# *PoreTally on HPC*

## What is PoreTally?

PoreTally is a de novo assembler for third generation sequencing technologies such as the Oxford Nanopore MinION. PoreTally assembles reads generated by these technologies, to produce a consensus sequence for a given organism or sample. The assembler uses a k-mer based approach, where it compares and clusters reads that share k-mers of a certain length. By clustering reads that share k-mers, PoreTally can identify and assemble regions of high read coverage and generate a consensus sequence for these regions. PoreTally is designed to be computationally efficient and handle large datasets, while still producing high-quality assemblies. It is also able to handle different types of data, such as long reads and error-prone reads, that are common in third-generation sequencing technologies. PoreTally can be used to assemble a wide range of organisms, including bacteria, fungi, plants, and animals.

Links:

GitHub

Official Paper

## Versions Available:

The following versions are available on the cluster:

- poreTally

## How to load PoreTally?

To load PoreTally, use the following commands:

```
#Load the PoreTally module
module load bio/poretally
```

To verify if the module is loaded correctly, use the following command,

```
# List all the module loaded in the environment
module list
```

In a fresh environment, this should show only poretally module listed.

## How to use PoreTally?

Here is the syntax to run poreTally,

```
poreTally run_benchmark \
                -w path/to/working_directory \
                -t 16 \
                -f path/to/fast5_files
                -s path/to/slurm_header.json \
                -r reference_genome.fasta \
                -f path/to/fast5_files_dir \
                -p minimap2_miniasm canu path/to/my_pipeline.yaml \
                -g gene_annotation.gff \
                -i user_info.yaml \
                --git
git@github.com:username/repository_to_store_results.git \
                --reads-fastq path/to/read_fastqs_dir
```

Here is a general outline for using poretally in HPC,

1. Prepare the input data: The input data for PoreTally is usually a set of FASTQ or FASTA files that contain the sequencing reads. These files should be transferred to the HPC cluster.
2. Load the required modules: Before running PoreTally, you need to load the required modules for the software, such as the version of the compiler used to build the software, and any dependencies.
3. Specify the parameters: PoreTally has several command-line options that allow you to specify the parameters for the assembly, such as the k-mer size, the number of threads to use, etc.

4. Submit the job: Once you have prepared the input data, loaded the required modules, and specified the parameters, you can submit the job to the HPC cluster using a scheduler, such as Slurm.

See the GitHub link provided above for the full command line options and tutorials.

Here is a sample slurm script to run poretally using slurm,

```
#!/bin/bash

#SBATCH --job-name=poretally
#SBATCH --output=poretally.out
#SBATCH --error=poretally.err
#SBATCH --p main
#SBATHC --qos main
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=16
#SBATCH --mem=64GB
#SBATCH --time=12:00:00

# load PoreTally
module load bio/poretally

# specify input and output files
reads=input.fastq
output=output.fasta

# run PoreTally
poreTally -i $reads -o $output -k 31 -t 16
```

In this script, we first specify the job name, output and error file names, number of nodes, tasks and cpus per task and memory required. Then we load the PoreTally module. Next, we specify the input and output files. Finally, we run PoreTally using the input and output files, a k-mer length of 31, and utilizing 16 threads.

## *Where to find help?*

If you are confused or need help at any point, please contact OIT at the following address.

https://ua-app01.ua.edu/researchComputingPortal/public/oitHelp