Esther Olabisi-Adeniyi Student ID: 1238826

Project 3: Variant Calling (Identifying SNPs)

Introduction

Variant calling is a critical component of numerous scientific endeavors. In medical research for instance, researchers need reliable data about genomic variants to investigate the relationship between specific variants and diseases (Hwang et al., 2015). Population geneticists also rely on variant data to analyze patterns among samples and draw inferences about populations (Lou et al., 2021, p. 5967). There are several variant callers including Bcftools mpileup, Freebayes, and Genome Analysis Tool Kit HaplotypeCaller (GATK-HC), all of which have their strengths and weaknesses (Hwang et al, 2015). One example is that Freebayes tends to perform better than Bcftools at SNP calls when variants are filtered for quality (Hwang et al., 2015). Variant callers also typically lean toward different types of errors: Freebayes tends to call heterozygous SNPs as homozygous while Bcftools tends to call homozygous SNPs as heterozygous (Hwang et al., 2015). Therefore, one must take these potential errors into consideration when choosing a variant caller.

For this project, Freebayes and Bcftools were used to detect SNPs from alignment data of 10 burbot individuals measured against a high-quality burbot reference genome. The main objective is to compare Freebayes and Bcftools SNP identification results based on four factors: the number of SNPs across both callers, minor allele frequencies of all SNPs, depth of SNPs per locus across all individuals, and mean depth of coverage per individual.

Results

Freebayes had a total of 46,722 SNPs while Bcftools had 12,995 SNPs (Figure 1). 8,997 SNPs were common to both callers, representing about 69% of all Bcftools SNPs and only about 19% of all Freebayes SNPs. The minor allele frequency plot for Bcftools shows that more loci (~1,500) had alleles with low frequencies (~0 to 0.25) as expected. Furthermore, number of loci decreased as minor allele frequencies increased, generally (Figure 2). Freebayes on the other hand shows that alleles with intermediate (~0.5) and higher (~0.9) frequencies were more common across loci (Figure 2). For depth of SNPs per locus across all individuals, Freebayes appears to be more sensitive to depth levels since it detects more depth of coverage overall (Figure 3). Mean depth of coverage per individual was also considered. Compared to Bcftools, Freebayes reports greater mean depths of coverage for all individuals (Figure 4).

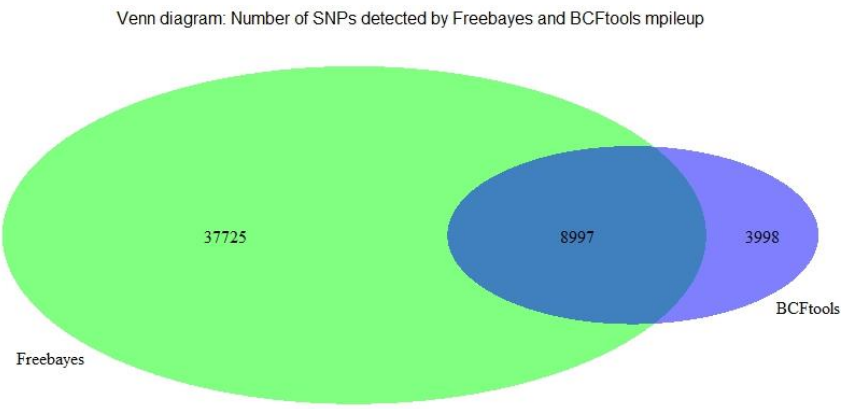

Figure 1: Unique and shared number of SNPs between Freebayes and BCFtools.

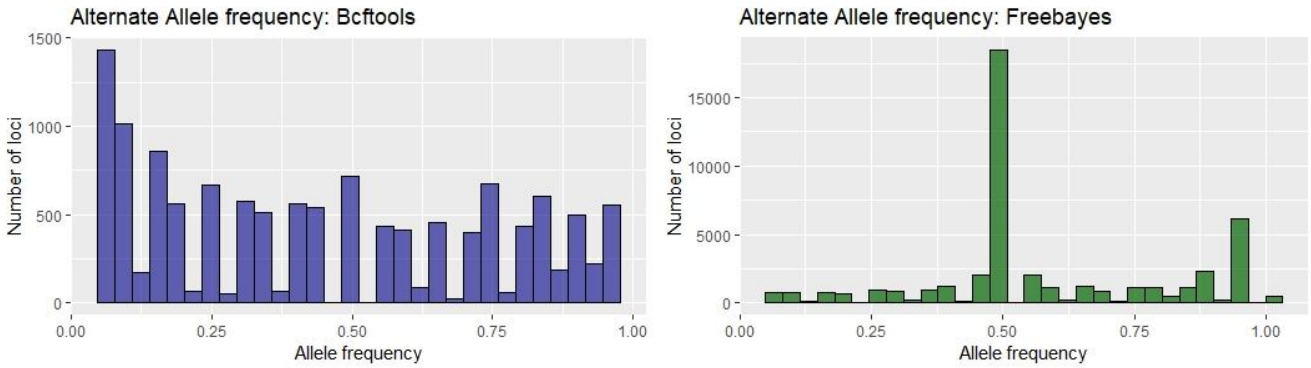

Figure 2: Allele frequency counts for Freebayes and Bcftools

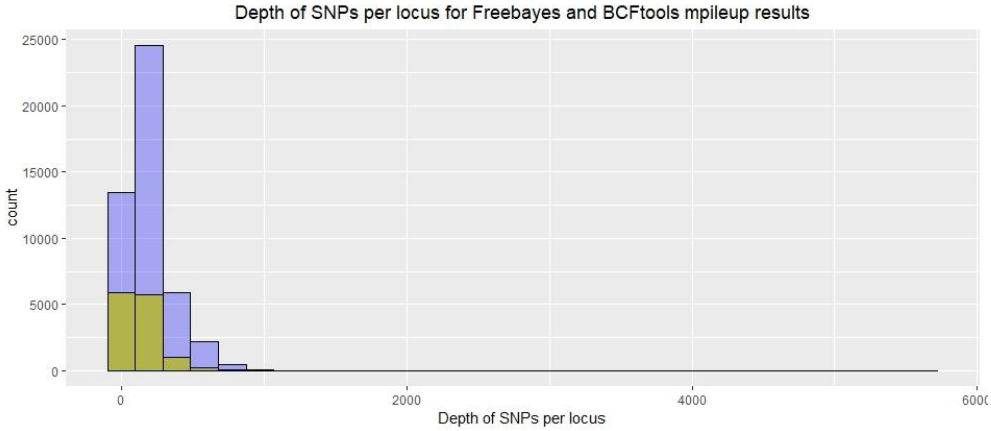

Figure 3: Overlapping histograms: depth of all SNPs summed across individuals.
Yellow = Bcftools. Blue = Freebayes.

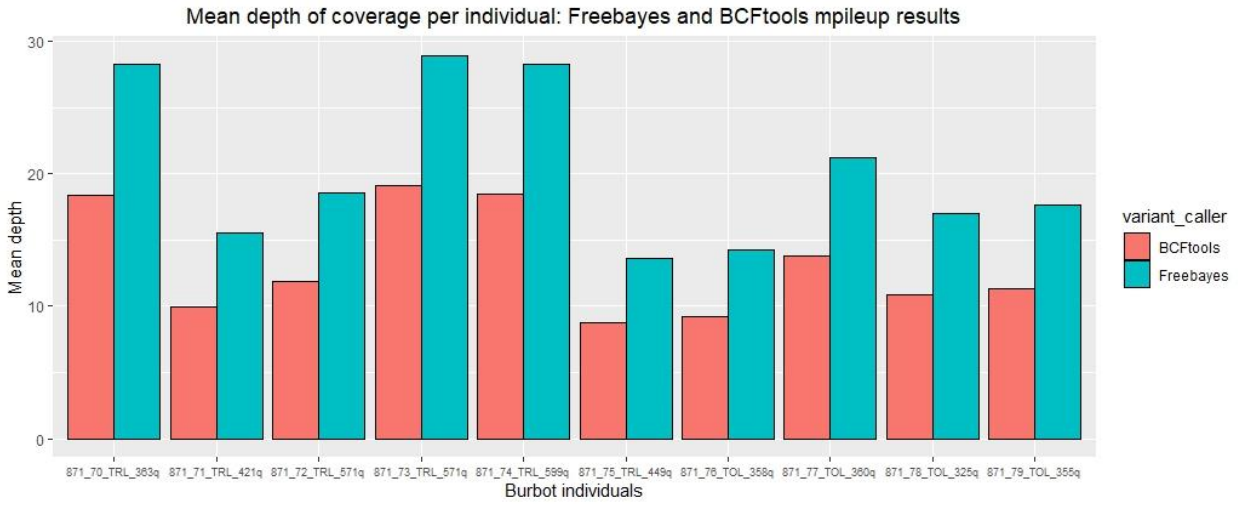

Figure 4: Mean depth of reads per locus per individual.

Discussion

The number of variants identified by Freebayes and Bcftools were compared. Freebayes called about three times the number of SNPs called by Bcftools (Figure 1). In addition to its unique SNP calls, Freebayes also identified more than half (~69%) of the same SNPs identified by Bcftools, suggesting that the former may be the better variant caller since it detected most of what the latter did. When Hwang et al. (2015) compared both variant callers; they also found that Freebayes performed better when variants were filtered by quality score. That might explain why Freebayes performed better for my analysis since I also filtered both results to retain only high-confident SNPs (QUAL >19). Depth of SNPs per locus further suggests that Freebayes is a better SNP caller because Freebayes detected greater depths of coverage across loci than Bcftools (Figure 3). When comparing mean depth of coverage per individual between both variant callers, we see a recurring pattern in which Freebayes exhibits more sensitivity to depth of coverage (Figure 4). For every burbot individual that was sampled for SNPs, Freebayes reported more mean depth.

Meanwhile, one of Bcftools' strengths is expressed in its minor allele frequency spectrum, which follows the expected pattern i.e. minor alleles with low frequencies are the most occurring while those with high frequencies have fewer occurrences (Figure 2). The unusual bimodal pattern from Freebayes probably reflects its tendency to read heterozygous SNPs as homozygous (Hwang et al, 2015) which causes otherwise heterozygote SNPs to have more minor alleles. These conclusions demonstrate that in as much as Freebayes seems to have the best performance as a variant caller, it also has its weakness in the same manner that Bcftools has its advantages. If possible, to maximize performance, a minimum of two variant callers should be considered when carrying out variant identification

Methods

Bcftools mpileup and Freebayes variant callers were used to call variants among 10 burbot fish against a burbot reference genome. I carried out my data analysis on Graham HPC cluster and produced figures with RStudio 2022.12.0. This is the Graham directory path to all scripts, files, and subdirectories used: /scratch/eolabisi/MyProject3/

To access the data need for this project, I copied Project3 directory to my directory

```
cp /scratch/emandevi/genomic_methods_w23/Project3/ .
```

Next, I created a shell script 'run_sam_to_bam.sh' for the sbatch conversion of sam files to bam files. The script also sorts and indexes the bam files after conversion.

```
#assign variables
samfile=$1
basename=`echo $samfile | sed 's/\.gz.sam//'`

#convert sam to bam
echo "Converting sam to bam for $basename"
samtools view -b -S -o bam_assem/$basename.bam $samfile

#sort and index bam files
echo "Sorting and indexing bam files for $basename"
samtools sort bam_assem/$basename.bam –o
bam_assem/$basename.sorted.bam
samtools index bam_assem/$basename.sorted.bam
```

Next, I created another assembly in bwa_assem to hold all sorted and indexed bam files. Then I used a 'for' loop to do all sam to bam conversions:

```
mkdir bam_assem

for file in *.sam; do sbatch run_sam_to_bam.sh $file; done
```

Now that all sorted and indexed bam files have been created, the next step was to create a script 'run_bcf_caller.sh' to call variants among burbot individuals against the provided reference genome. Filters were applied to select only SNPs, excluding insertions and deletions. I filtered out low confident SNPs by setting the quality score (QUAL) to a minimum of 19. The output was a vcf file containing variant information for all 10 individuals.

```
#Run bcftools so that results include depths (DP & AD), and
genotype quality GQ, while excluding indels

echo "- Calling variants for all bam files..."

bcftools mpileup -a DP,AD -f burbot_2021.fasta *.sorted.bam |
bcftools call -m --variants-only --format-fields GQ --skip-
variants indels > $output_file.bcf


#Filter bcf from the previous step to exclude monomorphic SNPs
and keep only variants with a quality score >19.
```

```
echo "- Further filtering results for a final vcf file..."

bcftools filter -e 'AC==0 || AC==AN' $output_file.bcf | bcftools
filter --include 'QUAL > 19' > $output_file.vcf
```

To call variants with the second caller, freebayes, read groups must be added to sorted bam files first. Therefore, I created a script 'run_freebayes_caller.sh' that uses picard to add read groups to the files, samtools to re-index the new files, and finally freebayes to calls variants. The same filters applied in bcftools mpileup were applied here for consistency and a second vcf file was produced.

```
echo "- Adding Read Groups to all bam files..."
java -jar $EBROOTPICARD/picard.jar AddOrReplaceReadGroups
I=$bamfile O=$rg_bamfile RGID=$bamfile RGLB=lib1 RGPL=illumina
RGPU=unit1 RGSM=$bamfile

echo "- Reindexing new bam files containing read groups..."
samtools index $rg_bamfile

#Apply the same filters as in bcftools mpileup
echo "- Calling variants for all bam files..."
freebayes -f burbot_2021.fasta -b *_rg.sorted.bam | vcffilter -f
'TYPE = snp' -f 'QUAL > 19' | vcffilter -f "! ( AC = 0 )" |
vcffilter -f "! ( AC = AN )" > $output_file.vcf
```

To compare the number of SNPs and overlap across both vcf files, I used the 'isec' option from bcftools, which reports shared, and unique variants of both files. However, since the command only works on bgzipped files, I had to bgzip and reindex the vcf files before proceeding.

```
for file in *.vcf; do bgzip $file; done

for file in *.gz; do bcftools index $file; done

bcftools isec -p isec bcf_results.vcf.gz freebayes_result.vcf.gz
```

Bcftools isec produced four new vcf files and a README.txt. Below is the content of the README file, describing each of the vcfs.

```
Using the following file names:
isec/0000.vcf   for records private to  bcf_results.vcf.gz
isec/0001.vcf   for records private to  freebayes_result.vcf.gz
isec/0002.vcf   for records from bcf_results.vcf.gz shared by both      bcf_results.vcf.gz freebayes_result.vcf.gz
isec/0003.vcf   for records from freebayes_result.vcf.gz shared by both bcf_results.vcf.gz freebayes_result.vcf.gz
(END)
```

I extracted unique and shared number of SNPs with bcftools stats and stored them in a text file. I also manually edited the text file in nano text editor to obtain a better format for downstream analysis in RStudio.

```
for file in *.vcf; do echo $file; bcftools stats $file | grep
"number of SNPs:"; echo ""; done > compared_SNPs.txt
```

I used vcftools to calculate allele frequency across the two vcf files and stored the outputs as text files. Since we need to plot biallelic SNPs only, I also filtered out allele frequency results with more than two number of alleles.

```
#Allele freq
vcftools --gzvcf bcf_results.vcf.gz --freq --out bcf_freq
cat bcf_freq.frq > bcf_freq.txt

vcftools --gzvcf freebayes_result.vcf.gz --freq --out
freebayes_freq
cat freebayes_freq.frq > freebayes_freq.txt

#extract only biallelic SNPs by selecting only rows with '2' in
the third column.
cat bcf_freq.txt | awk 'BEGIN{FS="\t"}$3~/[2]/' > bcf_freq_bi.txt
cat freebayes_freq.txt | awk 'BEGIN{FS="\t"}$3~/[2]/' >
freebayes_freq_bi.txt
```

Next, I calculated the depth of all SNPs summed across individuals as well as the mean depth per individual. All text files will now be visualized in RStudio.

```
#depth of all SNPs across all individuals
vcftools --gzvcf freebayes_result.vcf.gz --site-depth -c >
all_freebayes_SNP_dp.txt

vcftools --gzvcf bcf_results.vcf.gz --site-depth -c >
all_BCF_SNP_dp.txt

#mean depth per individual for both VCFs
vcftools --gzvcf freebayes_result.vcf.gz --depth -c >
mean_dp_ind_freebayes.txt
vcftools --gzvcf bcf_results.vcf.gz --depth -c >
mean_dp_ind_BCF.txt
```

RStudio CODE

```
############Compare the number of SNPs and the overlap -----
#############
library("VennDiagram")
library("gridExtra")
library("ggplot2")
library("tidyverse")

#Import table containing number of SNPs. Add row names according to
the descriptions in README.txt.
no_of_SNPs <- read.table("compared_SNPs.txt", header = F)
row.names(no_of_SNPs) <- c("BCFtools", "Freebayes", "Shared1",
"Shared2")

#Manually typing in the values is more efficient in this case than
selecting table cells.
```

```
BCFtools <-  3998 + 8997
Freebayes <- 37725 + 8997
Shared <- 8997

#Draw venn diagram with number of shared and unique number of SNPs.
The option to represent the plot in percentages is not working as
expected (see commented line of code) so I went with raw values,
unfortunately.
grid.newpage()
venn <- draw.pairwise.venn(area1=BCFtools, area2=Freebayes,
cross.area=Shared,
category=c("BCFtools","Freebayes"),fill=c("blue","green"),
col='transparent')
#draw.pairwise.venn(area1=BCFtools, area2=Freebayes,
cross.area=Shared,
category=c("BCFtools","Freebayes"),fill=c("blue","green"),
col='transparent', print.mode = 'percent')

#Add title
grid.arrange(gTree(children = venn), top="Venn diagram: Number of SNPs
detected by Freebayes and BCFtools mpileup")


############Plot the allele frequency of alternate alleles ----
############

#import allele frequency tables and add column names
bcf_allele_freq<- read.table("bcf_freq_bi.txt", header = F)
colnames(bcf_allele_freq) <- c("CHROM", "POS", "N_ALLELES", "N_CHR",
"REF_ALLELE_FREQ", "ALT_ALLELE_FREQ")

freebayes_allele_freq <- read.table("freebayes_freq_bi.txt", header =
F)
colnames(freebayes_allele_freq) <- c("CHROM", "POS", "N_ALLELES",
"N_CHR", "REF_ALLELE_FREQ", "ALT_ALLELE_FREQ")

#Edit the alternate allele frequency column to remove allele alphabets
and colons, leaving numbers only. Convert values from 'character'
class to 'numeric.'
bcf_allele_freq$ALT_ALLELE_FREQ <- as.numeric(gsub("[AGCT]+:", "",
bcf_allele_freq$ALT_ALLELE_FREQ))
freebayes_allele_freq$ALT_ALLELE_FREQ <- as.numeric(gsub("[AGCT]+:",
"", freebayes_allele_freq$ALT_ALLELE_FREQ))

#histogram for bcftools allele frequency
ggplot(bcf_allele_freq, aes(x=ALT_ALLELE_FREQ))+
  geom_histogram(aes(y=..count..), color="black", fill="darkblue",
alpha=0.6)+
  ggtitle("Alternate Allele frequency: Bcftools")+
  xlab("Allele frequency") + ylab("Number of loci")
```

```
#histogram for freebayes allele frequency
ggplot(freebayes_allele_freq, aes(x=ALT_ALLELE_FREQ))+
  geom_histogram(aes(y=..count..), color="black", fill="darkgreen",
alpha=0.7)+
  ggtitle("Alternate Allele frequency: Freebayes")+
  xlab("Allele frequency")+ ylab("Number of loci")



#############Plot the depth of all SNPs (summed across individuals)---
-#####
#import tables containing depths. Add a new column to each table
indicating the source of the results. This distinction will be
important for creating a histogram in the next step. Finally, merge
both tables into one dataframe.
bcf_snp_dp <- read.table("all_BCF_SNP_dp.txt", header = T)
bcf_snp_dp$variant_caller <- "BCFtools"

freebayes_snp_dp <- read.table("all_freebayes_SNP_dp.txt", header = T)
freebayes_snp_dp$variant_caller <- "Freebayes"

merged_snp_dp <- merge.data.frame(bcf_snp_dp, freebayes_snp_dp, all=T)

#Plot a histogram of SNP depth results for freebayes and bcftools
mpileup. Yellow represents BCFtools while blue represents freebayes.
ggplot(merged_snp_dp, aes(x=SUM_DEPTH))+
  geom_histogram(data=subset(merged_snp_dp,
variant_caller=="BCFtools"),
                 fill="yellow", position="identity",
                 color="black")+
  geom_histogram(data=subset(merged_snp_dp,
variant_caller=="Freebayes"),
                 fill="blue",
                 position="identity",  alpha=0.3, color="black")+
  xlab("Depth of SNPs per locus")+
  ggtitle("Depth of SNPs per locus for Freebayes and BCFtools mpileup
results")+
  theme(plot.title = element_text(hjust = 0.5))


###########Mean depth for each VCF (reads per locus per individual)---
-
#Import files, add variant caller names, and merge tables as in the
previous steps. I also used gsub to shorten the burbot IDs so that the
IDs fit in the axis as labels.
mean_dp_bcf <- read.table("mean_dp_ind_BCF.txt", header = T)
mean_dp_bcf$variant_caller <- "BCFtools"

mean_dp_freebayes <- read.table("mean_dp_ind_freebayes.txt", header =
T)
mean_dp_freebayes$variant_caller <- "Freebayes"
```

```
combined_mean_dp <- merge.data.frame(mean_dp_bcf, mean_dp_freebayes,
all=T)
combined_mean_dp$INDV <- gsub(".sorted.bam", "",
combined_mean_dp$INDV)


#Plot bar graphs for mean depth of coverage.
ggplot(combined_mean_dp, aes(x=INDV, y=MEAN_DEPTH,
fill=variant_caller))+
  geom_bar(stat="identity", position="dodge", color="black")+
  xlab("Burbot individuals")+ ylab("Mean depth")+
  ggtitle("Mean depth of coverage per individual: Freebayes and
BCFtools mpileup results")+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(axis.text.x = element_text(size = 7))
```

References

Hwang, S., Kim, E., Lee, I., & Marcotte, E. M. (2015). Systematic comparison of variant calling pipelines using gold standard personal exome variants. *Scientific Reports*, *5*(1). https://doi.org/10.1038/srep17875

Lou, R. N., Jacobs, A., Wilder, A. P., & Therkildsen, N. O. (2021). A beginner's guide to low-coverage whole genome sequencing for population genomics. *Molecular Ecology*, *30*(23), 5966–5993. https://doi.org/10.1111/mec.16077