Stacks on HPC cluster:

What is Stacks ?

Stacks is designed to work with any restriction-enzyme based data, such as GBS, CRoPS, and both single and double digest RAD. *Stacks* is designed as a modular pipeline to efficiently curate and assemble large numbers of short-read sequences from multiple samples. *Stacks* identifies loci in a set of individuals, either *de novo* or aligned to a reference genome (including gapped alignments), and then genotypes each locus. *Stacks* incorporates a maximum likelihood statistical model to identify sequence polymorphisms and distinguish them from sequencing errors. *Stacks* employs a Catalog to record all loci identified in a population and matches individuals to that Catalog to determine which haplotype alleles are present at every locus in each individual.

Stacks is implemented in C++ with wrapper programs written in Perl. The core algorithms are multithreaded via OpenMP libraries and the software can handle data from hundreds of individuals, comprising millions of genotypes.

The full documentation for the Stacks is found in the following links:

Documentation

Official Website

Versions Available:

- Stacks v1.4.2
- Stacks v2.53
-

How to load a version of Stacks?

To load a version of Stacks on the HPC, use the following command:

```
module avail bio/stacks
```

The version will be listed. To use a version of software, use following command:

```
module load bio/stacks/1.4.2
```

Verify by using this command:

```
module list
```

Stacks has a dependency on Perl. So, the output should be two modules- Perl and Stacks.

How to use Stacks on the cluster?

Since stacks is used to run the whole pipeline, it is preferred to run a batch script rather than using a bash terminal for better efficiency and

Usage:

```
process_radtags  -> Examines raw reads from an Illumina sequencing run
and first, checks that the barcode and the RAD cutsite are intact, and
demultiplexes the data.

process_shortreads -> Performs the same task as process_radtags for
fast cleaning of randomly sheared genomic or transcriptomic data, not
for RAD data.

clone_filter  -> Designed to identify PCR clones.

kmer_filter -> Allows paired or single-end reads to be filtered
according to the number or rare or abundant kmers they contain.

ustacks -> Takes as input a set of short-read sequences and aligns
them into exactly matching stacks (or putative alleles).

cstacks -> Builds a catalog from any set of samples processed by the
ustacks or pstacks programs.

sstacks  -> Sets of stacks, i.e. putative loci, constructed by the
ustacks program can be searched against a catalog produced by cstacks.
```

```
tsv2bam  -> Transpose data so that it is oriented by locus, instead of
by sample.

gstacks -> Examines a RAD data set one locus at a time, looking at all
individuals in the metapopulation for that locus.

populations -> Analyze a population of individual samples computing
several population genetics statistics as well as exporting a variety
of standard output formats.
```

Pipeline to run either a genetic map or population analysis.:

```
denovo_map.pl ->  Executes the pipeline, running ustacks to assemble
loci in each individual de novo, calling SNPs in each assembled locus.
It will then executing cstacks to build the catalog followed
by sstacks to match either the parents and progeny, or all the generic
samples against the catalog. Next, it will run tsv2bam to transpose
data from being store per-sample to be stored per-locus, then it will
run gstacks to assemble paired-end contigs (if paired-end data is
provided) and re-call SNPs using the population-wide data.


ref_map.pl  -> Executes the pipeline, running gstacks to build and
genotype single- or paired-end data and call SNPs using the
population-wide data per locus.
```

Other binaries and scripts are found on the following location:

```
ls /share/apps/stacks/bin/
```

See documentation for the use case of the additional binaries.

The Script:

To run a slurm job, the user must prepare input files. Users can use other software to do so. However, it is advised to purge all the modules before starting to use slurm script.

```
module purge
```

Use the following template for the script,

```
#!/bin/bash
#SBATCH -p threaded
#SBATCH -q threaded
#SBATCH --mem-per-cpu=4G
#SBATCH -n 1
#SBATCH -c 16

#Load the module
module load bio/stacks

#Go to the test directory
cd $SLURM_SUBIMT_DIR

# Run stacks on the population map data
denovo_map.pl -T 8 -M 4 -o ./stacks/ --samples ./samples --popmap
./popmaps/popmap
# Pass --paired for pair end data
```

Schedule the job with the following sbatch command.

sbatch script.sbatch

All the processed files will be generated in the same directory as the sbatch script.

Where to find help?

If you are stuck on some part or need help at any point, please contact OIT at the following address.

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