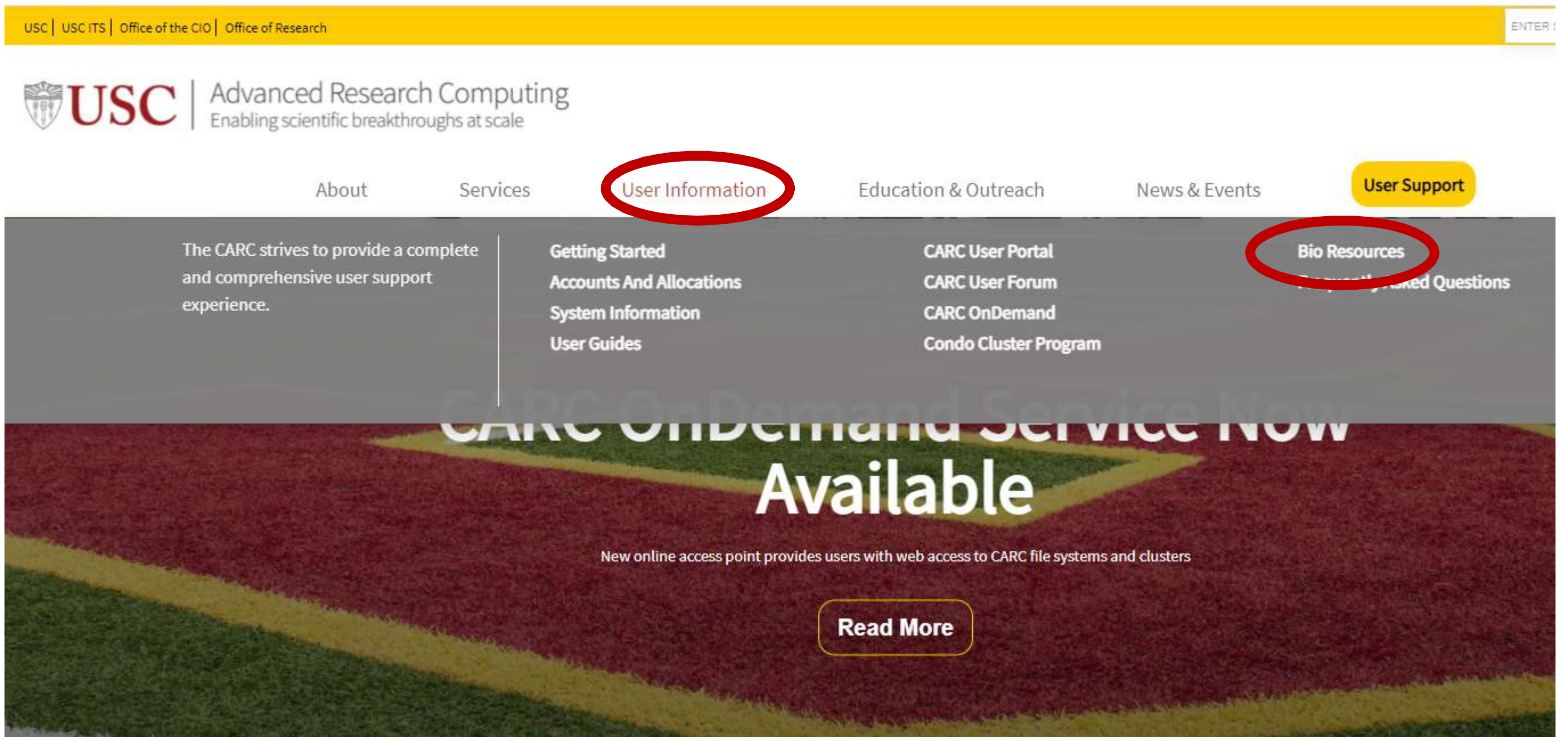
Bio Resources at CARC

Center for Advanced Research Computing
University of Southern California
Tomasz Osinski, PhD
Research Facilitator in Life Science

USC | Advanced Research Computing
Enabling scientific breakthroughs at scale

Bio Resources (updated twice a year)

https://carc.usc.edu/user-information/bio-resources



The CARC strives to provide a complete and comprehensive user support experience.

CARC OnDemand Service Now Available

New online access point provides users with web access to CARC file systems and clusters

Bio Resources (updated twice a year)

https://carc.usc.edu/user-information/bio-resources

- **Genomes** - reference sequences and annotations for commonly analyzed organisms
- **Genbank** - collection of all public 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)

Bio Resources (through command line)

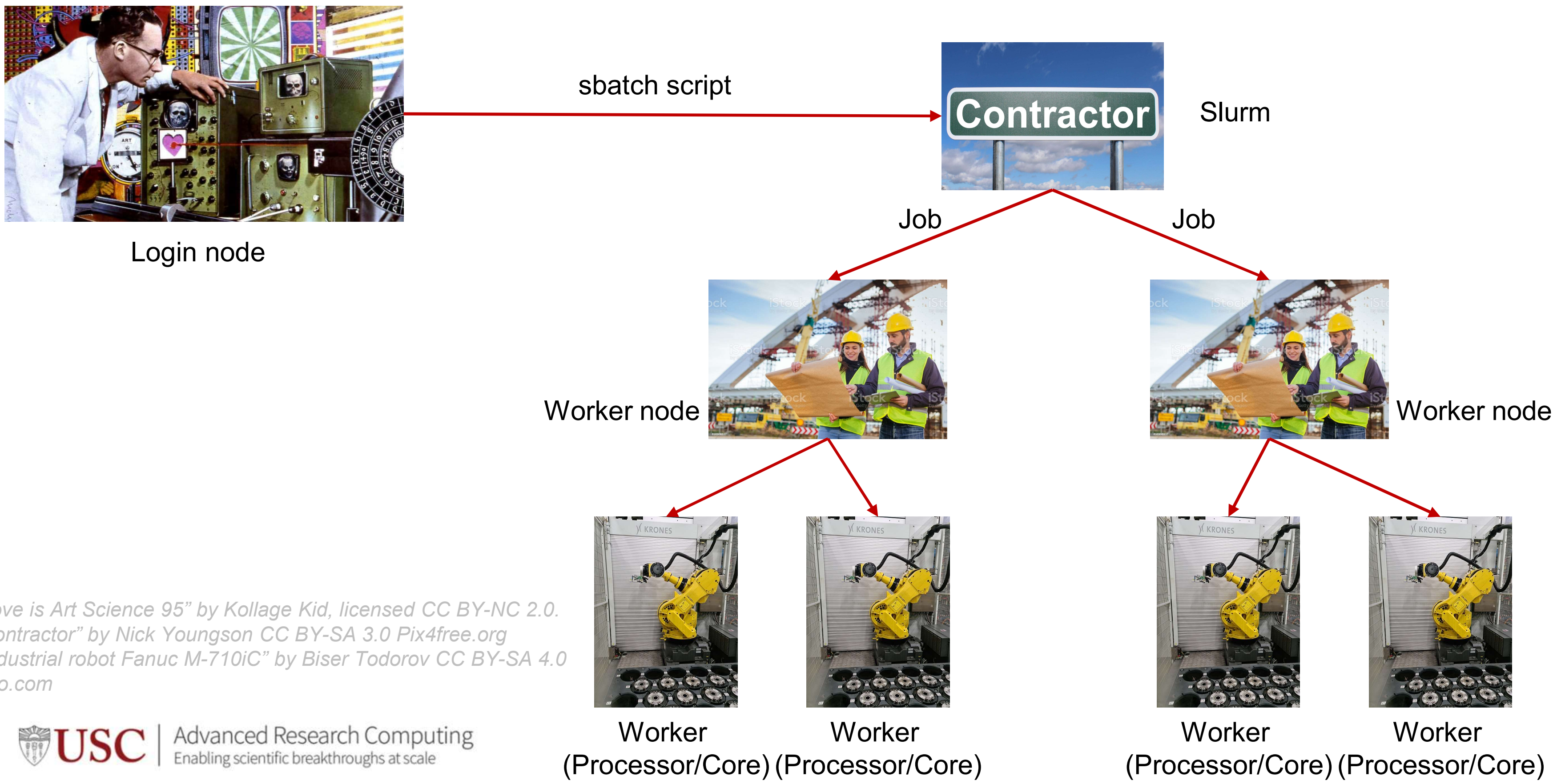
https://carc.usc.edu/user-information/bio-resources

- **Biogeotraces -** set of metagenomes, collected under the auspices of the bioGEOTRACES component of the international GEOTRACES program
- **TaraOceans -** marine microbial metagenomes sampled across space and time
- **Variant Effect Predictor cache -** VEP can use a variety of annotation sources to retrieve the transcript models used to predict consequence types. Cache contains all transcript models, regulatory features and variant data for a species and allows for an offline use of VEP

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
- **Scheduler** – Software that controls when jobs are run and the node they are run on
- **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?



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

- CD is completed, R is running, PD is waiting to run (pending)

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
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: sbatch

- Submit a job to the cluster
- `--partition` partition you want to summit to
  - Default is “main”
- `--nodes` Number of nodes
  - Default is 1
- `--ntasks` Number of CPUs per node
  - Default is 1
- Many more options
  - https://slurm.schedmd.com/sbatch

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

```
[ttrojan@discovery1 ~]$

[ttrojan@discovery1 ~]$ module purge

[ttrojan@discovery1 ~]$ module load gcc/9.2.0

[ttrojan@discovery1 ~]$ module load bowtie2

[ttrojan@discovery1 ~]$ module load bedtools2

[ttrojan@discovery1 ~]$ module load fastqc

[ttrojan@discovery1 ~]$ 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 (Best way to get data from CARC)
  - go to https://www.globus.org in your browser and click **Log In**
  - Search for **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

Review: Interactive Jobs

- When you need to provide unpredictable input

```
[ttrojan@discovery1 ~]$ hostname
discovery1.usc.edu
[ttrojan@discovery1 ~]$ salloc –p debug --ntasks=1 –mem=4g
[ttrojan@a02-26 ~]$ hostname
a02-26.hpc.usc.edu
[ttrojan@a02-26 ~]$ exit
exit
[ttrojan@discovery1 ~]$ hostname
discovery1.usc.edu

[ttrojan@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

Exercise:

- 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"
```

Bash Variables

```
cd raw-seq

i=1

ls -l yeast_${i}_50K.fastq

i=2

ls -l yeast_${i}_50K.fastq
```

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*

Prepare to Run Jobs

- Copy example data to your home directory

```
[ttrojan@discovery1 ~]$

[ttrojan@discovery1 ~]$ git clone https://github.com/uschpc/workshop-bioresources.git

[ttrojan@discovery1 ~]$ cd workshop-bioresources

[ttrojan@discovery1 ~]$ ls
```

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 --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
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

```
[ttrojan@discovery1 ~]$ sbatch fastqc1.job

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 debug fastqc.s osinski  R       0:02      1 a02-26
```

USC | Advanced Research Computing
Enabling scientific breakthroughs at scale

Check Output File for Errors

- Check Output File for Errors

```
[ttrojan@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 or /scratch2 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

Important Things to Note

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_NTASKS` – Number of reserved Processors
  - Assigned automatically by SLURM
  - Value is the multiple of the --ntasks and --nodes 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

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

```
#!/bin/bash
#SBATCH --nodes 1
#SBATCH --ntasks 10
#SBATCH --partition debug
#SBATCH --time 00:05:00
#SBATCH --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
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"
```

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
```

USC | Advanced Research Computing
Enabling scientific breakthroughs at scale

Check BLAST Job Stats with sacct

- sacct can get stats for a job after its completed

https://slurm.schedmd.com/sacct.html

```
[ttrojan@discovery1 ~]$ 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
```

USC | Advanced Research Computing
Enabling scientific breakthroughs at scale

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=ttrojan_001
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…

Without Job Arrays – Numbered Files

```
#!/bin/bash
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --partition main
#SBATCH --time 00:05:00
#SBATCH --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
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 --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
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
- Then create 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 --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
module purge
module load gcc/9.2.0
module load fastqc echo "Starting FastQC job‟
sleep 20
ls raw-seq/ > unnumbered-filenames.txt
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/workshop-bioresources
# ----------------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/workshop-bioresources
# ----------------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 --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
# ----------------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

- Number of processors and no working directory
- Better to use $SLURM_NTASKS

```
#!/bin/bash
# ----------------SLURM Parameters----------------
#SBATCH --partition main
#SBATCH --ntasks 20
#SBATCH --mem=10g
#SBATCH --nodes 1
#SBATCH --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
# ----------------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 --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
# ----------------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 --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
# ----------------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 --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
# ----------------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 --nodes 1
#SBATCH --mem=4g
#SBATCH --ntasks 1
#SBATCH --gres=gpu:p100:1
#SBATCH --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
# ----------------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 --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
# ----------------Load Modules--------------------
module purge
module load gcc/9.2.0
module load samtools
# ----------------Commands------------------------
samtools stats 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 --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
# ----------------Load Modules--------------------
module purge
module load gcc/9.2.0
module load samtools
# ----------------Commands------------------------
samtools stats example.bam
```

Genome mapping and tools: 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

Genome mapping and tools – overview I

- 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/

Genome mapping and tools – overview II

- bowtie
  - The first version of bowtie [Langmead et al. 2009] is optimal for:
    - short reads (under 50 bp)
    - reads without indels (insertions/deletions)
- bowtie2
  - The second version of 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

http://bowtie-bio.sourceforge.net

Genome mapping and tools – overview III

- 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

Genome mapping and tools – overview IV

- samtools package - A set of utilities for processing SAM/BAM files
- samtools 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`
  - Extract all the reads aligned to the range specified. An index of the input file is required

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

- samtools sort

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

- samtools index

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

http://samtools.sourceforge.net

Genome mapping and tools – overview IV

- samtools package - A set of utilities for processing SAM/BAM files
- samtools 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`
  - Extract all the reads aligned to the range specified. An index of the input file is required

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

- samtools sort

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

- samtools index

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

-b: output BAM
-S: read SAM

add a proper header

http://samtools.sourceforge.net

USC | Advanced Research Computing
Enabling scientific breakthroughs at scale

Genome mapping and tools – overview IV

- samtools 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)
```

- samtools faidx – index a FASTA file

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

- samtools merge – merge several BAM files into one

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

Genome mapping and tools – overview V

- 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

Genome mapping and tools – overview V

```
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?

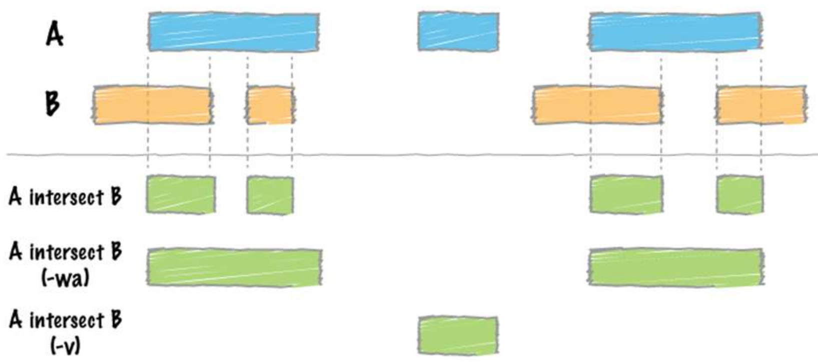
```
bedtools intersect -a cpg.bed -b exons.bed -wo
```

- 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
```



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 our 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
  - find the path on our website in Homo sapiens > UCSC > hg19 > Chromosomes (7th column)
  - https://carc.usc.edu/user-information/bio-resources/reference-genomes
  - Add `chr*.fa` at the end of the path
  - `cp path_above /home1/ttrojan/bio-bootcamp/results/read-mapping`
- 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, convert to BAM (`samtools view`)
- Count the number of insertions and deletions in SAM file (use `cut` for field 6, and `grep`)

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	10000000	20000000
```

- 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 Jendrzejek
Jimi Chu
James K Hong
Chris Taylor
Derek Strong
Cesar Sul
Marco Olguin
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 20
#SBATCH --time 01:00:00
#SBATCH --mem 4g
#SBATCH --account=ttrojan_001
#SBATCH --chdir /home1/ttrojan/workshop-bioresources
module purge
module load gcc/9.2.0
module load bowtie2
module load samtools
module load bedtools2
mkdir results/read-mapping
cp data/R*.gz results/read-mapping
gunzip results/read-mapping/R1.fastq.gz
gunzip results/read-mapping/R2.fastq.gz
cp -v /project/biodb/genomes/Homo_sapiens/UCSC/hg19/Sequence/Chromosomes/chr21.fa results/read-mapping/
bowtie2-build --threads $SLURM_NTASKS results/read-mapping/chr21.fa results/read-mapping/chr21index
bowtie2 --threads $SLURM_NTASKS -x results/read-mapping/chr21index -q results/read-mapping/R1.fastq > results/read-mapping/R1.sam
bowtie2 --threads $SLURM_NTASKS -x results/read-mapping/chr21index -q results/read-mapping/R2.fastq > results/read-mapping/R2.sam
samtools view results/read-mapping/R1.bam | cut -f 6 | grep -c 'D' > results/read-mapping/R1.no_of_deletions.txt
samtools view results/read-mapping/R1.bam | cut -f 6 | grep -c 'I' > results/read-mapping/R1.no_of_insertions.txt
samtools view results/read-mapping/R2.bam | cut -f 6 | grep -c 'D' > results/read-mapping/R2.no_of_deletions.txt
samtools view results/read-mapping/R2.bam | cut -f 6 | grep -c 'I' > results/read-mapping/R2.no_of_insertions.txt
samtools view -bS results/read-mapping/R1.sam > results/read-mapping/R1.bam
samtools view -bS results/read-mapping/R2.sam > results/read-mapping/R2.bam
samtools sort results/read-mapping/R1.bam > results/read-mapping/R1_sorted.bam
samtools sort results/read-mapping/R2.bam > results/read-mapping/R2_sorted.bam
samtools view -h -b results/read-mapping/R1_sorted.bam "chr21:10000000-20000000" > results/read-mapping/R1_sorted_region.bam
samtools view -h -b results/read-mapping/R2_sorted.bam "chr21:10000000-20000000" > results/read-mapping/R2_sorted_region.bam
bamToBed -i results/read-mapping/R1_sorted_region.bam > results/read-mapping/R1_sorted_region.bed
bamToBed -i results/read-mapping/R2_sorted_region.bam > results/read-mapping/R2_sorted_region.bed
bedtools intersect -a results/read-mapping/R1_sorted_region.bed -b results/read-mapping/R2_sorted_region.bed > results/read-mapping/reads.bed
```