UF HPC Training

NGS Mapping and Assembly

October 16, 2014

1. Log into UF HPC's Galaxy instance: http://galaxy.rc.ufl.edu/
2. Get some data:
    a. Shared Data: Data Libraries: Training datasets: wine_yeast.100K.fq
3. NGS: QC and manipulation: **FASTQ Groomer**
    a. Input FASTQ quality scores type: Sanger
4. NGS: QC and manipulation:FastQC: **FastQC:Read QC**
    a. Use the defaults or add a title for easier reference later
    b. Notice poor quality at ends of reads
5. NGS:QC and manipulation:**FASTQ Quality Trimmer**
    a. Window size: 5
    b. Quality score: 20
    c. Rerun FastQC on trimmed dataset
6. NGS: Mapping: **Map with Bowtie for Illumina**
    a. Use a built-in index:
    b. Select: S. cerevisiae (CGD) 2011
7. NGS:SAM Tools:**SAM-BAM**
    a. Convert your SAM file to BAM using the defaults
8. Click on the View in Trackster icon in the BAM results window
9. Select a chromosome and see where reads mapped

From the command line:

1. Login to HiPerGator: `ssh <user>@gator.rc.ufl.edu`
2. Go to your scratch space, make a directory called bowtie_test and cd into it
    a. `cd /scratch/lfs/$USER`
    b. `mkdir bowtie_test`
    c. `cd bowtie_test`
3. Copy/scratch/lfs/bio/training/2014-10-16/bowtie.pbs to bowtie_test
    a. `cp /scratch/lfs/bio/training/2014-10-16/bowtie.pbs .`
4. Edit the bowtie.pbs file to have your e-mail
    a. `nano bowtie.pbs`
5. Submit the Bowtie run
    a. `qsub bowtie.pbs`

6. Now let's look at Velvet: make a directory called velvet_test and cd into it
    a. `cd ..` (moves you up one directory)
    b. `mkdir velvet_test`
    c. `cd velvet_test`
7. Copy `/scratch/lfs/bio/training/2014-10-16/velet.pbs` to velvet_test
    a. `cp /scratch/lfs/bio/training/2014-10-16/velvet.pbs .`
8. Edit the velvet.pbs file to have your e-mail.
    a. `nano velvet.pbs`

9. Submit the velvet run
   a. `qsub velvet.pbs`
10. Compare the resulting contig files

Contents of velvet.pbs:

```
#!/bin/bash
#
#PBS -N velvet
#PBS -M <your e-mail>
#PBS -m abe
#PBS -o velvet.test.out
#PBS -e velvet.test.err
#PBS -l nodes=1:ppn=4
#PBS -l pmem=900mb
#PBS -l walltime=00:05:00
#

cd $PBS_O_WORKDIR

module load velvet

# Make and output
# directory for Velvet
mkdir test_run

#Run Velvet with kmer of 21
velveth test_run/ 21 -fastq -short \
   /scratch/lfs/bio/training/2014-10-16/wine_yeast.100k.fq
velvetg test_run/ -min_contig_lgth 500

#Get things ready to use threaded (OMP) version of Velvet
#Set OMP_THREAD_LIMIT--should be the same as ppn above
export OMP_THREAD_LIMIT=$PBS_NUM_PPN

#Set OMP_NUM_THREADS--should be 1 lower than ppn
NUM_THREADS=$((PBS_NUM_PPN-1))
export OMP_NUM_THREADS=$NUM_THREADS

echo Limiting Velvet to $PBS_NUM_PPN threads total with $NUM_THREADS slave
threads.

# Rerun Velvet using a kmer of 51, and the threaded version
# Note there isn't a flag to tell Velvet how many threads to use
# It will use all the cores on a node unless you tell it not to with
# $OMP_THREAD_LIMIT and $OMP_NUM_THREADS

mkdir test_run_kmer51
velveth_max99_OMP test_run_kmer51/ 51 -fastq -short \
  /scratch/lfs/bio/training/2014-10-16/wine_yeast.100k.fq
velvetg_max99_OMP test_run_kmer51/ -min_contig_lgth 500
```

Here are the PBS directives, the information for the scheduler:

In addition to CPUs, RAM and walltime, this has information for log files, and e-mail notification.

PBS directive lines start with #PBS and should be at the top of the file

Remember to run out of scratch space—the command, `cd $PBS_O_WORKDIR`, changes from home to where you typed qsub. This should be part of most scripts you make.

Note loading of the module for the application we are running, use the module system to save headaches!

Run velvet once using kmer of 21

Note that for the threaded version of Velvet, you need to set some environment variables

Run velvet again, this time using multiple CPUs, and kmer of 51

Note that this script runs Velvet twice as an example. You would not typically want to do this…*Either* run on a single core, like the first time through, and adjust resource requests to nodes=1:ppn=1, *or* run on multiple cores, and set OMP_THREAD_LIMIT and OMP_NUM_THREADS as in the example.