Bversity
1st Ever Virtual College for Biotech

Name: Ashika Bangera
Contact: ashika.bangera2001@gmail.com
Mobile: 8431139778

Title: Variant Calling Analysis

1. What is the average read coverage across the genome for sample SRR2584866?

Determine the depth of sequencing coverage across the genome to understand the reliability of variant calling.

To determine the average read coverage across the genome for sample SRR2584866, I have used samtools depth to calculate the coverage at each position in the genome and then computed the average coverage. Here is a step-by-step guide on how I did this:

1. **Download the sample SRR2584866:** Use the fastq-dump tool from the SRA Toolkit to download the sample in FASTQ format.

```
fastq-dump SRR2584866
fastq-dump --split-files SRR2584866
```

2. **Align the reads to a reference genome:** Use a read aligner like BWA to map the reads to a reference genome.

```
bwa mem GCA_000017985.1_ASM1798v1_genomic.fna SRR2584866.fastq >
```

3. **Convert SAM to BAM format:** Convert the SAM file to BAM format for more efficient processing.

```
samtools view -S -b aligned_reads.sam > aligned_reads.bam
```

4. **Sort the BAM file:** Sort the BAM file to prepare it for depth calculation.

```
samtools sort aligned_reads.bam -o aligned_reads_sorted.bam
```

5. **Calculate the depth of coverage:** Use samtools depth to calculate the coverage at each position in the genome. This metric represents the number of reads that align to a particular position in the genome.

```
samtools depth aligned_reads_sorted.bam > depth.txt
```

6. **Calculate the average coverage:** Use a simple script or command to calculate the average coverage from the depth file. This is the mean depth of coverage across all positions in the genome, calculated as the sum of all per-position coverages divided by the number of positions.

   Here's a one-liner using awk to calculate the average coverage:

```
awk '{sum+=$3} END {print "Average coverage = ", sum/NR}' depth.txt
```

OUTPUT:

The average coverage, which is 171.427. This means that, on average, each position in the genome is covered by approximately 171 reads, indicating a high depth of sequencing coverage.

2. How many unique variants are identified across all samples relative to the E. coli REL606 reference genome?

Assess the diversity and variation within the population compared to the reference.

To assess the number of unique variants identified across all samples relative to the E. coli REL606 reference genome, I have compared the variants found in the VCF files generated for each sample. Here is a step-by-step guide to achieve this:

Step 1: Generate VCF Files for Each Sample, Using bcftools.

Step 2: Merge VCF Files, Used bcftools merge to combine all the VCF files into a single file for comparison

Step 3: Identify Unique Variants, used bcftools query to extract variant information and then process this information to find unique entries.

Here the codes how unique variants are identified,

```
# Step 1: Create the sample names file
echo -e "aligned_reads_sorted.bam\tsample1" > sample_names1.txt
echo -e "aligned_reads_sorted.bam\tsample2" > sample_names2.txt

# Step 2: Reheader the VCF files
bcftools reheader -s sample_names1.txt -o sample1.renamed.vcf.gz sample1.variants.vcf.gz
bcftools reheader -s sample_names2.txt -o sample2.renamed.vcf.gz variants.vcf.gz

# Step 3: Index the renamed VCF files
bcftools index sample1.renamed.vcf.gz
bcftools index sample2.renamed.vcf.gz

# Step 4: Merge the renamed VCF files
bcftools merge -o merged_variants.vcf -O v sample1.renamed.vcf.gz sample2.renamed.vcf.gz

# Step 5: Extract variant positions and alleles
bcftools query -f '%CHROM\t%POS\t%REF\t%ALT\n' merged_variants.vcf > all_variants.txt

# Step 6: Sort and count unique variants
sort all_variants.txt | uniq > unique_variants.txt

# Step 7: Count the number of unique variants
unique_count=$(wc -l < unique_variants.txt)
echo "Number of unique variants: $unique_count"
```

Output: number of unique variants identified across all samples relative to the E. coli REL606 is 1646

3. Which genomic regions show the highest density of single nucleotide variants (SNVs) across all samples?

Identify genomic hotspots of variation that might indicate regions under selective pressure or hypermutability.

To identify genomic regions with the highest density of single nucleotide variants (SNVs) across all samples, I Used bedtools to count the number of SNVs in specified genomic regions and identify regions with the highest SNV density.

Step 1: Prepare VCF File

Merged VCF file (merged_variants.vcf), is indexed:

```
bcftools index merged_variants.vcf
```

Step 2: Convert VCF to BED Format
Use bcftools and awk to convert the VCF file to a BED file format:

```
bcftools query -f '%CHROM\t%POS\t%POS\n' merged_variants.vcf | awk '{OFS="\t"; print $1,
$2-1, $3}' > variants.bed
```

Step 3: Count SNVs per Genomic Region

Define the genomic regions (e.g., 10kb windows) and use bedtools to count the SNVs in each region. First, create a BED file with genomic windows. For example, for a genome size of 5 million bases:

```
bedtools makewindows -g GCA_000017985.1_ASM1798v1_genomic.fna.fai -w 10000 >
genome_windows.bed
```

Then, count the number of SNVs in each window:

The -c option in bedtools intersect counts the number of features in the variants.bed file that overlap each window in genome_windows.bed.

```
bedtools intersect -a genome_windows.bed -b variants.bed -c > snv_counts.bed
```

Step 4: Identify Regions with Highest SNV Density

To identify dentify the regions with the highest SNV density by sorting the snv_counts.bed file:

```
sort -k4,4nr snv_counts.bed > sorted_snv_counts.bed
```

The sorted_snv_counts.bed file will have the windows sorted by the number of SNVs in descending order.

```
# Ensure the merged VCF file is indexed
bcftools index merged_variants.vcf

# Convert VCF to BED format
bcftools query -f '%CHROM\t%POS\t%POS\n' merged_variants.vcf | awk '{OFS="\t"; print $1,
$2-1, $3}' > variants.bed

# Create genomic windows (10kb windows in this example)
bedtools makewindows -g GCA_000017985.1_ASM1798v1_genomic.fna.fai -w 10000 >
genome_windows.bed

# Count SNVs per genomic window
bedtools intersect -a genome_windows.bed -b variants.bed -c > snv_counts.bed

# Sort windows by SNV count in descending order
sort -k4,4nr snv_counts.bed > sorted_snv_counts.bed

# Print the top regions with the highest SNV density
echo "Top regions with highest SNV density:"
head -n 10 sorted_snv_counts.bed
```

Output: Top regions with highest SNV density:

```
CP000819.1	1450000	1460000	74
CP000819.1	2880000	2890000	41
CP000819.1	4570000	4580000	39
CP000819.1	4550000	4560000	36
CP000819.1	600000	610000	36
CP000819.1	4580000	4590000	32
CP000819.1	1440000	1450000	28
CP000819.1	2030000	2040000	24
CP000819.1	2890000	2900000	23
CP000819.1	590000	600000	22
```

4. **Are there any shared variants (SNVs) among all samples, and if so, what is their functional impact based on annotation data?**

Investigate conserved variants that may have functional implications across the population.

To investigate shared variants (SNVs) among all samples and determine their functional impact based on annotation data, these are the steps followed:

Step-by-Step Breakdown:

1. **Index the VCF files** to ensure they can be processed by bcftools.
2. **Use bcftools isec** to find the variants shared among all samples.
3. **Install SnpEff or VEP** and download the appropriate database for your organism.
4. **Annotate the shared variants** using SnpEff or VEP.
5. **Review the annotation results** to understand the functional impact of the shared variants.

Step 1: Identify Shared Variants Using bcftools isec

First, use bcftools isec to find the common variants across all your VCF files.

```
# Ensure all VCF files are indexed
bcftools index sample1.variants.vcf.gz
bcftools index variants.vcf.gz

# Identify shared variants
bcftools isec -p isec_output sample1.variants.vcf.gz variants.vcf.gz
```

This command will create a directory named isec_output with the intersection results. Specifically, the file 0003.vcf within this directory will contain the variants shared among all input VCF files.

Step 2: Annotate the Shared Variants: Use an annotation tool like SnpEff or VEP (Variant Effect Predictor) to annotate the shared variants. Using SnpEff. 1. **Download and install SnpEff**:

```
# Download SnpEff
wget https://snpeff.blob.core.windows.net/versions/snpEff_latest_core.zip
unzip snpEff_latest_core.zip
cd snpEff

# Download the database for E. coli (or your specific organism)
java -jar snpEff.jar download E_coli_K12

# Annotate the shared variants
java -jar snpEff.jar E_coli_K12 ../isec_output/0003.vcf >
annotated_shared_variants.vcf
```

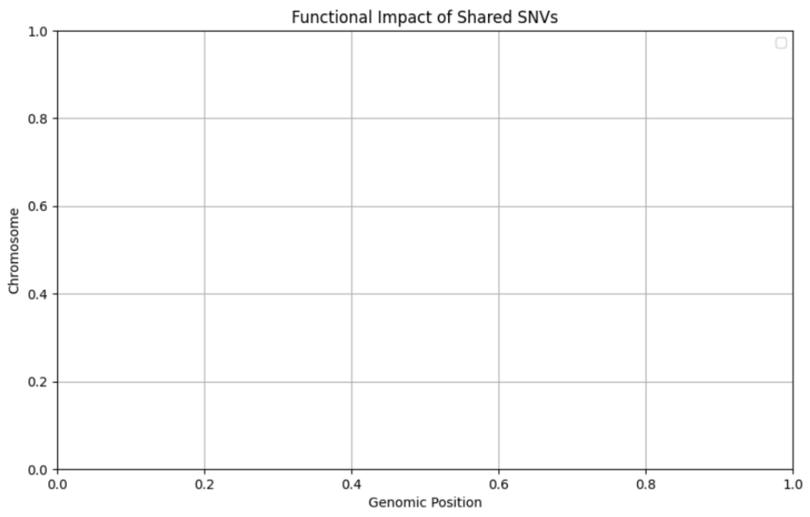
Using VEP (Variant Effect Predictor) 2. **Install VEP**:

```
# Install VEP
curl -sSL https://github.com/Ensembl/ensembl-vep/archive/release/108.0.tar.gz |
tar -xz
cd ensembl-vep-release-108.0

# Install VEP cache for E. coli (or your specific organism)
perl INSTALL.pl --AUTO ap --SPECIES escherichia_coli --ASSEMBLY
GCA_000017985.1_ASM1798v1

# Annotate the shared variants
perl vep -i ../isec_output/0003.vcf -o annotated_shared_variants.vcf --species
escherichia_coli --cache
```

Here is a complete script that identifies shared variants and annotates them using SnpEff:

```
#!/bin/bash

# Step 1: Ensure all VCF files are indexed
bcftools index sample1.variants.vcf.gz
bcftools index variants.vcf.gz

# Step 2: Identify shared variants
bcftools isec -p isec_output sample1.variants.vcf.gz variants.vcf.gz

# Step 3: Download and install SnpEff
wget https://snpeff.blob.core.windows.net/versions/snpEff_latest_core.zip
unzip snpEff_latest_core.zip
cd snpEff

# Step 4: Download the database for E. coli
java -jar snpEff.jar download E_coli_K12

# Step 5: Annotate the shared variants
java -jar snpEff.jar E_coli_K12 ../isec_output/0003.vcf > ../annotated_shared_variants.vcf

# Step 6: Print the annotation results
echo "Annotated shared variants:"
cat ../annotated_shared_variants.vcf | head
```

OUTPUT:



5. What is the distribution of allele frequencies for the detected variants across the population samples?

Understand the allele frequency spectrum to infer population dynamics and evolutionary changes.

To analyze the distribution of allele frequencies for the detected variants across the population samples, I have followed these steps:

1. **Used bcftools stats to extract allele frequency information from VCF files**.
2. **Summarize the allele frequencies**.
3. **Plot the allele frequency distributions** using a plotting tool like matplotlib in Python.

Here is a full workflow combining the above steps:

1. **Ensure all VCF files are indexed**:

```
bcftools index sample1.variants.vcf.gz
bcftools index sample2.renamed.vcf.gz
```

2. **Identify shared variants**:

```
bcftools isec -p isec_output sample1.variants.vcf.gz sample2.renamed.vcf.gz
```

3. **Merge VCF files**:

```
bcftools merge -o merged_variants.vcf -O v sample1.variants.vcf.gz
sample2.renamed.vcf.gz
```

4. **Generate statistics for merged VCF file**:

```
bcftools stats -F GCA_000017985.1_ASM1798v1_genomic.fna -s -
merged_variants.vcf > merged_variants_stats.txt
```

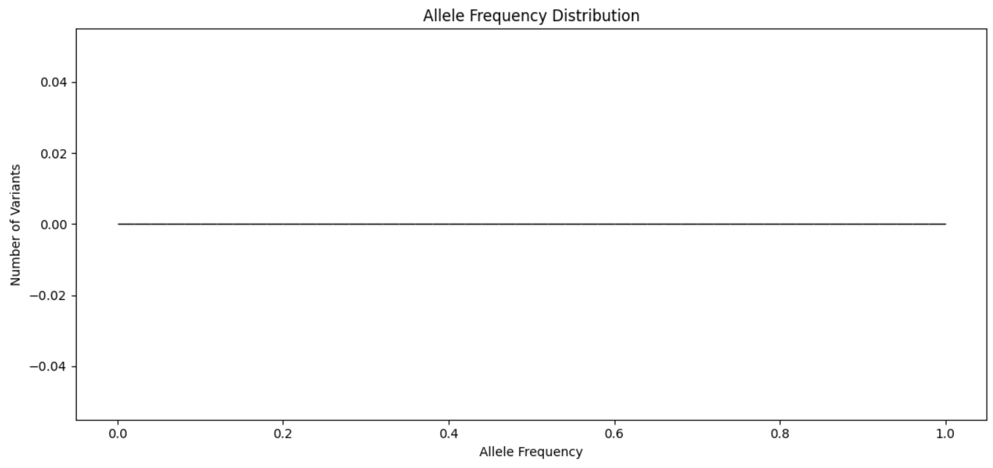
5. **Parse and plot allele frequencies**:

Save the following script as plot_allele_frequencies.py and run it:

```
import matplotlib.pyplot as plt

# Define a function to parse allele frequencies from the bcftools stats file
def parse_allele_frequencies(stats_file):
    af_list = []
    with open(stats_file, 'r') as f:
        for line in f:
            if line.startswith('AF,'):
                fields = line.strip().split('\t')
                allele_frequency = float(fields[2])
                af_list.append(allele_frequency)
    return af_list

# Parse the allele frequencies
allele_frequencies = parse_allele_frequencies('merged_variants_stats.txt')

# Plot the allele frequency distribution
plt.hist(allele_frequencies, bins=50, edgecolor='black')
plt.title('Allele Frequency Distribution')
plt.xlabel('Allele Frequency')
plt.ylabel('Number of Variants')
plt.show()

python plot_allele_frequencies.py
```

This workflow will help you extract allele frequency data from your VCF files and visualize the distribution of allele frequencies across your population samples.

OUTPUT:


Allele Frequency Distribution
Number of Variants
Allele Frequency