Quality filtering Tag-Seq data

This short guide explains how to quality filter unzipped Tag-Seq fastq files using the fastx_toolkit and custom perl scripts.

Move your samples into their own subdirectories and create a test subset

1. Make sure you are in the correct working directory.
2. Create a small subset file for testing, and put that in a test directory.

```
head -40000  15H_S44_R1_001.fastq > test.fastq
mkdir test
mv test.fastq test
```

3. Create a list of sample names for the sorting script to use

```
ls *001.fastq | cut -d "_" -f 1 > samples.list
```

   a. Now, use this list as input for a bash script that will move each sample into its own subdirectory.

```
organize.sh samples.list
```

   b. Check to see that your directories have been created.

   c. What is in these newly created directories? Let's check.

```
ll 15H/*
```

Trialing each step of the pipeline on your test subset

You will need to run the same series of commands on each file. We will build a job file that we will then feed to a batch script that will submit jobs for each of your newly created subdirectories to run the following series of commands for each read file. First, we will review what each of these commands is doing.

4. First, make a copy of your job submission script (in case of errors) and call it "QF.sh"

```
cp $SCRIPTS/job.sh QF.sh
```

   a. Then, nano into your QF.sh script and add the following command to the end of the file, write out and close the file.

```
RemovePCRDups.pl -i test.fastq -o dedup.fastq -j 5 > dedup_out.txt
```

   b. Then, submit your job to the job scheduler.

```
qsub QF.sh
```

c. While we're waiting for these to complete, what is each script doing? How would you figure it out?

d. Are your jobs finished running? How would you check?

e. Now lets see what new files have been created and what has been written to the standard error file. How many reads did we lose in this step?

```
ll -t

cat dedup_out.txt
```

5. Next, nano into your QF.sh script, comment out your prior 'RemovePCRDups.pl' line using a #, and add the following command to the end of the file, write out and close the file.

```
TagTrimmer.pl -b 1 -e 14 -i dedup.fastq -o NoAdap.fastq > NoAdap_out.txt
```

   a. Then, submit your job to the job scheduler, specifying the standard output and standard error files.

```
qsub QF.sh
```

   b. Check what new files have been created and what has been written to the standard error or output files.

6. Got the hang of it? Let's lather, rinse, and repeat for the remaining steps, commenting out the previous command and adding the next in the series. Note that each command should be written on a single line.

```
fastx_clipper -a AAAAAAAA -l 20 -Q33 -i NoAdap.fastq > clipper.fastq

##############################################################

qsub QF.sh

##############################################################

bbduk.sh in=clipper.fastq ref=/scratch/data/adaptors.fasta stats=stats.txt k=12 overwrite=t
        out=filter.fastq > filter_out.txt

##############################################################

qsub QF.sh

##############################################################

fastq_quality_filter -Q33 -q 20 -p 70 -i filter.fastq > clean.fastq

##############################################################

qsub QF.sh
```

Process all samples in a batch job

7. Now that you’ve got the hang of it, we now need to run this series of commands for each of our raw fastq files. Instead of running each sequentially, we will use a batch script to submit a separate job file for each of the subdirectories you created originally. To do this, open up your QF.sh job file again in nano and uncomment each of the lines in the series. Then, for your first command, replace test.fastq with a wildcard variable $NAME*".fastq". This command will work just as before, save that it will now recognize any file that begins with a sample name and ends with .fastq

```
RemovePCRDups.pl -i $NAME*".fastq" -o dedup.fastq -j 5
```

   a. Then, instead of submitting the single job file, we will pass the job file to the batch script which will submit it for each directory using our previously created samples.list as the guide. The only other argument this batch script requires is a prefix for the standard out and standard error files.

```
batch.sh QF.sh samples.list QualFilt
```

   b. This will take a bit. Go get some coffee. Use ‘qstat’ to check progress. You can also add additional lines to the job script file to get an email notification when your job is complete. Carly likes this option. Eli does not.

   c. Now, what proportion of reads did we discard in each of these steps?

```
wc -l */*.fastq
```

   If you wanted to calculate the proportion of reads remaining after processing, you could do this as:

```
echo “scale=3;clean_number/raw_number”|bc
```