WESTGRID

compute canada
regional partner

UBC

UBC100

Advanced Research Computing

WestGrid & ARC Summer School

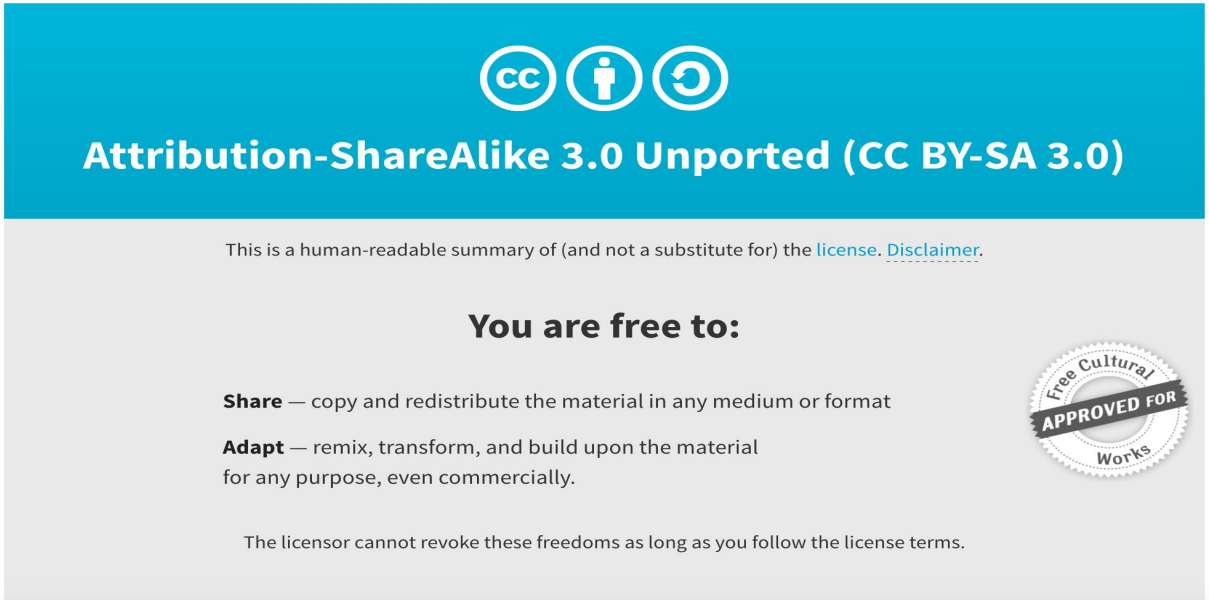
Short read mapping and visualization: Core aspects of next-generation sequencing data analysis

June 17th 2020 (9:00AM-12:00PM PST)

Phillip A Richmond & Oriol Fornes (TA)

Copyright Information

creative commons



The material is open source, and in this presentation no previous external work was utilized.

WESTGRID

UBC

UBC100

Advanced Research Computing

Welcome!

- Welcome to the Introduction to Short Read Mapping
- In this tutorial you will learn how to map Illumina short reads against a reference genome using the Compute Canada High Performance Computing (HPC) cluster “Cedar”
- Soft-start at 9:15, while we figure out login information and access to the Cedar cluster.
- This presentation: https://bit.ly/WGSSBioinformaticsJune17

WESTGRID

UBC 100

Advanced Research Computing

Interactive Experience

We hope this is an interactive experience for all of you.

Questions/Problems can be posted to the etherpad:

https://etherpad.opendev.org/p/_B8ETduCObCQ6BN-811p:

We have a fantastic TA (Dr. Oriol Fornes) to assist in answering questions and solving problems while I'm presenting, at the end of the session I can address unresolved questions

Speaker Bio

Phillip Richmond

PhD Candidate, Wasserman Lab, BC Children's Hospital Research Institute

Bioinformatics Program, University of British Columbia

https://phillip-a-richmond.github.io

Research:Maximizing the Utility of Whole Genome Sequencing in the Diagnosis of Rare Genetic Disorders

Previous work in Genomics: Genomic Contributions to Ethanol Sensitivity in Mice, Polyploid Evolution in Yeast, Brewing Yeast Genomics, Cancer Cell Epigenetics, Addiction Predisposition

Also loves teaching genomics, and my puppy Sherlock Holmes

WESTGRID

@sherlock_holmes_doodle

UBC

UBC100

Advanced Research Computing

Session Outline

- Lecture 1
  - Introduction to next generation sequencing data & diverse data types
  - Transcriptional cis-regulatory datasets and where to find them
  - Accessing videos and support
- Video 1 - Initialize Workshop Directory and Explore Data
- Video 2 - Map and convert data using BWA mem and Samtools
- Mini check-in
- Video 3 - Visualize data using IGV
- Problem sets
- Closing remarks

Session Outline (Rough Timeline)

- Lecture 1 (9:15-9:30)
  - Introduction to next generation sequencing data & diverse data types
  - Transcriptional cis-regulatory datasets and where to find them
  - Accessing videos and support
- Video 1 - Initialize Workshop Directory and Explore Data (9:30-10:15)
- Video 2 - Map and convert data using BWA mem and Samtools (10:15-11:00)
- Mini check-in (11:00-11:15)
- Video 3 - Visualize data using IGV (11:15-12:00)
- Problem sets
- Closing remarks

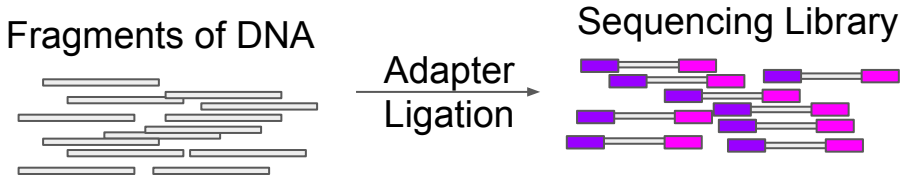
Session Outline

- Lecture 1
  - Introduction to next generation sequencing data & diverse data types
  - Transcriptional cis-regulatory datasets and where to find them
  - Accessing videos and support
- Video 1 - Initialize Workshop Directory and Explore Data
- Video 2 - Map and convert data using BWA mem and Samtools; call peaks with MACS2
- Mini Check-in
- Video 3 - Visualize data using IGV
- Problem sets
- Closing remarks

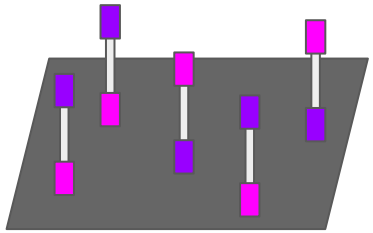
Next generation sequencing: Short-read sequencing

Fragments of DNA

Next generation sequencing: Short-read sequencing

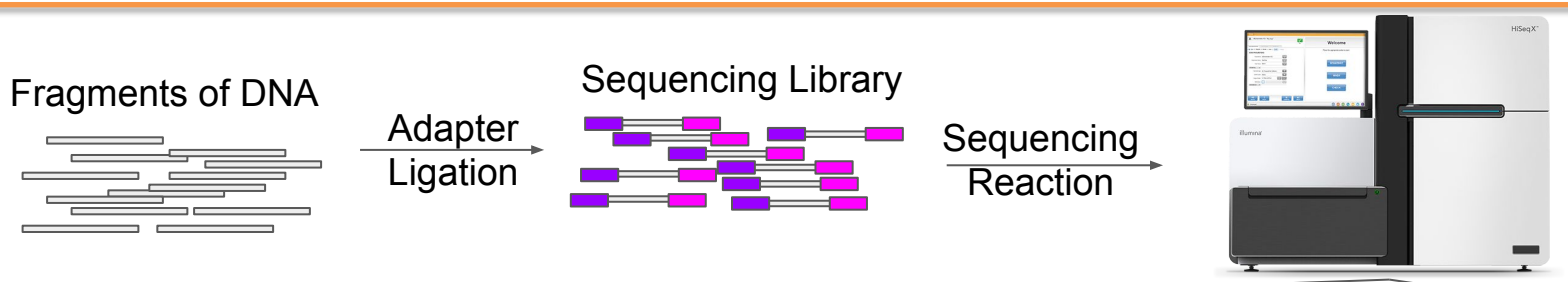


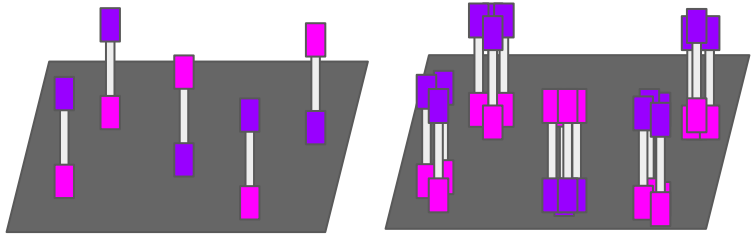
Next generation sequencing: Short-read sequencing



1-Ligate to flowcell

Next generation sequencing: Short-read sequencing

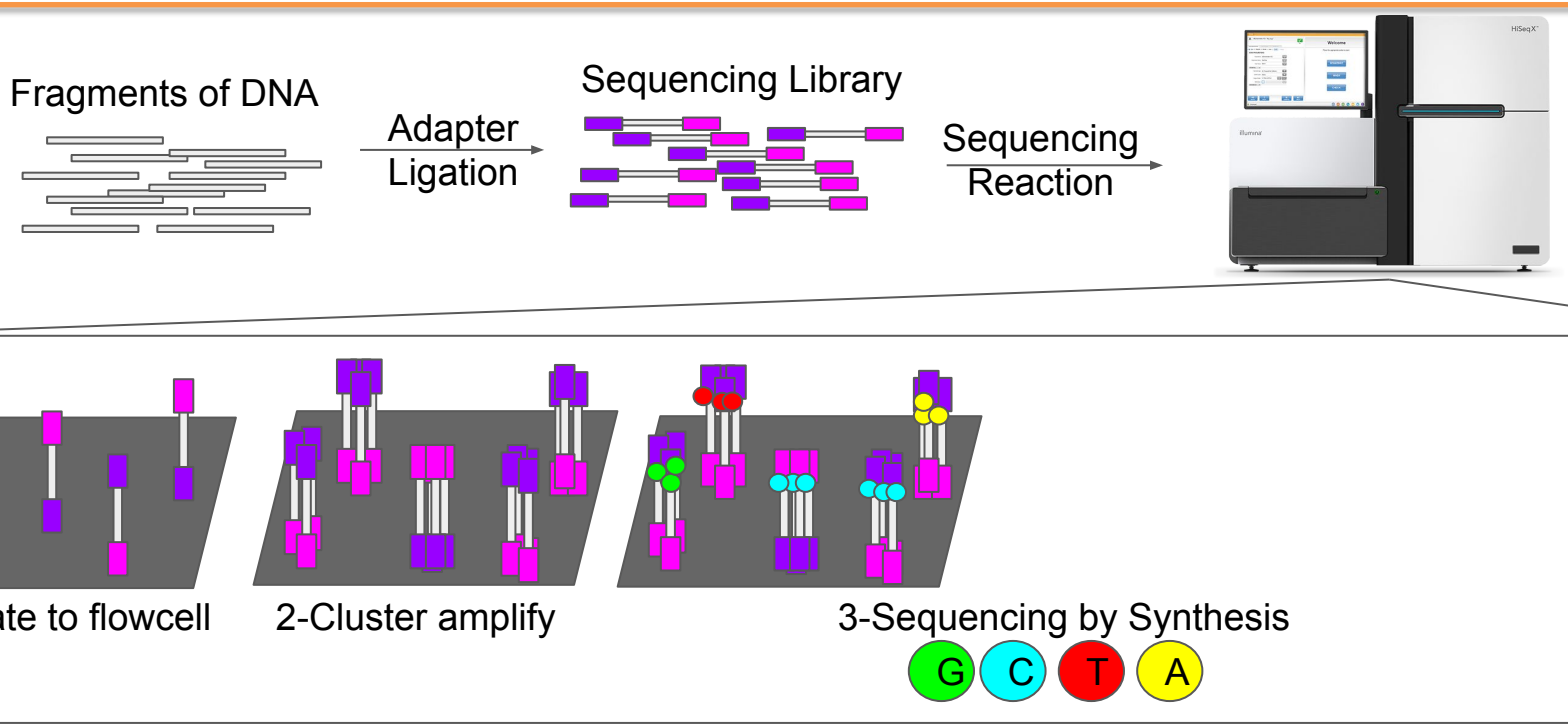




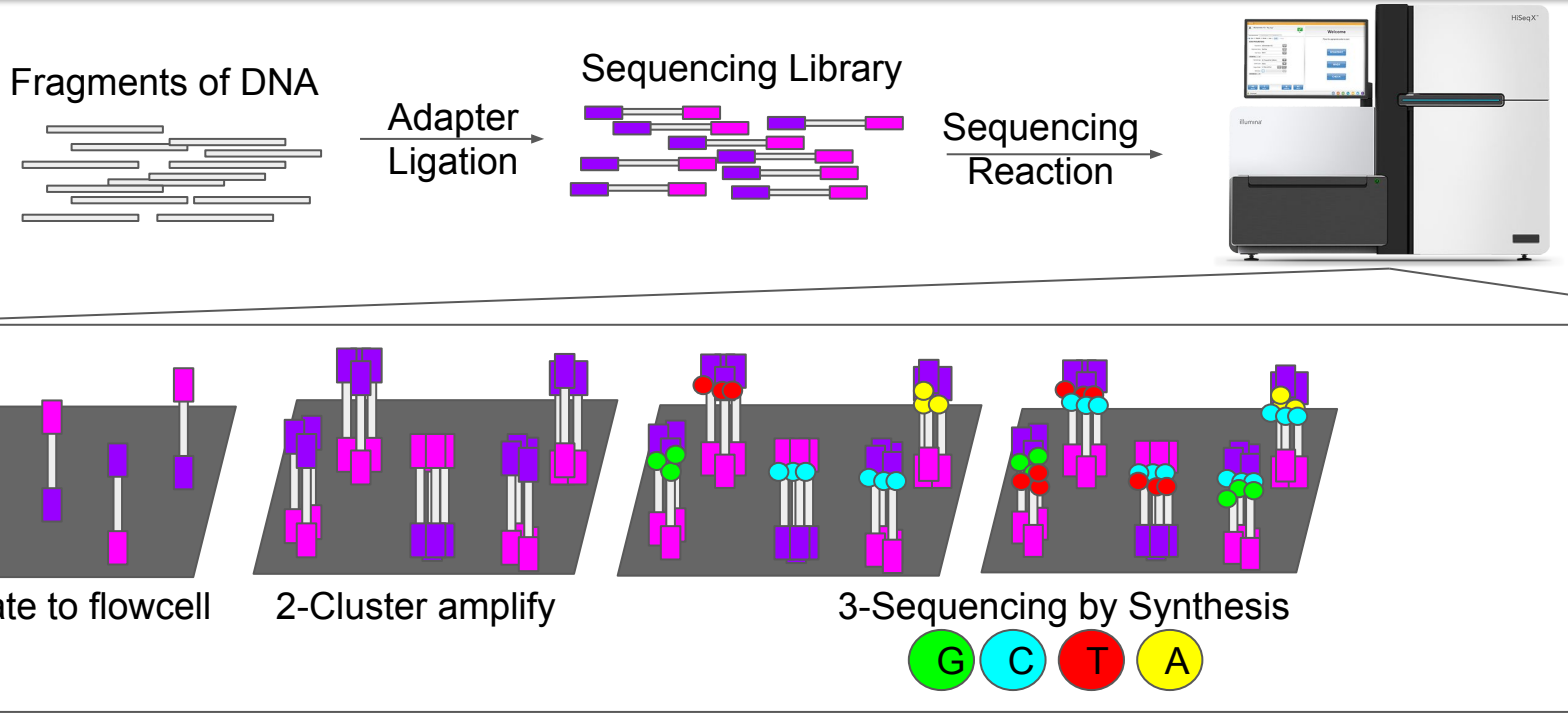
1-Ligate to flowcell

2-Cluster amplify

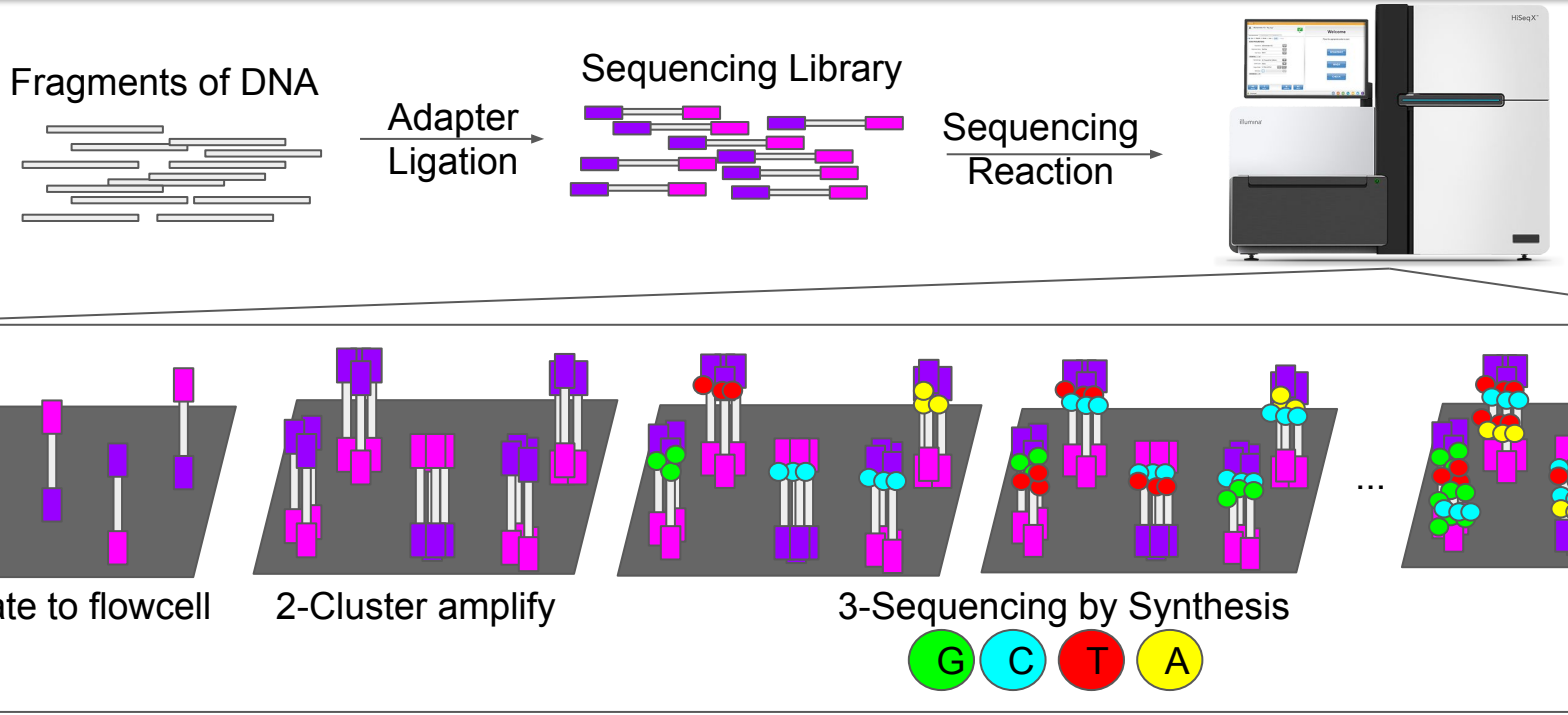
Next generation sequencing: Short-read sequencing



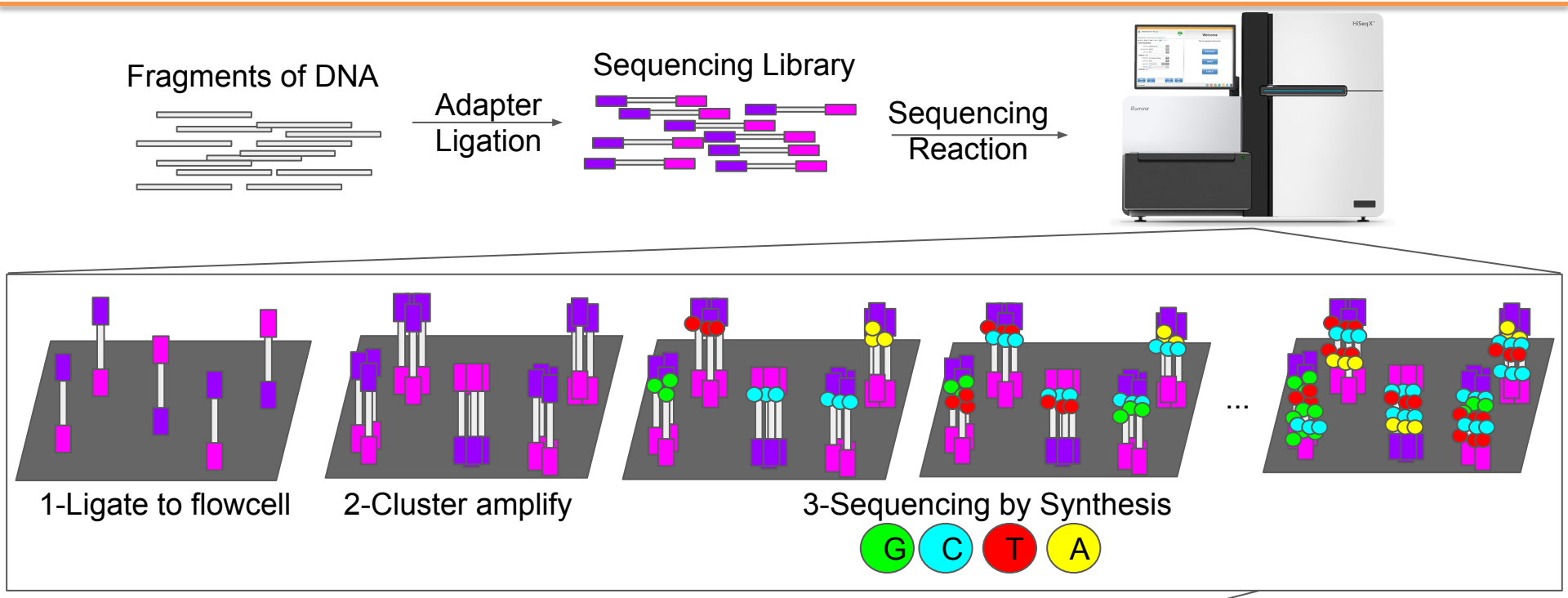
Next generation sequencing: Short-read sequencing



Next generation sequencing: Short-read sequencing



Next generation sequencing: Short-read sequencing



@Read1
TCTTGCGTACGTCTTCGATCGTA
+
!!@$@##@!%!@#$!!LLBBDKSNK

Convert to Fastq

WESTGRID

UBC

UBC100

Advanced Research Computing

Diverse Input Data, Same Output Format

- Different input data types still result in the same output data format
- Examples:
  - DNA-seq, ChIP-seq, RNA-seq, GRO-seq
- For non-DNA assays (e.g. RNA-seq/GRO-seq), they undergo a conversion from RNA-->cDNA before sequencing

EXAMPLE	MEANING
@K00171:617:HMMTNBBXX:1:1101:28686:1648 1:N:0:GACTAGTA TCTTGCGTACGTCTTCGATCGTA + !!@$@##@!%!@#$!!LLBBDKSNK	@Readname:And:Flowcell:Info 1 or 2 for read pair:N:0:Barcode Sequence "Plus Sign" ASCII-Quality Scores

Illumina machines use quality binning

```
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS.....................................................
..........................XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.....................
.....................................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII.....................
..................................JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ.....................
LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL.....................................................
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
|                        |    |        |                              |                     |
33                      59   64       73                            104                   126
 0........................26...31.......40
```

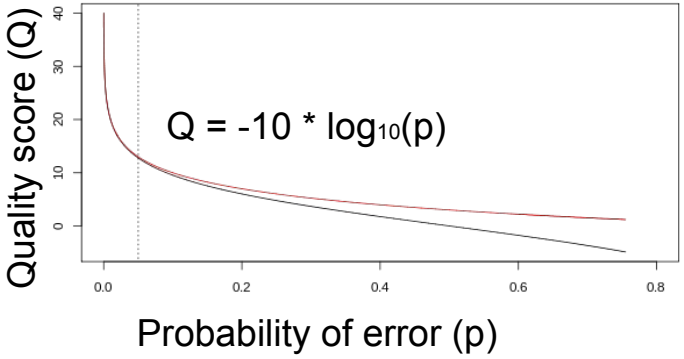


Table 2: Quality Score Bins for Optimized 8-Level Mapping

Q-Score Bins	Example of Empirically Mapped Q-Scores*
N (no call)	N (no call)
2–9	6
10–19	15
20–24	22
25–29	27
30–34	33
35–39	37
≥ 40	40

By replacing the quality scores between 19 and 25 with a new score of 22, data storage space can be conserved.

*The mapped quality score for each bin (except "N") is subject to change depending on individual Q-tables.

https://en.wikipedia.org/wiki/FASTQ_format

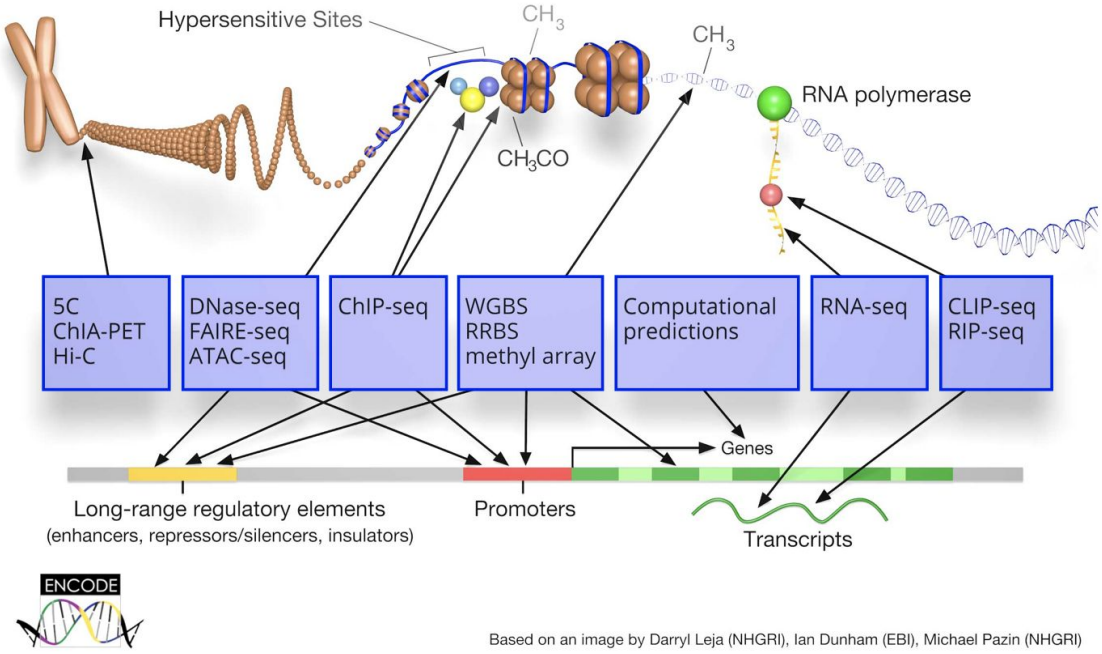
https://www.illumina.com/documents/products/technotes/techn
ote_understanding_quality_scores.pdf

Session Outline

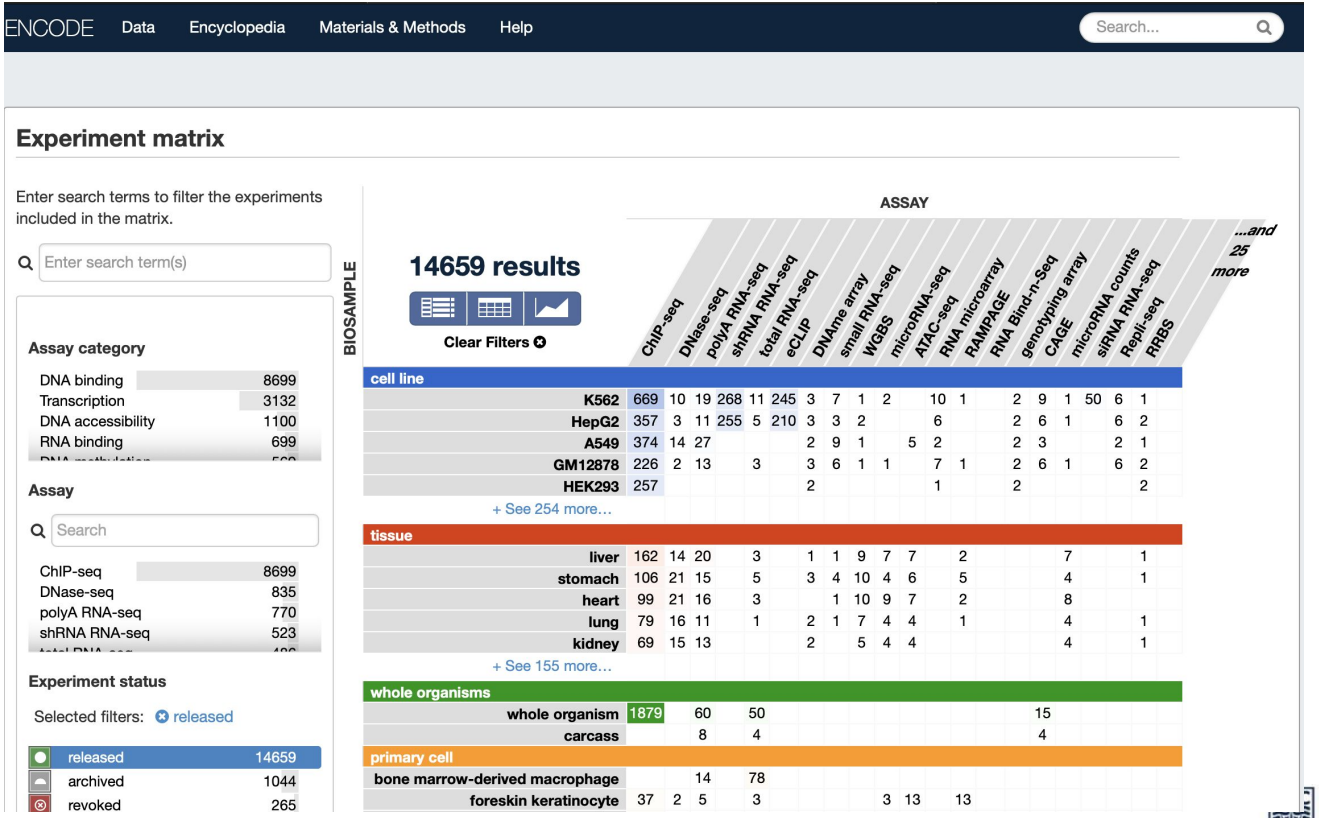
- Lecture 1
  - Introduction to next generation sequencing data & diverse data types
  - Transcriptional cis-regulatory datasets and where to find them
  - Accessing videos and support
- Video 1 - Initialize Workshop Directory and Explore Data
- Video 2 - Map and convert data using BWA mem and Samtools; call peaks with MACS2
- Lecture 2
  - Visualizing data and downstream analysis pipelines
- Video 3 - Visualize data using IGV
- Problem sets
- Closing remarks

ENCODE - Encyclopedia of DNA Elements

ENCODE is one of the many places to find open source data: www.encodeproject.org



You can download a diverse set of data across tissues/cell types



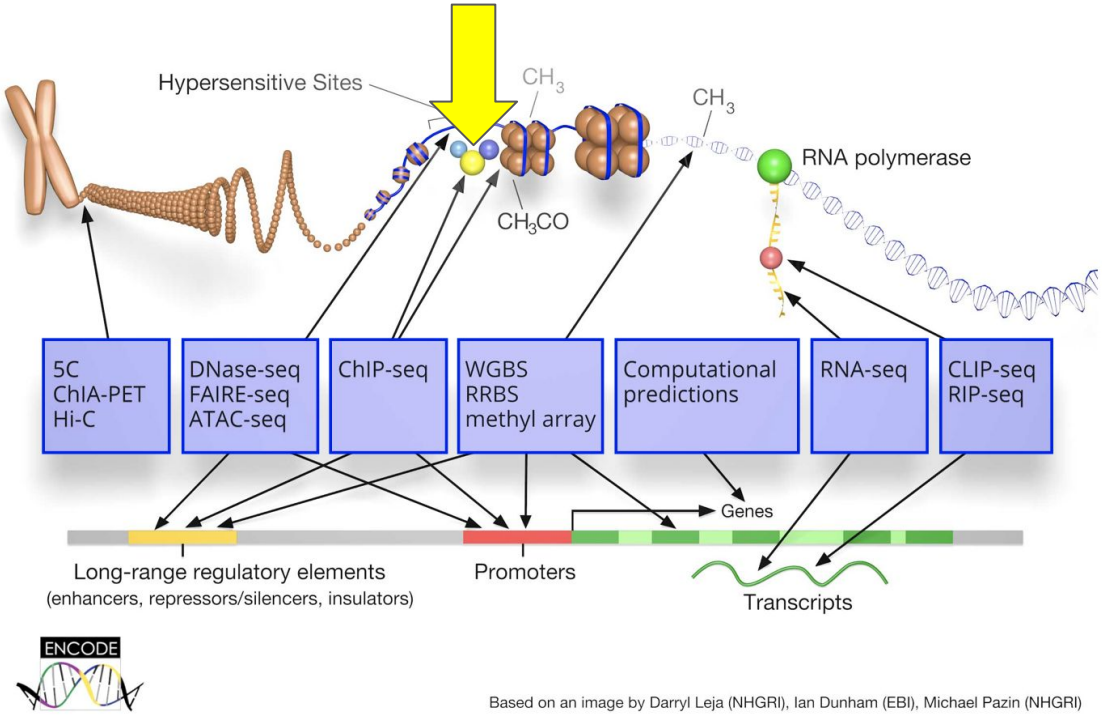
WESTGRID

UBC100

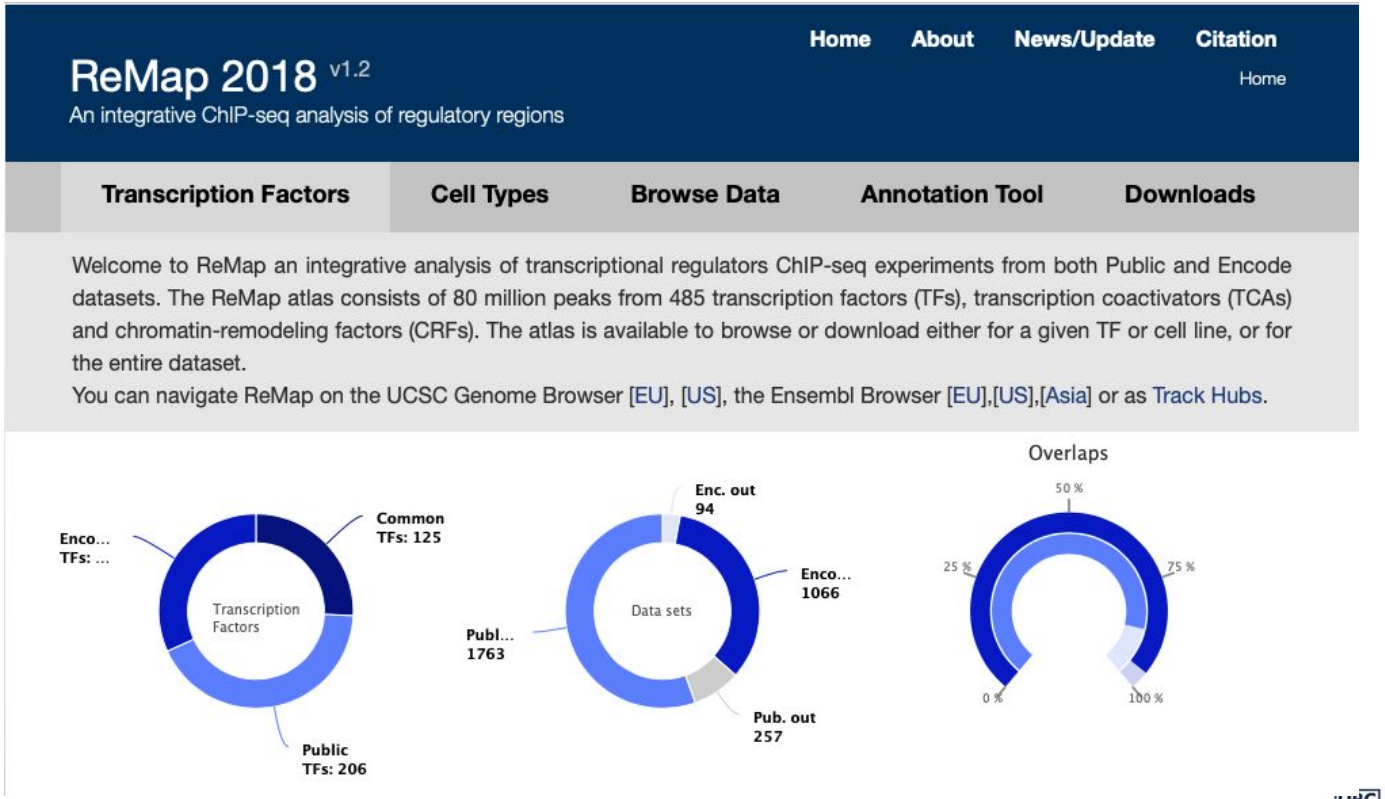
Advanced Research Computing

Transcription factor binding unified across sources

Remap - A resource for curated TF ChIP-seq data



Remap: http://pedagogix-tagc.univ-mrs.fr/remap/

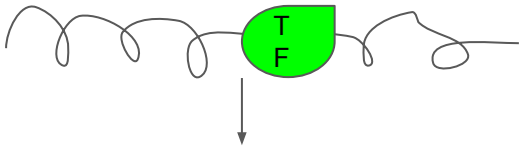


Chromatin Immunoprecipitation (ChIP-seq)

Chromatin Immunoprecipitation Sequencing (ChIP-seq) protocol:

Purpose: To find which sequences of DNA a specific protein interacts with, i.e Transcription Factor (TF).

1-Crosslink
DNA:Protein

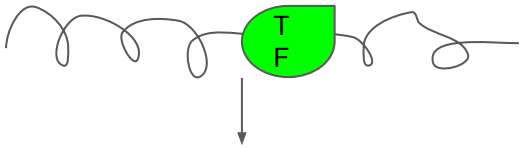


Chromatin Immunoprecipitation (ChIP-seq)

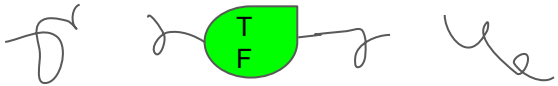
Chromatin Immunoprecipitation Sequencing (ChIP-seq) protocol:

Purpose: To find which sequences of DNA a specific protein interacts with, i.e Transcription Factor (TF).

1-Crosslink DNA:Protein



2-Shear

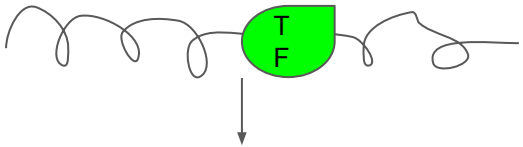


Chromatin Immunoprecipitation (ChIP-seq)

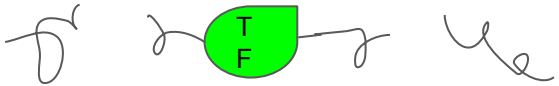
Chromatin Immunoprecipitation Sequencing (ChIP-seq) protocol:

Purpose: To find which sequences of DNA a specific protein interacts with, i.e Transcription Factor (TF).

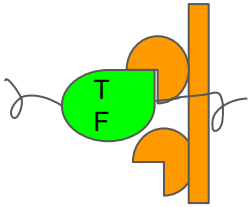
1-Crosslink DNA:Protein



2-Shear



3-Pull Down protein using anti-protein antibody on a column, wash away other DNA

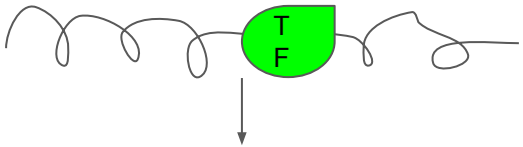


Chromatin Immunoprecipitation (ChIP-seq)

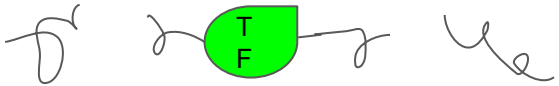
Chromatin Immunoprecipitation Sequencing (ChIP-seq) protocol:

Purpose: To find which sequences of DNA a specific protein interacts with, i.e Transcription Factor (TF).

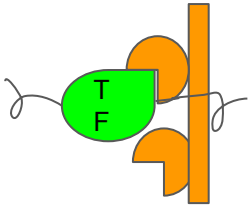
1-Crosslink DNA:Protein



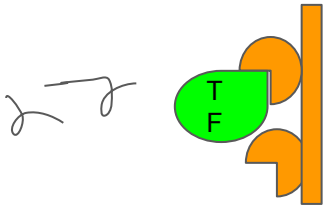
2-Shear



3-Pull Down protein using anti-protein antibody on a column, wash away other DNA



4-Reverse Crosslink

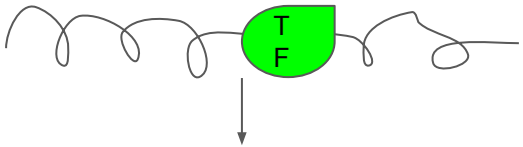


Chromatin Immunoprecipitation (ChIP-seq)

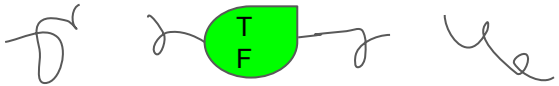
Chromatin Immunoprecipitation Sequencing (ChIP-seq) protocol:

Purpose: To find which sequences of DNA a specific protein interacts with, i.e Transcription Factor (TF).

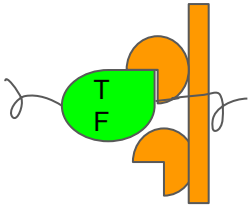
1-Crosslink DNA:Protein



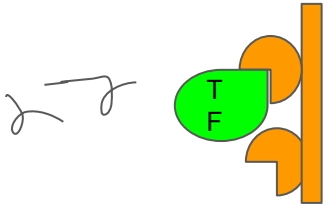
2-Shear



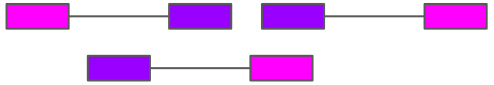
3-Pull Down protein using anti-protein antibody on a column, wash away other DNA



4-Reverse Crosslink



5-Ligate sequencing adapters



WESTGRID

UBC

UBC100

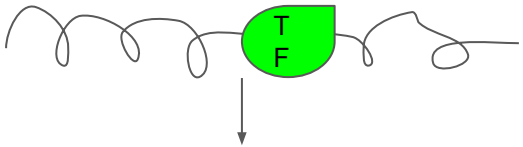
Advanced Research Computing

Chromatin Immunoprecipitation (ChIP-seq)

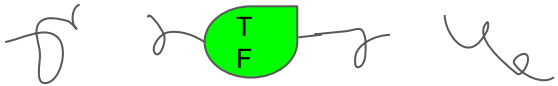
Chromatin Immunoprecipitation Sequencing (ChIP-seq) protocol:

Purpose: To find which sequences of DNA a specific protein interacts with, i.e Transcription Factor (TF).

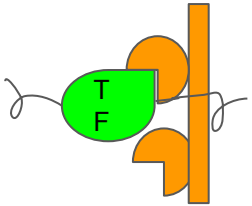
1-Crosslink DNA:Protein



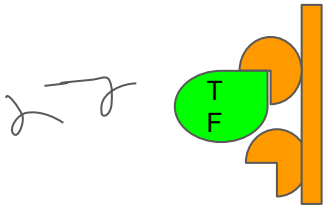
2-Shear



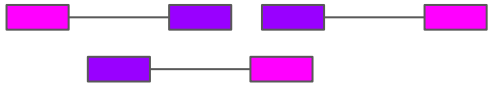
3-Pull Down protein using anti-protein antibody on a column, wash away other DNA



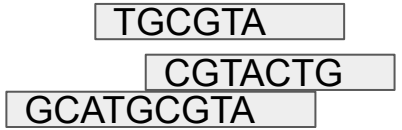
4-Reverse Crosslink



5-Ligate sequencing adapters



6-Sequence Library



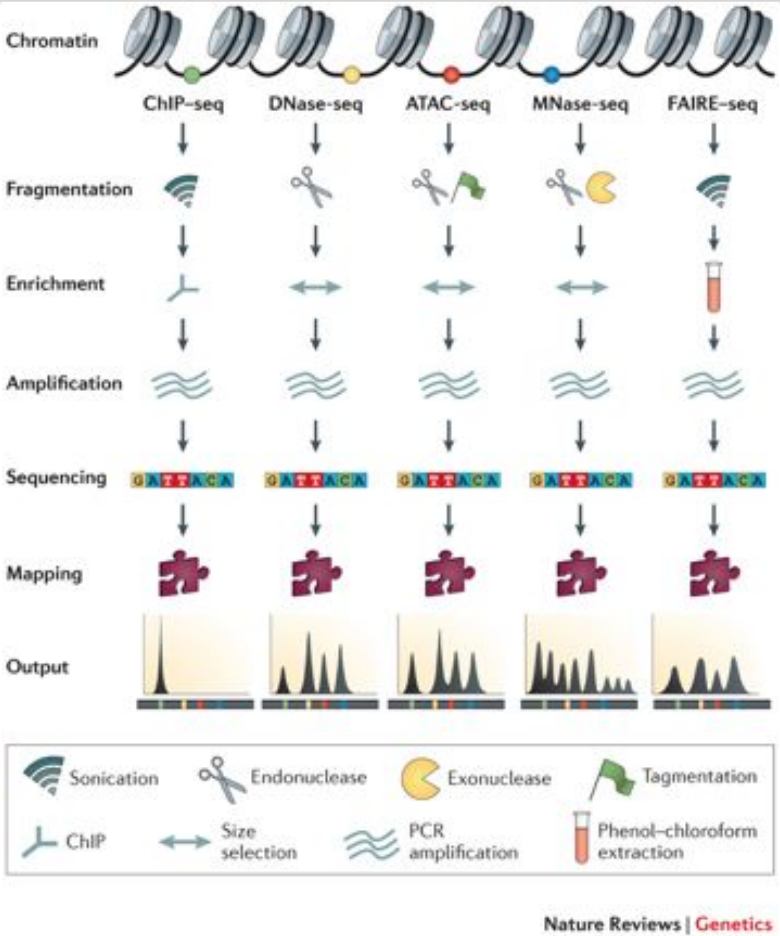
WESTGRID

UBC

UBC100

Advanced Research Computing

ATAC-seq represents open chromatin



Mapping data to a reference: ChIP-seq Peak Calling

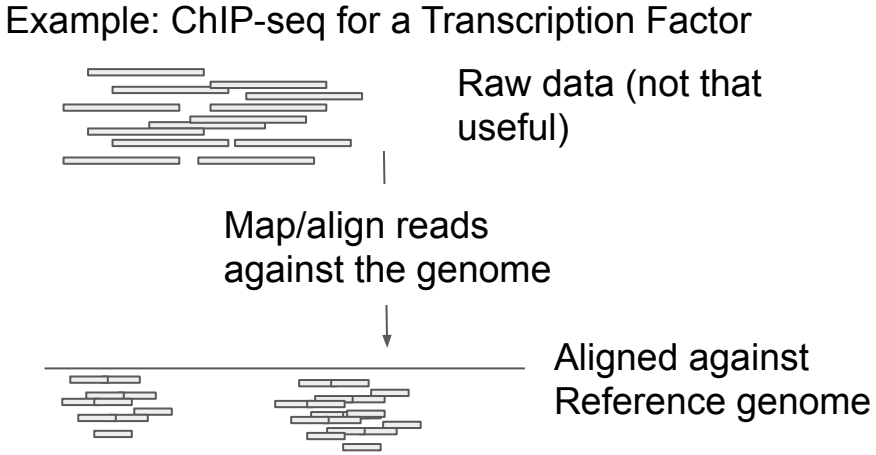
- Individually, the short sequencing reads do not have much information
- Collectively, they can represent something useful
- Analyzing short-read data takes two common forms:
  - **Reference-based mapping**
  - Assembly

Example: ChIP-seq for a Transcription Factor

Raw data (not that useful)

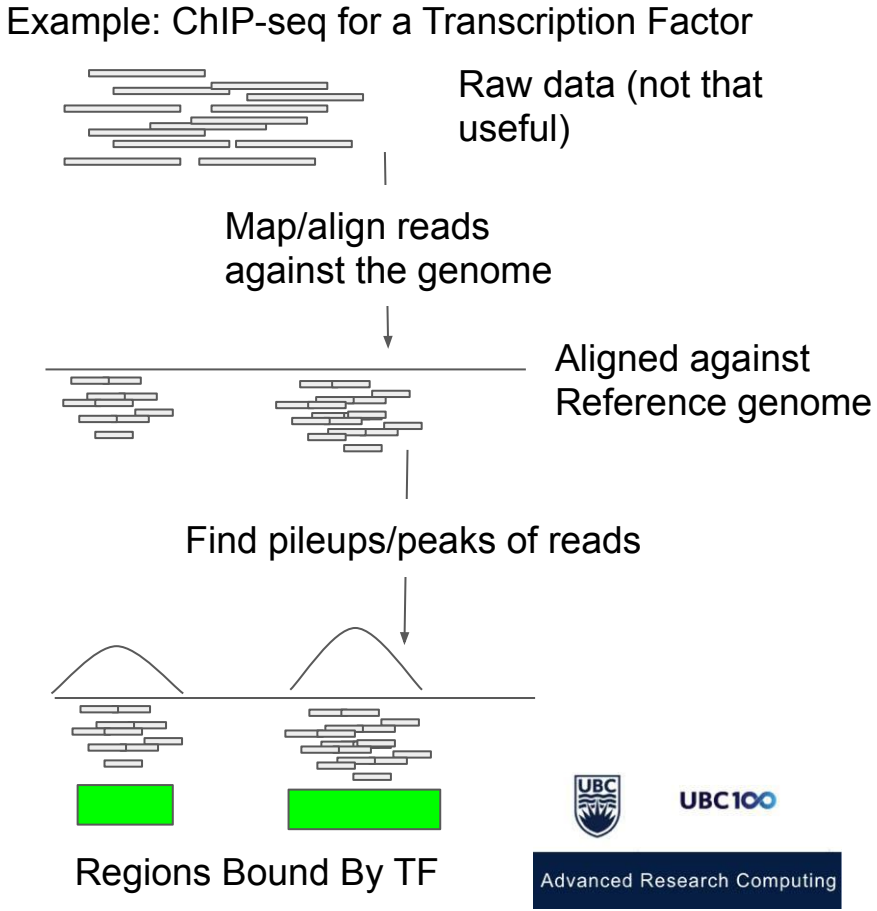
Mapping data to a reference: ChIP-seq Peak Calling

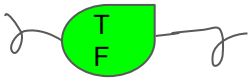
- Individually, the short sequencing reads do not have much information
- Collectively, they can represent something useful
- Analyzing short-read data takes two common forms:
  - **Reference-based mapping**
  - Assembly



Mapping data to a reference: ChIP-seq Peak Calling

- Individually, the short sequencing reads do not have much information
- Collectively, they can represent something useful
- Analyzing short-read data takes two common forms:
  - **Reference-based mapping**
  - Assembly





WESTGRID

UBC Advanced Research Computing

Session Outline

- Lecture 1
  - Introduction to next generation sequencing data & diverse data types
  - Transcriptional cis-regulatory datasets and where to find them
  - Accessing videos and support
- Video 1 - Initialize Workshop Directory and Explore Data
- Video 2 - Map and convert data using BWA mem and Samtools; call peaks with MACS2
- Mini Check-in
- Video 3 - Visualize data using IGV
- Problem sets
- Closing remarks

Accessing videos and support

Videos

- Video 1: https://youtu.be/CGvMm7JXQGs
- Video 2: https://youtu.be/x_uRLx1o9AM
- Video 3: https://youtu.be/UzCIGF4_OTg

Support:

Etherpad - https://etherpad.opendev.org/p/_B8ETduCObCQ6BN-811p

Session Outline

- Lecture 1
  - Introduction to next generation sequencing data & diverse data types
  - Transcriptional cis-regulatory datasets and where to find them
  - Accessing videos and support
- Video 1 - Initialize Workshop Directory and Explore Data
- Video 2 - Map and convert data using BWA mem and Samtools; call peaks with MACS2
- Mini Check-in
- Video 3 - Visualize data using IGV
- Problem sets
- Closing remarks

Video 1 content

1. Set up your workshop directory
   a. Create new directory inside: /scratch/richmonp/TRAINING/JUNE2020/
   b. Look at the tools directory: /scratch/richmonp/TRAINING/JUNE2020/TOOLS/
2. Explore files
   a. Reference genome files: fasta
   b. Raw data files: fastq
   c. Processed data files

Let's get started! Login to Cedar

You should have already attempted this by now, but as a reminder:

1. Open up a terminal (PC: MobaXterm, Putty | Mac/Linux: Terminal)
2. Login to Cedar

Command (login):

$ ssh <username>@cedar.computecanada.ca

$ ssh richmonp@cedar.computecanada.ca

NOTE: Whenever you see me represent something with the <>, I want you to replace it with what applies to you. Also, whenever there is a "$", I am showing you a command. Commands will be highlighted, with the format in yellow, and the actual example in green

We have a reservation for this course

--account=def-training-wa_cpu --reservation=wgss1-wr_cpu

This will get you access to the reservation, meaning your jobs won't wait in the queue. The reservation is active until 5PM today, June 17th.

Put it at the top of your job scripts, or when using salloc

Orienting yourself to this workshop directory

The workshop directory is located here:
/scratch/richmonp/TRAINING/JUNE2020/

Change into that directory:
$ cd /scratch/richmonp/TRAINING/JUNE2020/

Important subdirectories:
/scratch/richmonp/TRAINING/JUNE2020/Files/SCRIPTS/ -
Has scripts & templates that you can copy/use
/scratch/richmonp/TRAINING/JUNE2020/Files/RAW_DATA/ -
Has the some raw data that we can use today for analysis
/scratch/richmonp/TRAINING/JUNE2020/Files/PROCESS/ -
If nothing works for you today, these are some processed files that you can look at/visualize
/scratch/richmonp/TRAINING/JUNE2020/ -
This is where your own workshop directory will exist, and you have permission over it

Set up a workshop directory

```
$ mkdir <directory>
$ mkdir /scratch/richmonp/TRAINING/JUNE2020/SHERLOCK/
```

NOTE: If you need help, you will need to share permissions on your directory:

```
$ chmod ugo=rwx -R <directory>
$ chmod ugo=rwx -R /scratch/richmonp/TRAINING/JUNE2020/SHERLOCK/
```

For additional information about permissions and other common command-line functions see me during the problemset.

All you need is scripts

/scratch/richmonp/TRAINING/JUNE2020/Files/SCRIPTS/ has 3 scripts inside it:

H3K27Ac_Workshop.sh

POLR2A_Workshop.sh

ATAC-Seq_Workshop.sh

I'm going to copy these so I can play with them:

$ cp /scratch/richmonp/TRAINING/JUNE2020/Files/SCRIPTS/*sh SHERLOCK/

Breakdown of the script: Welcome to the mellow yellow

```
#!/bin/bash

#SBATCH --account=def-training-wa_cpu --reservation=wgss1-wr_cpu

## Mail Options
#SBATCH --mail-user=<yourEmailAddress>
#SBATCH --mail-type=ALL

## CPU Usage
#SBATCH --mem-per-cpu=4G
#SBATCH --cpus-per-task=8
#SBATCH --time=3:00:00
#SBATCH --nodes=1

## Output and Stderr
#SBATCH --output=%x-%j.out
#SBATCH --error=%x-%j.error
```

This header information contains info about the account to bill for these hours, I want it to mail me, how much RAM and CPUs I need over a single node, and where to send standard error and output

Breakdown of the script: Welcome to the mellow yellow

```
#!/bin/bash

#SBATCH --account=def-training-wa_cpu --reservation=wgss1-wr_cpu

## Mail Options
#SBATCH --mail-user=<yourEmailAddress>
#SBATCH --mail-type=ALL

## CPU Usage
#SBATCH --mem-per-cpu=4G
#SBATCH --cpus-per-task=8
#SBATCH --time=3:00:00
#SBATCH --nodes=1

## Output and Stderr
#SBATCH --output=%x-%j.out
#SBATCH --error=%x-%j.error
```

You will need to change this to be relevant to your own use case

This header information contains info about the account to bill for these hours, I want it to mail me, how much RAM and CPUs I need over a single node, and where to send standard error and output

Load my necessary tools

```
# Requirements
module load bwa/0.7.15
module load samtools/1.9

# Conda activate
source /scratch/richmonp/TRAINING/JUNE2020/TOOLS/opt/miniconda3/etc/profile.d/conda.sh
conda activate /scratch/richmonp/TRAINING/JUNE2020/TOOLS/opt/SummerSchool/
```

I'm also going to load the necessary modules (always try to keep version information with modules).

I have also installed a tool via conda, which is not available on the Cedar software stack. You can activate the environment which houses the tool using these two commands.

Details on how that install was possible can be found in the TOOLS/ subdirectory.

You're going to need a reference genome next

```
# Genome
GENOME="/cvmfs/ref.mugqic/genomes/species/Homo_sapiens.GRCh38/genome/Homo_sapiens.GRCh38.fa"
INDEX="/cvmfs/ref.mugqic/genomes/species/Homo_sapiens.GRCh38/genome/bwa_index/Homo_sapiens.GRCh38.fa"
```

Next, I specify the genome I want to use to map my data against. I realize you won't all work in human, but if you work in a model organism I recommend checking out this repository for genomes:

/cvmfs/ref.mugqic/genomes/

Here, I'm using their **BWA index**, and their **Fasta file**

Reference Genome, Fasta file format

Reference genomes are packaged into fasta files.

Format:
>chromosome1_Name OtherChromInfo AccessionInfo Etc.
NNNNNNATTCGTTGATGGATAGCATGATCAGTAGACATGACATGACAGATGAGGGATATGATGACCACCACC
CAGATTCCCGGCCGGCCGGCCGGCCCGGGCCGGCCGGCCGGGCCCGGCTATATATATATACATAG ….
>chromosome2_Name OtherChromInfo AccessionInfo Etc.
NNNNNNNCCCCGGCCGGCCGGCCGGCCCGGGCCGGCCGGCCGGGCCCGGCTATATATATATACATAGATG
ATCAGTAGACATGACATGACAGATGAGGGATATGATGACCACCACCCAGATTGGAGTTGCCAGAT

We need to “index” this genome in order to map to it. There are many different genome indexing strategies. For bwa, we use the command bwa index, which creates an FM-Index of the genome.
$ bwa index <in.fasta>
This will generate these files:
genome.fa.amb, genome.fa.ann, genome.fa.bwt, genome.fa.pac, genome.fa.sa

Since it’s done for us, we don’t have to redo this step.

Set some more variables

```
# Globals
NAME="<YOURNAMEHERE>"
ID="POLR2A"
WORKDIR=/scratch/richmonp/TRAINING/JUNE2020/$NAME/$ID
PLATFORM="Illumina"
SAMPLE="heart_left_ventricle"
THREADS=8
mkdir -p $WORKDIR
cd $WORKDIR

# Files
FASTQ_R1=$WORKDIR/$SAMPLE.$ID.R1.fastq.gz
FASTQ_R2=$WORKDIR/$SAMPLE.$ID.R2.fastq.gz
SAM_FILE=$WORKDIR/$SAMPLE.$ID.sam
BAM_FILE=$WORKDIR/$SAMPLE.$ID.bam
MACS2_DIR=$WORKDIR/MACS2
PEAKS_FILE=$MACS2_DIR/${SAMPLE}.${ID}.peaks.narrowPeak
CONTROL=$WORKDIR/${SAMPLE}.$ID.Control.bam
BIGWIG_FILE=$WORKDIR/$ID.$SAMPLE.bw
```

You will need to change this directory to be relevant to your own use case. E.g. change to “SHERLOCK”, keeping the “”

I’m setting a sample identifier, my own name, a working directory, and the threads I’m using (just setting equal to what we set above). I then create the working directory and change into it.

I HIGHLY RECOMMEND using variables like this within your scripts. It will make it possible to easily change out a single variable or path, and the script can remain functional

Set some more variables

```
# Globals
NAME="<YOURNAMEHERE>"
ID="POLR2A"
WORKDIR=/scratch/richmonp/TRAINING/JUNE2020/$NAME/$ID
PLATFORM="Illumina"
SAMPLE="heart_left_ventricle"
THREADS=8
mkdir -p $WORKDIR
cd $WORKDIR

# Files
FASTQ_R1=$WORKDIR/$SAMPLE.$ID.R1.fastq.gz
FASTQ_R2=$WORKDIR/$SAMPLE.$ID.R2.fastq.gz
SAM_FILE=$WORKDIR/$SAMPLE.$ID.sam
BAM_FILE=$WORKDIR/$SAMPLE.$ID.bam
MACS2_DIR=$WORKDIR/MACS2
PEAKS_FILE=$MACS2_DIR/${SAMPLE}.${ID}.peaks.narrowPeak
CONTROL=$WORKDIR/${SAMPLE}.$ID.Control.bam
BIGWIG_FILE=$WORKDIR/$ID.$SAMPLE.bw
```

These files will be the raw data files (renamed), the output SAM, and BAM files, (ignore the MACS2 + PEAKS/CONTROL stuff), and a BigWig file.

I'm setting a sample identifier, my own name, a working directory, and the threads I'm using (just setting equal to what we set above). I then create the working directory and change into it.

I HIGHLY RECOMMEND using variables like this within your scripts. It will make it possible to easily change out a single variable or path, and the script can remain functional

Set even more variables, and download some data

```
# Download ENCODE data
wget https://www.encodeproject.org/files/ENCFF999VOH/@@download/ENCFF999VOH.fastq.gz
wget https://www.encodeproject.org/files/ENCFF435KUB/@@download/ENCFF435KUB.fastq.gz
wget https://www.encodeproject.org/files/ENCFF925AMH/@@download/ENCFF925AMH.bam
mv $WORKDIR/ENCFF999VOH.fastq.gz $FASTQ_R1
mv $WORKDIR/ENCFF435KUB.fastq.gz $FASTQ_R2
mv $WORKDIR/ENCFF925AMH.bam $CONTROL
```

Here I download the data from ENCODE, using links I retrieved from navigating the website.

Then I rename files so they fit my convention (instead of ENCFF999VOH, etc.).

If you want to explore lots of these datasets to download, use the https://www.encodeproject.org website (Hint: this is a part of Problem set 1).

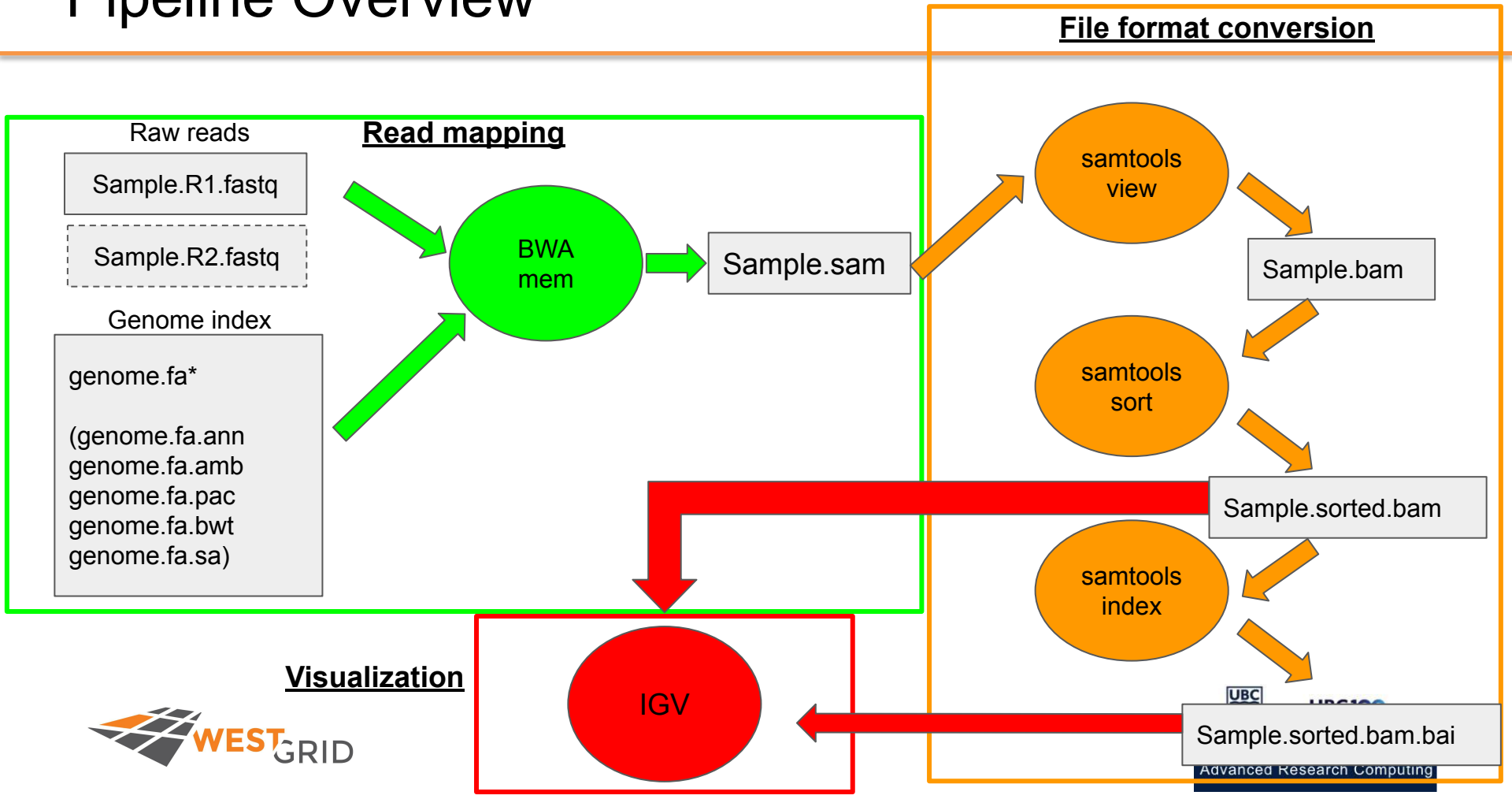
Session Outline

- Lecture 1
  - Introduction to next generation sequencing data & diverse data types
  - Transcriptional cis-regulatory datasets and where to find them
  - Accessing videos and support
- Video 1 - Initialize Workshop Directory and Explore Data
- Video 2 - Map and convert data using BWA mem and Samtools; generate bigWigs with deeptools
- Mini Check-in
- Video 3 - Visualize data using IGV
- Problem sets
- Closing remarks

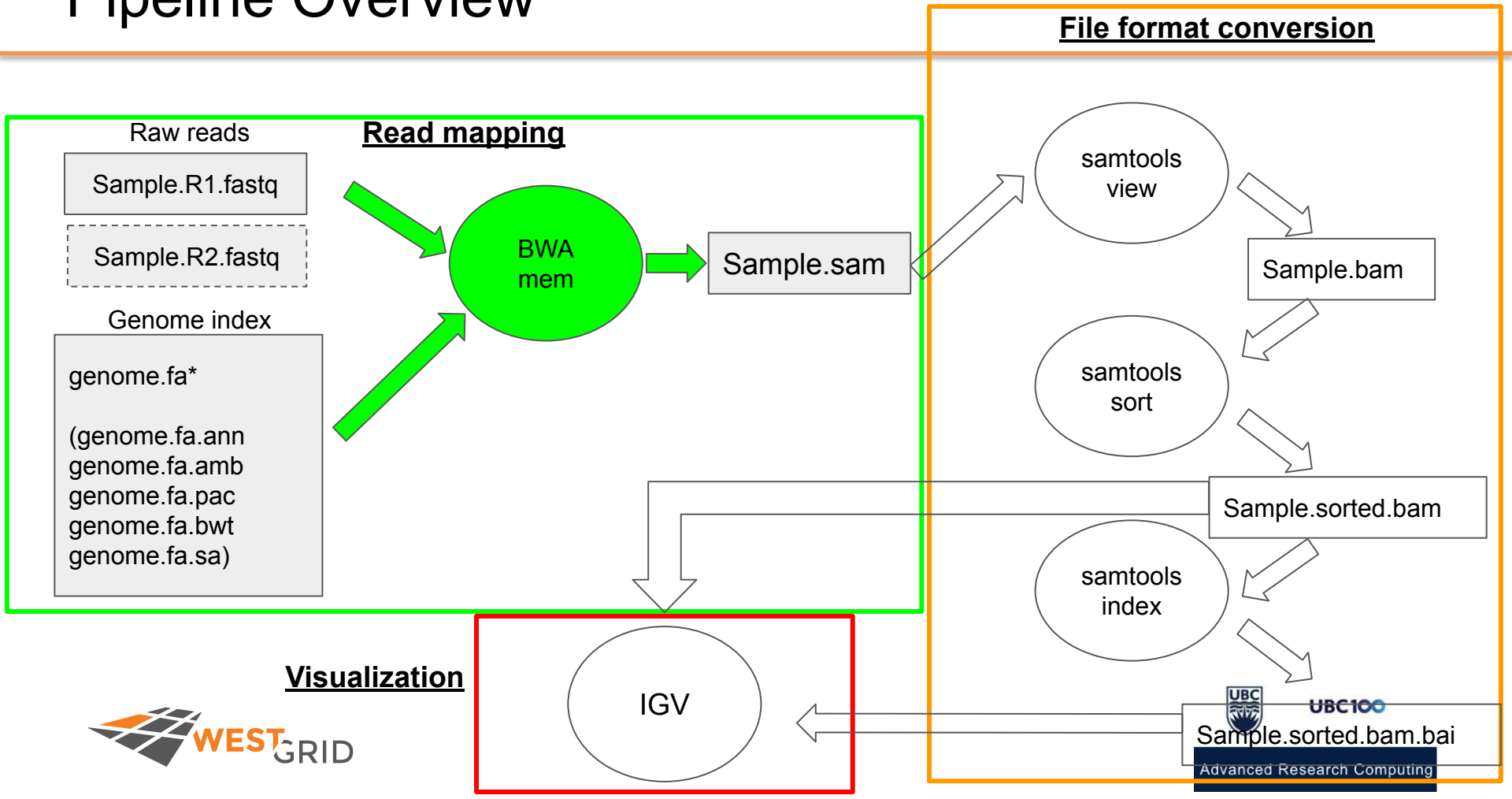
Video 2 content

1. Overview of basic read-mapping pipeline
2. Software usage
   a. BWA mem
   b. Samtools
   c. DeepTools
3. Edit scripts and submit to scheduler
4. Examine output files (from pre-analyzed data)

Pipeline Overview



Pipeline Overview



WESTGRID

UBC Advanced Research Computing

Learning the bwa mem command

First we need to load the module that has the bwa command in it

```
$ module load bwa/0.7.15
```

Next we will call the bwa mem command to see how it's used

```
$ bwa mem
```

Let's break down this usage statement:

```
$ bwa mem [options]  <idxbase>  <in1.fq> [in2.fq]
```

[ ] is an optional argument, <> is required and is asking you to replace what's inside with the appropriate value

Example (From your workshop directory):

```
$ bwa  mem
/cvmfs/ref.mugqic/genomes/species/Homo_sapiens.GRCh38/genome/bwa_index/Homo_sapiens.GRCh38.fa
Sample1_R1.fastq  Sample1_R2.fastq  > Sample1.sam
```

Mapping Reads to the Genome within our script

```
# Map reads to genome
if [ ! -f $SAM_FILE ]; then
        # bwa mem [options] <idxbase> <in1.fq> [in2.fq]
        bwa mem -t $THREADS -M \
        -R "@RG\tID:$ID\tSM:$SAMPLE\tPL:$PLATFORM" \
        $INDEX $FASTQ_R1 $FASTQ_R2 > $SAM_FILE
fi
```

The little if/fi statements are to check if the output file exists, and if it does not exist, then perform the little command inside the block.

The BWA mem command is in the block, and at a minimum it needs an indexed genome, and an input fastq. I also add options -t for multithreading (using more cores), -R for a readgroup identifier (required for many tools), and -M for mapping split/secondary hits (not always needed). I also capture the standard out and place it into a SAM file.

The output SAM file

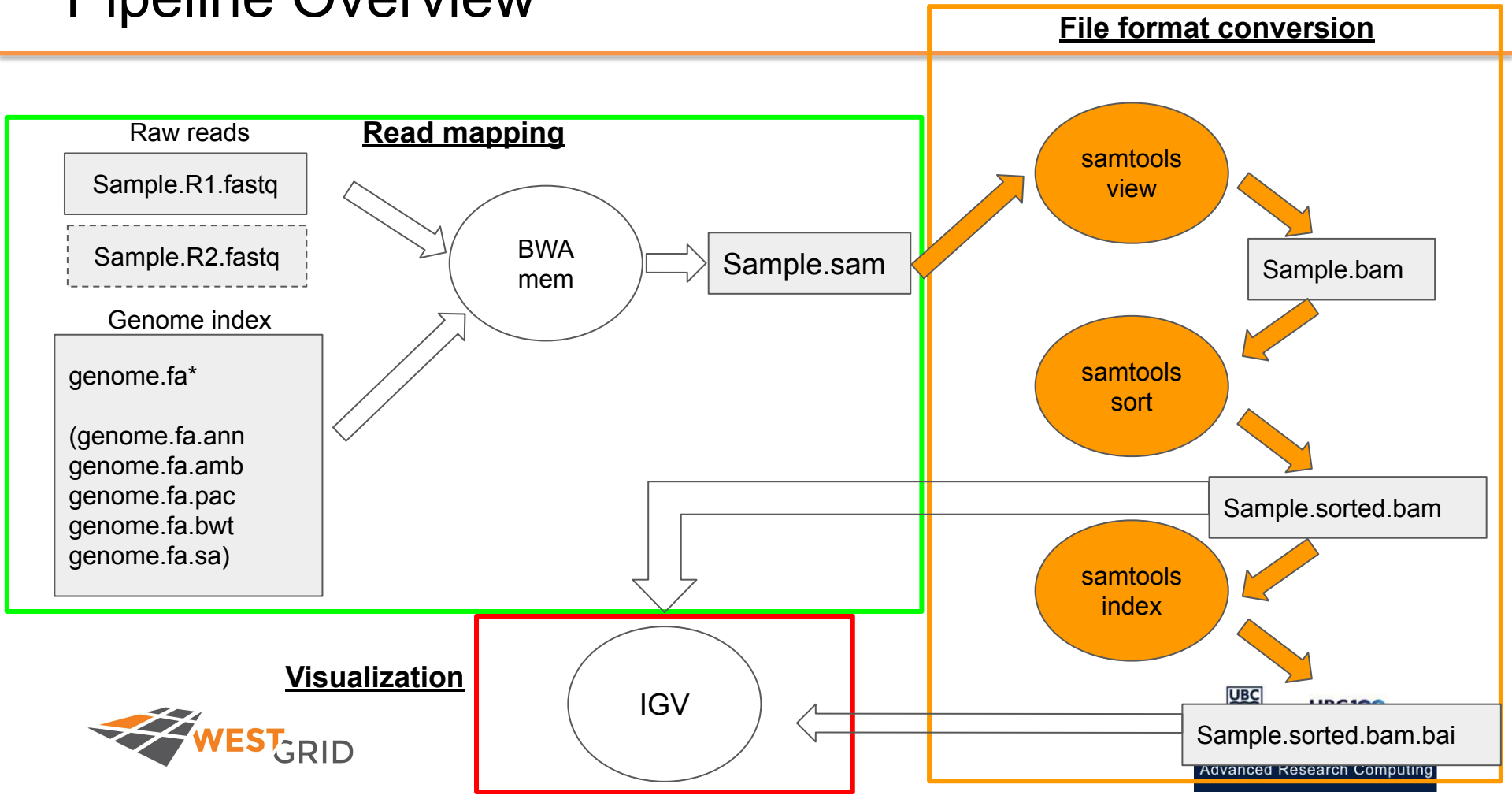
@SQ - Sequence (contig/chromosome) from reference file
@PG - Program information about mapping
@RG - Read group information (we won't have any here)

Tab delimited, each line is 1 read. Pairs will be next to each other in the file (e.g.
Line1: Read1
Line2: Read2

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,254}`	Query template NAME
2	FLAG	Int	$[0,2^{16}-1]$	bitwise FLAG
3	RNAME	String	`\*|[!-()+-<>-~][!-~]*`	Reference sequence NAME
4	POS	Int	$[0,2^{31}-1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0,2^{8}-1]$	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[!-()+-<>-~][!-~]*`	Ref. name of the mate/next read
8	PNEXT	Int	$[0,2^{31}-1]$	Position of the mate/next read
9	TLEN	Int	$[-2^{31}+1,2^{31}-1]$	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	segment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+33

https://**sam**tools.github.io/hts-specs/**SAM**v1.pdf

Pipeline Overview



Then we convert, sort, and index the bam file

```
# Convert SAM to sorted BAM
# create sorted BAM
if [ ! -f $BAM_FILE ]; then
        samtools view -Shu --threads `expr $THREADS - 1` $SAM_FILE | \
        samtools sort --threads `expr $THREADS - 1` -o $BAM_FILE -
        samtools index $BAM_FILE
fi
```

Here, I'm using the | to skip the step of saving the bam file, and then sorting it.

I link the two commands together to first convert the sam into bam using samtools view, and then sorting it using samtools sort.

I also add a multi-threading option, but samtools asks for "additional threads" so I take my thread# - 1.

The index command will create a .bai file next to the .bam file (file.bam.bai), which is needed for downstream tools

An easier version of samtools can be found here

```
$ module load samtools/1.9
```

We will use 3 samtools operations: view, sort, and index (in that order)

```
$ samtools   view  -b  <in.sam>   -o  <out.bam>
$ samtools   view  -b  Sample1.sam   -o  Sample1.bam

$ samtools   sort   <in.bam>  -o <out.sorted.bam>
$ samtools   sort   Sample1.bam  -o  Sample1.sorted.bam

$ samtools   index  <in.sorted.bam>
$ samtools   index   Sample1.sorted.bam
```

Last step, create bigWig from BAM file for coverage visualization

```
# BAM to BigWig
if [ ! -f $BIGWIG_FILE ]; then
        bamCoverage -b $BAM_FILE -of bigwig -p $THREADS -o $BIGWIG_FILE
fi
```

This is a little bonus step beyond typical mapping and conversion, which allows us to visualize a histogram of coverage within IGV.

bamCoverage is a part of the DeepTools package.

Output files from the entire script

```
[richmonp@cedar1 POLR2A]$ ls -lahtr
total 43G
drwxr-x--- 6 richmonp richmonp 4.0K Jun 16 12:43 ..
-rw-r----- 1 richmonp richmonp 2.6G Jun 16 12:44 heart_left_ventricle.POLR2A.R1.fastq.gz
-rw-r----- 1 richmonp richmonp 2.8G Jun 16 12:44 heart_left_ventricle.POLR2A.R2.fastq.gz
-rw-r----- 1 richmonp richmonp 4.0G Jun 16 12:45 heart_left_ventricle.POLR2A.Control.bam
-rw-r----- 1 richmonp richmonp  28G Jun 16 12:48 heart_left_ventricle.POLR2A.sam
-rw-r----- 1 richmonp richmonp 4.8G Jun 16 12:48 heart_left_ventricle.POLR2A.bam
-rw-r----- 1 richmonp richmonp 2.5M Jun 16 12:49 heart_left_ventricle.POLR2A.bam.bai
drwxr-x--- 2 richmonp richmonp 4.0K Jun 16 12:49 .
-rw-r----- 1 richmonp richmonp 181M Jun 16 12:49 POLR2A.heart_left_ventricle.bw
```

When the script finishes, this will be the final output.

We won't be using the Control.BAM for any analysis, but this is an important file to have for ChIP-seq experiments since it represents the expected background/noise, and can be subtracted out using tools like MACS2 when doing peak calling (not covered today).

Problem Set 1

1. Edit the scripts for processing data from the heart left ventricle for POLR2A, H3K27Ac, and ATAC-Seq.
2. Submit the jobs and ensure they can run.
   a. NOTE: processing this data takes ~1.5 hours using 8 cores. So start it, make sure it runs without errors, and continue working.
3. Go to www.encodeproject.org, find an ATAC-seq dataset in human, create a SLURM script to download + process the data
   a. Should just be repurposing the script we just used above.
4. (Optional) From www.encodeproject.org, find an ATAC-seq dataset in mouse (*Mus musculus*), download it, and map it against the mouse genome.
   a. Hint, mouse genome BWA index:
   /cvmfs/ref.mugqic/genomes/species/Mus_musculus.GRCm38/genome/bwa_index/Mus_musculus.GRCm38.fa

Session Outline

- Lecture 1
  - Introduction to next generation sequencing data & diverse data types
  - Transcriptional cis-regulatory datasets and where to find them
  - Accessing videos and support
- Video 1 - Initialize Workshop Directory and Explore Data
- Video 2 - Map and convert data using BWA mem and Samtools; generate bigWigs with deeptools
- Mini Check-in
- Video 3 - Visualize data using IGV
- Problem sets
- Closing remarks

Checkin in...

We will briefly reconvene now. Aiming at 11:00-11:15 for this check-in.

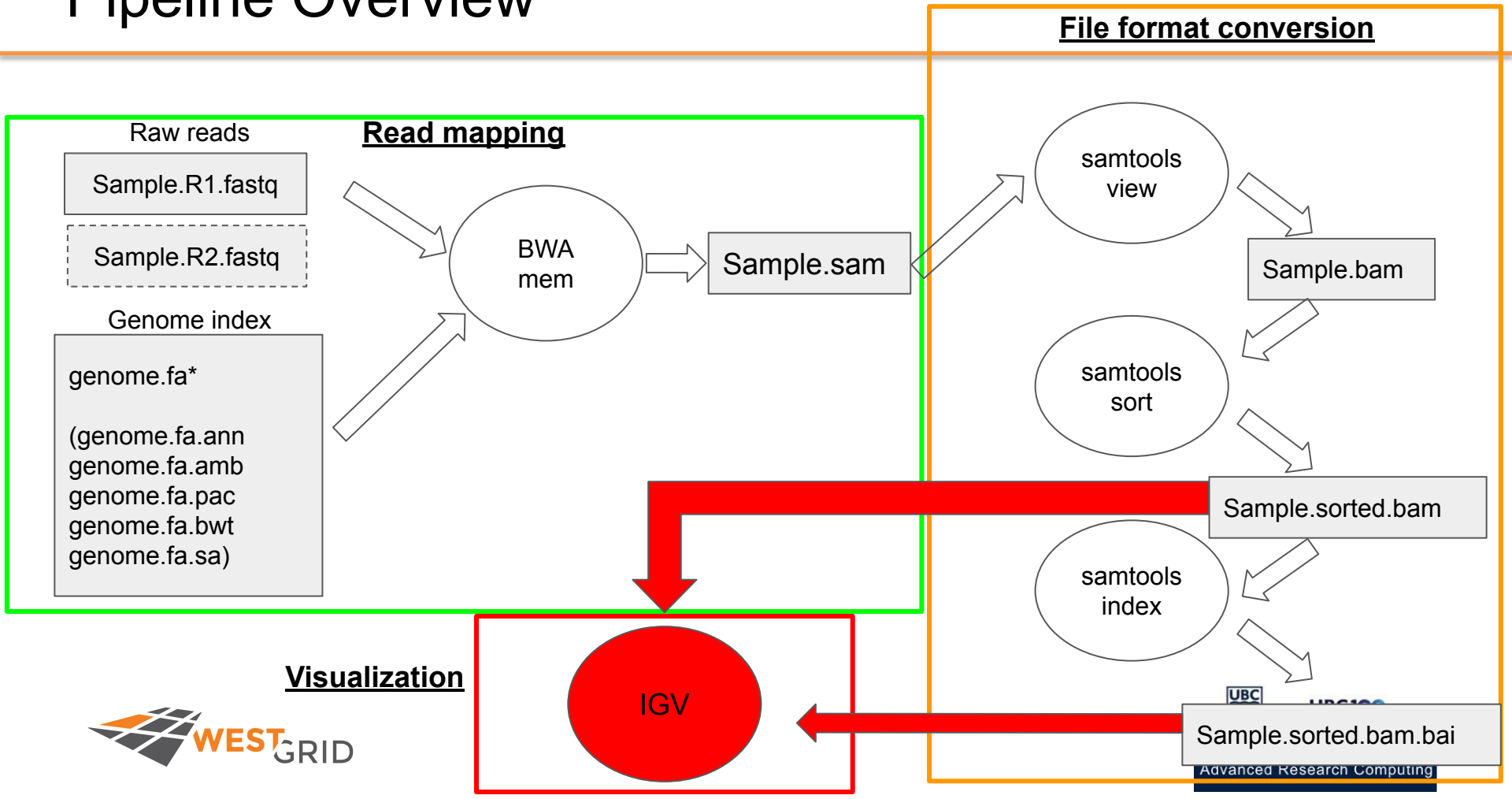
Session Outline

- Lecture 1
  - Introduction to next generation sequencing data & diverse data types
  - Transcriptional cis-regulatory datasets and where to find them
  - Accessing videos and support
- Video 1 - Initialize Workshop Directory and Explore Data
- Video 2 - Map and convert data using BWA mem and Samtools; generate bigWigs with deeptools
- Mini Check-in
- Video 3 - Visualize data using IGV
- Problem sets
- Closing remarks

Video 3

1. Transfer files to your own computer OR mount via sshfs
   a. Transfer options: FileZilla, WinSCP, scp on linux/mac
2. Download and open IGV
3. Load data in IGV
4. Zoom into a region and take a snapshot

Pipeline Overview



WESTGRID

Advanced Research Computing

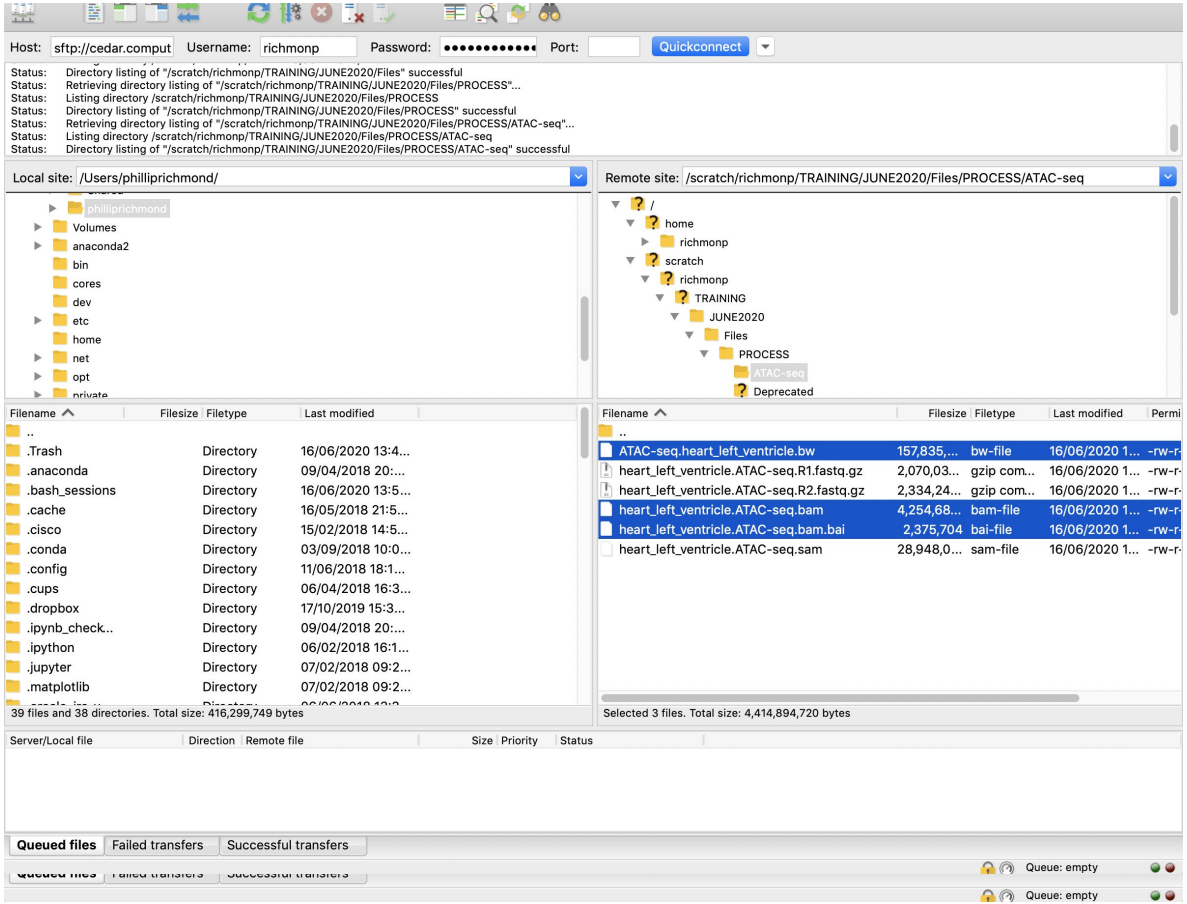
Use FileZilla, or scp to transfer files onto your own computer

Transfer the .sorted.bam, .sorted.bam.bai, and .bw files onto your local machine.

You can use filezilla, or command line scp, or another file transfer protocol/client

FileZilla: (https://filezilla-project.org)

Host: cedar.computecanada.ca
Username: <yourUsername>
Password: <yourPassword>
Port: 22



Alternatively, use sshfs/OSX-fuse (mac)

This allows your computer to mount the cedar server drive remotely, so that the data stays on cedar but you can access it for visualization.

SSHFS/OSX-Fuse: https://osxfuse.github.io

(Don’t have the equivalent example for PC)

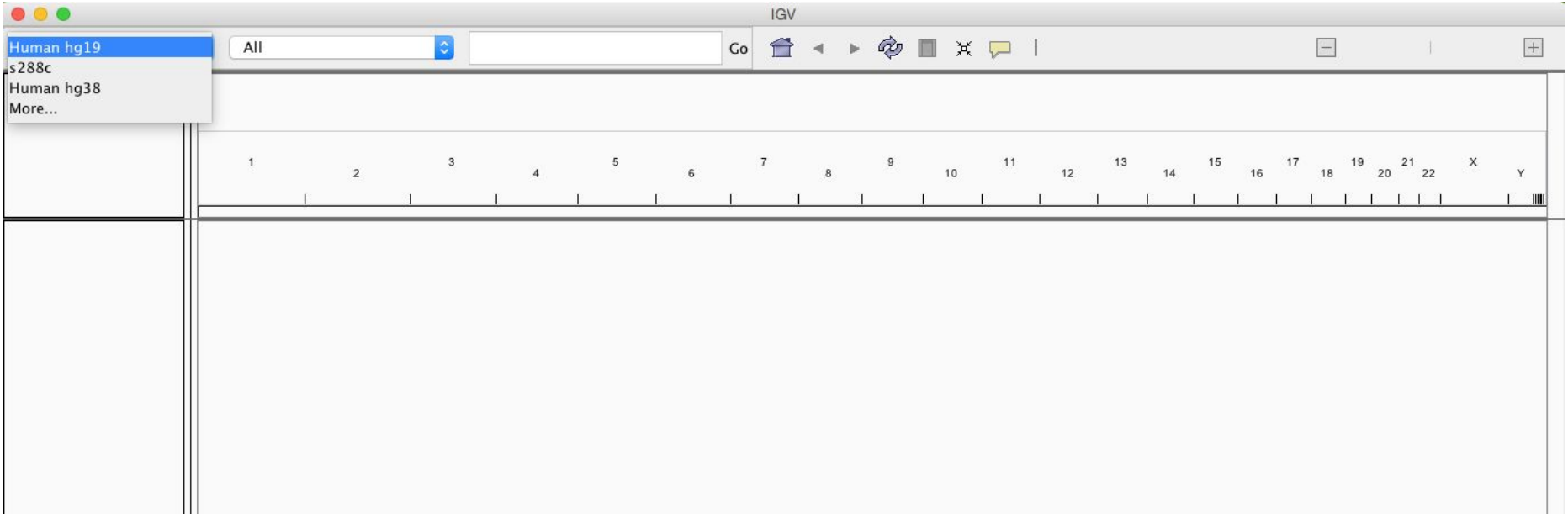
While transferring, download IGV

If you don’t have IGV already installed, you can download it here:

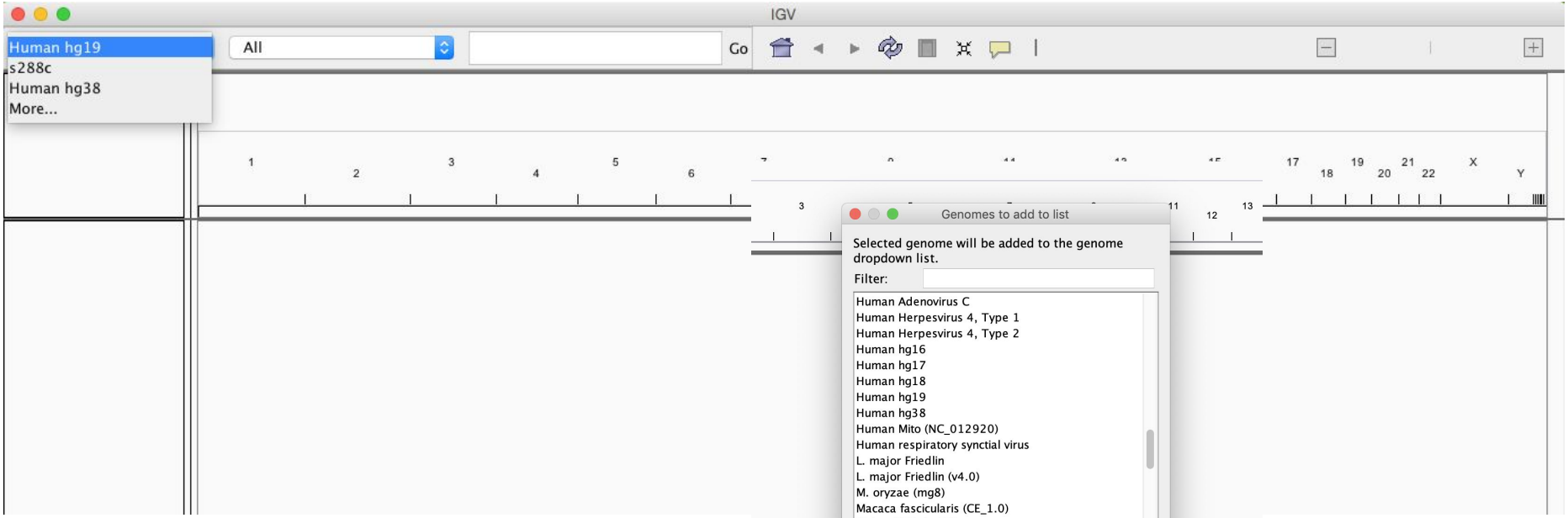
http://software.broadinstitute.org/software/igv/download

Then open IGV.

Open up IGV on your computer, load hg38.



Open up IGV on your computer, load hg38.



If Human hg38 isn't in your drop down, click on More…, and then scroll down to find it.

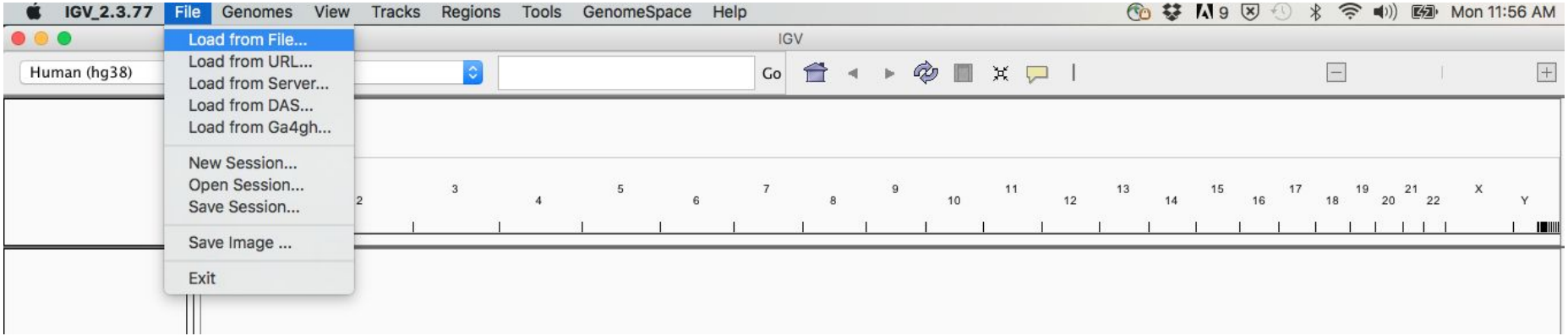
WESTGRID

UBC

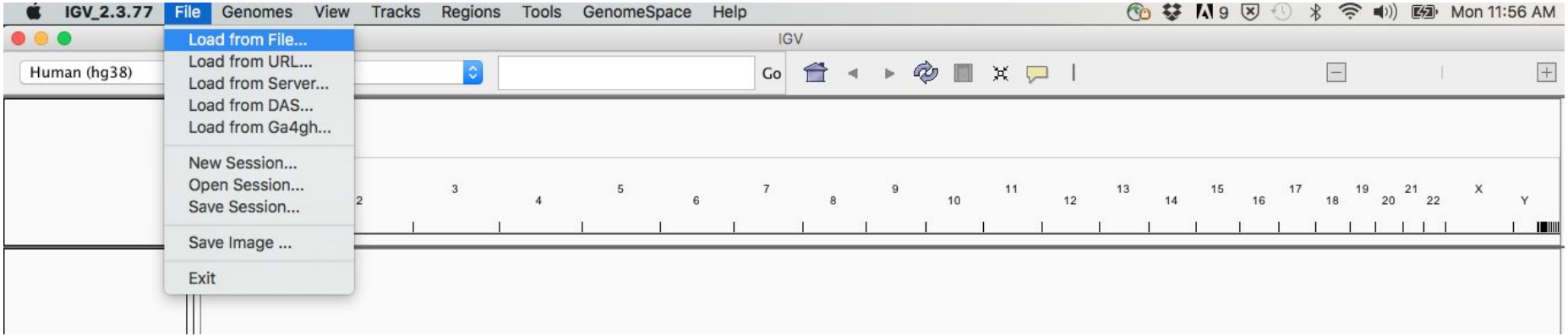
UBC100

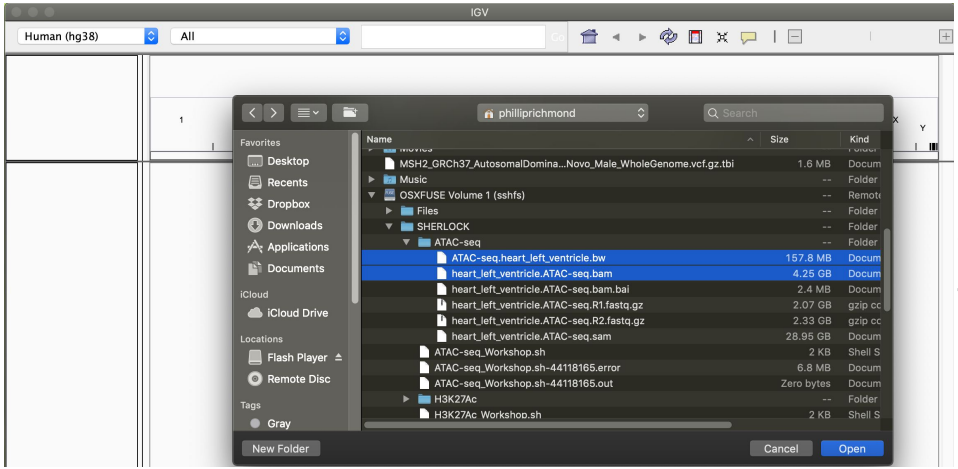
Advanced Research Computing

File→ Load from File: Load the .bam and .bw we just created



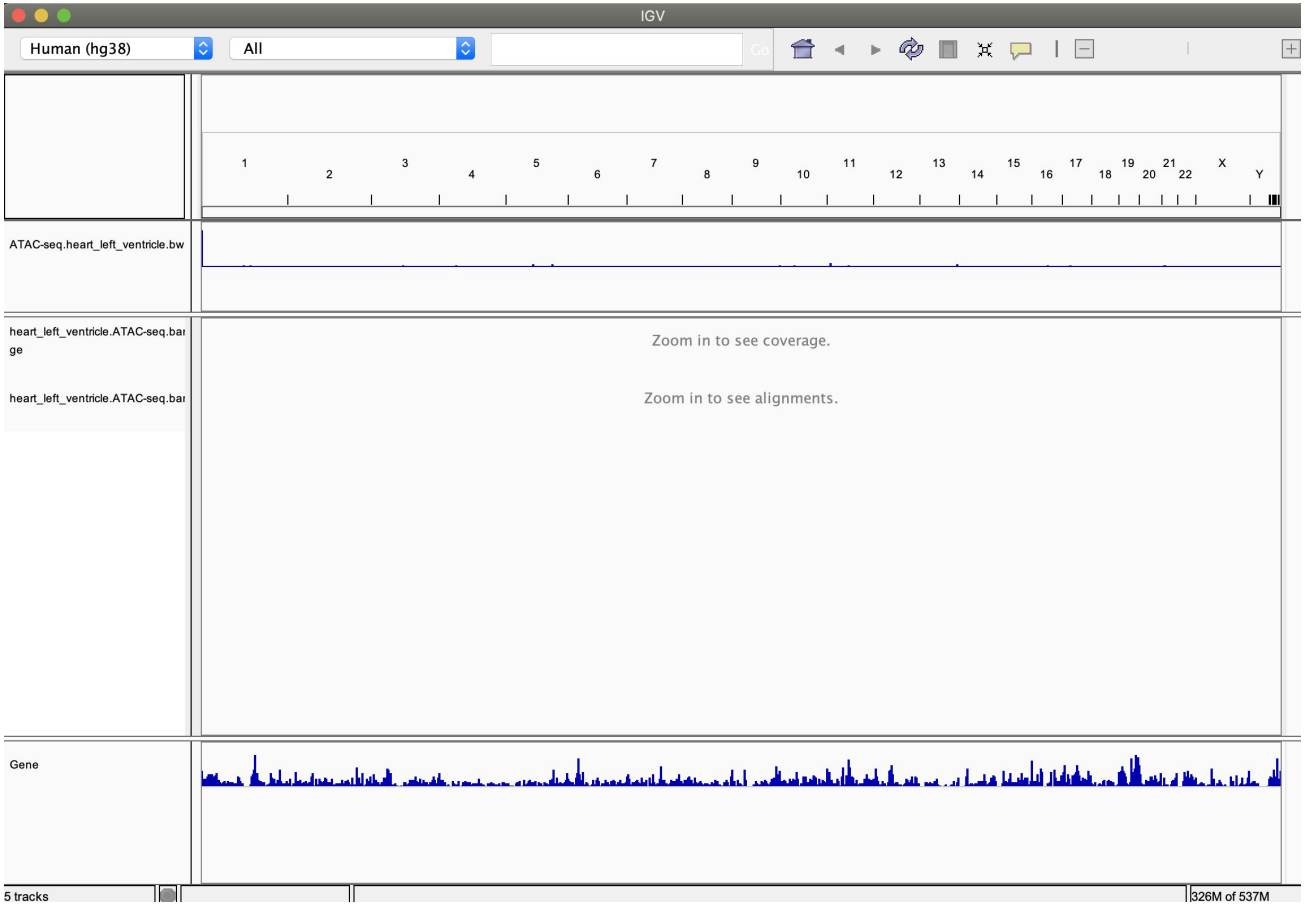
File→ Load from File: Load the .bam and .bw files



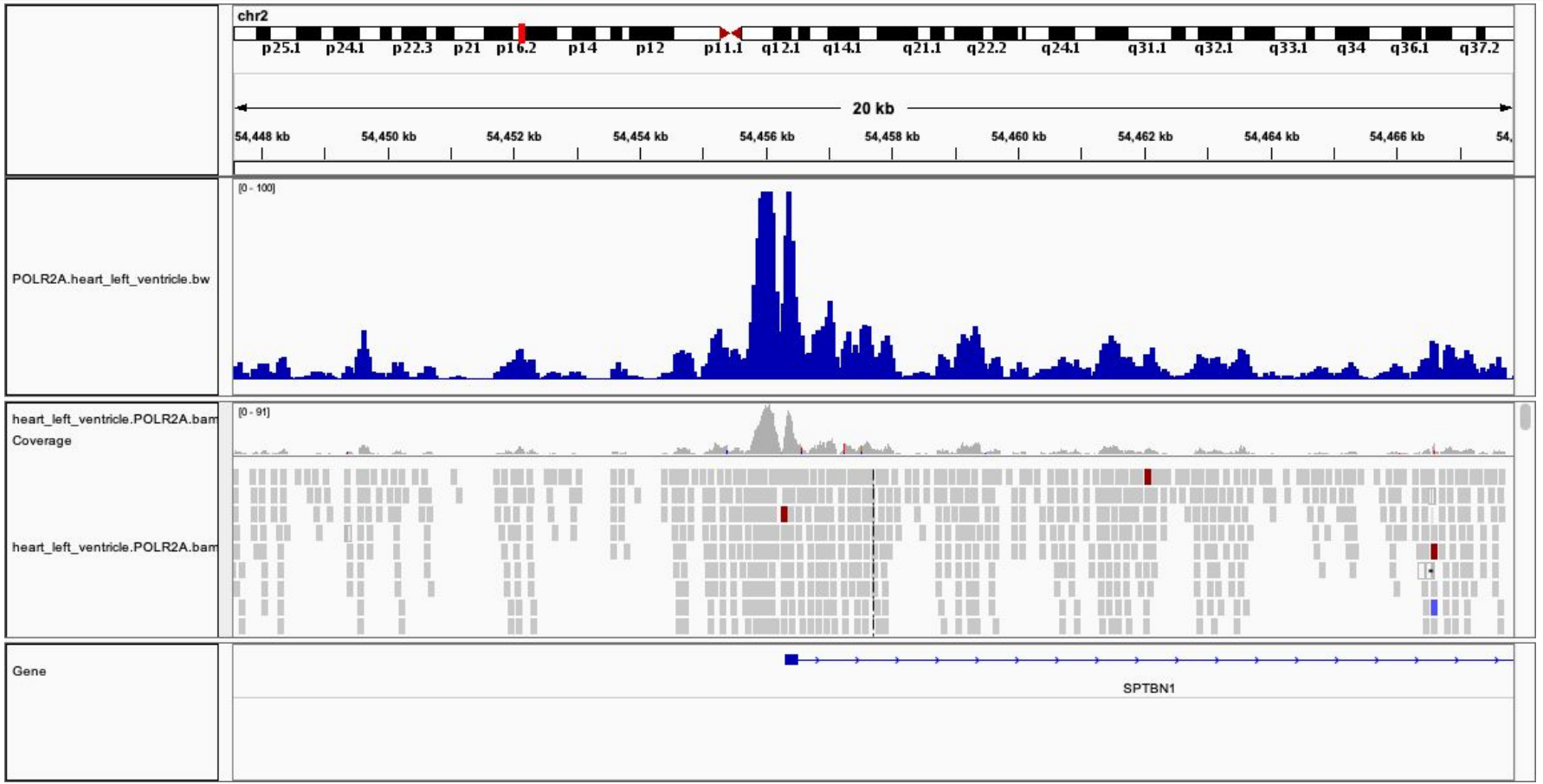


TIP: You do not load the .bai file

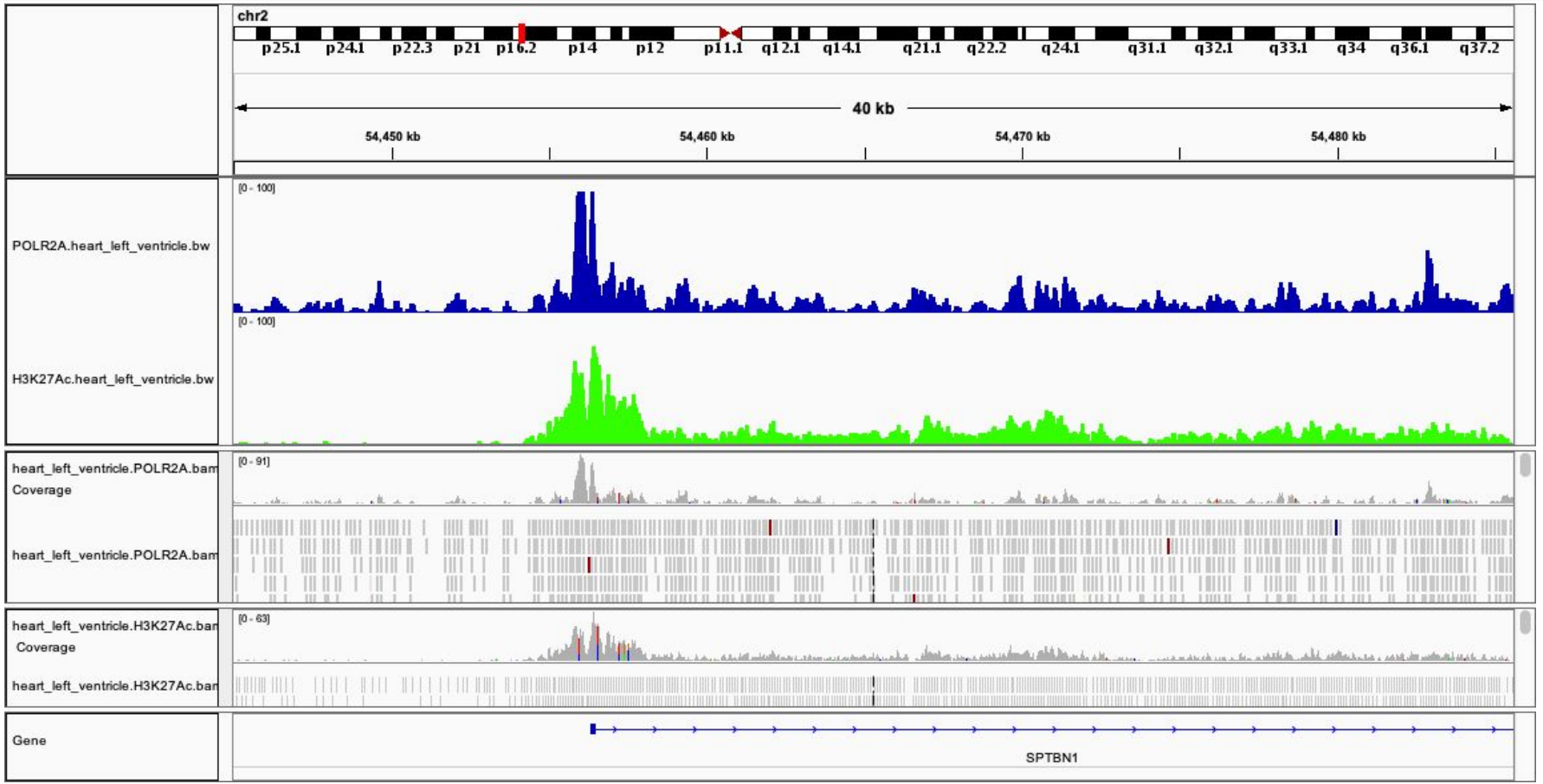
Loaded files from the home screen...let's zoom in!



Zoom to SPTBN1, entering into the search box



Add more data sets, modify colors and track heights, y-limits for range of values.



Problem Set 2

1. Load and visualize data from at least 2 experiments.
   a. If your scripts haven't finished running, use the files within: /scratch/richmonp/TRAINING/JUNE2020/Files/PROCESS/
2. Color each of the BigWig tracks a different color, and set their height to 100.
3. Take a snapshot of a TSS with read pileup and the BigWig tracks, save as a PNG.
4. (optional) Find a pileup of these signals outside of the genic regions in the genome. Hint, these are enhancers :).
5. Email your images to phillip.a.richmond@gmail.com

End of Lecture, what to do next

Go outside and enjoy the weather

Acknowledgements

- Phil Richmond (Teacher)
  - PhD Candidate in the Wasserman Lab, enjoys teaching
- Number 1 TA
  - Da real MVP: Dr. Oriol Fornes Crepo
- WestGrid https://www.westgrid.ca/ (Alex Razoumov)
- UBC ARC https://arc.ubc.ca (Roman Baranowski, Jerry Li)

WEST GRID

UBC 100

Advanced Research Computing

If you cannot use sbatch to run the scripts...

You can use salloc:

salloc --account=ubcss19-wa_cpu --reservation=ubcss19-wr_cpu --mem-per-cpu=4G --nodes=1 --cpus-per-task=8

Then, once your reservation works, you can run the scripts via:

bash <scriptName.sh>

FLASH DEBUGGING

$ samtools sort Sample1.bam -o Sample1.sorted.bam
Crazy characters printing to the screen

Fix: This sort command doesn't use a -o
Unless you specify -T and -O as well.
$ samtools sort Sample1.bam Sample1.sorted

$ samtools view -bS Sample1.sam Sample1.bam
Crazy characters printing to the screen

Fix: This commands needs a -o for the output
$ samtools view -bS Sample1.sam -o Sample1.bam

$ samtools index Sample1.bam
[E::hts_idx_push] unsorted positions
samtools index: "Sample1.bam" is corrupted or unsorted

Fix: Order matters. Sort before you index
$ samtools index Sample1.sorted.bam

$ bwa mem -t ../GENOME/genome.fa Sample_R1.fastq Sample_R2.fastq
[E::bwa_idx_load_from_disk] fail to locate the index files

Fix: the -t option requires an integer. Otherwise, all the other positional arguments are out of place.
$ bwa mem -t 4 ../GENOME/genome.fa Sample_R1.fastq Sample_R2.fastq

ERROR: Loading SAM/BAM index files are not supported: /Users/phillipprichmond/Desktop/NA20845.chr19.subregion.sorted.bam.bai
Load the SAM or BAM file directly.

Fix: Make sure you load the .bam file,
The .bai file just needs to be in the same directory
As the .bam file

WESTGRID

UBC UBC100

Advanced Research Computing