A Comparative Analysis of Computational Indel Calling Pipelines for Next Generation Sequencing Data

Jacob Porter, Jonathan Berkhahn, and Liqing Zhang
Computer Science, Virginia Tech, Blacksburg, VA, USA

Abstract - *Insertions and deletions (indels), are one of the most common class of mutations in the human genome. Correctly detecting and identifying indels is important in the study of human genetics and disease. We evaluated the precision and recall of combinations of read mapping and indel calling software on calling short and longer indels with variable read coverage on simulated data. We examined the popular read mappers BFAST, Bowtie2, BWA, and Shrimp and the indel callers Dindel, FreeBayes, and SNVer. Interestingly there were interactions between read mappers and indel callers. On simulated data, the BFAST-Dindel and Shrimp-SNVer pipelines showed superior performance in most cases. Real data from human chromosome 22 with indels determined from an alternative indel pipeline were used to validate the computational pipelines and to assess run-time. The Shrimp-SNVer pipeline was the most accurate, while pipelines with FreeBayes did poorly. We discuss reasons for pipeline accuracy.*

Keywords: indel calling bowtie2 bfast bwa snver

1 Introduction

Indels are the second most common class of mutation in the human genome [1]. They consist of an insertion or deletion of one or more DNA bases into a genome. This can have far ranging effects concerning gene expression and genetic disease [1]. Detecting and identifying indels is a multistep process that can introduce error at every step. Starting with a set of DNA sequence reads and a reference DNA genome, reads are first mapped to the reference genome with a mapping program and then the mapped results are inputted into indel calling software to identify indels.

A growing variety of software is available for read mapping and indel calling. New tools are constantly being developed with an eye towards better performance and increased accuracy. As the variety of available tools and the complexity of the technologies involved in indel calling increases, it becomes increasingly important to understand the relationships between mapping software and indel calling software. While previous work [2, 3] assessed the accuracy of indel calling by only varying mapping software or by only varying indel calling software, we studied the accuracy of mapper and indel calling software combinations. We evaluated the accuracy of pipelines consisting of four popular mapping programs and three indel calling programs on simulated data based on a portion of human chromosome one. For mapping, BFAST [7], Bowtie2 [4], BWA [5], and SHRIMP [6] were selected. For indel calling, we used Dindel [8], Freebayes [10], and SNVer [9]. We varied the coverage of the reads inputted to the mappers to study the effects of different levels of read coverage on the precision and recall of called indels. We evaluated the accuracy of these pipelines on indels from 1-30 bases long.

Furthermore, pipeline accuracy was assessed with real human data from chromosome 22. The indels were validated with an alternate method described in the paper [12].

The remainder of this paper is organized as follows. Section 2 discusses the read mapping software and the indel calling software we selected. It discusses methods we used to generate our simulated and real data sets, and the statistics that we used to evaluate the accuracy of our pipelines. Section 3 discusses the accuracy results of the pipelines and runtime on real data. Section 4 concludes.

2 Methods

2.1 Software Workflow

We selected mapping software that was both widely used and that covered a variety of different algorithms. Bowtie2 [4] and Burrows-Wheeler Aligner (BWA, [5]) were both popular tools that use the Burrows-Wheeler transform to map reads. SHRiMP [6] and BFAST [7] are both hash-based mapping tools. Shrimp creates a hash table index of the read sequences, but BFAST creates a hash table index of the reference sequences. For indel calling, we selected two programs that use Bayesian statistics, Dindel [8] and Freebayes [10]. SNVer [9] is based on a frequentist binomial-binomial model developed by the SNVer authors.

All of our experiments were run on SystemG nodes. SystemG is a research cluster at Virginia Tech. Each node had two quad-core 2.8 GHz Intel Xeon processors and 8 gigabytes of RAM. The mappers were run with four threads when possible, but the indel callers were single-threaded only.

Each tool was run with default settings since that is how the tools will most likely be used. The workflow consisted of creating SAM files from each read-mapper and then

transforming the SAM files into BAM files with samtools. Finally, each indel-caller produced a VCF file from the BAM files. The following are the version numbers for the software: bfast-0.7.0a, bowtie2-2.1.0, bwa-0.7.1, SHRiMP-2.2.3, dindel-1.01, freebayes-0.9.9, and SNVer-0.4.1. The real data workflow was similar to the simulated data workflow. The differences are that Shrimp was run with --no-qv-check and --qv-offset 33 because the real data were Sanger traces rather than Illumina reads, and Dindel was run with --numWindowsPerFile 1000000. Dindel was not used much on real data because at one day of running, it was still not finished. The workflow and arguments used were the following.

1. Read Mapping:

BFAST:
```
bfast fasta2brg -f reference.fasta
bfast index -f reference.fasta -m
1111111111111111111111 -w 14 -n 4
bfast match -f reference.fasta -r reads.fastq -n 4 -t 1>
bfast.matches.bmf 2> bfast.matches.out
bfast localalign -n 4 -t -f reference.fasta -m
bfast.matches.bmf 1> bfast.aligned.baf 2> bfast.aligned.out
bfast postprocess -f reference.fasta -i bfast.aligned.baf -o 3
-a 3 > align.sam
```

Bowtie 2:
```
bowtie2-build reference.fasta reference
bowtie2 -x ref -U reads.fastq -S align.sam
```

BWA:
```
bwa index reference.fasta
bwa aln reference.fasta reads.fastq > align.sai
bwa samse reference.fasta align.sai reads.fastq > align.sam
```

SHRiMP:
```
gmapper -N 4 reads.fastq reference.fasta > align.sam
```

2. BAM Conversion:

Samtools:
```
samtools faidx reference.fasta
samtools view -b -S align.sam > align.bam
samtools sort align.bam align.sorted
samtools index align.sorted.bam
```

3. Indel Calling

Dindel:
```
dindel --ref reference.fasta --outputFile dindel_output --
bamFile align.sorted.bam --analysis getCIGARindels
makeWindows.py --inputVarFile dindel_output.variants.txt
--windowFilePrefix realign_windows --
numWindowsPerFile 1000
dindel –analysis indels --bamFile align.sorted.bam –
doDiploid --ref reference.fasta --varFile
realign_windows.1.txt --libFile dindel_output.libraries.txt --
outputFile stage3_output
echo "stage3_output.glf.txt" > list.txt
mergeOutputDiploid.py -i list.txt -o indels.vcf -r
reference.fasta
```

Freebayes:
```
freebayes --no-snps --no-mnps --no-complex -b
align.sorted.bam -f reference.fasta -v indels.vcf
```

SNVer:
```
java -jar SNVerIndividual.jar -i align.sorted.bam -o
indels.vcf -r reference.fasta
```

2.2 Simulated Data

The simulated data was generated from 10 megabases of chromosome one from a publicly available human genome available from the National Center for Biotechnology Information. Artificial mutations were introduced using inGAP, a software tool for the manipulation of genetic data [11]. SNPs were inserted at a divergence rate of 0.1%, and indels were inserted at a divergence rate of 0.02%. These values were chosen since they were realistic [1]. Indel lengths were uniformly distributed from one to thirty bases. Ten replicates of simulated reads were produced to generate average and error statistics for the tests. Reads of uniform 50 base pair length were generated from the mutation sequences in the fastq file format using inGAP. Reads of length 50 were chosen because indel identification is more complex for shorter single-end reads since they “lack insert length variance” [2], so short single end reads represented a good test of indel pipeline sensitivity. In order to study the effects of varying coverage on the accuracy of the pipelines, reads were generated for 10x, 50x, and 100x coverage for each of the mutation sequences.

2.3 Real Data

Applied Biosystems (Sanger) paired-end traces from the set Chr_22_7340 were identified and downloaded from the NCBI trace archive. These traces were used in a Devine lab study that searched for indels in human chromosome 22 [12]. The paper identified 6487 indels for the Chr_22_7340 traces.

We cleansed the traces of contamination using NCBI VecScreen where traces with vector contamination in the middle were discarded, and traces with vector contamination on the ends had the contamination trimmed off. After this, there were 217,924 traces with sizes as much as 2000bp. Since short read mappers perform poorly with very long sequences, 10 million 100bp portions of the traces were sampled with replacement in order to simulate short reads. The 10 million simulated single-end sequences were run through the pipelines with timing tracked with the Linux “date” command.

2.4 Indel Detection

A confusion matrix for each pipeline on each data set was created that recorded true positives, false positives and false negatives. Indels were recorded as true positives if the predicted indel's position was plus or minus 5 nucleotides of the actual indel's position and the predicted length was within 5 percent of the actual length (with all lengths set to be one if 5 percent of the actual length was less than 1). The indel had to be correctly classified as an insertion or a deletion to be marked a true positive; otherwise, it was classified as a false positive. The sequence identity of predicted and actual indels was not checked since differences in sequence identity were rare. A false positive was a predicted indel that didn't meet the preceding criteria, and a false negative was an actual indel that wasn't identified by the indel classifier. Precision, recall, and F1-score were calculated for all pipelines to assess accuracy. Python 2.6 and Bash scripts were created to do the statistical analysis and workflow.

3 Results and Discussion

3.1 Analysis of F1-Score and Coverage on Simulated Data

In our results on simulated data with indels of size 1-30 bases there were clearly pipelines that performed better than other pipelines as measured by the F1-score (Figure 1). For each pipeline, Figure 1 shows average F1-score and the minimum and maximum F1-score of the 10 replicates. Most indels called had fewer than 10 bases.

Figure 1 shows that pipelines with 10X coverage have the best F1-scores, and that 50X and 100X coverage perform less well. This was explained by a tradeoff between precision and recall caused by both increasing false positives and increasing true positives. As coverage increased, recall increased since indel callers return more predicted indels and thus more genuine indels. However, there were more false positives as coverage increased, so precision went down as coverage increased. The general downward trend of the F1-score was because precision decreased more than recall increased with increasing coverage. This suggests that there is some coverage amount that maximizes F1-score for the data, and increasing coverage isn't always desirable. This result was consistent with other work that showed statistically significant precision and recall trends with increasing coverage [2].

The three top performing pipelines were BFAST-Dindel (avg F1-score 0.66), SHRIMP-SNVer (0.53), and BFAST-SNVer (0.51) in the 10X coverage for 1-30 indels. The BFAST-Dindel pipeline had the best average F1-score for all coverage amounts (Figure 1). SHRIMP pipelines were interesting since the F1-score varied considerably. The Shrimp-SNVer pipeline was among the top performing, but Shrimp-Freebayes and Shrimp-Dindel performed poorly. By default, Shrimp mapped some reads to multiple positions.

Read mappers use a seed and extend strategy, and BFAST's seeding strategy used a sliding window at every base. Bowtie2 used multiple 20bp seeds with an offset determined by the read length. Perhaps BFAST's seed strategy allowed it to be more

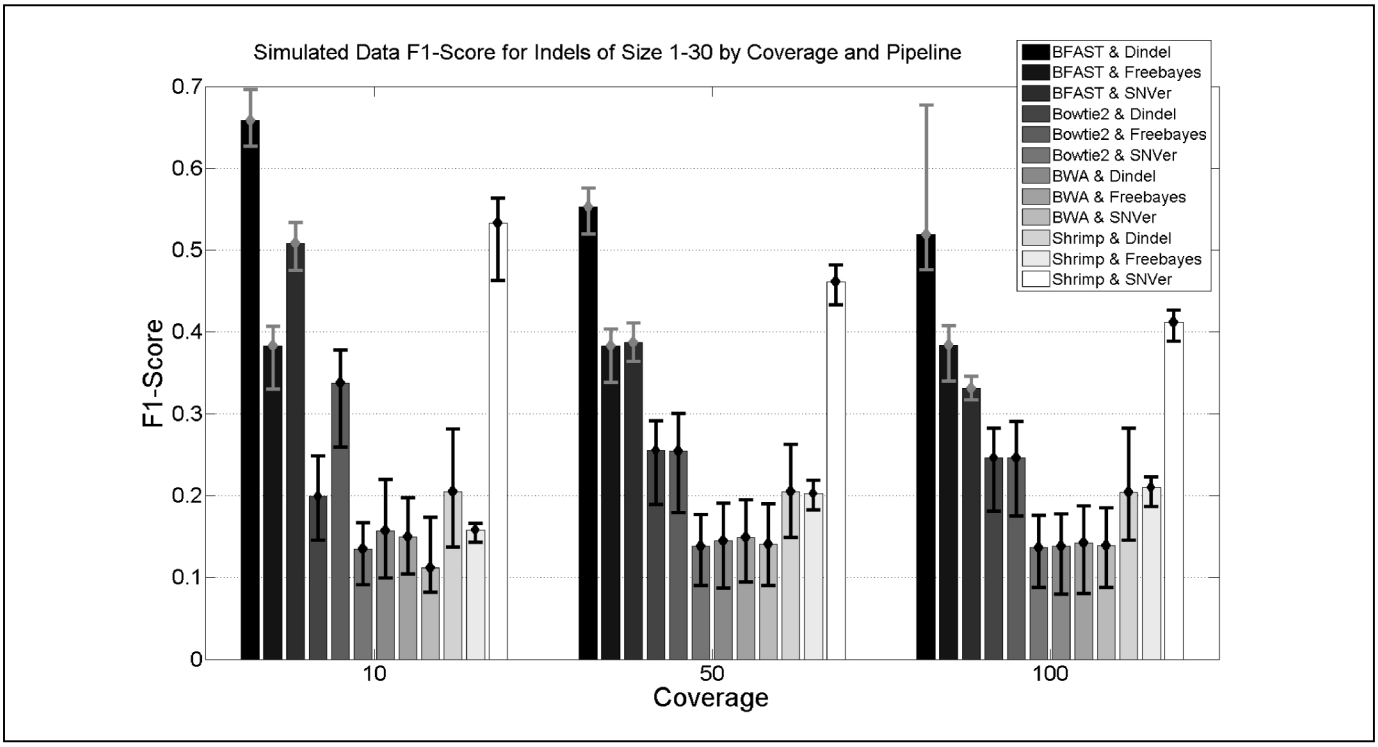

Figure 1 : The F1-score of indel calling pipelines on simulated data with reads containing indels of 1-30 bases. The results are divided into sets of 10X, 50X, and 100X coverage. The F1-score is shown with average, low and high scores.

accurate. All the mappers' extension phases are similar since they use local or global alignment algorithms [4, 5, 6, 7].

The F1-score difference for 10X coverage between the best pipeline (BFAST-Dindel) and the worst pipeline (BWA-SNVer) was about 0.546. BWA generally performed poorly and this could be because it only supported gaps less than 10 bases in its alignments [5]. Even though BWA and Bowtie2 used a similar seeding strategy with the Burrows-Wheeler transform, Bowtie2 pipelines usually had better F1-scores. Bowtie2's split seed approach handles some variation in the seed [4].

Interestingly, there isn't one clear indel caller that did the best overall. Bowtie2-SNVer was among the lowest performing while SHRIMP-SNVer was among the best performing. Dindel did well with BFAST but not very well with SHRIMP.

3.2 Precision and Recall on Simulated Data With Smaller and Longer Indels

Figures 2 and 3 show a comparison of the effects of longer indels on precision and recall at 10X coverage. Pipelines performed worse for data with indels of 1-30 bases (Figure 3) than for indels with 1-10 bases (Figure 2). Figure 3's precision-recall tuples are generally shifted left when compared to Figure 2. The Bowtie2-Freebayes pipeline did noticeably better with 1-10 indel lengths.

The precision-recall plots show which pipelines are conservative, which generous, and which are balanced. The pipelines involving BWA and Bowtie2 were the most conservative with high precision but low recall. Pipelines involving Freebayes were the most generous with low precision but higher recall. BFAST-Dindel and Shrimp-SNVer had the most balanced precision and recall results with (0.648,0.771) and (0.640,0.799) respectively for indels of length 1-10. The BFAST-Dindel pipeline performed better than Shrimp-SNVer for indels of length 1-30.

Bowtie2 pipelines generally appeared mediocre in our tests. BFAST pipelines had generally good performance with different indel callers while SHRIMP pipeline performance varied considerably with indel calling software. BWA pipelines performed poorly.

3.3 Accuracy of Real Data

The only accurately called indels on the real data were smaller than 5bp. Figure 4 shows the F1-scores of the real data pipelines. SNVer pipelines had the best F1-score. Similar to the simulated data, pipelines with Freebayes were too generous with high recall but low precision. Average precision and recall for Freebayes was 0.000440955 and 0.010906428, but with SNVer it was 0.00117591 and 0.001079081. Thus, SNVer was more conservative in indel calling. The Shrimp-SNVer and Bowtie2-SNVer pipelines did the best while BWA-Freebayes was the worst. The choice of read mapper made little difference, and this could be because only small indels were called. True positives were few relative to indels called (Table 1). For the BWA mappings, Dindel completed in 6.6 hours with similar precision (0.00062) and recall (0.0026) to the BFAST-SNVer pipeline (0.00079, 0.0012).

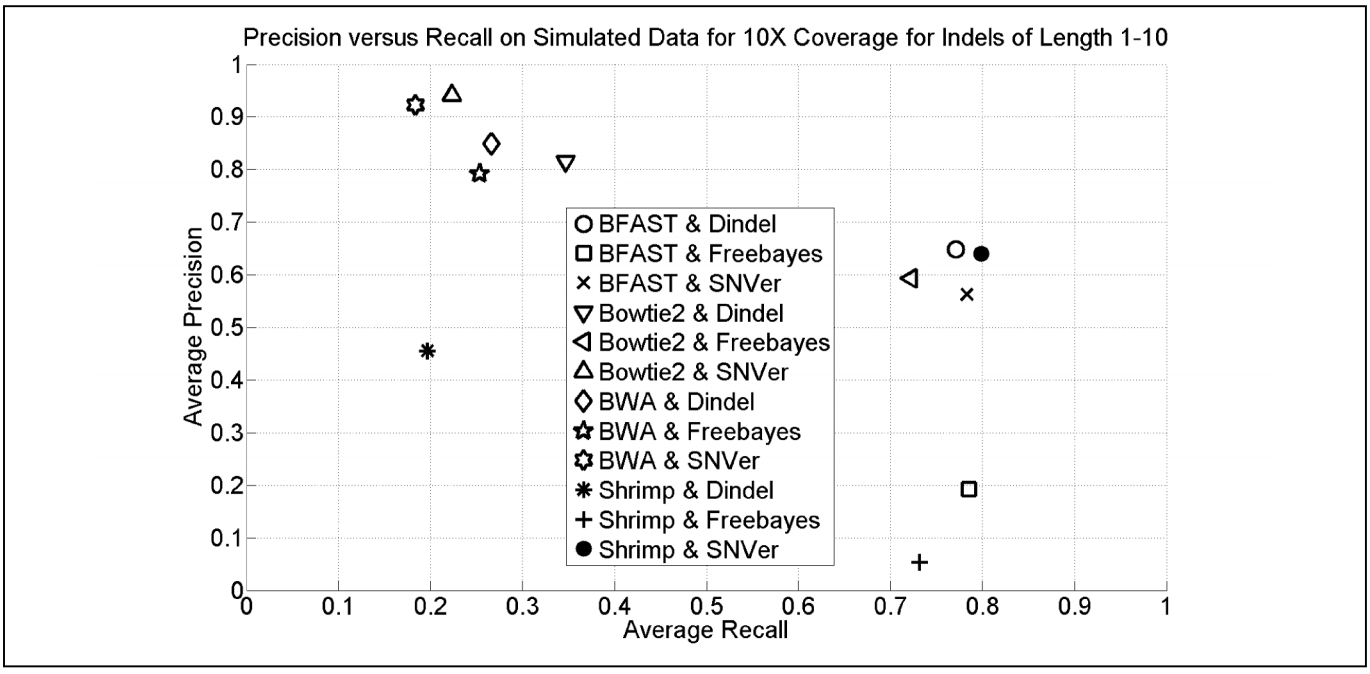

Figure 2 : Average precision and average recall for 10 simulated data replicates for the indel calling pipelines. Precision and recall was calculated for indels with only 1-10 bases at 10X coverage. The reads contained indels as large as 30 bases.

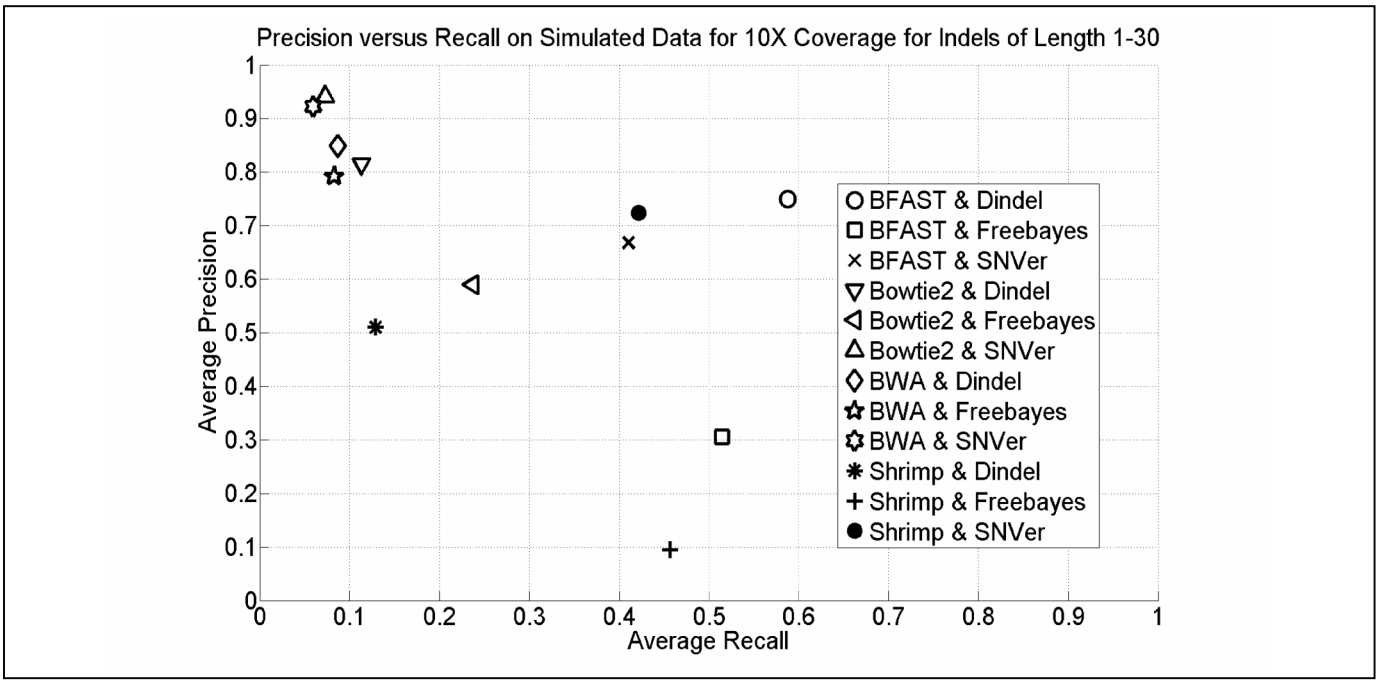

Figure 3 : Average precision and average recall for 10 simulated data replicates for the indel calling pipelines. Precision and recall was calculated for all indels. Indels had 1-30 bases at 10X coverage.

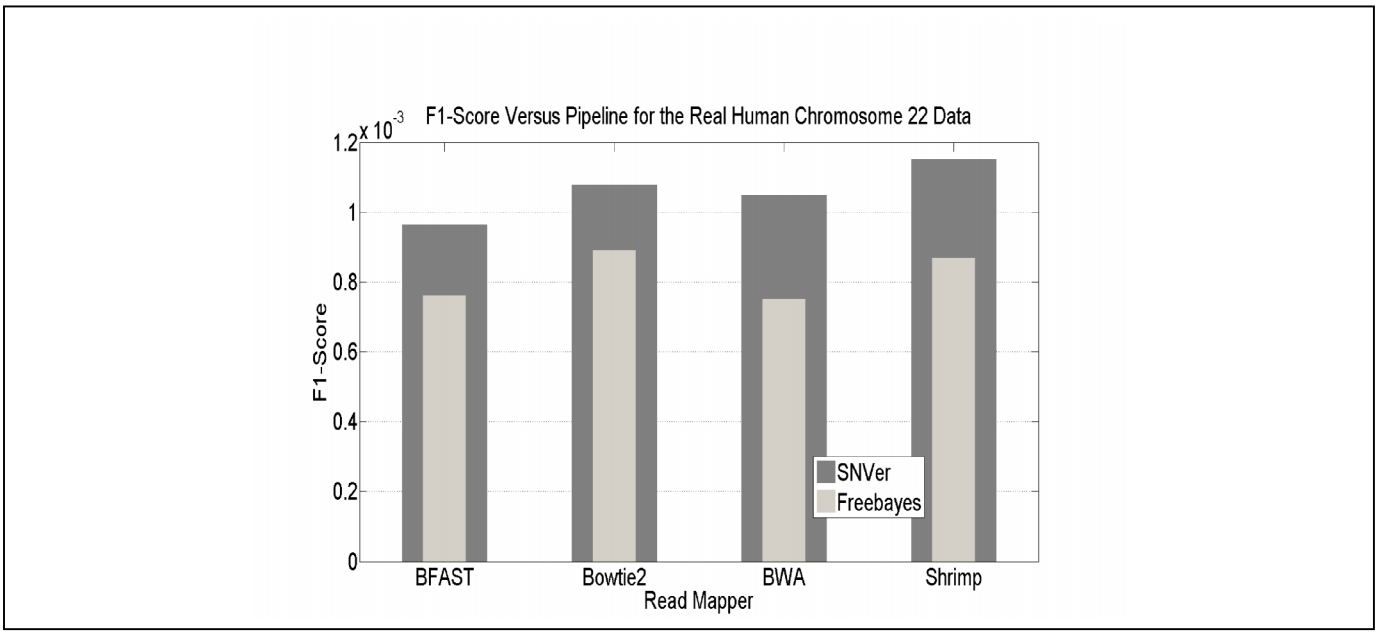

Figure 4 : F1-Score for the indel calling pipelines on 10 million real human chromosome 22 traces.

3.4 Run-Time Performance on Real Data

Table 1 summarizes run-time performance for the pipelines for the real human chr22 data. Bowtie2 and BWA, had the fastest runtimes at 27 and 23 minutes respectively. BFAST and Shrimp were the slowest mappers, and both used a sliding window hashing seed strategy. BFAST was about 10 minutes slower, but Shrimp was 6.2 times slower than BWA, making Shrimp pipelines the slowest. The Shrimp read mapping percent is more than 100 percent since it mapped some reads to multiple positions by default.

The indel callers did not have multithreading, so they were slow. Freebayes was always faster than SNVer, and SNVer took 145 minutes with Shrimp's input making the Shrimp-SNVer pipeline the slowest. Dindel took over a day (except

Table 1 : Mapper, caller, pipeline run-time, percent mapped, and indels called on 10 million real human 100bp reads

Mapper	Caller	Minutes	Total Minutes	% Reads Mapped	Total Indels Called	True Indels
BFAST		38		0.5437498		
	Samtools	5				
	SNVer	18	61		20486	8
	Freebayes	37	80		358574	139
Bowtie2		27		0.4850195		
	Samtools	4				
	SNVer	19	50		9812	6
	Freebayes	16	47		114679	54
BWA		23		0.412648		
	Samtools	4				
	SNVer	18	45		6388	5
	Freebayes	9	36		30789	14
Shrimp		143		1.7618786		
	Samtools	4				
	SNVer	145	292		9145	9
	Freebayes	59	206		168684	76

with BWA input), making it less tolerable for big indel calling projects; however, Dindel has the ability to split its work into multiple files for manual multiprocessing.

4 Conclusions

To our knowledge, this work is the first to look at the accuracy of the combination of mapping software and indel calling software with larger (>10 nucleotides) indels. F1-score, a measure of accuracy, fell with increased coverage, belying expectations. Indel calling accuracy depended on the combination of mapping software and indel calling software. Some of the top performing pipelines were BFAST-Dindel, SHRIMP-SNVer, and BFAST-SNVer on simulated data. The best pipeline had an F1-score 0.6 higher than the worst pipeline on simulated reads. On real data, SNVer pipelines were more accurate than FreeBayes pipelines in all cases. SNVer and Shrimp can have slow runtimes, but Dindel was by far the slowest. Future work could include exploring the parameter space of the tools to observe the effects of argument selection on sensitivity.

5 References

[1] Mullaney JM, Mills RE, Pittard WS, *et al.* Small insertions and deletions (INDELs) in human genomes. *Hum Mol Genet* 2010;19:R 131-b

[2] Neumann JA, Isakov O, Shomron N. Analysis of insertion-deletion from deep-sequencing data: software evaluation for optimal detection. *Brief Bioinform* 2013 Jan;14(1):46-55

[3] Pabinger S. *et al.* survey of tools for variant analysis of next-generation genome sequencing data. *Brief Bioinform* 2013 Jan 21.

[4] Langmead B, Salzberg S. Fast gapped-read alignment with Bowtie 2. *Nature Methods*. 2012, 9:357-359.

[5] Li H. and Durbin R. Fast and accurate short read alignment with Burrows-Wheeler Transform. *Bioinformatics* 2009, 25:1754-60.

[6] David M, Dzamba M, Lister D, Ilie L, Brudno M. SHRiMP2: sensitive yet practical Short Read Mapping. *Bioinformatics* 2011, 27:1011-102.

[7] Homer N, Merriman B, Nelson SF. BFAST: An alignment tool for large scale genome resequencing. *PLoS ONE* (2009). 4(11): e7767.

[8] CA Albers, G Lunter, Daniel G MacArthur, Gilean McVean, Willem H Ouwehand, Richard Durbin. Dindel: Accurate indel calls from short-read data. *Genome Research*: 2010

[9] Erik Garrison, Gabor Marth. Haplotype-based variant detection from short-read sequencing. ArXiv:1207.3907[q-bio.GN]

[10] Wei Z, Wang W, Hu P, Lyon GJ, Hakonarson H. SNVer: a statistical tool for variant calling in analysis of pooled or individual next-generation sequencing data. *Nucleic Acids Res.* 2011 Oct; 39(19):e132

[11] Qi J, Zhao F, Buboltz A, *et al.* InGAP: an integrated next-generation genome analysis pipeline. *Bioinformatics* 2010;26:127-9.

[12] Ryan E. Mill, Stephen Pittard, et al. Natural genetic variation caused by small insertions and deletions in the human genome. *Genome Res*. 2011. 21: 830-839