WESTGRID

compute canada
regional partner

UBC

UBC100

Advanced Research Computing

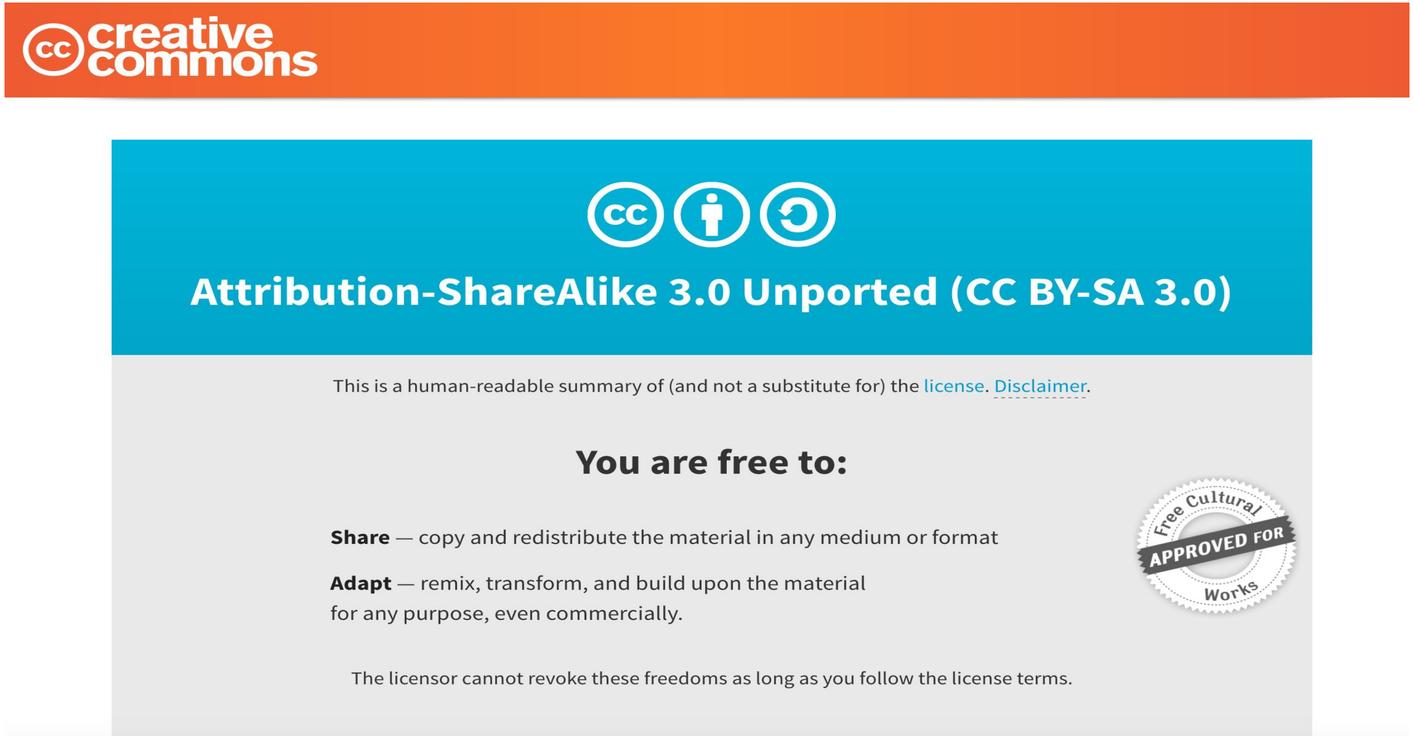
Education Outreach and Training Tutorials

Introduction to Short Read Mapping: The foundation of next generation sequencing analysis

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

Phillip A Richmond

Copyright Information



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"
- If you can, follow along with me. But if I move too fast (and I will for some people), just listen and take notes.
- This presentation will be recorded and the slides will remain available indefinitely

WESTGRID

UBC UBC100

Advanced Research Computing

Interactive Experience

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

Questions/Problems can be posted to the Sli.do:

https://www.sli.do

Code: M519

We have a couple TAs to assist in answering questions and solving problems, at the end of the session I can address unresolved questions

Your own cheat sheet

Copy paste commands from the github gist:

Github Gist
()

Each command is broken down as follows:

```
# What it does (name_of_command)
## Basic/advanced usage
### template example
Actual Command Line
```

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 new puppy Sherlock Holmes (https://sherlockthedoubledoodle.wordpress.com)

WESTGRID

UBC UBC100

Advanced Research Computing

Session Outline

- Introduction to next generation sequencing data & diverse data types
- Mapping reads to the genome using BWA mem
  - Interactive (salloc)
  - Scheduler (sbatch <jobscript>)
- Problem set 1
- Data visualization
- Problem set 2
- Closing remarks and downstream pipelines

Session Outline

- Introduction to next generation sequencing data & diverse data types
- Mapping reads to the genome using BWA mem
  - Interactive (salloc)
  - Scheduler (sbatch <jobscript>)
- Problem set 1
- Data visualization
- Problem set 2
- Closing remarks and downstream pipelines

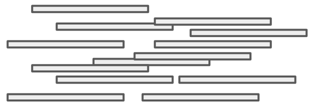
WESTGRID

UBC UBC100

Advanced Research Computing

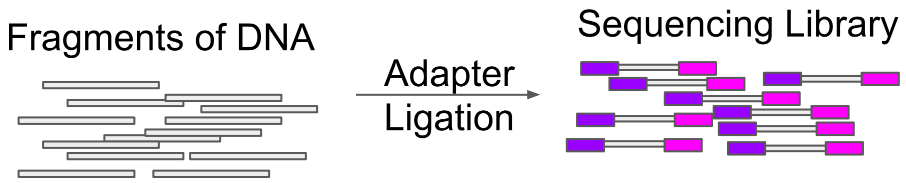
Next generation sequencing: Short-read sequencing

Fragments of DNA

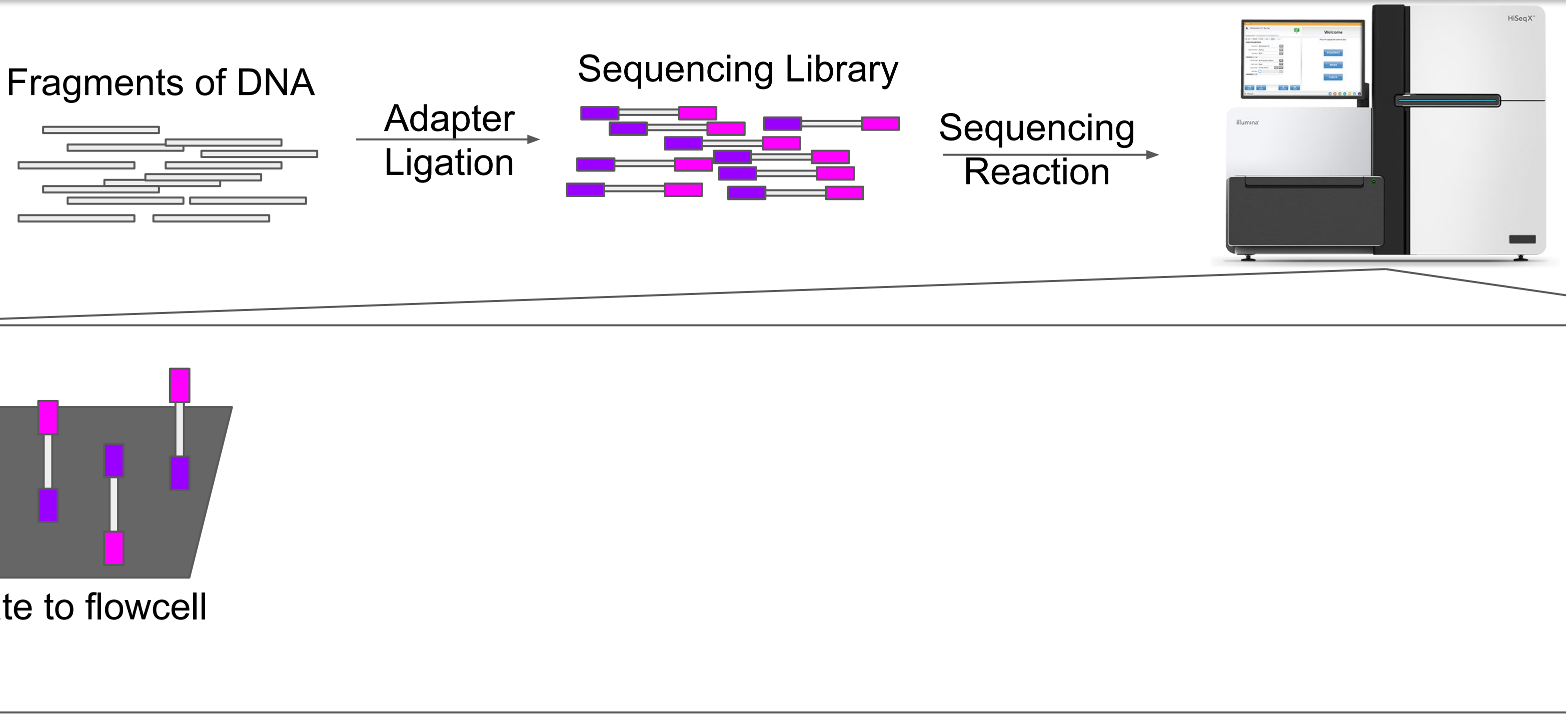


Advanced Research Computing

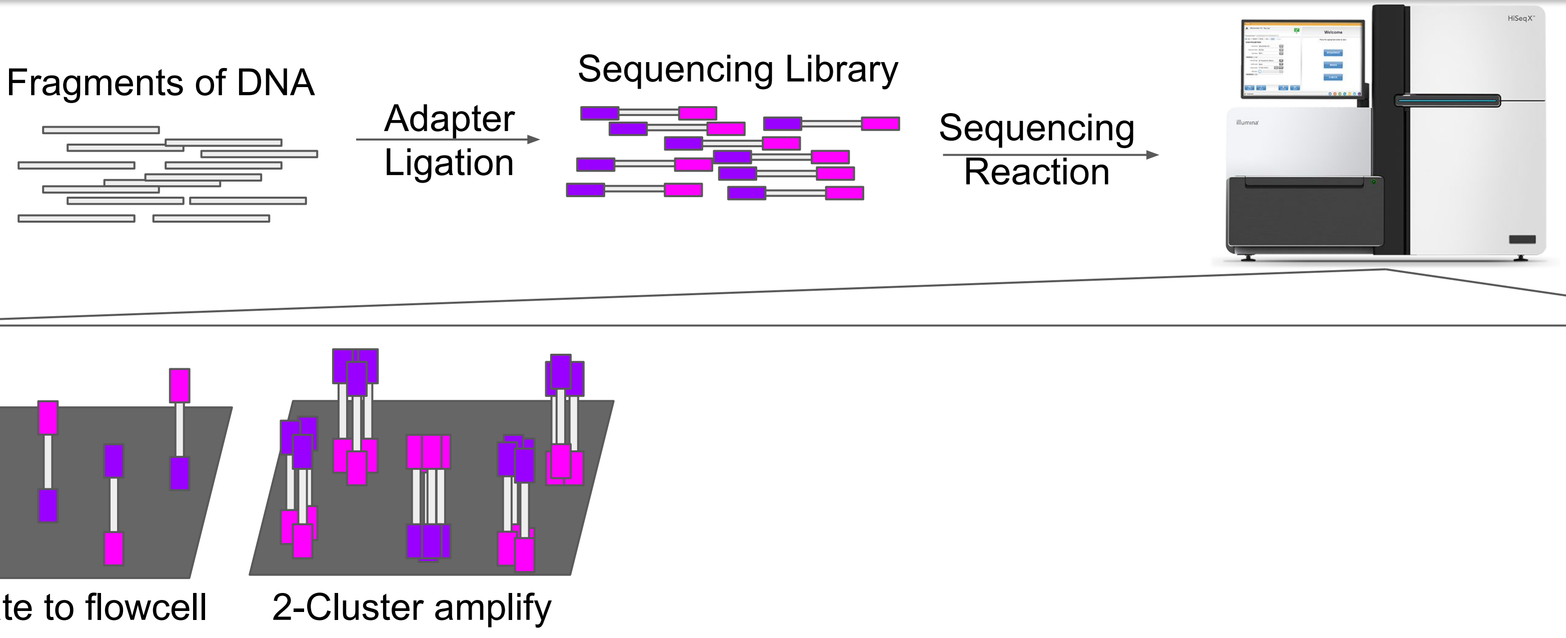
Next generation sequencing: Short-read sequencing



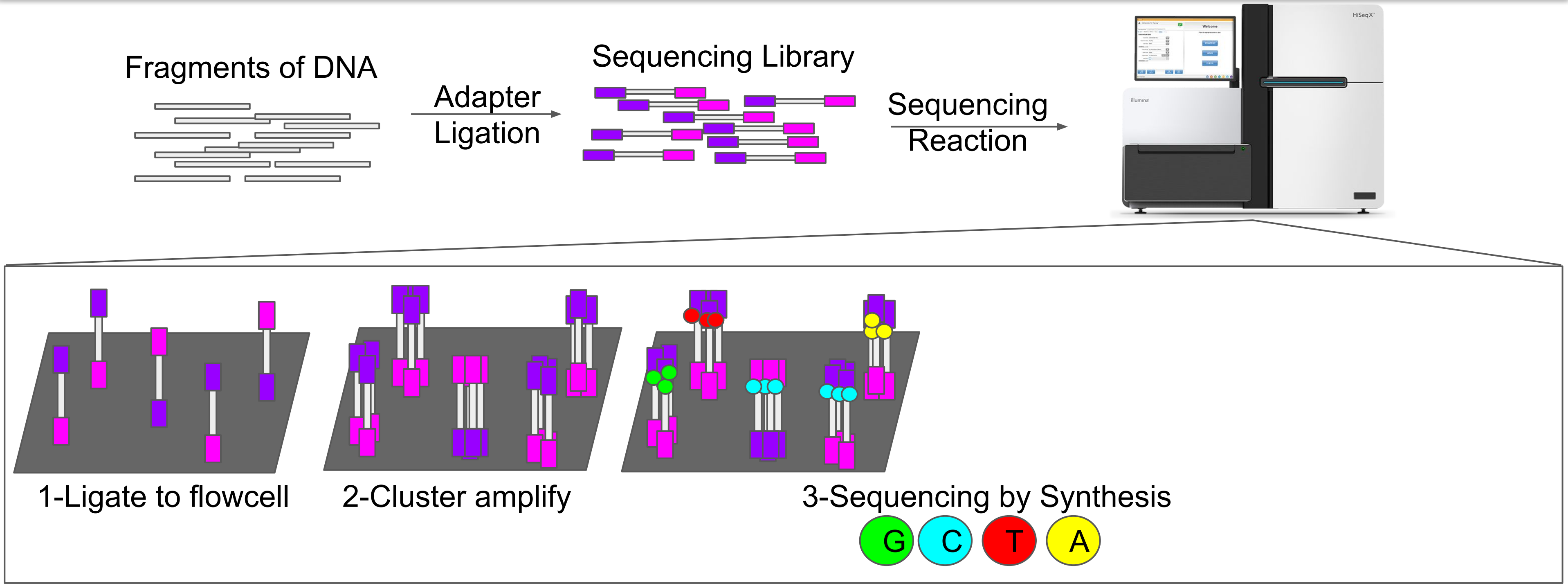
Next generation sequencing: Short-read sequencing



Next generation sequencing: Short-read sequencing



Next generation sequencing: Short-read sequencing

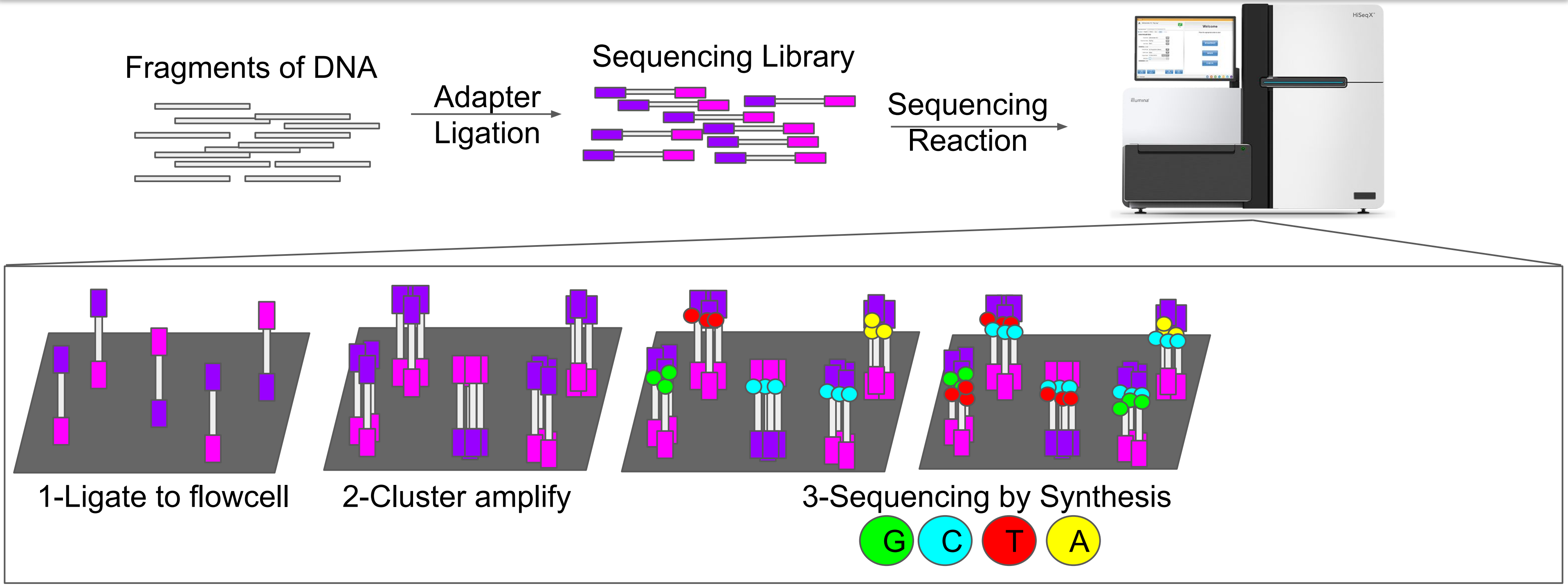


WESTGRID

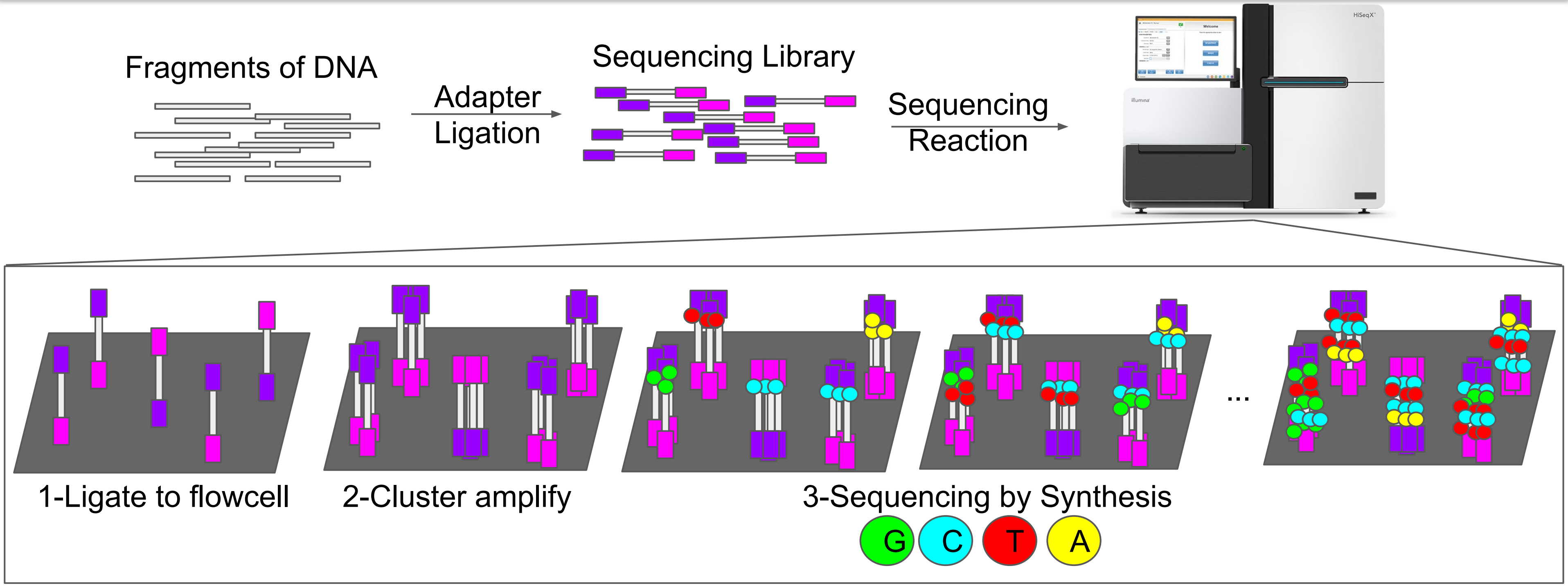
UBC UBC100

Advanced Research Computing

Next generation sequencing: Short-read sequencing



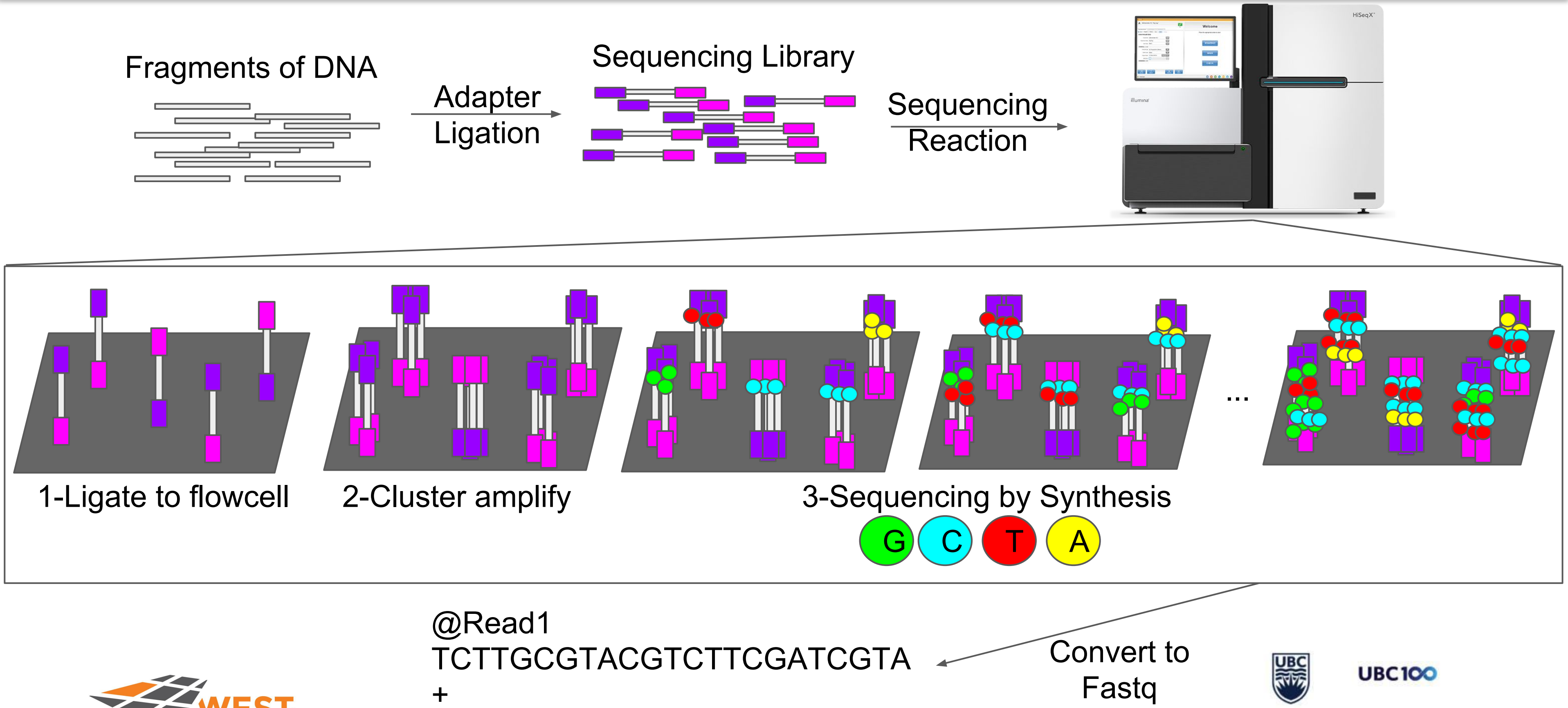
Next generation sequencing: Short-read sequencing



WESTGRID

UBC
UBC100
Advanced Research Computing

Next generation sequencing: Short-read sequencing



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

WESTGRID

UBC UBC100

Advanced Research Computing

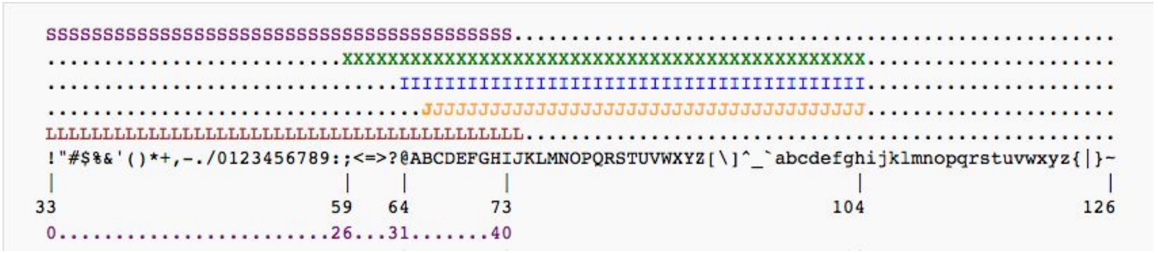
Diverse Input Data, Same Output Format

- 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

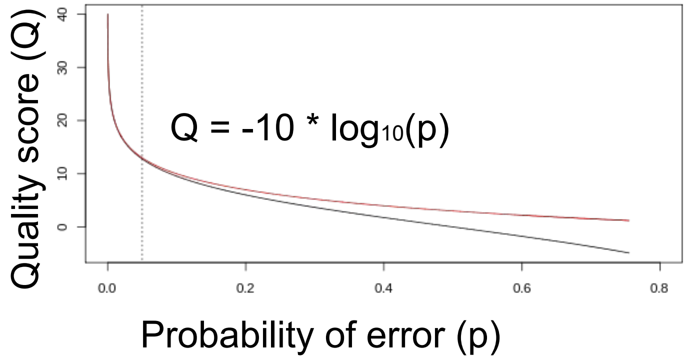
EXAMPLE	MEANING
@K00171:617:HMMTNBBXX:1:1101:28686:1648 1:N:0:GACTAGTA	@Readname:And:Flowcell:Info 1 or 2 for read pair:N:0:Barcode
TCTTGCGTACGTCTTCGATCGTA	Sequence
+	"Plus Sign"
!!@$@##@!%!@#$!!LLBBDKSNK	ASCII-Quality Scores

Diverse Input Data, Same Output Format

EXAMPLE	MEANING
@K00171:617:HMMTNBBXX:1:1101:28686:1648 1:N:0:GACTAGTA	@Readname:And:Flowcell:Info 1 or 2 for read pair:N:0:Barcode
TCTTGCGTACGTCTTCGATCGTA	Sequence
+	“Plus Sign”
BBBBCCA?>><>=;:BBBBBBBBB	**ASCII-Quality Scores**



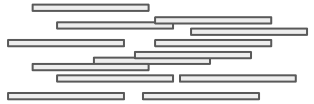
```
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS.....................................................
..........................XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX......................
...............................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII......................
.................................JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ...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



Reference-based Mapping: DNA-seq Variant 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: DNA-seq and Variant Calling

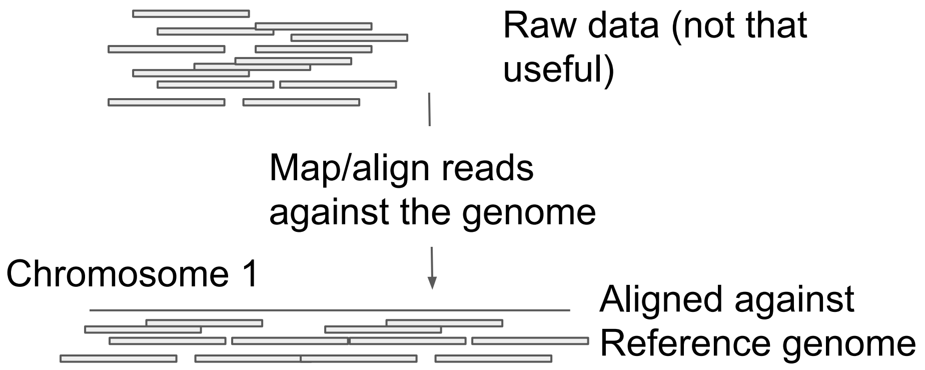


Raw data (not that useful)

Reference-based Mapping: DNA-seq Variant 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

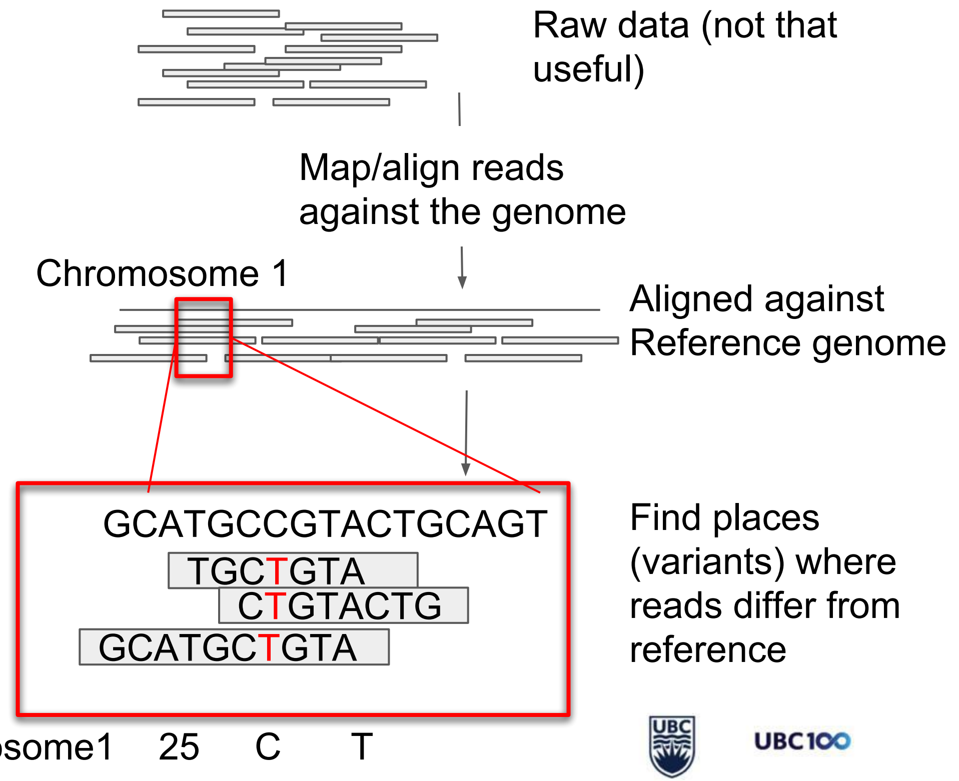
Example: DNA-seq and Variant Calling


Raw data (not that useful)
Map/align reads against the genome
Chromosome 1
Aligned against Reference genome

Reference-based Mapping: DNA-seq Variant 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: DNA-seq and Variant Calling

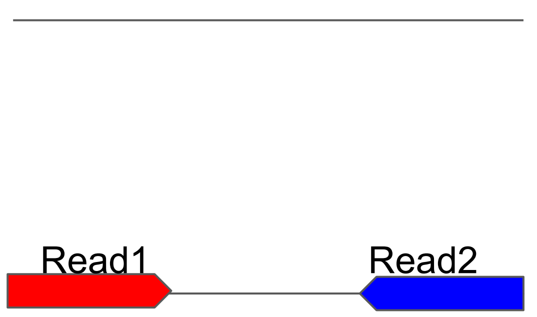

Raw data (not that useful)
Map/align reads against the genome
Chromosome 1
Aligned against Reference genome
GCATGCCGTACTGCAGT
TGCTGTA
CTGTACTG
GCATGCTGTA
Find places (variants) where reads differ from reference
Chromosome1 25 C T

Paired-end DNA-sequencing

Most DNA sequencing is now paired-end

In paired end sequencing, you sequence two ends of the same fragment of DNA

This way, when you map back to the reference genome, you know more info about how Read1 and Read2 should map (More on this later)



Piece of DNA, ~500bp total length

Sequence from each end, pointing towards the middle of the piece of DNA

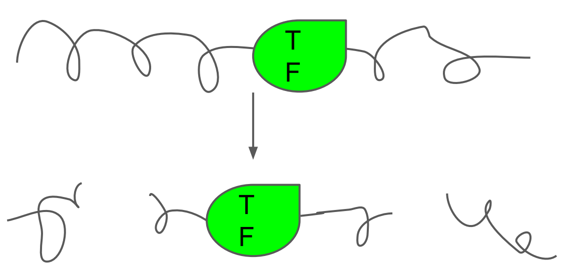
Other Applications: ChIP-seq

Chromatin Immunoprecipitation Sequencing (ChIP-seq) protocol:

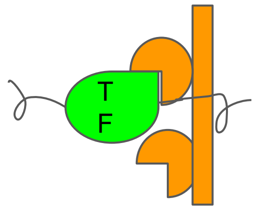
Purpose: To find which sequences of DNA a specific protein interacts with, i.e Transcription Factor (TF).

1-Crosslink DNA:Protein

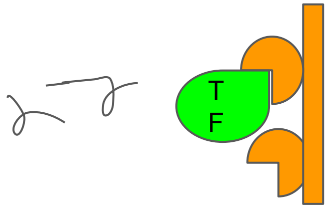
2-Shear



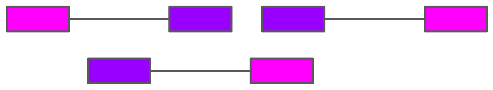
3-Pull Down protein using anti-protein antibody on a column, wash away other DNA



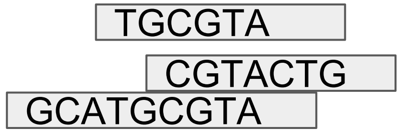
4-Reverse Crosslink



5-Ligate sequencing adapters



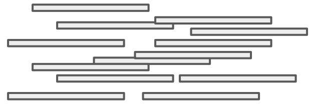
6-Sequence Library



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

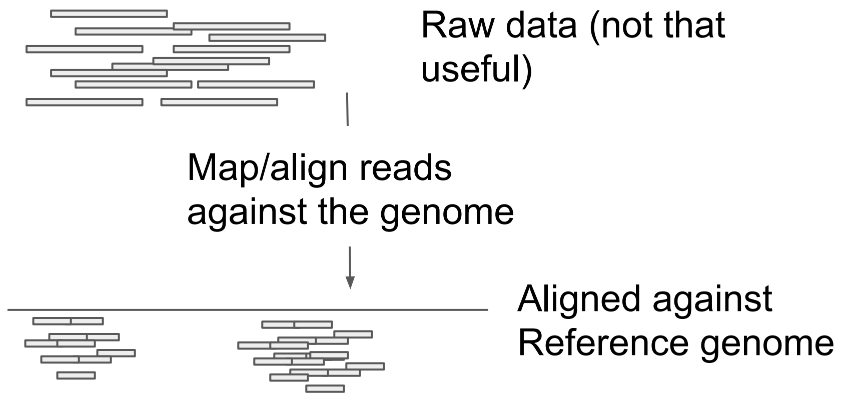


Raw data (not that useful)

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

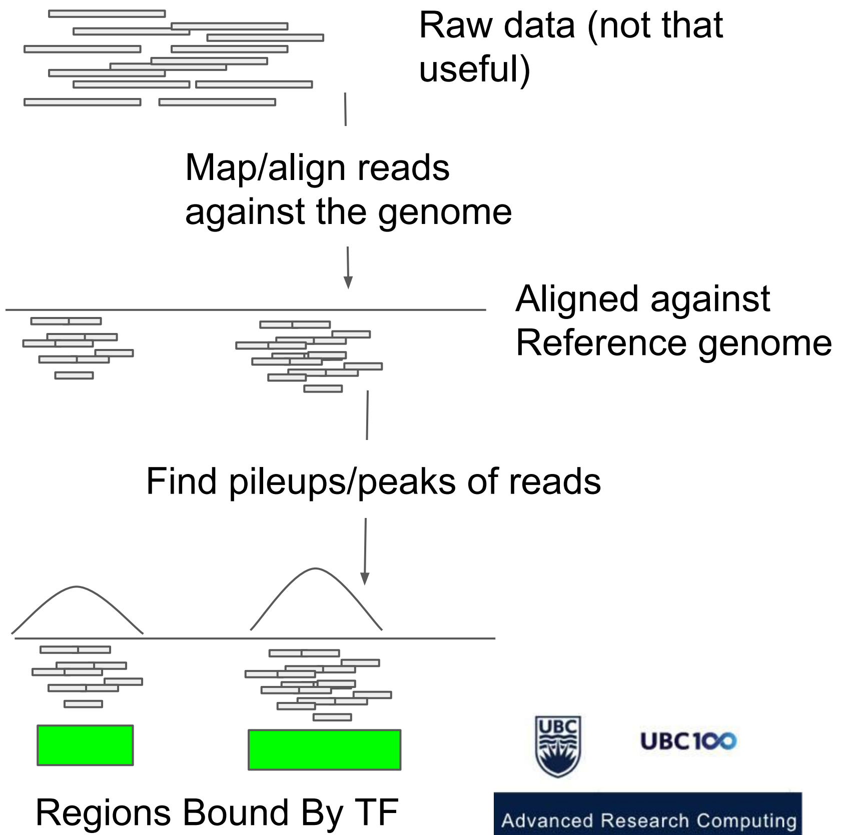
Example: ChIP-seq for a Transcription Factor

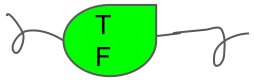

Raw data (not that useful)
Map/align reads against the genome
Aligned against Reference genome

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)
Map/align reads against the genome
Aligned against Reference genome
Find pileups/peaks of reads
Regions Bound By TF



Session Outline

- Introduction to next generation sequencing data & diverse data types
- **Mapping reads to the genome using BWA mem**
  - **Interactive (salloc)**
  - **Scheduler (sbatch <jobscript>)**
- Problem set 1
- Data visualization
- Problem set 2
- Closing remarks and downstream pipelines

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

Orienting yourself to this workshop directory

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

Change into that directory:

```
$ cd /scratch/richmonp/TRAINING/
```

Important subdirectories:

/scratch/richmonp/TRAINING/Files/SCRIPTS/ -
    Has scripts & templates that you can copy/use

/scratch/richmonp/TRAINING/Files/RAW_DATA/ -
    Has the raw data that we will be using today for analysis

/scratch/richmonp/TRAINING/Files/PROCESS/ -
    If nothing works for you today, these are some processed files that you can look at/visualize

/scratch/richmonp/TRAINING/JUNE2018/ -
    This is where your own workshop directory will exist, and you have permission over it

WESTGRID

UBC UBC100

Advanced Research Computing

Set up a workshop directory

```
$ mkdir <directory>
$ mkdir /scratch/richmonp/TRAINING/JUNE2018/RICHMOND/
```

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/JUNE2018/RICHMOND/
```

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

Enter into an interactive instance: salloc

The salloc command allows you to “log-in” to a specific node. The command is as follows:

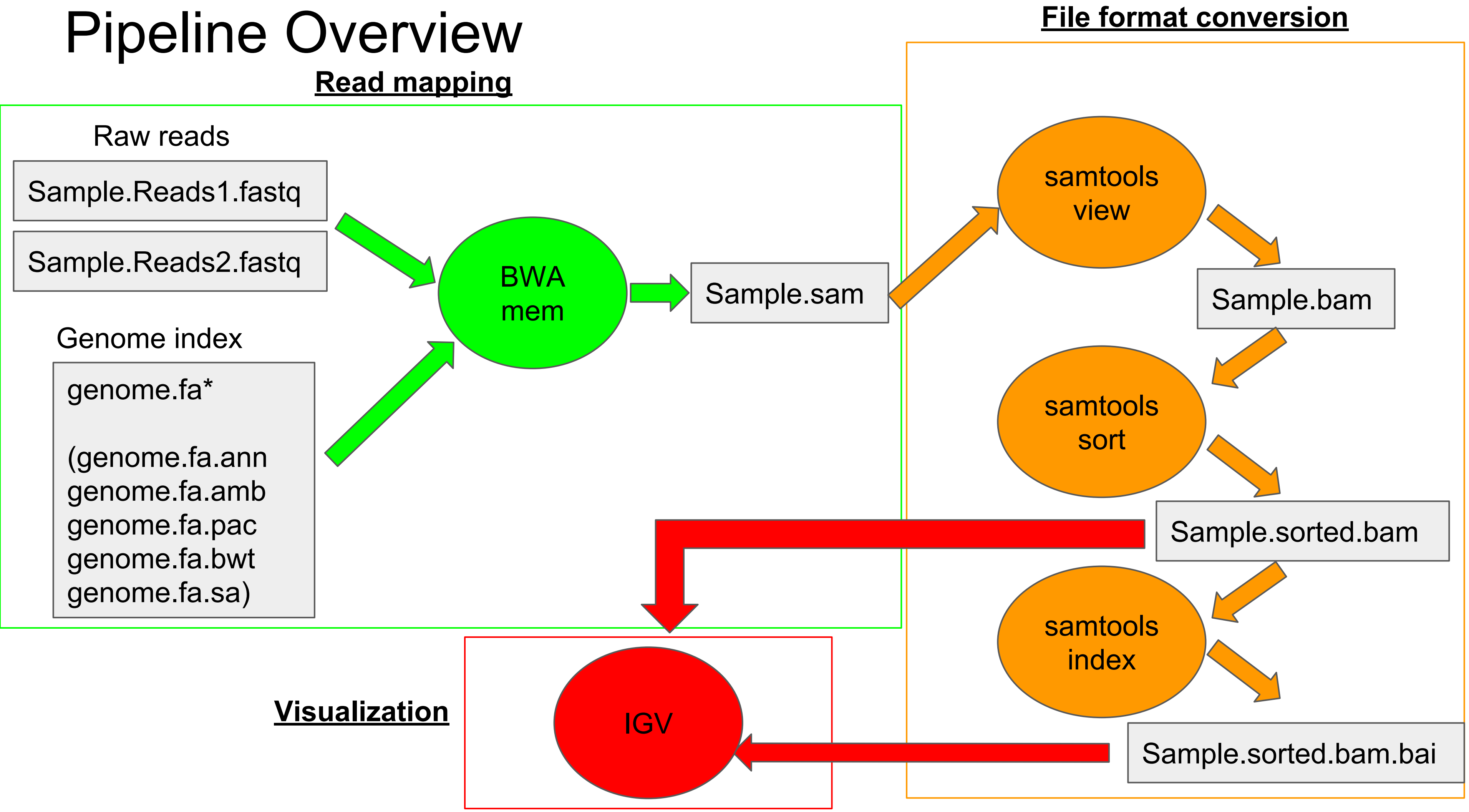
```
$ salloc <options>
```

This command will ask for 1 node, 4CPUs, and 2G/CPU:

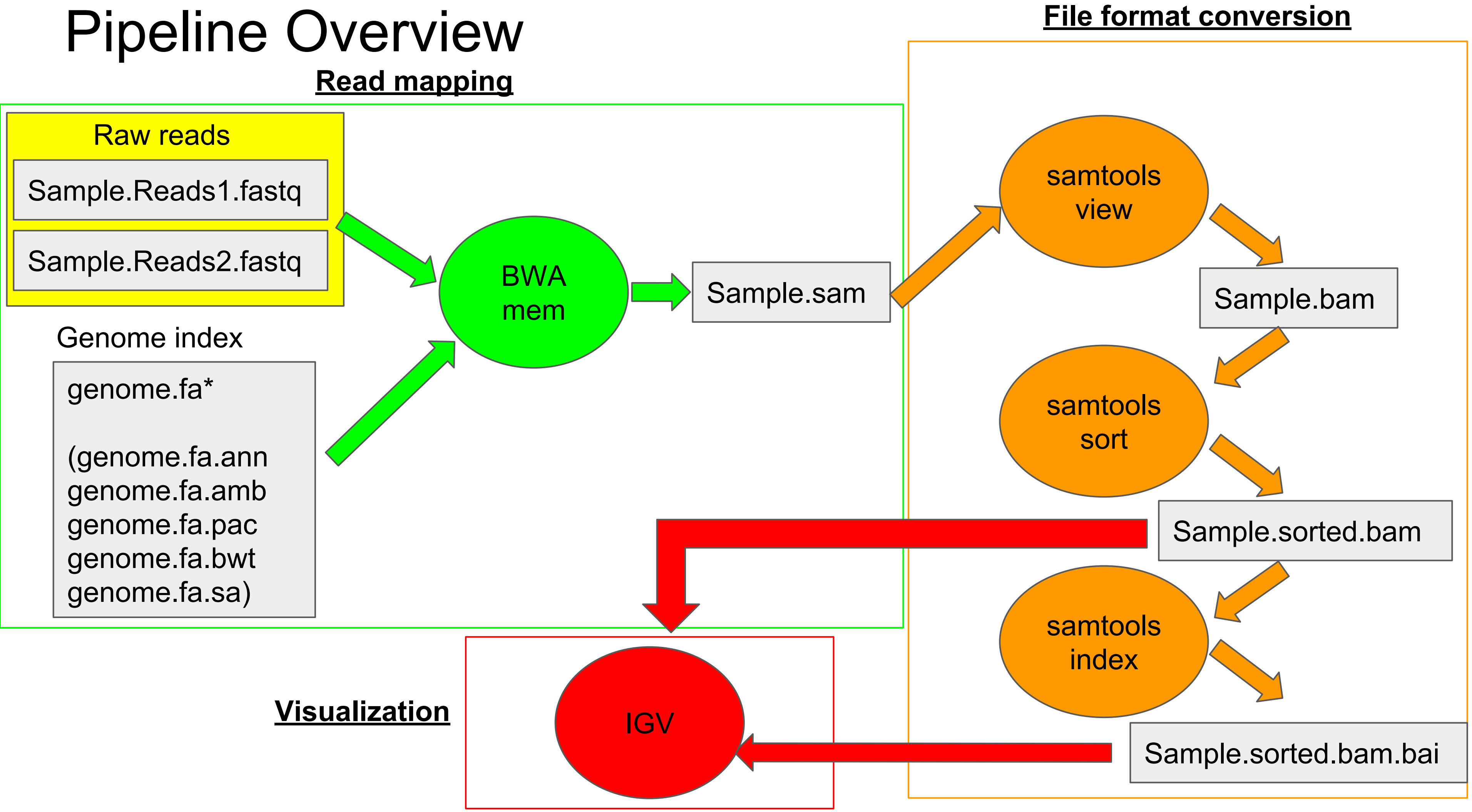
```
$ salloc --account=wgssubc-wa_cpu --reservation=wgssubc-wr_cpu --nodes=1
--mem-per-cpu=2048M --cpus-per-task=4
```

Advanced Research Computing

Pipeline Overview



Pipeline Overview



Let’s take a look at our fastq files

```
$ more /scratch/richmonp/TRAINING/Files/RAW_DATA/Sample1_R1.fastq
```

Note, that this file has a SRR readnames, since it was downloaded from the SRA:

@SRR098401.47362517**/1**

The **/1** denotes that this is read1 of a paired end dataset. Looking at the first read in the R2 file shows the pair to this read with /2:

```
$ more /scratch/richmonp/TRAINING/Files/RAW_DATA/Sample1_R2.fastq
```

@SRR098401.47362517**/2**

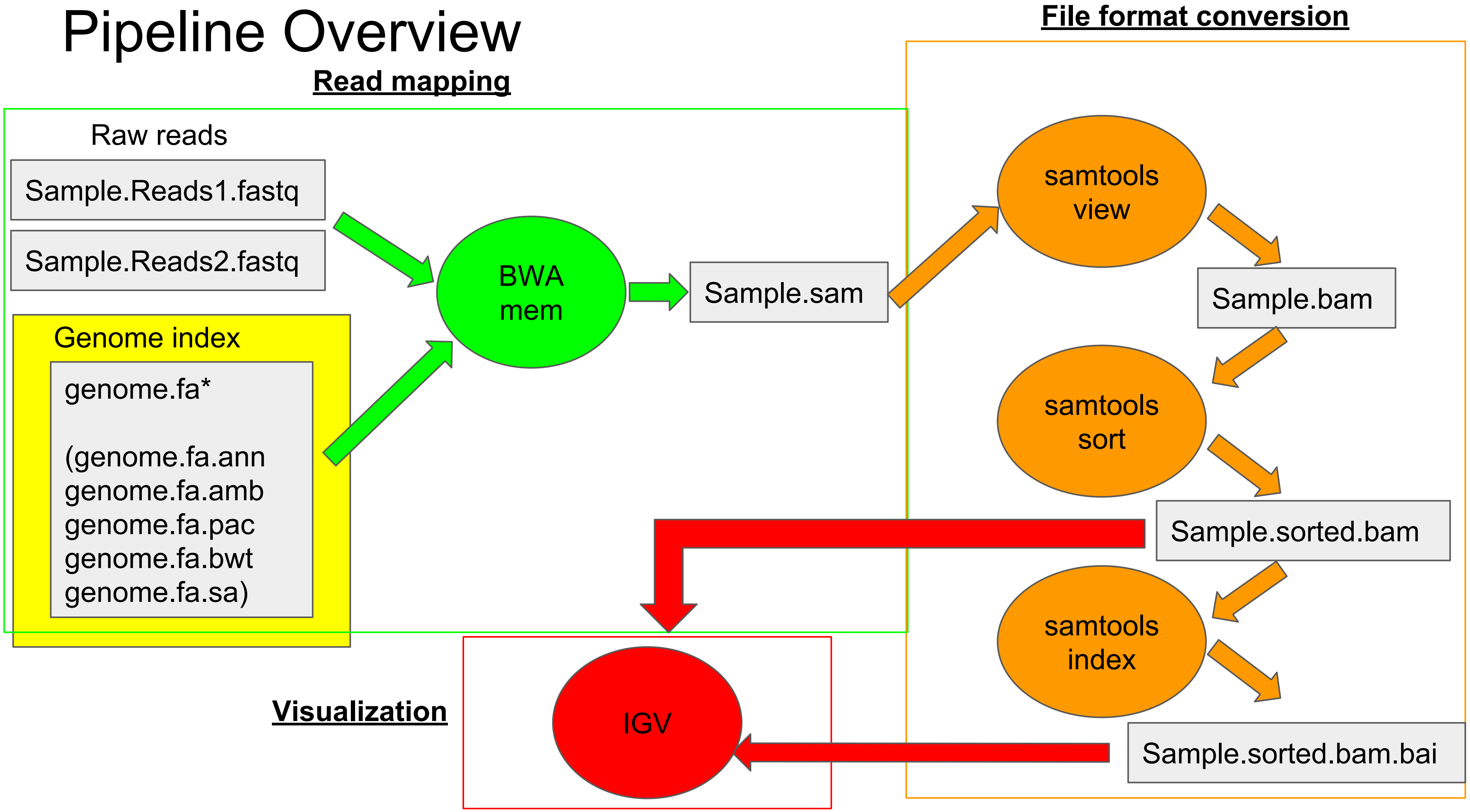
Copy both these fastq files into your own workshop directory:

```
$ cp /scratch/richmonp/TRAINING/Files/RAW_DATA/Sample1_*
/scratch/richmonp/TRAINING/JUNE2018/<YourDirectory>
```

WESTGRID

UBC UBC100 Advanced Research Computing

Pipeline Overview



Reference Genome, Fasta file format

Reference genomes are packaged into fasta files.
Format:

```
>chromosome1_Name OtherChromInfo AccessionInfo Etc.
NNNNNNATTCGTTGATGGATAGCATGATCAGTAGACATGACATGACAGATGAGGGATATGATGACCA
CCACCCAGATTCCCGGCCGGCCGGCCGGCCCGGGCCGGCCGGCCGGGCCCGGCTATATATATATA
CATAG ….
>chromosome2_Name OtherChromInfo AccessionInfo Etc.
NNNNNNNCCCCGGCCGGCCGGCCGGCCCGGGCCGGCCGGCCGGGCCCGGCTATATATATATACAT
AGATGATCAGTAGACATGACATGACAGATGAGGGATATGATGACCACCACCCAGATTGGAGTTGCCA
GAT
```

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

But...luckily we already have pre-built genomes!

Thanks to the team at McGill, who has built the mugqic (no idea what that word is), we have pre-built genomes

They are located here: /cvmfs/ref.mugqic/genomes/species/

Today, we are using Homo_sapiens.GRCh38:
Take a look inside this directory:

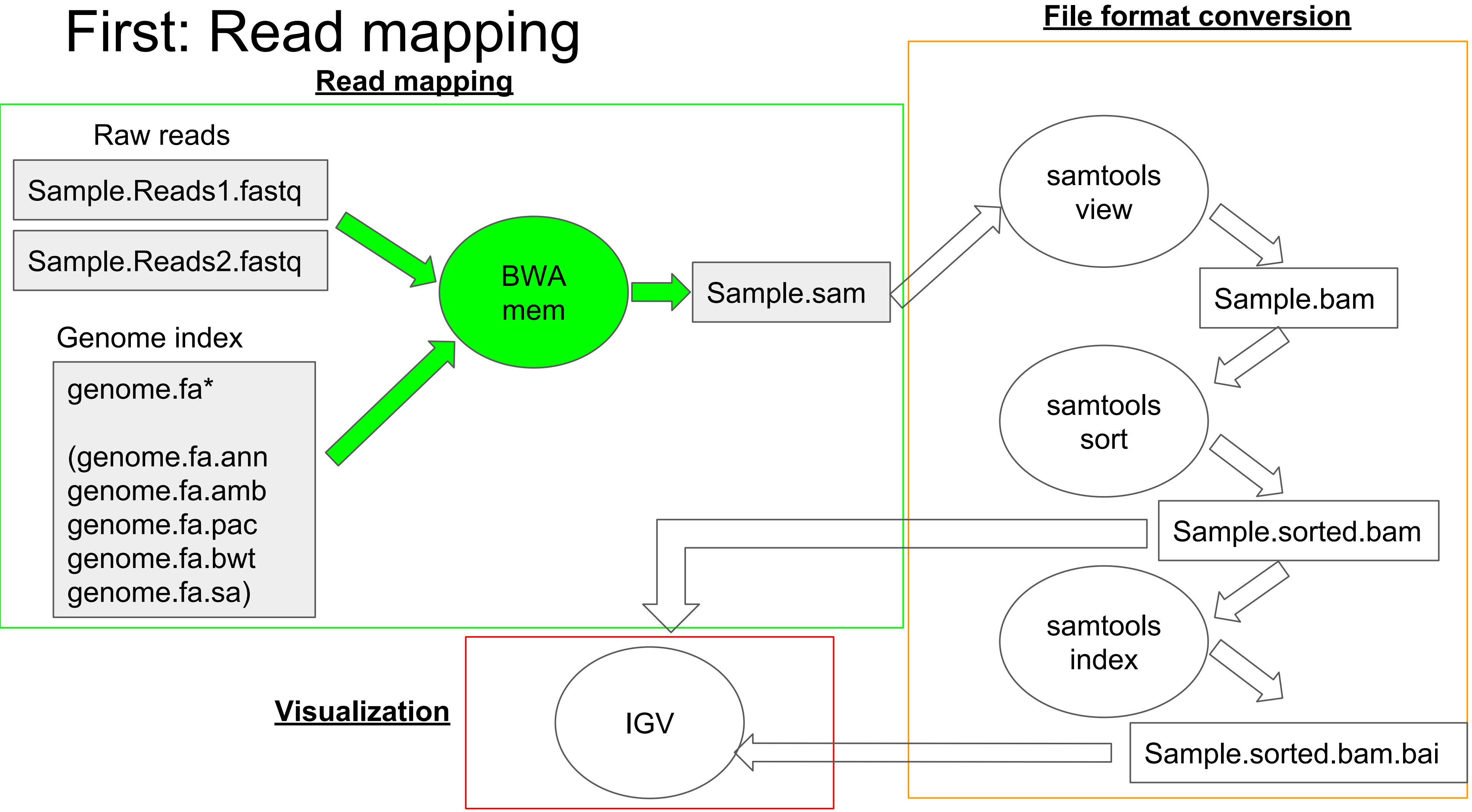
```
$ ls /cvmfs/ref.mugqic/genomes/species/Homo_sapiens.GRCh38/genome/
```

There is a fasta file there we can use:
/cvmfs/ref/mugqic/genomes/species/Homo_sapiens.GRCh38/genome/Homo_sapiens.GRCh38.fa
You can take a look at this file:

```
$ more /cvmfs/ref.mugqic/genomes/species/Homo_sapiens.GRCh38/genome/Homo_sapiens.GRCh38.fa
```

And a BWA index, which we refer to by pointing at this file:
/cvmfs/ref.mugqic/genomes/species/Homo_sapiens.GRCh38/genome/bwa_index/Homo_sapiens.GRCh38.fa

First: Read mapping



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
```

UBC100

Advanced Research Computing

The output SAM file

@SQ - Sequence (contig/chromosome) from reference file

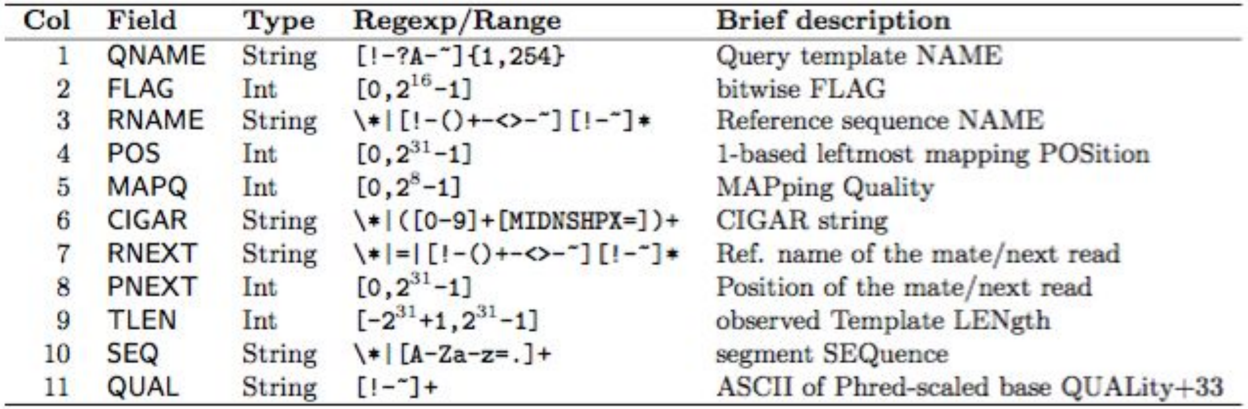
@PG - Program information about mapping

@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

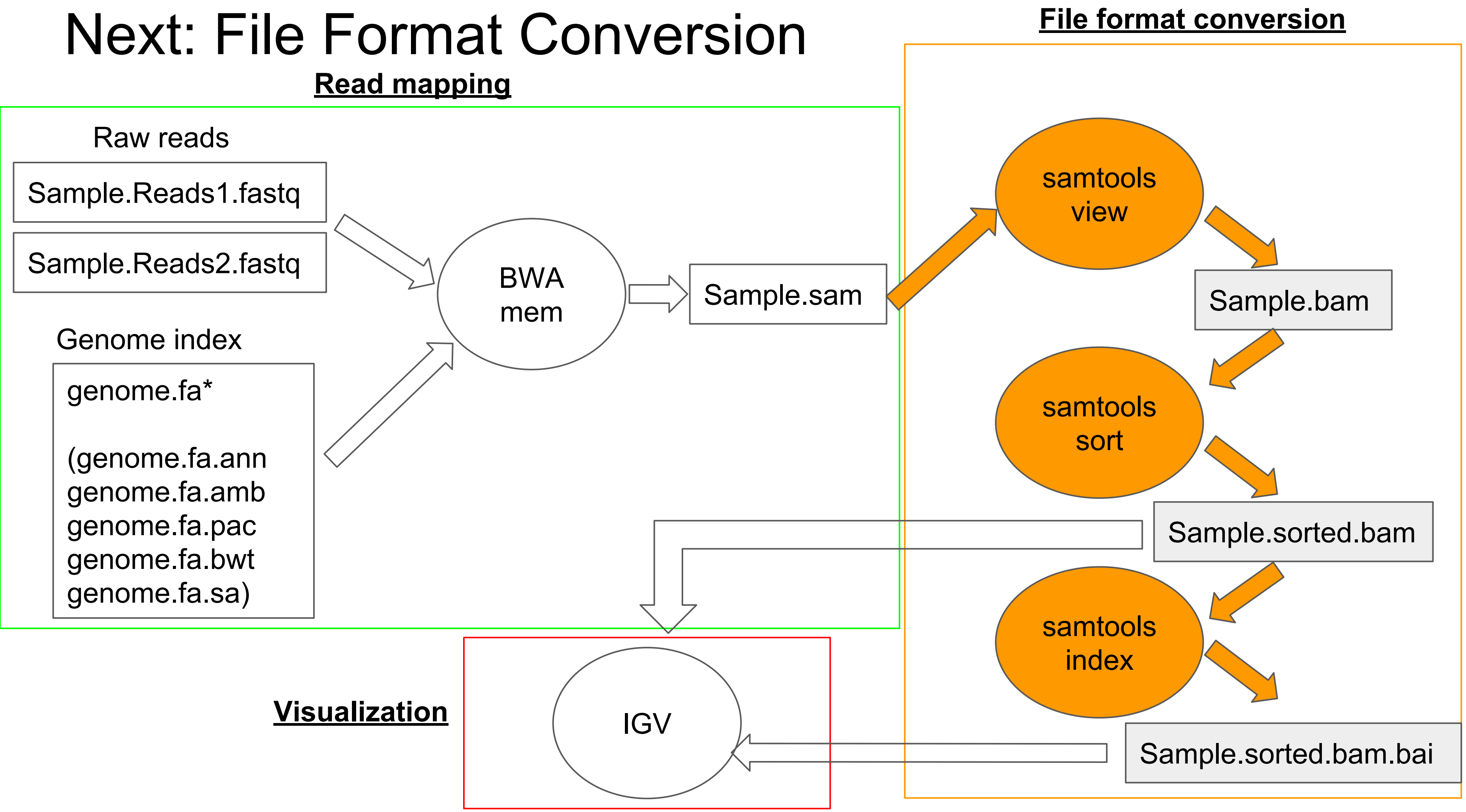
Line2: Read2



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

Next: File Format Conversion



Learning the samtools commands

```
$ module load samtools/1.3.1
```

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
```

WESTGRID

UBC UBC100

Advanced Research Computing

Bam file is a binary format of that sam file

We cannot look at these binary files the same way as we look at text files

Downstream applications will almost always ask for a .bam file

Sorting is necessary for downstream applications

Index will be required for IGV

Before we visualize our data, we will create a shell script that can execute all the commands we just ran

Building Pipeline Shell Scripts

In general, I like to build shell scripts in three steps:

1. Make a basic shell script with the commands, and run it from the command line while in an salloc instance with: sh <shellscript>
   a. Make sure it runs and completes without an error
2. Add a the header to a shell which has directions for the SLURM scheduler, and submit it to the queue
3. Generalize your shell script with variables to allow for easier re-use on different samples

Building Pipeline Shell Scripts

In general, I like to build shell scripts in three steps:

1. Make a basic shell script with the commands, and run it from the command line while in an salloc instance with: sh <shellscript>
   a. Make sure it runs and completes without an error
2. Add a the header to a shell which has directions for the SLURM scheduler, and submit it to the queue
3. Generalize your shell script with variables to allow for easier re-use on different samples

Advanced Research Computing

Edit Pipeline_v1.sh and re-run within salloc instance

Copy the Pipeline_v1.sh script into your workshop directory and edit it

```
$ cp /scratch/richmonp/TRAINING/Files/SCRIPTS/Pipeline_v1.sh
/scratch/richmonp/TRAINING/JUNE2018/RICHMOND
```

Change RICHMOND to be your own directory

Then run it with the sh command:

```
$ sh /scratch/richmonp/TRAINING/JUNE2018/RICHMOND/Pipeline_v1.sh
```

Once it finishes, we can check our output to know that this script is functional

Building Pipeline Shell Scripts

In general, I like to build shell scripts in three steps:

1. Make a basic shell script with the commands, and run it from the command line while in an salloc instance with: sh <shellscript>
    a. Make sure it runs and completes without an error
2. Add the header to the shell script which has directions for the SLURM scheduler, and submit it to the queue
3. Generalize your shell script with variables to allow for easier re-use on different samples

WESTGRID

UBC UBC100

Advanced Research Computing

Example Header for SLURM job

```
$ cp /scratch/richmonp/TRAINING/Files/SCRIPTS/ExampleHeader.sh
/scratch/richmonp/TRAINING/JUNE2018/<YourDirectory>
```

```
#!/bin/bash

#SBATCH --account=wgssubc-wa_cpu --reservation=wgssubc-wr_cpu

## Mail Options
#SBATCH --mail-user=youremail@email.com
#SBATCH --mail-type=ALL

## CPU Usage
#SBATCH --mem-per-cpu=2048M
#SBATCH --cpus-per-task=4
#SBATCH --time=2-0:00
#SBATCH --nodes=1

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

This is specific to the workshop, and you need to use it today

Make sure you edit this to be your own email address

This is where we specify CPU requirements. More info on this can be found on Cedar Documentation and from Roman's Tutorial yesterday :)

Where our standard output and standard error file will go

Concatenate ExampleHeader.sh and Pipeline_v1.sh

We can easily add the header to the top of our existing pipeline script using the cat command (from within your workshop directory):

```
$ cat   ExampleHeader.sh   Pipeline_v1.sh   >   Pipeline_v2.sh
```

Change the output files to be called Sample1_PipelineV2*

Once we are happy with our script, we will submit it to the queue

Now we can run our job in the queue

Submit job using sbatch

```
$ sbatch  <file.sh>
$ sbatch /scratch/richmonp/TRAINING/JUNE2018/RICHMOND/Pipeline_v2.sh
```

Check job status using squeue

```
$  squeue  -u <username>
$  squeue  -u richmonp
```

When the job is finished, we can check our output files (.sam, .bam, sorted.bam, .sorted.bam.bai) and our .out/.error files

Building Pipeline Shell Scripts

In general, I like to build shell scripts in three steps:

1. Make a basic shell script with the commands, and run it from the command line while in an salloc instance with: sh <shellscript>
   a. Make sure it runs and completes without an error
2. Add the header to the shell script which has directions for the SLURM scheduler, and submit it to the queue
3. Generalize your shell script with variables to allow for easier re-use on different samples

WESTGRID

UBC UBC100

Advanced Research Computing

Pipeline_v3 as an example of using variables in scripts

```
THREADS=4
SAMPLE_ID=Bart_Simpson
WORKING_DIR=/scratch/richmonp/TRAINING/Files/PROCESS/
FASTQR1=/scratch/richmonp/TRAINING/Files/RAW_DATA/Sample2_R1.fastq
FASTQR2=/scratch/richmonp/TRAINING/Files/RAW_DATA/Sample2_R2.fastq
BWA_INDEX=/cvmfs/ref.mugqic/genomes/species/Homo_sapiens.GRCh38/genome/bwa_index/Homo_sapiens.GRCh38.fa
GENOME_FASTA=/cvmfs/ref.mugqic/genomes/species/Homo_sapiens.GRCh38/genome/Homo_sapiens.GRCh38.fa
```

Here, I am setting variables at the top of the file, and then referring to them within the commands below

This allows for easier re-purposing of scripts.

Now we will take a quick break, then work on the problem set

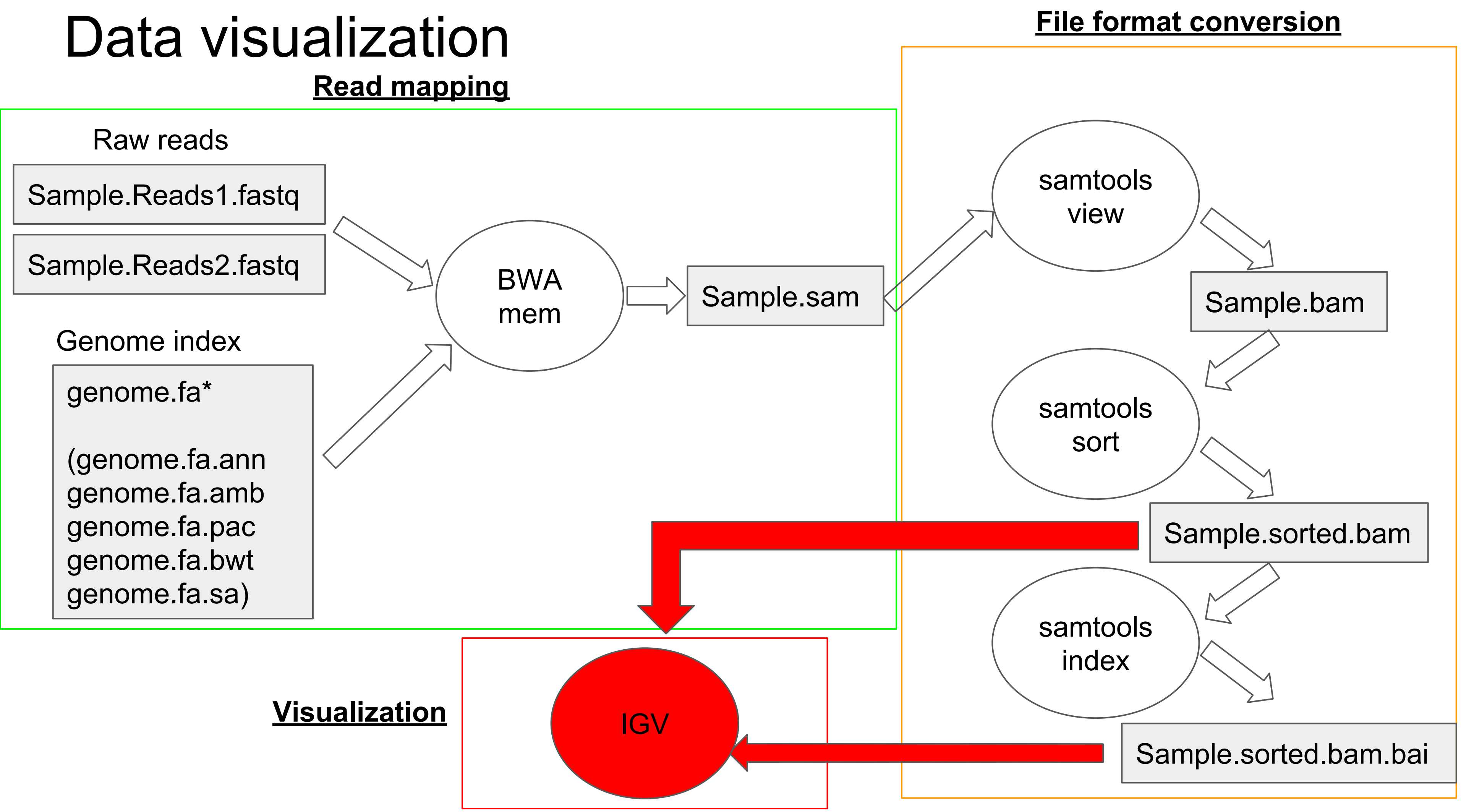
The problem set will have you map different input data files, which we will be using for visualization

Included in this problem set is are files for ChIP-seq data :)

Problem Set:

/scratch/richmonp/TRAINING/Files/PROBLEMSET/ProblemSet1.md

Data visualization

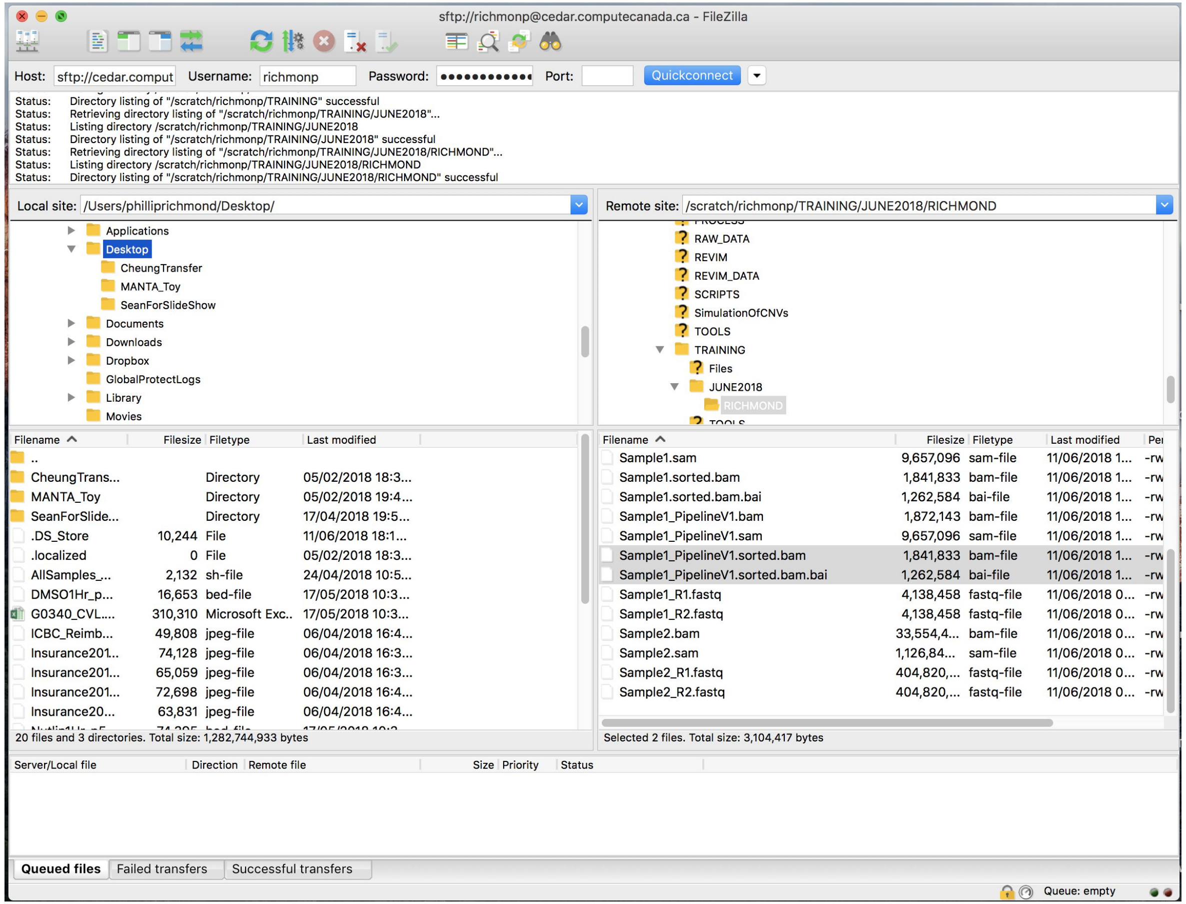


Use FileZilla to transfer files onto your own computer

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

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

FileZilla: (https://filezilla-project.org/download.php?type=client)



sftp://richmonp@cedar.computecanada.ca - FileZilla

Host: sftp://cedar.comput Username: richmonp Password: ●●●●●●●●●●●● Port: Quickconnect

```
Status:    Directory listing of "/scratch/richmonp/TRAINING" successful
Status:    Retrieving directory listing of "/scratch/richmonp/TRAINING/JUNE2018"...
Status:    Listing directory /scratch/richmonp/TRAINING/JUNE2018
Status:    Directory listing of "/scratch/richmonp/TRAINING/JUNE2018" successful
Status:    Retrieving directory listing of "/scratch/richmonp/TRAINING/JUNE2018/RICHMOND"...
Status:    Listing directory /scratch/richmonp/TRAINING/JUNE2018/RICHMOND
Status:    Directory listing of "/scratch/richmonp/TRAINING/JUNE2018/RICHMOND" successful
```

Local site: /Users/phillipirichmond/Desktop/

- Applications
- Desktop
  - CheungTransfer
  - MANTA_Toy
  - SeanForSlideShow
- Documents
- Downloads
- Dropbox
- GlobalProtectLogs
- Library
- Movies

Remote site: /scratch/richmonp/TRAINING/JUNE2018/RICHMOND

- RAW_DATA
- REVIM
- REVIM_DATA
- SCRIPTS
- SimulationOfCNVs
- TOOLS
- TRAINING
  - Files
  - JUNE2018
    - RICHMOND

Filename	Filesize	Filetype	Last modified
..			
CheungTrans...		Directory	05/02/2018 18:3...
MANTA_Toy		Directory	05/02/2018 19:4...
SeanForSlide...		Directory	17/04/2018 19:5...
.DS_Store	10,244	File	11/06/2018 18:1...
.localized	0	File	05/02/2018 18:3...
AllSamples_...	2,132	sh-file	24/04/2018 10:5...
DMSO1Hr_p...	16,653	bed-file	17/05/2018 10:3...
G0340_CVL....	310,310	Microsoft Exc..	17/05/2018 10:3...
ICBC_Reimb...	49,808	jpeg-file	06/04/2018 16:4...
Insurance201...	74,128	jpeg-file	06/04/2018 16:3...
Insurance201...	65,059	jpeg-file	06/04/2018 16:3...
Insurance201...	72,698	jpeg-file	06/04/2018 16:4...
Insurance20...	63,831	jpeg-file	06/04/2018 16:4...

20 files and 3 directories. Total size: 1,282,744,933 bytes

Filename	Filesize	Filetype	Last modified	Pe
Sample1.sam	9,657,096	sam-file	11/06/2018 1...	-rw
Sample1.sorted.bam	1,841,833	bam-file	11/06/2018 1...	-rw
Sample1.sorted.bam.bai	1,262,584	bai-file	11/06/2018 1...	-rw
Sample1_PipelineV1.bam	1,872,143	bam-file	11/06/2018 1...	-rw
Sample1_PipelineV1.sam	9,657,096	sam-file	11/06/2018 1...	-rw
Sample1_PipelineV1.sorted.bam	1,841,833	bam-file	11/06/2018 1...	-rw
Sample1_PipelineV1.sorted.bam.bai	1,262,584	bai-file	11/06/2018 1...	-rw
Sample1_R1.fastq	4,138,458	fastq-file	11/06/2018 0...	-rw
Sample1_R2.fastq	4,138,458	fastq-file	11/06/2018 0...	-rw
Sample2.bam	33,554,4...	bam-file	11/06/2018 0...	-rw
Sample2.sam	1,126,84...	sam-file	11/06/2018 0...	-rw
Sample2_R1.fastq	404,820,...	fastq-file	11/06/2018 0...	-rw
Sample2_R2.fastq	404,820,...	fastq-file	11/06/2018 0...	-rw

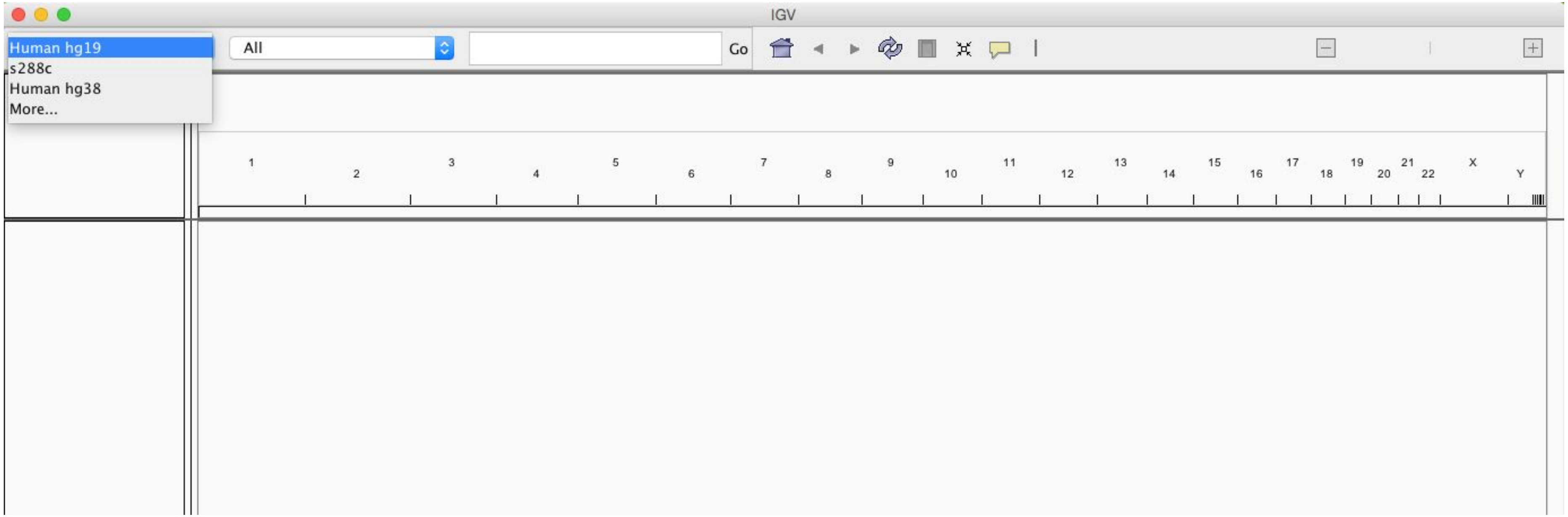
Selected 2 files. Total size: 3,104,417 bytes

Server/Local file	Direction	Remote file	Size	Priority	Status

Queued files | Failed transfers | Successful transfers

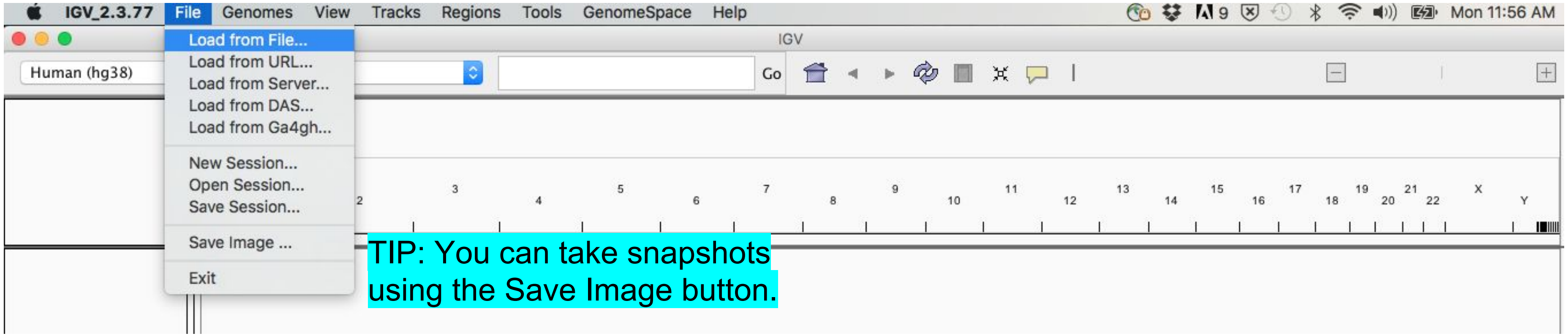
Queue: empty

Open up IGV on your computer, load hg38

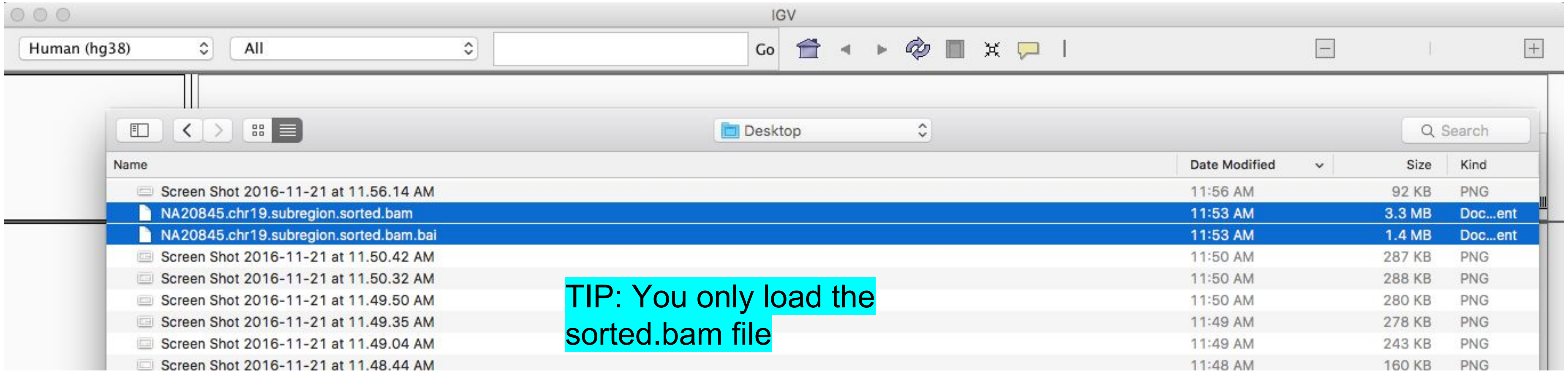

IGV
Human hg19
s288c
Human hg38
More...
All
Go
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 X Y

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

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



TIP: You can take snapshots using the Save Image button.



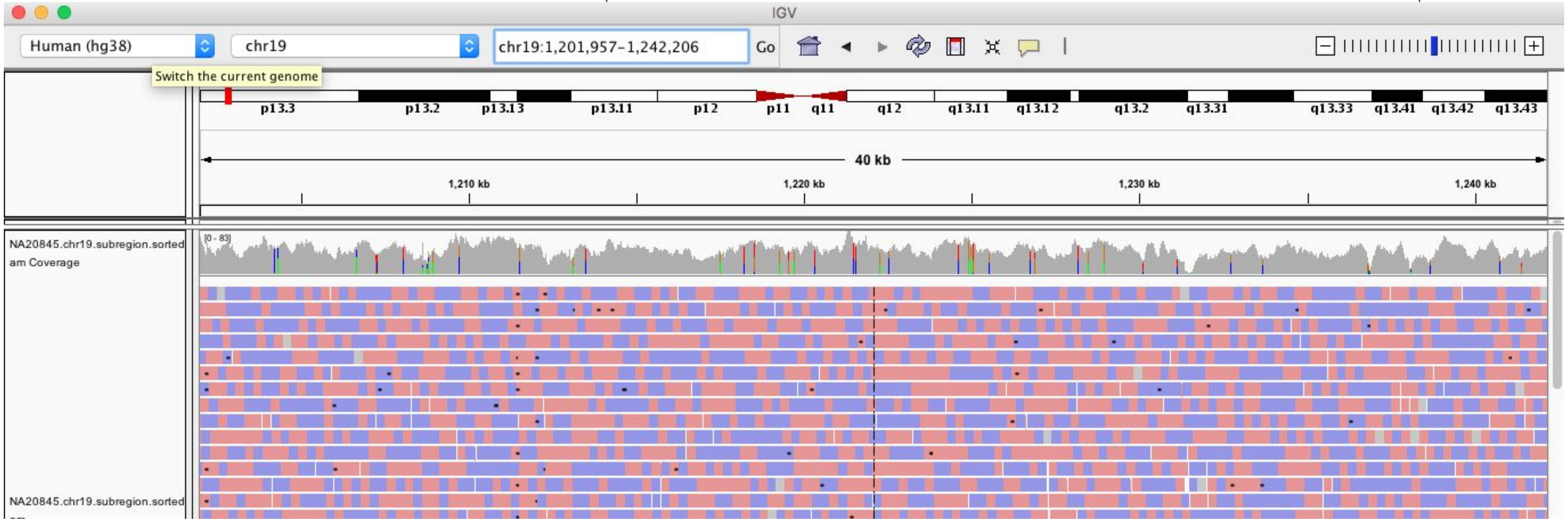
Name	Date Modified	Size	Kind
Screen Shot 2016-11-21 at 11.56.14 AM	11:56 AM	92 KB	PNG
NA20845.chr19.subregion.sorted.bam	11:53 AM	3.3 MB	Doc...ent
NA20845.chr19.subregion.sorted.bam.bai	11:53 AM	1.4 MB	Doc...ent
Screen Shot 2016-11-21 at 11.50.42 AM	11:50 AM	287 KB	PNG
Screen Shot 2016-11-21 at 11.50.32 AM	11:50 AM	288 KB	PNG
Screen Shot 2016-11-21 at 11.49.50 AM	11:50 AM	280 KB	PNG
Screen Shot 2016-11-21 at 11.49.35 AM	11:49 AM	278 KB	PNG
Screen Shot 2016-11-21 at 11.49.04 AM	11:49 AM	243 KB	PNG
Screen Shot 2016-11-21 at 11.48.44 AM	11:48 AM	160 KB	PNG

TIP: You only load the sorted.bam file

In the search box, type: chr19:1,201,956-1,242,206

Search box

Zoom tool



Advanced Research Computing

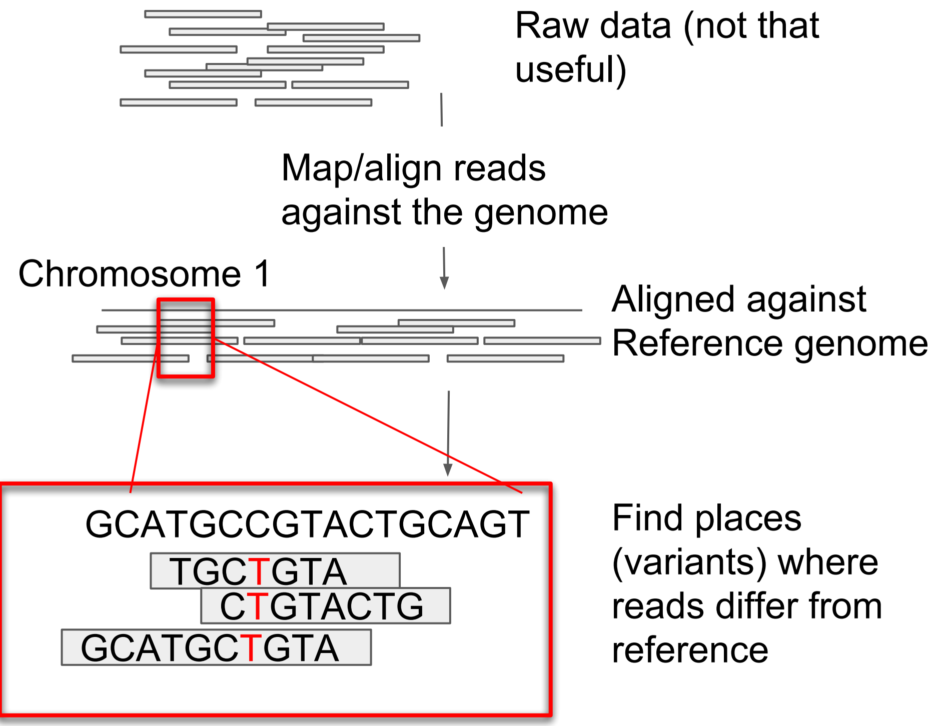
Explore some of the BAM files you have generated

Play around in IGV, check out the different settings and options for visualization

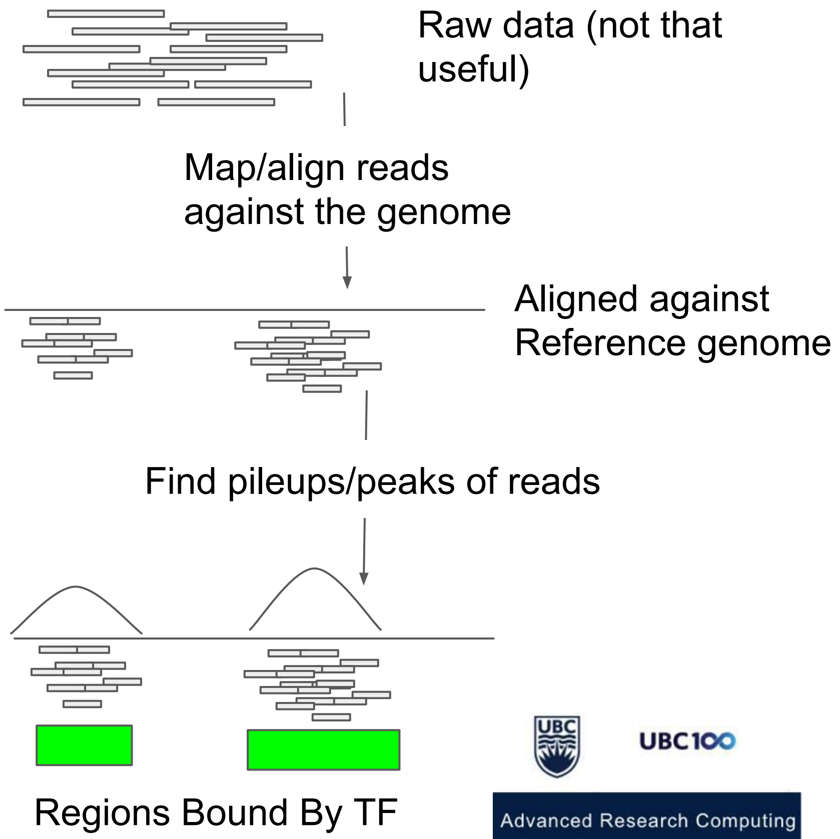
Then we will move to the last part of the course, where I show you some additional pipeline pieces but won't go into any details

Beyond Mapped Reads

Example: DNA-seq and Variant Calling


Raw data (not that useful)
Map/align reads against the genome
Chromosome 1
Aligned against Reference genome
GCATGCCGTACTGCAGT
TGCTGTA
CTGTACTG
GCATGCTGTA
Find places (variants) where reads differ from reference

Example: ChIP-seq for a Transcription Factor


Raw data (not that useful)
Map/align reads against the genome
Aligned against Reference genome
Find pileups/peaks of reads
Regions Bound By TF

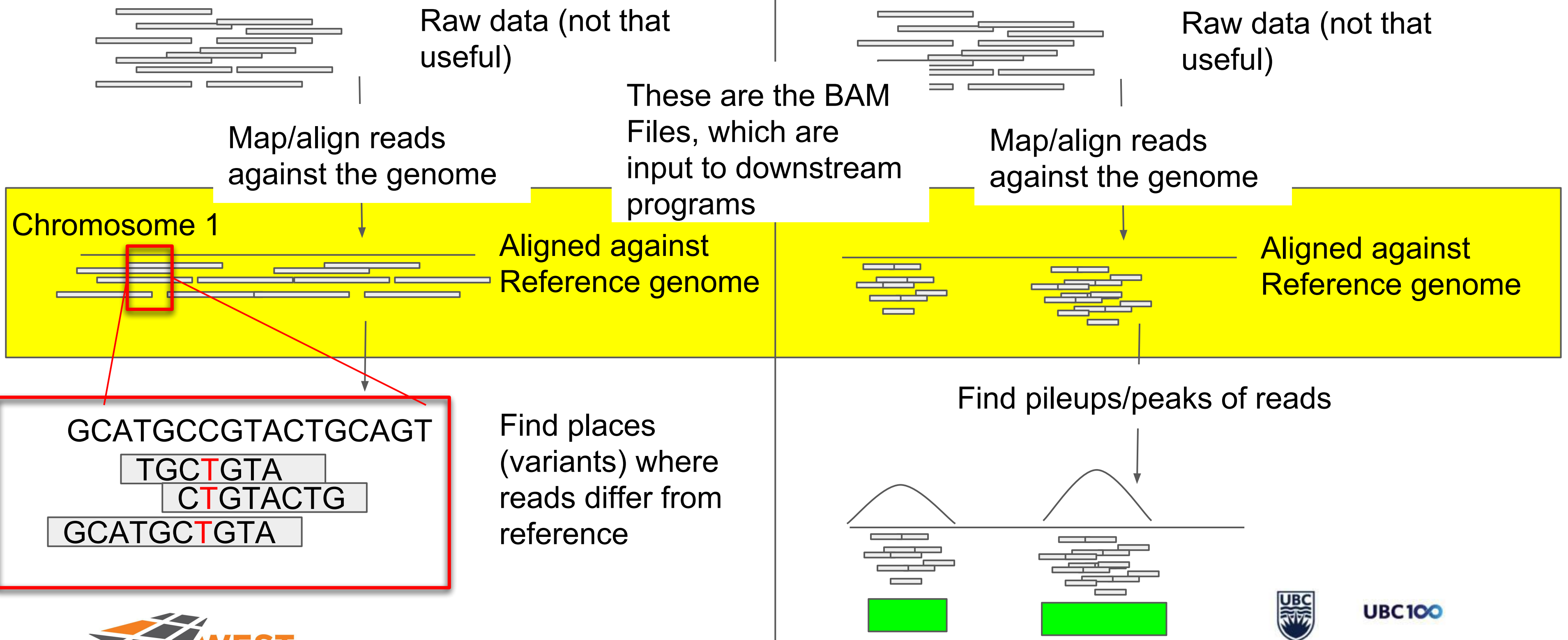
WESTGRID

UBC UBC100 Advanced Research Computing

Beyond Mapped Reads

Example: DNA-seq and Variant Calling

Example: ChIP-seq for a Transcription Factor


Raw data (not that useful)
Map/align reads against the genome
Chromosome 1
Aligned against Reference genome
GCATGCCGTACTGCAGT
TGCTGTA
CTGTACTG
GCATGCTGTA
Raw data (not that useful)
Map/align reads against the genome
Aligned against Reference genome
Find pileups/peaks of reads

These are the BAM Files, which are input to downstream programs

Find places (variants) where reads differ from reference

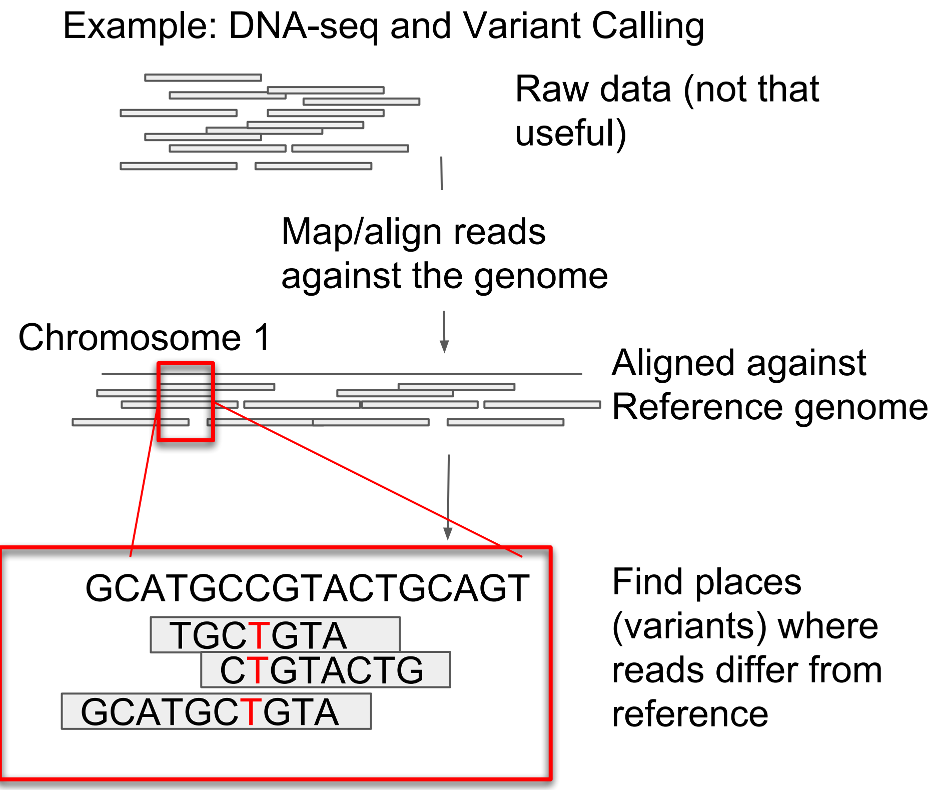
Regions Bound By TF

WESTGRID

UBC 100

Advanced Research Computing

Beyond Mapped Reads - DNA variant calling


Example: DNA-seq and Variant Calling
Raw data (not that useful)
Map/align reads against the genome
Chromosome 1
Aligned against Reference genome
GCATGCCGTACTGCAGT
TGCTGTA
CTGTACTG
GCATGCTGTA
Find places (variants) where reads differ from reference

Many tools can be used for variant calling.

We will use a simple variant caller: vcftools

While I don't have time to go over variant calling in this session, I have provided you with a script that can run variant calling on your input BAM file.

/scratch/richmonp/TRAINING/Files/SCRIPTS/Bam2VCF_BartSimpson.sh

The output of this pipeline is a VCF file, which contains variants. VCF Files can be loaded and vizualized in IGV

Beyond Mapped Reads - ChIP-seq Peak Calling

A few approaches can be used for calling "peaks" within ChIP-seq data

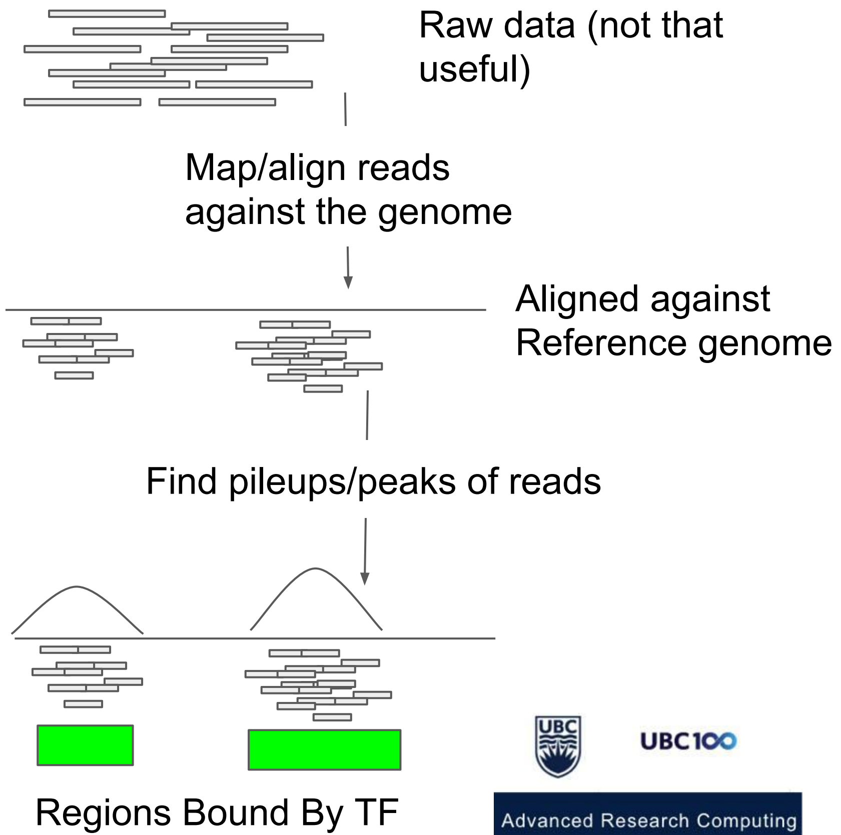
We will use the MACS2 package, which I have installed into: /scratch/richmonp/TRAINING/TOOLS/

While I don't have time to go over peak calling in this session, I have provided you with a script that can run variant calling on your input BAM file.

/scratch/richmonp/TRAINING/Files/SCRIPTS/MACS2_SRR1448786.sh

One of the outputs is a bed file and a bedgraph file (.bdg) which can be loaded and visualized in IGV

Example: ChIP-seq for a Transcription Factor


Raw data (not that useful)
Map/align reads against the genome
Aligned against Reference genome
Find pileups/peaks of reads
Regions Bound By TF

End of Lecture, what to do next

- Take a quick break
- Ask a question
- Do Problem Set 2
- Go outside and enjoy the weather

- Additional Genomics Resources:
    - https://phillip-a-richmond.github.io/Introduction-to-Genomic-Analysis/

Acknowledgements

- Phil Richmond (Teacher)
  - PhD Student Wasserman Lab, enjoys teaching
- Assorted TAs
  - Da real MVPs: Oriol, Rashedul, Robin
- WestGrid https://www.westgrid.ca/ (Jana Makar)

WESTGRID

UBC UBC100

Advanced Research Computing

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
```

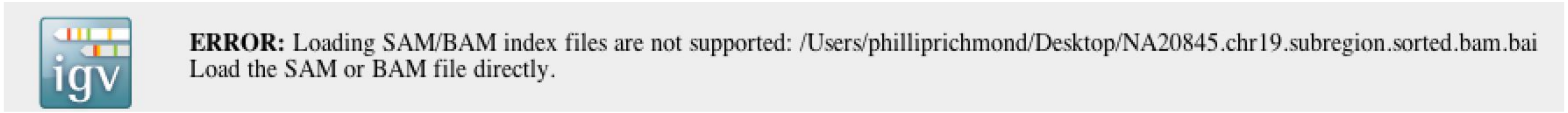
Fix: Order matters. Sort before you index

```
$ 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/philliprichmond/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