# Standard Operating Procedure (SOP)

## Title: Bioinformatics Workflow for Genome Assembly and Gene Variation Analysis of *Microsporidia sp.* from *Anopheles* Mosquitoes

Version: 1.0

Prepared by: [Your Name]

Date: [Insert Date]

Institution: [Insert Lab/Organization Name]

## 1. Purpose

This SOP outlines the standardized bioinformatics protocol for processing DNBSeq 150 bp paired-end reads from *Microsporidia*-infected *Anopheles* mosquito tissues. The pipeline performs quality control, host read removal, microbial decontamination, genome assembly, annotation, and gene variation analysis including clustering.

## 2. Scope

This procedure is designed for graduate-level bioinformatics practitioners and is intended to facilitate reproducible genomic analyses of microsporidian symbionts. It supports comparative genomics and molecular epidemiology studies of microsporidia across different geographical regions.

## 3. Requirements

### 3.1. Software & Tools

Install the following bioinformatics tools via Conda:

```bash
conda install -y -c bioconda fastqc multiqc bwa samtools kraken2 \

unicycler quast busco augustus genemarks repeatmodeler repeatmasker \

blast mafft orthofinder
```

## 4. Input

- Paired-end sequencing reads: `*.fq.gz` files from DNBSeq platform

- Reference genomes for host species: *Anopheles arabiensis* and *A. gambiae*

- Kraken2 database (e.g., `minikraken_8GB`)

- Augustus model species or trained parameters for Microsporidia

---

## 5. Procedure

### 5.1. Quality Control

Tools: FastQC, MultiQC

```bash
fastqc raw_reads/*.fq.gz -o qc_output/

multiqc qc_output/
```

~~~
```
~~~

### 5.2. Host Read Removal

Tools: BWA, Samtools

```bash
bwa index host_reference.fa
bwa mem host_reference.fa reads_R1.fq.gz reads_R2.fq.gz | samtools view -bS - | samtools sort -o host_mapped.bam
samtools index host_mapped.bam
```

### 5.3. Decontamination

Tools: Kraken2

```bash
kraken2 --db minikraken_8GB --paired clean_R1.fq clean_R2.fq \
--report kraken_report.txt --unclassified-out clean_R#.fq --use-names
```

### 5.4. De Novo Genome Assembly

Tool: Unicycler

```bash
unicycler -1 clean_R1.fq -2 clean_R2.fq -o assembly_dir
```

### 5.5. Gene Prediction

Tools: Augustus, GeneMarkS

```bash
augustus --species=microsporidia assembly.fasta > augustus_output.gff
gmsn.pl --seq assembly.fasta --genome-type euk --output gms_output
```

### 5.6. Repeat Masking

Tools: RepeatModeler, RepeatMasker

```bash
BuildDatabase -name genome_db assembly.fasta
RepeatModeler -database genome_db -pa 4
```

### 5.7. Genome Quality Assessment

Tools: QUAST, BUSCO

```bash
quast assembly.fasta -o quast_output
busco -i assembly.fasta -l microsporidia_odb10 -m genome -o busco_output
```

### 5.8. Gene Clustering and Variation Analysis

Tool: OrthoFinder

```bash
orthofinder -f protein_directory/
```

## 6. Expected Output

- Quality control reports

- Filtered read files

- Assembled genome in FASTA format

- GFF annotations from Augustus and GeneMarkS

- Repeat annotation files

- BUSCO and QUAST reports

- OrthoFinder clustering results

## 7. Troubleshooting

- Ensure tools are correctly installed with appropriate versions.

- Validate paths and file names, especially for large paired-end datasets.

- For Augustus, consider training a species-specific model for better gene prediction.

## 8. References

1. FastQC - https://www.bioinformatics.babraham.ac.uk/projects/fastqc

2. MultiQC - Ewels et al., Bioinformatics, 2016

3. BWA - Li & Durbin, Bioinformatics, 2009

4. Samtools - Danecek et al., Gigascience, 2021

5. Kraken2 - Wood et al., Genome Biol, 2019

6. Unicycler - Wick et al., PLOS Comp Biol, 2017

7. Augustus - Stanke et al., Nucleic Acids Res, 2004

8. GeneMarkS - Besemer et al., Nucleic Acids Res, 2001

9. RepeatModeler - Flynn et al., PNAS, 2020

10. BUSCO - Simão et al., Bioinformatics, 2015

11. QUAST - Gurevich et al., Bioinformatics, 2013

12. OrthoFinder - Emms & Kelly, Genome Biol, 2019

## Appendix B: Simplified For-Loop Version (Early Learners)

```python
import os

# Define paths
raw_reads_dir = "all_reads/other_reads"
output_dir = "output"
kraken_db = "/mnt/lustre/bsp/DB/KRAKEN2/minikraken_8GB_20200312"

# Ensure output directories exist
os.makedirs(output_dir, exist_ok=True)
```

```
# Loop through paired-end files
for fq1 in os.listdir(raw_reads_dir):
    if fq1.endswith("_1.fq.gz"):
        fq2 = fq1.replace("_1.fq.gz", "_2.fq.gz")
        fq1_path = os.path.join(raw_reads_dir, fq1)
        fq2_path = os.path.join(raw_reads_dir, fq2)
        sample = fq1.replace("_1.fq.gz", "")

        # Kraken2 classification
        os.system(f"kraken2 --db {kraken_db} --paired --classified-out {output_dir}/{sample}_classified#.fq "
                  f"--unclassified-out {output_dir}/{sample}_unclassified#.fq --report {output_dir}/{sample}_report.txt "
                  f"{fq1_path} {fq2_path}")

        # Unicycler assembly
        os.system(f"unicycler -1 {output_dir}/{sample}_unclassified_1.fq -2 {output_dir}/{sample}_unclassified_2.fq "
                  f"-o {output_dir}/unicycler_{sample} --no_pilon --threads 32")
```