Summer 2020

Unix/Linux for Informatic Analysis
https://github.com/sabrsyed/InformaticsTools_2020

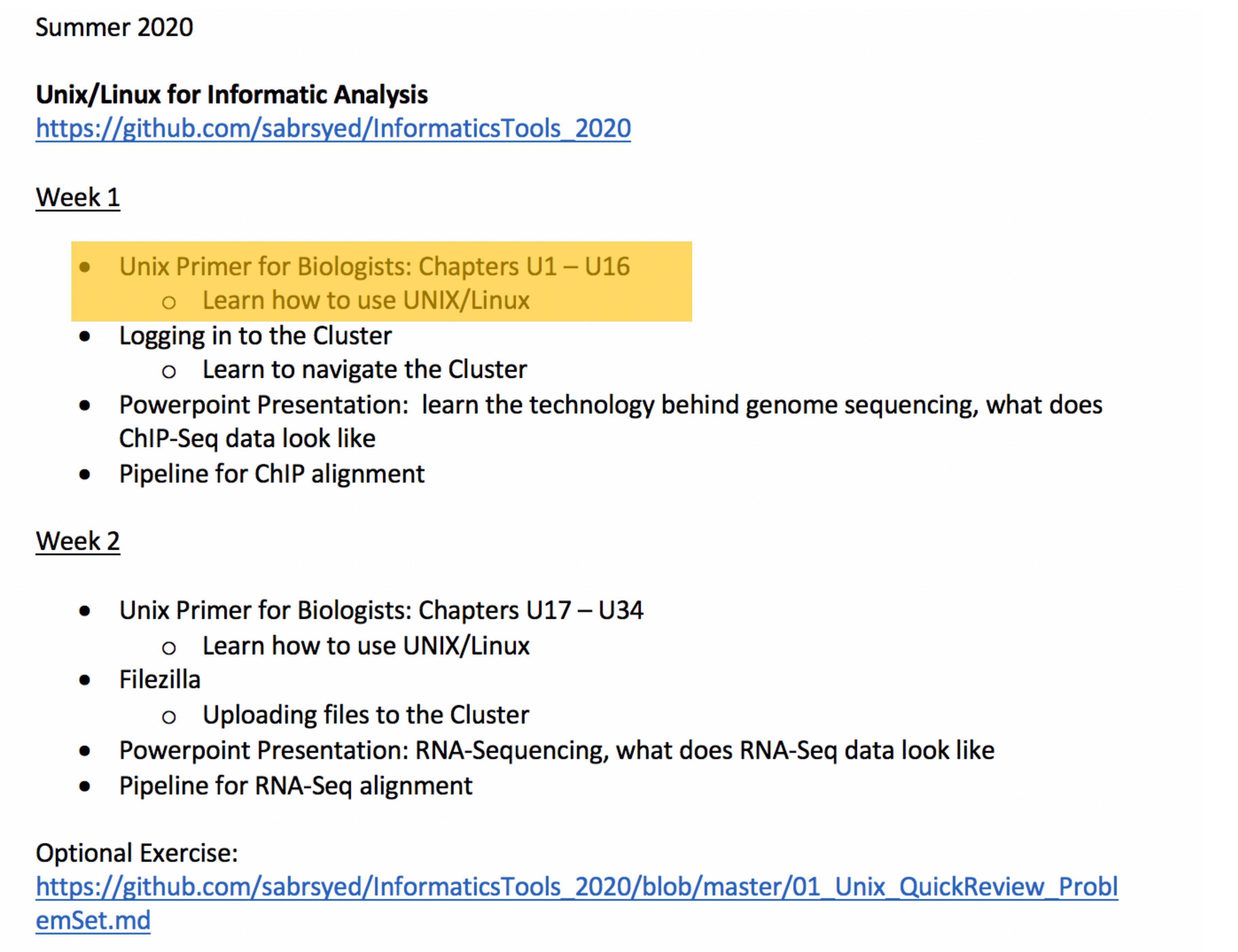
Week 1

- Unix Primer for Biologists: Chapters U1 – U16
    - Learn how to use UNIX/Linux
- Logging in to the Cluster
    - Learn to navigate the Cluster
- Powerpoint Presentation: learn the technology behind genome sequencing, what does ChIP-Seq data look like
- Pipeline for ChIP alignment

Week 2

- Unix Primer for Biologists: Chapters U17 – U34
    - Learn how to use UNIX/Linux
- Filezilla
    - Uploading files to the Cluster
- Powerpoint Presentation: RNA-Sequencing, what does RNA-Seq data look like
- Pipeline for RNA-Seq alignment

Optional Exercise:
https://github.com/sabrsyed/InformaticsTools_2020/blob/master/01_Unix_QuickReview_ProblemSet.md



Summer 2020

Unix/Linux for Informatic Analysis
https://github.com/sabrsyed/InformaticsTools_2020

Week 1

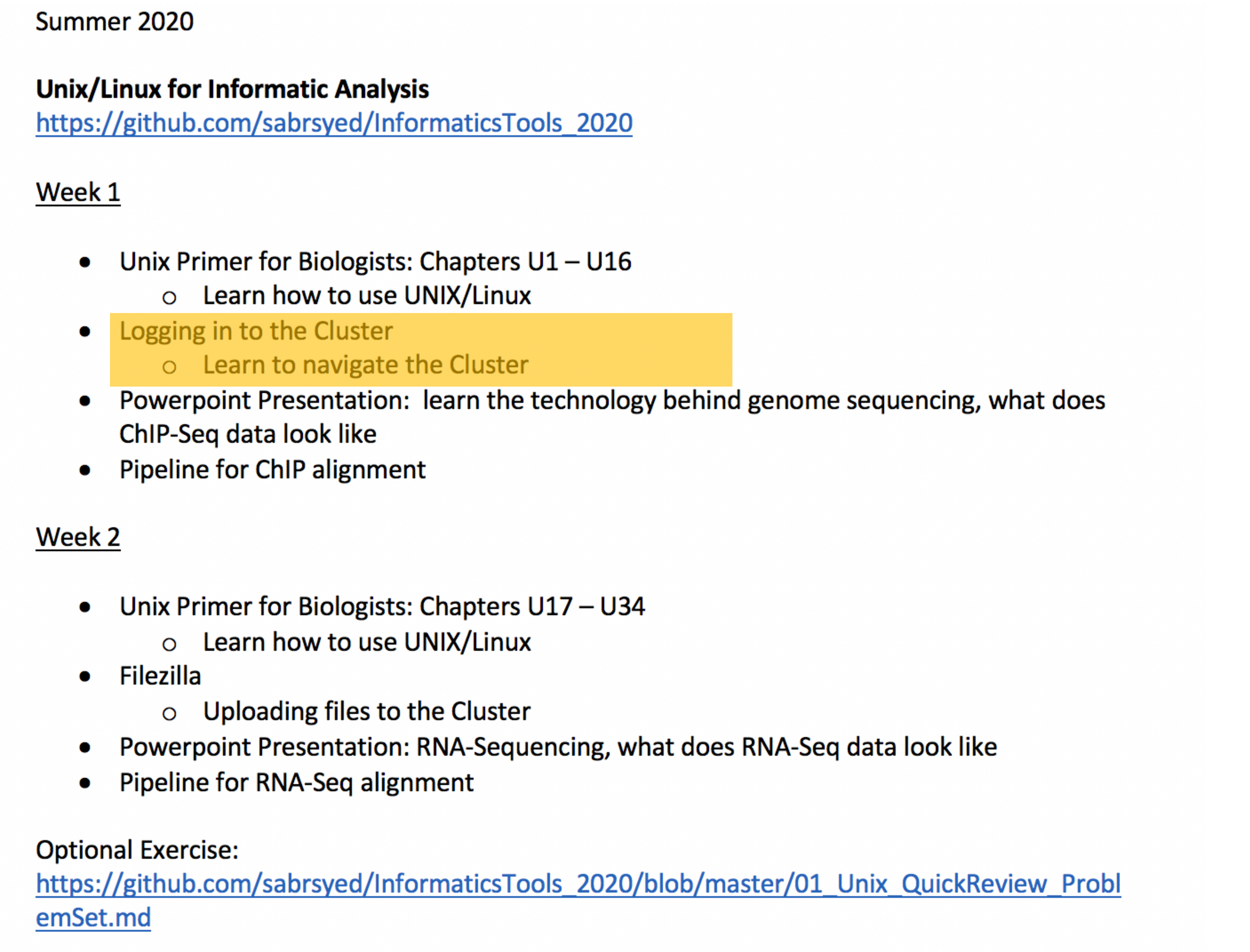
- Unix Primer for Biologists: Chapters U1 – U16
  - Learn how to use UNIX/Linux
- Logging in to the Cluster
  - Learn to navigate the Cluster
- Powerpoint Presentation: learn the technology behind genome sequencing, what does ChIP-Seq data look like
- Pipeline for ChIP alignment

Week 2

- Unix Primer for Biologists: Chapters U17 – U34
  - Learn how to use UNIX/Linux
- Filezilla
  - Uploading files to the Cluster
- Powerpoint Presentation: RNA-Sequencing, what does RNA-Seq data look like
- Pipeline for RNA-Seq alignment

Optional Exercise:
https://github.com/sabrsyed/InformaticsTools_2020/blob/master/01_Unix_QuickReview_ProblemSet.md

Unix Primer for Biologists, Chapters U1-U16



Summer 2020

Unix/Linux for Informatic Analysis
https://github.com/sabrsyed/InformaticsTools_2020

Week 1

- Unix Primer for Biologists: Chapters U1 – U16
  - Learn how to use UNIX/Linux
- Logging in to the Cluster
  - Learn to navigate the Cluster
- Powerpoint Presentation: learn the technology behind genome sequencing, what does ChIP-Seq data look like
- Pipeline for ChIP alignment

Week 2

- Unix Primer for Biologists: Chapters U17 – U34
  - Learn how to use UNIX/Linux
- Filezilla
  - Uploading files to the Cluster
- Powerpoint Presentation: RNA-Sequencing, what does RNA-Seq data look like
- Pipeline for RNA-Seq alignment

Optional Exercise:
https://github.com/sabrsyed/InformaticsTools_2020/blob/master/01_Unix_QuickReview_ProblemSet.md

Why Cluster?

Massive data coming from Deep Sequencing needs to be

- stored
- (parallel) processed

It is not feasible to process this kind of data even using a high-end computer.

MGHPCC

University of Massachusetts Green High Performance Computing Cluster

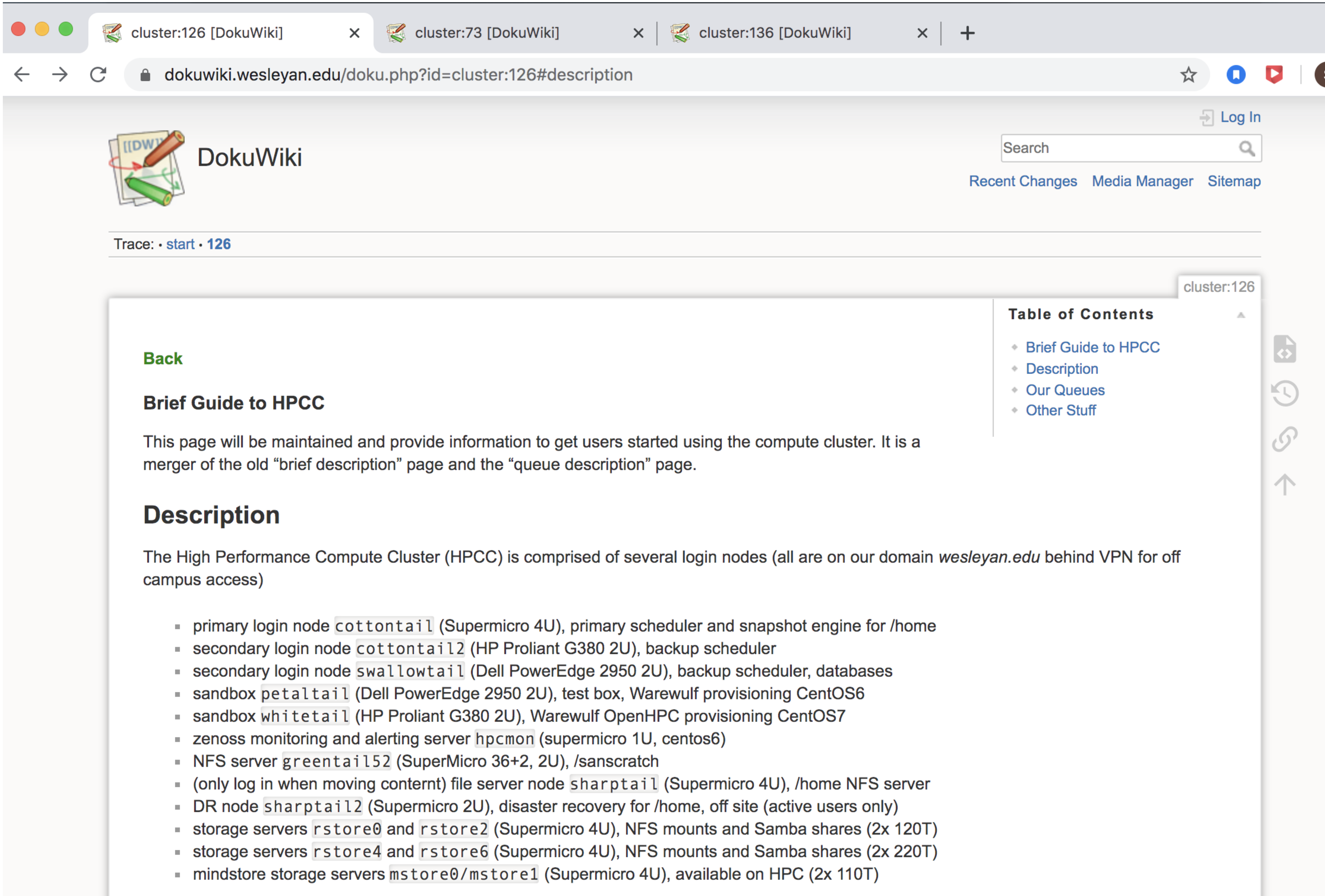
HPCC $\equiv$ GHPCC $\equiv$ MGHPCC $\equiv$ the Cluster

HPC : High performance computing

Cluster : a number of similar things that occur together

Computer Cluster : A set of computers connected together that work as a single unit

MGHPCC has over 10K+ cores available and 400+ TB of high performance storage. It is located in Holyoke MA and provides computing services to the five campuses of UMass.



Back

Brief Guide to HPCC

This page will be maintained and provide information to get users started using the compute cluster. It is a merger of the old "brief description" page and the "queue description" page.

Description

The High Performance Compute Cluster (HPCC) is comprised of several login nodes (all are on our domain *wesleyan.edu* behind VPN for off campus access)

- primary login node `cottontail` (Supermicro 4U), primary scheduler and snapshot engine for /home
- secondary login node `cottontail2` (HP Proliant G380 2U), backup scheduler
- secondary login node `swallowtail` (Dell PowerEdge 2950 2U), backup scheduler, databases
- sandbox `petaltail` (Dell PowerEdge 2950 2U), test box, Warewulf provisioning CentOS6
- sandbox `whitetail` (HP Proliant G380 2U), Warewulf OpenHPC provisioning CentOS7
- zenoss monitoring and alerting server `hpcmon` (supermicro 1U, centos6)
- NFS server `greentail52` (SuperMicro 36+2, 2U), /sanscratch
- (only log in when moving conternt) file server node `sharptail` (Supermicro 4U), /home NFS server
- DR node `sharptail2` (Supermicro 2U), disaster recovery for /home, off site (active users only)
- storage servers `rstore0` and `rstore2` (Supermicro 4U), NFS mounts and Samba shares (2x 120T)
- storage servers `rstore4` and `rstore6` (Supermicro 4U), NFS mounts and Samba shares (2x 220T)
- mindstore storage servers `mstore0/mstore1` (Supermicro 4U), available on HPC (2x 110T)

All queues are available for job submissions via all login nodes. Some nodes on Infiniband switches for parallel computational jobs (queues: me256fd, hp12). Our total job slot count is roughly 2,144 with our physical core count 1,480. Our total teraflops compute capacity is about 58 cpu side, 25 gpu side (double precision floating point) and 702 gpu side (mixed mode). Our total memory footprint is about 528 GB gpu side, 8,532 GB cpu side.

Home directory file system are provided (via NFS or IPoIB) by the node `sharptail` (our file server) from a direct attached disk array. In total, 10 TB of /home disk space is accessible to the users. Node `greentail52` makes available 55 TB of scratch space at /sanscratch via NFS. In addition all nodes provide local scratch space at /localscratch (excludes queue tinymem). The scheduler automatically makes directories in both these scratch areas for each job (named after JOBPID). Backup services for /home are provided via disk-to-disk point-in-time snapshots from node `sharptail` to node `cottontail` disk arrays. (daily, weekly, monthly snapshots are mounted read only on `cottontail` for self-serve content retrievals). Some faculty have their home directories on node `ringtail` which provides 33 TB via /home33. Some faculty also have their own storage (2x 110 TB via /mindstore). In addition no-quota, no-backup user directories can be requested in /homeextra1 (7 T) or /homeextra2 (5 T). All home directories will migrate to a FreeNAS/ZFS appliance named `hpcstore` in 2020 (190T usable, scalable to 1.2P).

Two (old) Rstore storage servers each provide about 104 TB of usable backup space which is not mounted on the compute nodes. Each Rstore server's content is replicated to a dedicated passive standby server of same size, located in same data center but in different racks. As of Spring 2019 we have added two new Rstore servers of 220 T each, fully backed up with replication.

Our Queues

Commercial software has their own queue limited by available licenses. There are no scheduler license resources, just queue jobs up in appropriate queue. Commercial software jobs are processed on the nodes of mw256fd and mw128.

Queue	Nr Of Nodes	Total GB Mem Per Node	Total Cores In Queue	Switch	Hosts	Notes
stata	*na*	*na*	*na*	QDR Infiniband	*any host*	6 licenses

Note: Matlab and Mathematica now have “unlimited licenses”.

Queue	Nr Of Nodes	Total GB Mem Per Node	Job SLots In Queue	Switch	Hosts	Notes
hp12	32	12	256	QDR infiniband	n1-n32	CPU
mwgpu	5	256	120	QDR infiniband	n33-n37	GPU & CPU
mw256fd	8	256	192	QDR infiniband	n38-n45	CPU
tinymem	14	32	448	gigabit ethernet	n39-n59	CPU
mw128	18	128	648	gigabit ethernet	n60-n77	CPU
amber128	1	128	24	gigabit ethernet	n78	GPU & CPU
exx96	12	96	432	gigabit ethernet	n79-n90	GPU & CPU

Reaching the Nodes

We do **NOT** use the head node (ghpcc06) to process big data.
We use the cluster nodes to process it.

How do we reach the nodes?

We submit our commands as jobs to a *job scheduler* and the job scheduler finds an available node for us having the sufficient resources (cores & memory.)

Job Scheduler

Job Scheduler is a software that manages the resources of a cluster system. It manages the program execution in the nodes. It puts the *jobs* in a (priority) queue and executes them on a node when the requested resources become available.

Let’s submit another job and specify the resources this time.
To set

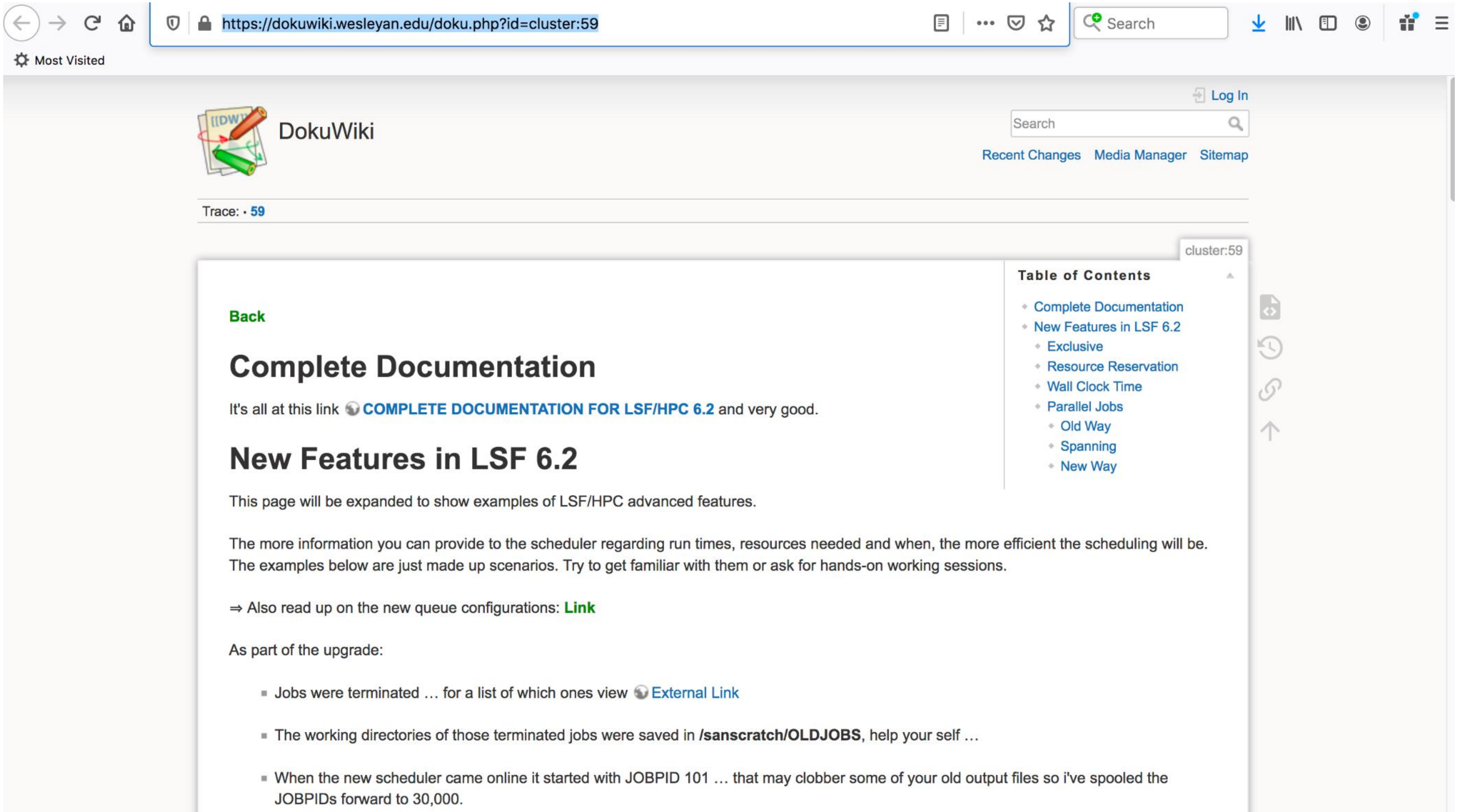
1. We explicitly state that we request a single core, `-n 1`
2. The memory limit to 1024 MB, we add `-R rusage[mem=1024]`
3. Time limit to 20 minutes, we add `-W 20`
4. Queue to short, we add `-q short`

```
$   bsub  -n 1 -R rusage[mem=1024] -W 20 -q short "sleep 300"
```

We need 4 cores as we’l run our process in 4 threads, so we need `-n 4`.
2 GB = 2048 MB, so we need the parameter `-R rusage[mem=2048]`.
We can **estimate** the running time to be 20 / 4 = 5 hours = 300 mins. So, let’s ask for 330 mins to be on the safer side.

```
$ bsub -R span[hosts=1] -n 4 -R rusage[mem=2048] -W 330 -q long "~/bin/myscript.pl -p 4"
```

https://dokuwiki.wesleyan.edu/doku.php?id=cluster:59



Back

Complete Documentation

It's all at this link **COMPLETE DOCUMENTATION FOR LSF/HPC 6.2** and very good.

New Features in LSF 6.2

This page will be expanded to show examples of LSF/HPC advanced features.

The more information you can provide to the scheduler regarding run times, resources needed and when, the more efficient the scheduling will be. The examples below are just made up scenarios. Try to get familiar with them or ask for hands-on working sessions.

⇒ Also read up on the new queue configurations: **Link**

As part of the upgrade:

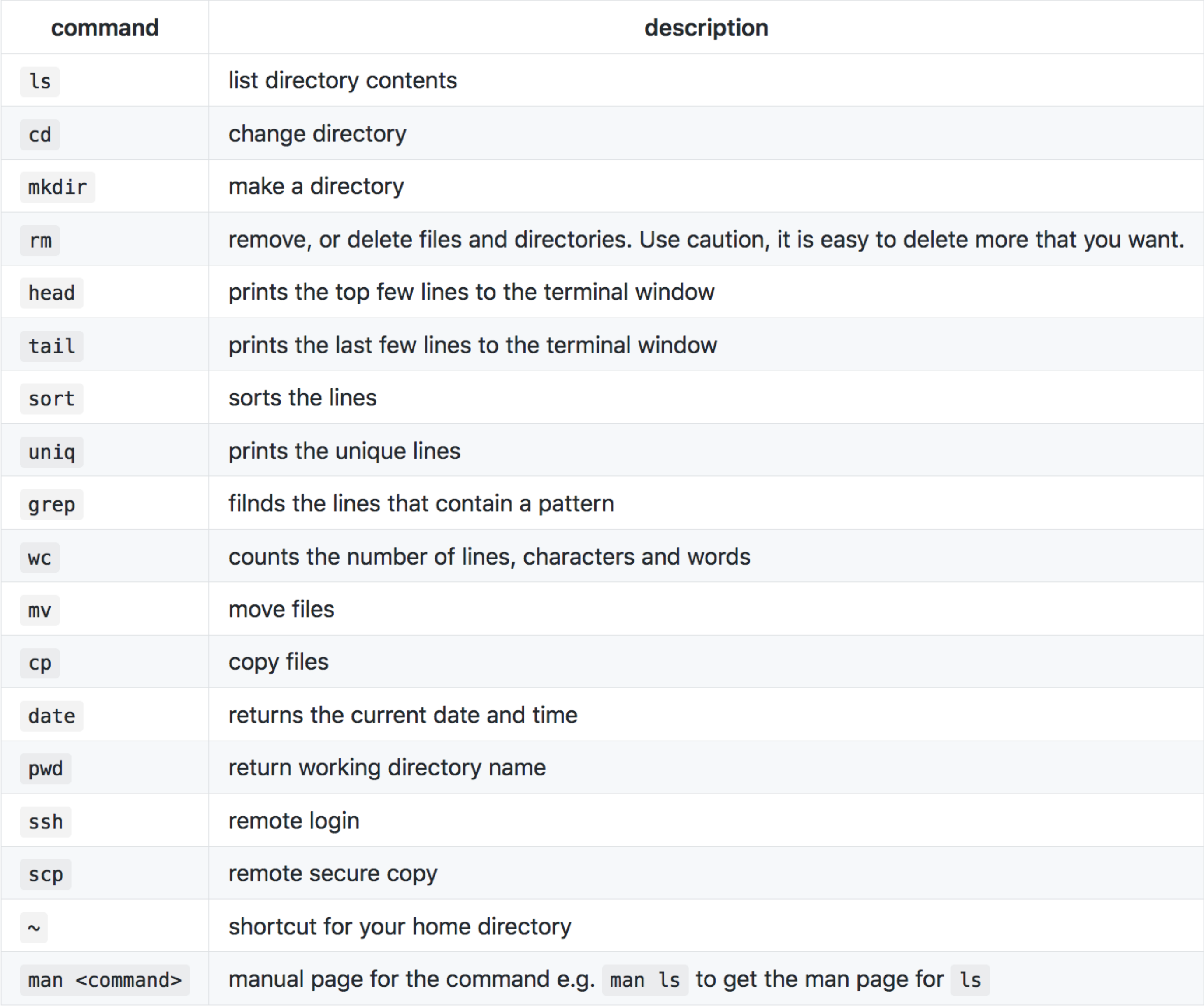
- Jobs were terminated … for a list of which ones view External Link
- The working directories of those terminated jobs were saved in **/sanscratch/OLDJOBS**, help your self …
- When the new scheduler came online it started with JOBPID 101 … that may clobber some of your old output files so i've spooled the JOBPIDs forward to 30,000.

Running Jobs on Linux/Cluster

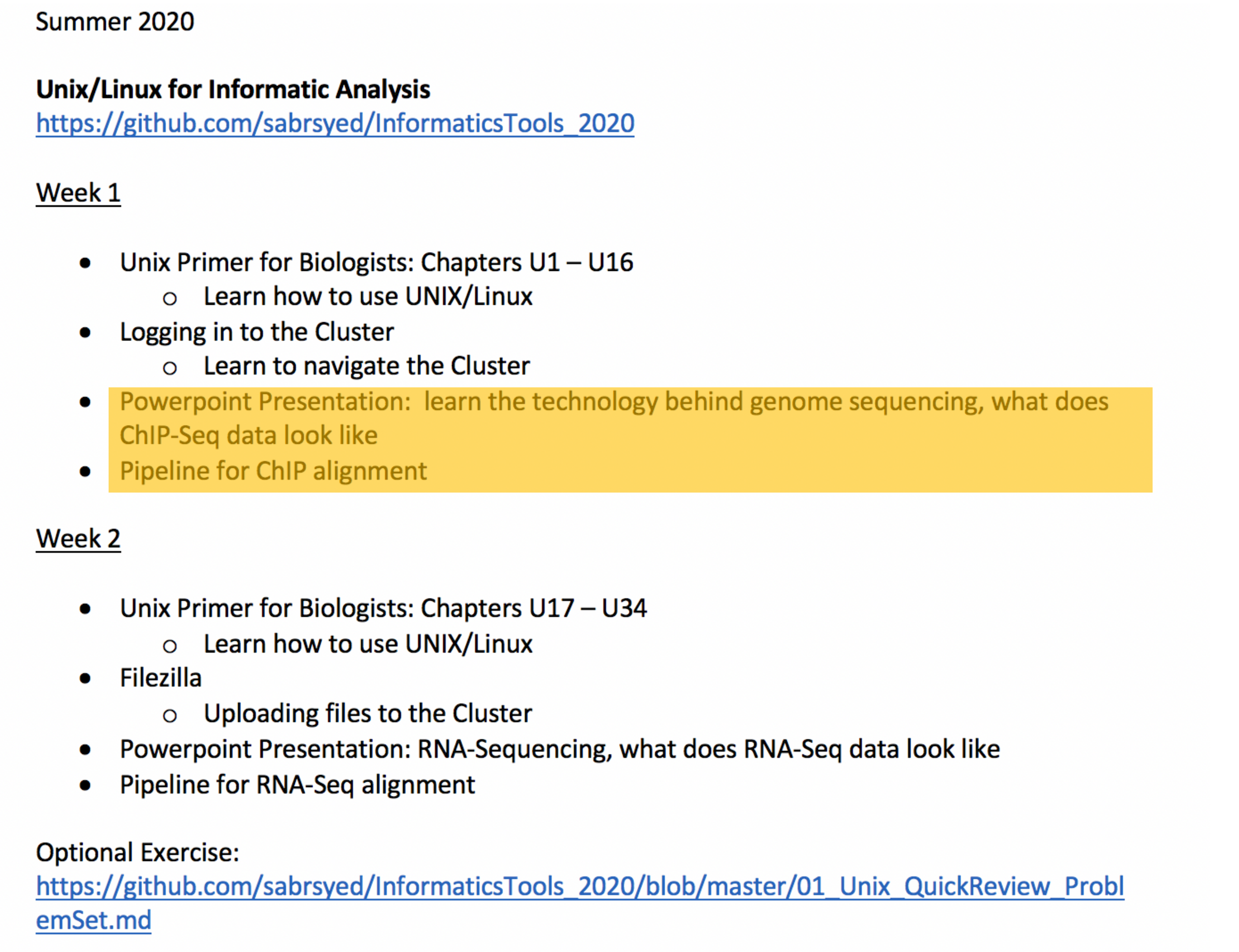
http://barc.wi.mit.edu/education/hot_topics/lsf/Running_jobs_on_Linux_Cluster.pdf

bjobs	checking submitted jobs
bjobs –a	checking recently ended jobs
bjobs -l JOBID	see details of a particular job using the job id #
bpeek JOBID	peek at the stdout and stderr output of unfinished job
bkill JOBID	kills jobs

Quick Review



command	description
`ls`	list directory contents
`cd`	change directory
`mkdir`	make a directory
`rm`	remove, or delete files and directories. Use caution, it is easy to delete more that you want.
`head`	prints the top few lines to the terminal window
`tail`	prints the last few lines to the terminal window
`sort`	sorts the lines
`uniq`	prints the unique lines
`grep`	filnds the lines that contain a pattern
`wc`	counts the number of lines, characters and words
`mv`	move files
`cp`	copy files
`date`	returns the current date and time
`pwd`	return working directory name
`ssh`	remote login
`scp`	remote secure copy
`~`	shortcut for your home directory
`man <command>`	manual page for the command e.g. `man ls` to get the man page for `ls`
less or zless	read text files/read .gz compressed text files



Summer 2020

Unix/Linux for Informatic Analysis
https://github.com/sabrsyed/InformaticsTools_2020

Week 1

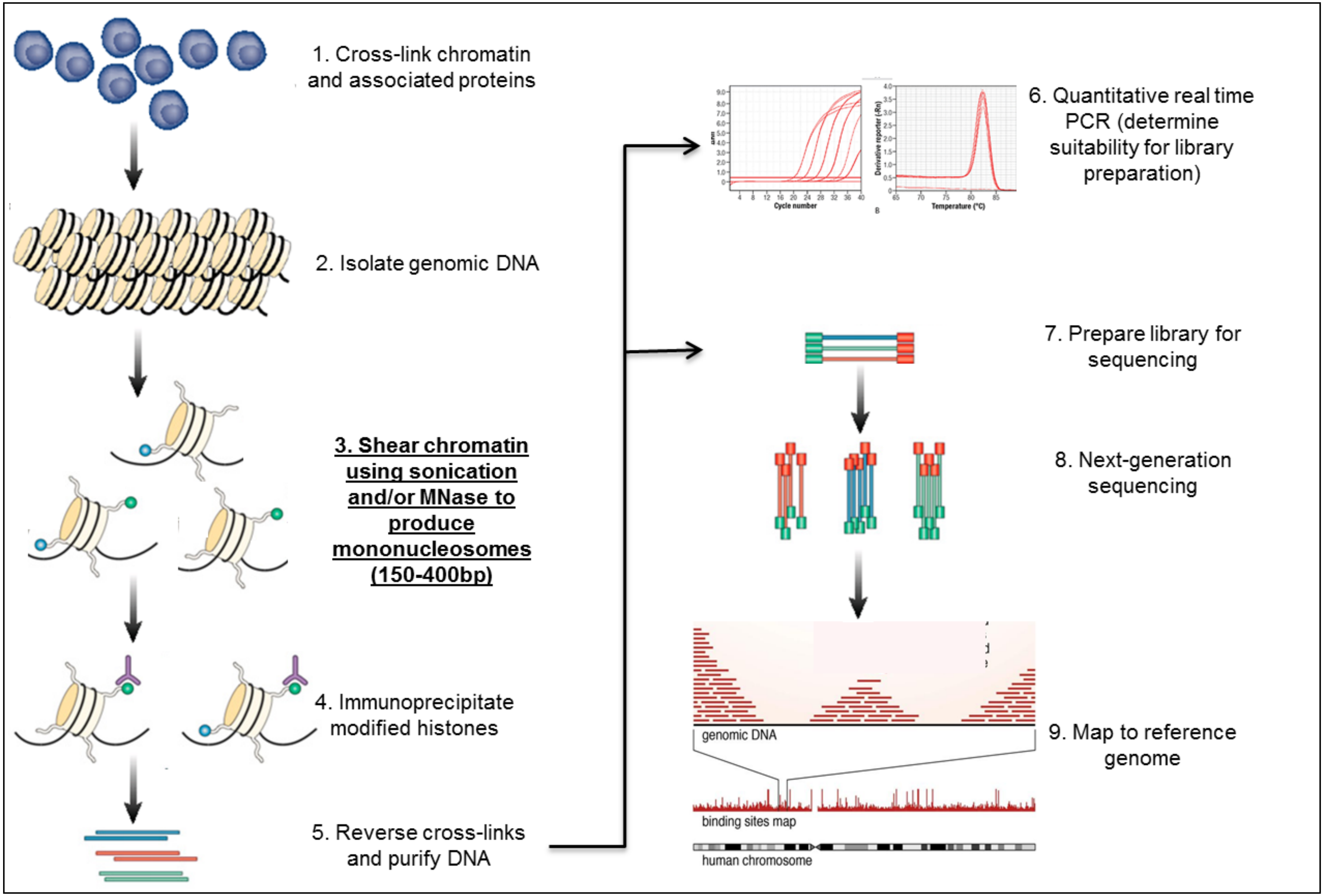
- Unix Primer for Biologists: Chapters U1 – U16
  - Learn how to use UNIX/Linux
- Logging in to the Cluster
  - Learn to navigate the Cluster
- Powerpoint Presentation: learn the technology behind genome sequencing, what does ChIP-Seq data look like
- Pipeline for ChIP alignment

Week 2

- Unix Primer for Biologists: Chapters U17 – U34
  - Learn how to use UNIX/Linux
- Filezilla
  - Uploading files to the Cluster
- Powerpoint Presentation: RNA-Sequencing, what does RNA-Seq data look like
- Pipeline for RNA-Seq alignment

Optional Exercise:
https://github.com/sabrsyed/InformaticsTools_2020/blob/master/01_Unix_QuickReview_ProblemSet.md

ChIP-Seq (Chromatin Immunoprecipitation)



1. Cross-link chromatin and associated proteins

2. Isolate genomic DNA

<u>3. Shear chromatin using sonication and/or MNase to produce mononucleosomes (150-400bp)</u>

4. Immunoprecipitate modified histones

5. Reverse cross-links and purify DNA

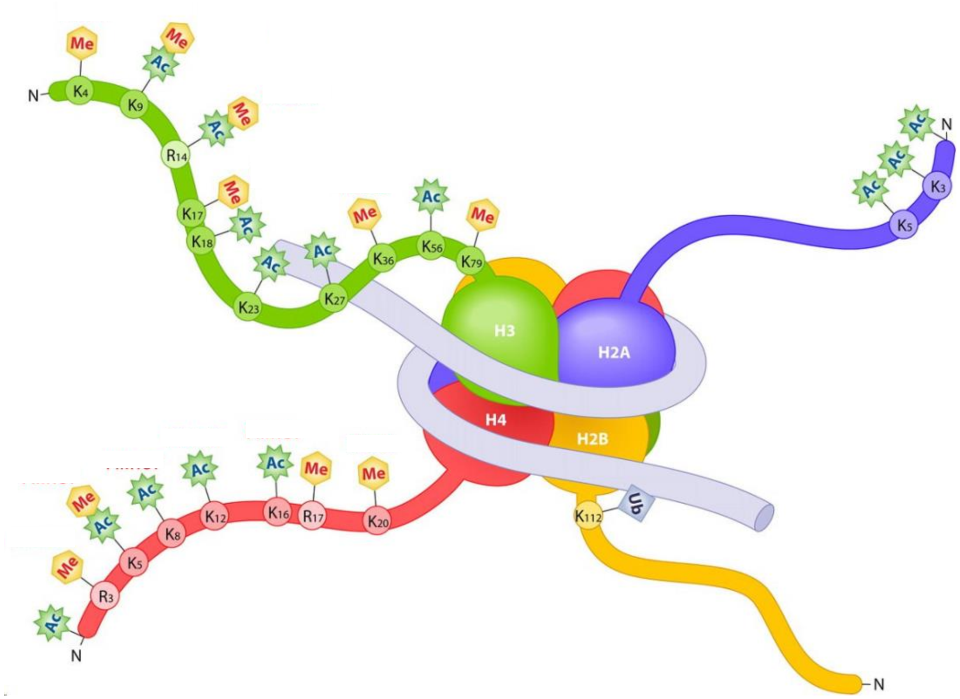
6. Quantitative real time PCR (determine suitability for library preparation)

7. Prepare library for sequencing

8. Next-generation sequencing

9. Map to reference genome

Control of gene expression by histone modifications



- The nucleosome is made up of dimers of core histones H2A, H2B, H3, H4 with 147 base pairs of double stranded DNA wrapped around the nucleosome

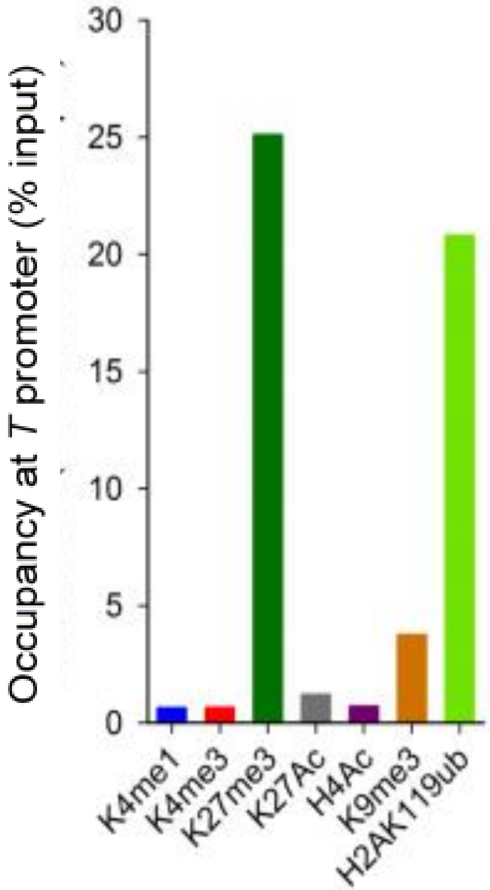
Activating histone modifications

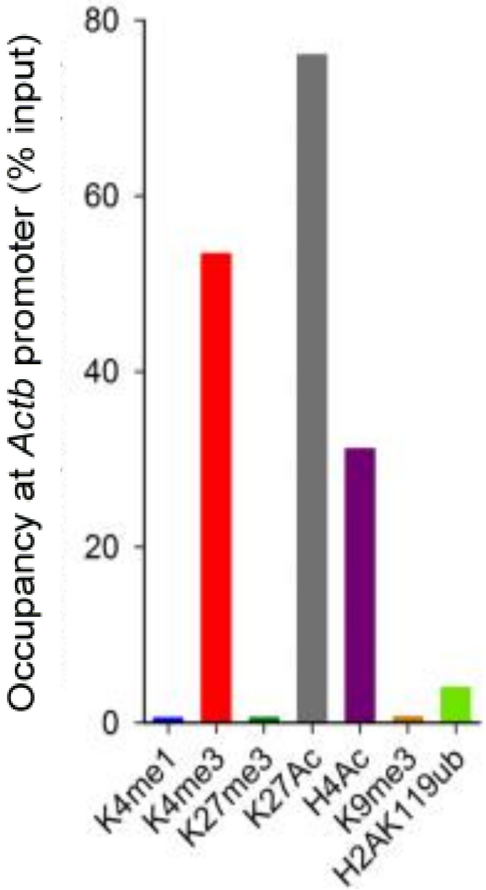
H3K4me3
H3K27ac
H4ac
H3K4me1

Repressive histone modifications

H3K27me3
H3K9me3
H2AK119ubi1

ChIP-qPCR





Activating histone modifications

H3K4me3

H3K27ac

H4ac

Repressive histone modifications

H3K27me3

H3K9me3

H2AK119ubi1

H3K4me1

HiSeq 2000

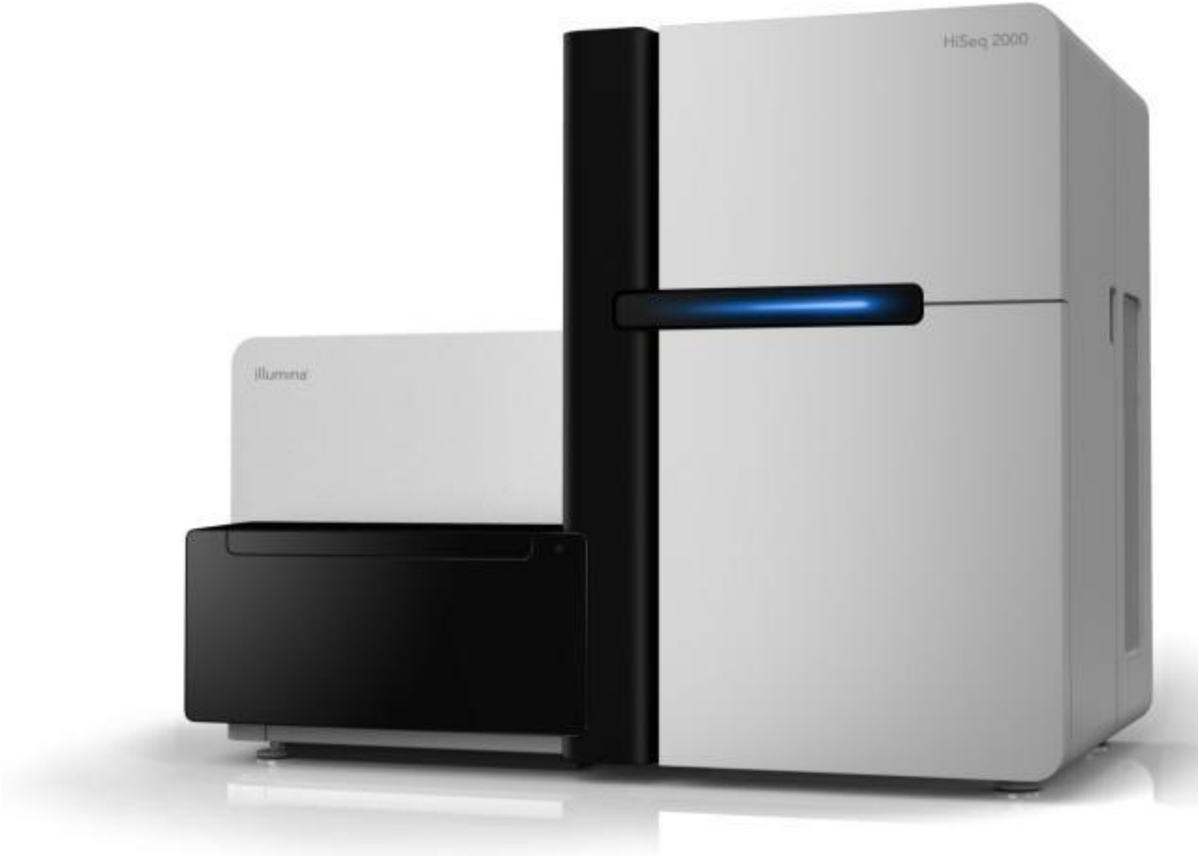
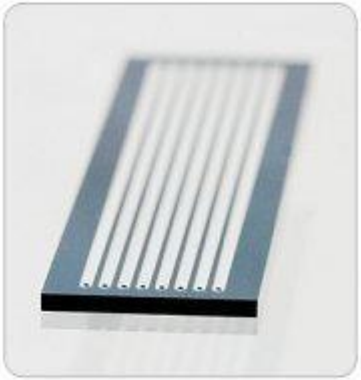


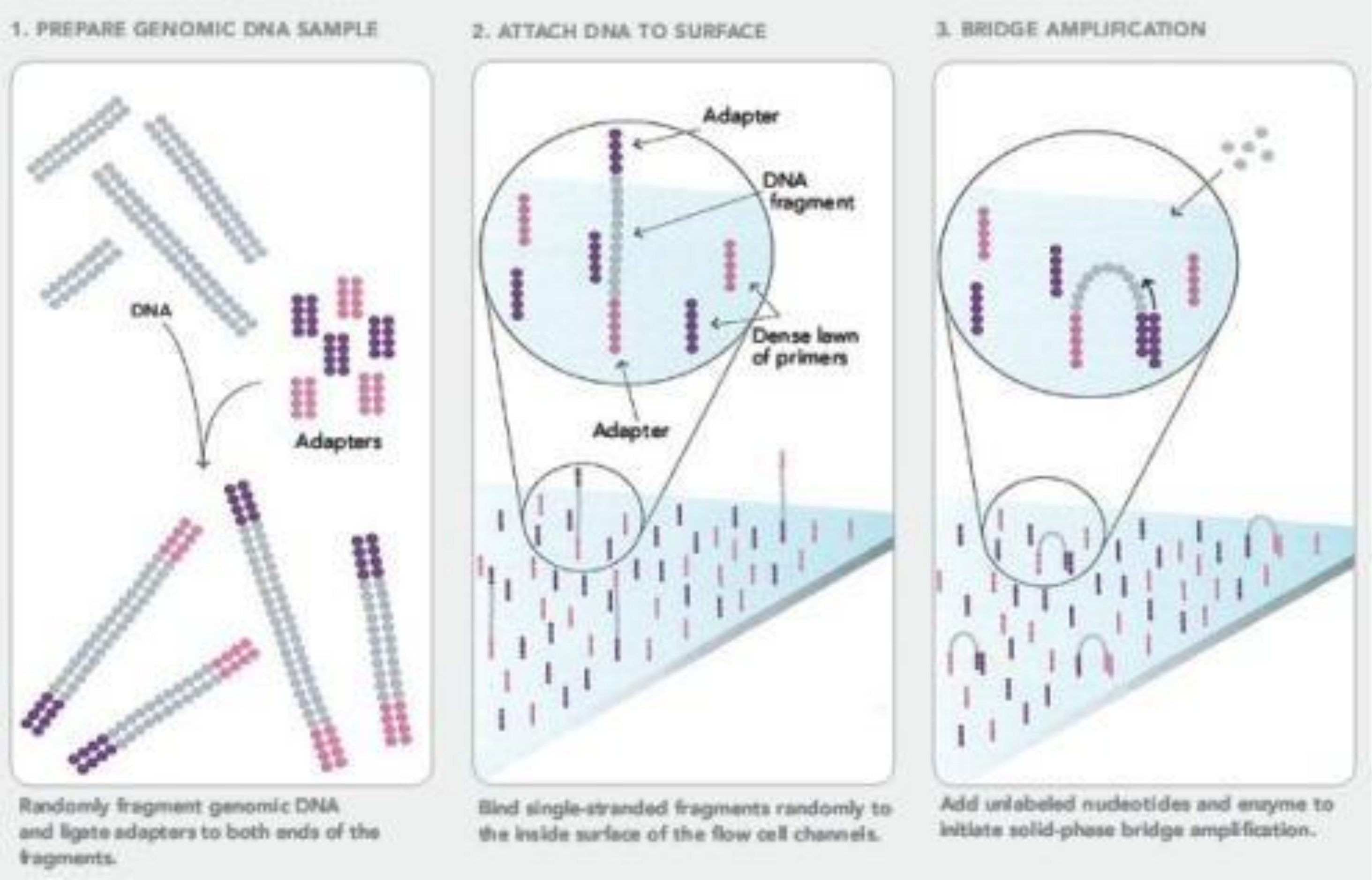
FIGURE 1: ILLUMINA GENOME ANALYZER FLOW CELL



Up to eight samples can be loaded onto the flow cell for simultaneous analysis on the Illumina Genome Analyzer.

Illumina 2000 HiSeq

Bridge Amplification

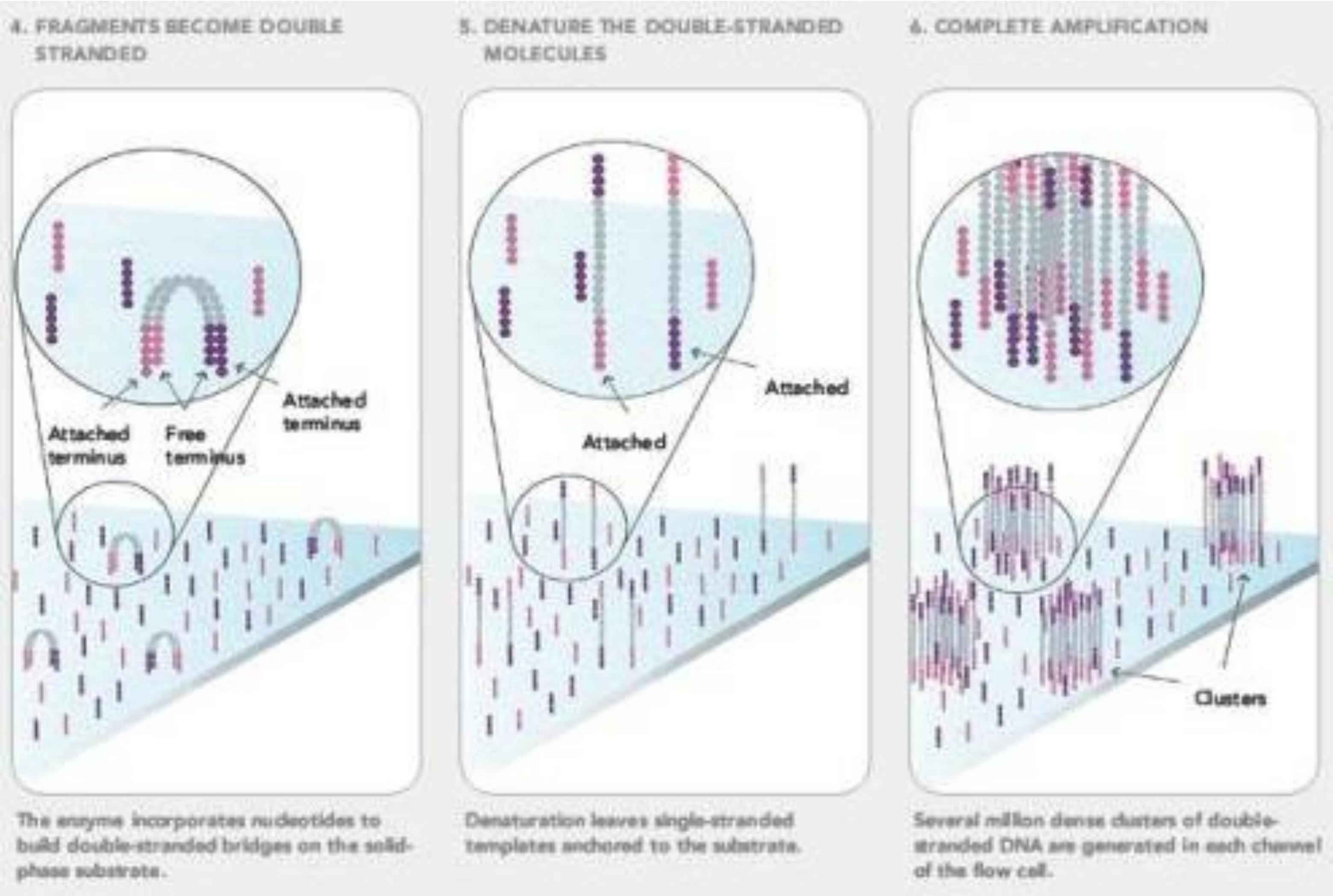


Randomly fragment genomic DNA and ligate adapters to both ends of the fragments.

Bind single-stranded fragments randomly to the inside surface of the flow cell channels.

Add unlabeled nucleotides and enzyme to initiate solid-phase bridge amplification.

Bridge Amplification



4. FRAGMENTS BECOME DOUBLE STRANDED

5. DENATURE THE DOUBLE-STRANDED MOLECULES

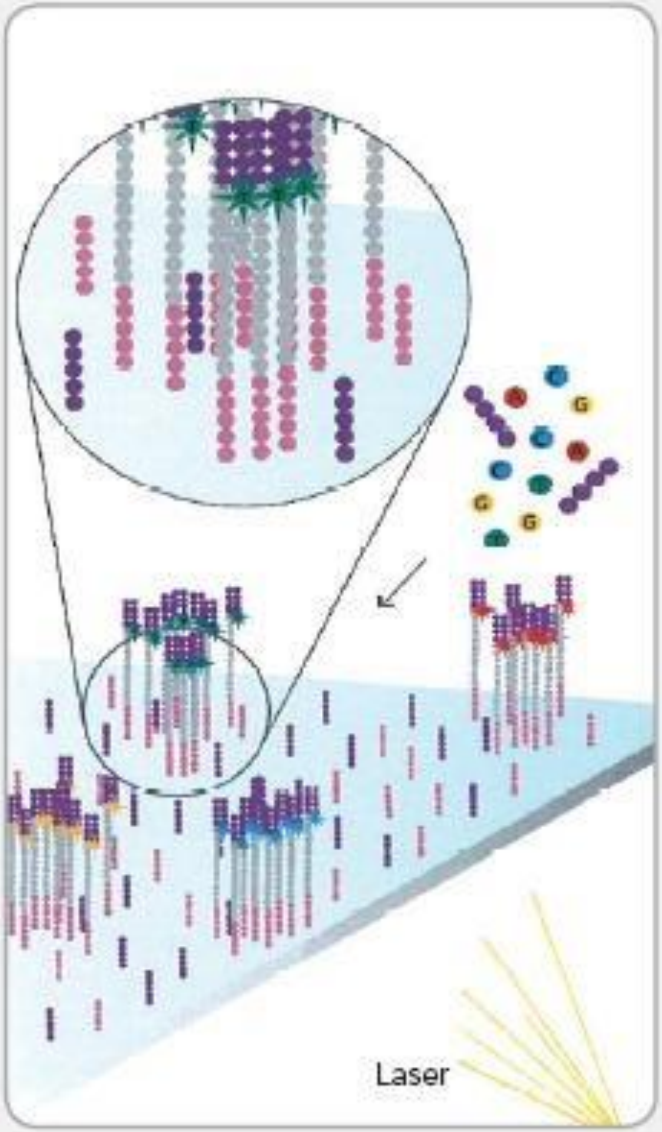
6. COMPLETE AMPLIFICATION

The enzyme incorporates nucleotides to build double-stranded bridges on the solid-phase substrate.

Denaturation leaves single-stranded templates anchored to the substrate.

Several million dense clusters of double-stranded DNA are generated in each channel of the flow cell.

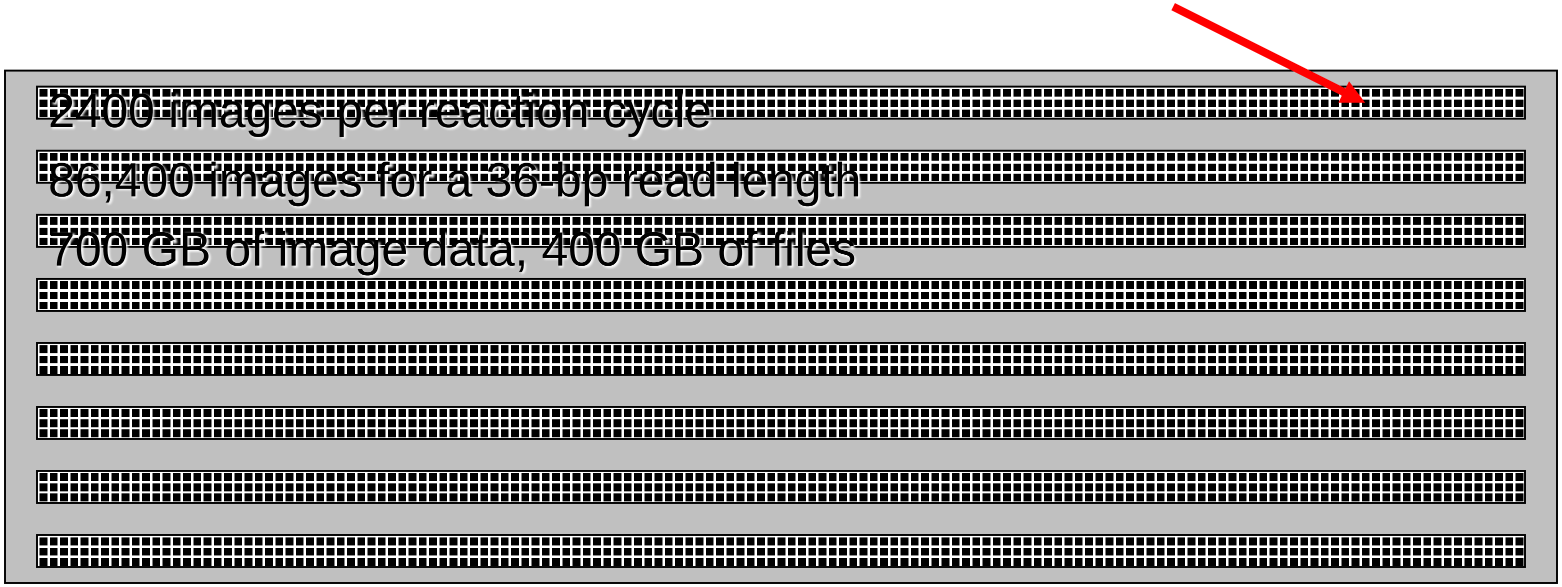
7. DETERMINE FIRST BASE



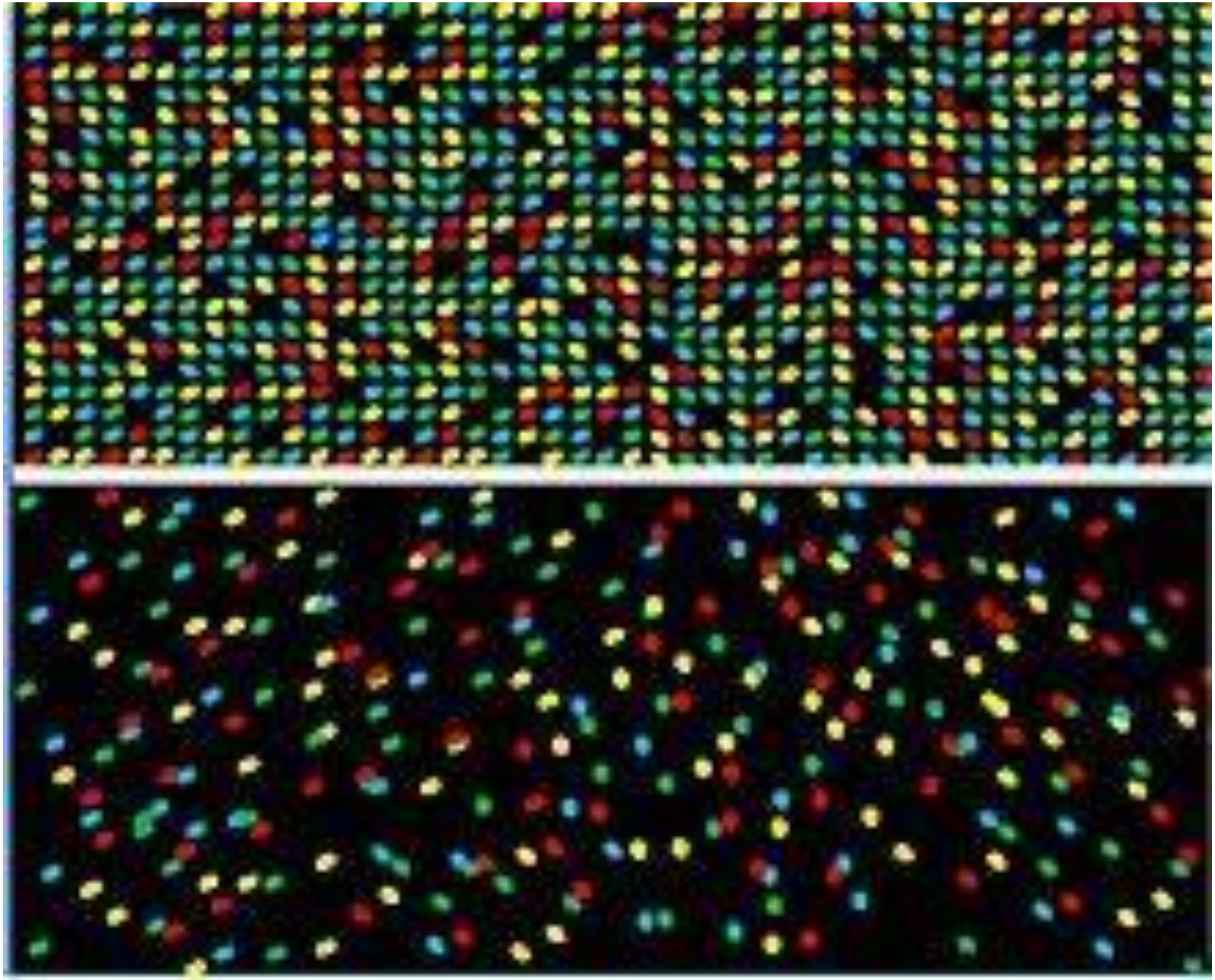
First chemistry cycle: to initiate the first sequencing cycle, add all four labeled reversible terminators, primers and DNA polymerase enzyme to the flow cell.

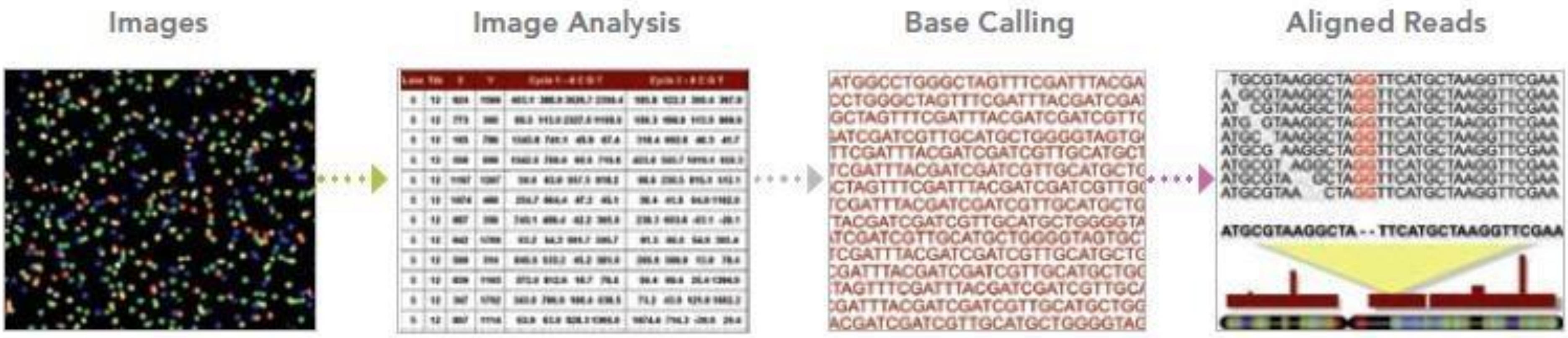
Raw Data is Images

- 8 channels per flow cell
- 300 tiles per channel
- 20,000 clusters/reads per tile



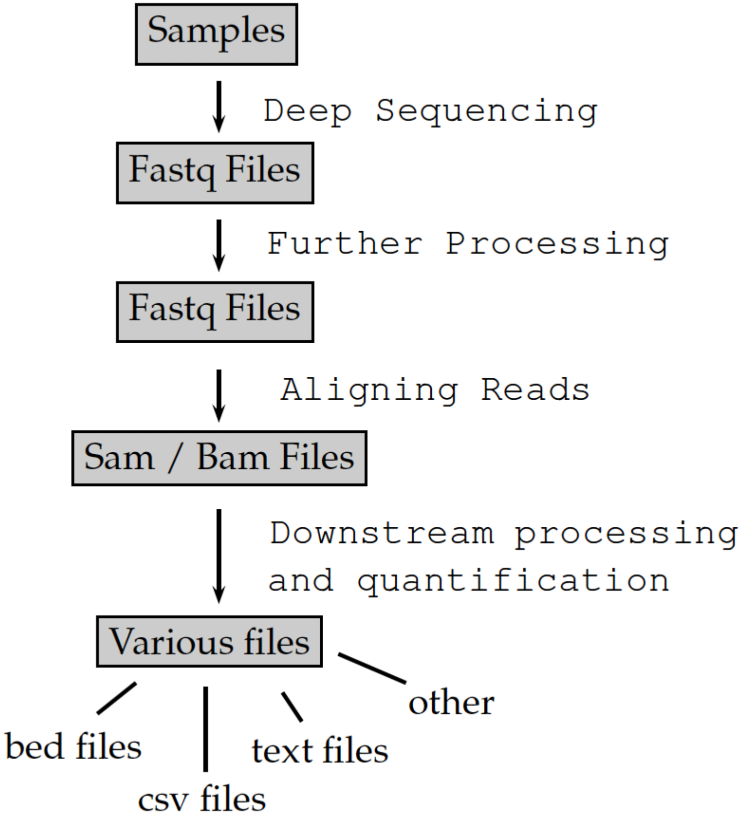
- 2400 images per reaction cycle
- 86,400 images for a 36-bp read length
- 700 GB of image data, 400 GB of files





- Interpret images as intensities
- Convert intensities to base calls
- Assemble reads into complete sequence
  - ⇒ 36-bp read length
  - ⇒ 4 to 6 million reads per flow cell lane

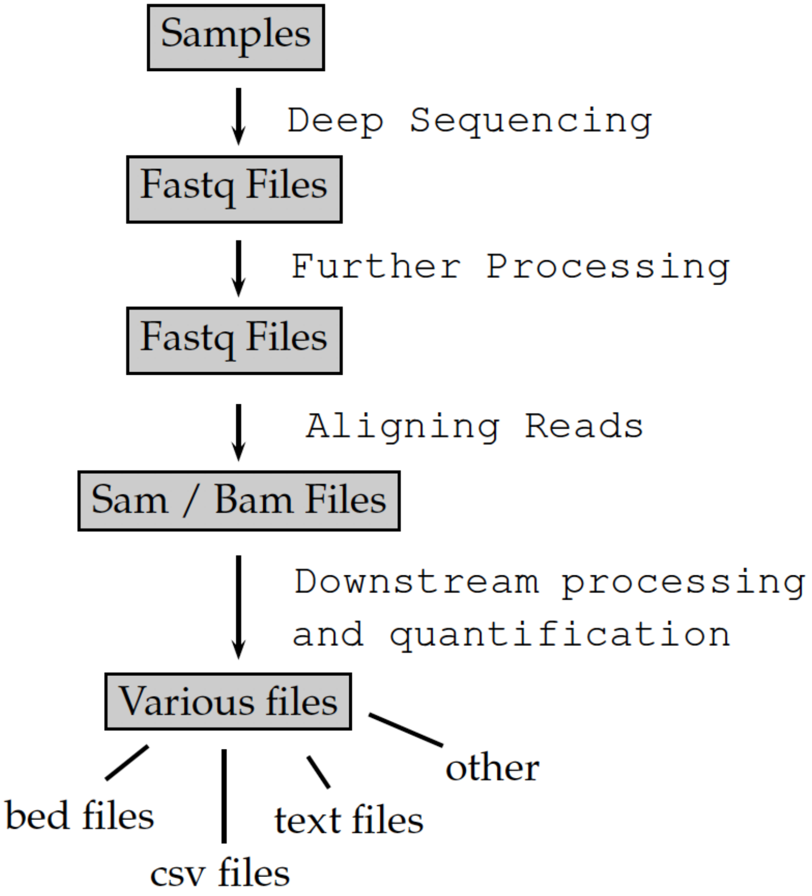
A Typical Deep-Sequencing Workflow



Deep Sequencing Data pipelines involve a lot of text processing.

This is an oversimplified model and your workflow can look different from this!

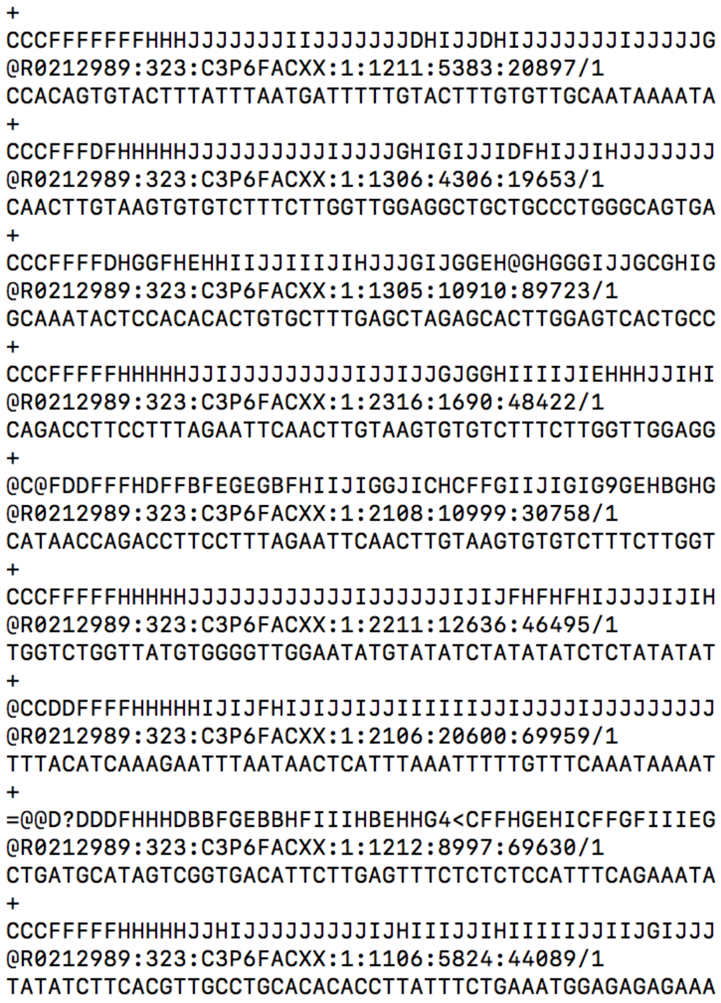
SAM / BAM Files



When a fastq file is aligned against a reference genome, a sam or a bam file is created as the ultimate output of the alignment. These files tell us where and how reads in the fastq file are mapped.

Sequencing Workflow

- FASTQ: a text-based format for storing nucleotide sequences (reads) and their quality scores. [1]

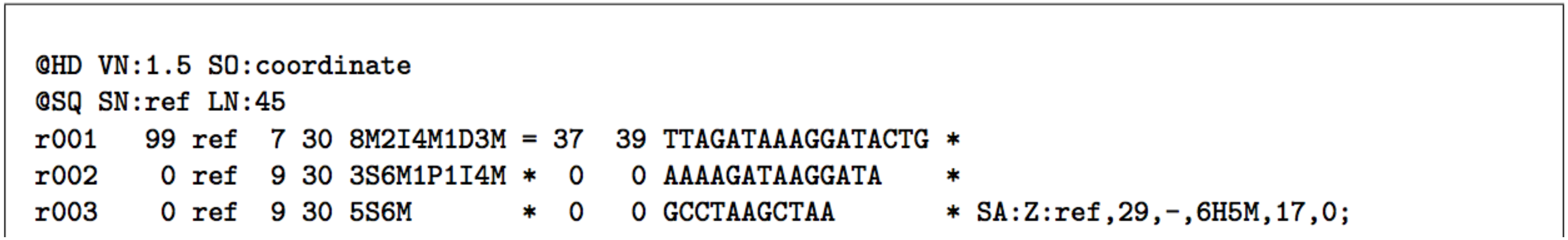


```
+
CCCFFFFFFFHHHJJJJJJJIIJJJJJJJDHIJJDHIJJJJJJJIJJJJJG
@R0212989:323:C3P6FACXX:1:1211:5383:20897/1
CCACAGTGTACTTTATTTAATGATTTTTGTACTTTGTGTTGCAATAAAATA
+
CCCFFFDFHHHHHJJJJJJJJJJIJJJJGHIGIJJIDFHIJJIHJJJJJJJ
@R0212989:323:C3P6FACXX:1:1306:4306:19653/1
CAACTTGTAAGTGTGTCTTTCTTGGTTGGAGGCTGCTGCCCTGGGCAGTGA
+
CCCFFFFDHGGFHEHHIIJJIIIJIHJJJGIJGGEH@GHGGGIJJGCGHIG
@R0212989:323:C3P6FACXX:1:1305:10910:89723/1
GCAAATACTCCACACACTGTGCTTTGAGCTAGAGCACTTGGAGTCACTGCC
+
CCCFFFFFHHHHHJJIJJJJJJJJJIJJIJJGJGGHIIIIJIEHHHJJIHI
@R0212989:323:C3P6FACXX:1:2316:1690:48422/1
CAGACCTTCCTTTAGAATTCAACTTGTAAGTGTGTCTTTCTTGGTTGGAGG
+
@C@FDDFFFHDFFBFEGEGBFHIIJIGGJICHCFFGIIJIGIG9GEHBGHG
@R0212989:323:C3P6FACXX:1:2108:10999:30758/1
CATAACCAGACCTTCCTTTAGAATTCAACTTGTAAGTGTGTCTTTCTTGGT
+
CCCFFFFFHHHHHJJJJJJJJJJJIJJJJJIJIJFHFHFHIJJJJIJIH
@R0212989:323:C3P6FACXX:1:2211:12636:46495/1
TGGTCTGGTTATGTGGGGTTGGAATATGTATATCTATATATCTCTATATAT
+
@CCDDFFFFHHHHHIJIJFHIJIJJIJJIIIIIIJJIJJJJIJJJJJJJJ
@R0212989:323:C3P6FACXX:1:2106:20600:69959/1
TTTACATCAAAGAATTTAATAACTCATTTAAATTTTTGTTTCAAATAAAAT
+
=@@D?DDDFHHHDBBFGEBBHFIIIHBEHHG4<CFFHGEHICFFGFIIIEG
@R0212989:323:C3P6FACXX:1:1212:8997:69630/1
CTGATGCATAGTCGGTGACATTCTTGAGTTTCTCTCTCCATTTCAGAAATA
+
CCCFFFFFHHHHHJJHIJJJJJJJJJIJHIIIJJIHIIIIIJJIIJGIJJJ
@R0212989:323:C3P6FACXX:1:1106:5824:44089/1
TATATCTTCACGTTGCCTGCACACACCTTATTTCTGAAATGGAGAGAGAAA
```

Sequencing Workflow

- BAM: The Sequence Alignment/Mapping (SAM) format is a text-based format for storing read alignments against reference sequences and it is interconvertible with the binary BAM format.

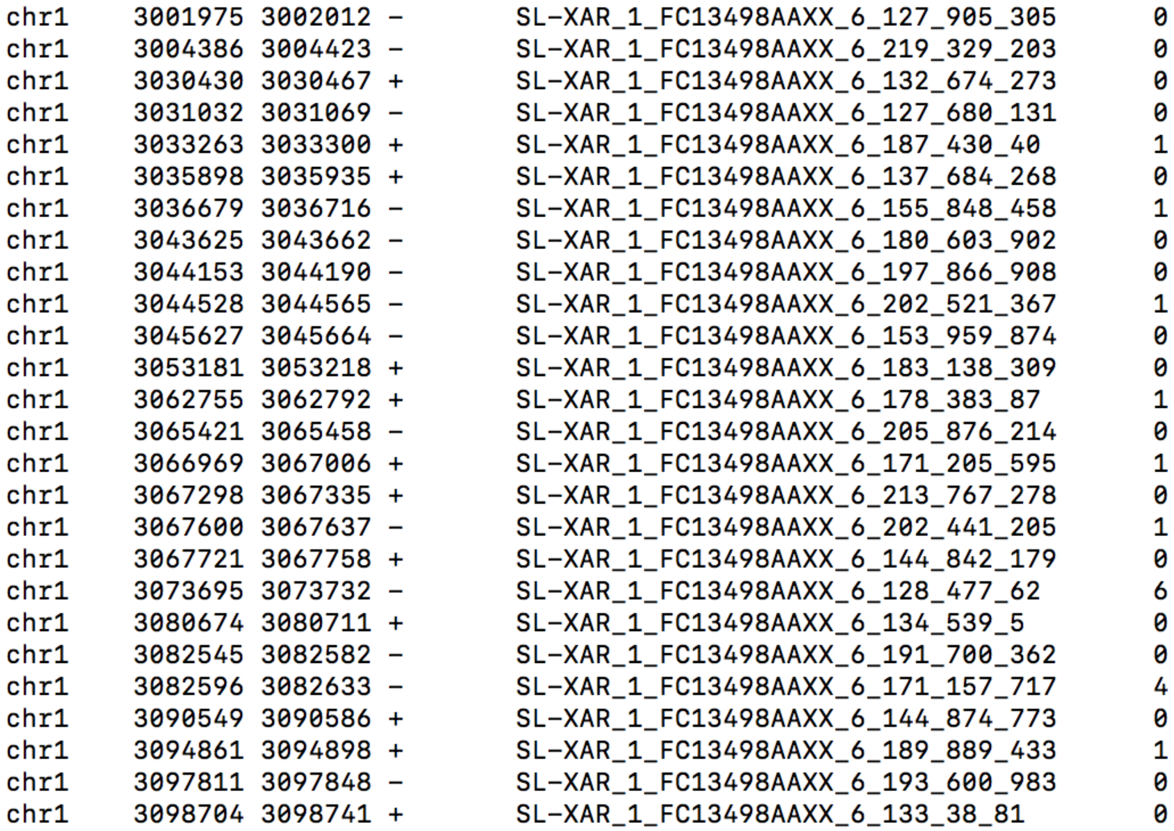
Each alignment line has 11 mandatory fields for essential alignment information such as mapping position, and variable number of optional fields for flexible or aligner specific information.



```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
```

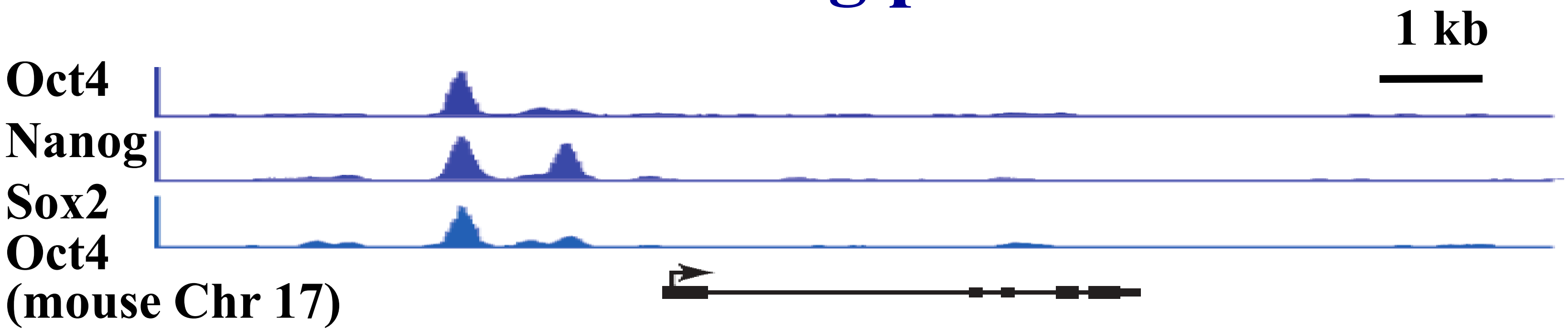
Sequencing Workflow

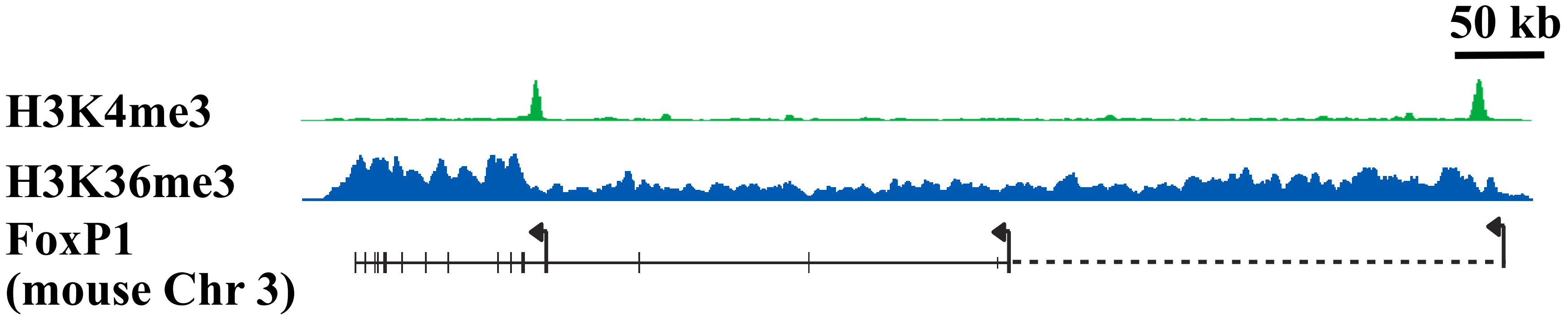
- BED file

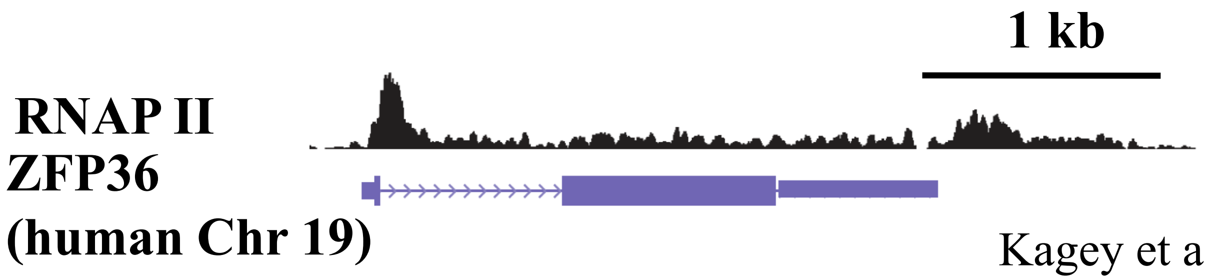


```
chr1    3001975 3002012 -       SL-XAR_1_FC13498AAXX_6_127_905_305      0
chr1    3004386 3004423 -       SL-XAR_1_FC13498AAXX_6_219_329_203      0
chr1    3030430 3030467 +       SL-XAR_1_FC13498AAXX_6_132_674_273      0
chr1    3031032 3031069 -       SL-XAR_1_FC13498AAXX_6_127_680_131      0
chr1    3033263 3033300 +       SL-XAR_1_FC13498AAXX_6_187_430_40       1
chr1    3035898 3035935 +       SL-XAR_1_FC13498AAXX_6_137_684_268      0
chr1    3036679 3036716 -       SL-XAR_1_FC13498AAXX_6_155_848_458      1
chr1    3043625 3043662 -       SL-XAR_1_FC13498AAXX_6_180_603_902      0
chr1    3044153 3044190 -       SL-XAR_1_FC13498AAXX_6_197_866_908      0
chr1    3044528 3044565 -       SL-XAR_1_FC13498AAXX_6_202_521_367      1
chr1    3045627 3045664 -       SL-XAR_1_FC13498AAXX_6_153_959_874      0
chr1    3053181 3053218 +       SL-XAR_1_FC13498AAXX_6_183_138_309      0
chr1    3062755 3062792 +       SL-XAR_1_FC13498AAXX_6_178_383_87       1
chr1    3065421 3065458 -       SL-XAR_1_FC13498AAXX_6_205_876_214      0
chr1    3066969 3067006 +       SL-XAR_1_FC13498AAXX_6_171_205_595      1
chr1    3067298 3067335 +       SL-XAR_1_FC13498AAXX_6_213_767_278      0
chr1    3067600 3067637 -       SL-XAR_1_FC13498AAXX_6_202_441_205      1
chr1    3067721 3067758 +       SL-XAR_1_FC13498AAXX_6_144_842_179      0
chr1    3073695 3073732 -       SL-XAR_1_FC13498AAXX_6_128_477_62       6
chr1    3080674 3080711 +       SL-XAR_1_FC13498AAXX_6_134_539_5        0
chr1    3082545 3082582 -       SL-XAR_1_FC13498AAXX_6_191_700_362      0
chr1    3082596 3082633 -       SL-XAR_1_FC13498AAXX_6_171_157_717      4
chr1    3090549 3090586 +       SL-XAR_1_FC13498AAXX_6_144_874_773      0
chr1    3094861 3094898 +       SL-XAR_1_FC13498AAXX_6_189_889_433      1
chr1    3097811 3097848 -       SL-XAR_1_FC13498AAXX_6_193_600_983      0
chr1    3098704 3098741 +       SL-XAR_1_FC13498AAXX_6_133_38_81        0
```

Binding profile

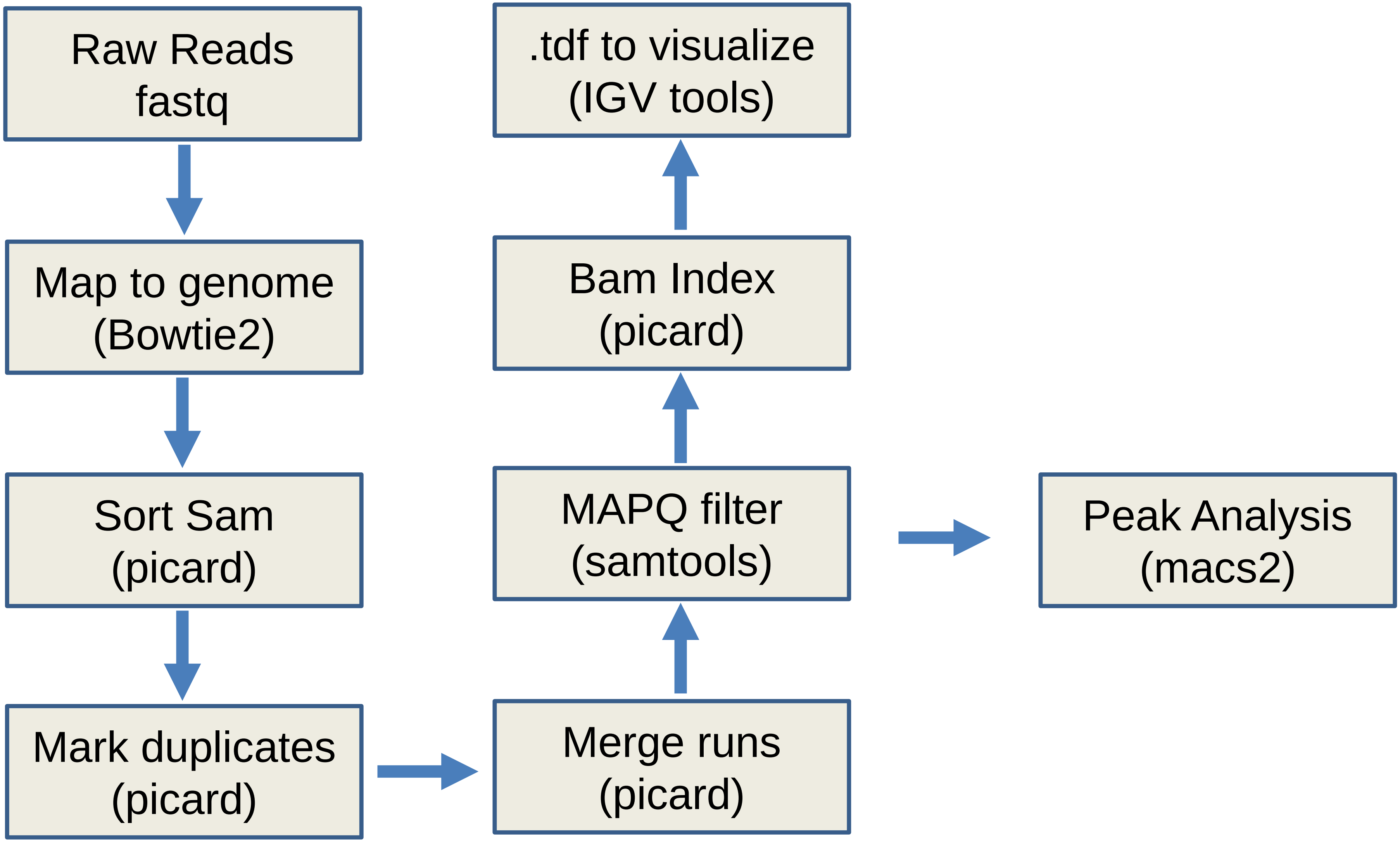






Kagey et al. (2010). Nature 467: 430
Mikkelsen et al. (2007). Nature 448: 553
Pepke et al. (2009). Nat. Methods 6: S22

My ChIP-Sequencing Workflow



Raw Reads
fastq

Map to genome
(Bowtie2)

Sort Sam
(picard)

Mark duplicates
(picard)

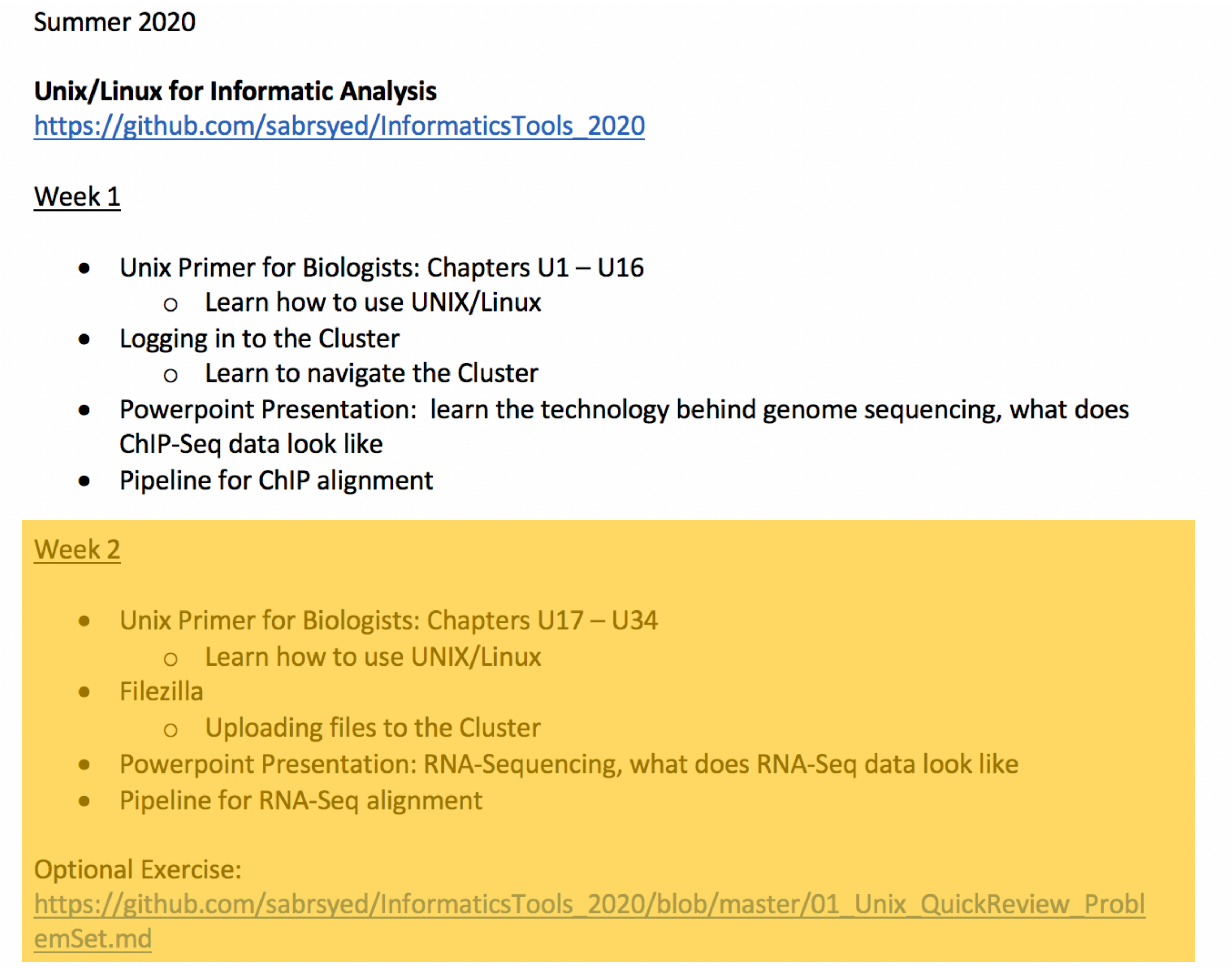
Merge runs
(picard)

MAPQ filter
(samtools)

Bam Index
(picard)

.tdf to visualize
(IGV tools)

Peak Analysis
(macs2)



Summer 2020

Unix/Linux for Informatic Analysis
https://github.com/sabrsyed/InformaticsTools_2020

Week 1

- Unix Primer for Biologists: Chapters U1 – U16
    - Learn how to use UNIX/Linux
- Logging in to the Cluster
    - Learn to navigate the Cluster
- Powerpoint Presentation: learn the technology behind genome sequencing, what does ChIP-Seq data look like
- Pipeline for ChIP alignment

Week 2

- Unix Primer for Biologists: Chapters U17 – U34
    - Learn how to use UNIX/Linux
- Filezilla
    - Uploading files to the Cluster
- Powerpoint Presentation: RNA-Sequencing, what does RNA-Seq data look like
- Pipeline for RNA-Seq alignment

Optional Exercise:
https://github.com/sabrsyed/InformaticsTools_2020/blob/master/01_Unix_QuickReview_ProblemSet.md