Quality Control of Illumina Data at the Command Line

Quick UNIX Introduction:

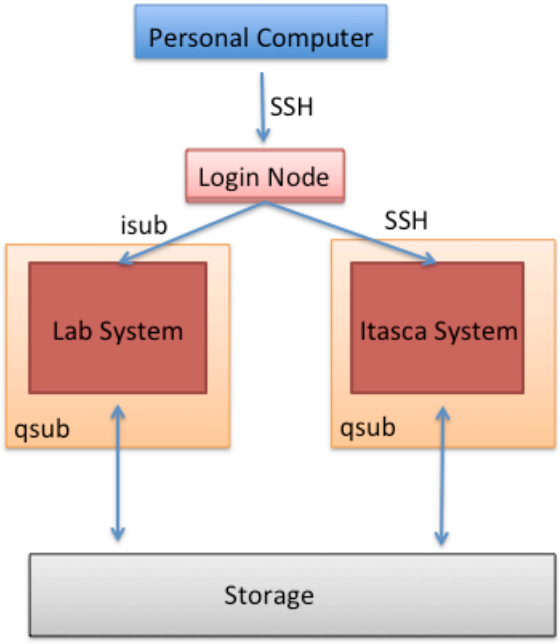
UNIX is an operating system like OSX or Windows. The interface between you and the UNIX OS is called “the shell”. There are a few flavors of shell but the MSI standard is bash. The shell is what you see, your environment, when you open PuTTY or Terminal.

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.
https://www.msi.umn.edu/labs/pbs

Itasca System: This is the newer, faster, fancier computer system. Your group will need to have SUs to use this system. SUs are free; Itasca is awesome; sign up for SUs!
https://www.msi.umn.edu/resources/job-queues

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:

Komodo Edit: http://www.activestate.com/komodo-edit/downloads
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/
Setup: https://www.msi.umn.edu/support/filezilla
This SFTP/FTP client will help you move files into and out of your MSI space.

Windows Users Only:
PuTTY: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
Setup: https://www.msi.umn.edu/support/putty

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

```
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.

```
Last login: Sun Sep 14 09:09:43 2014 from x-134-84-0-80.vpn.umn.edu
--------------------------------------------------------------------------
              University of Minnesota Supercomputing Institute
--------------------------------------------------------------------------
For assistance please contact us at
https://www.msi.umn.edu/support/help.html,
help@msi.umn.edu, or 612-626-0802.
--------------------------------------------------------------------------
This is a login host. Please avoid running resource-intensive tasks on
this machine. Instead, use the isub command to be logged into a node
appropriate for long-running interactive jobs. See 'isub --help' for
options
or https://www.msi.umn.edu/remote-access for more information.

Or, for non-interactive tasks, submit your job to one of MSI's HPC
resources (details at https://www.msi.umn.edu/hpc) or lab queues (details
at https://www.msi.umn.edu/labs/pbs).
--------------------------------------------------------------------------
Unused files in /tmp and /scratch.local will be
automatically deleted after 30 days.
--------------------------------------------------------------------------
------
ljmills@login03 [~] %
```

As the @loginXX after your username in the ternimal 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 beable 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 either the Lab or Itasca systems. We are going to connect to the lab system though a command called isub.

```
ljmills@login02 [~] % isub -m 8gb -w 4:00:00
qsub: waiting for job 217341.nokomis0015.msi.umn.edu to start
qsub: job 217341.nokomis0015.msi.umn.edu ready
```

When you use the isub command you are reserving computer space on the Lab system. We used two "flags" with this command –m and –w. These falgs allow us to specify how much memory and time we want to reserve and then use on the Lab system. There are many other "flags" that can be used with isub which can be displayed with the –h flag.

```
ljmills@labq59 [~] % isub -h
ljmills@labq59 [~] %
```

Lets 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@labq59 [~] % pwd
/home/msistaff/ljmills
ljmills@labq59 [~] %
```

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

```
ljmills@labq59 [~] % ls
```

What about in your group directory?

```
ljmills@labq59 [~] % cd ..
ljmills@labq59 [/home/msistaff] % ls
```

Lets go to a directory that will have some files in it.

```
ljmills@labq59 [~] % cd /home/msistaff/public/
ljmills@labq59 [/home/msistaff/public] % ls
garbe                          qcIllumina       ucsc.hg19.fasta.fai
hg19_canonical_PARmaskedonY.fa  ucsc.hg19.fasta
```

Lets get a different view of the directory contents.

```
ljmills@login03 [/home/msistaff/public] % ls -lh
total 6.7G
drwxr-sr-x. 4 jgarbe   msistaff 4.0K Jun 13 08:43 garbe
drwxr-s---. 2 ljmills  msistaff 4.0K Sep 17 17:19 qcIllumina
-rw-rw-r--. 1 jgarbe   msistaff 3.0G Dec 11  2013 ucsc.hg19.fasta
-rw-------. 1 chenzler msistaff 3.5K Dec 11  2013 ucsc.hg19.fasta.fai
```

The data we will need for the tutorial is in the qcIllumina directory. What do these fastq files look like?

```
ljmills@login03 [/home/msistaff/public] % cd qcIllumina/
ljmills@login03 [/home/msistaff/public/qcIllumina] % ls
Tutorial_file_R1.fastq  Tutorial_file_R2.fastq
ljmills@login03 [/home/msistaff/public/qcIllumina] % less
Tutorial_file_R1.fastq
```

Move back to your home directory. There are three ways to do this, directly type in your home directoy after cd, just use cd or use the ~ which represents your home direcotry.

```
ljmills@labq59 [ ] % cd /home/msistaff/ljmills/
ljmills@labq59 [~] % cd
ljmills@labq59 [~] % cd ~
```

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.

```
ljmills@labq59 [ ] % cd tutorial
ljmills@labq59 [~] % cp
/home/msistaff/ljmills/public/qcIllumina/Tutorial_file_R1.fastq
~/tutorial
ljmills@labq59 [~] % cp
/home/msistaff/ljmills/public/qcIllumina/Tutorial_file_R2.fastq
~/tutporial
```

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

```
jmills@labq59 [~/tutorial] % cd /home/msistaff/public/qcIllumina/
ljmills@labq59 [/home/msistaff/public/qcIllumina] % ls
Tutorial_file_R1.fastq  Tutorial_file_R2.fastq
ljmills@labq59 [/home/msistaff/public/qcIllumina] % cp
Tutorial_file_R*.fastq /home/msistaff/ljmills/tutorial/
```

Using the fastq files in your tutorial 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). 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 tyring to use, they tend to have good information and examples of how to use the software (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/).

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

```
ljmills@labq59 [~/tutorial] % module load fastqc
ljmills@labq59 [~/tutorial] % fastqc Tutorial_file_R1.fastq
ljmills@labq59 [~/tutorial] % ls
```

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.

```
ljmills@labq59 [~/tutorial] % fastqc Tutorial_file_R1.fastq
ljmills@labq59 [~/tutorial] % fastqc Tutorial_file_R2.fastq
ljmills@labq59 [~/tutorial] % fastqc Tutorial_file_R*.fastq
```

Lets use a piece of software that is complex. Trimmomatic is the software we will used 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

```
ljmills@labq59 [~/tutorial] % module load trimmomatic
ljmills@labq59 [~/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@labq59 [~/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 fastq 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 $TRIMMOMATIC/trimmomatic.jar PE -phred33 Tutorial_file_R1.fastq
Tutorial_file_R2.fastq R1.PE.fastq R1.SE.fastq R2.PE.fastq R2.SE.fastq
ILLUMINACLIP:$TRIMMOMATIC/adapters/TruSeq2-PE.fa:2:30:10:2:true LEADING:3
TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:30
fastqc R*.PE.fastq
```

This PBS script is writen in a scripting language called bash. The very first line is called the Sha-Bang and tells the system you are on how to interperate the folllowing commands. Bash is a scripting language that has the ability do 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 interperted 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.

Lets submit this PBS script to the Lab queue using the qsub command.

```
ljmills@labq55 [~/tutorial] % qsub tutorial_trim.sh
217511.nokomis0015.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.

```
ljmills@labq55 [~/tutorial] % qstat -u ljmills

nokomis0015.msi.umn.edu:

Req'd    Req'd       Elap
Job ID                  Username    Queue    Jobname          SessID  NDS   TSK   Memory
Time    S   Time
----------------------- ----------- -------- ---------------- ------ ----- ------ ------
217495.nokomis0015.msi  ljmills     lab      STDIN             27890     1      1  8000m
04:00:00 R  00:15:48
217511.nokomis0015.msi  ljmills     lab      tutorial_trim     30671     1      4   15gb
01:00:00 R  00:00:08

nokomis0015.msi.umn.edu:
                                                                                Req'd
Req'd       Elap
Job ID                  Username    Queue    Jobname          SessID  NDS   TSK   Memory
Time    S   Time
----------------------- ----------- -------- ---------------- ------ ----- ------ ------
217495.nokomis0015.msi  ljmills     lab      STDIN             27890     1      1  8000m
04:00:00 R  00:15:48
217511.nokomis0015.msi  ljmills     lab      tutorial_trim     30671     1      4   15gb
01:00:00 R  00:00:08
ljmills@labq55 [~/tutorial] %
```

You should have two jobs running, one will be the isub 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 though the C state even if there wasn't an error.

What is in your tutorial directory onces the job finishes? What do the error and output files contain?

```
ljmills@labq55 [~/tutorial] % ls
R1.PE.fastq         R2.PE.fastq         Tutorial_file_R1.fastq
Tutorial_file_R2_fastqc.html  tutorial_trim.sh
R1.PE_fastqc.html  R2.PE_fastqc.html  Tutorial_file_R1_fastqc.html
Tutorial_file_R2_fastqc.zip
R1.PE_fastqc.zip   R2.PE_fastqc.zip   Tutorial_file_R1_fastqc.zip
trimmomatic.error
R1.SE.fastq         R2.SE.fastq         Tutorial_file_R2.fastq
trimmomatic.out
```

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 lets look at another PBS bash script that is a bit more flexiable. Copy fastqc_only.sh in to your tutorial dirctory the open it in Komodo Edit.

```
ljmills@labq55 [~/tutorial] % cp
/home/msistaff/public/qcIllumina/fastqc_only.sh ~/tutorial
```

Fastqc_only.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 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.

```
#!/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 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.

```
#!/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 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 varialble 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!