🐙 Day 2 PM, Working with NGS data on Hydra-3

How to begin

Copy the Day2 directory in /pool/cluster0/workshop/Hydra_workshop_2015/Day2/ to your working directory in /pool/cluster0/workshop/username/

Running jobs on Hydra

1. Quality Assessment/Quality check (QA/QC) with FastQC - module `bioinformatics/fastqc/0.10.1`

FastQC (www.bioinformatics.babraham.ac.uk/projects/fastqc/) aims to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines.

A. Generate a qsub file to run your data through FastQC - Qsub generator (https://hydra-3.si.edu/tools/QSubGen/) commands:

```
    fastqc sequences.fq
```

-- **Please use all minimum values for memory and CPUs for this job.**

- Example Qsub Generator output:

```
# /bin/bash
# ----------------Parameters---------------------- #
#$ -S /bin/bash
#$ -q sThC.q
#$ -cwd
#$ -j y
#$ -N FastQC
#$ -o FastQC.log
#$ -m abe
#$ -M <username>@si.edu
#
# ----------------Modules------------------------- #
module load bioinformatics/fastqc/0.10.1
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
#
fastqc sequences.fq
#
echo = `date` job $JOB_NAME done
```

B. Transfer Qsub generator file to hydra from your local computer. *Hint: you can use* `scp` *or copy into a new file in* `vi` *.*

C. Execute qsub script to run FastQC on Hydra-3

- `qsub <filename>.qsub`

D. Download output (zip file) to your local computer

- HINT: use can use the `scp` command

E. Open HTML file and assess results.

2. Adaptor/Quality Trimming with TrimGalore - module `bioinformatics/trimgalore/0.4.0`

Trim Galore! (www.bioinformatics.babraham.ac.uk/projects/trim_galore/) is a wrapper script to automate quality and adapter trimming as well as quality control.

A. Generate a qsub file to run your data through FastQC - Qsub generator commands:
`trim_galore -q 30 --fastqc --clip_R1 5 --three_prime_clip_R1 5 sequences.fq`

-- *Note: Default quality encoding is Phred 33 (Sanger/Illumina 1.9). Please verify that the encoding is correct

from the results of FastQC.

-- **Please use all minimum values for memory and CPUs for this job.**

- Example Qsub Generator output:

```
# /bin/bash
# ----------------Parameters---------------------- #
#$ -S /bin/bash
#$ -q sThC.q
#$ -cwd
#$ -j y
#$ -N Trimgalore
#$ -o Trimgalore.log
#$ -m abe
#$ -M <username>@si.edu
#
# ----------------Modules------------------------- #
module load bioinformatics/trimgalore/0.4.0
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
#
trim_galore -q 30 --fastqc --clip_R1 5 --three_prime_clip_R1 5 sequences.fq
#
echo = `date` job $JOB_NAME done
```

B. Run TrimGalore on Hydra-3

- $ `qsub <filename>.qsub`

C. Download output of FastQC run (zip file) to your local computer - HINT: use can use the `scp` command.

D. Open HTML file and assess results.

3. Assembly with Velvet - module `bioinformatics/velvet/1.2.10`

Velvet (https://www.ebi.ac.uk/~zerbino/velvet/) is a de novo genomic assembler specially designed for short read sequencing technologies. Velvet runs in parallel (OPENMP: multi-thread). The velvet assembler runs in a two set process; by first running `velveth` and then by running `velvetg`. `velveth` helps you construct the dataset for the following program, `velvetg`, and indicate to the system what each sequence file represents. `velvetg` is the core of Velvet where the de Bruijn graph is built then manipulated.

A. Generate a qsub file to assemble your data (sequences.fq) with `velveth`. `velveth` takes in a number of sequence files, produces a hashtable, then outputs two files in an output directory (creating it if necessary), Sequences and Roadmaps, which are necessary to `velvetg`.

- Qsub generator commands for `velveth`:

```
export OMP_NUM_THREADS=$NSLOTS
export  MAXKMERLENGTH='51'
velveth ecoli.33 33 -short -fastq sequences.fq
```

-- **Please use all minimum values for memory and CPUs for this job.**

  - Example Qsub Generator output:

```
# /bin/bash
# ----------------Parameters---------------------- #
#$ -S /bin/bash
#$ -pe mthread 2
#$ -q sThC.q
#$ -l mres=1G
#$ -cwd
#$ -j y
#$ -N velveth_short
#$ -o velveth_short.log
#$ -m abe
#$ -M <username>@si.edu
#
# ----------------Modules------------------------- #
module load bioinformatics/velvet/1.2.10
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo + NSLOTS = $NSLOTS distributed over:cat $PE_HOSTFILE
#
export OMP_NUM_THREADS=$NSLOTS
export MAXKMERLENGTH='51'
velveth ecoli.33 33 -short -fastq sequences.fq
#
echo = `date` job $JOB_NAME done
```

B. Run `velveth` on Hydra-3

- $ `qsub velveth.qsub`

C. Generate a qsub file for `velvetg` to assemble your data in the directory generated from `velveth`.

- Qsub generator commands for `velvetg`:

```
export OMP_NUM_THREADS=$NSLOTS
export  MAXKMERLENGTH='51'
velvetg ecoli.33
```

-- **Please use all minimum values for memory and CPUs for this job.**

  - Example Qsub Generator output:

```
# /bin/bash
# ----------------Parameters---------------------- #
#$ -S /bin/bash
#$ -pe mthread 2
#$ -q sThC.q
#$ -l mres=1G
#$ -cwd
#$ -j y
#$ -N velvetg_short
#$ -o velvetg_short.log
#$ -m abe
#$ -M <username>@si.edu
#
# ----------------Modules------------------------- #
module load bioinformatics/velvet/1.2.10
#
# ----------------Your Commands------------------- #
#
echo + `date` job $JOB_NAME started in $QUEUE with jobID=$JOB_ID on $HOSTNAME
echo + NSLOTS = $NSLOTS distributed over:cat $PE_HOSTFILE
#
export OMP_NUM_THREADS=$NSLOTS
export MAXKMERLENGTH='51'
velvetg ecoli.33
#
echo = `date` job $JOB_NAME done
```

D. Transfer Qsub generator file to hydra from your local computer. *Hint: you can use `scp` or copy into a new file in `vi`.*

E. Run `velvetg` on Hydra-3

- $ `qsub velvetg.qsub`

F. Count the number of sequences in your output assembly.

- Navigate to the output directory with the output: `cd ecoli.33`
- Assembly output is called `contigs.fa`
- Count number of sequences; *Hint - use skills from Day 1 PM Practical*

For advanved users

4. Assembly with Trinity (OPTIONAL - now has trimmomatic included) - module `bioinformatics/trinity/2.0.6`

Trinity (http://trinityrnaseq.github.io/) assembles transcript sequences from Illumina RNA-Seq data. Trinity runs in an OPENMP parallelized environment and required a large amount of memory. Remember to use flags for an parallel environment. *Hint: use mthreads and himem flags*

A. Generate a qsub file to assemble your data with Trinity.

- Qsub generator commands:

```
Trinity --seqType fq --single sequences.fa --CPU $NSLOTS --max_memory 20G --full_
cleanup --output Trinity_sequences
```

B. Run Trinity on Hydra-3

C. Count the number of sequences in your output assembly - `Trinity.fasta`