Redundans on HPC

What is Redundans?

Redundans is a cutting-edge software for genome assembly and redundancy removal. It is designed for use in the assembly of large genomes, particularly in the case of plant and animal genomes that often have long repeating sequences and high levels of heterozygosity. The software uses a unique error correction algorithm that can handle high levels of error, ensuring that the final assembly is accurate and efficient.

The ultimate goal of Redundans is to produce a single consensus sequence that is both accurate and of reduced size, thus providing a more streamlined and usable final product. The software is a valuable tool for researchers and scientists working in the fields of genomics and computational biology, as it enables them to efficiently and accurately assemble and analyze large genomes.

Links:

Academic Paper

GitHub

Manual

Versions Available:

The following versions are available on the cluster:

- Redundans 0.14a

How to load Redundans?

To load Redundans, use the following commands:

```
#Load the Redundans module
module load bio/redundans
```

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 only load Redundans module without any dependencies.

How to use Redundans?

To use redundans, user should follow the following steps,

1. Preprocessing: The first step is to prepare the input reads for assembly. This may involve quality control, trimming, and filtering of the reads to remove contaminants and low-quality data.
2. Assembly: The next step is to run the assembly process, which involves aligning the input reads to form contigs and scaffolds. Redundans uses a combination of overlapping, de novo, and reference-based approaches to construct the assembly.
3. Error Correction: After the initial assembly, Redundans uses its unique error correction method to identify and correct errors in the assembly. This step is critical for ensuring the accuracy of the final assembly.
4. Consensus Generation: The final step is to generate a consensus sequence from the assembly. Redundans uses a majority voting approach to determine the most likely nucleotide at each position in the assembly, resulting in a single consensus sequence.
5. Post-Processing: After the consensus sequence has been generated, the final assembly can be subjected to further post-processing, such as annotation, gene prediction, and genome analysis.

There is a full tutorial for a test run on the GitHub page for reference.

Sample Tutorial

Here is a sample threaded slurm script to submit a job,

```
#!/bin/bash
#SBATCH --job-name=redundans
#SBATCH --nodes=1
#SBATCH --cpus-per-task=16
#SBATCH --mem=32GB
#SBATCH --partition=general
#SBATCH --output=bio-redundans.out
#SBATCH --p main
#SBATCH --qos main
# load required modules
module load bio/redundans

# specify the input reads and reference genome (if available)
INPUT_READS=reads.fastq
REFERENCE=reference.fasta

# specify the number of threads to use
THREADS=16

# run Redundans assembly process below this line
redundans.py --threads $THREADS --reference $REFERENCE $INPUT_READS -o
assembly_output
```

Submit the job to cluster using,

```
sbatch script.sh
```

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