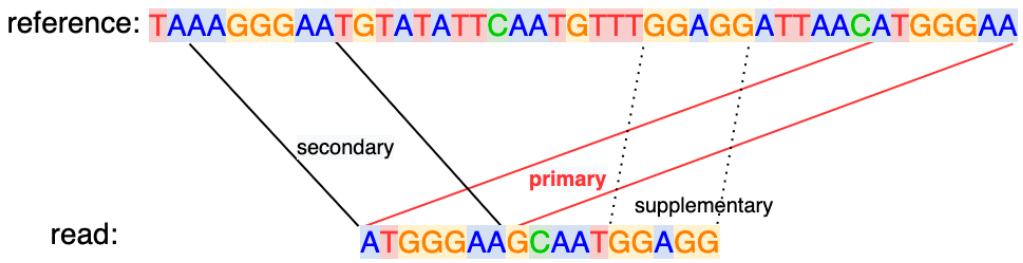

Figure 3.5: BWA secondary alignment, supplementary alignment.

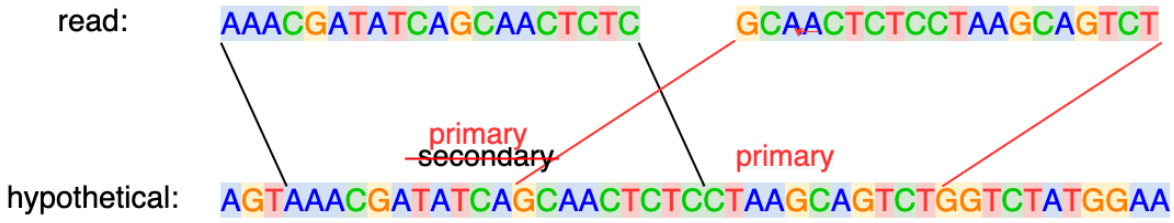

Figure 3.6: The read index is used to find reads matching a hypothetical genome sequence. BWA marks some matches as primary, etc., but we treat them all the same.

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.

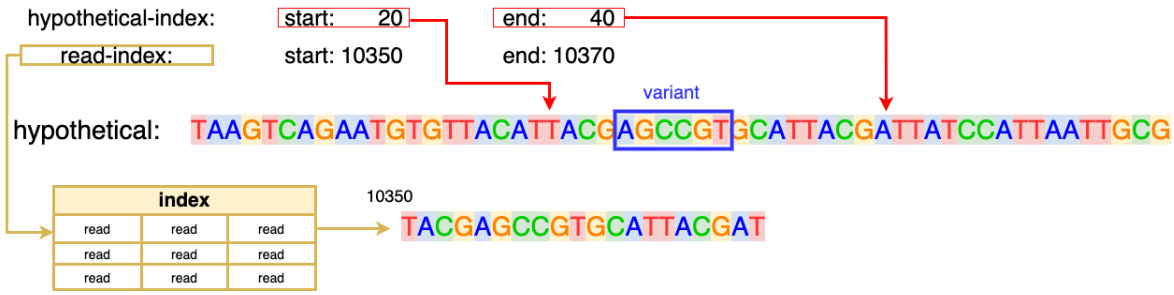

Figure 3.7: how to find matching read through read-index.

Through the index of hypothetical sequence, we can check whether it contains the

variant, if not, we will filter it, as Figure 3.8 shows. At the end of this step, we can obtain a set of reads meeting our requirements, and then add them to the pile-up.

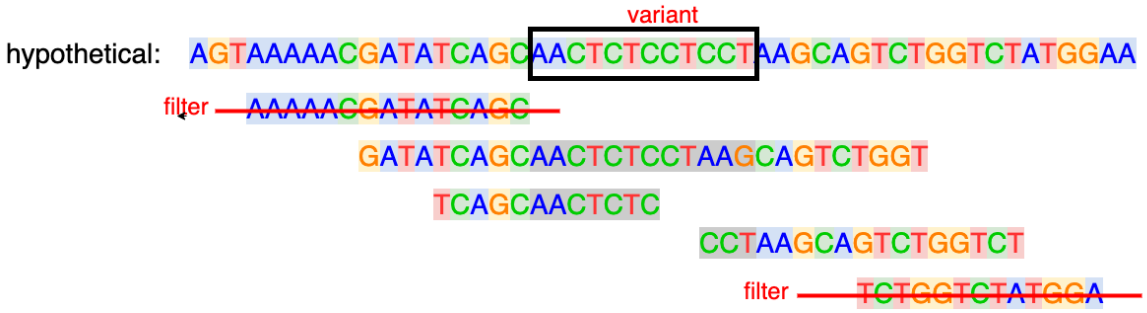

Figure 3.8: Filter reads that don't overlap the variant.

3.5 Integrating new reads into the pile-up

In the last step, we need to compare all the read sets obtained in the previous step with the reads in the pile-up. What is the pile-up? We start with a standard read mapping result stored in BAM file format. We can observe that reads is mapped to a position in the reference genome. Many reads may align to the same region, and this "pile" of reads is called the pile-up. Our final goal is to find reads that overlap with the same variant position but do not exist in the BAM file, (Figure 3.9 shows the pile-up reads which overlap on a variant position.)

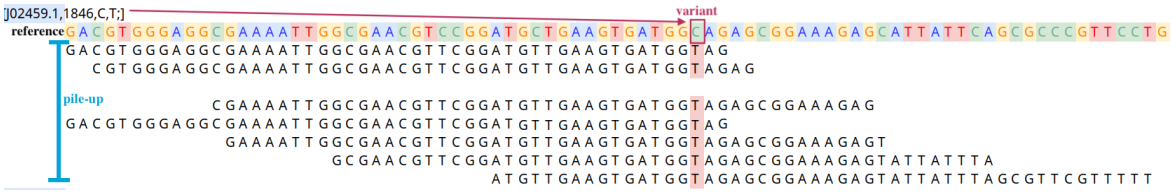

Figure 3.9: pile-up at the variant position.

We can get the pile-up from a BAM file quickly through the utility library `HTSlib`, and then compare the read set with these pile-up reads, if the read does not exist in

the pile-up, it means that we have got the results we want. The 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, including SAM flag and CIGAR string, SAM flag is represents the mapping status of this read and the CIGAR string encodes the detailed alignment of the read. We will explain how we get CIGAR string through two sets of indexes.

If we get the SAM flag and CIGAR directly through our two sets of indexes, what we get is the alignment information of hypothetical sequence mapping to read, like previous Figure 3.7 shows, but what EAGLE needs is the alignment information of read mapping to reference sequence. So we need do some processing on the match positions.

First of all, from the position of matches to in read-index, we can directly obtain matching reads. The other match position is relative to the start and end of our hypothetical sequence — a length 2ℓ sequence containing a candidate variant in the middle. We need to convert positions in this sequence into positions in the reference genome to help us add the read to the pile-up in such as way that EAGLE can consider it when evaluating that variant.

This problem can be divided into 4 cases(Figure 3.10):

1. the tail of read sequence overlaps with the variant.
2. the front of read sequence overlaps with the variant.
3. the center of read sequence overlaps with the variant.
4. all of read sequence overlaps with the variant.

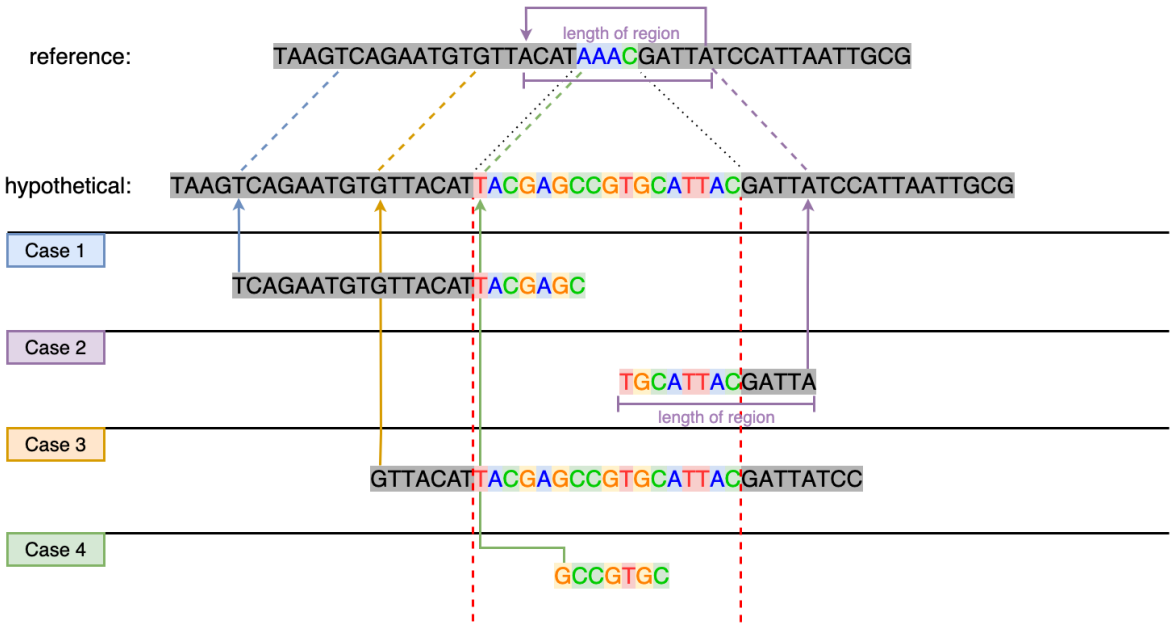

Figure 3.10: Four cases related read to variant. A read may overlap with part of a variant (left or right side), may completely include a variant, or be included by the variant.

For the first case and third case we can just directly get the reference position, and fourth case we use the position of variant. In the second 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 in the 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 the read's alignment information mapped against reference sequences (Figure 3.11). We use a small function we implemented to construct a CIGAR string for these reads.

Figure 3.11: How to convert CIGAR strings.

Through these steps we can add read to the pile-up for eagle to calculate likelihood.

Chapter 4

Experiment and Results

In this section, first we introduce dataset and some tools, then we will explain the steps of our experiment, analyze the experimental results in three different simulation scenarios, and finally discuss results using the VCF file from dbSNP divided it into SNPs and indels.

4.1 Dataset

EAGLE needs three main input files: the reference genome (FASTA), read data (FASTQ), and a variant file (VCF).

First, the reference genome sequencing data we used is Genome Reference Consortium Human Build 37 patch release 13 (GRCh37.p13) version from NCBI website. It includes autosomal chromosomes 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 nucleotide substitutions (also called single nucleotide polymorphisms or SNPs), and small-scale base deletions

or insertions (indels).

Finally, we will also use a simulated VCF file, the variant file is generated by our simulation, we provide more details below.

4.2 Simulation workflow

In order to carefully evaluate the effect of our modifications to EAGLE 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 occurrence of differences between a reference genome and an individual genome (Figure 4.1), and record the result of our modification to generate a VCF file. We refer to variants simulated in this way as "planted variants" and the resulting modified reference as the "doctored reference". An advantage of this approach is real read data can be used. A disadvantage is that only homozygous variants can be simulated.

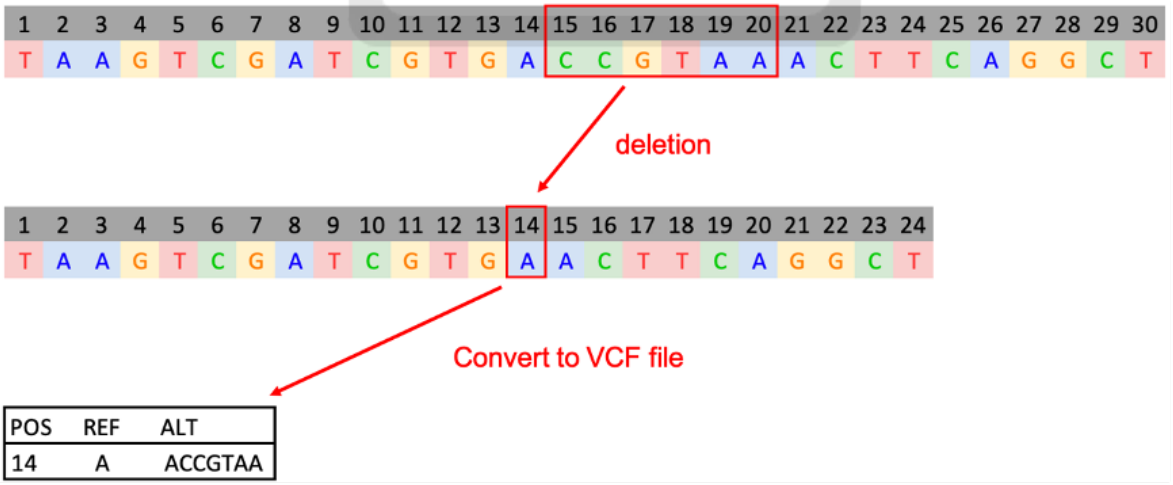

Figure 4.1: Simulate variants.

Then we will use BWA, following the traditional approach, and do the read mapping

alignment again, to obtained a doctored reference pile-up. Use the re-alignment step to simulate the occurrence of reference bias, because we modified the reference, when re-aligning, the pileup at the original position will have some reads affected by the change we made to the reference, and these affected reads will be misaligned or perhaps mapped elsewhere. After generating the files needed by EAGLE, we can start our experiment. The entire experiment process is outlined in Figure 4.2.

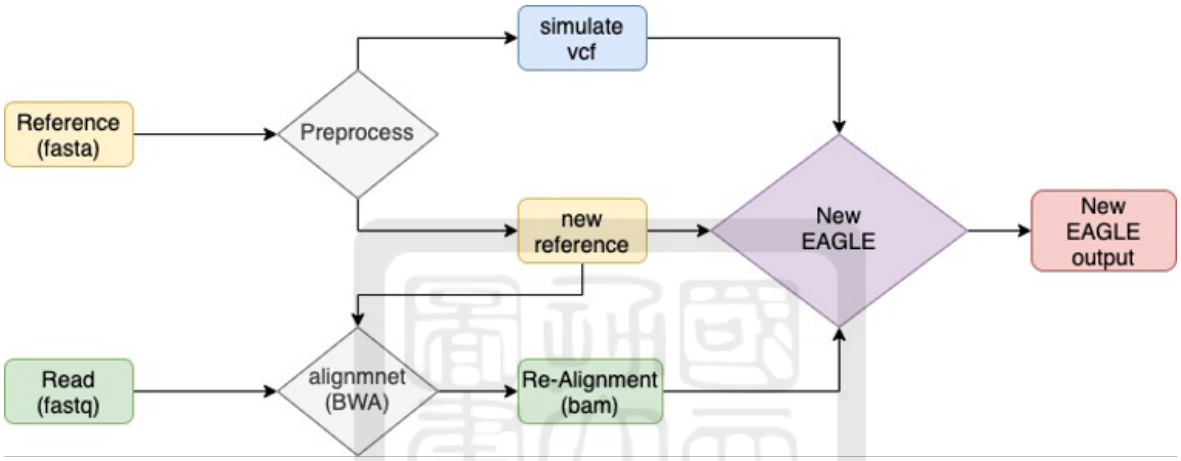

Figure 4.2: Simulation workflow.

In order to analyze the affect of read depth on our results, we used SAMtools to check the original reference pile-up read depth, and divide genome positions into three categories of pileup read depth: 5–20x, 20–40x and >40x (Table 4.1).

Table 4.1: total simulate indels amount

Pile-up Read Depth	**5x – 20x**	**20x – 40x**	**> 40x**
Indel length	1 – 50	1 – 50	1 – 50
Amount	20 for each	20 for each	20 for each
Total variants	1000	1000	1000

For each read depth category, and each length l from 1 to 50, we randomly selected

20 positions and deleted a string of l base pairs from the standard reference genome to obtain a doctored reference genome. We do this to simulate insertion variants, as the read data (which is real and presumably matches the standard reference genome in most positions) should contain the sequences we deleted from the reference. While examining the affect of indel length on our results we will discuss the results after re-alignment in two situations, one is that after re-alignment to the doctored reference, BWA still aligns some reads to the planted variant position, and the other is that after re-alignment, no read can be aligned to that position.

In the first case, the original 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 new results calculated by EAGLE.

Our evaluation is based on the log odds ratio output of EAGLE of each variant v:

$$\log \frac{\mathrm{P}[\mathrm{Alt}|v]}{\mathrm{P}[\mathrm{Ref}|v]}$$

4.3 Case 1: Low read depth

The first two rows of Table 4.2 show how often the EAGLE likelihood ratio changes after adding any new reads from our read index into consideration. We can see that in the case of low read depth, since only a small number of reads are mappable at all (even against the original reference sequence), once there is a reference bias (simulated by mapping against the doctored reference), EAGLE's evaluation can easily be severely affected. As expected, you can also see that the longer the length of indels, the more

serious the impact.

Table 4.2: low read depth variants

Indel Length	**1–10**		**11–20**		**21–30**		**31–40**		**41–50**		**Total**
Unchanged	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

Next, let's take a look at the changes after adding our read-index based method. As shown in table 4.3, EAGLE predicts that the position does not contain variants when only using the doctored reference pile-up; but in most cases (19 out of 23), when we find some reads matching the variants, we can help EAGLE judge whether the position contains variants.

Table 4.3: Number of planted variants missed at low read depth

Indel length / $\frac{\mathbf{P}[\text{Alt}]}{\mathbf{P}[\text{REF}]} < 1$	**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 that were misaligned due to reference bias. Figure 4.3 is one of the cases.

Figure 4.3: Example of new reads supporting a variant in a region with low pile-up read depth. The original reference genome is displayed at top, with a red bracket indicating the variant sequence deleted in our doctored reference.

Similarly, especially in repetitive genome regions, there are some new reads which are also similar to the reference sequence (Figure 4.4).

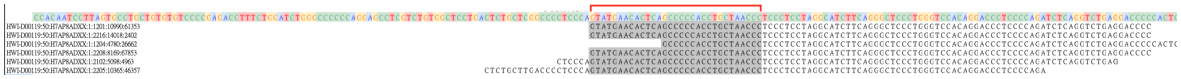

Figure 4.4: Example where new reads found are also similar to the reference sequence with low pile-up read depth. The original reference genome is displayed at top, with a red bracket indicating the variant sequence deleted in our doctored reference. Note the repetitive nature of the surrounding genome sequence.

Next, we can look at the comparison of EAGLE log odds ratio.

$$\text{log odds ratio change} = \text{log odds(after)} - \text{log odds(before)}$$

to represent the effect of our new method on EAGLE candidate variant evaluation.

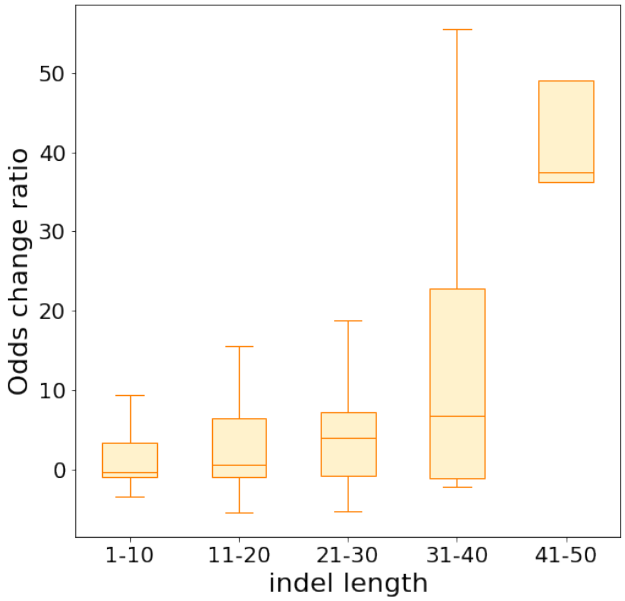

Figure 4.5: The vertical axis shows the change in EAGLE log odds ratio of variant to reference for indel variants, grouped by length (horizontal axis). Change here means the difference in the log odds ratio when using the read index versus only using the pile-up. Plotted for variants from low pile-up read depth regions.

In the case of low pile-up read depth, first of all, we can observe that after we add reads, the confidence value of most of the variants increases because we find more reads that contain the variant sequence (Figure 4.5), and regardless of the length of the indel, their distribution of logs odds ratio change is mostly above zero. This may mean that even in the case of low pile-up read depth, 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 4.3, 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.

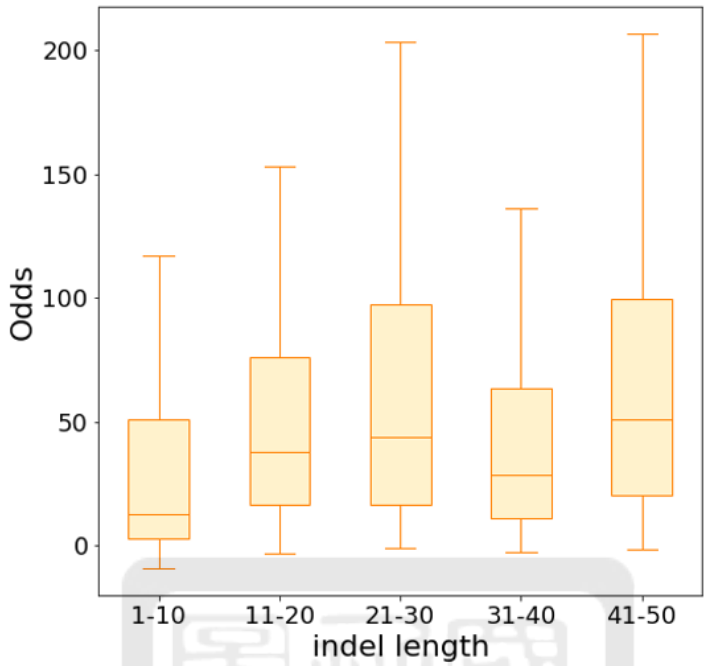

Figure 4.6: The vertical axis shows the EAGLE log odds ratio of variant to reference for indel variants, grouped by length (horizontal axis). Plotted for variants occurring in regions with no reads from low pile-up depth.

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 the variant occurs at this position, which is also in line with our expected results.

4.4 Case 2: Medium pile-up read depth

First of all, we can see that in this case, Table 4.4, because of the increase in the original pile-up read depth, the variants which cannot be called due to reference bias

Table 4.4: medium pile-up read depth variants

Indel Length	**1–10**		**11–20**		**21–30**		**31–40**		**41–50**		**Total**
Unchanged	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

are significantly reduced, and the number of reads that we can overlap with the variant position also increased. For the same reason, more misaligned reads are affected by reference bias.

Table 4.5: Changes in the number of variants with medium pile-up read depth

Indel length / $\frac{\mathbf{P}[\text{Alt}]}{\mathbf{P}[\text{REF}]} < 1$	**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 pile-up read depth, In EAGLE's initial judgment, the computed likelihood that position does not contain the 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 misaligned reads, the affect on variant evaluation has also increased, nearly 91% of variants in Table 4.5 have been changed to the correct situation. Figures 4.7 and 4.8 show some examples.

Figure 4.7: Example of new reads supporting a variant in a region with medium pile-up read depth. The original reference genome is displayed at top, with a red bracket indicating the variant sequence deleted in our doctored reference.

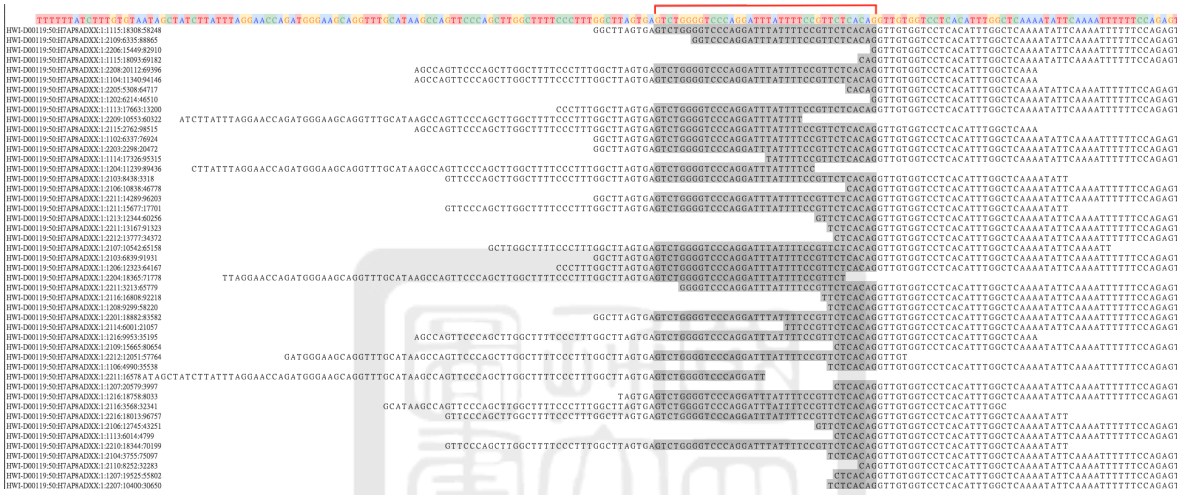

Figure 4.8: Example where new reads found are also similar to the reference sequence with medium pile-up read depth. The original reference genome is displayed at top, with a red bracket indicating the variant sequence deleted in our doctored reference. Note the repetitive nature of the surrounding genome sequence.

We can find that in bad cases, although we found reads similar to hypothetical variant 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).

Figure 4.9: The pile-up of reads aligned to the doctored genome for the variant from Figure 4.8. At top, the doctored reference genome is shown with an arrow indicating where the simulated variant sequence was deleted from the original reference.

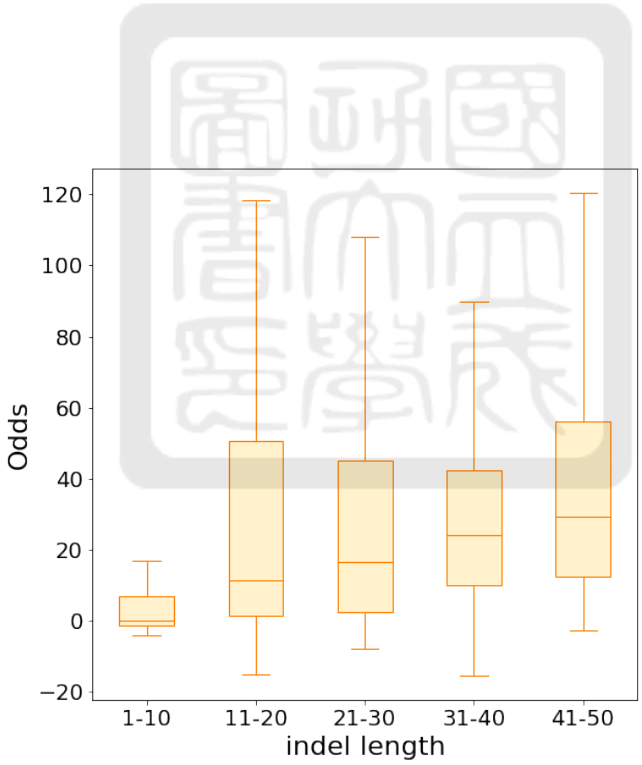

Figure 4.10: The vertical axis shows the change in EAGLE log odds ratio of variant to reference for indel variants, grouped by length (horizontal axis). Change here meaning the difference in the log odds ratio when when using the read index versus only using the pile-up. Plotted for variants from medium pile-up read depth regions.

Next, we can look at the comparison of EAGLE odds change for medium pile-up read depth (Figure 4.10). Basically, the change is similar to the situation of low pile-up read depth. We can also see larger change for longer indels. The difference is that we also have a good performance in the shorter indel. And the average change is better than that of low pile-up read depth. 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 box plot have Q3 greater than 0, which means that most of our reads can help us eliminate the influence of reference bias.

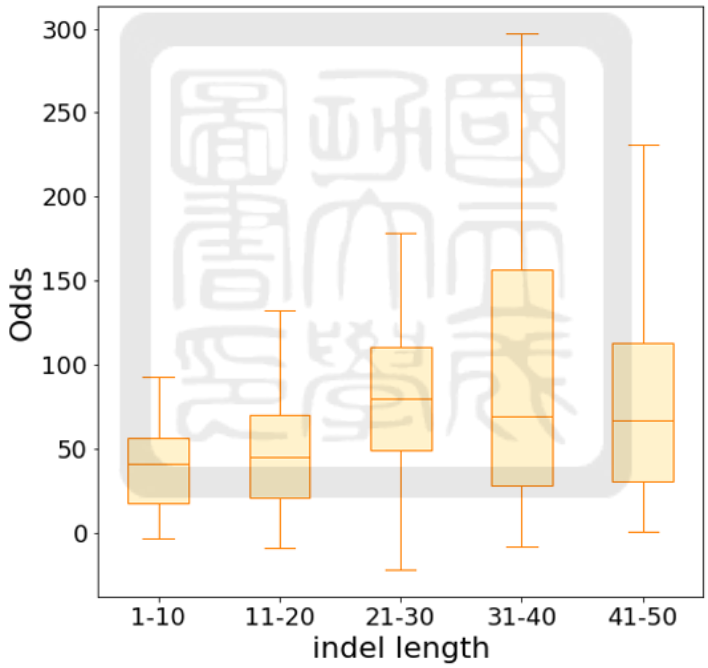

Figure 4.11: The vertical axis shows the EAGLE log odds ratio of variant to reference for indel variants, grouped by length (horizontal axis). Plotted for variants occurring in regions with no reads from medium pile-up depth.

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 make more correct conclusions (that the position does contain variants) because we can find reads supporting the variants. It can be seen that compared to the previous case, regardless of the length of the indel, their Q3 is some distance higher, this result is also as we predicted (for example see Figure 4.12).

Figure 4.12: Reads found via the read-index are shown for a position for which standard read mapping to the doctored reference found no matching reads (an empty pile-up). At top, the original reference genome sequence is shown.

4.5 Case 3: High pile-up read depth

Table 4.6: High pile-up read depth variants

Indel Length	1–10		11–20		21–30		31–40		41–50		Total
Unchanged	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	80%	605
New	26		17		16		100		120		279

Basically, in this case, we will observe results that are very similar to medium pile-up read depth, that's because there are quite a lot of reads, most positions can still

produce pileup (Table 4.6).

Table 4.7: Changes in the number of variants with high pile-up read depth

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	1	0	2

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

Figure 4.13: Example of find large amount new reads supporting a variant in a region with high pile-up read depth. The original reference genome is displayed at top, with a red bracket indicating the variant sequence deleted in our doctored reference.

Figure 4.14: Pileup after adding reads to high pile-up depth region. At top, the doctored reference genome is shown with an arrow indicating where the simulated variant sequence was deleted from the original reference.

Then, we look at the comparison of EAGLE odds in high pile-up read depth (Figure 4.15). It can be seen that compared with the previous example, it is very obvious that the longer the indel, the bigger the increase. 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 would have been lost, the better we can reduce the impact of reference bias.

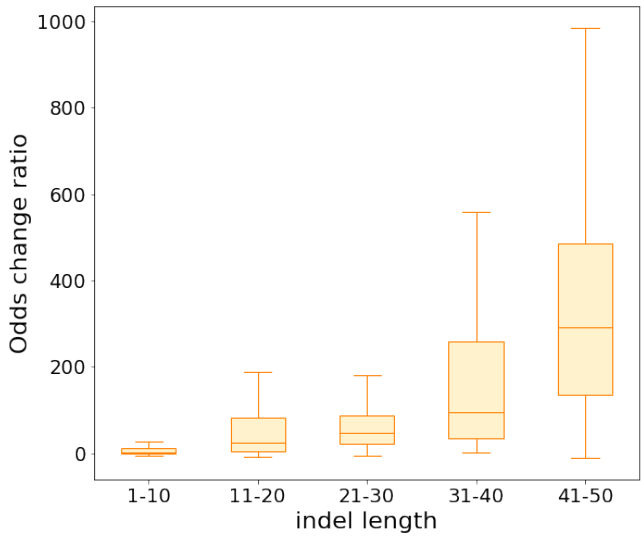

Figure 4.15: The vertical axis shows the change in EAGLE log odds ratio of variant to reference for indel variants, grouped by length (horizontal axis). Change here meaning the difference in the log odds ratio when when using the read index versus only using the pile-up. Plotted for variants from high pile-up read depth regions.

Finally, we look at the last row of Table 4.6, we can see the same result as the previous case medium pile-up read depth. Figure 4.15 shows these variant odds calculated by EAGLE.

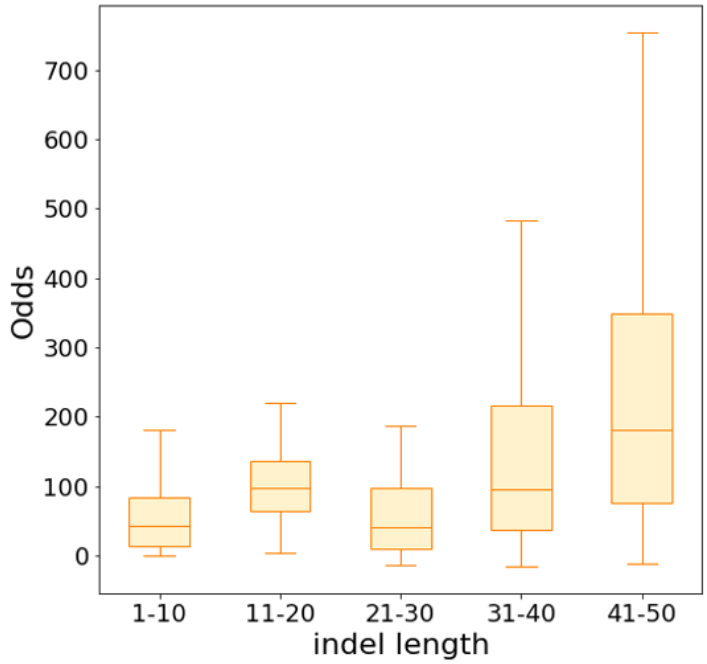

Figure 4.16: The vertical axis shows the EAGLE log odds ratio of variant to reference for indel variants, grouped by length (horizontal axis). Plotted for variants occurring in regions with no reads from high pile-up depth.

4.6 SNPs in dbSNP dataset

There are 15,936,832 SNP variants form the dbSNP 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: Reads found via the read-index are shown for a SNP in dbSNP.

Figure 4.18: The pileup corresponding to the Figure 4.17 candidate variant.

Figure 4.19: Reads found via the read-index are shown for a SNP in dbSNP. In this particular case, those reads actually match the reference genome better.

Figure 4.20: The pileup corresponding to the Figure 4.19 candidate variant.

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.

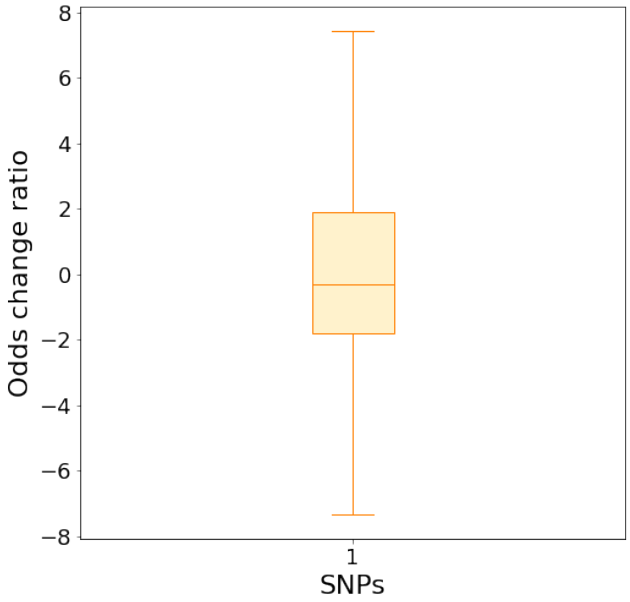

Figure 4.21: SNPs odds change ratio.

4.7 INDELs in dbSNP dataset

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

Figure 4.22: Matching reads with hypothetical sequence in INDEL

Figure 4.23: The pileup corresponding to the Figure 4.22 variants.

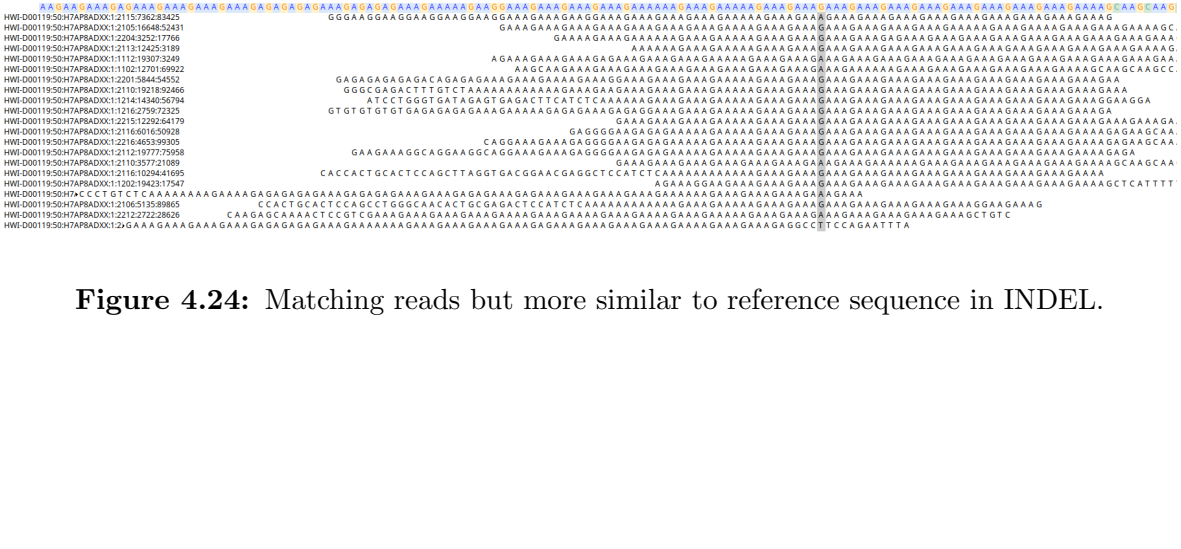

Figure 4.24: Matching reads but more similar to reference sequence in INDEL.

Figure 4.25: The pileup corresponding to the Figure 4.24 variants.

It can be seen that the reads we found gave EAGLE more evidence for judgment, and that the magnitude of change in INDEL was significantly greater than SNP, just like the experiment we simulated before.

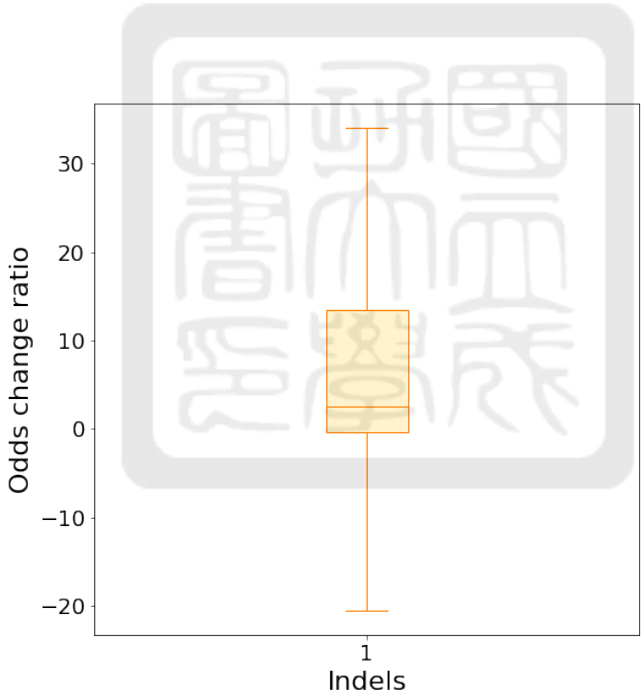

Figure 4.26: INDEL odds change ratio.

4.8 Execution time and memory consumption

Finally, let's discuss our execution time and memory consumption, our test data are as follows Table 4.8, 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 version	Garvan NA12878 HG001 HiSeq Exome	Single Nucleotide Polymorphism Database (dbSNP)
File Size	3.1G	5G	669.7MB

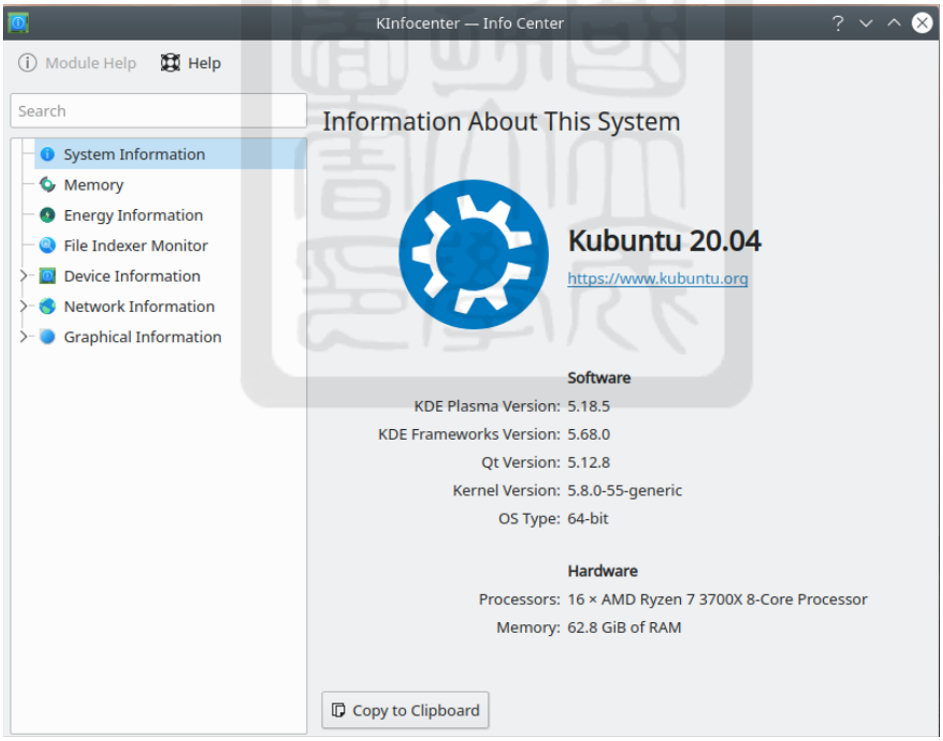

Figure 4.27: Test Environment.

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, but the time to 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.

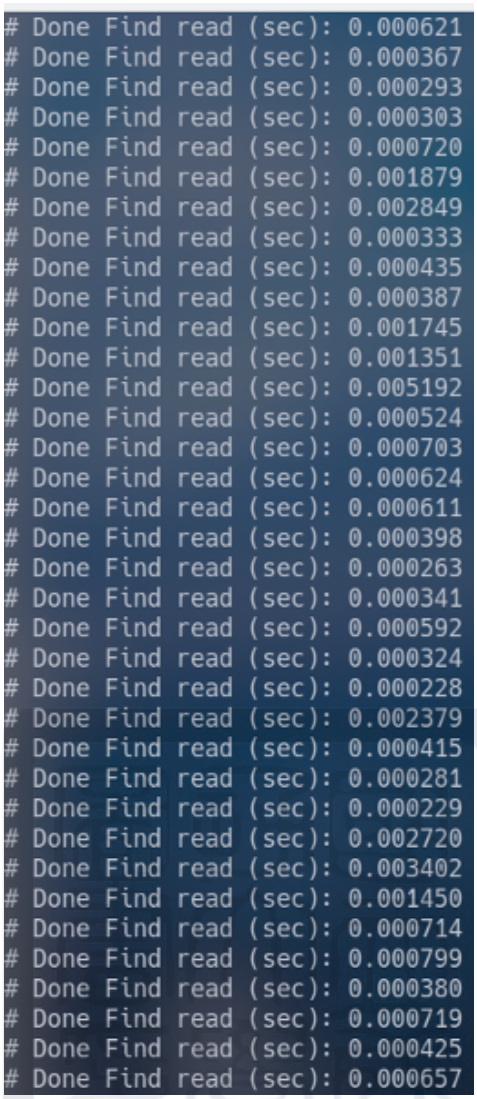

Figure 4.28: searching per hypothetical sequence time.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

In order to reduce the influence of reference bias and improve the accuracy of EAGLE for evaluating variants, we have expanded the function of EAGLE. While the original version of EAGLE relies on a standard read mapping (pile-up) to find reads relevant to a candidate variant, we use a read-index to find additional reads which match the variant genome sequence (but not necessarily match the reference genome sequence). We modified EAGLE to allow us to add these reads to the pile-up before EAGLE evaluates the candidate variant.

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 for longer indels the effect is particularly significant. At the same time, our modified EAGLE we can work normally in any read coverage situation.

We also examined dbSNP, a real data set of candidate variants. We observed good results as before, although the effect on SNPs calls is relatively limited. Our implementation is well integrated into EAGLE and can reduce reference bias by missing fewer relevant reads. Our use of a read-index requires only a modest time overhead. The additional memory required is significant, but still manageable.

5.2 Future Work

This thesis focused on comparing the results of EAGLE before and after using a read-index to consider more reads. As future work, our EAGLE with read-index method should also be compared to other variant candidate evaluation methods on more comprehensive benchmark datasets.

REFERENCES

[1] Rasmus Nielsen, Joshua S Paul, Anders Albrechtsen, and Yun S Song. Genotype and snp calling from next-generation sequencing data. *Nature Reviews Genetics*, 12(6):443–451, 2011.

[2] Ryan Poplin, Valentin Ruano-Rubio, Mark A DePristo, Tim J Fennell, Mauricio O Carneiro, Geraldine A Van der Auwera, David E Kling, Laura D Gauthier, Ami Levy-Moonshine, David Roazen, et al. Scaling accurate genetic variant discovery to tens of thousands of samples. *BioRxiv*, page 201178, 2018.

[3] Christopher T Saunders, Wendy SW Wong, Sajani Swamy, Jennifer Becq, Lisa J Murray, and R Keira Cheetham. Strelka: accurate somatic small-variant calling from sequenced tumor–normal sample pairs. *Bioinformatics*, 28(14):1811–1817, 2012.

[4] Erik Garrison and Gabor Marth. Haplotype-based variant detection from short-read sequencing. *arXiv preprint arXiv:1207.3907*, 2012.

[5] Ryan Poplin, Pi-Chuan Chang, David Alexander, Scott Schwartz, Thomas Colthurst, Alexander Ku, Dan Newburger, Jojo Dijamco, Nam Nguyen, Pegah T Afshar, et al. A universal snp and small-indel variant caller using deep neural networks. *Nature biotechnology*, 36(10):983–987, 2018.

[6] Daniel C Koboldt. Best practices for variant calling in clinical sequencing. *Genome Medicine*, 12(1):1–13, 2020.

[7] Xiaopeng Bian, Bin Zhu, Mingyi Wang, Ying Hu, Qingrong Chen, Cu Nguyen, Belynda Hicks, and Daoud Meerzaman. Comparing the performance of selected

variant callers using synthetic data and genome segmentation. *BMC bioinformatics*, 19(1):1–11, 2018.

[8] Jiayun Chen, Xingsong Li, Hongbin Zhong, Yuhuan Meng, and Hongli Du. Systematic comparison of germline variant calling pipelines cross multiple next-generation sequencers. *Scientific reports*, 9(1):1–13, 2019.

[9] Sohyun Hwang, Eiru Kim, Insuk Lee, and Edward M Marcotte. Systematic comparison of variant calling pipelines using gold standard personal exome variants. *Scientific reports*, 5(1):1–8, 2015.

[10] Sen Zhao, Oleg Agafonov, Abdulrahman Azab, Tomasz Stokowy, and Eivind Hovig. Accuracy and efficiency of germline variant calling pipelines for human genome data. *Scientific reports*, 10(1):1–12, 2020.

[11] Rui Tian, Malay K Basu, and Emidio Capriotti. Computational methods and resources for the interpretation of genomic variants in cancer. *BMC genomics*, 16(8):1–19, 2015.

[12] Adam Cornish and Chittibabu Guda. A comparison of variant calling pipelines using genome in a bottle as a reference. *BioMed research international*, 2015, 2015.

[13] Jennifer K Sehn. Insertions and deletions (indels). In *Clinical genomics*, pages 129–150. Elsevier, 2015.

[14] Stephen B Montgomery, David L Goode, Erika Kvikstad, Cornelis A Albers, Zhengdong D Zhang, Xinmeng Jasmine Mu, Guruprasad Ananda, Bryan Howie, Konrad J Karczewski, Kevin S Smith, et al. The origin, evolution, and func-

tional impact of short insertion–deletion variants identified in 179 human genomes. *Genome research*, 23(5):749–761, 2013.

[15] Tony Kuo, Martin C Frith, Jun Sese, and Paul Horton. Eagle: explicit alternative genome likelihood evaluator. *BMC medical genomics*, 11(2):1–10, 2018.

[16] Vitor Sousa and Jody Hey. Understanding the origin of species with genome-scale data: modelling gene flow. *Nature Reviews Genetics*, 14(6):404–414, 2013.

[17] Paolo Ferragina and Giovanni Manzini. Indexing compressed text. *Journal of the ACM (JACM)*, 52(4):552–581, 2005.

[18] Michael Burrows and David Wheeler. A block-sorting lossless data compression algorithm. In *Digital SRC Research Report*. Citeseer, 1994.

[19] Rui Martiniano, Erik Garrison, Eppie R Jones, Andrea Manica, and Richard Durbin. Removing reference bias and improving indel calling in ancient dna data analysis by mapping to a sequence variation graph. *Genome biology*, 21(1):1–18, 2020.

[20] Nae-Chyun Chen, Brad Solomon, Taher Mun, Sheila Iyer, and Ben Langmead. Reference flow: reducing reference bias using multiple population genomes. *Genome biology*, 22(1):1–17, 2021.

[21] Torsten Günther and Carl Nettelblad. The presence and impact of reference bias on population genomic studies of prehistoric human populations. *PLoS genetics*, 15(7):e1008302, 2019.

[22] Jessica Lau. Reference bias: Challenges and solutions. https://www.sevenbridges.com/reference-bias-challenges-and-solutions/.

[23] Erik Garrison, Jouni Sirén, Adam M Novak, Glenn Hickey, Jordan M Eizenga, Eric T Dawson, William Jones, Shilpa Garg, Charles Markello, Michael F Lin, et al. Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature biotechnology*, 36(9):875–879, 2018.

[24] Heng Li, Xiaowen Feng, and Chong Chu. The design and construction of reference pangenome graphs with minigraph. *Genome biology*, 21(1):1–19, 2020.