Quality Control of Illumina Data at the Command Line

Quick UNIX Introduction:
UNIX is an operating system like OSX or Windows. The interface between you and the UNIX OS is called “the shell”. There are a few flavors of shell but the MSI standard is bash. The shell is what you see, your environment, when you open PuTTY or Terminal.

UNIX acts like any other operating system and allows you to store and create files then use them during processes (e.g. running a program). Unlike OSX or Windows UNIX is not visual and has to be navigated using simple commands though the shell.

A Note about remote computing:
The goal of this tutorial is to introduce you to using MSI computational resources. This means you are going to use your computer to talk to other computers and tell those computers what to do. This also means that you have to make sure you know how to navigate a different computational environment (UNIX) and the data you need is somewhere MSI systems can access it (i.e. not on your personal computer).

Brief Outline of MSI:

Login Node: When you access MSI systems the login node is the first system that you connect to. The login node is only there to direct you to other MSI systems.
Lab System: A group of computers that can be accessed using the isub command. This is an older system with less computational power behind it. This group of computers is transitioning towards an interactive only system and soon you will not be able to submit batch jobs to Lab.

Itasca System: This is one of the faster, fancier computer systems. It is designed with multiple node computational jobs in mind.

**Mesabi System:** This is the system where you should be doing all of your work. Mesabi is 7 times larger than Itasca and has many queues designed for the type of computational work often done by biologists.

SUs: Service Units (SUs) are MSI method of tracking usage on Itasca and Mesabi. Starting with the 2016 yearly allocation each group will automatically be granted 70,000 SUs and access to Mesabi. A vast majority of groups will not need more than 70,000 SUs but MSI is happy to grant your group more SUs

PBS queuing: PBS is a queuing program that takes care of reserving and ordering jobs to be run on the different systems. PBS takes special commands that allow you to ask for a specific amount of time and computational power. Once you submit your job (i.e. program) it will get in line (in the queue) and will run once room opens up.

Storage: Each group is allotted some amount of storage the default amount is 100GB but your group can request as much storage as you can justify.

**Software that you downloaded:**

This editor will allow you to write and edit scripts and save them to your MSI space. By connecting Komodo Edit to MSI you can write, saved and edit documents and scripts that are saved in MSI space. Komodo Edit is an alternative to learning a true editor such as VI, Nano or Emacs. You should still learn how to use an editor because you might not always have Komodo and the editors can do much fancier tricks.

Preferences->Servers-> + button
Server type: SFTP
Name: MSI
Hostname: login.msi.umn.edu
Username: <your MSI username>
Password: <your MSI password>

**FileZilla Client:** [https://filezilla-project.org/](https://filezilla-project.org/)
Setup: [https://www.msi.umn.edu/support/faq/how-do-i-use-filezilla-transfer-](https://www.msi.umn.edu/support/faq/how-do-i-use-filezilla-transfer-)
This SFTP/FTP client will help you move files into and out of your MSI space.


This editor will allow you to write and edit scripts and save them to your MSI space. We will set this up in class.

**Windows Users Only:**

**PuTTY:** [http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)

**Setup:** [https://www.msi.umn.edu/content/connecting-hpc-resources](https://www.msi.umn.edu/content/connecting-hpc-resources)

Unlike Mac/OSX windows does not have a native UNIX terminal. PuTTY is a terminal emulator for Windows and will allow you to interact with MSI systems using UNIX commands.

**Goals:**
- Log into MSI and connect to the Lab system.
- Learn some basic UNIX commands.
- Learn how to load software using the module system
- Learn how to run FastQC and Trimmomatic at the command line
- Learn how to write a bash script to automate running FastQC and Trimmomatic
- Learn how to convert the bash script to work with PBS to submit jobs to the queue.

**Log into MSI systems**

```bash
Last login: Wed Sep 17 16:35:05 on ttys001
ljmills-MacBookAir:~ ljmills$ ssh ljmills@login.msi.umn.edu
ljmills@login.msi.umn.edu's password: **************
```
Once you login you will see the welcome screen that contains some nice information.

As the @loginXX after your username in the terminal indicates you are now connected to the login node. While you can move around the filesystem and use UNIX commands while connected to the login node you won't be able to use any of the software installed on MSI systems. In order to gain access to the installed software you will need to connect to one of the HPC or interactive HPC systems. We are going to connect to the HPC system Mesabi and then start an interactive session using the PBS command qsub -I

```
ljmills@login03 [~] % ssh mesabi
Password:
ljmills@ln0003 [~] %qsub -I -l nodes=1:ppn=4,mem=8gb,walltime=4:00:00
qsub: waiting for job 469118.mesabim3.msi.umn.edu to start
qsub: job 469118.mesabim3.msi.umn.edu ready
```

qsub
- -I flag creates an interactive session
- -l nodes=1:ppn=4 requests a single nodes and 4 processors
• mem=8gb requests 8gb of memory
• walltime=4:00:00 requests use of the above resources for 4 hours
• You can change each of these requests up to the maximum allowed on the system as dictated by the different queues. See: https://www.msi.umn.edu/queues

Once your session has stared, let's see where we are in the file system. When you log into the system for the first time you are automatically taken to your home directory. Your home directory will always have the format of /home/yourGroup/yourMSIaccount.

```
ljmills@cn0547 [~] % pwd
/home/msistaff/ljmills
ljmills@cn0547 [~] %
```

Let's take a look at what is in your home directory, do you have any files there?

```
ljmills@cn0547 [~] % ls
```

What about in your group directory?

```
ljmills@cn0547 [~] % cd ..
ljmills@cn0547 [/home/msistaff] % ls
```

Let's go to a directory that will have some files in it.

```
ljmills@cn0547 [/home/msistaff/] % cd
/home/msistaff/public/qcIllumina/
ljmills@cn0547 [/home/msistaff/public/qcIllumina] % ls
Tutorial_file_R1.fastq  Tutorial_file_R2.fastq  fastqc_only.sh
tutorial_trim.sh  tutorial_trim.sh~  tutorial_trim2.sh
```

Let's get a different view of the directory contents.

```
ljmills@cn0547 [/home/msistaff/public/qcIllumina] % ls -lh
total 277M
-rwxrwxrwx 1 ljmills msistaff 123M Sep 18 2014 Tutorial_file_R1.fastq
-rwxrwxrwx 1 ljmills msistaff 123M Sep 26 2014 Tutorial_file_R2.fastq
-rwxrwxr-- 1 ljmills msistaff 355 Sep 18 2014 fastqc_only.sh
-rwxrwxr-- 1 ljmills msistaff 558 Jul 17 14:50 tutorial_trim.sh
-rwxrwxr-- 1 ljmills msistaff 683 Sep 18 2014 tutorial_trim2.sh
```
Move back to your home directory. There are three ways to do this, directly type in your home directory after `cd`, just use `cd` or use the `~` which represents your home directory.

Create a directory for the fastq files named tutorial, move into that directory, then copy the fastq files into this folder. Don’t remember the name of the files, `ls` to take a look again. Tab completion will also help you here. When typing the names of folder or files pressing Tab will complete the name for you.

You can also use a wildcard (*) to copy both fastq files at the same time.

```bash
ljmills@cn0547 [~] % cd ~
ljmills@cn0547 [~] % mkdir tutorial
ljmills@cn0547 [~] % cd tutorial
ljmills@cn0547 [~/tutorial] % cp /home/msistaff/public/qcIllumina/Tutorial_file_R1.fastq .
ljmills@cn0547 [~/tutorial] % cp /home/msistaff/public/qcIllumina/Tutorial_file_R2.fastq .
ljmills@cn0547 [~/tutorial] % cp /home/msistaff/public/qcIllumina/Tutorial_file_R*.fastq .
```

Using the fastq files in your tutorial directory lets run FastQC to take a look at the data. Many of the pieces of software installed on MSI systems have pages describing how to use the software on the MSI website ([https://www.msi.umn.edu/sw/fastqc](https://www.msi.umn.edu/sw/fastqc)). The module load command will have to be used for all software installed on MSI systems. Without this command you will not have access to the software and will get “Command not found” errors. You should also look at the webpages for the software you are trying to use, they tend to have good information and examples of how to use the software ([http://www.bioinformatics.babraham.ac.uk/projects/fastqc/](http://www.bioinformatics.babraham.ac.uk/projects/fastqc/)).

Let’s use FastQC on our data. First let’s load the fastqc module then we can run fastqc using the default settings. What do you get out?

```bash
ljmills@cn0547 [~/tutorial] % module load fastqc
ljmills@cn0547 [~/tutorial] % fastqc Tutorial_file_R1.fastq
Started analysis of Tutorial_file_R1.fastq
```

What happens if you replace `Tutorial_file_R1.fastq` with the R2 file or with the wildcard symbol (*)? You can remove the files you created using the `rm` command if you between each of these `fastqc` commands.

```bash
ljmills@cn0547 [~/tutorial] % fastqc Tutorial_file_R1.fastq
ljmills@cn0547 [~/tutorial] % fastqc Tutorial_file_R2.fastq
ljmills@cn0547 [~/tutorial] % fastqc Tutorial_file_R*.fastq
```
Lets use a piece of software that is a bit more complex. Trimmomatic is the software we will use to trim low quality sequences and adapter contamination from our paired end fastq data. The software page for trimmomatic: https://www.msi.umn.edu/sw/trimmomatic

```bash
ljmills@cn0547 [~/tutorial] % module load trimmomatic
ljmills@cn0547 [~/tutorial] % java -jar $TRIMMOMATIC/trimmomatic.jar PE -phred33 Tutorial_file_R1.fastq Tutorial_file_R2.fastq R1.PE.fastq R1.SE.fastq R2.PE.fastq R2.SE.fastq
ILLUMINACLIP:$TRIMMOMATIC/adapters/TruSeq2-PE.fa:2:30:10:2:true
LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:30
ljmills@cn0547 [~/tutorial] % ls
```

What files were returned? What do the contents of those files look like? Run fastqc on the resulting R1.PE.fastq and R2.PE.fastq files. Can you use the wildcard (*) to run fastqc on both files more quickly?

While running jobs straight from the command line is useful there are some disadvantages:

1) You have to type the commands perfectly.
2) You don’t have a record of what you did.
3) It is not easy to run lots of commands in a row or to run the same command again.
4) You have to wait around for the software to finish before you can do something else.
5) YOU DON’T HAVE A RECORD OF WHAT YOU DID!!

Don’t worry there is an easy (ish) way to over come all of these issues… Submitting jobs via and PBS script!

Copy tutorial_trim.sh from /home/msistaff/public/qcIllumina into your tutorial directory. Then open this file in Komodo Edit, File -> Open -> Remote File. You will need to select MSI from from the Server drop down menu the navigate to the tutorial directory.
tutorial_trim.sh

#!/bin/bash -l

#PBS -l nodes=1:ppn=4,mem=15GB,walltime=1:00:00
#PBS -m ae
#PBS -e trimmomatic.error
#PBS -o trimmomatic.out
#PBS -N tutorial_trim

module load trimmomatic
module load fastqc

cd /home/msistaff/ljmills/tutorial

fastqc Tutorial_file_R*.fastq

java -jar $TRIMMOMATIC/trimmomatic.jar PE -phred33 Tutorial_file_R1.fastq Tutorial_file_R2.fastq {R1,PE.fastq R1.SE.fastq R2,PE.fastq R2.SE.fastq


fastqc R*.PE.fastq

This PBS script is written in a scripting language called bash. The very first line is called the Sha-Bang and tells the system you are on how to interpret the following commands. Bash is a scripting language that has the ability to do lots and lots of things. My favorite Bash scripting guide can be found here: http://www.tldp.org/LDP/abs/html/

Lines that begin with #PBS are commands that will be interpreted by the PBS queuing program. Like isub the PBS commands reserve a specific amount of computational resources to be used to complete the items in your script.

The rest of the script are the same commands that you typed into the terminal to run FastQC and Trimmomatic. What elements of this script do you need to change so that it will work for you? Hint: cd to your directory.

Let's submit this PBS script to the Lab queue using the qsub command.

ljmills@cn0548 [~/tutorial] % qsub tutorial_trim.sh

475480.mesabim3.msi.umn.edu

The number that pops up when you submit a job is the jobid and is confirmation that your job was submitted to the queue.
You can check the status of your job using qstat, you will also get an email when your job finishes or if it is cancelled because of an error (aborts). Using the –u flag will let you see only your jobs.

You should have two jobs running, one will be the i
sub job that you started to connect to the Lab system, the other will be the job you just submitted. The S column is the status of your job: R is running, Q is queued and C is cancelled. When a job finishes it will go through the C state even if there wasn’t an error.

What is in your tutorial directory once the job finishes? What do the error and output files contain?
While tutorial_trim.sh is a nice record of what you did it is not very flexible and can only be used on these specific files. So let's look at another PBS bash script that is a bit more flexible. Copy fastqc_only.sh in to your tutorial directory the open it in Komodo Edit.

```
ljmills@cn0548 [~/tutorial] % cp /home/msistaff/public/qcIllumina/fastqc_only.sh ~/tutorial
```

Fastqc_only.sh

```
#!/bin/bash

#PBS -l
  nodes=1:ppn=4,mem=15GB,walltime=1:00:00
#PBS -m ae
#PBS -e trimmomatic.error
#PBS -o trimmomatic.out
#PBS -N trimmomatic

module load trimmomatic
module load fastqc

cd /home/msistaff/ljmills/tutorial

FASTQ="/home/msistaff/ljmills/tutorial"

for F in Tutorial_file_R1.fastq Tutorial_file_R2.fastq
do
  fastqc -o $FASTQ $F
done
```

While this script looks very simple it has a few new concepts in it.

Variables- both FASTQ and F are variables. When the variable is first set you only need to give the name of the variable (i.e. FASTQ) but when you then refer to (try to use) the variable you will need to add a $ (i.e. $FASTQ).

For Loop- let you run the same command over a list. The general structure of a for loop in bash is
for arg in [list]
do
  command(s)
done
The list can be as long as you need it to be. It can also be a UNIX command that lists the files for you so you don’t have to type them in yourself. Edit your script to match the for loop below.

```bash
#!/bin/bash

#PBS -l nodes=1:ppn=4,mem=15GB,walltime=1:00:00
#PBS -m ae
#PBS -e trimmomatic.error
#PBS -o trimmomatic.out
#PBS -N trimmomatic

module load trimmomatic
module load fastqc

cd /home/msistaff/ljmills/tutorial

FASTQ="/home/msistaff/ljmills/tutorial"

for F in `ls *.fastq`
do
  fastqc -o $FASTQ $F
done
```
Now because Trimmomatic requires two input files you have to get a bit fancier with the for loop. First lets copy the R1 and R2 files but give them new names. Copy tutorial_trim2.sh into your tutorial directory and open it in Komodo Edit.

```bash
#!/bin/bash
#PBS -l nodes=1:ppn=4,mem=15GB,walltime=1:00:00
#PBS -m ae
#PBS -e trimmomatic.error
#PBS -o trimmomatic.out
#PBS -N trimmomatic

module load trimmomatic
module load fastqc

cd /home/msistaff/ljmills/tutorial

FASTQ="/home/msistaff/ljmills/tutorial"
for F in "Tutorial_file_R1.fastq Tutorial_file_R2.fastq" "new_tutorial_file_R1.fastq new_tutorial_file_R2.fastq"
do
  set -- $F
  R1="$FASTQ/$1"
  R2="$FASTQ/$2"
  fastqc -f fastq $R1 $R2 -o $FASTQ
  java -jar $TRIMMOMATIC/trimmomatic.jar PE -phred33 $R1 $R2 ${1}.PE ${1}.SE ${2}.PE ${2}.SE ILLUMINACLIP:$TRIMMOMATIC/adapters/TruSeq2-PE.fa:2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:20 MINLEN:36
  fastqc ${1}.PE ${2}.PE
done

Notice that the pairs of fastq files are in “ “. Grouping the pairs means that both pairs are passed into $F at once, set -- $F then breaks the two files into separate variables $1 and $2. {} are added to variable names to indicate where the variable name ends. $1 = ${1} != $1.PE

These PBS scripts are nice building blocks for many other projects. Please copy them and reuse them to fit your specific needs!