

2021 Summer bootcamp Bio Resources at CARC

CARC at USC, June 2021

Tomasz Osinski, PhD

Research Facilitator in Life Science



USC

Advanced Research Computing
Enabling scientific breakthroughs at scale

What am I doing here?

- My advisor told me to come
- I have been to a workshop like this, but I forgot most of it
- I need to analyze a lots of different samples
- Like now!
- My laptop gets too hot and is too slow to analyze a genome

What is a cluster?

- A group or bunch of something (dictionary)
- A group of computers working closely together (in computing)

Why use a cluster?

- To run many applications at once
- To run a parallel application quickly
- To share resources across a group of people
- To reduce the costs of the hardware by increasing the usage
- To have a reliable long run time when needed
- To reduce downtime due to hardware faults

Some terms

- **Head Node** – The system that controls the cluster
- **Worker (Compute) Node** – Systems that perform the computations in a cluster
- **Login Node** – System that users log into to use a cluster
- **Storage Node** – System that contains storage available for the cluster
- **Scheduler** – Software that controls when jobs are run and the node they are run on
- **Path** – Where something is in the directory structure
- **I/O** – General term for the transfer of data across a medium

Some terms

- **Queue/Partition** – A structure to which worker nodes and jobs are assigned
- **Serial** – A job that runs within one node on one processor
- **Multithreaded** – A job that with one node on more than one processor
- **Parallel** – A job that can be run across several nodes
- **Shell** – A program that users employ to type commands
- **Script** – A file that contains a series of commands that are executed
- **Job** – A chunk of work that has been submitted to the cluster

How does it work?

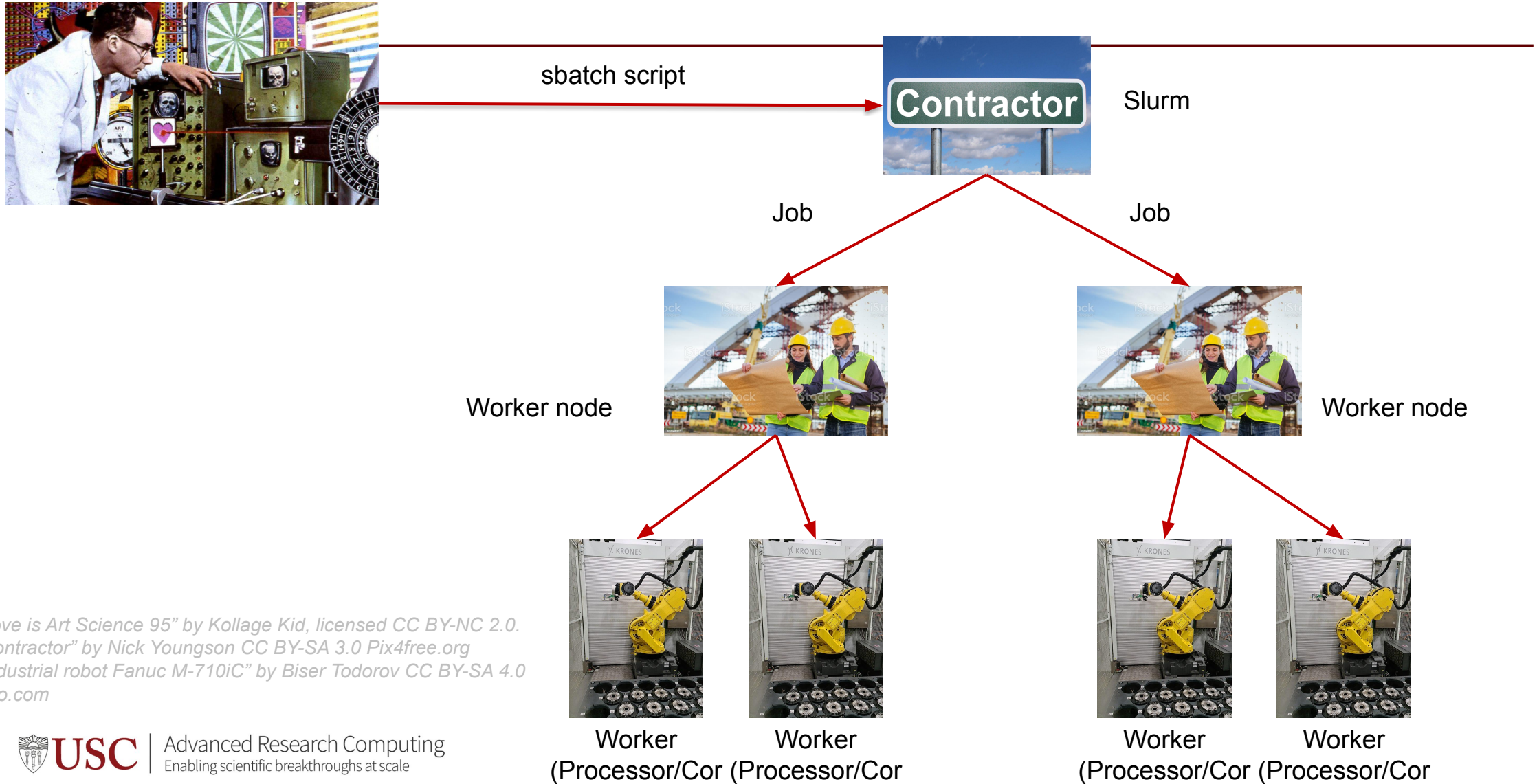


Image: "Love is Art Science 95" by Kollage Kid, licensed CC BY-NC 2.0.
Image: "Contractor" by Nick Youngson CC BY-SA 3.0 Pix4free.org
Image: "Industrial robot Fanuc M-710iC" by Biser Todorov CC BY-SA 4.0
Istockphoto.com

How does it work?

- Think of a cluster as a building contractor
- The site manager (**the job scheduler SLURM**) determines what work need to be done and in what order. He works in an office (**the head node**)
- The foremen (**worker nodes**) manage a team of workers (**processors**). The foremen manage the teams with differing specialties (**queues/partitions**)
- When you want something done, you can drop off blueprints (**a job script**) with the site manager.
- You can also say you want to talk to a worker directly, then the site manager will assign worker(s) from a foreman's team.

Resources at CARC

- Discovery (list what is there)
- Endeavour condo cluster
- Storage
- Network
- Biomedical databases or Bio Resources

Bio Resources (updated quarterly)

- **Genomes** - ready-to-use reference sequences and annotations for commonly analyzed organisms
- **Genbank** - annotated collection of all publicly available nucleotide sequences and their protein translations.
- **Genome Taxonomy Database (GTDB)** - an initiative to establish a standardized microbial taxonomy based on genome phylogeny.
- **Pfam Database** - large collection of protein families, each represented by multiple sequence alignments and hidden Markov models (HMMs).
- **TIGRFAMs** - curated multiple sequence alignments, Hidden Markov Models (HMMs) for protein sequence classification
- **UniProt** – UniProtKB (curated protein information), UniRef (closely related sequences), UniParc (all protein sequences, consisting only of unique identifiers and sequences)

What partition should I use?

- **debug** - small, short or test jobs that take less than 30m; short queue
- **main** (default) - most jobs; up to 48h runtime (2 days)
- **epyc-64** - larger multithreaded jobs; up to 48h (2 days)
- **largemem** - jobs that require huge amount of memory (up to 1TB); up to 168h (7 days)
- **oneweek** - long running jobs; up to 168h (7 days)

Log into CARC

- Open the terminal:
 - Mac: Applications>Utilities>Terminal or open Spotlight and start typing “terminal”
 - Windows: Start menu>cmd (or use PuTTY or Cygwin)
 - Linux: System tools>Terminal or Accessories>Terminal or search for Terminal
- Type “ssh ttrojan@discovery1.usc.edu”
- Enter your password
- Choose an option in Duo-2FA, and confirm your access
- Answer “No” when asked to save your password
- If successful, your prompt should look something like:
[ttrojan@discovery1 ~]

Review of Important Commands: squeue

Shows the status of jobs running in the queues

```
[osinski@discovery1 ~]$ squeue | head
      4679566      main discover sunwool  R      2:19:33      8 d23-[13,15-16],e21-14,e22-[08-09,12],e23-01
      4680126      main discover sunwool  R        39:11      8 d23-[13-16],e22-[05-06,08-09]
      4678655      main job.slur liukuang  R     11:09:20      1 d14-08
      4679445      main  1086-7B  asareh   R      4:18:00      1 d11-46
      4679444      main  1086-7B  asareh   R      4:19:31      1 d05-40
```

You can also show the status of just your jobs

```
[osinski@discovery1 ~]$ squeue -u osinski
      JOBID PARTITION      NAME      USER ST      TIME  NODES NODELIST(REASON)
      3678639    epyc-64    test_1    osinski PD       0:00      4 (Resources)
      3678721    epyc-64    test_2    osinski PD       0:00      4 (Priority)
      3675759    epyc-64    test_3    osinski  R    1-01:48:12      2 b22-[29-30]
```

CD is complete, R is running, PD is waiting to run

Review of Important Commands: sinfo

Show the properties of queues and nodes

```
[osinski@discovery1 ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug      up       30:00      6   idle a02-26,e05-[42,76,78,80],e22-13
epyc-64     up  2-00:00:00    32  alloc b22-[01-32]
main*      up  2-00:00:00    16  alloc$ d11-[02-04],e16-[01-13]
main*      up  2-00:00:00    11  maint e16-[14-24]
main*      up  2-00:00:00     4  drain* d17-[03,30],d23-[11-12]
main*      up  2-00:00:00     6  down*  d17-[39-44]
main*      up  2-00:00:00     1  drain d06-23
main*      up  2-00:00:00   236   mix
d05-[10,13-15,26,28,31-32,34-42],d06-[15-22,25-28],d11-[09-13,17-18,21,24,27-31,33-41,43-45,47],d13-[02-03,05-06,08-10],d14-[04-18],d17-[04,10-13,16-23,27-29,31-38],d18-[01-24,27-29,32-38],d22-[51-52],d23-[10,13-16],e06-[01-12,14-15,22,24],e07-[01-08,10-16,18],e13-[36-37,40-43,45-48],e15-[06-07,10-12,14,16,20,22,24],e17-[01-02,04,06-07,09-24],e19-[21,23],e20-[15-18,22,25,40],e21-[04,07-08,12,16],e22-[01,04-06,08-09]
main*      up  2-00:00:00    78  alloc
d05-[06-09,11-12,27,29-30,33],d06-24,d11-[19-20,22-23,25-26,32,42,46],d13-[04,07,11],d14-03,d17-[05-09],d18-[25-26,30-31],e06-13,e07-09,e09-18,e10-12,e11-[26-27,29,45,47],e13-[11,26,28-30,38-39,44],e15-[02,15,17-19,21,23],e21-[01-03,05-06,09-11,13,15],e22-[02-03,07,10-12,14-16],e23-[01-02]
main*      up  2-00:00:00    42   idle
d11-[14-16],d17-[14-15,24-26],e06-[16-21],e13-[31-35],e14-[41-48],e15-[01,03-05,08-09,13],e20-[19-20,23-24,36,38,42],e21-14
oneweek    up  7-00:00:00     4   mix e01-[52,76],e02-[70-71]
oneweek    up  7-00:00:00     9  alloc e01-[46,60],e02-[40-46]
oneweek    up  7-00:00:00    34   idle e01-[48,62,64],e02-[48-69,72-80]
largemem   up  7-00:00:00     2   mix a16-[02,04]
largemem   up  7-00:00:00     1  alloc a16-03
```

Review of Important Commands: scontrol

Get more details about a job

```
[osinski@discovery1 ~]$ scontrol show job 3675759
JobId=3675759 JobName=test_3
  UserId=osinski(600458) GroupId=osinski_703(91797) MCS_label=N/A
  Priority=1072 Nice=0 Account=osinski_703 QOS=normal
  JobState=RUNNING Reason=None Dependency=(null)
  Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
  RunTime=1-02:01:31 TimeLimit=2-00:00:00 TimeMin=N/A
  SubmitTime=2021-05-30T15:57:10 EligibleTime=2021-05-30T15:57:10
  AccrueTime=2021-05-30T15:57:10
  StartTime=2021-05-30T15:57:23 EndTime=2021-06-01T15:57:23 Deadline=N/A
  PreemptEligibleTime=2021-05-30T15:57:23 PreemptTime=None
  SuspendTime=None SecsPreSuspend=0 LastSchedEval=2021-05-30T15:57:23
  Partition=epyc-64 AllocNode:Sid=68.181.11.7:3273398
  ReqNodeList=(null) ExcNodeList=(null)
  NodeList=b22-[29-30]
  BatchHost=b22-29
  NumNodes=2 NumCPUs=128 NumTasks=128 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
  TRES=cpu=128,mem=496G,node=2,billing=252
  Socks/Node=* NtasksPerN:B:S:C=64:0:*:* CoreSpec=*
  MinCPUsNode=64 MinMemoryNode=2G MinTmpDiskNode=0
  Features=(null) DelayBoot=00:00:00
  OverSubscribe=NO Contiguous=0 Licenses=(null) Network=(null)
  Command=/bin/bash
  WorkDir=/home1/osinski/test_3
  MailUser=osinski@usc.edu MailType=INVALID_DEPEND,BEGIN,END,FAIL,REQUEUE,STAGE_OUT
  NtasksPerTRES:0
```

Review of Important Commands: module

Loads the necessary environment for a program

- `module avail`
 - Shows all modules available, or all the software installed
- `module load`
 - Load the environment for a program
- `module list`
 - Shows modules loaded
- `module unload`
 - Removes a loaded module
- `module purge`
 - Removes all loaded modules

Pay attention to module messages

```
module purge
```

```
module load gcc/9.2.0
```

```
module purge
```

```
module load bedtools2
```

```
module purge
```

```
module load blast-plus
```

Review: Transferring Files

- SFTP

- Cyberduck (OSX, Windows) <https://cyberduck.io>
- WinSCP (Windows) <https://winscp.net/eng/index.php>
- FileZilla (OSX, Windows, Linux) <https://filezilla-project.org>

- Globus Online

- Transfers large files reliably and easily
- Best way to get data from CARC
- go to <https://www.globus.org> in your browser and click **Log In**. Type **University of Southern California** in the box that says "Use your existing organizational login"

- Command line tools - rsync or scp

More info: <https://carc.usc.edu/user-information/user-guides/data-management/transfer-overview>

Resources

- CARC home page
 - <https://carc.usc.edu>
- Bio Resources at CARC
 - <https://carc.usc.edu/user-information/bio-resources>
- CARC User Forum
 - <https://hpc-discourse.usc.edu/categories>
- SLURM tutorials
 - <https://slurm.schedmd.com/tutorials.html>
- SLURM quick reference
 - <https://slurm.schedmd.com/pdfs/summary.pdf>

Resources

- CARC home page
 - <https://carc.usc.edu>
- Bio Resources at CARC
 - <https://carc.usc.edu/user-information/bio-resources>
- CARC User Forum □ the most value for the community!
 - <https://hpc-discourse.usc.edu/categories>
- SLURM tutorials
 - <https://slurm.schedmd.com/tutorials.html>
- SLURM quick reference
 - <https://slurm.schedmd.com/pdfs/summary.pdf>

Lets get going

- Detailed policies and directions
 - <https://carc.usc.edu/user-information/getting-started>
- Do not install software yourself, contact us
 - <https://carc.usc.edu/education-and-outreach/office-hours> (Tue, 2:30-5:00)
 - Submit a ticket! (<https://carc.usc.edu/user-support/>)
 - When we install software, it is available to everyone
- Program running slow? Submit a ticket!
- Don't know what resources to use? Submit a ticket!
- Any other questions? Submit a ticket or visit our forum

Review: Interactive Jobs

When you need to provide unpredictable input

```
[osinski@discovery1 ~]$ hostname
discovery1.usc.edu
[osinski@discovery1 ~]$ srun -p debug --pty bash
[osinski@a02-26 ~]$ hostname
a02-26.hpc.usc.edu
[osinski@a02-26 ~]$ exit
exit
[osinski@discovery1 ~]$ hostname
discovery1.usc.edu
[osinski@discovery1 ~]$
```

Review: Bash Scripts

Bash scripts are a series of commands that can be grouped together within files to accomplish a series of tasks.

This allows you to run one command instead of several successive commands.

Review: Example Bash Script

- Start an interactive job to the debug queue
- This program sleeps for 10 seconds and then prints out “Hello World”
- Make this file, give it execute permissions, and run

```
#!/bin/bash
# This program:
# sleeps for 10 seconds
# then prints "Hello World"

sleep 10

echo "Hello World"
```


FastQC

- FastQC is a quality control application for high throughput sequence data
- Checks the quality of their sequence data
- Generates an HTML report
- <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>

Prepare to Run Jobs

Copy example data to your home directory

```
mkdir bio-bootcamp
cd bio-bootcamp/
cp -r /home1/osinski/bio-bootcamp-src/data/raw-seq .
cp -r /home1/osinski/bio-bootcamp-src/data/raw-seq-ordered .
cp -r /home1/osinski/bio-bootcamp-src/data/blast .
mkdir -p results/fastqc-rawseq
mkdir results/blast
mkdir results/fastqc-rawseq-ordered
mkdir results/fastqc-rawseq-ordered-arr
mkdir results/fastqc-rawseq-unordered
```

Important Commands: sbatch

Submit a job to the cluster

- -p partition you want to submit to
 - Default is “main”
- -N Number of nodes
 - Default is 1
- -n Number of CPUs per node
 - Default is 1
- Many more options
 - <https://slurm.schedmd.com/sbatch>

Create the FastQC Job Script

Use a text editor to create a file name samplefastqc.sh that contains what follows:

```
#!/bin/bash
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --partition debug
#SBATCH --mail-user=ttrojan@usc.edu
#SBATCH --mail-type=ALL
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
#SBATCH --account=<account_id>

module purge
module load gcc/9.2.0
module load fastqc
echo "Example FastQC start"
sleep 20
fastqc -o results/fastqc-rawseq raw-seq/yeast_1_50K.fastq
echo "Example FastQC end"
```

Run the FastQC Job Script

- Submit the job

```
[osinski@discovery1 ~]$ sbatch examplefastqc.sh  
Submitted batch job 33723
```

- Check the status of the job

```
[osinski@discovery1 ~]$ squeue -u osinski
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
33723	dbeug	fastqc.s	osinski	R	0:02	1	a02-26

Check Output File for Errors

```
[osinski@discovery1 ~]$ cat slurm-33723.out
Started analysis of yeast_1_50K.fastq
Approx 5% complete for yeast_1_50K.fastq
Approx 10% complete for yeast_1_50K.fastq
Approx 15% complete for yeast_1_50K.fastq
Approx 20% complete for yeast_1_50K.fastq
Approx 25% complete for yeast_1_50K.fastq
Approx 30% complete for yeast_1_50K.fastq
Approx 35% complete for yeast_1_50K.fastq
Approx 40% complete for yeast_1_50K.fastq
Approx 45% complete for yeast_1_50K.fastq
Approx 50% complete for yeast_1_50K.fastq
Approx 55% complete for yeast_1_50K.fastq
Approx 60% complete for yeast_1_50K.fastq
Approx 65% complete for yeast_1_50K.fastq
Approx 70% complete for yeast_1_50K.fastq
Approx 75% complete for yeast_1_50K.fastq
Approx 80% complete for yeast_1_50K.fastq
Approx 85% complete for yeast_1_50K.fastq
Approx 90% complete for yeast_1_50K.fastq
Approx 95% complete for yeast_1_50K.fastq
Approx 100% complete for yeast_1_50K.fastq
Analysis complete for yeast_1_50K.fastq
```

Important Things to Note

- Job length
 - If over 24 hours, can this be split up, can threads be increased?
- Many small files
 - To be avoided!
 - Group into larger files
- Data
 - Save space by removing temp files
 - Archive data as soon as reasonable
 - Let us know if you are adding several TB of data
 - Use /scratch whenever possible for temporary files

Important Things to Note

- Make sure you are not on the login node when you launch an application
 - You can check the system you are on by typing “hostname”
- Make sure you reserve as many processors as you need
 - A mismatch here can increase your runtime or wait time
- Make sure you reserve as much RAM as needed
 - Overestimating increases wait time, underestimating crashes
- Know which resources work the best
 - Sometimes using a debug or epyc-64 is better

No HIPAA Data is
allowed on the cluster!

(we are working on that part)

SLURM Environment Variables

- `$SLURM_JOB_ID` – The job number
 - Assigned automatically by SLURM
- `$SLURM_JOB_NAME` – The name of the job
 - Similar to the script name
 - Or you can specify one with `-J`
- `$SLURM_NTASK` – Number of reserved Processors
 - Assigned automatically by SLURM
 - Value is the multiple of the `-n` and `-N` values

Multi-Processor Jobs

- The program must support it!
- Our default nodes have mostly 20 cores. Some programs loose efficiency after 8 or 16 processors.
- Wait time and run time adds up if not properly submitted
- Try “program --help” or “man program”
- Use `$SLURM_NTASKS`

Create the BLAST Job Script

```
#!/bin/bash
#SBATCH --nodes 1
#SBATCH --ntasks 10
#SBATCH --partition debug
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
#SBATCH -time 00:05:00
#SBATCH --account=<account_id>

module purge
module load gcc/9.2.0
module load blast-plus
echo "Start BLAST Job"
blastp -db swissprot \
-query blast/query.txt \
-out results/blast/results.txt \
-num_threads $SLURM_NTASKS
echo "Finish BLAST Job"
```

Replace **swissprot** with the path to the v5 of swissprot db obtained from
<https://carc.usc.edu/user-information/bio-resources/genbank>

Run the BLAST Job Script

- Submit the job

```
[ttrojan@discovery1 ~]$ sbatch blast1.job  
Submitted batch job 4773117
```

- Check the status of the job

```
[ttrojan@discovery1 ~]$ squeue -u ttrojan
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
4773117	Main	blast1.j	ttrojan	R	0:02	1	a02-d11

Check BLAST Job Stats with sacct

- sacct can get stats for a job after its completed
- <https://slurm.schedmd.com/sacct.html>

```
sacct -j 4773117 --format=JobID,State,Elapsed,NCPUS,MaxRSS
```

Check BLAST Job Stats with sacct

- sacct can get stats for a job after its completed
- <https://slurm.schedmd.com/sacct.html>

```
sacct -j 4773117 --format=JobID,State,Elapsed,NCPUS,MaxRSS
```

```
[ttrojan@discovery1 ~]$ sacct -j 4773117 --format=JobID,State,Elapsed,NCPUS,MaxRSS
```

JobID	State	Elapsed	NCPUS	MaxRSS
4773117	COMPLETED	00:00:09	10	
4773117.bat+	COMPLETED	00:00:09	10	1228K
4773117.ext+	COMPLETED	00:00:09	10	832K

GPU Jobs – Example

Use gpu partition

Reserve gpus with --gres parameter

```
#!/bin/bash
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --gres=gpu:p100:1
#SBATCH --mail-user=ttrojan@usc.edu
#SBATCH --mail-type=ALL
#SBATCH --chdir /home1/ttrojan
#SBATCH --account=<account_id>

module load gcc/8.3.0
module load cuda/10.0.130
module load motioncor2
```


Job Arrays

- A way to run the same commands on many (hundreds, thousands) of datasets/samples.
- A variable called `$SLURM_ARRAY_TASK_ID` is used to determine the element of the array being run.
- `#SBATCH --array=1-1000`
- `$SLURM_ARRAY_TASK_ID` becomes 1 in first job, 2 in second job, etc...

Bash Variables

```
cd raw-seq  
  
i=1  
  
ls -l yeast_${i}_50K.fastq  
  
i=2  
  
ls -l yeast_${i}_50K.fastq
```

Without Job Arrays – Numbered Files

```
#!/bin/bash
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --partition main
#SBATCH --time 00:05:00
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
#SBATCH --account=<account_name>

module purge
module load gcc/9.2.0
module load fastqc
echo "Starting FastQC job"

fastqc -o results/fastqc-rawseq-ordered raw-seq-ordered/yeast_1_50K.fastq
fastqc -o results/fastqc-rawseq-ordered raw-seq-ordered/yeast_2_50K.fastq
fastqc -o results/fastqc-rawseq-ordered raw-seq-ordered/yeast_3_50K.fastq
fastqc -o results/fastqc-rawseq-ordered raw-seq-ordered/yeast_4_50K.fastq
fastqc -o results/fastqc-rawseq-ordered raw-seq-ordered/yeast_5_50K.fastq
fastqc -o results/fastqc-rawseq-ordered raw-seq-ordered/yeast_6_50K.fastq

echo "Finish FastQC job"
```

Job Arrays – Numbered Files

Here is an example SLURM script for a job array. Save as fastqc_numbered_array.job

```
#!/bin/bash
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --partition main
#SBATCH --time 00:05:00
#SBATCH --array=1-6
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
#SBATCH --account=<account_name>

module purge
module load gcc/9.2.0
module load fastqc
echo "Starting FastQC job"

sleep 20

fastqc -o results/fastqc-rawseq-ordered-arr \
raw-seq-ordered/yeast_${SLURM_ARRAY_TASK_ID}_50K.fastq

echo "Finish FastQC job"
```

View Job Array

```
#squeue -u uscnetid
```

```
[ttrojan@disocvery1 bio-bootcamp]$ squeue -u ttrojan
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
1152	main	bash ttrojan	R	2:17:32	1	d05-40	
1153	main	bash ttrojan	R	2:17:12	1	d05-40	
1207_1	main	numbered ttrojan	R	0:02	1	d05-41	
1207_2	main	numbered ttrojan	R	0:02	1	d05-40	
1207_3	main	numbered ttrojan	R	0:02	1	d05-42	
1207_4	main	numbered ttrojan	R	0:02	1	d05-45	
1207_5	main	numbered ttrojan	R	0:02	1	d05-44	
1207_6	main	numbered ttrojan	R	0:02	1	d05-44	

Job Arrays – Unnumbered Files

- Start by creating a list of all of the unnumbered filenames

```
ls raw-seq/ > unnumbered-filenames.txt
```

Job Arrays – Unnumbered Files (cont)

Here is an example slurm array script for fastqc jobs that have unnumbered filenames.

```
#!/bin/bash
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --partition main
#SBATCH --time 00:05:00
#SBATCH --array=1-6
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
#SBATCH --account=<account_name>

module purge
module load gcc/9.2.0
module load fastqc echo "Starting FastQC job"

sleep 20

line=$(sed -n -e "$SLURM_ARRAY_TASK_ID p" unnumbered-filenames.txt)

fastqc -o results/fastqc-rawseq-unordered raw-seq/${line}
echo "Finish FastQC job"
```

Job Dependencies

- Instructions on how jobs relate to other jobs
- Useful for if you want to run a series of jobs that depend on the output from other jobs
- Examples:

-d depend=afterok:jobid

Starts after jobid has finished without errors.

-d depend=afterok:jobid,before:jobid2

Starts after jobid is finished, but not until jobid2 has started.

-d depend:afterok:jobid

-d depend:afterok:jobid2

Starts after both jobid and jobid2 have finished.

-d depend=afterokarray:jobid

Starts after the job array jobid has finished without errors.

Job Dependencies

- Why would you do this:
 - Mostly for job pipelines, a series of programs that depend on each other's output that are all submitted at once.

Example:

Step 1:

```
[ttrojan@discovery1 ~]$ sbatch preprocessing-step.sh  
Submitted batch job 18866
```

Step 2:

```
[ttrojan@discovery1 ~]$ sbatch -d after:18866 job-array-step.sh  
Submitted batch job 18870
```

Step 3:

```
[ttrojan@discovery1 ~]$ sbatch -d afterok:18870 postprocessing-step.sh  
Submitted batch job 18867
```

What is Wrong

```
#!/bin/bash
# -----SLURM Parameters-----
#SBATCH --partition main
#SBATCH --ntasks 1
#SBATCH --mem=1g
#SBATCH --nodes 1
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
# -----Load Modules-----
# -----Commands-----
python3 /home1/ttrojan/script.py
```

What is Wrong

The module is not loaded

```
#!/bin/bash
# -----SLURM Parameters-----
#SBATCH --partition main
#SBATCH --ntasks 1
#SBATCH --mem=1g
#SBATCH --nodes 1
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
# -----Load Modules-----
module purge
module load gcc/9.2.0
module load python/3.7.6
# -----Commands-----
python3 /home1/ttrojan/script.py
```

What is Wrong II

```
#!/bin/bash
# -----SLURM Parameters-----
#SBATCH --partition main
#SBATCH --ntasks 20
#SBATCH --mem=10g
#SBATCH --nodes 1
# -----Load Modules-----
module purge
module load gcc/9.2.0
module load blast-plus
# -----Commands-----
blastn -query fasta.file -db database_name -outfmt 6 \
-num_alignments 1 -num_descriptions 1 -out output_file
```

What is Wrong II

Number of processors and no working directory

```
#!/bin/bash
# -----SLURM Parameters-----
#SBATCH --partition main
#SBATCH --ntasks 20
#SBATCH --mem=10g
#SBATCH --nodes 1
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
# -----Load Modules-----
module purge
module load gcc/9.2.0
module load blast-plus
# -----Commands-----
blastn -query fasta.file -db database name -outfmt 6 num_alignments 1 \
-num_descriptions 1 -out output_file -num_threads 20
```

What is Wrong II

Better to use \$SLURM_NTASKS

```
#!/bin/bash
# -----SLURM Parameters-----
#SBATCH --partition main
#SBATCH --ntasks 20
#SBATCH --mem=10g
#SBATCH --nodes 1
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
# -----Load Modules-----
module purge
module load gcc/9.2.0
module load blast-plus
# -----Commands-----
blastn -query fasta.file -db database name -outfmt 6 num alignments 1 \
-num_descriptions 1 -out output_file -num_threads $SLURM_NTASKS
```

What is Wrong III

```
#!/bin/bash
# -----SLURM Parameters-----
#SBATCH --partition main
#SBATCH --ntasks 1
#SBATCH --mem=200g
#SBATCH --nodes 1
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
# -----Load Modules-----
module purge
module load gcc/8.3.0
module load R
# -----Commands-----
Rscript /home1/ttrojan/R_example.R
```

What is Wrong III

Wrong partition/mem requirements too high

```
#!/bin/bash
# -----SLURM Parameters-----
#SBATCH --partition epyc-64
#SBATCH --ntasks 1
#SBATCH --mem=200g
#SBATCH --nodes 1
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
# -----Load Modules-----
module purge
module load gcc/8.3.0
module load R
# -----Commands-----
Rscript /home1/ttrojan/R_example.R
```


What is Wrong IV

```
#!/bin/bash
# -----SLURM Parameters-----
#SBATCH --partition main
#SBATCH --nodes 1
#SBATCH --mem=4g
#SBATCH --ntasks 1
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
# -----Load Modules-----
module purge
module load gcc/8.3.0
Module load cuda/10.0.130
module load motioncor2
# -----Commands-----
python /home1/ttrojan/motioncor2.job
```

What is Wrong IV

Gpu resources not specified

```
#!/bin/bash
# -----SLURM Parameters-----
#SBATCH --partition main
#SBATCH --gres=gpu:p100:1
#SBATCH --nodes 1
#SBATCH --mem=4g
#SBATCH --ntasks 1
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
# -----Load Modules-----
module purge
module load gcc/8.3.0
Module load cuda/10.0.130
module load motioncor2
# -----Commands-----
python /home1/ttrojan/motioncor2.job
```

What is Wrong V

```
# -----SLURM Parameters-----
#SBATCH --partition main
#SBATCH --ntasks 1
#SBATCH --mem=15g
#SBATCH --nodes 1
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
# -----Load Modules-----
module purge
module load gcc/9.2.0
module load samtools
# -----Commands-----
samtools stats /home1/ttrojan/example.bam
```

What is Wrong V

No bash shebang line, `#!/bin/bash`

Can use long names for SBATCH parameters

```
#!/bin/bash
# -----SLURM Parameters-----
#SBATCH --partition main
#SBATCH --ntasks 1
#SBATCH --mem=15g
#SBATCH --nodes 1
#SBATCH --chdir /home1/ttrojan/bio-bootcamp
# -----Load Modules-----
module purge
module load gcc/9.2.0
module load samtools
# -----Commands-----
samtools stats /home1/ttrojan/example.bam
```

Genome mapping and tools – overview and example

- Biases in NGS technology
- Quality control
- Error correction
- Read mapping
- Useful tools

Known Biases

- Sequencing errors at 3'-end of reads

Known Biases

- Sequencing errors at 3'-end of reads
- CG-rich regions are higher covered

Known Biases

- Sequencing errors at 3'-end of reads
- CG-rich regions are higher covered
- G -> T and A -> C errors

Known Biases

- Sequencing errors at 3'-end of reads
- CG-rich regions are higher covered
- G -> T and A -> C errors
- Sequences preceding errors are G-rich

Known Biases

- Sequencing errors at 3'-end of reads
- CG-rich regions are higher covered
- G -> T and A -> C errors
- Sequences preceding errors are G-rich
- Nucleotide content bias across the read

Known Biases

- Sequencing errors at 3'-end of reads
- CG-rich regions are higher covered
- G -> T and A -> C errors
- Sequences preceding errors are G-rich
- Nucleotide content bias across the read
- PCR biases

Known Biases

- Sequencing errors at 3'-end of reads
- CG-rich regions are higher covered
- G -> T and A -> C errors
- Sequences preceding errors are G-rich
- Nucleotide content bias across the read
- PCR biases
- Mappability bias

Known Biases

- Sequencing errors at 3'-end of reads
- CG-rich regions are higher covered
- G -> T and A -> C errors
- Sequences preceding errors are G-rich
- Nucleotide content bias across the read
- PCR biases
- Mappability bias
- Higher coverage of the 3'-end

Known Biases

- Sequencing errors at 3'-end of reads
- CG-rich regions are higher covered
- G -> T and A -> C errors
- Sequences preceding errors are G-rich
- Nucleotide content bias across the read
- PCR biases
- Mappability bias
- Higher coverage of the 3'-end
- Contamination by under-spliced RNAs

Known Biases

- Sequencing errors at 3'-end of reads
- CG-rich regions are higher covered
- G -> T and A -> C errors
- Sequences preceding errors are G-rich
- Nucleotide content bias across the read
- PCR biases
- Mappability bias
- Higher coverage of the 3'-end
- Contamination by under-spliced RNAs
- Influence of RNA secondary structure

Known Biases

- Sequencing errors at 3'-end of reads
- CG-rich regions are higher covered
- G -> T and A -> C errors
- Sequences preceding errors are G-rich
- Nucleotide content bias across the read
- PCR biases
- Mappability bias
- Higher coverage of the 3'-end
- Contamination by under-spliced RNAs
- Influence of RNA secondary structure
- Coverage non-uniformity across transcripts

Quality control and error
correction!

FASTQ format

- FASTQ format is a standard format for storing reads.

```
@SEQ_ID
GATTTGGGTTCTAAGTAAAGTCTGATTAAGTAAATCTATTGTTAACTTACAGTTT
+
!''*(((('**+))%$%—)(%%%)1**~+*'''))**55CCF>>>>>CCCCCCCCG5
```

← Raw sequence

← Quality values

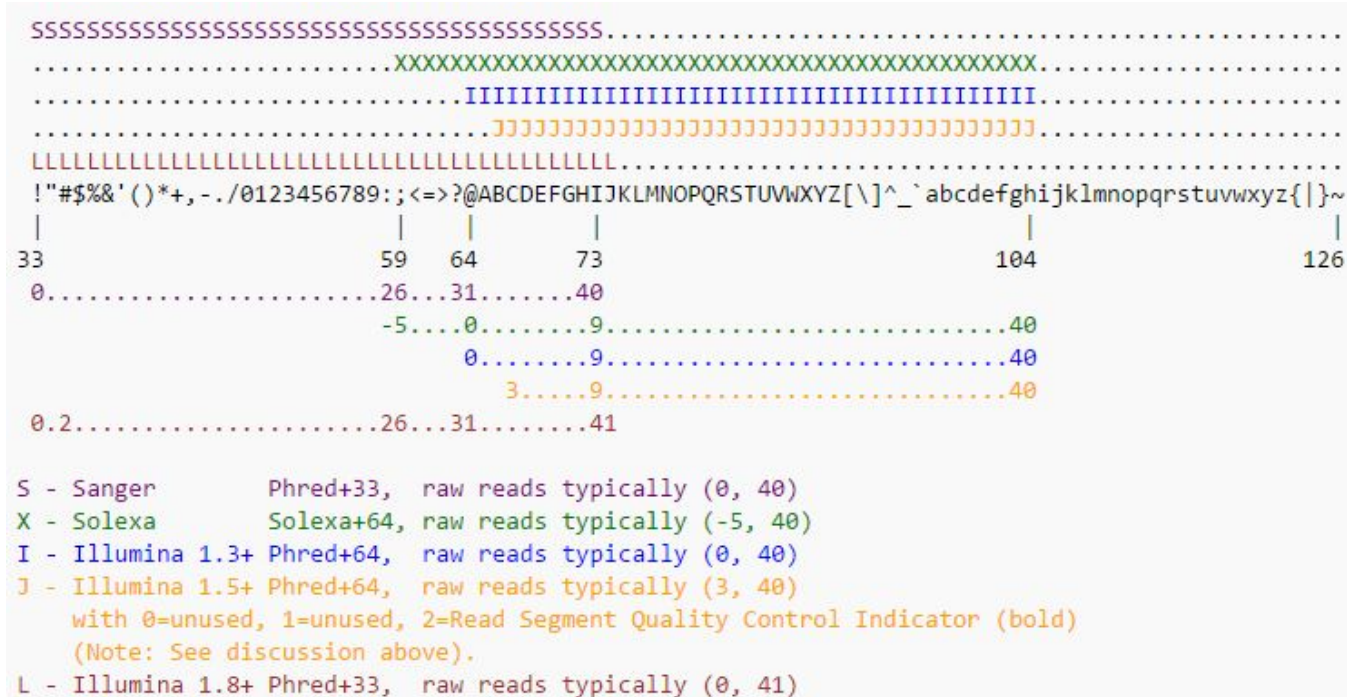
- Illumina sequence identifier:

```
@EAS139:136:-C/8bVJ:2:2184:15343:197393 1:Y:18:ATCAG
```

EAS139	the unique instrument name
136	the run id
FC708VJ	the flowcell id
2	flowcell lane
2184	tile number within the flowcell lane
15343	x-coordinate of the cluster within the tile
197393	y-coordinate of the cluster within the tile
1	the member of a pair, 1 or 2 (paired-end or mate-pair reads only)
Y	Y if the read is filtered, N otherwise
18	0 when none of the control bits are on, otherwise it is an even number
ATCAG	index sequence

FASTQ format

- Quality values are encoded as ASCII symbols. The range of values depends on the technology:



- A quality value Q (*Phred Quality Score*) is an integer mapping of p (the probability that the corresponding base call is incorrect):

$$Q_{\text{sanger}} = -10 \log_{10} p$$

Quality control

FastQC program is the most popular tool to check quality of reads.

- Number of reads
- Quality of reads
- Nucleotide content
- Percent of duplicated reads
- Overrepresented sequences

FastQC output



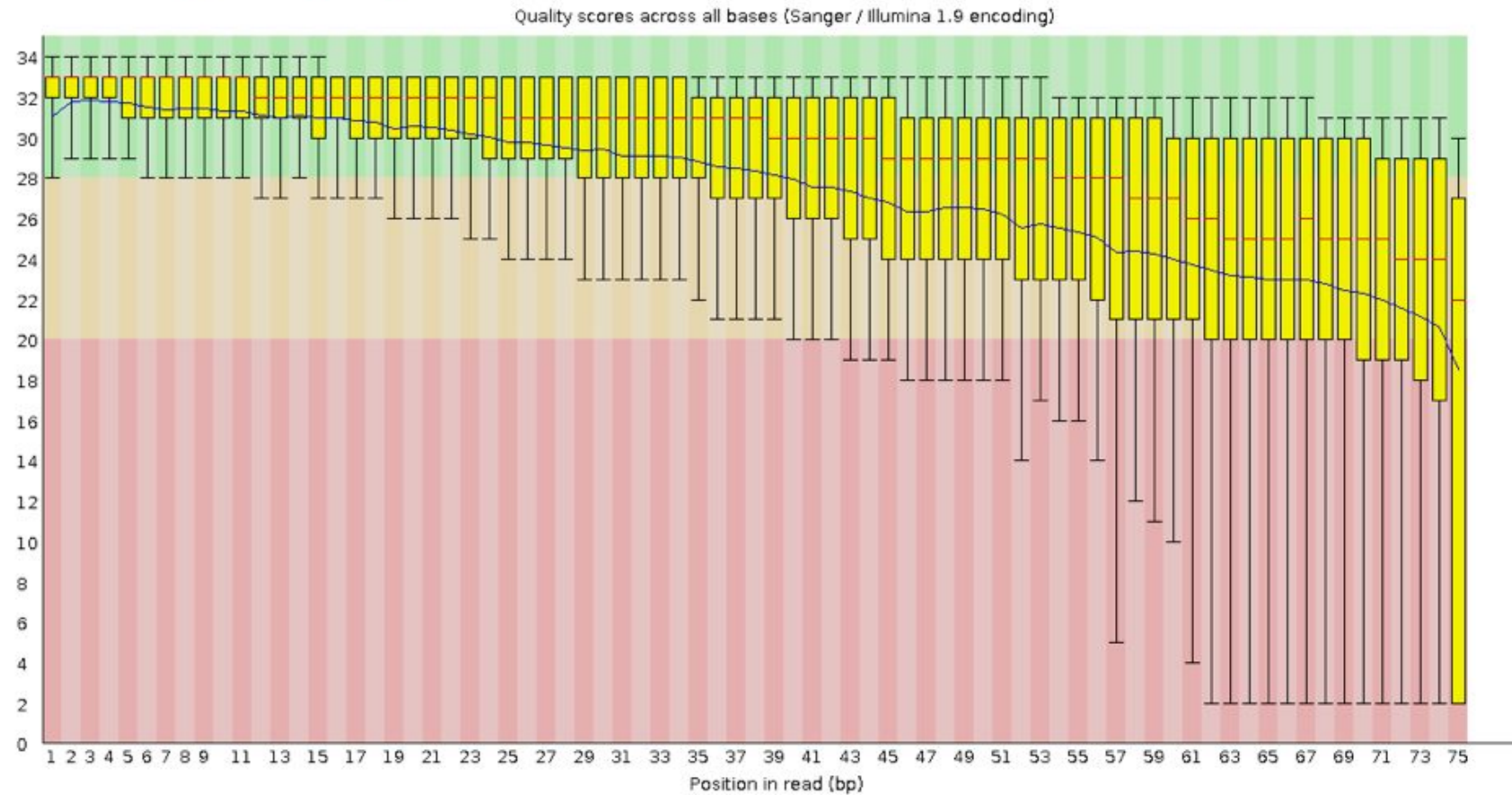
Basic Statistics

Measure	Value
Filename	fastq_data.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	16772669
Sequence length	75
%GC	45

[Back to summary](#)

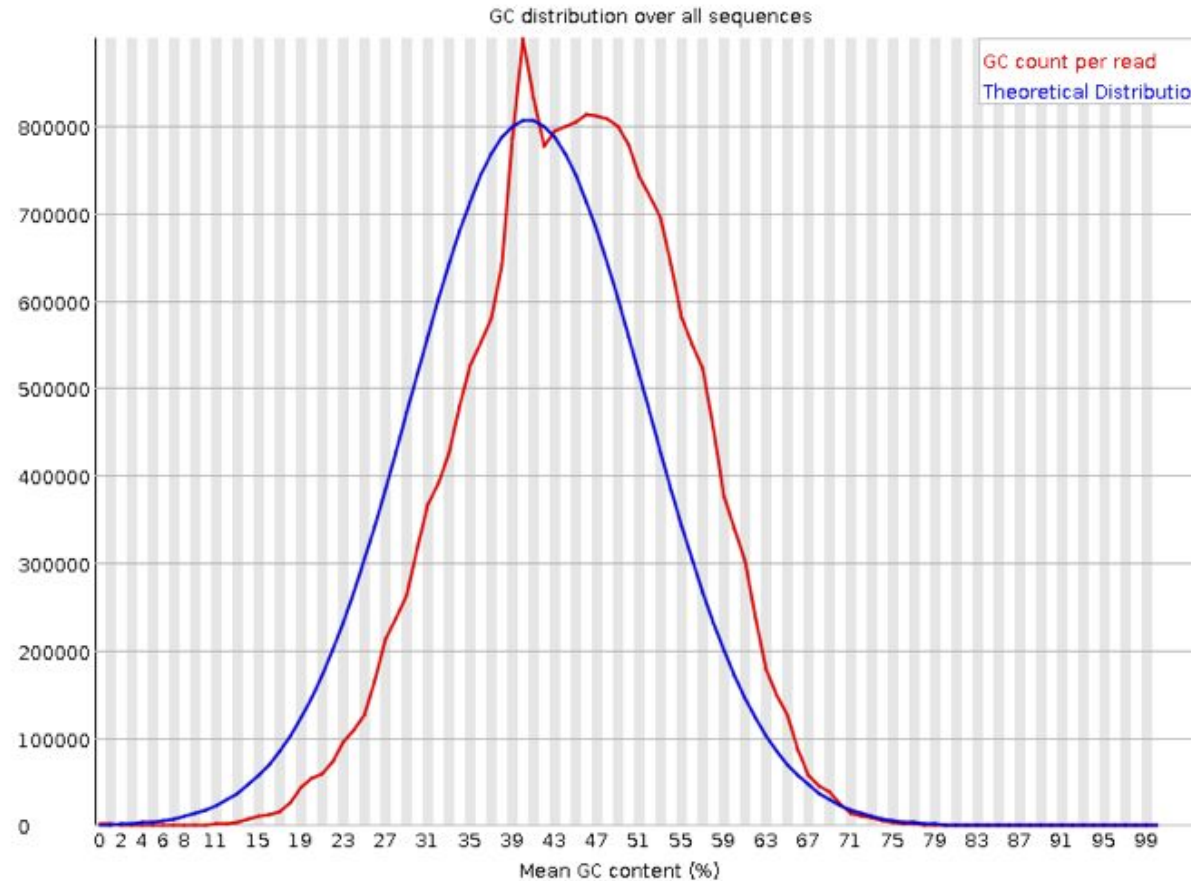
FastQC output

❌ Per base sequence quality



FastQC output

✖ Per sequence GC content



The presence of a spike around 40% indicates that sequences with 40% GC-content are over-represented.

! Overrepresented sequences

[Back to summary](#)

Possible reasons:

- 
- USC | Advanced Research Computing
Enabling scientific breakthroughs at scale

Error correction

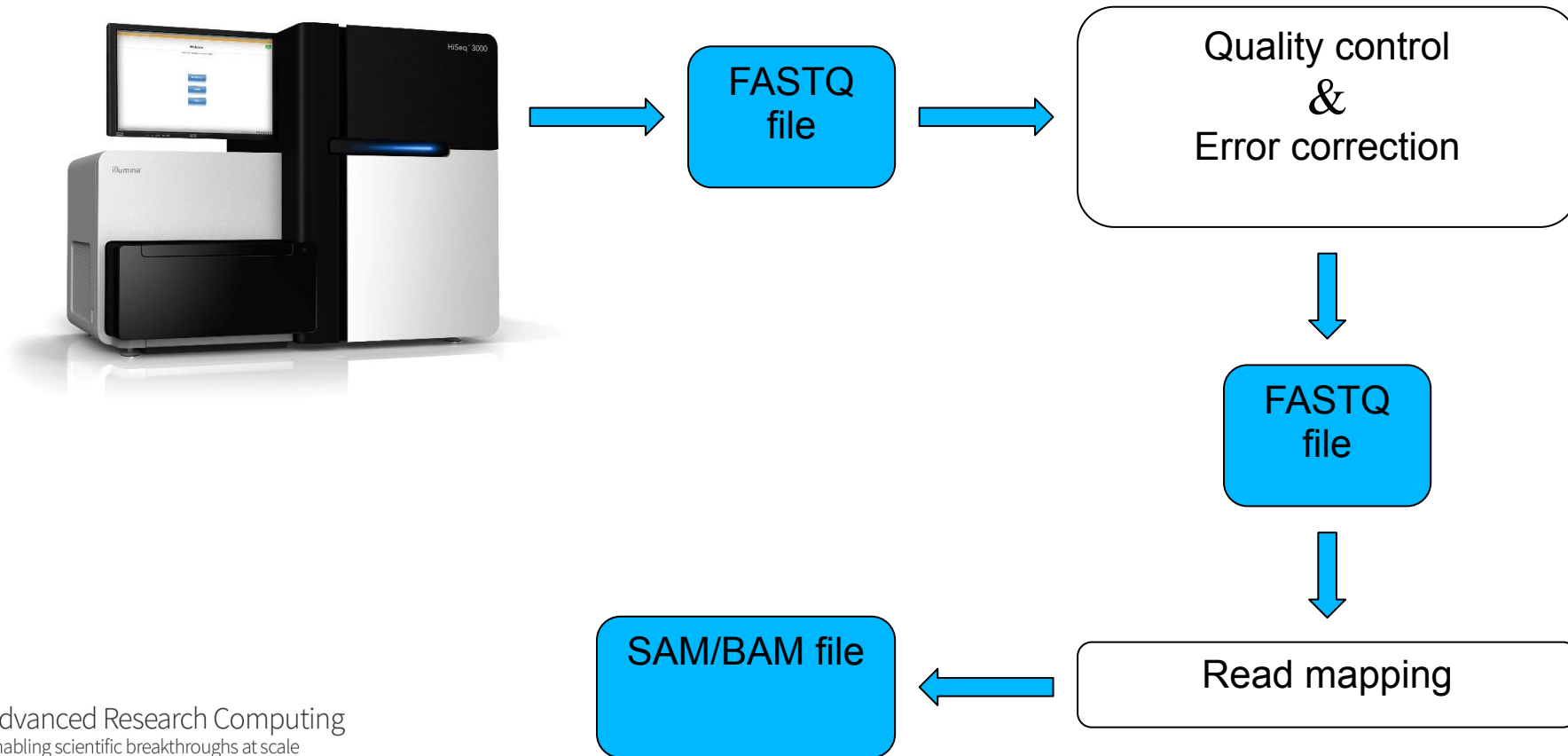
- Trimming
 - Remove adapters
 - Trim low-quality nucleotides at the 3'-end of reads
 - *Trimmomatic* software
- Correction of errors
 - Algorithms of error correction are based on k-mer counts:

```
ATCGTAGCTATTCGATCTAGATGCATCG
ATCGTAGCTATTCGATCTAGATGCATCG
ATCGTAGCTATTCGATCTAGATGCATCG
ATCGTAGCTATTCGATCTAGATGCATCG
ATCGTAGCTATTCGATCTAGATGCATCG
ATCGTAGCTATTCGATTTAGATGCATCG
ATCGTAGCTATTCGATCTAGATGCATCG
ATCGTAGCTATTCGATCTAGATGCATCG
```

- *Blooco*, *Lighter* software

Read mapping

- The genome of the sequenced organism is known ('reference genome').
- *D. melanogaster*, *H. sapiens*, *S. cerevisiae*, ...



Read mapping

- Aim: to find coordinates of reads in the reference genome.
- Challenges:
 - Millions of short sequences
 - Sequences are often paired
 - Errors are not randomly distributed
- Most popular programs are bowtie and bwa (both use Burrows-Wheeler Transform algorithm). Two-step approach:
 - Create an index for the reference genome (one time for one genome).
 - Map reads to the reference genome using this index.

bowtie

- The first version of *bowtie* [Langmead et al. 2009] is optimal for:
 - short reads (under 50 bp)
 - reads without indels (insertions/deletions)
- *bowtie2* [Langmead & Salzberg 2012] is optimal for:
 - long reads (more than 50 bp)
 - reads with indels
 - various alignment options
- Each version has its own index file format (*bowtie-build* / *bowtie2-build* tools).
- A popular RNA-seq analysis toolset (*tophat*, *cufflinks*) is based on *bowtie* / *bowtie2*.

bwa

- bwa backtrack [Li, Durbin 2009]:
 - for short reads (< 100bp)
- bwa bwasw [Li, Durbin 2010]:
 - for long reads (70bp - 1Mbp)
 - short indels
- bwa mem [Li 2013]:
 - for long reads (70bp - 1Mbp)
 - faster and more efficient

SAM/BAM format

- Suppose we have an alignment:

```
Coord      12345678901234 5678901234567890123456789012345
ref        AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

+r001/1      TTAGATAAAGGATA*CTG
+r002        aaaAGATAA*GGATA
+r003        gcctaAGCTAA
+r004                ATAGCT.....TCAGC
-r003                ttagctTAGGC
-r001/2                CAGCGGCAT
```

```
@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;
r004   0 ref 16 30 6M14N5M * 0 0 ATAGCTTCAGC *
r003 2064 ref 29 17 6H5M * 0 0 TAGGC * SA:Z:ref,9,+,5S6M,30,1;
r001 147 ref 37 30 9M = 7 -39 CAGCGGCAT * NM:i:1
```

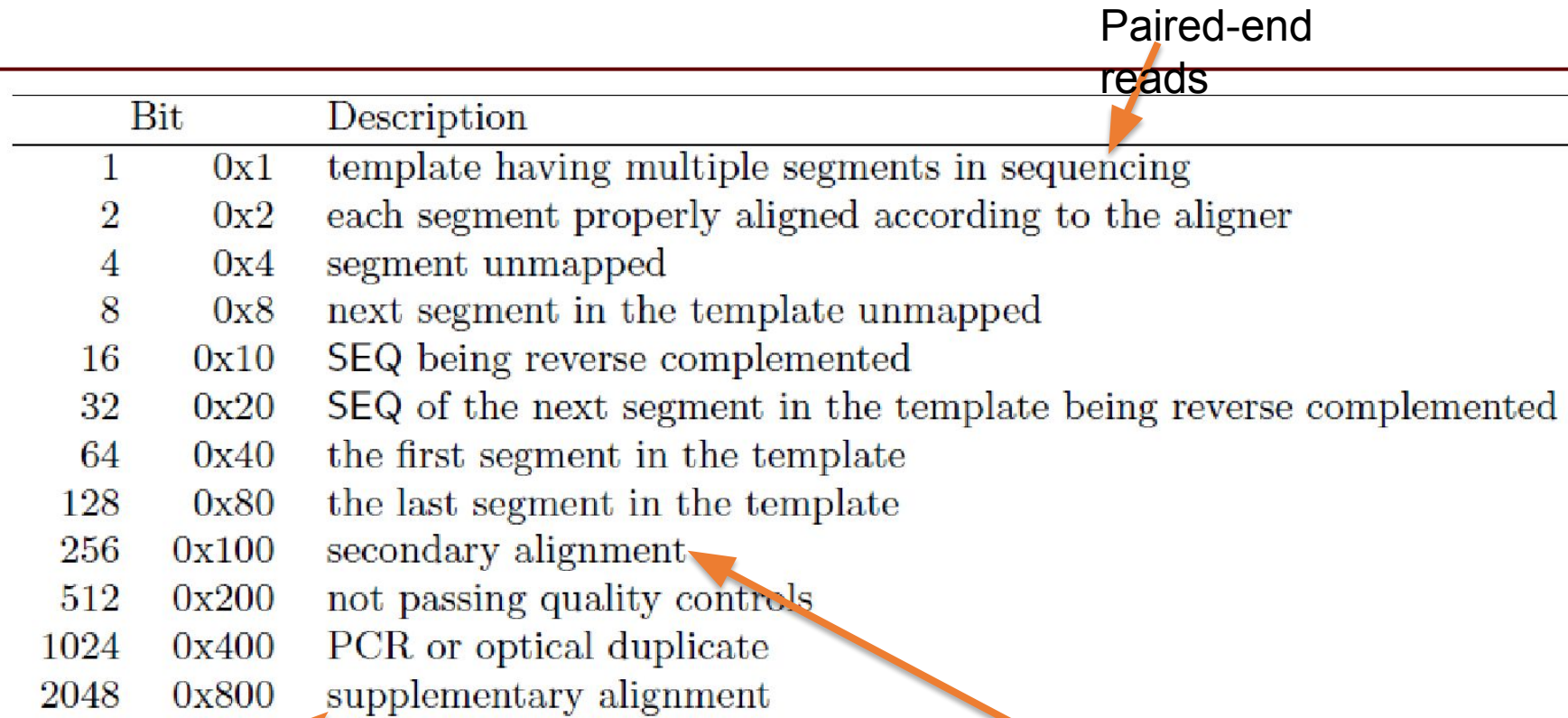
SAM/BAM format

- Each line represents the alignment of one read.
- Each line has 11 mandatory fields:

Col	Field	Type	Regex/Range	Brief description
1	QNAME	String	[!-?A-~]{1,255}	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

- FLAG: next slide.
- MAPQ: MAPping Quality. It equals $-10 \log_{10} \text{Pr}\{\text{mapping position is wrong}\}$, rounded to the nearest integer. A value 255 indicates that the mapping quality is not available.

FLAG field



Paired-end reads

Bit	Description
1	0x1 template having multiple segments in sequencing
2	0x2 each segment properly aligned according to the aligner
4	0x4 segment unmapped
8	0x8 next segment in the template unmapped
16	0x10 SEQ being reverse complemented
32	0x20 SEQ of the next segment in the template being reverse complemented
64	0x40 the first segment in the template
128	0x80 the last segment in the template
256	0x100 secondary alignment
512	0x200 not passing quality controls
1024	0x400 PCR or optical duplicate
2048	0x800 supplementary alignment

Chimeric alignment
(different
chromosomes)

Multiple mapped read, and
this is not the best
alignment

SamTools package

- A set of utilities for processing SAM/BAM files.

- **view**

Convert a bam file into a sam file.

```
samtools view sample.bam > sample.sam
```

Convert a sam file into a bam file.

```
samtools view -bS sample.sam > sample.bam
```

-b: output
BAM

-S: read SAM

Extract all the reads aligned to the range specified. An index of the input file is required.

```
samtools view sample_sorted.bam "chr1:10-13"
```

The same reads as above, but instead of displaying, write the reads to a new bam file.

```
samtools view -h -b sample_sorted.bam "chr1:10-13" > tiny_sorted.bam
```

- **sort**

```
samtools sort unsorted_in.bam sorted_out
```

- **index**

```
samtools index sorted.bam (creates an index file, sorted.bam.bai)
```

SamTools package

- **flagstat – report basic statistics**

```
samtools flagstat sample.bam
```

An example of output:

```
4198456 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 duplicates
4022089 + 0 mapped (95.80%:-nan%)
4198456 + 0 paired in sequencing
2099228 + 0 read1
2099228 + 0 read2
3796446 + 0 properly paired (90.42%:-nan%)
4013692 + 0 with itself and mate mapped
8397 + 0 singletons (0.20%:-nan%)
167574 + 0 with mate mapped to a different chr
72008 + 0 with mate mapped to a different chr (mapQ>=5)
```

- **faidx – index a FASTA file**

```
samtools faidx ref.fasta (creates an index file ref.fasta.fai)
```

- **merge – merge several BAM files into one**

```
samtools merge out.bam in1.bam in2.bam
```

UCSC genome browser

<http://genome.ucsc.edu/>

BedTools package

- `bamtobed` - Convert BAM alignments to BED (& other) formats.
 - `bamtofastq` - Convert BAM records to FASTQ records.
 - `bedtobam` - Convert intervals to BAM records.
 - `closest` - Find the closest, potentially non-overlapping interval.
 - `complement` - Extract intervals `_not_` represented by an interval file.
 - `coverage` - Compute the coverage over defined intervals.
 - `genomecov` - Compute the coverage over an entire genome.
 - `getfasta` - Use intervals to extract sequences from a FASTA file.
 - `intersect` - Find overlapping intervals in various ways.
 - `shuffle` - Randomly redistribute intervals in a genome.
 - `sort` - Order the intervals in a file.
-

bedtools intersect

- Report the intervals that represent overlaps between your two files:

```
bedtools intersect -a cpg.bed -b exons.bed
```

- Report the original feature in each file:

```
bedtools intersect -a cpg.bed -b exons.bed -wa -wb
```

- How many base pairs of overlap were there?

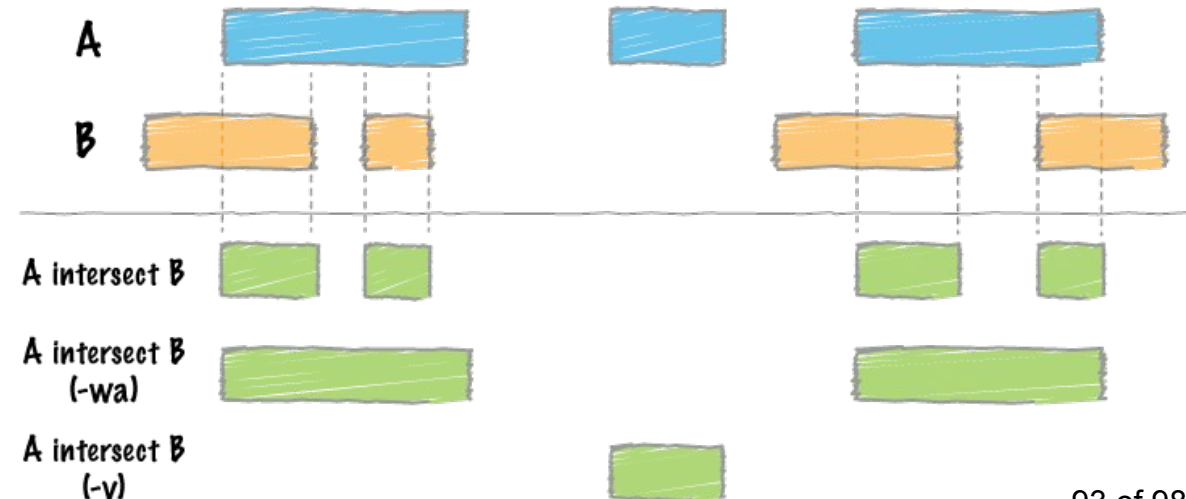
```
bedtools intersect -a cpg.bed -b exons.bed -wo
```

- Counting the number of overlapping features:

```
bedtools intersect -a cpg.bed -b exons.bed -c
```

- Find features that DO NOT overlap:

```
bedtools intersect -a cpg.bed -b exons.bed -v
```



How to get help?

-
- ~~Read the manual!~~
 - Can't find the answer? Read it again.
 - Use Google 😊
 - Most likely your problem was encountered before, and solved by someone else.
 - Ask for help at specialized forums:
 - seqanswers.com – the most popular
 - biostars.com

Exercise

- There are paired reads of some DNA sequencing experiment of the human sample:
`bio-bootcamp/R1.fastq.gz`
`bio-bootcamp/R2.fastq.gz`
- You will study some particular region of the human genome
- Map reads to the human reference genome (version hg19 – find path on carc.usc.edu’ Bio Resources)
- Extract reads that map to your region only
- Upload the reads to UCSC genome browser as a custom track
- count the number of insertions and deletions in SAM file

How To: Mapping

- load bowtie2 program:

```
module purge
module load gcc/9.2.0
module load bowtie2
```

- Copy sequence of a chromosome your region is located at as a FASTA file

```
/project/biodb/genomes/Homo_sapiens/UCSC/hg19/Sequence/Chromosomes/  
chr*.fa.gz
```

- Decompress the file using `gunzip`
- Map reads to this chromosome using `bowtie2` with the standard parameters

Don't forget to make an index (`bowtie-build2`) of the chromosome before mapping!

- You will get a SAM file as an output.
- Count the number of insertions and deletions in SAM file (use `cut`)

How To: Extracting reads

- load BedTools:

```
module purge
```

```
module load gcc/9.2.0
```

```
module load bedtools2
```

- Create a tab-delimited BED file with the coordinates of your region:

```
chr21 1000000 2000000
```

- Convert SAM file with mapped reads to BAM file using `samtools view`
- Use `bedtools intersect` to extract the reads from the BAM file

You'll need a BED file to upload the result to UCSC genome browser, so figure out how to make `bedtools intersect` to produce an output in BED format.

How To: UCSC custom track

- Upload the BED file to UCSC genome browser

‘Add custom track’ button ☐ Choose file ☐ Submit

Thanks to the whole CARC team:

BD Kim
Bill Jendrzek
Jimi Chu
Asya Shklyar
James K Hong
Chris Taylor
Derek Strong
Cesar Sul
Marco Olguin
Andrea Renney
Ryan Sim
Carolina Jin

Reference material:

- HPCBio (Holmes J., Clark L., Drnevicch J., Valizadegan N.)
- CNRG (Davidson D., Leigh J.)
- Skoltech (Khrameeva E.)

Mapping exercise: Answers

```
#!/bin/bash
```

```
#SBATCH --partition main
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --time 01:00:00
#SBATCH --chdir /home1/ttrojan/bio-bootcamp/exercise
#SBATCH --account <account_id>
#SBATCH --mem 4g
```

```
module purge
module load gcc/9.2.0
module load bowtie2
module load bedtools2
cd /home1/ttrojan/bio-bootcamp/exercise
cp bio-bootcamp-src/R*.gz ./
gunzip R1.fastq.gz
gunzip R2.fastq.gz
```