Day 02 Workshop Instruction Manual

Disclaimer: The primary goal of this workshop is to explore how to use a high-performance computing (HPC) environment. We will use various bioinformatics tools as examples to understand the cluster environment. Please be sure to consult the application manual for each tool, and choose the options and flags that are most appropriate for your own research question.

Table of Contents

I. Recap of Day 1

- We logged into the cluster using:

```
ssh user01@omics.c3.ca
(user01@omics.c3.ca) Password:
Last login: Wed Oct 22 18:02:58 2025 from 137.82.20.161
```

- We navigated to the working directory on the cluster:

```
cd /scratch/user01/omics_workshop/
```

- We successfully established our working space, created directories, installed or ensured required tools, and fetched raw sequencing data.

- Today we will build on that: we will explore how to move into the **pre-processed data stage**, manage files, begin working with tools for **quality control and trimming**, and run our first mini-pipeline.

- We are going to work with the data downloaded in Exercise 2 on Day 01.

II. Data Transfer – Metadata

Before we begin processing reads, we need to transfer metadata from our personal device to the cluster.
We should have a file called SraRunTable.csv on our device from yesterday.

Windows PowerShell

```
PS C:\Users\tnandi\OneDrive - UBC\Desktop> ls .\SraRunTable.csv
    Directory: C:\Users\tnandi\OneDrive - UBC\Desktop

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---l         10/21/2025      3:09 PM      11082 SraRunTable.csv
```

```
PS C:\Users\tnandi\OneDrive - UBC\Desktop> scp SraRunTable.csv
user01@omics.c3.ca:/home/user01/scratch/omics_workshop/input_da
ta/metadata/metadata_02/
(user01@omics.c3.ca) Password:
SraRunTable.csv              100%   11KB  349.1KB/s   00:00
```

macOS / Linux

```
$ rsync -avx SraRunTable.csv
user01@omics.c3.ca:/home/user01/scratch/omics_workshop/input_da
ta/metadata/metadata_02/
```

Verify the transfer: the file on the cluster

```
[user01@login1 ~]$ ls -l \
/home/user01/scratch/omics_workshop/input_data/metadata/metadata_02
total 12
-rw-r-----. 1 user01 user01 11082 Oct 23 02:02 SraRunTable.csv
```

III. Allocating Compute Resources

Before starting any heavy computation, we need to reserve compute resources (memory, CPUs, time) on the cluster. This helps ensure our job runs smoothly and we do not overload shared resources.

We will use the `salloc` command:

```
[user01@login1~]$ salloc --mem=10G --time=1:00:00 --nodes=1 \
 --cpus-per-task=1
salloc: Granted job allocation 8
salloc: Waiting for resource configuration
salloc: Nodes node1 are ready for job
[user01@node1 ~]$
```

Note: Tomorrow we will cover the job scheduler in more depth (e.g., sbatch, srun).

IV. Navigating to Raw Data Directory

Now we will move into the directory where raw reads are stored:

```
$ cd \ /home/user01/scratch/omics_workshop/input_data/raw_data/raw_data_02
[user01@ node1 raw_data_02]$ ls -l
total 19636520
drwxr-x---. 2 user01 user01          29 Oct 22 18:47 SRR25110243
-rw-r-----. 1 user01 user01 10053897622 Oct 22 19:22 SRR25110243_1.fastq
-rw-r-----. 1 user01 user01 10053897622 Oct 22 19:22 SRR25110243_2.fastq
```

V. Understanding the FASTQ Format

We are working with files ending in '`.fastq`', which contain sequenced reads.
Example: View the first read from '`SRR25110243_1.fastq`'

```
$ head -n 4 SRR25110243_1.fastq
```

FASTA FORMAT

```
@SRR25110243.1 A00742:558:HMMFLDSX5:1:1101:25174:1000
length=150

NACGCCGCTGTTGCCCGCCCAGCTGTCACCGTCGCTCGCTACGCGGCTCCCGCCGTCTCCTA
CGCTGCTCCCGCCGTGACCGCTGTCCACCCCGCTCCGGCTGTTTCCTACGCCGCTCCCGCGG
TCTCCTATGCCGCTGTGGCTCGCCCC

+SRR25110243.1 A00742:558:HMMFLDSX5:1:1101:25174:1000
length=150

#FF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FF:F,F:,FFF,FF::F:FF,F
F:F,F,,:FF:FF,:F,FFF:,F,FFFFF,F,F,FF,FF,FF:FF:,F:,FFFF:FFF:F,F
F:F,F,F,:FF:F::FFFF:FFFFF,
```

```
    begins with @ and a read identifier.
    the nucleotide sequence of the read.
    begins with +, sometimes repeats the identifier.
    contains quality score characters — one per base in the
    sequence.
```

On line 4 we will see something like:

```
#FF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FF:F,F:,FFF,FF::F:FF,FF:F,F,,:FF:FF
,:F,FFF:,F,FFFFF,F,F,FF,FF,FF:FF:,F:,FFFF:FFF:F,FF:F,F,F,:FF:F::FFFF:FFFFF,
```

Each of those symbols (like #, F, : and so on) actually encodes a number, specifically a PHRED quality score. The PHRED score relates to how confident the sequencer is in the base call. In plain language: higher score → higher confidence.

VI. Sub-sampling Reads using SeqKit

We will now subsample the reads to a manageable size for this workshop. We will use the tool 'SeqKit'.

```
$ module load seqkit
$ cd /home/user01/scratch/omics_workshop/preprocessed_data
$ mkdir D2_set01
$ cd D2_set01

$ seqkit sample -p 0.10 -s 2025 \
~/scratch/omics_workshop/input_data/raw_data/raw_data_02/SRR25110243_1.fastq \
-o subsampled_R1.fastq.gz &

$ seqkit sample -p 0.10 -s 2025 \
~/scratch/omics_workshop/input_data/raw_data/raw_data_02/SRR25110243_2.fastq \
-o subsampled_R2.fastq.gz &
```

Note: `-p 0.10` means ~10% of reads;
`-s 2025` sets the random seed for reproducibility.

Check file sizes:

```
$ ls -l \
~/scratch/omics_workshop/input_data/raw_data/raw_data_02/SRR25110243_1.fastq

-rw-r-----. 1 user01 user01 10053897622 Oct 22 19:22
/home/user01/scratch/omics_workshop/input_data/raw_data/raw_dat
a_02/SRR25110243_1.fastq
```

```
$ pwd
/home/user01/scratch/omics_workshop/preprocessed_data/D2_set01

$ ls -l subsampled_R1.fastq.gz
-rw-r-----. 1 user01 user01 180942100 Oct 24 00:30
subsampled_R1.fastq.gz
```

Here is what is going on:

Original file size

The original SRR25110243_1.fastq file was ~**10,053,897,622** bytes (~10.05 GB). That is because it includes all the sequenced reads for that run, with full depth.

Subsampling (~10%)

We will use a tool (e.g., SeqKit) to extract about 10% of the reads (-p 0.10). That means instead of 100% of the reads, we now have ~10% of them.

Compression (gzip .gz)

Next, by specifying the output filename as subsampled_R1.fastq.gz, the tool automatically compressed the file into gzip format. The gzipped file is ~180,942,100 bytes (~0.18 GB).

Evaluate sequences count:

```
$ seqkit stat subsampled_R1.fastq.gz
$ seqkit stat subsampled_R2.fastq.gz
```

We should see stats like:

```
file                        format  type    num_seqs       sum_len  min_len  avg_len
max_len
subsampled_R1.fastq.gz      FASTQ   DNA    2,255,622   338,343,300      150      150
150
```

VII. Quality Assessment with FastQC

We will assess read quality using FastQC.

First, module load:

```
$ module spider fastqc
$ module spider fastqc/0.11.9
$ module load StdEnv/2020
$ module load fastqc/0.11.9

$ fastqc -h
```

Run FastQC:

```
$ fastqc --threads 1 subsampled_R1.fastq.gz
```

```
Started analysis of subsampled_R1.fastq …
Approx 5% complete for subsampled_R1.fastq
Approx 10% complete for subsampled_R1.fastq
Approx 15% complete for subsampled_R1.fastq

.
.
.
.



Approx 90% complete for subsampled_R1.fastq
Approx 95% complete for subsampled_R1.fastq
Analysis complete
```

VIII. Inspecting the FASTQC Output

After running quality control on our reads, we should have generated files with names like:

```
subsampled_R1_fastqc.zip
subsampled_R2_fastqc.zip
```

These are ZIP files from FastQC containing the QC report for each of our paired-end read files. Let's unpack them and check out the summary.

Unzip the report

```
$ unzip subsampled_R1_fastqc.zip
```

This will create a directory (e.g., `subsampled_R1_fastqc`) containing files like `summary.txt`, `fastqc_data.txt`, and an `Images/` directory with plots.

Do the same for the R2 file:

```
$ unzip subsampled_R2_fastqc.zip
```

Open the summary file

```
$ cd subsampled_R1_fastqc
$ less summary.txt
```

The `summary.txt` gives we a quick overview of how each QC module performed (Pass / Warn / Fail).

For example, we will see lines like:

```
PASS    Basic Statistics        subsampled_R1.fastq
PASS    Per base sequence quality       subsampled_R1.fastq
PASS    Per tile sequence quality       subsampled_R1.fastq
PASS    Per sequence quality scores     subsampled_R1.fastq
WARN    Per base sequence content       subsampled_R1.fastq
PASS    Per sequence GC content subsampled_R1.fastq
PASS    Per base N content      subsampled_R1.fastq
PASS    Sequence Length Distribution    subsampled_R1.fastq
WARN    Sequence Duplication Levels     subsampled_R1.fastq
PASS    Overrepresented sequences       subsampled_R1.fastq
PASS    Adapter Content subsampled_R1.fastq
```

This helps we quickly spot any modules flagged for caution.

Interpret the results

If a module says **PASS**, that's good — no obvious problems flagged in that area.
If we see **WARN** or **FAIL**, don't panic — it just means *we should take a closer look*.

Key modules to check:
Per base sequence quality – Are most bases high quality (Q30 or above)?
Per base sequence content – Do base proportions look uniform (for DNA) or reflect expected bias (for e.g., RNA-seq)?
Adapter content / Overrepresented sequences – Are there lots of reads showing adapter sequences or repeats?

IX. Preprocessing step to filter the low-quality reads?

So far, we have taken a look at read quality using FastQC. Now, we will use a bioinformatics application called **Trimmomatic** to filter out poor-quality reads and trim low-quality bases from our samples.

Trimmomatic has a variety of options for trimming. Here, we will use a sliding window of size 3, removing bases when their **Phred** score drops below 20. We will also discard any reads shorter than 50 bases after trimming.

```
[user01@login1 D2_set01]$ salloc --mem=10G --time=1:00:00
[user01@node1 D2_set01]$ module load trimmomatic
[user01@node1 D2_set01]$ pwd
/home/user01/scratch/omics_workshop/preprocessed_data/D2_set01
[user01@node1 D2_set01]$ java -jar \
$EBROOTTRIMMOMATIC/trimmomatic-0.39.jar PE \
subsampled_R1.fastq.gz subsampled_R2.fastq.gz \
subsampled_R1.trim.fastq.gz R1_un.fastq.gz \
subsampled_R2.trim.fastq.gz R2_un.fastq.gz \
SLIDINGWINDOW:3:20 MINLEN:50


real    3m51.861s
user    3m35.057s
sys     0m3.812s
```

X. Improve Job Efficiency

Let's try using multiple CPUs to speed things up on the cluster. Request 4 CPUs and 10 GB memory for 1 hour:

```
[user01@login1 D2_set01]$ salloc --mem=10G --time=1:00:00 \
--cpus-per-task=2
```

Then re-run Trimmomatic using 2 threads:

```
[user01@node1 D2_set01]$ module load trimmomatic
[user01@node1 D2_set01]$ time java -jar \
$EBROOTTRIMMOMATIC/trimmomatic-0.39.jar PE -threads 2 \
subsampled_R1.fastq.gz subsampled_R2.fastq.gz \
subsampled_R1.trim.fastq.gz R1_un.fastq.gz \
subsampled_R2.trim.fastq.gz R2_un.fastq.gz \
SLIDINGWINDOW:3:20 MINLEN:50
```

We should notice a reduction in runtime compared to running with a single thread.

```
Example output:
Quality encoding detected as phred33

Input Read Pairs: 2255622
Both Surviving: 2001479 (88.73%)
Forward Only Surviving: 86692 (3.84%)
Reverse Only Surviving: 99020 (4.39%)
Dropped: 68431 (3.03%)
TrimmomaticPE: Completed successfully

real    1m28.042s
user    3m0.136s
sys     0m3.240s
```

XI. Indexing the Reference Genome

Next, we will prepare our reference genome for alignment.

```
$ cd /home/user01/scratch/omics_workshop/input_data
$ mkdir refgenome
$ cd refgenome

$ wget \
https://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/052/857/255/GCA_052857255.1_AS
M5285725v1/GCA_052857255.1_ASM5285725v1_genomic.fna.gz
$ module load bwa

$ bwa index GCA_052857255.1_ASM5285725v1_genomic.fna.gz
```

Indexing makes searching for alignments faster. This step only needs to be done once.
If indexing takes too long, you can use the **pre-indexed genome** provided here:

```
~/projects/def-sponsor00/Day02/refgenome
```

XII. Aligning Reads to the Reference Genome

Now that we have our preprocessed reads and indexed reference genome, it's time to **align the reads** — in other words, find out *where in the genome* each short read belongs.

We will navigate to the output directory

```
$ cd ~/scratch/omics_workshop/output_data
```

The alignment step can be resource-intensive, so we will request a node with enough memory and CPUs for this task.

```
[user01@login1 output_data]$ salloc --mem=10G \
--cpus-per-task=2 --time=2:00:00
```

Once the job starts, we will be placed into a compute node (for example, `node2`).

We will use the **BWA-MEM** algorithm — a fast and accurate aligner for Illumina paired-end reads.

Here is the command:

```
[user01@node2 output_data]$ module load bwa
[user01@node2 output_data]$ bwa mem -t 2 \
~/projects/def-sponsor00/Day02/refgenome/GCA_052857255.1_ASM5285725v1_genomic.fna.gz \
~/scratch/omics_workshop/preprocessed_data/D2_set01/subsampled_R1.fastq.gz \
~/scratch/omics_workshop/preprocessed_data/D2_set01/subsampled_R2.fastq.gz \
> subsampled_out.sam &
```

Let us break this down:

bwa mem runs the BWA-MEM algorithm.
-t 2 — uses 2 CPU threads to speed things up.
The first argument is the reference genome.
The next two are the paired-end FASTQ files (R1 and R2).
> redirects the output to a SAM file (subsampled_out.sam).
The & at the end runs the job in the background.

As BWA runs, we will see messages like:

```
[M::bwa_idx_load_from_disk] read 0 ALT contigs
[M::process] read 133334 sequences (20000100 bp)...
[M::process] read 133334 sequences (20000100 bp)...
[M::mem_pestat] # candidate unique pairs for (FF, FR, RF, RR): (48, 53885,
14, 37)
[M::mem_pestat] analyzing insert size distribution for orientation FF...
[M::mem_pestat] (25, 50, 75) percentile: (90, 192, 387)
[M::mem_pestat] low and high boundaries for computing mean and std.dev: (1,
981)
[M::mem_pestat] mean and std.dev: (179.55, 150.07)
[M::mem_pestat] low and high boundaries for proper pairs: (1, 1278)
[M::mem_pestat] analyzing insert size distribution for orientation FR...
[M::mem_pestat] (25, 50, 75) percentile: (221, 272, 344)...
...
...
[main] Version: 0.7.17-r1188
[main] CMD: bwa mem -t 2
/home/user01/scratch/omics_workshop/input_data/refgenome/GCA_052857255.1_AS
M5285725v1_genomic.fna.gz
/home/user01/scratch/omics_workshop/preprocessed_data/D2_set01/subsampled_R
1.fastq.gz
/home/user01/scratch/omics_workshop/preprocessed_data/D2_set01/subsampled_R
2.fastq.gz
[main] Real time: 809.417 sec; CPU: 1491.659 sec
```

These lines report:

The number of reads processed
The pairing orientations (FR is usually the main one for Illumina data)
Total runtime and CPU time

These numbers vary, they depend on our dataset.

```
real    13m29.444s
user    24m33.581s
sys     0m18.086s
```

Once completed, we should see a file called: `subsampled_out.sam`. This SAM (Sequence Alignment Map) file contains all the alignment information.

We can explore all available bwa options with

```
[user01@node2 output_data]$ bwa -help
```

XIII. Understanding the Alignment Section (SAM / BAM File Format)

Once the alignment process completes, BWA produces an output file in SAM format (Sequence Alignment/Map).
The SAM file has two parts:

1. Header section: starts with `@` and contains information about the reference genome and alignment run.
2. Alignment section: each subsequent line corresponds to one read and its alignment details.

We can view the review the '.sam' format as follows:

```
[user01@node2 output_data]$ less subsampled_out.sam
```

It should start with header lines like:

```
@SQ     SN:CM128357.1   LN:560970318
@SQ     SN:CM128358.1   LN:235277662
@SQ     SN:CM128359.1   LN:227218120
.
.
.
@PG     ID:bwa  PN:bwa  VN:0.7.17-r1188 CL:bwa mem -t 2
/home/user01/scratch/omics_workshop/input_data/refgenome/GCA_05
2857255.1_ASM5285725v1_genomic.fna.gz
/home/user01/scratch/omics_workshop/preprocessed_data/D2_set01/
subsampled_R1.fastq.gz
/home/user01/scratch/omics_workshop/preprocessed_data/D2_set01/
subsampled_R2.fastq.gz
```

These describe the reference sequences and the BWA command used, perfect for verifying our alignment setup.

Following the header is the alignment section.
Each line represents one read and contains **11** mandatory fields plus optional fields that may vary depending on the aligner and parameters used.

Let us look at an example entry:

```
SRR25110243.2   99      CM128364.1      36903209        60
41S109M =       36903227        168
GGGAGACTGCTAATGCTCCAGACCGCCTGGCCGACCCTCCGAGAGCCAGTGCTGCGCCCATTC
CTTCTCTGGTCCATTGCGGTGGGCGAGGCATCGGATGCCTGCGACAGTGCCGCTGCCGTCGGA
GTTGAGATCTGAAGTGGCACTGTG
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FF
FFFFFFFFFFFFF:FFFFFFFFFF  NM:i:1  MD:Z:36C72      MC:Z:150M
MQ:i:60 AS:i:104        XS:i:0
SA:Z:CM128364.1,36895837,+,43M107S,60,0;
```

Here is how the fields map:

FIELD	VALUE IN LINE	DESCRIPTION
QNAME	`SRR25110243.2`	This is the read identifier.
FLAG	`99`	The bitwise flag describing this read's pairing, strand, etc.
RNAME	`CM128364.1`	The name of the reference sequence it aligned to.
POS	`36903209`	The 1-based leftmost alignment position on the reference.
MAPQ	`60`	The mapping quality score.
CIGAR	`41S109M`	Indicates 41 bases are soft-clipped, then 109 matched.
RNEXT	`=`	The next (mate) read is on the same reference (so "=").
PNEXT	`36903227`	Position of the mate read.
TLEN	`168`	Observed template length (insert size) for this read pair.
SEQ	`GGGAGACTGCT`… (long sequence)	The actual read sequence aligned.
QUAL	`FFFFFFFFF`… (long string of F's and other characters)	ASCII-encoded base quality string matching SEQ length.
OPTIONAL TAGS	`NM:i:1 MD:Z:36C72 MC:Z:150M MQ:i:60 AS:i:104 XS:i:0 SA:Z:CM128364.1,36895837,+, 43M107S,60,0;`	These provide additional aligner-specific info (e.g., number of mismatches, alternative alignments).

The SAM file is a text file describing how each read aligns to the reference genome.
Because SAM files can be very large, we usually convert them to the compressed BAM format.

The compressed binary version of SAM is called a BAM file.

We will convert the SAM file to BAM format with the view command and tell this command that the input is in SAM format (-S) and to output BAM format (-b):

```
$ module load samtools
$ samtools view -S -b subsampled_out.sam > subsampled_out.bam
```

This reduces file size and makes it easier to sort, index, and query the alignments later.

After we have aligned reads and generated a BAM file, we can run flagstat to learn more about this bam file.

```
$ samtools flagstat subsampled_out.bam
5367653 + 0 in total (QC-passed reads + QC-failed reads)
4511244 + 0 primary
0 + 0 secondary
856409 + 0 supplementary
0 + 0 duplicates
0 + 0 primary duplicates
5219208 + 0 mapped (97.23% : N/A)
4362799 + 0 primary mapped (96.71% : N/A)
4511244 + 0 paired in sequencing
2255622 + 0 read1
2255622 + 0 read2
3528008 + 0 properly paired (78.20% : N/A)
4350128 + 0 with itself and mate mapped
12671 + 0 singletons (0.28% : N/A)
67970 + 0 with mate mapped to a different chr
53923 + 0 with mate mapped to a different chr (mapQ>=5)
```

References

1. Data Carpentry: https://datacarpentry.org/
2. European Nucleotide Archive: https://www.ebi.ac.uk/ena/browser/home
3. National Center for Biotechnology Information: https://www.ncbi.nlm.nih.gov/
4. Digital Research Alliance of Canada: https://www.alliancecan.ca/en
5. O'Leary NA, Cox E, Holmes JB, Anderson WR, Falk R, Hem V, Tsuchiya MTN, Schuler GD, Zhang X, Torcivia J, Ketter A, Breen L, Cothran J, Bajwa H, Tinne J, Meric PA, Hlavina W, Schneider VA. Exploring and retrieving sequence and metadata for species across the tree of life with NCBI Datasets. Sci Data. 2024 Jul 5;11(1):732. doi: 10.1038/s41597-024-03571-y. PMID: 38969627; PMCID: PMC11226681.
6. Lyu B, Li J, Niemeyer B, Stanley D, Song Q. Identification and characterization of ecdysis-related neuropeptides in the lone star tick *Amblyomma americanum*. Front Endocrinol (Lausanne). 2023 Aug 25;14:1256618. doi: 10.3389/fendo.2023.1256618. PMID: 37693356; PMCID: PMC10490126.

Acknowledgments

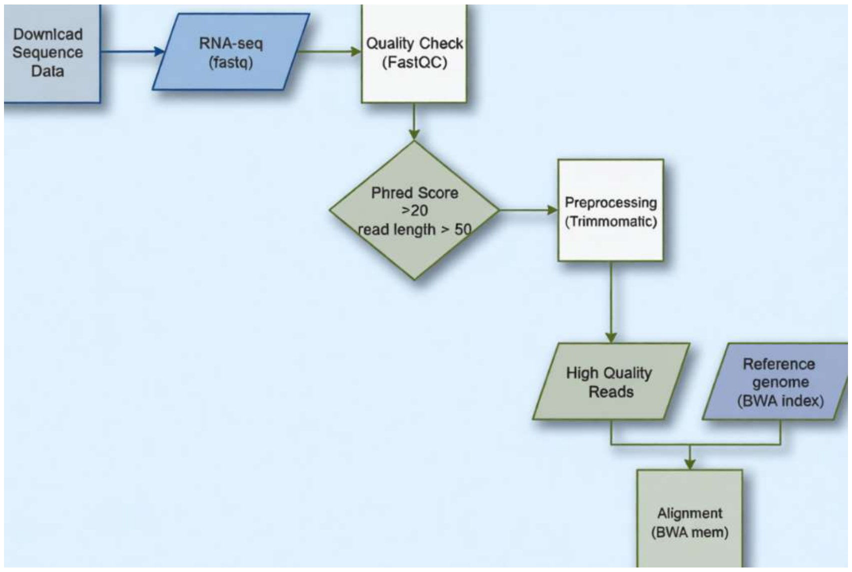
I would like to thank the **Digital Research Alliance of Canada** for providing the infrastructure support that made this workshop possible. I am sincerely grateful to **Jayson To** for his constant support and encouragement, and for ensuring that all arrangements were in place for the smooth facilitation of the workshop. I wish to extend my deep appreciation to **Michael Tang** for his technical expertise and assistance in setting up the workshop cluster, and to the entire **Advanced Research Computing (ARC) team at UBC** for their multifaceted support throughout the process.

End of Day 2 — Take-Home Message

We covered a lot of ground, and we have taken the important steps toward mastering the hpc environment and how to improve the job efficiency.

```
Workflow Overview

Raw sequence reads → (QC with FastQC) → Filter/Trim low-quality
reads (with Trimmomatic) → (Align to reference genome with
BWA-MEM) → SAM/BAM file (view alignment using SAMtools)
Index the reference genome for alignment.
```



What's coming tomorrow:
We will build scripts for job submission based on the commands we learned today!

Look forward to seeing you all tomorrow!