**Workflow: Treat A as Reference and Compare B**

1. **Quality Check**

# Download FastQC (latest version)

> wget https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc\_v0.12.1.zip

# Unzip the downloaded file

> unzip fastqc\_v0.12.1.zip

# Make the FastQC program executable

> cd FastQC

> chmod +x fastqc

**#** FastQC quality check

> ./FastQC/fastqc A\_R1.fastq.gz A\_R2.fastq.gz (paired-end)

> ./FastQC/fastqc \*.fastq.gz (For multiple files)

# raw fastq file trimming using bbmap(Install via Conda)

> conda install -c bioconda bbmap

# to find path for adapter file(for adapter trimming)

> bbduk.sh in1=sample\_R1.fastq in2=sample\_R2.fastq \

out1=trimmed\_R1.fastq out2=trimmed\_R2.fastq \

ref=/full/path/to/adapters.fa ktrim=r k=23 mink=11 hdist=1 tpe tbo threads=40

# to find path for file which have adapter sequences

>find ~/miniconda3/envs/ -name adapters.fa

1. **Assemble Sample A into a Reference**

Since A is raw reads, you’ll need to assemble them to make a usable pseudo-reference genome.

You can use:

* SPAdes (good for small/medium genomes)
* MEGAHIT (very fast, for larger genomes)
* Flye (if long reads)

[SPAdes](https://github.com/ablab/spades) (Recommended for small to medium genomes)

* Pros:
  + High accuracy
  + Handles paired-end and mate-pair reads well
  + Easy to run
* Cons:
  + Not optimal for large genomes >1 Gb

#SPAdes assembly (In SPAdes folder)

> bin/spades.py --isolate -1 A\_R1.fastq.gz -2 A\_R2.fastq.gz -o spades\_assembly -t 36

1. **Assembly Quality Check**
2. **QUAST**

* QUAST is the go-to tool for evaluating assembly quality.

#Installation (Linux)

> sudo apt update

> sudo apt install quast

#QUAST quality check

> quast.py -o quast\_output /path\_to/scaffolds.fasta

1. **BUSCO**

* BUSCO (Benchmarking Universal Single-Copy Orthologs) checks how complete your assembly is in terms of expected genes.

#Installation (Linux)

> sudo apt install busco

# to download a lineage dataset (e.g., for rice)

>busco --download lineage\_dataset embryophyta\_odb10

#make new environment in conda for BUSCO

> conda create -n busco\_env -c bioconda -c conda-forge busco

> conda activate busco\_env

#running BUSCO

> busco -i /mnt/path\_to/SPAdes-4.1.0-Linux/spades\_assembly/scaffolds.fasta -l embryophyta\_odb10 -o busco\_1A\_output -m genome --cpu 20

1. **Alignment B to A (Assembly)**

#BWA installation

> sudo apt update

> sudo apt install bwa

#Index A’s Assembly (pseudo-reference)

> bwa index path\_to/scaffolds.fasta

#Align Sample B to A’s Assembly

> bwa mem path\_to/scaffolds.fasta B\_R1.fastq.gz B\_R2.fastq.gz > B\_vs\_A.sam

1. **Convert SAM to Sorted BAM**

#Samtools installation

> sudo apt install samtools

#converting SAM to BAM

> samtools view -@ 40 -Sb B\_vs\_A.sam | samtools sort -o B\_vs\_A.sorted.bam

> samtools index -@ 40 B\_vs\_A.sorted.bam

#to check alignment statistics

> samtools flagstat -@ 40 B\_vs\_A.sorted.bam

1. **Call Variants: B vs A**

This will give you all the **SNPs** and **INDELs** in B compared to A.

#Use bcftools to find variants in B with respect to A's genome

> bcftools mpileup -f A\_assembly/scaffolds.fasta B\_vs\_A.sorted.bam -Ou | bcftools call -mv -Ov --threads 40 -o B\_vs\_A.vcf

1. **Summarize Similarities and Differences**

#Count total SNPs and INDELs

> grep -v "^#" B\_vs\_A.vcf | wc -l

# Separate SNPs and INDELs

> bcftools stats B\_vs\_A.vcf > B\_vs\_A.stats

#Filtering in bcftools

* QUAL<50 — Only keep variants with a quality score of 50 or higher.
* INFO/DP<20 — Minimum read depth of 20.
* MQ<40 — Mapping quality at least 40.
* INFO/MQ0F>0.1 — Remove variants where >10% of reads have mapping quality 0 (unreliable).

> bcftools filter -s LOWQUAL \

-e 'QUAL<50 || INFO/DP<20 || MQ<40 || INFO/MQ0F>0.1' \

B\_vs\_A.vcf -o B\_vs\_A\_filtered.vcf

#After filtering, run this to keep only PASS variants

> bcftools view -f PASS B\_vs\_A\_filtered.vcf -o B\_vs\_A\_pass\_strict.vcf

#converting file to a normat txt file

> bcftools stats B\_vs\_A\_pass\_strict.vcf > B\_vs\_A\_filtered\_strict\_stats.txt

1. **Annotation**

* Using snpEff for annotation

it will annotate the VCF file using the **Oryza\_indica** genome database, that database is available in your snpEff setup.

snpEff -v -canon -noLog Oryza\_indica B\_vs\_A\_pass\_strict.vcf > B\_vs\_A\_pass\_strict\_annotated.vcf